Data Mining Exercises

Rok Gomišček, Martin Stražar, Tomaž Curk

February 29, 2020

Contents

Delovno okolje

Namestitev

Načinov namestitve delovnega okolja je več. Izberite tisto, ki vam najbolj ustreza.

```
In [1]: !cat skripte/pip_install.sh
```

```
# !/bin/bash

# Essentials
pip install --upgrade numpy
pip install --upgrade scipy
pip install --upgrade Pillow
pip install --upgrade matplotlib
#pip install --upgrade GPy

# Orange and requirements
pip install --upgrade Orange3

# Scikit-learn
pip install --upgrade sklearn

# iPython notebook and requirements
pip install --upgrade terminado
pip install --upgrade functools32
pip install --upgrade jupyter
pip install --upgrade jupyter_contrib_nbextensions

# Test installations
python -c "import Orange"
python -c "import sklearn"
python -c "import numpy"
python -c "import scipy"
python -c "import matplotlib"
#python -c "import GPy"
python -c "import jupyter"
```

Conda

```
In [2]: !cat skripte/conda_install.sh
```

```
conda install -c conda-forge jupyter_contrib_nbextensions
conda install -c conda-forge jupyter_nbextensions_configurator
jupyter contrib nbextension install --user
jupyter nbextensions_configurator enable --user

jupyter nbextension install https://rawgit.com/jfbercher/latex_envs/master/latex_envs.zip  --user
jupyter nbextension enable latex_envs/latex_envs

pip install jupyter_latex_envs
jupyter nbextension install --py latex_envs
jupyter nbextension enable --py latex_envs

jupyter nbextension install https://rawgit.com/jfbercher/jupyter_nbTranslate/master/nbTranslate.zip --us
jupyter nbextension enable nbTranslate/main
```

Virtualenv

Docker

Git and GitHub

Git is a Version Control System (VCS). It can bed used for both collaboration between developers and for controlling one's own work. Git manages the files in a repository - and keeps track of the change history.

GitHub is a web-based paltform for hosting Git repositories. It enables many developers participating on the same project.

You can read more about Git and Github at https://guides.github.com/

Some basic commands

- `git init`: initializes a new repository in the selected folder
- `git clone`: copy the contents of the remote repository to the disk
- `git add`: add selected modified files to save
  - `git add .`: adds all files
  - `git add file.ipynb`: adds a specified file
- `git commit -m 'message` ': the command to save the selected files
- `git status`: shows which files have been modified and which modified files will be saved
- `git pull`: refresh the local repository with changes to the remote repository
- `git push`: send changes to the remote repository
- `git branch`: shows the local branches, creates them, or changes the active branch
- `git merge`: merges changes between two branches

A complete list of commands and parameters is available at https://git-scm.com/docs.

Installation

When installing Git on Windows, select the option **Enable symbolic links**.

Chapter 1

Data preparation

1.1 Library numpy

The numpy [1] library provides numerical computing in Python. It contains effective implementation of data structures such as vectors, matrices, and arrays. All data structures are derived from the data type array. Most computational operations are implemented in lower-level languages (Fortran, C). We can create an array in different ways:

- by converting Python lists or tuples,
- using the functions arange, linspace, and the like,
- by reading data from files.

```
In [1]: import numpy as np
```

1.1.1 Conversion of lists into multi-dimensional arrays

We use the constructor array directly by submitting a list. If we give a list of numbers, we get a vector:

```
In [2]: v = np.array([1, 2, 3, 4])
        v

Out[2]: array([1, 2, 3, 4])
```

If we give a list of lists, we get a matrix:

```
In [3]: M = np.array([[1, 2], [3, 4]])
        M

Out[3]: array([[1, 2],
               [3, 4]])
```

Regardless of the shape, the objects v and M are of type ndarray.

```
In [4]: type(v), type(M)

Out[4]: (numpy.ndarray, numpy.ndarray)
```

The difference is in their dimensions. The object v is a vector with four elements, and M is a 2 x 2 matrix.

```
In [5]: v.shape

Out[5]: (4,)

In [6]: M.shape
```

```
Out[6]: (2, 2)
```

Similarly, we can display the number of items in the entire list.

```
In [7]: M.size
```

```
Out[7]: 4
```

Question 1-1-1 We can compose arrays of any dimension. Try to create a list of lists (of lists, …) and check out what its dimensions are!

```
In [8]: # Sestavi strukturo poljubnih dimenzij in preveri njeno dimenzijo in velikost
        # X =
```

Answer

1.1.2 Differences between lists and arrays

The structure `numpy.ndarray` still looks like a lis of lists (of lists, …). What's the difference?

Some quick facts:

- Python lists can contain any type of object that can vary within the list (dynamic typing). They do not support mathematical operations such as matrix multiplication. Implementation of such operations would be very inefficient due to dynamic typing.
- Arrays are **statically typed** and **homogeneous**. The data type of elements is determined at the time of creation.
- As a result, arrays are memory-efficient, since they occupy a fixed space in memory.

Determine the type of elements in the current array:

```
In [9]: M.dtype
```

```
Out[9]: dtype('int64')
```

Inserting any type of data into the array can lead to problems. Try:

```
In [10]: M[0,0] = "hello"
```

```
        ---------------------------------------------------------------------------

        ValueError                                Traceback (most recent call last)

        <ipython-input-10-e1f336250f69> in <module>
    ----> 1 M[0,0] = "hello"


        ValueError: invalid literal for int() with base 10: 'hello'
```

Set the data type when creating an array, for example, complex numbers:

```
In [11]: M = np.array([[1, 2, 3], [1, 4, 9]], dtype=complex)

         M
```

```
Out[11]: array([[ 1.+0.j,  2.+0.j,  3.+0.j],
                [ 1.+0.j,  4.+0.j,  9.+0.j]])
```

Let's change the type of elements in the array during execution:

```
In [12]: M = M.astype(float)
         M
```

```
/Users/tomazc/anaconda3/lib/python3.6/site-packages/ipykernel_launcher.py:1: ComplexWarning: Casting co
  """Entry point for launching an IPython kernel.
```

```
Out[12]: array([[ 1.,  2.,  3.],
                [ 1.,  4.,  9.]])
```

We can use data types: `int`, `float`, `complex`, `bool`, `object`.

Sizes, in bits, can be explicitly given: `int64`,`int16`, `float128`,`complex128`.

1.1.3 Using arrays

First, let's take a look at how to use arrays.

1.1.3.1 Addressing

Elements are addressed using square brackets, similar to lists.

```
In [13]: # v je vektor; naslavljamo ga po njegovi edini dimenziji
         v[0]
```

```
Out[13]: 1
```

```
In [14]: # matriko M naslavljamo z dvema podatkoma - naslov je sedaj terka
         M[1,1]
```

```
Out[14]: 4.0
```

Addressing one dimension first returns rows.

```
In [15]: M[1]
```

```
Out[15]: array([ 1.,  4.,  9.])
```

By using `:` we say that we want all elements in the corresponding dimension. How to implement access to the entire first column with lists? You will need some `for` loops. The addressing syntax substantially simplifies this.

```
In [16]: M[1, :] # Vrstica
```

```
Out[16]: array([ 1.,  4.,  9.])
```

```
In [17]: M[:, 1] #  Stolpec, precej enostavno.
```

```
Out[17]: array([ 2.,  4.])
```

Individual elements can be changed with assignment statements.

```
In [18]: M[0, 0] = 9
```

```
In [19]: M
```

```
Out[19]: array([[ 9.,  2.,  3.],
                [ 1.,  4.,  9.]])
```

We can set them by the whole dimension.

```
In [20]: M[1, :] = 0
         M[:, 2] = -1
```

```
In [21]: M
Out[21]: array([[ 9.,  2., -1.],
                [ 0.,  0., -1.]])
```

1.1.3.2 Cutting

Cutting arrays is a common concept. An arbitrary sub-array is obtained by addressing `M[from:to:step]`:

```
In [22]: A = np.array([1, 2, 3, 4, 5])
         A
Out[22]: array([1, 2, 3, 4, 5])
In [23]: A[1:3]
Out[23]: array([2, 3])
```

We can also change the addressed sub-arrays.

```
In [24]: A[1:3] = [-2, -3]
         A
Out[24]: array([ 1, -2, -3,  4,  5])
```

Any of the cutting parameters may also be omitted.

```
In [25]: A[::] # Privzete vrednosti parametrov od:do:korak.
Out[25]: array([ 1, -2, -3,  4,  5])
In [26]: A[::2] # korak velikosti 2
Out[26]: array([ 1, -3,  5])
In [27]: A[:3] # prvi trije elementi
Out[27]: array([ 1, -2, -3])
In [28]: A[3:] # elementi od tretjega naprej
Out[28]: array([4, 5])
```

Negative indices refer to the end of the array:

```
In [29]: A = np.array([1, 2, 3, 4, 5])
In [30]: A[-1]
Out[30]: 5
```

The last three elements:

```
In [31]: A[-3:]
Out[31]: array([3, 4, 5])
```

Cutting also works in multi-dimensional fields.

```
In [32]: A = np.array([[n+m*10 for n in range(5)] for m in range(5)])
         A
Out[32]: array([[ 0,  1,  2,  3,  4],
                [10, 11, 12, 13, 14],
                [20, 21, 22, 23, 24],
                [30, 31, 32, 33, 34],
                [40, 41, 42, 43, 44]])
```

```
In [33]: # pod-polje izvirnega polja A
         A[1:4, 1:4]
```

```
Out[33]: array([[11, 12, 13],
                [21, 22, 23],
                [31, 32, 33]])
```

Elements can be skipped.

```
In [34]: A[::2, ::2]
```

```
Out[34]: array([[ 0,  2,  4],
                [20, 22, 24],
                [40, 42, 44]])
```

1.1.3.3 Addressing arrays using a second structure

Arrays can also be addressed using other arrays or lists.

```
In [35]: row_indices = [1, 2, 3]
         A[row_indices]
```

```
Out[35]: array([[10, 11, 12, 13, 14],
                [20, 21, 22, 23, 24],
                [30, 31, 32, 33, 34]])
```

```
In [36]: col_indices = [1, 2, -1]
         A[row_indices, col_indices]
```

```
Out[36]: array([11, 22, 34])
```

We can also use *masks*. These are structures with `bool` data indicating whether or not the element in the corresponding location will be selected.

```
In [37]: B = np.array([n for n in range(5)])
         B
```

```
Out[37]: array([0, 1, 2, 3, 4])
```

```
In [38]: row_mask = np.array([True, False, True, False, False])
         B[row_mask]
```

```
Out[38]: array([0, 2])
```

A little different way of determining the mask.

```
In [39]: row_mask = np.array([1, 0, 1, 0, 0], dtype=bool)
         B[row_mask]
```

```
Out[39]: array([0, 2])
```

This method can be used to conditionally address elements according to their content.

```
In [40]: x = np.array([0, 4, 2, 2, 3, 7, 10, 12, 15, 28])
         x
```

```
Out[40]: array([ 0,  4,  2,  2,  3,  7, 10, 12, 15, 28])
```

```
In [41]: mask = (5 < x) * (x < 12.3)
         mask
```

```
Out[41]: array([False, False, False, False, False,  True,  True,  True, False, False], dtype=bool)
```

```
In [42]: x[mask]
```

```
Out[42]: array([ 7, 10, 12])
```

Question 1-1-2 Test combinations of all already mentioned addressing methods. Address at the same time, for example, lines with cutting and columns with conditional addressing. Creates more than a two-dimensional structure. Make sure you understand the result of each addressing.

```
In [43]: # Preizkusi več načinov naslavljanja hkrati.
         A[A[:, 0]>10, 0:2 ]
         # ...
         # ...

Out[43]: array([[20, 21],
                [30, 31],
                [40, 41]])
```

Answer

1.1.3.4 Functions for creating arrays

The `numpy` library contains functions for generating common array types. Let's look at some examples.

The `arange` range

```
In [44]: np.arange(0, 10, 1) # od, do, korak

Out[44]: array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])

In [45]: np.arange(-1, 1, 0.1)

Out[45]: array([ -1.00000000e+00,  -9.00000000e-01,  -8.00000000e-01,
                 -7.00000000e-01,  -6.00000000e-01,  -5.00000000e-01,
                 -4.00000000e-01,  -3.00000000e-01,  -2.00000000e-01,
                 -1.00000000e-01,  -2.22044605e-16,   1.00000000e-01,
                  2.00000000e-01,   3.00000000e-01,   4.00000000e-01,
                  5.00000000e-01,   6.00000000e-01,   7.00000000e-01,
                  8.00000000e-01,   9.00000000e-01])
```

Ranges `linspace` and `logspace`

Attention: the start and end points are also included.

```
In [46]: np.linspace(0, 10, 25) # od, do, stevilo med sabo enako oddaljenih tock

Out[46]: array([  0.        ,   0.41666667,   0.83333333,   1.25      ,
                  1.66666667,   2.08333333,   2.5       ,   2.91666667,
                  3.33333333,   3.75      ,   4.16666667,   4.58333333,
                  5.        ,   5.41666667,   5.83333333,   6.25      ,
                  6.66666667,   7.08333333,   7.5       ,   7.91666667,
                  8.33333333,   8.75      ,   9.16666667,   9.58333333,  10.        ])

In [47]: np.logspace(0, 10, 11, base=np.e) # Poskusi z drugo osnovo (bazo): 2, 3, 10

Out[47]: array([  1.00000000e+00,   2.71828183e+00,   7.38905610e+00,
                  2.00855369e+01,   5.45981500e+01,   1.48413159e+02,
                  4.03428793e+02,   1.09663316e+03,   2.98095799e+03,
                  8.10308393e+03,   2.20264658e+04])
```

Random arrays, `numpy.random` module

```
In [48]: from numpy import random
         random.seed(42)  # zagotovi ponovljivost naključnih rezultatov
```

Uniformly distributed values in the interval [0,1]:

```
In [49]: random.rand(5, 5)

Out[49]: array([[ 0.37454012,  0.95071431,  0.73199394,  0.59865848,  0.15601864],
                [ 0.15599452,  0.05808361,  0.86617615,  0.60111501,  0.70807258],
                [ 0.02058449,  0.96990985,  0.83244264,  0.21233911,  0.18182497],
                [ 0.18340451,  0.30424224,  0.52475643,  0.43194502,  0.29122914],
                [ 0.61185289,  0.13949386,  0.29214465,  0.36636184,  0.45606998]])
```

Normally distributed values with mean 0 and variance 1:

```
In [50]: random.randn(5, 5)

Out[50]: array([[-0.62947496,  0.59772047,  2.55948803,  0.39423302,  0.12221917],
                [-0.51543566, -0.60025385,  0.94743982,  0.291034  , -0.63555974],
                [-1.02155219, -0.16175539, -0.5336488 , -0.00552786, -0.22945045],
                [ 0.38934891, -1.26511911,  1.09199226,  2.77831304,  1.19363972],
                [ 0.21863832,  0.88176104, -1.00908534, -1.58329421,  0.77370042]])
```

The diagonal matrix `diag`

The diagonal should contain 1, 2, and 3.

```
In [51]: np.diag([1, 2, 3])

Out[51]: array([[1, 0, 0],
                [0, 2, 0],
                [0, 0, 3]])
```

The diagonal should be removed from the main diagonal for k places. Attention, the dimension of the matrix increases accordingly.

```
In [52]: np.diag([1, 2, 3], k=1)

Out[52]: array([[0, 1, 0, 0],
                [0, 0, 2, 0],
                [0, 0, 0, 3],
                [0, 0, 0, 0]])
```

Zeroes and ones - `zeros`, `ones`

```
In [53]: np.zeros((3, 3))

Out[53]: array([[ 0.,  0.,  0.],
                [ 0.,  0.,  0.],
                [ 0.,  0.,  0.]])

In [54]: np.ones((3, 3))

Out[54]: array([[ 1.,  1.,  1.],
                [ 1.,  1.,  1.],
                [ 1.,  1.,  1.]])
```

1.1.4 Basic computational operations

The key to using interpreted languages is to make the most of the vector operations. Avoid excessive use of loops. As many operations as possible are implemented as operations between matrices and vectors, for example, as vector or matrix multiplication.

1.1.4.1 Array operations with scalar

We use the usual arithmetic operations for multiplication, addition, and division with scalars.

```
In [55]: v1 = np.arange(0, 5)

In [56]: v1 * 2

Out[56]: array([0, 2, 4, 6, 8])

In [57]: v1 + 2

Out[57]: array([2, 3, 4, 5, 6])

In [58]: A * 2, A + 2

Out[58]: (array([[ 0,  2,  4,  6,  8],
                 [20, 22, 24, 26, 28],
                 [40, 42, 44, 46, 48],
                 [60, 62, 64, 66, 68],
                 [80, 82, 84, 86, 88]]), array([[ 2,  3,  4,  5,  6],
                 [12, 13, 14, 15, 16],
                 [22, 23, 24, 25, 26],
                 [32, 33, 34, 35, 36],
                 [42, 43, 44, 45, 46]]))
```

1.1.4.2 Array-array operations (elements-wise)

Operations between multiple fields are by default executed element-wise. For example, element-wise multiplication is achieved using the `*` operator.

```
In [59]: A * A

Out[59]: array([[   0,    1,    4,    9,   16],
                [ 100,  121,  144,  169,  196],
                [ 400,  441,  484,  529,  576],
                [ 900,  961, 1024, 1089, 1156],
                [1600, 1681, 1764, 1849, 1936]])

In [60]: v1 * v1

Out[60]: array([ 0,  1,  4,  9, 16])
```

Attention, array dimensions must match.

```
In [61]: A.shape, v1.shape

Out[61]: ((5, 5), (5,))

In [62]: A * v1

Out[62]: array([[  0,   1,   4,   9,  16],
                [  0,  11,  24,  39,  56],
                [  0,  21,  44,  69,  96],
                [  0,  31,  64,  99, 136],
                [  0,  41,  84, 129, 176]])
```

1.1.5 Iteration through array elements

We try to stick to the principle of avoiding using loops over the array elements. The reason is the slow implementation of loops in interpreted languages, such as Python. Sometimes, however, we can not avoid loops. Loop `for` is a meaningful solution.

```
In [63]: v = np.array([1,2,3,4])

         for element in v:
             print(element)

1
2
3
4


In [64]: M = np.array([[1,2], [3,4]])

         for row in M:
             print("row", row)

             for element in row:
                 print(element)

row [1 2]
1
2
row [3 4]
3
4
```

The `enumerate` generator is used when we want to iterate through elements and possibly change their values.

```
In [65]: for i, row in enumerate(M):
             print("row index", i, "row", row)

             for j, element in enumerate(row):
                 print("col index", j, "element", element)

                 # Kvadriramo vsakega od elementov
                 M[i, j] = element ** 2

row index 0 row [1 2]
col index 0 element 1
col index 1 element 2
row index 1 row [3 4]
col index 0 element 3
col index 1 element 4
```

We get an array where each element is a square of the original value.

```
In [66]: M

Out[66]: array([[ 1,  4],
                [ 9, 16]])
```

Learn more about the numpy library in [1, 2, 3, 4].

1.2 Example: temperature statistics in Stockholm

We will use the `numpy` library on the case of daytime temperature data in Stockholm. Data includes metrics for each day between 1800 and 2011. They are stored in a file where the lines represent measurements. Individual data - year, month, day and measured temperature - are separated by comma.

```
In [1]: from csv import DictReader

        fp = open('podatki/stockholm.csv', 'rt')
        reader = DictReader(fp)

        for row in reader:
            print(row)
            break  # izpisi samo prvo vrstico

OrderedDict([('Year', '1800'), ('Month', '1'), ('Day', '1'), ('Temp', '-6.1')])
```

Presenting data in the form of a dictionary is useful for its clarity, but the calculation will be much faster, if we load the data as an array.

```
In [2]: import numpy as np
        np.set_printoptions(suppress=True)

        data = np.loadtxt('podatki/stockholm.csv', delimiter=",", skiprows=1)
        data

Out[2]: array([[ 1800. ,     1. ,     1. ,    -6.1],
               [ 1800. ,     1. ,     2. ,   -15.4],
               [ 1800. ,     1. ,     3. ,   -15. ],
               …,
               [ 2011. ,    12. ,    29. ,     4.9],
               [ 2011. ,    12. ,    30. ,     0.6],
               [ 2011. ,    12. ,    31. ,    -2.6]])
```

Check the data size: the number of lines (*measurements*, *samples*) and the number of columns (*attributes*).

```
In [3]: data.shape

Out[3]: (77431, 4)
```

Columns store data in this order: `year`, `month`, `day` and `temperature`.

Let's take a look at all the measurements made in 2011. We create the binary vector `data [:, 0] == 2011`, which contains the `True` value on the relevant positions and is used to address the data.

```
In [4]: data[data[:, 0] == 2011]

Out[4]: array([[ 2011. ,     1. ,     1. ,    -2.3],
               [ 2011. ,     1. ,     2. ,    -3.6],
               [ 2011. ,     1. ,     3. ,    -6.9],
               …,
               [ 2011. ,    12. ,    29. ,     4.9],
               [ 2011. ,    12. ,    30. ,     0.6],
               [ 2011. ,    12. ,    31. ,    -2.6]])
```

Question 1-2-1 Print out the temperature 200 years ago, for example, the temperature on December 5, 1817.

```
In [5]:
```

Answer

1.2.1 Data Processing

Let's introduce operations that tell us something about the data. We will calculate some basic statistics.

1.2.1.1 Average, arithmetic mean

Daily temperature is in column with index 3 (fourth column). Calculate the average of all measurements.

```
In [5]: np.mean(data[:, 3])
```

```
Out[5]: 6.1971096847515854
```

We find that the average daily temperature in Stockholm over the past 200 years was pleasant 6.2 C.

Question 1-2-2 What is the average temperature in January (month with the number '1')?

```
In [6]:
```

Answer

1.2.1.2 Standard deviation and variance

```
In [6]: np.std(data[:,3]), np.var(data[:,3])
```

```
Out[6]: (8.2822716213405734, 68.596023209663414)
```

Question 1-2-3 In what month is the temperature deviation the biggest?

```
In [7]: # Poišči mesec z največjim odklonom oz. varianco v temperaturi.
        # Namig: zgradi seznam terk oblike (odklon v temperaturi, mesec)
        # ...
```

Answer

1.2.1.3 Minimum and maximum value

Let's find the lowest daily temperature:

```
In [8]: data[:,3].min()
```

```
Out[8]: -25.800000000000001
```

Let's find the highest daily temperature:

```
In [9]: data[:,3].max()
```

```
Out[9]: 28.300000000000001
```

Question 1-2-4 The month and year when the maximum temperature was recorded.

```
In [10]: # Poišči mesec in leto, kjer smo v povprečju beležili najvišjo temperaturo.
         # Namig: zgradi seznam terk oblike (povprecna temperatura, (leto, mesec))
         # ...
```

Answer

1.2.1.4 Sum, product

Temperature is usually not summed up. Nevertheless, take the opportunity to see the functions of the sum and the product.

```
In [11]: data[:, 3].sum()  # vsota vseh temperatur

Out[11]: 479848.40000000002

In [12]: data[:, 3].sum() / data.shape[0]   # dobimo ravno aritmetično sredino

Out[12]: 6.1971096847515854

In [13]: # prva vrstica v podatkih ...
         data[0, :]

Out[13]: array([ 1800. ,     1. ,     1. ,    -6.1])

In [14]: # ... in njen produkt
         np.prod(data[0, :])

Out[14]: -10980.0
```

1.2.2 Global warming?

Let's answer a few more questions. According to Stockholm, the rumors circulate that the temperature is increasing from year to year.

```
In [15]: # Izračunajmo povprečno temperaturo za vsako leto posebej
         letna_povprečja = dict()

         for leto in range(1800, 2012):
             # Uporabimo pogojno naslavljanje polja
             letna_povprečja[leto] = data[data[:, 0] == leto, 3].mean()
```

Question 1-2-5 Write years when the average temperature is higher than last year.

```
In [16]: # Izpiši vsako leto, ki ima večjo povprečno temperaturo od prejšnjega
         #
```

Find the 10 warmest years.

```
In [17]: # Poišči 10 najtoplejših let
         #
```

Answer

The last years are really suspiciously warm. Try to display data using the matplotlib library.

```
In [18]: %matplotlib inline
         %config InlineBackend.figure_formats = ['jpg']
         import matplotlib
```

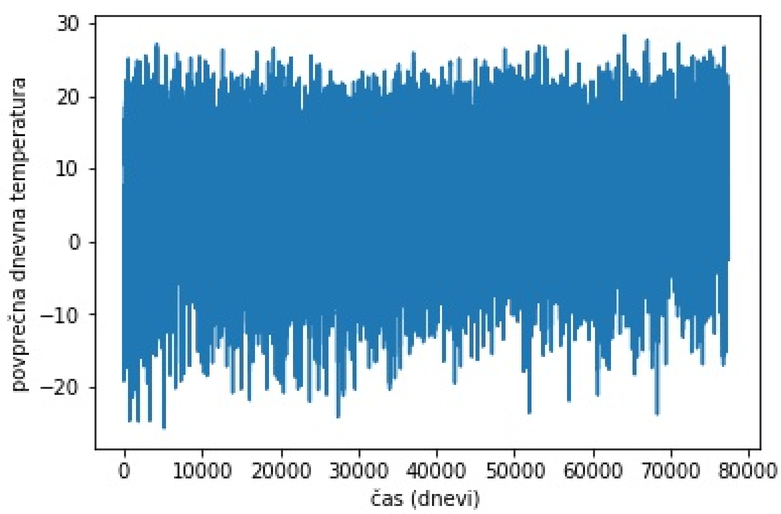
```
matplotlib.figure.Figure.__repr__ = lambda self: (
    f"<{self.__class__.__name__} size {self.bbox.size[0]:g}"
    f"x{self.bbox.size[1]:g} with {len(self.axes)} Axes>")
import matplotlib.pyplot as plt
plt.style.use('PR.mplstyle')
```

```
    ---------------------------------------------------------------------------

    FileNotFoundError                         Traceback (most recent call last)

    ~/anaconda3/lib/python3.6/site-packages/matplotlib/style/core.py in use(style)
    112             try:
--> 113                 rc = rc_params_from_file(style, use_default_template=False)
    114                 _apply_style(rc)


    ~/anaconda3/lib/python3.6/site-packages/matplotlib/__init__.py in rc_params_from_file(fname, fa
   1028     """
-> 1029     config_from_file = _rc_params_in_file(fname, fail_on_error)
   1030


    ~/anaconda3/lib/python3.6/site-packages/matplotlib/__init__.py in _rc_params_in_file(fname, fai
    944     rc_temp = {}
--> 945     with _open_file_or_url(fname) as fd:
    946         try:


    ~/anaconda3/lib/python3.6/contextlib.py in __enter__(self)
     80         try:
---> 81             return next(self.gen)
     82         except StopIteration:


    ~/anaconda3/lib/python3.6/site-packages/matplotlib/__init__.py in _open_file_or_url(fname)
    929             encoding = "utf-8"
--> 930         with io.open(fname, encoding=encoding) as f:
    931             yield f


    FileNotFoundError: [Errno 2] No such file or directory: 'PR.mplstyle'


During handling of the above exception, another exception occurred:


    OSError                                   Traceback (most recent call last)

    <ipython-input-18-78ed3b6a1a07> in <module>
      6     f"x{self.bbox.size[1]:g} with {len(self.axes)} Axes>")
      7 import matplotlib.pyplot as plt
----> 8 plt.style.use('PR.mplstyle')
```

```
~/anaconda3/lib/python3.6/site-packages/matplotlib/style/core.py in use(style)
    117                             "not a valid URL or path. See `style.available` for "
    118                             "list of available styles.")
--> 119                     raise IOError(msg % style)
    120 
    121 


OSError: 'PR.mplstyle' not found in the style library and input is not a valid URL or path. See
```

Let's make a new image (*figure*) and plot the average temperatures against time.

```
In [19]: plt.figure()
         plt.plot(data[:, 3])
         plt.xlabel("čas (dnevi)")  # Vedno označimo osi.
         plt.ylabel("povprečna dnevna temperatura");
```

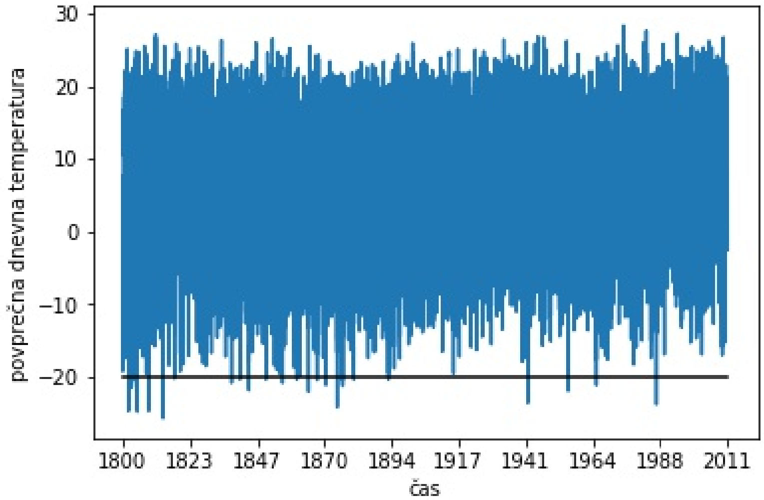


Quite opaque. Try expanding the image by changing `plt.figure (figsize = (width, height))`, where `height 'and` width' are given in inches or inch (default `(5, 3)`).

However, we observe that the frequency of days with a temperature lower than -20.0 C is decreasing. Let's see.

```
In [20]: plt.figure()

         # Narišimo izvirne podatke
         plt.plot(data[:, 3])

         # Z vodoravno črto označimo -20.0 C.
         plt.plot([0, len(data)], [-20, -20], color="black")

         # Spremenimo še oznako x-osi. Dodajmo 10 enako oddaljenih kazalcev.
         ticks = np.arange(0, len(data), len(data)//9, dtype=int)
         plt.xticks(ticks)
         plt.gca().set_xticklabels(data[ticks, 0].astype(int))

         # Vedno označimo osi.
```

```
plt.xlabel("čas")
plt.ylabel("povprečna dnevna temperatura")
plt.show()
```



From the 80s of the last century, we really did not have any particular cold days. However, we would like to further simplify the display. Let's show each year with one point, which should show the average temperature of the year.

Question 1-2-6 Draw a picture of the average annual temperature. Use the `plt.plot (x, y)` function where `x` is the vector of years, andy is the vector of the corresponding average temperatures. Do you think the temperature really grows over the years?

```
In [21]: # Pomagaj si s letna_povprečja.
         # Os x: leto
         # Os y: povprečna letna temperatura
         # ...
```

Answer

Chapter 2

Plotting data

2.1 Library matplotlib

Matplotlib is a library for 2D and 3D drawing in the Python programming language. It includes:

- Controlling individual image elements.
- Export results in the form of PNG, PDF, SVG, EPS, and PGF.
- Support LaTeXsyntax

An essential part of the library's usefulness is that images can be built entirely using commands, which eliminates the need for manual editing. The latter works very well for use in scientific work, where we can generate complex visualizations on various data without the need to change the program code.

The library's website is also a rich source of additional examples: http://matplotlib.org/

```
In [1]: %matplotlib inline
        import matplotlib
        %config InlineBackend.figure_format = 'jpg'
        matplotlib.figure.Figure.__repr__ = lambda self: (
            f"<{self.__class__.__name__} size {self.bbox.size[0]:g}"
            f"x{self.bbox.size[1]:g} with {len(self.axes)} Axes>")
        import matplotlib.pyplot as plt
        plt.style.use('PR.mplstyle')

        matplotlib.__version__
```

```
        ---------------------------------------------------------------------------

        FileNotFoundError                         Traceback (most recent call last)

        ~/anaconda3/lib/python3.6/site-packages/matplotlib/style/core.py in use(style)
        112             try:
    --> 113                 rc = rc_params_from_file(style, use_default_template=False)
        114                 _apply_style(rc)


        ~/anaconda3/lib/python3.6/site-packages/matplotlib/__init__.py in rc_params_from_file(fname, fa
       1028     """
    -> 1029     config_from_file = _rc_params_in_file(fname, fail_on_error)
```

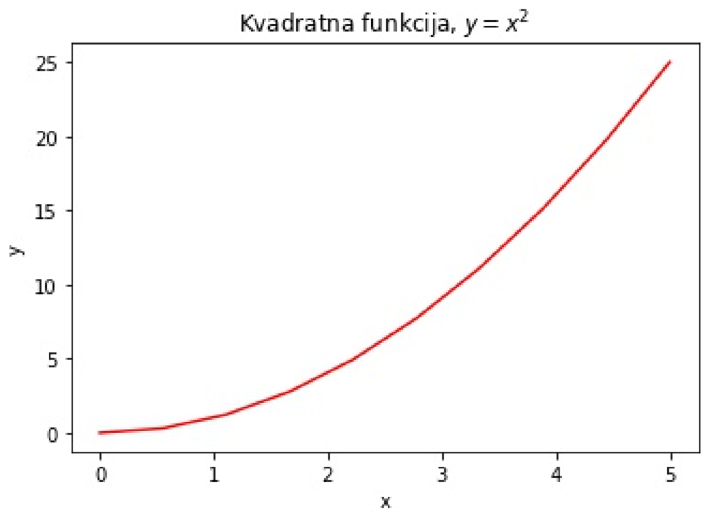
```
   1030


   ~/anaconda3/lib/python3.6/site-packages/matplotlib/__init__.py in _rc_params_in_file(fname, fai
   944     rc_temp = {}
--> 945     with _open_file_or_url(fname) as fd:
   946         try:


   ~/anaconda3/lib/python3.6/contextlib.py in __enter__(self)
    80         try:
---> 81             return next(self.gen)
    82         except StopIteration:


   ~/anaconda3/lib/python3.6/site-packages/matplotlib/__init__.py in _open_file_or_url(fname)
   929             encoding = "utf-8"
--> 930         with io.open(fname, encoding=encoding) as f:
   931             yield f


   FileNotFoundError: [Errno 2] No such file or directory: 'PR.mplstyle'


During handling of the above exception, another exception occurred:


   OSError                                   Traceback (most recent call last)

   <ipython-input-1-6de4f0886ee3> in <module>
     6     f"x{self.bbox.size[1]:g} with {len(self.axes)} Axes>")
     7 import matplotlib.pyplot as plt
----> 8 plt.style.use('PR.mplstyle')
     9
    10 matplotlib.__version__


   ~/anaconda3/lib/python3.6/site-packages/matplotlib/style/core.py in use(style)
   117                         "not a valid URL or path. See `style.available` for "
   118                         "list of available styles.")
--> 119                 raise IOError(msg % style)
   120
   121


   OSError: 'PR.mplstyle' not found in the style library and input is not a valid URL or path. See
```

Simple picture in the `matplotlib` environment:

```
In [2]: import numpy as np
        np.random.seed(42)

        x = np.linspace(0, 5, 10)
        y = x ** 2
```

The plot function accepts parameters: * data on ordinate, * data on abscissa, * other parameters (formatting, ...)

In [3]:
```
plt.figure()
plt.plot(x, y, 'r')
plt.xlabel('x')
plt.ylabel('y')
plt.title('Kvadratna funkcija, $y=x^2$');
```



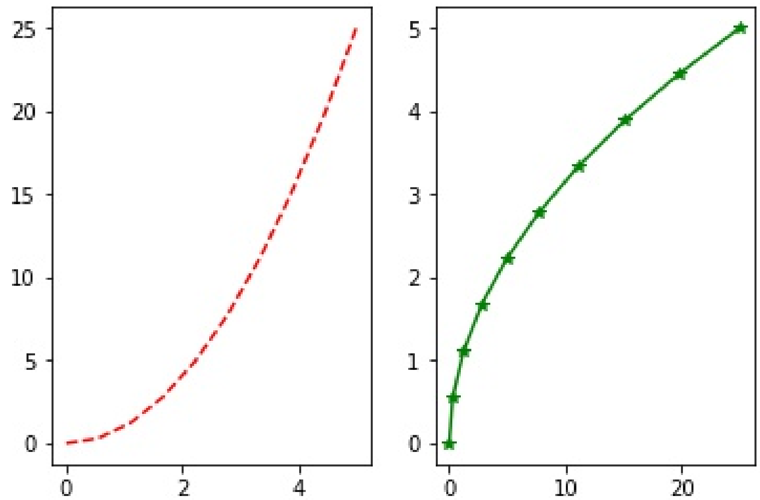
Question 2-1-1 Draw functions x^{-3}, x^{-2}, x^{-1}, x^0, x^1, x^2, x^3 on the interval $(0, 5]$. This is accomplished by repeatedly calling the plot function, once for each curve.

In [4]:
```
# Nariši funkcije na isti graf
# ...
```

Answer

Using the subplot environment, we can create an image with multiple figures.

In [5]:
```
plt.subplot(1, 2, 1)     # ustvari sliko z 1 x 2 platnoma in izberi prvo platno
plt.plot(x, y, 'r--')    # nariši podatke
plt.subplot(1, 2, 2)     # na isti sliki izberi drugo platno
plt.plot(y, x, 'g*-');   # ...
```



2.1.1 An object-oriented way of working with `matplotlib`

In the above examples, we used an interface where each image was part of the *global* environment. This mode is useful for simpler images. The object-oriented way provides advanced visualizations, especially in cases where we deal with more than one image at a time.

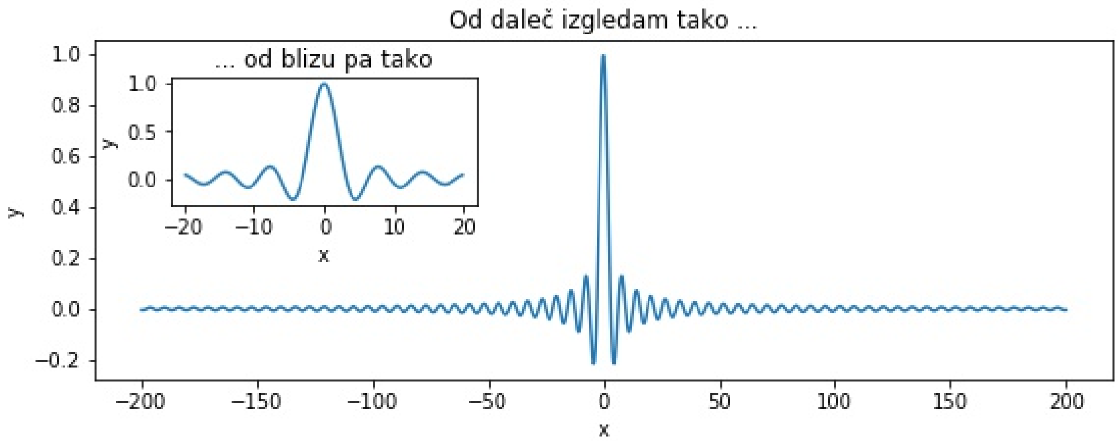
Two essential parts of an object-oriented environment are the objects **figure** and **axis**. One image contains one or more axes. Axis is a container with a coordinate system in which we draw objects (lines, columns, shapes, ...).

Create a new image in the `fig` variable and add it to the new axis, accessed via the `axes` variable.

```
In [6]: fig = plt.figure()

        # poskušaj spreminjati parametre
        axes = fig.add_axes([0.1, 0.1, 0.8, 0.8]) # x koordinata osi relativno na sliko,
                                                   # y koordinata osi relativno na sliko,
                                                   # širina,
                                                   # višina

        axes.plot(x, y, 'r')

        axes.set_xlabel('x')
        axes.set_ylabel('y')
        axes.set_title('Ponovno kvadratna funkcija');
```

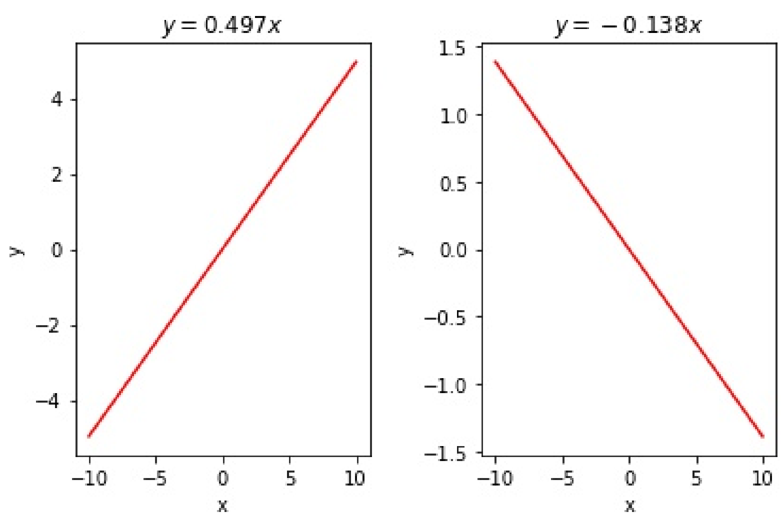
Now we have complete control over the insertion of the axis. You can add an arbitrary number of axes to the picture, which may also overlap.

```
In [7]: fig = plt.figure(figsize=(9, 3))

        x = np.linspace(-200, 200, 1000)
        y = np.sin(x)/x

        axes1 = fig.add_axes([0.1, 0.1, 0.8, 0.8])  # glavna os
        axes2 = fig.add_axes([0.16, 0.51, 0.24, 0.3], facecolor='white') # os znotraj glavne osi.
        # pomemben je tudi vrstni red ustvarjanja!

        # Prikazi večji interval
        axes1.plot(x, y)
        axes1.set_xlabel('x')
        axes1.set_ylabel('y')
        axes1.set_title('Od daleč izgledam tako ...')

        # Prikaži okolico ničle
        axes2.plot(x[450:550], y[450:550])
        axes2.set_xlabel('x')
        axes2.set_ylabel('y')
        axes2.set_title('... od blizu pa tako');
```



For the axes arranged in a nicely arranged network, we can use the `subplots` manager.

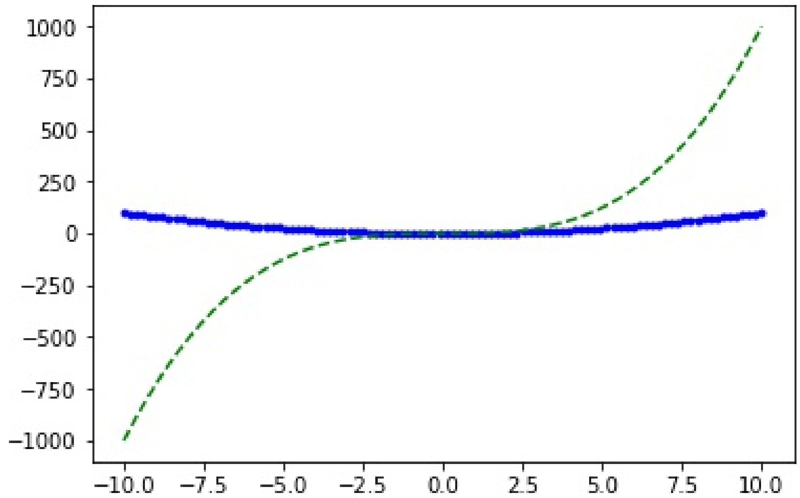
```
In [8]: fig, axes = plt.subplots(nrows=1, ncols=2)

        x = np.linspace(-10, 10, 100)

        for ax in axes:                           # sprehodimo se po oseh
            k = np.random.randn(1, 1)[0]    # narišimo naključno premico
            y = k * x

            ax.plot(x, y, 'r')
            ax.set_xlabel('x')
            ax.set_ylabel('y')
            ax.set_title('$y = %.3f x$'% k )

        fig.tight_layout()    # poskrbi da se osi ne prekrivajo
```



2.1.2 Coloring

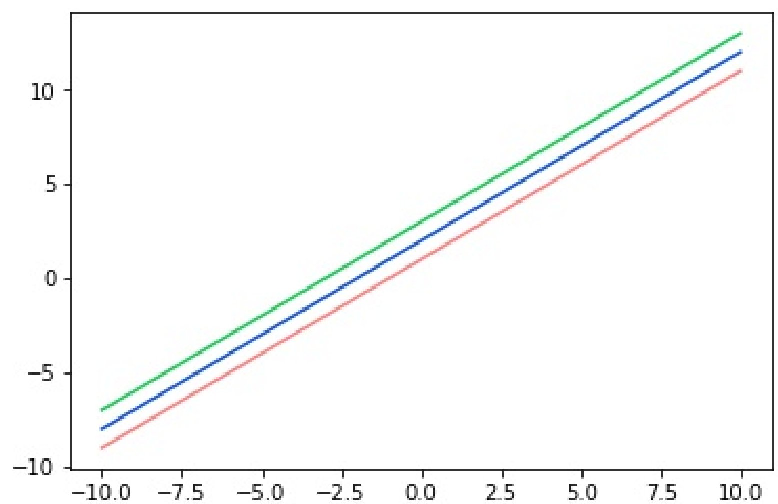
The simplest way to adjust the colors is a style that is similar to the MATLAB environment; `g` represents the green color, `b` blue, `r` red, etc.

```
In [9]: plt.figure()
        plt.plot(x, x**2, 'b.-') # modra črta s označenimi točkami
        plt.plot(x, x**3, 'g--'); # green dashed line
```



Alternatively, use the `color = ...` argument where the color is given by its name or RGB code.

```
In [10]: plt.figure()
         plt.plot(x, x+1, color="red", alpha=0.5) # Parameter alpha določa transparentnost; preizkusi!
         plt.plot(x, x+2, color="#1155dd")
         plt.plot(x, x+3, color="#15cc55");
```

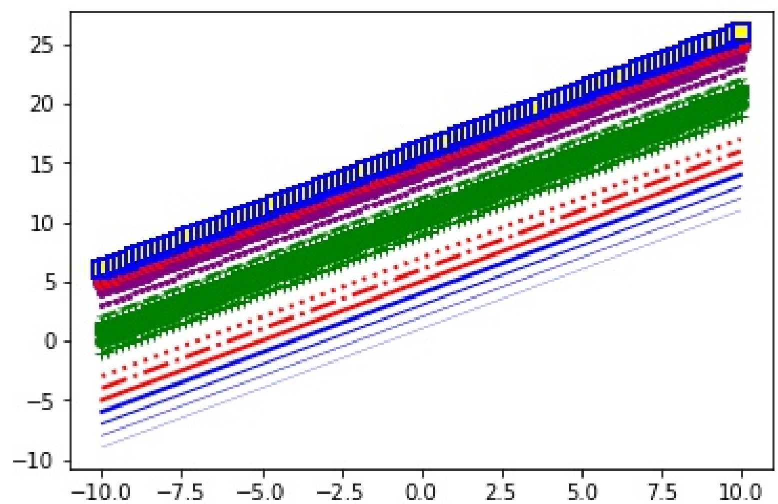


2.1.3 Styles

Let's try to change the other properties: the thickness of the lines and different point markings.

```
In [11]: fig, ax = plt.subplots()

         ax.plot(x, x+1, color="blue", linewidth=0.25)
         ax.plot(x, x+2, color="blue", linewidth=0.50)
         ax.plot(x, x+3, color="blue", linewidth=1.00)
```

```
ax.plot(x, x+4, color="blue", linewidth=2.00)

# možnosti za izgled črte so ‘-‘, ‘--’, ‘-.’, ‘:’, ‘steps’
ax.plot(x, x+5, color="red", lw=2, linestyle='-')
ax.plot(x, x+6, color="red", lw=2, ls='-.')
ax.plot(x, x+7, color="red", lw=2, ls=':')


# oznake za točke: marker = '+', 'o', '*', 's', ',', '.', '1', '2', '3', '4', ...
ax.plot(x, x+ 9, color="green", lw=2, ls='--', marker='+')
ax.plot(x, x+10, color="green", lw=2, ls='--', marker='o')
ax.plot(x, x+11, color="green", lw=2, ls='--', marker='s')
ax.plot(x, x+12, color="green", lw=2, ls='--', marker='1')

# velikost in barva označb
ax.plot(x, x+13, color="purple", lw=1, ls='-', marker='o', markersize=2)
ax.plot(x, x+14, color="purple", lw=1, ls='-', marker='o', markersize=4)
ax.plot(x, x+15, color="purple", lw=1, ls='-', marker='o', markersize=8,
        markerfacecolor="red")
ax.plot(x, x+16, color="purple", lw=1, ls='-', marker='s', markersize=8,
        markerfacecolor="yellow", markeredgewidth=2, markeredgecolor="blue");
```



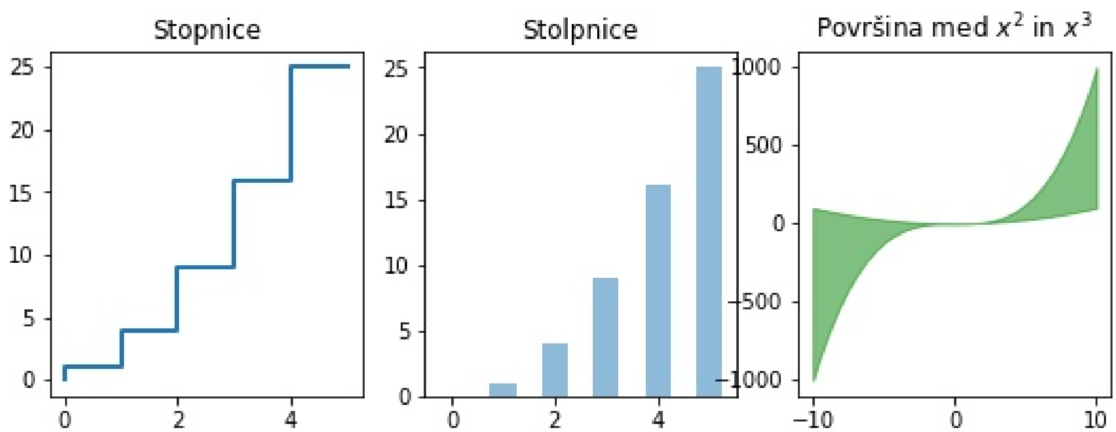
2.1.4 Visualization of different types of data

Let’s take a look at other methods that are suitable for drawing different types of data. Of course, the mode of display depends on the type and characteristics of the data that we want to emphasize with visualization.

```
In [12]: n = np.array([0,1,2,3,4,5])
```

```
In [13]: fig, axes = plt.subplots(1, 3, figsize=(9, 3))

         # Stopnice
         axes[0].step(n, n**2, lw=2)
         axes[0].set_title("Stopnice")

         # Stolpični diagram
         axes[1].bar(n, n**2, align="center", width=0.5, alpha=0.5)
```

```
axes[1].set_title("Stolpnice")

# Površina med krivuljama kvadratne in kubične funkcije
axes[2].fill_between(x, x**2, x**3, color="green", alpha=0.5);
axes[2].set_title("Površina med $x^2$ in $x^3$");
```



2.1.5 Probability distributions

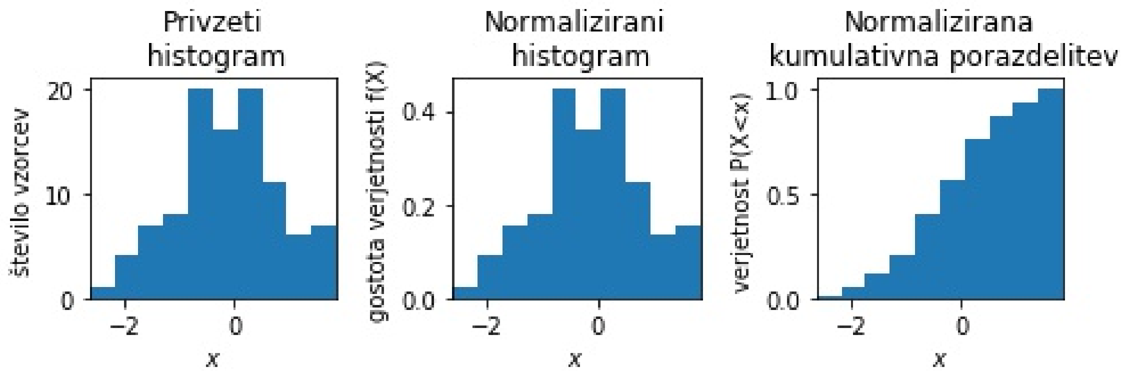
The probability of distributing a finate number of samples is often represented by the histogram - a column diagram representing the number or probability of the value of the variable.

Let x be a random variable distributed over a normal (Gaussian) distribution with the mean $\mu = 0$ and the standard deviation of $\sigma = 1$.

We take N random samples of the variable x. The `hist` function displays the bar graph of the probability distribution with respect to the results of the sampling.

In [14]:
```
# Histogram verjetnostne porazdelitve števil
N = 100
data = np.random.randn(N) # vzorčimo N točk
fig, axes = plt.subplots(1, 3, figsize=(7, 2.5))

axes[0].hist(data, bins=10)
axes[0].set_title("Privzeti\n histogram")
axes[0].set_xlim((min(data), max(data)));
axes[0].set_ylabel("število vzorcev")
axes[0].set_xlabel("$x$")

axes[1].hist(data,  density=True, bins=10)
axes[1].set_title("Normalizirani\n histogram")
axes[1].set_xlim((min(data), max(data)));
axes[1].set_ylabel("gostota verjetnosti f(X)")
axes[1].set_xlabel("$x$")

axes[2].hist(data, cumulative=True, bins=10, density=True)
axes[2].set_title("Normalizirana\n kumulativna porazdelitev")
axes[2].set_ylabel("verjetnost P(X<x)")
axes[2].set_xlim((min(data), max(data)));
axes[2].set_xlabel("$x$")
```

```
fig.tight_layout()
```



Question 2-1-2 Try to change the number of samples `N` and the number of `bins`. Are any settings more appropriate than others depending on the number of samples?

In [15]:

Answer

Question 2-1-3 The `randn` function assumes the center of $\mu = 0$ and the standard deviation $\sigma = 1$. How to model an arbitrary center and standard deviation, e.g. $\mu = 5$ and $\sigma = 0.5$?

In [15]:

Answer

2.1.6 Heat maps and contours

Heat maps are used to display the functions of two variables. We draw the function of two variables:

$$z = sin(x)cos(y)$$

In [15]:
```
def func(x, y):
    return np.sin(x) * np.cos(y)
```

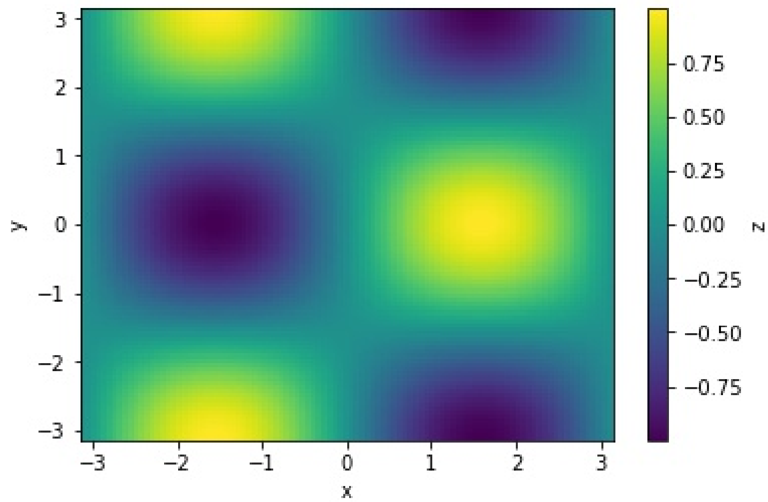
In [16]:
```
x = np.linspace(-np.pi, np.pi, 100)
y = np.linspace(-np.pi, np.pi, 100)
X,Y = np.meshgrid(x, y)
Z = func(X, Y)
```

In `matplotlib` we can choose between several options.

2.1.6.1 The `pcolor` function

In [17]:
```
fig, ax = plt.subplots()

p = ax.pcolor(X, Y, Z,)
cb = fig.colorbar(p, ax=ax, label="z")
ax.set_xlabel("x")
ax.set_ylabel("y")
```

```
ax.set_xlim(-np.pi, np.pi)
ax.set_ylim(-np.pi, np.pi);
```

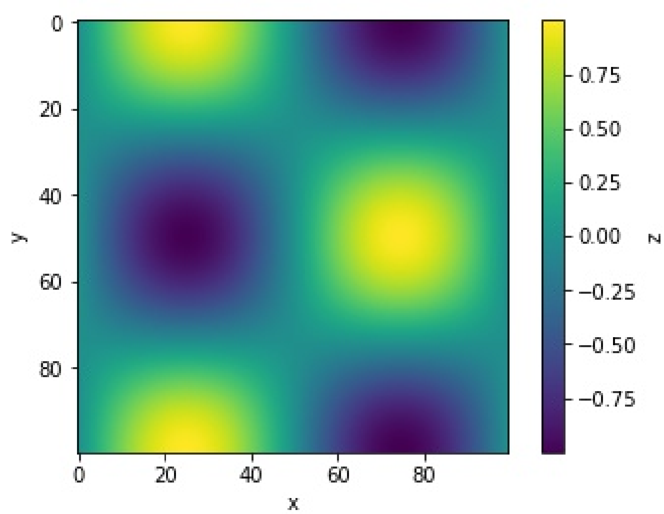


The blue values are embedded in the screen, while the yellow ones are convex.

2.1.6.2 The imshow function

We get a cleaner image by using an interpolation algorithm.

```
In [18]: fig, ax = plt.subplots()

         im = ax.imshow(Z)
         im.set_interpolation('bilinear')
         cb = fig.colorbar(im, ax=ax, label="z")
         ax.set_xlabel("x")
         ax.set_ylabel("y");
```



2.1.6.3 The contour function

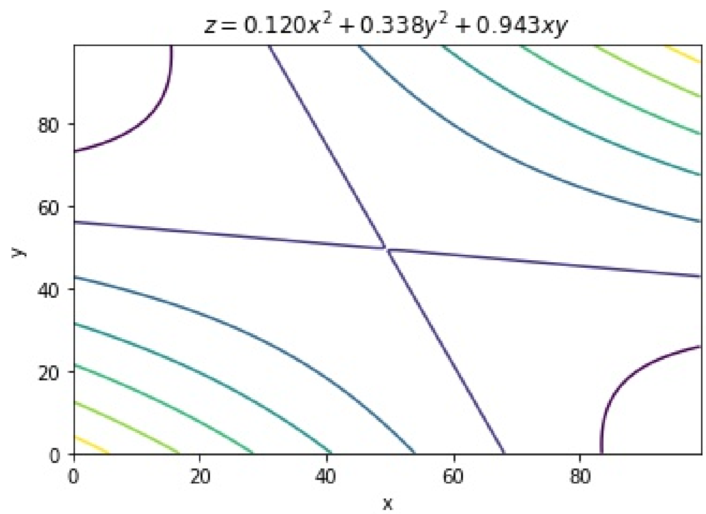
Contours are used to display points with the same value of the function - as with isohypses, we associate points with the same height on the map.

Draw a random square function

$$z = a_1x^2 + a_2y^2 + a_3xy$$

where the coefficients of a_1, a_2, a_3 are determined randomly.

```
In [19]: def random_square_function_2D(x, y, a):
             return a[0] * x**2 + a[1] * y**2 + a[2] * x * y

         a = np.random.rand(3, 1)
         x1 = np.linspace(-np.pi, np.pi, 100)
         y1 = np.linspace(-np.pi, np.pi, 100)
         X,Y = np.meshgrid(x1, y1)
         Z = random_square_function_2D(X, Y, a)

         fig, ax = plt.subplots()
         im = ax.contour(Z)
         ax.set_title("$z = %.3f x^2 + %.3f y^2 + %.3f x y$" % (a[0], a[1], a[2]))
         ax.set_xlabel("x")
         ax.set_ylabel("y");
```



2.1.7 Control over the axis size

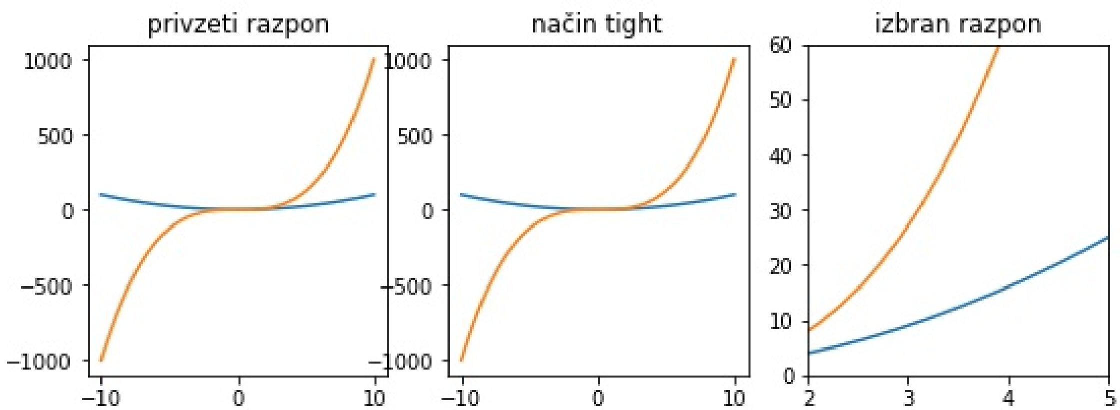
In this section, we will change the size of the image and set the range of the data to be displayed.

2.1.7.1 Range

For better image clarity, we limit it only to the data domain: manually using `set_ylim` and `set_xlim` or automatically with `axis('tight')`.

```
In [20]: x = np.linspace(-10, 10, 100)

         fig, axes = plt.subplots(1, 3, figsize=(9, 3))
```

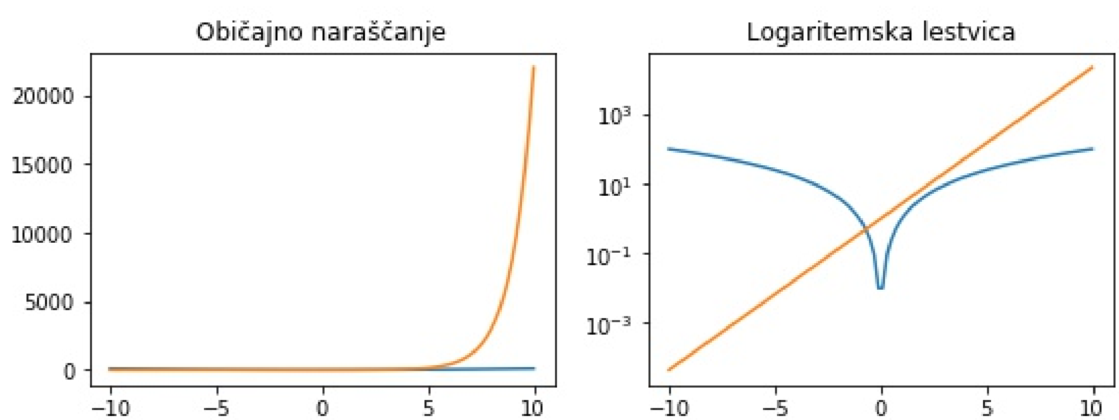
```
        axes[0].plot(x, x**2, x, x**3)
        axes[0].set_title('privzeti razpon')

        axes[1].plot(x, x**2, x, x**3)
        axes[1].axis('tight')
        axes[1].set_title('način tight')

        axes[2].plot(x, x**2, x, x**3)
        axes[2].set_ylim([0, 60])
        axes[2].set_xlim([2, 5])
        axes[2].set_title('izbran razpon');
```



2.1.7.2 Logarithmic scale

We can simply set the logarithmic increment of intervals on individual axes.

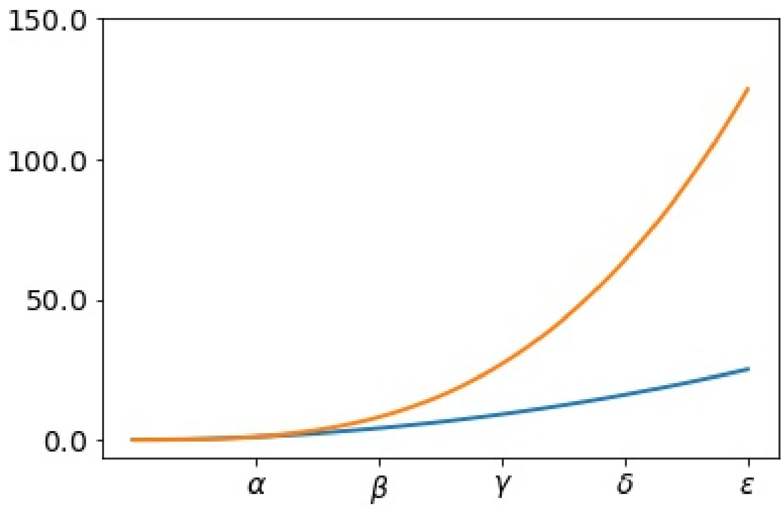
```
In [21]: fig, axes = plt.subplots(1, 2, figsize=(9, 3))

        axes[0].plot(x, x**2, x, np.exp(x))
        axes[0].set_title("Običajno naraščanje")

        axes[1].plot(x, x**2, x, np.exp(x))
        axes[1].set_yscale("log")
        axes[1].set_title("Logaritemska lestvica");
```



2.1.8 Setting the marks on the axes

Using the `set_xticks` and `set_yticks` methods, we set locations of tags, then set the tags explicitly with `set_xticklabels` and `set_yticklabels`.

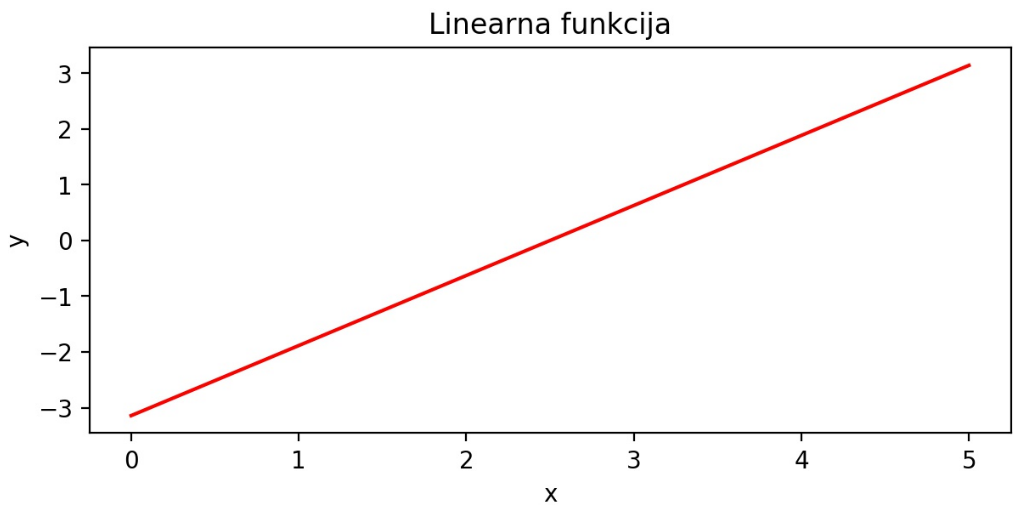
```
In [22]: fig, ax = plt.subplots()
         x = np.linspace(0, 5, 100)
         ax.plot(x, x**2, x, x**3, lw=2)

         ax.set_xticks([1, 2, 3, 4, 5])
         ax.set_xticklabels(
             [r'$\alpha$', r'$\beta$', r'$\gamma$', r'$\delta$', r'$\epsilon$'],
             fontsize=14
         )

         yticks = [0, 50, 100, 150]
         ax.set_yticks(yticks)
         ax.set_yticklabels(["$%.1f$" % y for y in yticks], fontsize=14);
```



2.1.9 Size, ratio and resolution

The size of the image is determined with `figsize` in inches (inches, 1 in = 2.4 cm) with resolution `dpi` - number of pixels per inch). The latter command creates a 1400x600 pixel image.

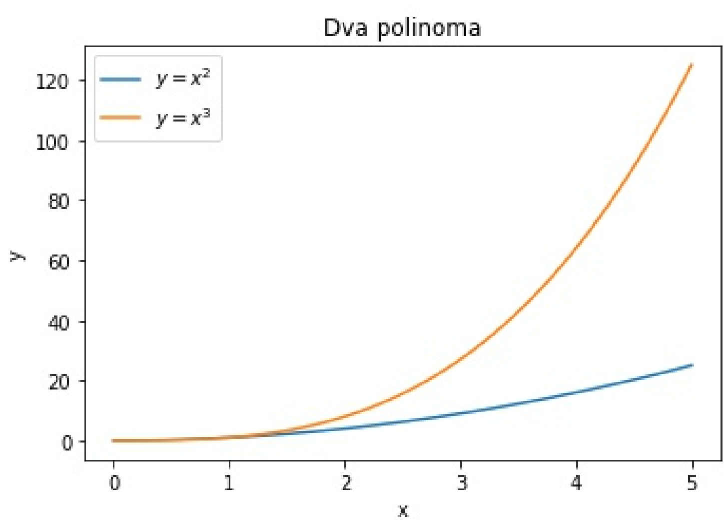
```
In [23]: fig, axes = plt.subplots(figsize=(7, 3), dpi=200)

         x = np.linspace(0, 5, 100)
         y = np.linspace(-np.pi, np.pi, 100)

         axes.plot(x, y, 'r')
         axes.set_xlabel('x')
         axes.set_ylabel('y')
         axes.set_title('Linearna funkcija');
```



2.1.10 Legend, tags and titles

To better read the image, we often add a title, an axis and a legend. In all places, LaTeXsyntax can be used. A quality picture contains most of the above mentioned items.

```
In [24]: fig, ax = plt.subplots()

         ax.plot(x, x**2, label="$y = x^2$")
         ax.plot(x, x**3, label="$y = x^3$")
         ax.legend(loc=2) # upper left corner
         ax.set_xlabel('x')
         ax.set_ylabel('y')
         ax.set_title('Dva polinoma');
```



2.1.11 Saving an image

To save, we use the `savefig` method, where we can choose between PNG, JPG, EPS, SVG, PGF and PDF formats.

```
In [25]: fig.savefig('slika.png')
```

Set resolution in DPI units.

```
In [26]: fig.savefig('slika.png', dpi=200)
```

2.2 Example: Winter Olympics, Sochi 2014

On the case of information about the Olympic Games, we will get to know the tabular presentation of the data (attribute value) in the Orange package. We will try some common ways to graphically display data.

```
In [1]: %matplotlib inline
        %config InlineBackend.figure_formats = ['jpg']
        import matplotlib
        matplotlib.figure.Figure.__repr__ = lambda self: (
            f"<{self.__class__.__name__} size {self.bbox.size[0]:g}"
            f"x{self.bbox.size[1]:g} with {len(self.axes)} Axes>")

        import matplotlib.image as mpimg
        import matplotlib.pyplot as plt
        plt.style.use('PR.mplstyle')

        ---------------------------------------------------------------------------

        FileNotFoundError                         Traceback (most recent call last)

        ~/anaconda3/lib/python3.6/site-packages/matplotlib/style/core.py in use(style)
        112             try:
    --> 113                 rc = rc_params_from_file(style, use_default_template=False)
        114                 _apply_style(rc)


        ~/anaconda3/lib/python3.6/site-packages/matplotlib/__init__.py in rc_params_from_file(fname, fa
       1028     """
    -> 1029     config_from_file = _rc_params_in_file(fname, fail_on_error)
       1030


        ~/anaconda3/lib/python3.6/site-packages/matplotlib/__init__.py in _rc_params_in_file(fname, fai
        944     rc_temp = {}
    --> 945     with _open_file_or_url(fname) as fd:
        946         try:


        ~/anaconda3/lib/python3.6/contextlib.py in __enter__(self)
         80         try:
    ---> 81             return next(self.gen)
         82         except StopIteration:


        ~/anaconda3/lib/python3.6/site-packages/matplotlib/__init__.py in _open_file_or_url(fname)
        929             encoding = "utf-8"
    --> 930         with io.open(fname, encoding=encoding) as f:
```

```
    931             yield f


    FileNotFoundError: [Errno 2] No such file or directory: 'PR.mplstyle'


During handling of the above exception, another exception occurred:


    OSError                                   Traceback (most recent call last)

    <ipython-input-1-f11bad626864> in <module>
      8 import matplotlib.image as mpimg
      9 import matplotlib.pyplot as plt
---> 10 plt.style.use('PR.mplstyle')


    ~/anaconda3/lib/python3.6/site-packages/matplotlib/style/core.py in use(style)
    117                         "not a valid URL or path. See `style.available` for "
    118                         "list of available styles.")
--> 119                 raise IOError(msg % style)
    120
    121


    OSError: 'PR.mplstyle' not found in the style library and input is not a valid URL or path. See
```

2.2.1 Data presentation

This time we are dealing with athletes who took part in the Winter Olympics in the Russian resort Sochi near the Black Sea in 2014.

The following data (attributes) are available for each athlete:

- name and surname,
- age in years,
- date of birth,
- gender,
- height,
- body weight,
- no. won gold medals,
- no. won silver medals,
- no. won bronze medals,
- no. of all medals won,
- The sports category,
- the country it represents.

Question 2-2-1 With what kind of data type would you present each of the attributes?

In [2]:

Answer

So far, we have learned ways to store numerical data, such as integers and decimal numbers. Numerical data, such as the country and the name of the competitor, can not be easily represented in numerical form. We will use the `Orange` library, which stores the following data types along with numbers:

- [c]**ontinuous** attributes to represent numerical data (including integers),
- [d]**iscrete** attributes have a stock of values from a finite set. For example. Gender is an element of the `{man, woman}` set, or ice-cream flavors `{chocolate, vanilla, strawberry}`. Note that, unlike with numbers, there is no order between the elements of such sets.
- [s]**tring** of characters, stores the sets of characters of any (final) length.

Question 2-2-2 Which of the three types of data would you use for each of the athletes' attributes? You can find the solution if you look at the first few lines of the file `athletes.tab`.

In [2]:

Answer

2.2.2 `Orange` software package

We load the data into the object table, `Table`. The data types of attributes are specified in the file.

```
In [2]: from Orange.data.filter import SameValue
        from Orange.data import Table
        data = Table('podatki/athletes.tab')
```

Domain is a set of column names.

```
In [3]: data.domain
```

```
Out[3]: [age, gender, height, weight, gold_medals, silver_medals, bronze_medals, total_medals, sport, c
```

Check the types of individual attributes.

```
In [4]: for column in data.domain.variables:
            print(column, type(column))
```

```
age ContinuousVariable
gender DiscreteVariable
height ContinuousVariable
weight ContinuousVariable
gold_medals ContinuousVariable
silver_medals ContinuousVariable
bronze_medals ContinuousVariable
total_medals ContinuousVariable
sport DiscreteVariable
country DiscreteVariable
```

For discrete attributes we can access the set of values.

```
In [5]: data.domain['sport'].values
```

```
Out[5]: ['Alpine Skiing',
         'Biathlon',
         'Bobsleigh',
         'Cross-Country',
         'Curling',
         'Freestyle Skiing',
         'Ice Hockey',
```

```
         'Luge',
         'Nordic Combined',
         'Short Track',
         'Skeleton',
         'Ski Jumping',
         'Snowboard',
         'Speed Skating']
```

We can access individual lines:

```
In [6]: print(data[0])
        print()
        print(data[1:3])
```

```
[17, Male, 1.72, 68, 0, 0, 0, 0, Freestyle Skiing, United States] {1996-04-12, Aaron Blunck}

[[27, Male, 1.85, 85, 0, 0, 0, 0, Snowboard, Italy] {1986-05-14, Aaron March},
 [21, Male, 1.78, 68, 0, 0, 0, 0, Short Track, Kazakhstan] {1992-06-30, Abzal Azhgaliyev}]
```

We can access the attributes of each line. These modes are equivalent to accessing the sport on the sportsman in the first line:

```
In [7]: print(data[0, 8])
        print(data[0, data.domain['sport']])
        print(data[0, data.domain.index('sport')])
        print(data[0, 'sport'])  # neposredno po imenu atributa
```

```
Freestyle Skiing
Freestyle Skiing
Freestyle Skiing
Freestyle Skiing
```

We also access multiple columns at the same time:

```
In [8]: print(data[0, ['sport', 'name', 'age']])
        print(data[0, [8, 0, -2]])
```

```
[[Freestyle Skiing, 17] {Aaron Blunck}]
[[Freestyle Skiing, 17] {Aaron Blunck}]
```

Numerical data are stored in the numpy table within the `Table` object. We will not find names, date of birth, and country in this matrix. Why?

```
In [9]: data.X
```

```
Out[9]: array([[ 17.  ,   1.  ,   1.72, …,   0.  ,   5.  ,  79.  ],
               [ 27.  ,   1.  ,   1.85, …,   0.  ,  12.  ,  36.  ],
               [ 21.  ,   1.  ,   1.78, …,   0.  ,   9.  ,  39.  ],
               …,
               [ 28.  ,   0.  ,   1.68, …,   0.  ,  12.  ,  28.  ],
               [ 22.  ,   1.  ,   1.76, …,   1.  ,   5.  ,  16.  ],
               [ 19.  ,   0.  ,   1.58, …,   0.  ,   9.  ,  30.  ]])
```

2.2.3 Selecting a subset of rows

We use a filter to select a subset of rows. Let's create a filter object that includes a condition and call it on a subset of the data.

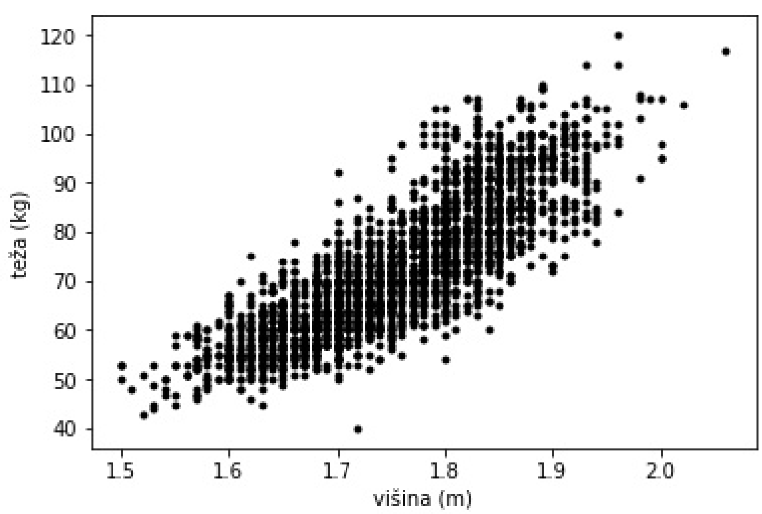
```
In [10]: # ustvarimo filter, SameValue(spremenljivka, vrednost)
         filt = SameValue(data.domain['sport'], 'Alpine Skiing')

         # izberi vse alpske smučarje
         data_subset = filt(data)
         data_subset
```

```
Out[10]: [[21, Male, 1.86, 82, 0, …] {1992-07-30, Adam Barwood},
          [18, Male, 1.70, 76, 0, …] {1995-04-22, Adam Lamhamedi},
          [23, Male, 1.78, 80, 0, …] {1990-09-13, Adam Zampa},
          [21, Female, 1.62, 56, 0, …] {1992-09-28, Adeline Baud},
          [29, Male, 1.82, 80, 0, …] {1984-09-18, Adrien Theaux},
          …
         ]
```

2.2.4 Display points in space

Let's see if the height and weight of athletes are linked. For each athlete we draw a point in the space of two variables - a Scatter plot.

```
In [11]: plt.figure()
         x = data.X[:, 2]    # višina
         y = data.X[:, 3]    # teža
         plt.plot(x, y, "k.")
         plt.xlabel('višina (m)')
         plt.ylabel('teža (kg)');
```



Question 2-2-3 It looks like the variables are linked. Are height and weight really connected? The answer to this question can be obtained with correlation measures. Using the latter, we measure whether two random variables are connected.

Pearson's correlation between the variables X and Y is defined with the following expression:

$$\rho = \frac{(x - \bar{x})(y - \bar{y})}{\sigma_x \sigma_y}$$

where x and y are vector of samples of random variables X and Y, $\bar{x}$ and $\bar{y}$ are mean values, σ_x, σ_y standard deviation. The ρ measure takes the values in the interval [-1, 1], where the -1 value means that the variables are negatively correlated - they are inversely proportional, and the value 1 is proportional. The value 0 indicates that the variables are independent.

```
In [12]: # implementiraj funkcijo, ki vrne Pearsonovo mero korelacije za vektorja x, y
         def pearson(x, y):
             pass
```

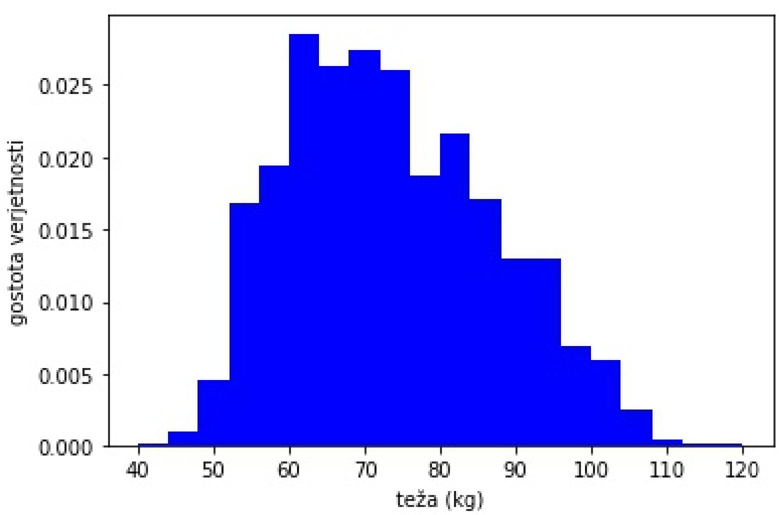
```
In [13]: # preveri ali sta višina in teža povezani
         pearson(x, y)
```

Answer

2.2.5 Display distributions

We present uncertainty in the observation of a random variable with the distribution function. The common way in which we obtain an estimate for the distribution of data is the use of a histogram - counting how many cases fall into the interval of the value of the variable. Let's look at an example of body weight.

```
In [14]: # porazdelitev tež
         weights = data.X[:, 3]

         plt.figure()
         plt.hist(weights, density=True, bins=20, color='blue')
         plt.xlabel('teža (kg)')
         plt.ylabel('gostota verjetnosti');
```



Question 2-2-4 Is the weight distribution different for different sports? What about the heights? Choose athletes of some sports and compare distributions between them.

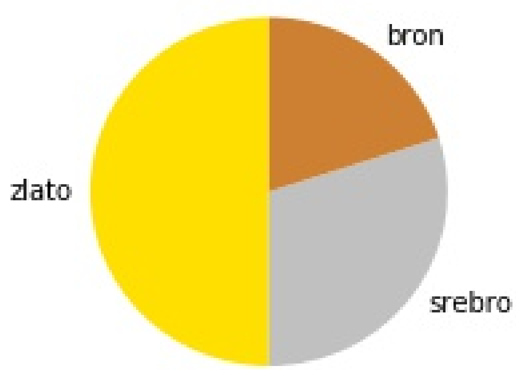
```
In [15]: # Primerjaj športe po porazdelitvi tež
         # Primerjaj športe po porazdelitvi višin
```

Answer

2.2.6 Prizes for reaching the highest places

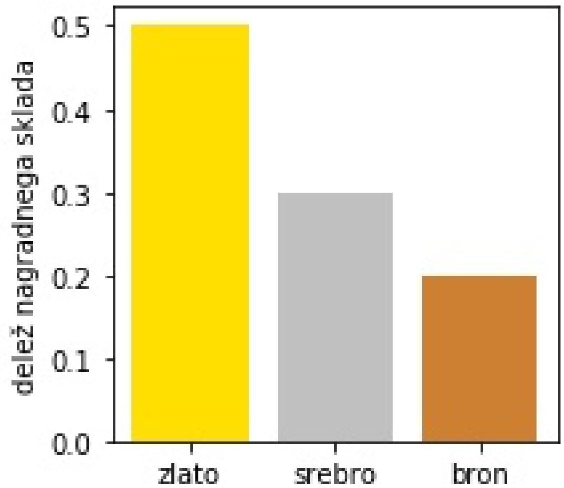
Another way of displaying distributions is a pie chart. Show what piece of cake each of the medals (gold $25,000, silver $15,000 bronze $10,000) brings .

```
In [16]: # prikaži primer slike in reprodukcija ; št medalj glede na državo
         # Nariši tortni diagram za vsako državo posebej

         # Denarni sklad; $25,000 za zlato, $15,000 za srebrno, $10,000 za bronasto medaljo
         total      = 25 + 15 + 10
         gold_ratio = 25 / total
         silv_ratio = 15 / total
         bron_ratio = 10 / total

         # barve medalj
         gold_color = '#FFDF00'
         silv_color = '#C0C0C0'
         bron_color = '#CD7F32'

         plt.figure(figsize=(3, 3))
         plt.pie((gold_ratio, silv_ratio, bron_ratio),
                 labels=('zlato', 'srebro', 'bron', ),
                 colors=(gold_color, silv_color, bron_color, ),
                 startangle=90);
```



An easier to read bar diagram:

```
In [17]: # lažje berljivi stolpični diagram
         plt.figure(figsize=(3, 3))
         plt.bar(range(3), height=(gold_ratio, silv_ratio, bron_ratio),
                 tick_label=('zlato', 'srebro', 'bron'),
                 color=(gold_color, silv_color, bron_color))
         plt.ylabel('delež nagradnega sklada');
```



2.2.7 Gender of participants

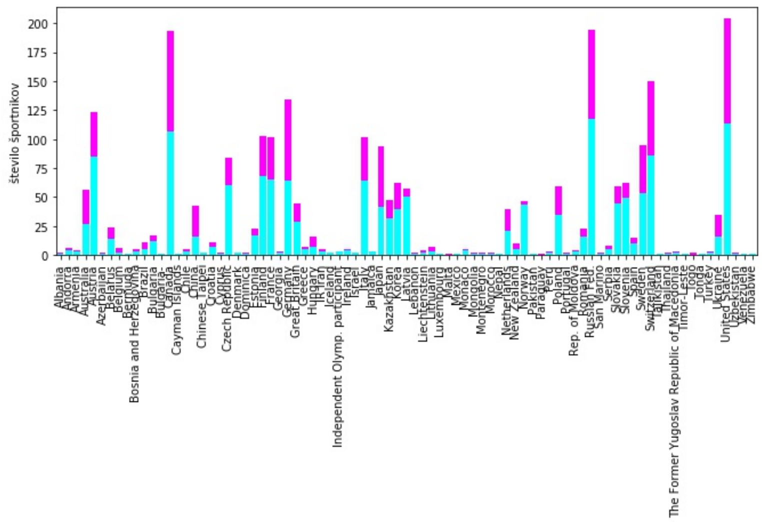
We show an even more informative distribution that shows the number of men and women participating in games for each country. First, calculate the distribution.

```
In [18]: countries = data.domain['country'].values
         gender_by_country = dict()

         for country in countries:
             # Filter by countries
             filt        = SameValue(data.domain['country'], country)
             data_subset = filt(data)

             # Filter males
             filt        = SameValue(data.domain['gender'], 'Male')
             data_subset_male = filt(data_subset)

             # Filter females
             filt        = SameValue(data.domain['gender'], 'Female')
             data_subset_female = filt(data_subset)

             # Store gender counts
             gender_by_country[country] = {
                 'Male': len(data_subset_male),
                 'Female': len(data_subset_female),
             }
```

Then draw a picture using the bar function:

```
In [19]: m = [gender_by_country[country]['Male'] for country in countries]
         f = [gender_by_country[country]['Female'] for country in countries]
         x = range(len(countries))

         plt.figure(figsize=(11, 4))
         plt.bar(x, m, color='cyan', align='center')
         plt.bar(x, f, bottom=m, color='magenta', align='center')
         plt.xlim(-0.5, len(countries)-0.5)
```

```
plt.xticks(x)
plt.gca().set_xticklabels(countries, rotation=90)
plt.ylabel('število športnikov');
```



Question 2-2-5 Add a legend to the graph.

In [20]:

Answer

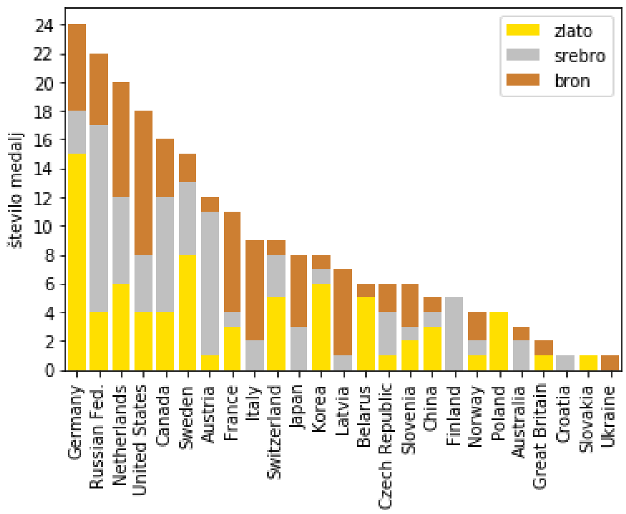
Question 2-2-6 Edit the above graph so that the sport is arranged by the number of participants and add the legend.

In [20]:

Answer

2.2.8 The most successful countries

Question 2-2-7 Draw a picture, similar to the one below. The diagram shows the distribution of individual medals by country. Tip: Prepare the data first, then draw a diagram. Take a look at previous examples.




```
In [20]: # izračunaj distribucijo medalj
```

```
In [21]: # izriši distribucijo
```

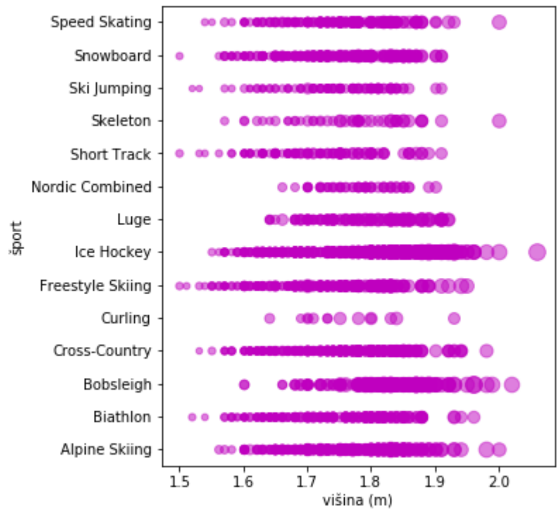
Answer

2.2.9 Composite visualizations

The purpose of a good visualization is the correct amount of data in a given space. This should not be too big, but we want to make the most of the space. Let's look at the example of drawing distribution of data about height and weight by sport category.

```
In [22]: # priprava podatkov
         # teža in višina glede na sport; sport se nahaja v 8 stolpcu
         sports = data.domain['sport'].values
         weights_by_sport = dict()
         heights_by_sport = dict()
         ages_by_sport = dict()

         for sport in sports:
             filt = SameValue(data.domain['sport'], sport)
             data_subset = filt(data)

             w = data_subset[:, data.domain.index('weight')].X.ravel()
             h = data_subset[:, data.domain.index('height')].X.ravel()
             a = data_subset[:, data.domain.index('age')].X.ravel()

             weights_by_sport[sport] = w
             heights_by_sport[sport] = h
             ages_by_sport[sport]    = a
```

Question 2-2-8 Draw a picture, similar to the one below. The diagram shows the height distribution by sports. For each player, we draw a point where the point size is proportional to the weight of the athlete. Axes x and y will be used to draw the height on the x-axis, and the y-axis will be an individual sporting industry.




```
In [23]: # napiši kodo za izris slike
```

Aswer

Question 2-2-9 Edit the graph above so that the sports are arranged by the average height. Try also to change the quantities on the individual axes (x, y, the size of the dots).

```
In [24]:
```

Answer

Chapter 3

Distributions and outliers

The probability distribution P is a function over the random variable X, which assigns a probability to each possible value of the variable - a value in the interval $[0, 1]$. The variable X can be continuous, discrete, one- or multi- dimensional.

The value of $P(X)$ is for each possible value of the variable X (the entire definition range), and the sum over the definition range must be the same as 1.

For each probability distribution that we will learn below, we give: * the definition area (i.e., what is the X); * form (a formula that assigns a probability to each value of X), * parameters (constants that determine the values and / or the shape of the function)

Guide: *The choice of the distribution to model depends on the nature of the data.*

In [1]:
```
%matplotlib inline
%config InlineBackend.figure_formats = ['jpg']
import matplotlib
matplotlib.figure.Figure.__repr__ = lambda self: (
    f"<{self.__class__.__name__} size {self.bbox.size[0]:g}"
    f"x{self.bbox.size[1]:g} with {len(self.axes)} Axes>")

import matplotlib.pyplot as plt
plt.style.use('PR.mplstyle')
import numpy as np
np.random.seed(42)
```

```
---------------------------------------------------------------------------

FileNotFoundError                         Traceback (most recent call last)

~/anaconda3/lib/python3.6/site-packages/matplotlib/style/core.py in use(style)
    112             try:
--> 113                 rc = rc_params_from_file(style, use_default_template=False)
    114                 _apply_style(rc)


~/anaconda3/lib/python3.6/site-packages/matplotlib/__init__.py in rc_params_from_file(fname, fa
   1028     """
-> 1029     config_from_file = _rc_params_in_file(fname, fail_on_error)
   1030
```

```
    ~/anaconda3/lib/python3.6/site-packages/matplotlib/__init__.py in _rc_params_in_file(fname, fail
    944     rc_temp = {}
--> 945     with _open_file_or_url(fname) as fd:
    946         try:


    ~/anaconda3/lib/python3.6/contextlib.py in __enter__(self)
     80         try:
---> 81             return next(self.gen)
     82         except StopIteration:


    ~/anaconda3/lib/python3.6/site-packages/matplotlib/__init__.py in _open_file_or_url(fname)
    929             encoding = "utf-8"
--> 930         with io.open(fname, encoding=encoding) as f:
    931             yield f


    FileNotFoundError: [Errno 2] No such file or directory: 'PR.mplstyle'


During handling of the above exception, another exception occurred:


    OSError                                   Traceback (most recent call last)

    <ipython-input-1-9e9777efd08e> in <module>
      7
      8 import matplotlib.pyplot as plt
----> 9 plt.style.use('PR.mplstyle')
     10 import numpy as np
     11 np.random.seed(42)


    ~/anaconda3/lib/python3.6/site-packages/matplotlib/style/core.py in use(style)
    117                         "not a valid URL or path. See `style.available` for "
    118                         "list of available styles.")
--> 119                 raise IOError(msg % style)
    120
    121


    OSError: 'PR.mplstyle' not found in the style library and input is not a valid URL or path. See
```

3.1 Gaussian (normal) distribution

The normal (or Gaussian) distribution is the distribution over the whole range of real numbers. It is one of the most common distributions, which is used in practice, since a lot of data is bell-shaped. The function is *symmetric* and is given by two parameters, the mean and the variance.

Variable type: one- or multi- dimensional, continuous.

Definition range: $(-\infty, +\infty)$

Format:

$$P(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-(x-\mu)^2}{2\sigma^2}}$$

Parameters: * μ middle/hope * σ^2 varianca

```
In [2]: from scipy.stats import multivariate_normal as mvn

        # Parametri določajo obliko funkcije
        mu     = 0   # sredina
        sigma2 = 1   # varianca

        n = 500  # velikost vzorca
        sample = mvn.rvs(mu, sigma2, size=n)  # naključen vzorec n primerov

        xr = np.linspace(-5, 5, 100)               # interval X
        P  = [mvn.pdf(x, mu, sigma2) for x in xr]  # porazdelitvena funkcija

        # Histogram - porazdelitev naključnih VZORCEV x glede na P(x)
        plt.figure(figsize=(9, 3))
        plt.subplot(1, 2, 1)
        plt.title("vzorec")
        plt.hist(sample)  #
        plt.xlabel("X")
        plt.ylabel("število primerov")

        # Graf porazdelitvene funkcije
        plt.subplot(1, 2, 2)
        plt.title("graf porazdelitve")
        plt.plot(xr, P)   # nariši P(x)
        plt.ylabel("P(x)")
        plt.xlabel("X");
```

```
        ---------------------------------------------------------------------------

        NameError                                 Traceback (most recent call last)

        <ipython-input-2-9a8f12c221c3> in <module>
          8 sample = mvn.rvs(mu, sigma2, size=n)  # naključen vzorec n primerov
          9
    ---> 10 xr = np.linspace(-5, 5, 100)               # interval X
         11 P  = [mvn.pdf(x, mu, sigma2) for x in xr]  # porazdelitvena funkcija
         12


        NameError: name 'np' is not defined
```

3.1.1 Learning the parameters

In practice, we do not know the real values of the parameters. *Parameters are learned from the sample.* The advantage of the process is that we can then conclude on new samples, i.e., each possible value of the variable is determined by the probability.

We have a sample of random variable X of size n.

$$X_1, X_2, ..., X_n$$

For a normal distribution, we get the *estimate* for the parameters as follows:

$\mu = E[X_i] = \bar{X}$

$\sigma^2 = \frac{n-1}{n} E[(X_i - \bar{X})^2] = \frac{n-1}{n} var[x]$

The μ value is the average of the sample. The σ^2 value is the corrected variance of the sample.

We estimate the parameters from the sample:

```
In [3]: mu_fit = np.mean(sample)
        sigma2_fit = (n-1)/n * np.var(sample)

        mu_fit, sigma2_fit


        ---------------------------------------------------------------------------

        NameError                                 Traceback (most recent call last)

        <ipython-input-3-2546e0e7037a> in <module>
    ----> 1 mu_fit = np.mean(sample)
          2 sigma2_fit = (n-1)/n * np.var(sample)
          3
          4 mu_fit, sigma2_fit


        NameError: name 'np' is not defined
```

The estimated values of the parameters are similar to the real values ($\mu = 0$, $\sigma^2 = 1$).

In one picture we compare the distribution with the learned parameters with the correct distribution:

```
In [4]: P_fit = [mvn.pdf(x, mu_fit, sigma2_fit) for x in xr]

        plt.figure()
        plt.hist(sample,    label="vzorec", normed=True)
        plt.plot(xr, P,     label="P(X) resnična", linewidth=2.0)
        plt.plot(xr, P_fit, label="P(X) ocenjena", linewidth=2.0)
        plt.legend();


        ---------------------------------------------------------------------------

        NameError                                 Traceback (most recent call last)

        <ipython-input-4-444fba6df4f3> in <module>
    ----> 1 P_fit = [mvn.pdf(x, mu_fit, sigma2_fit) for x in xr]
```

```

plt.figure()
plt.hist(sample,    label="vzorec", normed=True)
plt.plot(xr, P,     label="P(X) resnična", linewidth=2.0)


NameError: name 'xr' is not defined
```

Question 3-1-1 Check how the accuracy of the parameter estimation changes with the size n of the sample.

In [5]:

Answer

3.2 Student's distribution

Student's distribution (or t-distribution) is the distribution over the entire range of real numbers. Its shape is symmetrical and similar to normal distribution. It is less sensitive to *outliers in small samples*.

Variable type: one-dimensional, continuous.

Definition range: $x \in (-\infty, +\infty)$

Format:

$$P(x) = \frac{\Gamma[(\nu+1)/2]}{\sqrt{\nu\pi}\,\Gamma(\nu/2)}\left(1+\frac{x^2}{\nu}\right)^{-(\nu+1)/2}$$

Parameters: * ν number of degrees of freedom

In [5]:
```python
from scipy.stats import t as student

# Parametri določajo obliko funkcije
nu = 2 # prostostne stopnje

n = 8  # velikost vzorca
sample = student.rvs(nu, size=n)  # naključen vzorec n primerov spremenljivke

xr = np.linspace(-5, 5, 100)  # interval X
P  = [student.pdf(x, nu) for x in xr]  # porazdelitvena funkcija

# Histogram - porazdelitev naključnih VZORCEV x glede na P(x)
plt.figure(figsize=(9, 3))
plt.subplot(1, 2, 1)
plt.title("Vzorec")
plt.hist(sample)  #
plt.xlabel("X")
plt.ylabel("število primerov")

# Graf porazdelitvene funkcije
plt.subplot(1, 2, 2)
plt.title("Graf porazdelitve")
plt.plot(xr, P)  # nariši P(x)
plt.ylabel("P(x)")
plt.xlabel("X");
```

```
---------------------------------------------------------------------------

NameError                                 Traceback (most recent call last)

<ipython-input-5-5b4ee6bcbbcb> in <module>
      7 sample = student.rvs(nu, size=n)  # naključen vzorec n primerov spremenljivke
      8
----> 9 xr = np.linspace(-5, 5, 100)  # interval X
     10 P  = [student.pdf(x, nu) for x in xr]  # porazdelitvena funkcija
     11


NameError: name 'np' is not defined
```

3.2.1 Learning the parameters from the sample

Most distributions in the `scipy` library contain a `fit` function, which calculates the most likely values of the distribution parameters relative to the sample.

In one picture we compare the distribution with the learned parameters with the correct distribution

In [6]:
```
pars = student.fit(sample)
P_fit = [student.pdf(x, *pars) for x in xr ]

plt.figure()
plt.hist(sample,    label="vzorec", normed=True)
plt.plot(xr, P,     label="P(X) resnična", linewidth=2.0)
plt.plot(xr, P_fit, label="P(X) ocenjena", linewidth=2.0)
plt.legend();
```

```
---------------------------------------------------------------------------

NameError                                 Traceback (most recent call last)

<ipython-input-6-ff3cd866462a> in <module>
      1 pars = student.fit(sample)
----> 2 P_fit = [student.pdf(x, *pars) for x in xr ]
      3
      4 plt.figure()
      5 plt.hist(sample,    label="vzorec", normed=True)


NameError: name 'xr' is not defined
```

Question 3-1-2 Generate a sample with a small number (up to 20) of samples from normal distribution. Compare distribution estimates by means of normal and Student's distribution. Which distribution better evaluates the true distribution?

In [7]:
```
# Primerjaj Normalno in Studentovo porazdelitev pri majhnem vzorcu
# ...
```

Answer

3.3 Beta Distribution

The beta distribution is the distribution of the variable in the *limited interval* $[0,1]$. Its shape is very flexible, it can have one or two *maximums*. The distribution can be translated to any interval $[a,b]$ with summation (translation) and multiplying (spreading/narrowing) of the interval.

Variable type: x, one-dimensional, continuous, on a limited interval.

Definition range: $x \in [0,1]$

Format:

$$P(x) = \frac{1}{\mathrm{B}(\alpha,\beta)}\, x^{\alpha-1}(1-x)^{\beta-1}$$

Parameters: * a * b

```
In [8]: from scipy.stats import beta

        # Parametri določajo obliko funkcije
        a, b = (3, 2)    # parametra a, b

        n = 100                             # velikost vzorca
        sample = beta.rvs(a, b, size=n)        # naključen vzorec n primerov spremenljivke

        xr = np.linspace(0, 1, 100)                # interval X
        P  = [beta.pdf(x, a, b) for x in xr]  # porazdelitvena funkcija

        # Histogram - porazdelitev naključnih VZORCEV x glede na P(x)
        plt.figure(figsize=(9, 3))
        plt.subplot(1, 2, 1)
        plt.title("Vzorec")
        plt.hist(sample)  #
        plt.xlabel("X")
        plt.ylabel("Število primerov")

        # Graf porazdelitvene funkcije
        plt.subplot(1, 2, 2)
        plt.title("Graf porazdelitve")
        plt.plot(xr, P)   # nariši P(x)
        plt.ylabel("P(x)")
        plt.xlabel("X");
```

```
        ---------------------------------------------------------------------------

        NameError                                 Traceback (most recent call last)

        <ipython-input-8-0e4cc5111124> in <module>
          7 sample = beta.rvs(a, b, size=n)        # naključen vzorec n primerov spremenljivke
          8
    ----> 9 xr = np.linspace(0, 1, 100)                # interval X
         10 P  = [beta.pdf(x, a, b) for x in xr]  # porazdelitvena funkcija
         11
```

```
NameError: name 'np' is not defined
```

Question 3-1-3 Change the parameters a and b. How does the shape of the function change?

In [9]:

Answer

3.3.1 Learning the parameters from the sample

We use the `fit` function also to learn the Beta distribution parameters.

In one picture we compare the distribution with the learned parameters with the correct distribution.

In [9]:
```
parameters = beta.fit(sample)
P_fit = [beta.pdf(x, *parameters) for x in xr ]

plt.figure()
plt.hist(sample,    label="Vzorec", normed=True)
plt.plot(xr, P,     label="P(X) resnična", linewidth=2.0)
plt.plot(xr, P_fit, label="P(X) ocenjena", linewidth=2.0)   # ocenjena porazdelitev je model
plt.legend();
```

```
---------------------------------------------------------------------------

NameError                                 Traceback (most recent call last)

<ipython-input-9-1dd1df6971b7> in <module>
      1 parameters = beta.fit(sample)
----> 2 P_fit = [beta.pdf(x, *parameters) for x in xr ]
      3
      4 plt.figure()
      5 plt.hist(sample,    label="Vzorec", normed=True)


NameError: name 'xr' is not defined
```

Question 3-1-4 Change the parameters a and b and the size n of the sample. How does the quality of the fitting change?

In [10]:

Answer

3.4 Example: finding unfunny jokes

This time we will look at the Jester dataset, which is quite similar to that of the homework. It is a collection of 100 jokes rated by 23,500 users with a rating of -10 (disastrous) to 10 (excellent). The assessment is therefore a continuous variable.

Our main goal will be to model statistics in a dataset using known distributions. This will allow us to find **outliers among the jokes** and evaluate their statistical significance - the likelihood that it is an outlier or not.

Let's start with a random joke from the dataset:

```
A mechanical, electrical and a software engineer from Microsoft were driving through the
desert when the car broke down. The mechanical engineer said "It seems to be a problem
with the fuel injection system, why don't we pop the hood and I'll take a look at it."
To which the electrical engineer replied, "No I think it's just a loose ground wire, I'll
get out and take a look." Then, the Microsoft engineer jumps in. "No, no, no. If we just
close up all the windows, get out, wait a few minutes, get back in, and then reopen the
windows everything will work fine."
```

```
In [1]: %matplotlib inline
        %config InlineBackend.figure_formats = ['jpg']
        import matplotlib
        matplotlib.figure.Figure.__repr__ = lambda self: (
            f"<{self.__class__.__name__} size {self.bbox.size[0]:g}"
            f"x{self.bbox.size[1]:g} with {len(self.axes)} Axes>")

        import matplotlib.pyplot as plt
        plt.style.use('PR.mplstyle')
        import numpy as np

        ---------------------------------------------------------------------------

        FileNotFoundError                         Traceback (most recent call last)

        ~/anaconda3/lib/python3.6/site-packages/matplotlib/style/core.py in use(style)
        112             try:
    --> 113                 rc = rc_params_from_file(style, use_default_template=False)
        114                 _apply_style(rc)


        ~/anaconda3/lib/python3.6/site-packages/matplotlib/__init__.py in rc_params_from_file(fname, fa
       1028     """
    -> 1029     config_from_file = _rc_params_in_file(fname, fail_on_error)
       1030


        ~/anaconda3/lib/python3.6/site-packages/matplotlib/__init__.py in _rc_params_in_file(fname, fai
        944     rc_temp = {}
    --> 945     with _open_file_or_url(fname) as fd:
        946         try:


        ~/anaconda3/lib/python3.6/contextlib.py in __enter__(self)
         80         try:
    ---> 81             return next(self.gen)
         82         except StopIteration:


        ~/anaconda3/lib/python3.6/site-packages/matplotlib/__init__.py in _open_file_or_url(fname)
        929             encoding = "utf-8"
```

```
--> 930         with io.open(fname, encoding=encoding) as f:
    931             yield f


    FileNotFoundError: [Errno 2] No such file or directory: 'PR.mplstyle'


During handling of the above exception, another exception occurred:


    OSError                                   Traceback (most recent call last)

    <ipython-input-1-779ea8d3610b> in <module>
      7
      8 import matplotlib.pyplot as plt
----> 9 plt.style.use('PR.mplstyle')
     10 import numpy as np


    ~/anaconda3/lib/python3.6/site-packages/matplotlib/style/core.py in use(style)
    117                         "not a valid URL or path. See `style.available` for "
    118                         "list of available styles.")
--> 119                 raise IOError(msg % style)
    120
    121


    OSError: 'PR.mplstyle' not found in the style library and input is not a valid URL or path. See
```

The data is a matrix of magnitude 23500×100 with continuous values. The value of 99 represents an unknown value; therefore, such values must not be taken into account.

```
In [2]: X = np.genfromtxt('podatki/jester-data.csv', delimiter=',',)[:, 1:]
        X[np.where(X == 99)] = float("nan")  # neznanih vrednosti ne smemo upoštevati

        print("velikost:", X.shape)
        print("skupno število ocen:", X.size - np.sum(np.isnan(X)))


        ---------------------------------------------------------------------------

        NameError                                 Traceback (most recent call last)

        <ipython-input-2-2286ff858560> in <module>
    ----> 1 X = np.genfromtxt('podatki/jester-data.csv', delimiter=',',)[:, 1:]
          2 X[np.where(X == 99)] = float("nan")  # neznanih vrednosti ne smemo upoštevati
          3
          4 print("velikost:", X.shape)
          5 print("skupno število ocen:", X.size - np.sum(np.isnan(X)))


        NameError: name 'np' is not defined
```

Let's see what is the distribution of all valid ratings.

```
In [3]: data = X[np.isnan(X) == False]
        plt.hist(data, bins=100)
        plt.xlabel("x - Ocena šale")
        plt.ylabel("P(x)");
```

```
        ---------------------------------------------------------------------------

        NameError                                 Traceback (most recent call last)

        <ipython-input-3-3a0ac0860547> in <module>
    ----> 1 data = X[np.isnan(X) == False]
          2 plt.hist(data, bins=100)
          3 plt.xlabel("x - Ocena šale")
          4 plt.ylabel("P(x)");


        NameError: name 'X' is not defined
```

We see that most ratings are neutral (around 0), many positive (between 3 and 10) and some very bad (-10). The least is the average bad (-9 to -1). Nevertheless, this distribution has the following problems: * The sample is not impartial. Each user rated a different number of jokes. * The distribution does not look like any of the known.

How would you compare the jokes with respect to their grades?

Let's first look at how many valid ratings each joke received:

```
In [4]: (np.isnan(X) == False).sum(axis=0)   # vsota po posameznih šalah
```

```
        ---------------------------------------------------------------------------

        NameError                                 Traceback (most recent call last)

        <ipython-input-4-12b9b805f100> in <module>
    ----> 1 (np.isnan(X) == False).sum(axis=0)   # vsota po posameznih šalah


        NameError: name 'np' is not defined
```

Each joke received a few thousand ratings, which is sufficient for a statistical comparison.

Let's imagine two new random variables: * X is the average of individual jokes, * Y is the variance of individual jokes.

Important: The variables are derived from two calculated quantities. The variables X and Y are not parameters of normal distribution!

For each of these variables X and Y we have a sample size of 100, one case for each joke. In calculating, we should skip the unknown values:

```
In [5]: means     = []
        variances = []
        for i in range(X.shape[1]):
            s = np.mean(X[:, i][np.isnan(X[:, i]) == False])
            v = np.var(X[:, i][np.isnan(X[:, i]) == False])
```

```
    means.append(s)
    variances.append(v)
```

```
---------------------------------------------------------------------------

NameError                                 Traceback (most recent call last)

<ipython-input-5-f72e975a5add> in <module>
  1 means     = []
  2 variances = []
----> 3 for i in range(X.shape[1]):
  4     s = np.mean(X[:, i][np.isnan(X[:, i]) == False])
  5     v = np.var(X[:, i][np.isnan(X[:, i]) == False])


NameError: name 'X' is not defined
```

Question 3-2-1 What is the interpretation of X and Y? What does it mean if the joke has a high variance among all ratings? What does it mean if a joke has a high average rating?

Answer

Let's write out some of the best, worst rated jokes, and some with a high and low variance. You can read and compare them for fun, e.g., open the file `podatki/jokes/init1.html`:

```
A man visits the doctor. The doctor says "I have bad news for you. You have cancer and
Alzheimer's disease". The man replies "Well, thank God I don't have cancer!"
```

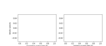
```
In [6]: n = 3
        for data, name in [(means, "Povprečje (X)"), (variances, "Varianca (Y)")]:
            inxs = np.argsort(data)[:n]
            print("Kriterij: %s" % name)
            print("\tSpodnjih %d:" % n)
            for i in inxs:
                print("\t\tšala %d, povp.: %.2f, var.: %.2f" % (i+1, means[i], variances[i]))

            inxs = np.argsort(data)[::-1][:n]
            print("\tZgornjih %d:" % n)
            for i in inxs:
                print("\t\tšala %d, povp.: %.2f, var.: %.2f" % (i+1, means[i], variances[i]))
            print()
```

```
---------------------------------------------------------------------------

NameError                                 Traceback (most recent call last)

<ipython-input-6-755b10deed5b> in <module>
  1 n = 3
  2 for data, name in [(means, "Povprečje (X)"), (variances, "Varianca (Y)")]:
----> 3     inxs = np.argsort(data)[:n]
  4     print("Kriterij: %s" % name)
  5     print("\tSpodnjih %d:" % n)
```

```
NameError: name 'np' is not defined
```

Let's also draw up the distribution of samples X and Y.

```
In [7]: plt.figure(figsize=(9, 3))
        plt.subplot(1, 2, 1)
        plt.hist(means, normed=False, bins=12)
        plt.xlabel("povprečna ocena šale (X)")
        plt.ylabel("število vzorcev")
        plt.text(-3, 2, "< osamelci?", rotation=90, verticalalignment="bottom", color="red")

        plt.subplot(1, 2, 2)
        plt.hist(variances, normed=False, bins=12)
        plt.xlabel("varianca ocene šale (Y)");
```



This looks better. Most jokes are therefore on average positive, very few are negative. The distribution look like familiar distributions, where the majority of cases (jokes) are distributed around the mean value, but there are fewer extreme values.

Let's look for a moment at the distribution of average ratings. It seems that we have some **outliers** - very bad jokes rated less than $X = -2$. How significant is the drop from $X = -2$ down? To answer this question, let us learn the basics of data modeling using probability distributions.

The average rating seems to be normally distributed. What are the most likely distribution parameters?

```
In [8]: from scipy.stats import multivariate_normal as mvn

        data = means

        # Ocenimo parametre normalne (Gaussove) porazdelitve
        n = len(data)
        mu = np.mean(data)                # ocena sredine
        sigma2 = (n-1)/n * np.var(data) # ocena variance

        plt.figure()
        counts, bins, _ = plt.hist(data, normed=True, label="vzorec", bins=10)    # dobimo razpon
        pdf = [mvn.pdf(x, mu, sigma2) for x in bins]                              # pdf: [p]robability
        plt.plot(bins, pdf, "-", label="model", linewidth=2.0)
```

```
plt.xlabel("povprečna ocena (X)")
plt.ylabel("P(X)")

plt.legend(loc=2);

---------------------------------------------------------------------------

NameError                                 Traceback (most recent call last)

<ipython-input-8-917593c22610> in <module>
  5 # Ocenimo parametre normalne (Gaussove) porazdelitve
  6 n = len(data)
----> 7 mu = np.mean(data)              # ocena sredine
  8 sigma2 = (n-1)/n * np.var(data) # ocena variance
  9


NameError: name 'np' is not defined
```

We can estimate that the distribution quite well matches the pattern. How statistically significant are jokes with a measurement value less than $X = -2.0$? *How unusually bad are these jokes really?* In order to answer this question we will calculate the so called *p-value*. Using the p-value, we evaluate *the oddness* of the measurement, in our case, the average joke rating.

Definition. The p-value is the probability that a given or less (or greater) value is obtained when sampling one value of the random variable.

The easiest way to look at the definition is graphically. Let's look at the distribution function obtained with the estimated parameters μ and σ^2.

```
In [9]: # Meritev, ki bi jo radi statisticno ocenili
        qx = -2


        # Izračunamo P(x) za dovolj velik interval
        xr    = np.linspace(-5, 5, 100)
        width = xr[1] - xr[0]       # sirina intervala
        Px = [mvn.pdf(x, mu, sigma2) * (xr[1]-xr[0])   for x in xr]


        # Vse vrednosti, ki so manjše ali enake od qx
        ltx     = xr[xr <= qx]

        # Množimo s širino intervala, da dobimo ploščino pod krivuljo
        P_ltx = [mvn.pdf(x, mu, sigma2) * width for x in ltx]

        # p-vrednost: ploscina pod krivuljo P(x) za vse vrednosti, manjse od qx
        p_value = np.sum(P_ltx)

        # Graf funkcije
        plt.figure()
        plt.plot(xr, Px, linewidth=2.0)
        plt.fill_between(ltx, 0, P_ltx, alpha=0.2)
        plt.text(qx, mvn.pdf(qx, mu, sigma2) * width,
```

```
                 "p=%f" % p_value,
                 horizontalalignment="right",
                 verticalalignment="center",
                )

        plt.xlabel("X - povprečna ocena šale.")
        plt.ylabel("P(X)")
        plt.legend()


        # Poglejmo, ali je meritev statistično značilna pri danem pragu alpha (0.05, 0.01, 0.001 ... )
        alpha = 0.05
        if p_value < alpha:
                sig = "JE"
        else:
                sig = "NI"

        # Rezultat statističnega testa
        print("Verjetnost šale z oceno %.3f ali manj: " % qx + "%.3f" % (100 * p_value) +  " %")
        print("Nenavadnost šale %s statistično značilna (prag = %.3f" % (sig, 100*alpha), "%)")
```

```
        ---------------------------------------------------------------------------

        NameError                                 Traceback (most recent call last)

        <ipython-input-9-e30847b2d35c> in <module>
          4
          5 # Izračunamo P(x) za dovolj velik interval
    ----> 6 xr    = np.linspace(-5, 5, 100)
          7 width = xr[1] - xr[0]       # sirina intervala
          8 Px = [mvn.pdf(x, mu, sigma2) * (xr[1]-xr[0])   for x in xr]


        NameError: name 'np' is not defined
```

Now, for every extreme value in the data (either high or low), we can estimate statistically the value of its unusuality. At the set threshold, e.g. $\alpha = 0.05$ we can make a decision whether or not some measurements are outliers or not.

Question 3-2-2 Write out all the outlier jokes whose average estimate of X is statistically significant at the threshold of $\alpha = 0.05$. Find also the outliers among *well-rated* jokes.

In [10]:

Answer

Question 3-2-3 Try modelin the distribution with other distributions (Student's, Beta). Is any of these distributions more appropriate?

In [10]:

Answer

Question 3-2-4 Repeat the analysis for the variable Y - variance of joke ratings. Answer the questions: * Which of the distributions (Normal, Student, Beta) best fits the pattern? * What are statistically significant jokes (with high or low variance)? * What does it mean if the joke has a high or low value of Y?

```
In [10]:
```

Answer

Chapter 4

Clustering

By using *uncontrolled modeling* methods we discover unknown structure in data. The central assumption of such methods is that there are subsets of similar cases in the data.

In [1]:
```
import numpy as np
np.random.seed(42)

%matplotlib inline
%config InlineBackend.figure_formats = ['jpg']
import matplotlib
matplotlib.figure.Figure.__repr__ = lambda self: (
    f"<{self.__class__.__name__} size {self.bbox.size[0]:g}"
    f"x{self.bbox.size[1]:g} with {len(self.axes)} Axes>")

import matplotlib.pyplot as plt
plt.style.use('PR.mplstyle')
```

```
---------------------------------------------------------------------------

FileNotFoundError                         Traceback (most recent call last)

~/anaconda3/lib/python3.6/site-packages/matplotlib/style/core.py in use(style)
    112             try:
--> 113                 rc = rc_params_from_file(style, use_default_template=False)
    114                 _apply_style(rc)


~/anaconda3/lib/python3.6/site-packages/matplotlib/__init__.py in rc_params_from_file(fname, fa
   1028     """
-> 1029     config_from_file = _rc_params_in_file(fname, fail_on_error)
   1030


~/anaconda3/lib/python3.6/site-packages/matplotlib/__init__.py in _rc_params_in_file(fname, fai
    944     rc_temp = {}
--> 945     with _open_file_or_url(fname) as fd:
    946         try:
```

```
    ~/anaconda3/lib/python3.6/contextlib.py in __enter__(self)
     80         try:
---> 81             return next(self.gen)
     82         except StopIteration:


    ~/anaconda3/lib/python3.6/site-packages/matplotlib/__init__.py in _open_file_or_url(fname)
    929             encoding = "utf-8"
--> 930         with io.open(fname, encoding=encoding) as f:
    931             yield f


    FileNotFoundError: [Errno 2] No such file or directory: 'PR.mplstyle'


During handling of the above exception, another exception occurred:


    OSError                                   Traceback (most recent call last)

    <ipython-input-1-4b7405d91cad> in <module>
     10
     11 import matplotlib.pyplot as plt
---> 12 plt.style.use('PR.mplstyle')


    ~/anaconda3/lib/python3.6/site-packages/matplotlib/style/core.py in use(style)
    117                         "not a valid URL or path. See `style.available` for "
    118                         "list of available styles.")
--> 119                 raise IOError(msg % style)
    120
    121


    OSError: 'PR.mplstyle' not found in the style library and input is not a valid URL or path. See
```

4.1 K-means clustering

K-means clustering is one of the simpler and often used methods of an uncontrolled algorithm. Computational efficiency is also an important advantage. You can find additional details in literature.

Question 4-1-1 Implement a Group Detection Algorithm with K-means clustering. Hel pyourself with the following pseudocode:

```
Randomly select *K* points - centers.
**repeat**
     Determines the nearest center of each point.
     Calculate new centers - the centers of the respective groups.
**until** centers are no longer changing.
```

Calculate the distance between the points $\vec{x} = (x_1, x_2, ...x_p)$ and $\vec{y} = (y_1, y_2, ...y_p)$ using the Euclidean distance:

$$\|\vec{x}-\vec{y}\| = \sqrt{(x_1-y_1)^2+(x_2-y_2)^2+\ldots+(x_p-y_p)^2}$$

In [2]:

Answer

Question 4-1-2 What is the time complexity of the algorithm in relation to the number of cases and the number of attributes?

answer space

Answer

In [2]:
```
class KMeans:

    def __init__(self, k=10, max_iter=100):
        """
        Initialize KMeans clustering model.

        :param k
            Number of clusters.
        :param max_iter
            Maximum number of iterations.
        """
        self.k          = k
        self.max_iter   = max_iter

    def fit(self, X):
        """
        Fit the Kmeans model to data.

        :param X
            Numpy array of shape (n, p)
            n: number of data examples
            p: number of features (attributes)

        :return
            labels: array of shape (n, ), cluster labels (0..k-1)
            centers: array of shape (k, p, )
        """

        n, p    = X.shape
        labels  = np.random.choice(range(self.k), size=n, replace=True)

        # Choose k random data points for initial centers
        centers = np.array([X[i] for i in np.random.choice(range(X.shape[0]),
                                                           size=self.k)])
        i = 0
        while i < self.max_iter:

            # Find nearest cluster
            for j, x in enumerate(X):
                ki = np.argmin(np.sum((centers - x) ** 2, axis=1))
                labels[j] = ki
```

```
            # Store previous centers
            previous_centers = centers.copy()

            # Move centroid
            for ki in range(self.k):
                centers[ki] = X[labels == ki].mean(axis=0)
            i += 1

        return labels, centers
```

Rešitev je dosegljiva v resitve/voditelji.py.

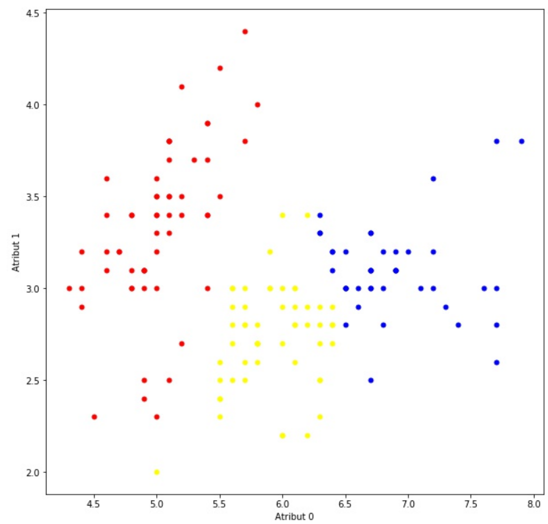
```
In [3]: %run 204-1.ipynb
```

```
ERROR:root:File `'204-1.ipynb.py'` not found.
```

4.1.1 Podatki

Metodo testiramo na podatkovni zbirki Iris, kjer za cvetlice v treh razredih merimo različne dimenzije cvetnih oz. venčnih listov. V podatkih najdemo tri gruče, približno takole. V rešitvi sta prikazana samo prva dva atributa, zato se gruče navidez prekrivajo.

```
In [4]: from sklearn.datasets import load_iris
        data = load_iris()

        X              = data["data"]
        true_clusters = data["target"]

        # Testirajte razred KMeans

        # Trenutno so gruče dodeljene naključno
        model = KMeans(k=3, max_iter=10)
        labels, centers = model.fit(X[:, :2])

        plt.figure(figsize=(10, 10))
        color = {0:"red", 1:"blue", 2:"yellow"}
        for c, x in zip(labels, X):
            plt.plot(x[0], x[1], ".", color=color[c], markersize=10.0)
        plt.xlabel("Atribut 0")
        plt.ylabel("Atribut 1")
        plt.show()
```



Question 4-1-3 Draw the status of the labels and the centre of the groups in each iteration of the algorithm.

```
In [5]: # ...
```

4.1.2 Evaluating the effectiveness of group discovery

Assessing the effectiveness of group discovery is one of the unresolved challenges of machine learning. We know measures like silhouette or adjusted random index.

Alternatively, we can know the real classes to which the data belongs. This is true for the iris database, where the flowers are arranged in three classes.

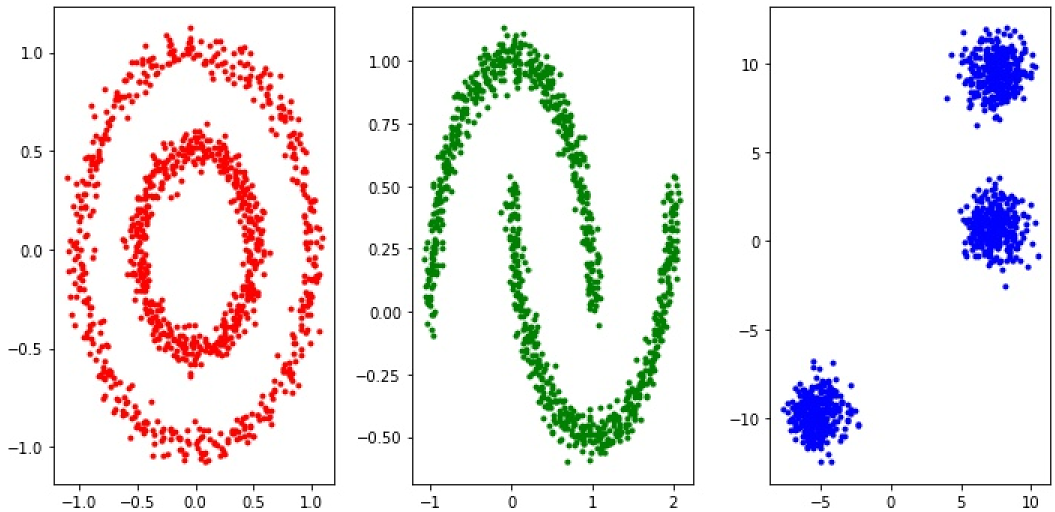
Question 4-1-4 Check how well the result of your method matches real classes? How will you measure the match? Do you encounter any problems?

```
In [6]: print(true_clusters)    # Resnični razredi primerov

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

Question 4-1-5 Test the method in the lower, synthetic data examples. How does the KMeans method work? Why?

```
In [7]: from sklearn.datasets import make_circles, make_blobs, make_moons

        n_samples=1000
        noisy_circles, _ = make_circles(n_samples=n_samples, factor=.5, noise=.05)
        noisy_moons, _ = make_moons(n_samples=n_samples, noise=.05)
        blobs, _ = make_blobs(n_samples=n_samples, random_state=8)


        fig, axes = plt.subplots(nrows=1, ncols=3, figsize=(10, 5))
```

```
axes[0].plot(noisy_circles[:, 0], noisy_circles[:, 1], "r.")
axes[1].plot(noisy_moons[:, 0], noisy_moons[:, 1], "g.")
axes[2].plot(blobs[:, 0], blobs[:, 1], "b.")

fig.tight_layout()
plt.show()
```



4.1.3 DBSCAN method

Question 4-1-6 Test the DBSCAN method. Does this method work better on the same data? Why? You find the answer in the method description.

```
In [8]: from sklearn.cluster import DBSCAN

        # model = DBSCAN(...)
        # model.fit(X)
        # labels = model.predict(X)
```

4.2 Hierarchical clustering

At the lectures we learned the hierarchical clustering algorithm. Its main characteristic is that it allows comparison of objects on the basis of knowledge of distance measures between them. Data representation is therefore not necessarily limited to vector spaces.

The algorithm is deterministic and does not presume the number of clusters. The clustering result will be calculated at once for all possible clustered numbers in the $[1, n]$ interval, and deciding on a number will taken place after the calculation.

Think. What is the time complexity of the algorithm for hierarchical clustering? How does it compare with the K-means method?

```
In [1]: import numpy as np

        %matplotlib inline
        %config InlineBackend.figure_formats = ['jpg']
        import matplotlib
```

```
matplotlib.figure.Figure.__repr__ = lambda self: (
    f"<{self.__class__.__name__} size {self.bbox.size[0]:g}"
    f"x{self.bbox.size[1]:g} with {len(self.axes)} Axes>")

import matplotlib.pyplot as plt
plt.style.use('PR.mplstyle')

import Orange
import scipy.cluster.hierarchy as sch
import scipy
```

```
    ---------------------------------------------------------------------------

    FileNotFoundError                         Traceback (most recent call last)

    ~/anaconda3/lib/python3.6/site-packages/matplotlib/style/core.py in use(style)
    112             try:
--> 113                 rc = rc_params_from_file(style, use_default_template=False)
    114                 _apply_style(rc)


    ~/anaconda3/lib/python3.6/site-packages/matplotlib/__init__.py in rc_params_from_file(fname, fa
   1028     """
-> 1029     config_from_file = _rc_params_in_file(fname, fail_on_error)
   1030


    ~/anaconda3/lib/python3.6/site-packages/matplotlib/__init__.py in _rc_params_in_file(fname, fai
    944     rc_temp = {}
--> 945     with _open_file_or_url(fname) as fd:
    946         try:


    ~/anaconda3/lib/python3.6/contextlib.py in __enter__(self)
     80         try:
---> 81             return next(self.gen)
     82         except StopIteration:


    ~/anaconda3/lib/python3.6/site-packages/matplotlib/__init__.py in _open_file_or_url(fname)
    929             encoding = "utf-8"
--> 930         with io.open(fname, encoding=encoding) as f:
    931             yield f


    FileNotFoundError: [Errno 2] No such file or directory: 'PR.mplstyle'


During handling of the above exception, another exception occurred:


    OSError                                   Traceback (most recent call last)
```

```
<ipython-input-1-a65c3d83fbbe> in <module>
      9
     10 import matplotlib.pyplot as plt
---> 11 plt.style.use('PR.mplstyle')
     12
     13 import Orange


~/anaconda3/lib/python3.6/site-packages/matplotlib/style/core.py in use(style)
    117                         "not a valid URL or path. See `style.available` for "
    118                         "list of available styles.")
--> 119                 raise IOError(msg % style)
    120
    121


OSError: 'PR.mplstyle' not found in the style library and input is not a valid URL or path. See
```

4.2.1 Data

Today's data is reminiscent of (well-known to older generations) an album of animal pictures. It contains 59 animal species and 16 attributes that describe the associated anatomical characteristics. Animals are divided into 7 classes.

```
In [2]: data = Orange.data.Table('podatki/zoo.tab')
        print(data.domain)
        print(data[:2])

        print(np.linalg.norm(data.X[0]-data.X[1]))
        print(np.linalg.norm(data.X[0]-data.X[1], ord=1))


        ---------------------------------------------------------------------------

        NameError                                 Traceback (most recent call last)

        <ipython-input-2-09d556922ece> in <module>
    ----> 1 data = Orange.data.Table('podatki/zoo.tab')
          2 print(data.domain)
          3 print(data[:2])
          4
          5 print(np.linalg.norm(data.X[0]-data.X[1]))


        NameError: name 'Orange' is not defined
```

We are especially interested in the matrix `X`, which stores the data in numerical form.

```
In [3]: X = data.X
        Y = data.Y
        print(X.shape)
        print(X)
```

```
---------------------------------------------------------------------------

NameError                                 Traceback (most recent call last)

<ipython-input-3-8fe55c01bd25> in <module>
----> 1 X = data.X
      2 Y = data.Y
      3 print(X.shape)
      4 print(X)


NameError: name 'data' is not defined
```

The clustering result is obtained using the `scipy.cluster.hierarchy` module and the `linkage` method. The latter calculates the connection in the tree (dendrogram) with respect to a given distance measure (`metric`) and the method of measuring the distance between clusters (`method`).

```
In [4]: L = sch.linkage(X, method="average", metric="cityblock")
```

```
---------------------------------------------------------------------------

NameError                                 Traceback (most recent call last)

<ipython-input-4-f9b3263f20b0> in <module>
----> 1 L = sch.linkage(X, method="average", metric="cityblock")


NameError: name 'sch' is not defined
```

Using the `dendrogram` function, we draw a tree and organize it with labels. The function works in conjunction with the already known `matplotlib` library.

```
In [5]: plt.figure(figsize=(25, 6))
        labels = [row["name"].value for row in data]
        D      = sch.dendrogram(L, labels=labels, leaf_font_size=15)
        plt.ylabel("Razdalja")
        plt.show()
```

```
---------------------------------------------------------------------------

NameError                                 Traceback (most recent call last)

<ipython-input-5-08bf720563cc> in <module>
      1 plt.figure(figsize=(25, 6))
----> 2 labels = [row["name"].value for row in data]
      3 D      = sch.dendrogram(L, labels=labels, leaf_font_size=15)
      4 plt.ylabel("Razdalja")
      5 plt.show()


NameError: name 'data' is not defined
```

```
<Figure size 1800x432 with 0 Axes>
```

Okay, for the first try. Nevertheless, the dendrogram seems somewhat flat. Verify how the graph affects various …

4.2.2 Linkage methods

Linkage methods determine how to calculate the distance between two arbitrarily large points points.

- Single linkage (method = "single"); The distance between the clusters is the distance between the closest points of the clusters.
- Average linkage (method = "average"); The average distance between all pairs of points.
- Centroid linkage (method = "centroid"); Calculates the centers of clusters in the space and their mutual distance. The distance measure is necessarily Euclidean.

Question 4-2-1 Test various forms of the dendrogram with respect to the selected distance measure.

```
In [6]: # Preizkusi različne načine merjenja razdalje med gručami
        L = sch.linkage(X, method="single",)
```

```
        ---------------------------------------------------------------------------

        NameError                                 Traceback (most recent call last)

        <ipython-input-6-d282c08e9e82> in <module>
          1 # Preizkusi različne načine merjenja razdalje med gručami
    ----> 2 L = sch.linkage(X, method="single",)


        NameError: name 'sch' is not defined
```

Answer

Is the Euclidean distance really the best way to compare attributes that are discrete? Not always.

4.2.3 Distance measures

The method of determining the interpretation of the distance between the points $\vec{x} = (x_1, x_2, ..., x_p)$ and $\vec{y} = (y_1, y_2, ..., y_p)$ affects the result of hierarchical clustering. Choosing the right size depends on the nature of the data and answers the question as much as possible: what does it mean when two examples are similar?

The choice of the appropriate measure may be affected by:

- The presence of missing values
- Data presentation (vectors, character strings, images, …)
- Attributes type and interpretation of values

Some common distance measures: * Euclidean distance (metric = "euclidean")

$$d(\vec{x}, \vec{y}) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + ... + (x_p - y_p)^2}$$

- Manhattan distance (metric = "cityblock")

$$d(\vec{x},\vec{y}) = |x_1 - y_1| + |x_2 - y_2| + ... + |x_p - y_p|$$

- Cosine distance (metric = "cosine")

Represents the cosine of the angle between the $\vec{x}$ and $\vec{y}$ vectors - smaller angle means greater similarity. Useful for comparing the similarities between vectors, disregarding their absolute size.

$$d(\vec{x},\vec{y}) = 1 - \frac{\vec{x} \cdot \vec{y}}{\|\vec{x}\|\|\vec{y}\|}$$

- Jaccard index (metric = "jaccard")

Measures the proportion of matches between (x_i, y_i) pairs at the same positions, where at least one of the values x_i or y_i is greater than zero. Suitable for use in cases where we are dealing with missing values or discrete attributes.

Find the complete list of distances in documentation.

Think. Try to remember the type of data where it would be sensible to use each single measure.

4.2.4 Determining the number of clusters

How many clusters are in the data? It is difficult to answer this question and it is also considered an open question in the field of machine learning. Nevertheless, we know some of the indicators that we roughly divide on * supervised (true data classes are known) * unsupervised (only the characteristics and/or distance between the examples are known)

To determine the belonging of examples to clusters, we use the `fcluster` function. The latter receives the `t` parameter, which determines the distance at which we cut the dendrogram, i.e. remove all links that are longer than a given length. The remaining related components of the dendrogram graph thus form groups.

```
In [7]: L = sch.linkage(X, method="average", metric="cityblock")
        t = 3.5
        predictions = sch.fcluster(L, t=t, criterion="distance").ravel()
        classes     = data.Y.ravel()   # resnicni razredi

        print("Primer", "Resnični razred", "Gruča")
        for row, category, prediction in list(zip(data, classes, predictions ))[2:10]:
                print("%s\t%d\t%d" % (row["name"], category, prediction))
```

```
        ---------------------------------------------------------------------------

        NameError                                 Traceback (most recent call last)

        <ipython-input-7-fcc7194cb2fb> in <module>
    ----> 1 L = sch.linkage(X, method="average", metric="cityblock")
          2 t = 3.5
          3 predictions = sch.fcluster(L, t=t, criterion="distance").ravel()
          4 classes     = data.Y.ravel()   # resnicni razredi
          5


        NameError: name 'sch' is not defined
```

Re-draw the dendrogram and cut it at the given distance. Below we'll see a number of estimates of clustering performance.

```
In [8]: D = sch.dendrogram(L, labels=labels)
        plt.plot([0, 1000], [t, t], "k--")
        plt.ylabel("Razdalje")
        plt.show()
```

```
        ---------------------------------------------------------------------------

        NameError                                 Traceback (most recent call last)

        <ipython-input-8-1311667edc45> in <module>
    ----> 1 D = sch.dendrogram(L, labels=labels)
          2 plt.plot([0, 1000], [t, t], "k--")
          3 plt.ylabel("Razdalje")
          4 plt.show()


        NameError: name 'sch' is not defined
```

4.2.4.1 Common shared information

The measure of common shared information is useful when information on the real classes, which includes examples, is available. It is not unreasonable to emphasize that real classes should not be used in case sharing.

Random assignments of cluster labels have the value of the total shared information close to `0.0` for each value of the number of groups and number of cases. The complete clustering of clusters with existing classes has a value of `1`. The measure does not depend on the presentation of data, i.e. it is not necessary to have data in the vector space, as it depends only on labels.

```
In [9]: from sklearn.metrics import adjusted_mutual_info_score
        score = adjusted_mutual_info_score(classes, predictions)
        score
```

```
        ---------------------------------------------------------------------------

        NameError                                 Traceback (most recent call last)

        <ipython-input-9-12bf9055f1a0> in <module>
          1 from sklearn.metrics import adjusted_mutual_info_score
    ----> 2 score = adjusted_mutual_info_score(classes, predictions)
          3 score


        NameError: name 'classes' is not defined
```

4.2.4.2 Silhouette coefficient

The silhouette coefficient is an unsupervised measure in the area between `-1` (wrongly assigned groups) and`1` (very dense, well-separated groups). The greater internal density within the groups and the greater the distance are proportional with the coefficient. Even this measure does not assume that the data is in the vector space, but it depends on the selected distance.

Question 4-2-2 Check how the rating varies according to the selected distance measure. Which measure of distance best estimates clusters? Is the result meaningful?

```
In [10]: from sklearn.metrics import silhouette_score

         score = silhouette_score(X, predictions, metric="cityblock")
         score
```

```
        ---------------------------------------------------------------------------

        NameError                                 Traceback (most recent call last)

        <ipython-input-10-6cdd011a4efa> in <module>
          1 from sklearn.metrics import silhouette_score
          2
    ----> 3 score = silhouette_score(X, predictions, metric="cityblock")
          4 score


        NameError: name 'X' is not defined
```

Answer

Question 4-2-3 Perform a clustering analysis on animal data by selecting the appropriate linking method, distance measure, and number of clusters. Use one of the similarity measures presented and find a combination of the above settings so that the clustering result is as high as possible.

```
In [11]: # ...
```

Answer

4.3 Example: genomic data

The degree of development in the field of biotechnology allows for the acquisition of substantially more data on organisms. One of the common data types we compare genes are genetic records. They are ready for presentation in computing, since they can be generalized to successive four nucleotides: A, C, G, T. The entire gene that determines everything from your eye color to the tendency to certain illnesses is given with something more than 3×10^{12} in the long sequence DNA.

When decomposing, the transcription and combining of DNA records of parents occurs. Of course, this process is not complete, so there are errors - mutacij. The long-term consequence of mutations is a tale of different animal species, which means that more related species have more similar genetic records.

From the genetic database, we have ordered a series of mitochondrial genes for 13 species: `'Gorilla gorilla'`, `'Homo sapiens'`, `'Carassius auratus auratus'`, `'Delphinus capensis'`, 'Chamaeleo calyptratus', 'Canis lupus familiaris', 'Homo sapiens neanderthalensis', 'Rattus norvegicus', 'Equus caballus', 'Daboia russellii', 'Pan troglodytes', 'Takifugu rubripes', 'Pongo abelii', 'Sus scrofa'.

First we obtain the data from the internet.

```
In [1]: from Bio import Entrez
        from Bio import SeqIO
        import json

        species = [
            ("Homo sapiens",              "NC_012920.1"),
            ("Pan troglodytes",           "NC_001643.1"),
            ("Equus caballus",            "NC_001640.1"),
            ("Chamaeleo calyptratus",     "NC_012420.1"),
            ("Delphinus capensis",        "NC_012061.1"),
            ("Takifugu rubripes",         "NC_004299.1"),
            ("Canis lupus familiaris",    "NC_002008.4"),
            ("Gorilla gorilla",           "NC_001645.1"),
            ("Pongo abelii",              "NC_002083.1"),
            ("Sus scrofa",                "NC_000845.1"),
            ("Daboia russellii",          "NC_011391.1"),
            ("Carassius auratus auratus", "NC_006580.1"),
            ("Rattus norvegicus",         "AC_000022.2"),
            ("Homo sapiens neanderthalensis", "NC_011137.1"),
        ]

        # Data loading
        infile = "podatki/seqs.json"
        seqs = dict()
        for name, sid in species:
            print("Loading ...", name)
            t = False
            while not t:
                try:
                    handle = Entrez.efetch(db="nucleotide", rettype="gb", id=sid,
                                           email="a@gmail.com")
                    rec = SeqIO.read(handle, "gb")
                    handle.close()
                    t = True
                except:
                    continue
            seqs[name] = str(rec.seq)

        json.dump(seqs, open(infile, "w"))
```

```
Loading … Homo sapiens
Loading … Pan troglodytes
Loading … Equus caballus
Loading … Chamaeleo calyptratus
Loading … Delphinus capensis
Loading … Takifugu rubripes
Loading … Canis lupus familiaris
Loading … Gorilla gorilla
Loading … Pongo abelii
```

```
Loading … Sus scrofa
Loading … Daboia russellii
Loading … Carassius auratus auratus
Loading … Rattus norvegicus
Loading … Homo sapiens neanderthalensis
```

```
In [2]: import json
        sequences = json.load(open("podatki/seqs.json"))
        print(sequences["Homo sapiens"])
        print(len(sequences["Homo sapiens"]))
```

```
GATCACAGGTCTATCACCCTATTAACCACTCACGGGAGCTCTCCATGCATTTGGTATTTTCGTCTGGGGGGTATGCACGCGATAGCATTGCGAGACGCTGGAG
16569
```

Question 4-3-1 How could you compare animal species with respect to the records that are given as character strings? The first idea is to convert the data into a vector space in which we calculate distances. Tip: You can break sequences into smaller parts and count the number of occurrences of individual characters, pairs, triples, … k-tuples. You also take into account position in sequence.

Complete and help with the `seq_to_kmer_count` function, which converts the string into a vector of the number of occurrences of all possible k-tuples.

Translate the data into the appropriate format, perform hierarchical clustering and display results. Are the species on the dendrogram meaningful? You need to get a picture:

```
In [3]: from itertools import product
        def seq_to_kmer_count(seq, k=4):
            """
            Pretvori zaporedje seq v vektor x.
                 AAAA AAAC AAAG AAAT ... TTTG TTTT
            x = [   1   1      2   10 ...   12    7]
            len(x) == len(seq) - k + 1
            """
            ktuples = list(zip(*[seq[i:] for i in range(k)]))      # razbijemo trenutni niz seq na k-ter
            kmers   = list(product(*(k*[["A", "C", "T", "G"]])))  # vse mozne k-terke

            x = np.zeros((len(kmers), ))
            ### Your code here ###
            # for i, kmer in enumerate(kmers)
            # ...

            return x
```

```
In [4]: # ...
        # Za vsako zaporedje (organizem), izračunaj x
        # X = np.array([x1, x2, ..., x13]) - matrika 13 x 256 (k=4)
        # pozeni gručenje
```

Answer

Chapter 5

Supervised learning

The scenario for *controlled modeling* methods is often the following. The data is presented with couples

$$(\vec{x}_1, y_1), (\vec{x}_2, y_2), ...(\vec{x}_n, y_n)$$

where $\vec{x}_i$ is called *independent*, and y_i *dependent* variables. We are interested in the *mapping* $h(\vec{x})$, which maps the values of the independent variable to the dependent, with the error ϵ_i. So,

$$y_i = h(\vec{x}_i) + \epsilon_i$$

The variables $\vec{x}_i$, y can generally be continuous, discrete and others. The $h(\vec{x})$ mapping represents the *model* of the data. The mapping can be any arbitrary mathematical function (or also an algorithm, a program) that depends on one or more *parameters*.

Machine learning is often regarded as a search for parameters (or of the function itself) so that the error ϵ_i is as small as possible.

5.1 Linear regression

Linear regression is an example of a simple model where we assume: * both dependent as independent variables are real numbers * the dependent variable is a linear combination of independent ones * The ϵ error is normally distributed with the hope of $\mu_\epsilon = 0$ and unknown variance

The dependent variables are in general vectors in the p-dimensional space of real numbers, $\vec{x} = (x_1, x_2, ...x_p)$.

The model is of form

$$h(\vec{x}) = \beta_1 x_1 + \beta_2 x_2 + ... + \beta_p x_p + \beta_0$$

where the vector $\vec{\beta} = (\beta_0, \beta_1, ...\beta_p)$ represents unknown parameters or coefficients. The model is therefore a line (for $p = 1$) or a plane in p-dimensional space.

Learning is the search (optimization) of parameters $\vec{\beta}$ with the aim of reducing the average error in the data.

$$\min_\beta \frac{1}{n} \sum_1^n (y_i - h(\vec{x}_i))^2 = \frac{1}{n} \sum_1^n \epsilon^2$$

The value of the above term is called **the mean square error** (or MSE). From a statistical point of view it represents the **unexplained variance**.

At this time, we will not derive algorithms for minimizing the above expression, but rather focus on practical use. More information is available here.

```
In [1]: %matplotlib inline
        %config InlineBackend.figure_formats = ['jpg']
        import matplotlib
        matplotlib.figure.Figure.__repr__ = lambda self: (
            f"<{self.__class__.__name__} size {self.bbox.size[0]:g}"
            f"x{self.bbox.size[1]:g} with {len(self.axes)} Axes>")

        import matplotlib.pyplot as plt
        plt.style.use('PR.mplstyle')

        import numpy as np
        from sklearn.linear_model import LinearRegression, Lasso, Ridge
        from sklearn.metrics import mean_squared_error


        ---------------------------------------------------------------------------

        FileNotFoundError                         Traceback (most recent call last)

        ~/anaconda3/lib/python3.6/site-packages/matplotlib/style/core.py in use(style)
        112             try:
    --> 113                 rc = rc_params_from_file(style, use_default_template=False)
        114                 _apply_style(rc)


        ~/anaconda3/lib/python3.6/site-packages/matplotlib/__init__.py in rc_params_from_file(fname, fa
       1028     """
    -> 1029     config_from_file = _rc_params_in_file(fname, fail_on_error)
       1030


        ~/anaconda3/lib/python3.6/site-packages/matplotlib/__init__.py in _rc_params_in_file(fname, fail
        944     rc_temp = {}
    --> 945     with _open_file_or_url(fname) as fd:
        946         try:


        ~/anaconda3/lib/python3.6/contextlib.py in __enter__(self)
         80         try:
    ---> 81             return next(self.gen)
         82         except StopIteration:


        ~/anaconda3/lib/python3.6/site-packages/matplotlib/__init__.py in _open_file_or_url(fname)
        929             encoding = "utf-8"
    --> 930         with io.open(fname, encoding=encoding) as f:
        931             yield f
```

```
    FileNotFoundError: [Errno 2] No such file or directory: 'PR.mplstyle'


During handling of the above exception, another exception occurred:


    OSError                                   Traceback (most recent call last)

    <ipython-input-1-e5cae4941953> in <module>
      7
      8 import matplotlib.pyplot as plt
----> 9 plt.style.use('PR.mplstyle')
     10
     11 import numpy as np


    ~/anaconda3/lib/python3.6/site-packages/matplotlib/style/core.py in use(style)
    117                         "not a valid URL or path. See `style.available` for "
    118                         "list of available styles.")
--> 119                 raise IOError(msg % style)
    120
    121


    OSError: 'PR.mplstyle' not found in the style library and input is not a valid URL or path. See
```

Let's start with a simple example with one independent variable x and a dependent variable y.

```
In [2]: data = np.loadtxt("podatki/sintetični/data_A.txt")
        x    = data[:, [0]]
        y    = -data[:, [1]]

        print(x.shape)
        print(y.shape)

        plt.figure()
        plt.plot(x, y, "k.")
        plt.xlabel("x")
        plt.ylabel("y")
        plt.show()


        ---------------------------------------------------------------------------

        NameError                                 Traceback (most recent call last)

        <ipython-input-2-aed59bb829e1> in <module>
    ----> 1 data = np.loadtxt("podatki/sintetični/data_A.txt")
          2 x    = data[:, [0]]
          3 y    = -data[:, [1]]
          4
          5 print(x.shape)
```

```
NameError: name 'np' is not defined
```

The data strongly resembles a line.

Let's try to find a linear model that will reduce the mean square error.

In the left image, we display the values of a model for all values x on a given interval.

The right picture shows the value of *remainders* $y_i - h(\vec{x}_i)$. The better the model fits the data, the less connected the dependent variable and the remainder will be.

```
In [3]: from scipy.stats import pearsonr
        def plot_fit_residual(x, y, yp):

            # Model
            fig, axes  = plt.subplots(nrows=1, ncols=2, figsize=(15, 4))
            axes[0].plot(x.ravel(), y.ravel(), "k.",  label="Podatki")
            axes[0].plot(x.ravel(), yp.ravel(), "g-", label="Model h(x)")
            axes[0].set_xlabel("x")
            axes[0].set_ylabel("y")
            axes[0].legend(loc=4)

            # Ostanki
            r = pearsonr(y.ravel(), y.ravel()-yp.ravel())[0]
            axes[1].plot(y.ravel(), y.ravel()-yp.ravel(), "k.", label="Ostanek")
            axes[1].set_xlabel("y")
            axes[1].set_ylabel("y-h(x)")
            axes[1].set_title("Graf ostankov, R=%.3f" % r)
            axes[1].legend(loc=4)
            plt.show()
```

```
In [4]: # Ucenje modela
        model = LinearRegression()
        model.fit(x, y)

        print(model.intercept_,model.coef_)


        # Napoved vrednosti za podatke
        hx = model.predict(x)

        plot_fit_residual(x, y, hx)
```

```
        ---------------------------------------------------------------------------

        NameError                                 Traceback (most recent call last)

        <ipython-input-4-2f0316d70b18> in <module>
          1 # Ucenje modela
    ----> 2 model = LinearRegression()
          3 model.fit(x, y)
          4
          5 print(model.intercept_,model.coef_)
```

```
NameError: name 'LinearRegression' is not defined
```

Let's measure the mean square error ...

```
In [5]: mean_squared_error(hx, y)
```

```
        ---------------------------------------------------------------------------

        NameError                                 Traceback (most recent call last)

        <ipython-input-5-91031a8d16e9> in <module>
    ----> 1 mean_squared_error(hx, y)


        NameError: name 'mean_squared_error' is not defined
```

... which is equal to variance of difference.

```
In [6]: np.var(hx-y)
```

```
        ---------------------------------------------------------------------------

        NameError                                 Traceback (most recent call last)

        <ipython-input-6-320614238b73> in <module>
    ----> 1 np.var(hx-y)


        NameError: name 'np' is not defined
```

So we can get the *proportion of the explained variance*. The proportion in percent is easier to interpret intuitively.

```
In [7]: explained_var = 100.0 * ( np.var(y) - np.var(hx-y) ) /  np.var(y)
        print("Explained variance: %.2f " % explained_var + "%" )
```

```
        ---------------------------------------------------------------------------

        NameError                                 Traceback (most recent call last)

        <ipython-input-7-d7f1e86fae41> in <module>
    ----> 1 explained_var = 100.0 * ( np.var(y) - np.var(hx-y) ) /  np.var(y)
          2 print("Explained variance: %.2f " % explained_var + "%" )


        NameError: name 'np' is not defined
```

5.2 Polynomial regression

Let's look at the next motivational example.

```
In [8]: data = np.loadtxt("podatki/sintetični/data_B.txt")
        x    = data[:, [0]]
        y    = data[:, [1]]

        plt.figure()
        plt.plot(x, y, "k.")
        plt.show()
```

```
---------------------------------------------------------------------------

NameError                                 Traceback (most recent call last)

<ipython-input-8-4ceb80ce9610> in <module>
----> 1 data = np.loadtxt("podatki/sintetični/data_B.txt")
      2 x    = data[:, [0]]
      3 y    = data[:, [1]]
      4
      5 plt.figure()


NameError: name 'np' is not defined
```

Already at first glance it is clear that the line model will not be enough. If we pull the line through the data, we see that in some places, the data is erroneous. This is also seen on the residual graph, since the error obviously depends on the size of y, which we do not want.

```
In [9]: model = LinearRegression()
        model.fit(x, y)
        hx = model.predict(x)

        plot_fit_residual(x, y, hx)
```

```
---------------------------------------------------------------------------

NameError                                 Traceback (most recent call last)

<ipython-input-9-468e2b1f487e> in <module>
----> 1 model = LinearRegression()
      2 model.fit(x, y)
      3 hx = model.predict(x)
      4
      5 plot_fit_residual(x, y, hx)


NameError: name 'LinearRegression' is not defined
```

```
In [10]: explained_var = 100.0 * ( np.var(y) - np.var(hx-y) ) /  np.var(y)
         print("Explained variance: %.2f " % explained_var + "%" )
```

```
---------------------------------------------------------------------------

NameError                                 Traceback (most recent call last)
```

```
    <ipython-input-10-d7f1e86fae41> in <module>
----> 1 explained_var = 100.0 * ( np.var(y) - np.var(hx-y) ) /  np.var(y)
      2 print("Explained variance: %.2f " % explained_var + "%" )


    NameError: name 'np' is not defined
```

5.3 Polynomial regression model

Linear models can also model non-linear dependencies, which is somewhat surprising given the initial assumptions. The x value in this case is a one-dimensional variable (p=1).

Polynomial regression model in one dimension is a polynomial of degree D:

$$h(\vec{x}) = \beta_1 x + \beta_2 x^2 + ... + \beta_D x^D + \beta_0$$

The effect is achieved by appropriately arranging the space. The variable x is mapped into a vector by calculating the corresponding potencies:

$$x \to (x, x^2, x^3, ...x^D) = \vec{x}$$

In such an assembly, there is nothing more than a linear mapping!

```
In [11]: # Iz 1-D sestavimo nov 2-D prostor
         X = np.zeros((len(x), 2))
         X[:, 0] = x.ravel()
         X[:, 1] = x.ravel()**2

         # Učenje
         model = LinearRegression()
         model.fit(X, y)

         # Napoved
         hx = model.predict(X)

         plot_fit_residual(x, y, hx)
```

```
    ---------------------------------------------------------------------------

    NameError                                 Traceback (most recent call last)

    <ipython-input-11-18e49aed2024> in <module>
      1 # Iz 1-D sestavimo nov 2-D prostor
----> 2 X = np.zeros((len(x), 2))
      3 X[:, 0] = x.ravel()
      4 X[:, 1] = x.ravel()**2
      5
```

```
NameError: name 'np' is not defined
```

Question 5-1-1 Compare the explained variance of the linear and polynomial model.

```
In [12]: # ...
```

Answer

5.4 Overfitting

Of course, we often do not know the optimal model. The use of excessively complex models (complexity can be represented as the size of a family of functions) can lead to **overfitting**.

Let's look at the example of a 20 degree polynomial:

```
In [13]: def plot_coefficients(coef):
             coef=coef.ravel()
             D = len(coef)
             plt.title("Parametri modela")
             plt.bar(np.arange(D), coef)
             plt.xticks(np.arange(D))
             plt.grid()
             plt.ylabel("beta")
             plt.xlabel("d")
             plt.show()
```

```
In [14]: D = 20 # stopnja polinoma
         X = np.zeros((len(x), D))
         for d in range(0, D):
             X[:, d] = x.ravel()**d

         model = LinearRegression()
         model.fit(X, y)

         hx = model.predict(X)

         plot_fit_residual(X[:, 1], y, hx)
         plot_coefficients(model.coef_)
```

```
---------------------------------------------------------------------------

NameError                                 Traceback (most recent call last)

<ipython-input-14-1e1e1a76f177> in <module>
      1 D = 20 # stopnja polinoma
----> 2 X = np.zeros((len(x), D))
      3 for d in range(0, D):
      4     X[:, d] = x.ravel()**d
      5


NameError: name 'np' is not defined
```

The model seems to fit the data perfectly. The graph of the remains also shows a stimulating picture. The problem of over-fitting occurs when **predicting new data**.

Question 5-1-2 Measure the explained variance of the polynomial model.

```
In [15]: # ...
```

Answer

5.5 Solution: punishing excessively complex models

In addition to minimizing the mean square error, we can also penalize the *complexity of the models* when looking for a solution. Therefore, we want the parameters found in the geometric sense to be as small as possible. This procedure is also known as regularization. The degree of regularization is monitored by the parameter α, which is defined by the users. The two most common models are: * Regression Lasso

"Punishment of the Manhattan distance of the vector $\vec{\beta}$ from the baseline"

$$\min_{\beta} \sum_{1}^{n} (y_i - h(\vec{x}_i))^2 + \alpha \|\vec{\beta}\|_1$$

Pro: returns **sparse** parameter vectors $\vec{\beta}$. Most of the components β_j will be 0 - VERY DESIRABLE!

Con: complex planning of optimization algorithms

- Regression Ridge *"Penalizing the eclidic distance of the vector $\vec{\beta}$ from the starting point"*

$$\min_{\beta} \sum_{1}^{n} (y_i - h(\vec{x}_i))^2 + \alpha \|\vec{\beta}\|_2$$

Pro: Easy calculation

Con: Generally does not return rare parameter values.

```
In [16]: D = 20 # stopnja polinoma

         # Ustvarimo ustrezen prostor
         X = np.zeros((len(x), D))
         for d in range(0, D):
             X[:, d] = x.ravel()**d

         model = Lasso(alpha=0.1)
         model.fit(X, y)

         hx = model.predict(X)

         plot_fit_residual(X[:, 1], y, hx)
         plot_coefficients(model.coef_)
         model.coef_
```

```
        ---------------------------------------------------------------------------

        NameError                                 Traceback (most recent call last)

        <ipython-input-16-ac4ca0212785> in <module>
```

```
      2
      3 # Ustvarimo ustrezen prostor
----> 4 X = np.zeros((len(x), D))
      5 for d in range(0, D):
      6     X[:, d] = x.ravel()**d


    NameError: name 'np' is not defined
```

Question 5-1-3 What is the effect of the parameter `alpha` on a) the fitting quality, b) model coefficients? Try to model the data using regression `Ridge`.

```
In [17]: # ...
```

Answer

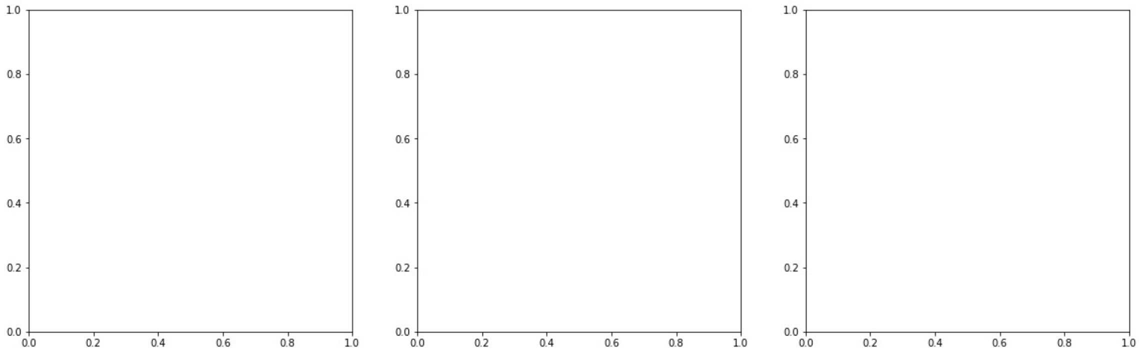
The function looks like “just the right” model fpr the data. On the graph of the coefficients (parameters) we see that the coefficients of lower levels of the polynomial are most of the weight, which is a less complex model.

Question 5-1-4 Find polynomial regression models for the following three sets of data. Choose the degree of the polynom and perhaps the type of the regularization model. Draw graph function and residue diagram. Comment the results.

Correct solutions (you find the coefficients and degree of polynomials in `podatki/sintetični/coefficients_*.txt`

```
In [18]: fig, axes = plt.subplots(nrows=1, ncols=3, figsize=(20, 6))

         for example, ax in zip(["C", "D", "E"], axes):
             data = np.loadtxt("podatki/sintetični/data_%s.txt" % example)
             x    = data[:, [0]]
             y    = data[:, [1]]

             ax.plot(x, y, "k.")
             ax.set_xlabel("x")
             ax.set_ylabel("y")
             ax.set_title("Primer %s" % example)

             # ...


    ---------------------------------------------------------------------------

    NameError                                 Traceback (most recent call last)

    <ipython-input-18-28eb6e4a7e29> in <module>
      2
      3 for example, ax in zip(["C", "D", "E"], axes):
----> 4     data = np.loadtxt("podatki/sintetični/data_%s.txt" % example)
      5     x    = data[:, [0]]
      6     y    = data[:, [1]]
```

```
NameError: name 'np' is not defined
```



Answer

5.6 Use in practice: sentiment analysis

Finally, let's look at a completely practical example of using regression models. There are 1101 book reviews in the database. Each review consists of text (string of characters, words) and ratings between 1 and 5 (1-terrible, 5-excellent). The original database and article are available here.

An example of a positive review of one of the books (rating = 5).

```
I'm a little late in reading this book.  I am trying to pace myself between the movies and the books so
  I think Goblet of Fire is the best in the series, so naturally it would be pretty difficult for Phoeni
  I didn't mind the length of the book, but it did seem to drag in a couple of places.  The gang spent t
  My biggest problem with the book was Dumbledore's secrecy.  Good stories have real roadblocks to keep
  Don't get me wrong.  I love the Harry Potter series.  And, perhaps my expectations have risen too high
```

An example of a negative review of one of the books (rating = 2).

```
This book was horrible.  If it was possible to rate it lower than one star i would have.  I am an avid r

I wish i had the time spent reading this book back so i could use it for better purposes.  This book was
```

We present each review in the space of the 4000 most common words or word pairs in the database (bag-of-words presentation). Each component of the line x (vector) counts how many time a word/pair of words appears in a given review.

```
In [19]: from pickle import load
         from os.path import join

         def load_data(dset):
             data = dict()

             indir = "podatki/%s/" % dset

             for name in "data", "target", "data_test", "target_test":
                 fname = join(indir, name + ".pkl")
                 data[name] = load(open(fname, "rb"))

                 fname = join(indir, "features.txt")
                 fp = open(fname, "rt")
                 data["features"] = list(map(lambda l: l.strip(), fp.readlines()))
```

```
        return data

    books = load_data("books")
    X = books["data"]
    y = books["target"]


    print(str(books['features'][:3]) + '...' + str(books['features'][-3:]))
    print(X.todense())
    print(y)
    print(X.shape, y.shape)
```

```
['the', 'a', 'and']…['colors', 'and_most', 'introduced']
[[ 3  4  0 …,  0  0  0]
 [ 1  1  1 …,  0  0  0]
 [ 0  0  2 …,  0  0  0]
 …,
 [ 4  2  1 …,  0  0  0]
 [10  2  5 …,  0  0  0]
 [ 7  3  3 …,  0  0  0]]
[1 2 2 …, 4 5 2]
(1101, 4000) (1101,)
```

The order of columns in the matrix X:

```
In [20]: features = books["features"]
         features
```

```
Out[20]: ['the',
          'a',
          'and',
          'to',
          'of',
          'this',
          'book',
          'is',
          'in',
          'i',
          'it',
          'for',
          'that',
          'this_book',
          'with',
          'but',
          'on',
          'not',
          'are',
          'have',
          'as',
          'of_the',
          'was',
          'be',
          'you',
          'in_the',
```

```
'an',
'all',
'read',
'from',
'if',
'about',
'one',
'or',
'by',
'at',
'is_a',
'more',
'the_book',
'what',
'very',
'my',
'who',
'so',
'has',
'like',
'some',
'good',
'would',
'his',
'there',
'<num>',
'to_the',
'how',
'they',
'he',
'it_is',
'out',
'just',
'other',
'book_is',
'will',
'much',
'can',
'great',
'this_is',
'do',
'no',
'your',
'when',
'up',
'only',
'which',
'and_the',
'than',
'on_the',
'to_be',
'even',
'many',
'had',
```

```
        'me',
        'get',
        'time',
        'also',
        'well',
        'their',
        "it's",
        'into',
        'for_the',
        'if_you',
        'them',
        'really',
        'first',
        'were',
        'books',
        'most',
        'reading',
        'then',
        'any',
        "don't",
        'been',
        'with_the',
        'author',
        'as_a',
        'because',
        'way',
        'know',
        'in_a',
        'could',
        'these',
        'people',
        'in_this',
        'story',
        'too',
        'life',
        'is_the',
        'little',
        'through',
        'i_was',
        'think',
        'of_a',
        'make',
        'i_have',
        'her',
        'want',
        'its',
        'those',
        'for_a',
        'work',
        'better',
        'written',
        'we',
        'does',
        'that_the',
```

```
'after',
'while',
'to_read',
'own',
'the_author',
'should',
'new',
'is_not',
'such',
'did',
'from_the',
'found',
'she',
'information',
'find',
'best',
'one_of',
'it_was',
'am',
'a_good',
'never',
'at_the',
'there_is',
'being',
'few',
'over',
'interesting',
'why',
'of_this',
'every',
'say',
'years',
'there_are',
'made',
'and_i',
'writing',
'two',
'that_i',
'see',
'real',
'with_a',
'use',
'another',
'a_book',
'a_great',
'i_am',
'have_been',
'however',
'same',
'want_to',
'i_would',
'each',
'all_the',
'world',
```

```
'now',
'still',
'before',
'love',
'go',
'without',
'book_i',
'recommend',
'about_the',
'where',
'the_same',
'may',
'by_the',
'looking',
'give',
'us',
'history',
'something',
'money',
'here',
'was_a',
'down',
'book_and',
'to_get',
'things',
'need',
'the_first',
'a_very',
"i'm",
'characters',
'thought',
'help',
'back',
'as_the',
'actually',
'some_of',
'excellent',
'book_for',
'though',
'our',
'must',
'take',
'ever',
'different',
'be_a',
'and_a',
'you_are',
"doesn't",
'nothing',
'both',
'book_to',
"didn't",
'thing',
'him',
```

```
'the_story',
'have_to',
'understand',
'to_make',
'a_few',
'off',
'pages',
'used',
'look',
'rather',
'lot',
'makes',
'buy',
'seems',
'reader',
'long',
'but_i',
'between',
'book_was',
'is_an',
'to_do',
'how_to',
'man',
'to_a',
'anyone',
'that_is',
'of_his',
'a_lot',
'they_are',
'but_the',
'times',
'someone',
'yet',
'old',
'would_be',
'might',
'far',
'stories',
'but_it',
'again',
'the_best',
'i_found',
'series',
'fact',
'easy',
'have_a',
"i've",
'chapter',
'quite',
'point',
'all_of',
'bought',
'example',
'you_can',
```

```
    'others',
    'out_of',
    'the_most',
    'i_think',
    'least',
    'around',
    'last',
    'true',
    'a_little',
    'bad',
    'style',
    'part',
    'is_that',
    'right',
    'highly',
    'end',
    'believe',
    'come',
    'going',
    'that_it',
    'as_well',
    'always',
    'almost',
    'short',
    'since',
    'i_had',
    "can't",
    'page',
    'hard',
    'would_have',
    'enough',
    'read_the',
    'useful',
    'into_the',
    'said',
    'put',
    'the_way',
    'authors',
    'novel',
    'to_say',
    'the_other',
    'making',
    'readers',
    'having',
    'learn',
    'instead',
    'review',
    'feel',
    'anything',
    'character',
    "i_don't",
    'read_this',
    'bit',
    'should_be',
```

```
'the_only',
'such_as',
'trying',
"that's",
'got',
'through_the',
'you_want',
'once',
'works',
'in_my',
'has_a',
'i_read',
'not_a',
'on_a',
'<year>',
'interested',
'when_i',
'given',
'the_reader',
'does_not',
'this_one',
'idea',
'worth',
'book_that',
'looking_for',
'lot_of',
'already',
'high',
'is_very',
'course',
'place',
'done',
'trying_to',
'person',
'along',
'more_than',
'reason',
'left',
'difficult',
'which_is',
'words',
'takes',
'mind',
'american',
'big',
'you_have',
'and_then',
'especially',
'often',
'in_his',
'ideas',
'that_are',
'experience',
'those_who',
```

```
 'will_be',
 'like_the',
 'keep',
 'comes',
 'away',
 'that_this',
 'at_least',
 'do_not',
 'with_this',
 'perhaps',
 'sure',
 'maybe',
 'try',
 'that_he',
 'simply',
 'case',
 'past',
 'plot',
 'and_his',
 'need_to',
 'tell',
 'wanted',
 'important',
 'start',
 'although',
 'school',
 'sense',
 'to_have',
 'of_all',
 'text',
 'everything',
 'reviews',
 'probably',
 'and_it',
 'human',
 'book_the',
 'problem',
 'children',
 'a_bit',
 'part_of',
 'less',
 'the_world',
 'whole',
 'goes',
 'are_not',
 'it_to',
 'full',
 'did_not',
 'disappointed',
 'knowledge',
 'pretty',
 'getting',
 'and_how',
 'wonderful',
```

```
    'interested_in',
    'next',
    'subject',
    'clear',
    'are_a',
    "you're",
    'second',
    'become',
    'based',
    'but_this',
    'i_thought',
    'young',
    'most_of',
    'home',
    'using',
    'title',
    "it's_a",
    'who_is',
    'could_have',
    'i_bought',
    'to_see',
    'came',
    'interest',
    'poor',
    'was_the',
    'year',
    'word',
    'personal',
    'called',
    'well_as',
    'unfortunately',
    'lives',
    'simple',
    'easy_to',
    'enjoy',
    'over_the',
    'book_but',
    'to_know',
    'examples',
    'like_a',
    'day',
    'sometimes',
    'not_the',
    'that_they',
    'kind',
    'today',
    'the_end',
    'as_i',
    'seem',
    'let',
    "isn't",
    "author's",
    'question',
    'family',
```

```
 'he_is',
 'small',
 'chapters',
 'write',
 'lack',
 'order',
 'it_would',
 'guide',
 'points',
 'to_find',
 'as_it',
 'the_last',
 'can_be',
 'going_to',
 'this_was',
 'is_one',
 'general',
 'i_can',
 'of_them',
 'i_did',
 'what_i',
 'shows',
 'truly',
 'able',
 'it_i',
 'issues',
 'three',
 'truth',
 'business',
 'doing',
 'so_much',
 'enjoyed',
 'the_characters',
 'to_learn',
 'from_a',
 'main',
 'able_to',
 'with_his',
 "there's",
 'god',
 'had_to',
 'etc',
 'gives',
 'is_to',
 'bought_this',
 'study',
 'is_no',
 'may_be',
 'great_book',
 'many_of',
 'problems',
 'that_you',
 'various',
 'are_the',
```

```
'reference',
'provides',
'change',
'class',
'has_been',
'seems_to',
'the_time',
'rather_than',
'an_excellent',
'historical',
'of_what',
'name',
'myself',
'ways',
'good_book',
'follow',
'against',
'disappointing',
'his_own',
'basic',
'i_could',
'loved',
'level',
'approach',
'several',
'beautiful',
'but_not',
'for_those',
'age',
'you_will',
'himself',
'view',
'living',
'number',
'of_their',
'else',
'book_in',
'john',
'of_how',
'job',
'say_that',
'of_her',
'ago',
'practical',
'get_a',
'material',
'possible',
'pictures',
'mr',
'started',
'insight',
'either',
'the_whole',
'stars',
```

```
'together',
'hand',
'students',
'cannot',
'large',
'complete',
'detail',
'themselves',
'says',
'recommended',
'set',
'i_will',
'of_course',
'to_go',
'throughout',
'gave',
'read_it',
'yourself',
'show',
'advice',
'to_use',
'he_has',
'parts',
'needs',
'lack_of',
'to_this',
'expect',
"i_didn't",
'writer',
'friends',
'fan',
'woman',
'liked',
'nice',
'for_you',
'<num>_years',
'present',
'because_i',
'within',
'waste',
'helpful',
'late',
'hope',
'seen',
'as_an',
'book_on',
'modern',
'to_help',
'it_has',
"the_author's",
'questions',
'boring',
'for_example',
'<num>_pages',
```

```
'introduction',
'fun',
'collection',
'hard_to',
'book_as',
'such_a',
'and_that',
'by_a',
'taking',
'extremely',
'because_of',
'much_better',
'to_me',
'tells',
'was_not',
'book_with',
'at_all',
'we_are',
'to_take',
'child',
'look_at',
'during',
'from_this',
'entire',
'people_who',
'for_my',
'way_to',
'reading_this',
'research',
'mostly',
'the_fact',
'kind_of',
'mean',
'understanding',
'covers',
'theory',
'so_many',
'for_me',
'full_of',
'war',
'perfect',
'when_the',
'form',
'lots',
'details',
'wanted_to',
'care',
'in_which',
'everyone',
'book_it',
'clearly',
'power',
'up_with',
'tried',
```

```
'based_on',
'save',
'beyond',
'lost',
'told',
'the_authors',
'up_to',
'and_this',
'history_of',
'social',
'finally',
'certainly',
'because_it',
'you_know',
'friend',
'he_was',
'what_the',
'kids',
'version',
'who_are',
'and_not',
'wrong',
'suggest',
'value',
'wants',
'took',
'number_of',
'quality',
'head',
'until',
'to_give',
'wish',
'gets',
'story_of',
'not_only',
'much_more',
'uses',
'later',
'talking',
'about_a',
'my_own',
"you_don't",
'fine',
'including',
'the_main',
'original',
'under',
'was_very',
'early',
'not_be',
'went',
'of_it',
'time_and',
'certain',
```

```
 'upon',
 'means',
 'culture',
 'writers',
 'someone_who',
 'book_has',
 'college',
 'science',
 'what_you',
 'complex',
 'of_these',
 'facts',
 'lots_of',
 'is_also',
 'section',
 'perspective',
 'the_subject',
 'resource',
 'to_understand',
 'for_an',
 'working',
 'too_much',
 'matter',
 'if_i',
 'so_i',
 'pick',
 'very_good',
 'rest',
 'among',
 'major',
 'christian',
 'fact_that',
 'after_reading',
 'it_and',
 'piece',
 'overall',
 'side',
 'attention',
 'have_the',
 'is_in',
 'ones',
 'it_does',
 'stuff',
 'fascinating',
 'yes',
 'couple',
 'a_new',
 'of_us',
 'their_own',
 'completely',
 'century',
 "won't",
 'could_be',
 'i_know',
```

```
    'novels',
    'writes',
    'line',
    'thinking',
    'have_read',
    'and_her',
    'felt',
    'of_my',
    'topic',
    'the_plot',
    'development',
    'wife',
    'due',
    'at_a',
    'and_to',
    'itself',
    'about_this',
    'for_all',
    'society',
    'the_writing',
    'not_for',
    'expected',
    'there_were',
    'half',
    'white',
    'system',
    'and_was',
    'what_they',
    'practice',
    'volume',
    'analysis',
    'know_what',
    'i_really',
    'for_this',
    'due_to',
    'end_of',
    'they_were',
    'and_he',
    'death',
    'events',
    'strong',
    'recommend_this',
    'ending',
    'reality',
    'black',
    'single',
    'live',
    'process',
    'in_fact',
    'known',
    'huge',
    'is_so',
    'of_our',
    'and_is',
```

```
'whose',
'reviewers',
'days',
'to_his',
'type',
'what_a',
'on_this',
'described',
'the_title',
'saying',
'country',
'as_much',
'what_is',
'the_<num>',
'language',
'and_other',
'cover',
'light',
'to_keep',
'had_a',
'in_order',
'serious',
'instead_of',
'imagine',
'art',
'on_how',
'the_rest',
'explain',
'be_the',
'which_i',
'giving',
'entertaining',
'opinion',
'finished',
'how_the',
"i'd",
'to_buy',
'women',
'much_of',
'the_truth',
'current',
'is_just',
'include',
'despite',
'agree',
'particular',
'read_and',
'than_the',
'four',
'deal',
'i_do',
'men',
'but_in',
'indeed',
```

```
'written_by',
'should_have',
'out_the',
'better_than',
'and_even',
'otherwise',
'guy',
'the_great',
'there_was',
'edition',
'knows',
'it_will',
'what_he',
'respect',
'taken',
'they_have',
'focus',
'fiction',
'needed',
'even_though',
"if_you're",
'sort_of',
'sense_of',
'like_this',
'the_real',
'and_in',
'a_real',
'likely',
'and_what',
'america',
"you'll",
'if_the',
'seemed',
'decided',
'up_the',
'particularly',
'starting',
'like_to',
'source',
'political',
'nor',
'successful',
'try_to',
'a_must',
'get_the',
'available',
'turn',
'library',
'call',
"wasn't",
'nature',
'design',
'the_point',
'worst',
```

```
         'tale',
         'reading_the',
         …]
```

A database of test cases is also included, where we can test the predictive accuracy of the model on new data.

```
In [21]: X_test = books["data_test"]
         y_test = books["target_test"]
```

Question 5-1-5 Use the aforementioned linear models for modeling data in the problem of the sentiment analysis. Measure the mean square error and the explained variance on test cases.

```
In [22]: # ...
```

Answer

Question 5-1-6 Can you find out which phrases have a strong positive and strong negative impact on the final rating of the review? Tip: use the value of the coefficients for each column.

```
In [23]: # ...
```

Answer

5.7 Naive Bayes Classifier

```
In [1]: import numpy as np

        from Orange.data import Table
        from Orange.data.filter import SameValue
```

5.7.1 Warmup example

There are 20 pupils in a year of sports high school. Each of them participates in one of the sports: `basketball`, `football`, `gymnastics`. We evaluated their height "on the eye" and assigned to each pupil one of the possible values: `low`, `average` or `high`.

How would you suggest the most appropriate sport for the new pupil Mark, who is of `average` height?

```
In [2]: data = Table('podatki/sportniki.tab')
```

```
In [3]: data.domain
```

```
Out[3]: [visina | sport]
```

```
In [4]: data
```

```
Out[4]: [[visok | kosarka],
         [visok | kosarka],
         [visok | kosarka],
         [visok | kosarka],
         [srednji | kosarka],
          …
        ]
```

To start, let's look at how popular each sport is:

```
In [5]: for sport in data.domain["sport"].values:
            subset = SameValue(data.domain["sport"], sport)(data)

            print(sport)
            print(subset)
            print()

            py     = len(subset) / len(data)
            print("Sport (Y): %s, število: %d, verjetnost P(Y): %f" % (sport, len(subset), py))
```

```
gimnastika
[[nizek | gimnastika],
 [nizek | gimnastika],
 [nizek | gimnastika],
 [srednji | gimnastika],
 [srednji | gimnastika]]

Sport (Y): gimnastika, število: 5, verjetnost P(Y): 0.250000
kosarka
[[visok | kosarka],
 [visok | kosarka],
 [visok | kosarka],
 [visok | kosarka],
 [srednji | kosarka],
 [srednji | kosarka],
 [nizek | kosarka],
 [visok | kosarka]]

Sport (Y): kosarka, število: 8, verjetnost P(Y): 0.400000
nogomet
[[srednji | nogomet],
 [srednji | nogomet],
 [srednji | nogomet],
 [visok | nogomet],
 [visok | nogomet],
 [nizek | nogomet],
 [nizek | nogomet]]

Sport (Y): nogomet, število: 7, verjetnost P(Y): 0.350000
```

The most popular sport is basketball, with 8 or 40participating in it. Our first suggestion is that Marko should play basketball. This result is not the most satisfying, as we see that among basketball players there are not many athletes of `medium` height. The reason? In calculating, we did not take into account the probability of the property or *attribute* about Mark's height.

The general probabilities of the classes that we calculated are called *a priori* probabilities.

Let us label them with $P(Y)$, where Y is a class variable.

In our example, Y takes on the values {`basketball`, `football`, `gymnastics`}.

```
In [6]: for sport in data.domain["sport"].values:
            subset_y = SameValue(data.domain["sport"],   sport)(data)
            subset_x = SameValue(data.domain["visina"], "srednji")(subset_y)
            p_xy = len(subset_x) / len(subset_y)
```

```
        print("Sport (Y): %s, št. srednje visokih: %d, verjetnost P(X=srednji|Y=%s): %f" % (sport, 
        print(subset_x)
        print()
```

```
Sport (Y): gimnastika, št. srednje visokih: 2, verjetnost P(X=srednji|Y=gimnastika): 0.400000
[[srednji | gimnastika],
 [srednji | gimnastika]]

Sport (Y): kosarka, št. srednje visokih: 2, verjetnost P(X=srednji|Y=kosarka): 0.250000
[[srednji | kosarka],
 [srednji | kosarka]]

Sport (Y): nogomet, št. srednje visokih: 3, verjetnost P(X=srednji|Y=nogomet): 0.428571
[[srednji | nogomet],
 [srednji | nogomet],
 [srednji | nogomet]]
```

Interesting! The likelihood of a “ medium ” height is the highest among footballers. Is the information sufficient to change the original decision?

The probabilities of $P(X|Y)$ are called pogojna verjetnost spremenljivke X pri znanem Y. It determines the probability that in the cases of the Y class the attribute X takes a certain value.

What probability does we really care about? We want the calculation to take Mark's height into account and evaluate the likelihood of each of the sports. That's the probability

$$P(Y|X)$$

or. in Mark's case

$$P(Y|X = srednji)$$

We use this probability to calculate this probability

5.8 Bayes form

In order to calculate the likely class for given attributes of $P(Y|X)$, we need probability for all possible combinations of the class Y and attributes X, which is denoted by $P(X,Y)$. It follows from the rules on conditional probability:

$$P(X,Y) = P(X|Y) \cdot P(Y) = P(Y|X) \cdot P(X)$$

What follows is Bayesov obrazec for calculating $P(Y|X)$:

$$P(Y|X) = \frac{P(X|Y) \cdot P(Y)}{P(X)}$$

The calculation of the probability of the class Y in the known attribute of X is therefore dependent on the a priori probability of the class $P(Y)$, the conditional probability of $P(X|Y)$, and the a priori probability of the attribute of $P(X)$. V In the example of Mark, then:

$$P(Y|X = srednji) = \frac{P(X = srednji|Y) \cdot P(Y)}{P(X = srednji)}$$

If we estimate the probability for each possible value of the class Y, then {`basketball,football,` `` gymnastics, we get the answer to the original question.

```
In [7]: for sport in data.domain["sport"].values:

            subset_y  = SameValue(data.domain["sport"],   sport)(data)        # vsi sportniki danega sp
            subset_x  = SameValue(data.domain["visina"], "srednji")(data)     # vsi srednje visoki ucen

            subset_xy = SameValue(data.domain["visina"], "srednji")(subset_y) # vsi srednje visoki ucen

            # Izracunamo verjetnosti
            p_y  = len(subset_y)  / len(data)
            p_x  = len(subset_x)  / len(data)
            p_xy = len(subset_xy) / len(subset_y)

            p_yx = (p_xy * p_y) / p_x

            print("Sport (Y): %s, napoved P(Y=%s | X=srednji): %f" % (sport, sport, p_yx))

Sport (Y): gimnastika, napoved P(Y=gimnastika | X=srednji): 0.285714
Sport (Y): kosarka, napoved P(Y=kosarka | X=srednji): 0.285714
Sport (Y): nogomet, napoved P(Y=nogomet | X=srednji): 0.428571
```

5.9 Implementation of the Naive Bayes Classifier

The *Naive Bayes classifier* assumes that the attributes are independent of each other, with known class.

$$P(Y|X_1, X_2, ..., X_p) = \frac{P(Y) \cdot P(X_1|Y) \cdot P(X_2|Y) \cdots P(X_p|Y)}{P(X)}$$

Vprašanje 5-2-1 Complete the implementation of the Naive Bayes classifier, which is defined in the lower section. It is necessary to complete the part of the code where we calculate * probability distribution of classes $P(Y)$ * probability distribution of attributes in the known class $P(X|Y)$

5.9.1 Conclusion on data

In the case of discrete attributes, both distributions can be obtained by *counting*. * $P(Y)$ *How many times does the* Y *class appear in the data?* * $P(X|Y)$ *How many times does the* X *attribute appear in the data that belong to the* Y *class?*

What about $P(X)$? This probability is sometimes difficult to calculate, especially for high-dimensional data, since it is not necessary that all combinations of attributes will be present in the data. Fortunately, this value does not affect the choice of the most likely grade for a particular case!

5.9.2 Predicting

For a new example $X^* = (X_1^*, X_2^*, ..., X_p^*)$ among all values of the $Y = y$ class, select one that maximizes the following expression:

$$\arg\max_y \; P(Y = y) \cdot P(X_1^*|Y = y) \cdot P(X_2^*|Y = y) \cdots P(X_p^*|Y = y)$$

5.9.3 Log-transformation

The problem with the above approach is rather practical; multiplying a large number of probabilities quickly leads to very small numbers that can exceed machine accuracy. The simplest solution that leads to the same class choice is the following

$$\arg\max_y \; \log P(Y = y) + \log P(X_1|Y = y) + \log P(X_2|Y = y) + ... + \log P(X_p|Y = y)$$

For the implementation help yourself with the passenger data from Titanic.

First, we divide the data into a learning and test set.

```
In [8]: from Orange.data import Table
        from numpy import random
        random.seed(42)  # zagotovi ponovljivost naključnih rezultatov

        data = Table('titanic')
        inxs = list(range(len(data)))
        n = len(inxs)

        random.shuffle(inxs)

        data_training = data[inxs[:n//2]]
        data_test     = data[inxs[n//2:]]

        data_training.save('podatki/titanic-training.tab')
        data_test.save('podatki/titanic-test.tab')
```

Load the learning data and calculate probabilities.

```
In [9]: data = Table('podatki/titanic-training.tab')
        print(data.domain.class_var)
        print(data.domain.class_var.values)

        # P(X=child | Y = yes)
        filt_child   = SameValue(data.domain["age"], "child")
        filt_survived  = SameValue(data.domain["survived"], "yes")

        p_xy = len(filt_survived(filt_child(data))) / len(filt_survived(data))
        p_xy
```

```
survived
['no', 'yes']
```

```
Out[9]: 0.08379888268156424
```

```
In [10]: class NaiveBayes:
             """
             Naive Bayes classifier.

             :attribute self.probabilities
                 Dictionary that stores
                     - prior class probabilities P(Y)
                     - attribute probabilities conditional on class P(X|Y)

             :attribute self.class_values
                 All possible values of the class.

             :attribute self.variables
                 Variables in the data.

             :attribute self.trained
                 Set to True after fit is called.
             """

             def __init__(self):
                 self.trained       = False
                 self.probabilities = dict()


             def fit(self, data):
                 """
                 Fit a NaiveBayes classifier.

                 :param data
                     Orange data Table.
                 """
                 class_variable      = data.domain.class_var    # class variable (Y)
                 self.class_values   = class_variable.values    # possible class values
                 self.variables      = data.domain.attributes    # all other variables (X)

                 n = len(data) # number of all data points

                 # Compute P(Y)
                 for y in self.class_values:

                     # A not too smart guess (INCORRECT)
                     self.probabilities[y] = 1/len(self.class_values)

                     # <your code here>
                     # Compute class probabilities and correctly fill
                     #   probabilities[y] = ...
                     # Select all examples (rows) with class = y

                     # </your code here>

                 # Compute P(X|Y)
                 for y in self.class_values:

                     # Select all examples (rows) with class = y
```

```
            filty = SameValue(class_variable, y)

            for variable in self.variables:
                for x in variable.values:

                    # A not too smart guess (INCORRECT)
                    p = 1 / (len(self.variables) * len(variable.values) * len(self.class_value

                    # P(variable=x|Y=y)
                    self.probabilities[variable, x, y] = p


                    # <your code here>
                    # Compute correct conditional class probability
                    #   probabilities[x, value, c] = ...
                    #
                    # Select all examples with class == y AND
                    # variable x == value
                    # Hint: use SameValue filter twice


                    # </your code here>

        self.trained = True


    def predict_instance(self, row):
        """
        Predict a class value for one row.

        :param row
            Orange data Instance.
        :return
            Class prediction.
        """
        curr_p = float("-inf")   # Current highest "probability" (unnormalized)
        curr_c = None            # Current most probable class

        for y in self.class_values:
            p = np.log(self.probabilities[y])
            for x in self.variables:
                p = p + np.log(self.probabilities[x, row[x].value, y])

            if p > curr_p:
                curr_p = p
                curr_c = y

        return curr_c, curr_p



    def predict(self, data):
        """
        Predict class labels for all rows in data.
```

```
        :param data
            Orange data Table.
        :return y
            NumPy vector with predicted classes.
        """

        n = len(data)
        predictions = list()
        confidences = np.zeros((n, ))

        for i, row in enumerate(data):
            pred, cf = self.predict_instance(row)
            predictions.append(pred)
            confidences[i] = cf

        return predictions, confidences
```

Rešitev je dostopna na: 205-2.ipynb

```
In [11]: %run '205-2.ipynb'
```

```
ERROR:root:File `'205-2.ipynb.py'` not found.
```

5.10 Using a classifier

An example of use on passenger data from Titanic.

```
In [12]: model = NaiveBayes()
         model.fit(data)
         model.probabilities
```

```
Out[12]: {'no': 0.5,
          'yes': 0.5,
          (DiscreteVariable(name='status', values=['crew', 'first', 'second', 'third']),
           'crew',
           'no'): 0.041666666666666664,
          (DiscreteVariable(name='status', values=['crew', 'first', 'second', 'third']),
           'first',
           'no'): 0.041666666666666664,
          (DiscreteVariable(name='status', values=['crew', 'first', 'second', 'third']),
           'second',
           'no'): 0.041666666666666664,
          (DiscreteVariable(name='status', values=['crew', 'first', 'second', 'third']),
           'third',
           'no'): 0.041666666666666664,
          (DiscreteVariable(name='age', values=['adult', 'child']),
           'adult',
           'no'): 0.08333333333333333,
          (DiscreteVariable(name='age', values=['adult', 'child']),
           'child',
           'no'): 0.08333333333333333,
          (DiscreteVariable(name='sex', values=['female', 'male']),
           'female',
```

```
  'no'): 0.08333333333333333,
 (DiscreteVariable(name='sex', values=['female', 'male']),
  'male',
  'no'): 0.08333333333333333,
 (DiscreteVariable(name='status', values=['crew', 'first', 'second', 'third']),
  'crew',
  'yes'): 0.041666666666666664,
 (DiscreteVariable(name='status', values=['crew', 'first', 'second', 'third']),
  'first',
  'yes'): 0.041666666666666664,
 (DiscreteVariable(name='status', values=['crew', 'first', 'second', 'third']),
  'second',
  'yes'): 0.041666666666666664,
 (DiscreteVariable(name='status', values=['crew', 'first', 'second', 'third']),
  'third',
  'yes'): 0.041666666666666664,
 (DiscreteVariable(name='age', values=['adult', 'child']),
  'adult',
  'yes'): 0.08333333333333333,
 (DiscreteVariable(name='age', values=['adult', 'child']),
  'child',
  'yes'): 0.08333333333333333,
 (DiscreteVariable(name='sex', values=['female', 'male']),
  'female',
  'yes'): 0.08333333333333333,
 (DiscreteVariable(name='sex', values=['female', 'male']),
  'male',
  'yes'): 0.08333333333333333}
```

```
In [13]: predictions, confidences = model.predict(data)

         for row, p, c in zip(data, predictions, confidences):
             print("Row=%s, predicted class=%s confidence=%.5f" % (row, p, c))
```

```
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[second, adult, female | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[second, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[second, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | yes], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[second, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, female | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | yes], predicted class=no confidence=-8.84101
Row=[first, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | yes], predicted class=no confidence=-8.84101
```

```
Row=[first, adult, female | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | yes], predicted class=no confidence=-8.84101
Row=[first, adult, female | yes], predicted class=no confidence=-8.84101
Row=[second, adult, female | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | yes], predicted class=no confidence=-8.84101
Row=[first, adult, male | yes], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[first, adult, female | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[second, adult, female | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, female | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | yes], predicted class=no confidence=-8.84101
Row=[first, adult, female | yes], predicted class=no confidence=-8.84101
Row=[third, adult, female | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | yes], predicted class=no confidence=-8.84101
Row=[first, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, female | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[second, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | yes], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, female | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | yes], predicted class=no confidence=-8.84101
Row=[first, adult, male | no], predicted class=no confidence=-8.84101
Row=[first, adult, female | yes], predicted class=no confidence=-8.84101
Row=[first, adult, male | no], predicted class=no confidence=-8.84101
Row=[second, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, female | no], predicted class=no confidence=-8.84101
Row=[second, adult, male | no], predicted class=no confidence=-8.84101
Row=[second, adult, female | yes], predicted class=no confidence=-8.84101
Row=[third, adult, female | yes], predicted class=no confidence=-8.84101
```

```
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | yes], predicted class=no confidence=-8.84101
Row=[first, adult, female | yes], predicted class=no confidence=-8.84101
Row=[third, adult, female | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[second, adult, female | yes], predicted class=no confidence=-8.84101
Row=[first, adult, male | no], predicted class=no confidence=-8.84101
Row=[first, adult, female | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | yes], predicted class=no confidence=-8.84101
Row=[third, adult, female | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, child, female | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[second, adult, male | no], predicted class=no confidence=-8.84101
Row=[second, adult, male | no], predicted class=no confidence=-8.84101
Row=[second, adult, female | yes], predicted class=no confidence=-8.84101
Row=[first, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, female | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, female | yes], predicted class=no confidence=-8.84101
Row=[second, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | yes], predicted class=no confidence=-8.84101
Row=[first, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | yes], predicted class=no confidence=-8.84101
Row=[third, adult, female | yes], predicted class=no confidence=-8.84101
Row=[first, adult, female | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[second, adult, male | no], predicted class=no confidence=-8.84101
Row=[first, adult, female | yes], predicted class=no confidence=-8.84101
Row=[third, adult, female | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[first, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[first, adult, male | no], predicted class=no confidence=-8.84101
Row=[second, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | yes], predicted class=no confidence=-8.84101
Row=[second, adult, male | no], predicted class=no confidence=-8.84101
Row=[first, adult, female | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
```

```
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, female | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | yes], predicted class=no confidence=-8.84101
Row=[third, adult, female | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[first, adult, female | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | yes], predicted class=no confidence=-8.84101
Row=[third, adult, female | no], predicted class=no confidence=-8.84101
Row=[second, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, female | yes], predicted class=no confidence=-8.84101
Row=[third, adult, male | yes], predicted class=no confidence=-8.84101
Row=[second, adult, male | no], predicted class=no confidence=-8.84101
Row=[first, adult, female | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[second, adult, female | yes], predicted class=no confidence=-8.84101
Row=[third, adult, female | no], predicted class=no confidence=-8.84101
Row=[first, adult, male | no], predicted class=no confidence=-8.84101
Row=[second, adult, female | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[first, adult, female | yes], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | yes], predicted class=no confidence=-8.84101
Row=[third, adult, female | yes], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[second, adult, male | no], predicted class=no confidence=-8.84101
Row=[first, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, female | yes], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[first, adult, male | yes], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[second, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, female | yes], predicted class=no confidence=-8.84101
Row=[second, child, male | yes], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | yes], predicted class=no confidence=-8.84101
Row=[first, adult, female | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | yes], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[first, child, male | yes], predicted class=no confidence=-8.84101
Row=[second, adult, female | yes], predicted class=no confidence=-8.84101
```

```
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | yes], predicted class=no confidence=-8.84101
Row=[first, adult, female | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | yes], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[second, adult, female | yes], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[second, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, female | yes], predicted class=no confidence=-8.84101
Row=[first, adult, female | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | yes], predicted class=no confidence=-8.84101
Row=[third, child, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | yes], predicted class=no confidence=-8.84101
Row=[first, adult, male | yes], predicted class=no confidence=-8.84101
Row=[third, adult, male | yes], predicted class=no confidence=-8.84101
Row=[first, adult, female | yes], predicted class=no confidence=-8.84101
Row=[second, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[first, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[first, adult, female | yes], predicted class=no confidence=-8.84101
Row=[second, adult, female | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[first, adult, male | yes], predicted class=no confidence=-8.84101
Row=[second, adult, female | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[first, adult, female | yes], predicted class=no confidence=-8.84101
Row=[second, adult, female | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, female | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[first, adult, female | yes], predicted class=no confidence=-8.84101
Row=[second, adult, male | no], predicted class=no confidence=-8.84101
Row=[first, adult, female | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, female | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
```

```
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | yes], predicted class=no confidence=-8.84101
Row=[first, adult, male | yes], predicted class=no confidence=-8.84101
Row=[first, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | yes], predicted class=no confidence=-8.84101
Row=[second, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[first, adult, female | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, female | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, female | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[second, adult, male | no], predicted class=no confidence=-8.84101
Row=[second, adult, female | yes], predicted class=no confidence=-8.84101
Row=[first, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[second, adult, male | no], predicted class=no confidence=-8.84101
Row=[second, adult, male | yes], predicted class=no confidence=-8.84101
Row=[first, adult, female | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[second, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[first, child, female | yes], predicted class=no confidence=-8.84101
Row=[second, adult, female | no], predicted class=no confidence=-8.84101
Row=[first, adult, female | yes], predicted class=no confidence=-8.84101
Row=[first, adult, male | no], predicted class=no confidence=-8.84101
Row=[second, adult, male | no], predicted class=no confidence=-8.84101
Row=[second, adult, female | yes], predicted class=no confidence=-8.84101
Row=[second, child, female | yes], predicted class=no confidence=-8.84101
Row=[third, adult, male | yes], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[first, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[first, adult, female | yes], predicted class=no confidence=-8.84101
Row=[first, adult, female | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[first, adult, male | no], predicted class=no confidence=-8.84101
Row=[second, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[first, adult, female | yes], predicted class=no confidence=-8.84101
```

```
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, female | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[first, adult, female | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[second, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[second, adult, female | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[first, adult, female | yes], predicted class=no confidence=-8.84101
Row=[second, adult, female | yes], predicted class=no confidence=-8.84101
Row=[first, adult, female | yes], predicted class=no confidence=-8.84101
Row=[first, adult, female | yes], predicted class=no confidence=-8.84101
Row=[second, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[first, adult, female | yes], predicted class=no confidence=-8.84101
Row=[third, adult, female | no], predicted class=no confidence=-8.84101
Row=[first, adult, female | yes], predicted class=no confidence=-8.84101
Row=[third, adult, female | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[second, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[second, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[first, adult, male | yes], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[first, adult, female | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[second, adult, female | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | yes], predicted class=no confidence=-8.84101
Row=[third, adult, female | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[second, adult, female | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | yes], predicted class=no confidence=-8.84101
Row=[first, adult, female | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
```

```
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | yes], predicted class=no confidence=-8.84101
Row=[second, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[first, adult, female | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[first, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, female | yes], predicted class=no confidence=-8.84101
Row=[first, adult, male | no], predicted class=no confidence=-8.84101
Row=[second, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[first, adult, male | no], predicted class=no confidence=-8.84101
Row=[first, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, child, female | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | yes], predicted class=no confidence=-8.84101
Row=[third, adult, female | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[second, adult, female | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, female | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | yes], predicted class=no confidence=-8.84101
Row=[third, adult, male | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, female | yes], predicted class=no confidence=-8.84101
Row=[first, adult, female | yes], predicted class=no confidence=-8.84101
Row=[second, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | yes], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, female | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, female | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[second, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, female | yes], predicted class=no confidence=-8.84101
Row=[third, adult, male | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, female | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
```

```
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[first, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, child, female | yes], predicted class=no confidence=-8.84101
Row=[third, adult, male | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | yes], predicted class=no confidence=-8.84101
Row=[third, adult, male | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[first, adult, female | yes], predicted class=no confidence=-8.84101
Row=[second, adult, female | yes], predicted class=no confidence=-8.84101
Row=[third, adult, female | yes], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[second, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[second, child, male | yes], predicted class=no confidence=-8.84101
Row=[first, adult, male | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[first, adult, female | yes], predicted class=no confidence=-8.84101
Row=[first, adult, male | yes], predicted class=no confidence=-8.84101
Row=[first, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, female | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | yes], predicted class=no confidence=-8.84101
Row=[first, adult, female | yes], predicted class=no confidence=-8.84101
Row=[third, child, female | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | yes], predicted class=no confidence=-8.84101
Row=[second, adult, female | yes], predicted class=no confidence=-8.84101
Row=[first, adult, male | no], predicted class=no confidence=-8.84101
Row=[first, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, child, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[second, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, female | no], predicted class=no confidence=-8.84101
Row=[first, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, female | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
```

```
Row=[third, adult, female | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[first, adult, female | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[second, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | yes], predicted class=no confidence=-8.84101
Row=[third, adult, male | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | yes], predicted class=no confidence=-8.84101
Row=[first, adult, male | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, female | no], predicted class=no confidence=-8.84101
Row=[second, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, child, male | no], predicted class=no confidence=-8.84101
Row=[third, child, female | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, child, female | no], predicted class=no confidence=-8.84101
Row=[second, adult, female | yes], predicted class=no confidence=-8.84101
Row=[first, adult, male | no], predicted class=no confidence=-8.84101
Row=[second, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, female | yes], predicted class=no confidence=-8.84101
Row=[third, child, male | no], predicted class=no confidence=-8.84101
Row=[second, adult, male | no], predicted class=no confidence=-8.84101
Row=[first, adult, female | yes], predicted class=no confidence=-8.84101
Row=[third, child, female | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[second, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, female | yes], predicted class=no confidence=-8.84101
Row=[third, child, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[second, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[first, adult, male | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | yes], predicted class=no confidence=-8.84101
Row=[second, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[first, adult, male | no], predicted class=no confidence=-8.84101
Row=[second, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[first, adult, male | yes], predicted class=no confidence=-8.84101
Row=[second, child, female | yes], predicted class=no confidence=-8.84101
```

```
Row=[third, adult, male | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[second, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[first, adult, female | yes], predicted class=no confidence=-8.84101
Row=[first, adult, female | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, child, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, female | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[second, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, female | yes], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, female | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[second, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | yes], predicted class=no confidence=-8.84101
Row=[second, adult, female | yes], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, female | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, female | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[second, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | yes], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, child, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[first, adult, male | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[first, adult, female | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[second, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[second, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[second, adult, female | yes], predicted class=no confidence=-8.84101
Row=[first, adult, male | yes], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
```

```
Row=[crew, adult, male | yes], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[second, child, female | yes], predicted class=no confidence=-8.84101
Row=[second, adult, male | no], predicted class=no confidence=-8.84101
Row=[second, adult, female | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | yes], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[second, adult, male | no], predicted class=no confidence=-8.84101
Row=[first, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[first, adult, female | yes], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[second, child, male | yes], predicted class=no confidence=-8.84101
Row=[second, adult, female | yes], predicted class=no confidence=-8.84101
Row=[third, adult, male | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | yes], predicted class=no confidence=-8.84101
Row=[first, adult, female | yes], predicted class=no confidence=-8.84101
Row=[second, child, female | yes], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[first, adult, female | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[second, adult, male | no], predicted class=no confidence=-8.84101
Row=[first, adult, female | yes], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[first, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | yes], predicted class=no confidence=-8.84101
Row=[second, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[first, adult, male | yes], predicted class=no confidence=-8.84101
Row=[second, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, female | no], predicted class=no confidence=-8.84101
Row=[second, adult, male | yes], predicted class=no confidence=-8.84101
Row=[first, adult, female | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[first, child, male | yes], predicted class=no confidence=-8.84101
Row=[first, adult, female | yes], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[second, adult, female | no], predicted class=no confidence=-8.84101
Row=[third, child, male | yes], predicted class=no confidence=-8.84101
```

```
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | yes], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, female | yes], predicted class=no confidence=-8.84101
Row=[third, adult, male | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[second, adult, female | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | yes], predicted class=no confidence=-8.84101
Row=[third, adult, female | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, female | yes], predicted class=no confidence=-8.84101
Row=[third, adult, female | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | yes], predicted class=no confidence=-8.84101
Row=[third, adult, female | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, female | yes], predicted class=no confidence=-8.84101
Row=[third, child, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[second, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | yes], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[second, adult, female | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | yes], predicted class=no confidence=-8.84101
Row=[first, adult, female | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, child, male | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[second, adult, male | no], predicted class=no confidence=-8.84101
Row=[second, adult, male | no], predicted class=no confidence=-8.84101
Row=[second, adult, female | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[first, adult, female | yes], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, female | yes], predicted class=no confidence=-8.84101
Row=[third, adult, female | no], predicted class=no confidence=-8.84101
Row=[first, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, female | yes], predicted class=no confidence=-8.84101
Row=[first, adult, male | yes], predicted class=no confidence=-8.84101
Row=[second, adult, female | no], predicted class=no confidence=-8.84101
```

```
Row=[crew, adult, male | yes], predicted class=no confidence=-8.84101
Row=[second, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[first, adult, female | yes], predicted class=no confidence=-8.84101
Row=[second, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, female | yes], predicted class=no confidence=-8.84101
Row=[second, adult, male | no], predicted class=no confidence=-8.84101
Row=[first, adult, female | yes], predicted class=no confidence=-8.84101
Row=[first, adult, female | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, female | yes], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[second, child, female | yes], predicted class=no confidence=-8.84101
Row=[second, adult, male | no], predicted class=no confidence=-8.84101
Row=[first, adult, female | yes], predicted class=no confidence=-8.84101
Row=[third, adult, female | no], predicted class=no confidence=-8.84101
Row=[first, adult, female | yes], predicted class=no confidence=-8.84101
Row=[second, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[first, adult, female | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | yes], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[second, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, female | yes], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, female | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[second, adult, female | yes], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, female | no], predicted class=no confidence=-8.84101
Row=[first, adult, female | yes], predicted class=no confidence=-8.84101
Row=[first, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[first, adult, female | yes], predicted class=no confidence=-8.84101
Row=[third, adult, male | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[first, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, child, male | no], predicted class=no confidence=-8.84101
Row=[first, adult, female | yes], predicted class=no confidence=-8.84101
Row=[third, adult, male | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, child, female | yes], predicted class=no confidence=-8.84101
Row=[second, adult, male | no], predicted class=no confidence=-8.84101
Row=[first, adult, female | yes], predicted class=no confidence=-8.84101
Row=[second, child, male | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
```

```
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[first, adult, male | no], predicted class=no confidence=-8.84101
Row=[second, adult, female | yes], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[second, adult, male | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, child, male | yes], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[first, adult, male | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, female | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[second, adult, female | yes], predicted class=no confidence=-8.84101
Row=[third, adult, male | yes], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[second, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[first, adult, female | yes], predicted class=no confidence=-8.84101
Row=[third, adult, female | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | yes], predicted class=no confidence=-8.84101
Row=[third, child, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, female | yes], predicted class=no confidence=-8.84101
Row=[second, child, male | yes], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[first, adult, male | yes], predicted class=no confidence=-8.84101
Row=[third, adult, female | no], predicted class=no confidence=-8.84101
Row=[first, adult, female | yes], predicted class=no confidence=-8.84101
Row=[third, child, female | yes], predicted class=no confidence=-8.84101
Row=[first, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, female | no], predicted class=no confidence=-8.84101
Row=[first, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[second, adult, female | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, female | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
```

```
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[first, adult, male | no], predicted class=no confidence=-8.84101
Row=[second, adult, female | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | yes], predicted class=no confidence=-8.84101
Row=[third, adult, female | no], predicted class=no confidence=-8.84101
Row=[second, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | yes], predicted class=no confidence=-8.84101
Row=[second, adult, female | yes], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, child, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, female | yes], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[first, adult, female | yes], predicted class=no confidence=-8.84101
Row=[third, child, female | yes], predicted class=no confidence=-8.84101
Row=[second, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, female | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[first, adult, female | yes], predicted class=no confidence=-8.84101
Row=[first, adult, female | yes], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[first, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[first, adult, female | yes], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, female | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, female | no], predicted class=no confidence=-8.84101
Row=[second, adult, female | no], predicted class=no confidence=-8.84101
Row=[second, adult, female | yes], predicted class=no confidence=-8.84101
```

```
Row=[third, child, female | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, female | yes], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | yes], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[first, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, female | no], predicted class=no confidence=-8.84101
Row=[first, adult, female | yes], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, child, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | yes], predicted class=no confidence=-8.84101
Row=[second, child, female | yes], predicted class=no confidence=-8.84101
Row=[third, adult, female | no], predicted class=no confidence=-8.84101
Row=[second, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, female | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | yes], predicted class=no confidence=-8.84101
Row=[first, adult, male | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | yes], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | yes], predicted class=no confidence=-8.84101
Row=[third, child, female | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[second, child, female | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[second, adult, female | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[first, adult, female | yes], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[second, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, child, female | no], predicted class=no confidence=-8.84101
Row=[first, adult, female | yes], predicted class=no confidence=-8.84101
Row=[second, adult, female | yes], predicted class=no confidence=-8.84101
Row=[second, adult, female | yes], predicted class=no confidence=-8.84101
Row=[third, child, male | no], predicted class=no confidence=-8.84101
Row=[first, adult, female | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
```

```
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[first, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, child, male | yes], predicted class=no confidence=-8.84101
Row=[second, child, male | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[second, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[second, adult, female | yes], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, female | yes], predicted class=no confidence=-8.84101
Row=[first, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | yes], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[second, adult, female | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, female | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[second, adult, female | yes], predicted class=no confidence=-8.84101
Row=[first, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, female | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[second, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, female | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[first, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | yes], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[second, adult, male | yes], predicted class=no confidence=-8.84101
Row=[first, adult, male | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[second, adult, male | no], predicted class=no confidence=-8.84101
Row=[first, adult, female | yes], predicted class=no confidence=-8.84101
Row=[first, adult, male | no], predicted class=no confidence=-8.84101
```

```
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[second, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[second, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[first, adult, female | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[first, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, female | no], predicted class=no confidence=-8.84101
Row=[third, child, female | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[second, adult, male | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[second, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, female | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, child, male | no], predicted class=no confidence=-8.84101
Row=[second, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[first, adult, female | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, female | yes], predicted class=no confidence=-8.84101
Row=[first, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[second, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, female | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[first, adult, male | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | yes], predicted class=no confidence=-8.84101
Row=[third, child, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, female | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | yes], predicted class=no confidence=-8.84101
Row=[first, adult, female | yes], predicted class=no confidence=-8.84101
```

```
Row=[first, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, female | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | yes], predicted class=no confidence=-8.84101
Row=[third, adult, female | no], predicted class=no confidence=-8.84101
Row=[first, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[second, child, male | yes], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[first, adult, male | yes], predicted class=no confidence=-8.84101
Row=[second, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[first, adult, female | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[second, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | yes], predicted class=no confidence=-8.84101
Row=[first, adult, male | yes], predicted class=no confidence=-8.84101
Row=[third, adult, female | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[first, adult, female | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, female | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[second, adult, male | yes], predicted class=no confidence=-8.84101
Row=[second, adult, female | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | yes], predicted class=no confidence=-8.84101
Row=[second, adult, male | no], predicted class=no confidence=-8.84101
Row=[first, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[first, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, child, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, child, male | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[second, adult, male | no], predicted class=no confidence=-8.84101
```

```
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, female | yes], predicted class=no confidence=-8.84101
Row=[second, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | yes], predicted class=no confidence=-8.84101
Row=[third, child, male | yes], predicted class=no confidence=-8.84101
Row=[first, adult, female | yes], predicted class=no confidence=-8.84101
Row=[third, adult, female | yes], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[first, adult, female | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[second, adult, male | no], predicted class=no confidence=-8.84101
Row=[second, adult, female | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, female | yes], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, child, female | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[first, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[second, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[first, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[crew, adult, male | no], predicted class=no confidence=-8.84101
Row=[first, adult, male | yes], predicted class=no confidence=-8.84101
Row=[crew, adult, male | yes], predicted class=no confidence=-8.84101
Row=[third, child, female | yes], predicted class=no confidence=-8.84101
Row=[third, adult, male | no], predicted class=no confidence=-8.84101
Row=[third, adult, male | yes], predicted class=no confidence=-8.84101
```

5.11 Assessing the performance of the classification

In order to evaluate the success of the classification, we compare each predicted example with the corresponding real class. The four possible outcomes of the comparison are as follows:

TP: True Positives (correctly predicted positive examples)

FP: False positives (wrongly predicted negative examples)

TN: True Negatives (correctly predicted negative examples)

FN: False negatives (wrongly predicted positive examples)

5.11.1 Ratio of correctly classified classes (classification accuracy)

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

Pros: * Simple calculation, clear interpretation * Useful measure for any number of classes

Cons: * It can be misleading with unbalanced class distributions

5.11.2 Precision, recall

$$p = \frac{TP}{TP + FP}$$

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

Pros: * Simple calculation, clear interpretation * Separation of both types of errors (incorrectly positive and wrongly negative examples) * Also applicable for unbalanced classroom distributions

Cons: * Applicable predominantly for classification in two classes * It is difficult to summarize both measures; the approximation is F1-value (F1-score)

$$F1 = 2\frac{p \cdot r}{p + r}$$

Do it yourself. Predict the classes on the test set. Compare the predicted classes with the real ones and measure the classification accuracy, precision, recall and F1 value.

In [14]:
```python
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score

# uporaba metod:
test_data      = Table('podatki/titanic-test.tab')
predictions, _ = model.predict(test_data)
truth          = [row["survived"].value for row in test_data]
accuracy_score(truth, predictions)
```

Out[14]: 0.67938237965485926

Challenge. Some attributes have a probability of 0 for each class. How would you repair the classifier?

In [15]:

Think. How to complete a classifier if some of the attributes could also be continuous? Hint: remember the exercises when we learned about the probability distributions of continuous variables.

Chapter 6

Non-negative matrix factorization and recommender systems

So far, we have considered models that predicted *one* dependent variable from *several independent*. In the scenario of a recommendation system, we have built a model for each user.

The main motivation of methods for recommending systems is that user models are *not independent*. We want a single model that will evaluate any combination of user and product, and implicitly exploit mutual information between different user models.

One of the models that are very commonly used in practice is the model of matrix factorization. This assumes a matrix of users and products that we present as a product of two matrices of a *lower rank*. The latter property allows compressing information and concluding new (unobserved, missing values) in the original matrix.

6.1 Introductory definitions

We can present the data matrix $\mathbf{X}$ containing missing values with the matrix factorization model as follows:

$$\mathbf{X} = \mathbf{W}\mathbf{H}^T + \mathbf{E}$$

,

therefore as a product of the $\mathbf{W}$ matrix representing the row space, $\mathbf{H}$ represents the column space, and $\mathbf{E}$ is the residue or error. The $\mathbf{W}, \mathbf{H}$ matrices are sometimes represented as a concurrent clustering of columns and rows. Matrices are of the following sizes:

$$\mathbf{X} \in \mathbb{R}^{m \times n}, \mathbf{W} \in \mathbb{R}^{m \times r}, \mathbf{H} \in \mathbb{R}^{n \times r}, \mathbf{E} \in \mathbb{R}^{m \times n}$$

We assume that the matrices $\mathbf{W}, \mathbf{H}$ are of *low rank*, which in practice means that the entire information from $\mathbf{X}$ is presented in a compressed form, that is,

$$r < m, r < n$$

.

We also assume that the matrices $\mathbf{X}$, $\mathbf{W}$ and $\mathbf{H}$ are non-negative. Then we talk about **non-negative matrix factorization (NMF)**.

$$x_{i,j} > 0, w_{i,k} > 0, h_{j,k} > 0, \forall i, j, k$$

.

The **E** error matrix does not have this limit (think: why?).

6.2 Problem definition

We want to find the matrices **W** and **H** so that the error value is as low as possible. This can be written as the following optimization problem:

$$\min_{\mathbf{W},\mathbf{H}} \|\mathbf{X} - \mathbf{W}\mathbf{H}^T\|_F^2 = \min_{\mathbf{W},\mathbf{H}} J$$

The $\|\mathbf{A}\|_F = \sqrt{\sum_{i,j} a_{i,j}^2}$ notation represents the *Frobenius norm* of matrix **A**. (think: Do you see the similarity with the mean square error that we have seen in the context of linear regression?)

The value J is called *criterion function*, and the problem of searching for a minimum is *optimization or minimization problem*. **The particularity** of recommender systems is that we calculate the error only on the known values in **X**. The criterion function is therefore:

$$J = \sum_{i,j|x_{i,j} \neq 0} (x_{i,j} - \sum_{l=1}^{r} w_{i,l} h_{j,l})^2$$

This particular problem does not have a globally optimal solution for the variables $\mathbf{W}, \mathbf{H}$. However, it can be solved, for example, by deriving the criterion function and moving in the negative direction of the gradient. We get *update rules* for values in $\mathbf{W}, \mathbf{H}$:

All values of $w_{i,k}$ and $h_{j,k}$ are corrected so that the value in the previous iteration *is corrected* in the negative direction of the gradient, with *step* η:

$$w_{i,k}^{(t+1)} = w_{i,k}^{(t)} - \eta \frac{\delta J}{\delta w_{i,k}} = w_{i,k}^{(t)} + \eta \sum_{j \ | \ x_{i,j} \neq 0} (x_{i,j} - \sum_{l=1}^{r} w_{i,l} h_{j,l})(w_{i,k}^{(t)})$$

$$h_{j,k}^{(t+1)} = h_{j,k}^{(t)} - \eta \frac{\delta J}{\delta h_{j,k}} = h_{j,k}^{(t)} + \eta \sum_{i \ | \ x_{i,j} \neq 0} (x_{i,j} - \sum_{l=1}^{r} w_{i,l} h_{j,l})(h_{j,k}^{(t)})$$

6.3 Stochastic gradient descent

Stochastic gradient descent (SGD) is a procedure for solving optimization problems that are not globally solvable, so we can calculate the derivative according to the criterion function for all the variables (in our case all $w_{i,k}$ and $h_{j,k}$). We did this in the previous part. The procedure for searching the *local minimum* is as follows.

1. Randomly set the values of all the variables $w_{i,k}$ and $h_{j,k}$. In our case $w_{i,k} > 0$ and $h_{j,k} > 0$ apply.
2. In the iteration $t = 1 ... T$: 2.1 In the random order, update $\forall i, k, j$

$$w_{i,k}^{(t+1)} = w_{i,k}^{(t)} - \eta \frac{\delta J}{\delta w_{i,k}}$$

$$h_{j,k}^{(t+1)} = h_{j,k}^{(t)} - \eta \frac{\delta J}{\delta h_{j,k}}$$

Schematic representation of the gradient descent for the hypothetical variable w, h and the criterion function $J(w, h)$.

Question 6-1-1 Complete the implementation of the NMF algorithm below by using the update rules in several iterations of a stochastic gradient descent. **Hint.** Only calculate $x_{i,j}$ values that are known (different from 0) when calculating the gradient. For effective implementation, calculate sums $\sum_{i \mid x_{i,j} \neq 0}$ and $\sum_{j \mid x_{i,j} \neq 0}$ first (before the beginning of iterations): * for each row of i we store non-columnar columns * for each column j we store non-linear rows

In [1]:
```
import numpy as np
import itertools
np.random.seed(42)

class NMF:

    """
    Fit a matrix factorization model for a matrix X with missing values.
    such that
        X = W H.T + E
    where
        X is of shape (m, n)    - data matrix
        W is of shape (m, rank) - approximated row space
        H is of shape (n, rank) - approximated column space
        E is of shape (m, n)    - residual (error) matrix
    """

    def __init__(self, rank=10, max_iter=100, eta=0.01):
        """
        :param rank: Rank of the matrices of the model.
        :param max_iter: Maximum nuber of SGD iterations.
        :param eta: SGD learning rate.
        """
        self.rank = rank
        self.max_iter = max_iter
        self.eta = eta


    def fit(self, X, verbose=False):
        """
        Fit model parameters W, H.
        :param X:
            Non-negative data matrix of shape (m, n)
            Unknown values are assumed to take the value of zero (0).
        """
        m, n = X.shape

        W = np.random.rand(m, self.rank)
        H = np.random.rand(n, self.rank)

        # Indices to model variables
```

```
        w_vars = list(itertools.product(range(m), range(self.rank)))
        h_vars = list(itertools.product(range(n), range(self.rank)))

        # Indices to nonzero rows/columns
        nzcols = dict([(j, X[:, j].nonzero()[0]) for j in range(n)])
        nzrows = dict([(i, X[i, :].nonzero()[0]) for i in range(m)])

        # nzrows[i] <- vrni stolpce j, tako da x_ij > 0


        # Errors
        self.error = np.zeros((self.max_iter,))

        for t in range(self.max_iter):
            np.random.shuffle(w_vars)
            np.random.shuffle(h_vars)

            for i, k in w_vars:
                # TODO: your code here
                # Calculate gradient and update W[i, k]
                pass

            for j, k in h_vars:
                # TODO: your code here
                # Calculate gradient and update H[j, k]
                pass

            self.error[t] = np.linalg.norm((X - W.dot(H.T))[X > 0])**2
            if verbose: print(t, self.error[t])

        self.W = W
        self.H = H


    def predict(self, i, j):
        """
        Predict score for row i and column j
        :param i: Row index.
        :param j: Column index.
        """
        return self.W[i, :].dot(self.H[j, :])


    def predict_all(self):
        """
        Return approximated matrix for all
        columns and rows.
        """
        return self.W.dot(self.H.T)
```

Rešitev najdete v 206-1.ipynb.

```
In [2]: %run 206-1.ipynb

ERROR:root:File `'206-1.ipynb.py'` not found.
```

Test the method on a matrix of random data.

```
In [3]: m = 100       # St. vrstic
        n = 80        # St. stolpcev
        rank = 5      # Rang model
        error = 0.1   # Nakljucni šum
        A = np.random.rand(m, rank*2)
        B = np.random.rand(n, rank*2)
        X = A.dot(B.T) + error * np.random.rand(m, n)  # generiramo podatke
```

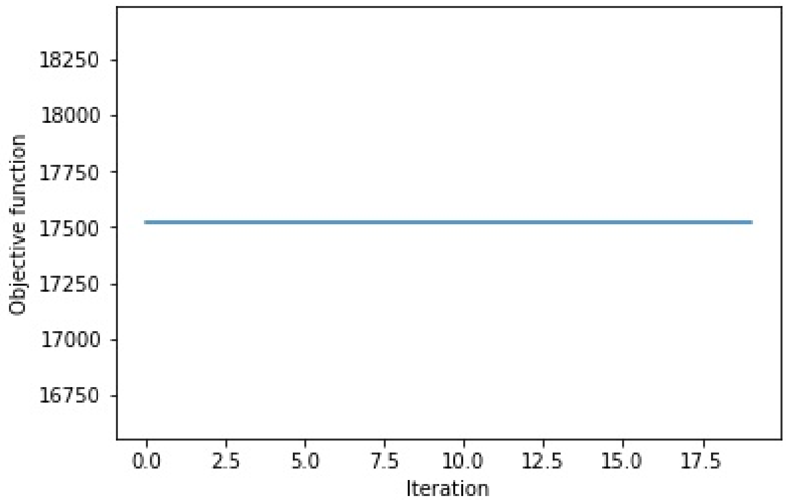
We start searching the parameters **W**, **H**.

```
In [4]: model = NMF(rank=rank, max_iter=20, eta=0.001)
        model.fit(X, verbose=True)
```

```
0 17520.902048
1 17520.902048
2 17520.902048
3 17520.902048
4 17520.902048
5 17520.902048
6 17520.902048
7 17520.902048
8 17520.902048
9 17520.902048
10 17520.902048
11 17520.902048
12 17520.902048
13 17520.902048
14 17520.902048
15 17520.902048
16 17520.902048
17 17520.902048
18 17520.902048
19 17520.902048
```

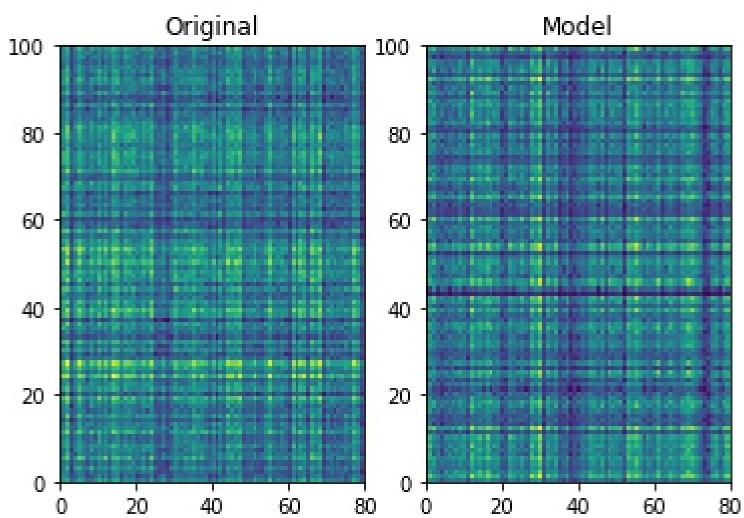
Error of the model falls with number of iterations.

```
In [5]: %matplotlib inline
        %config InlineBackend.figure_formats = ['jpg']
        import matplotlib
        matplotlib.figure.Figure.__repr__ = lambda self: (
            f"<{self.__class__.__name__} size {self.bbox.size[0]:g}"
            f"x{self.bbox.size[1]:g} with {len(self.axes)} Axes>")

        import matplotlib.pyplot as plt

        plt.figure()
        plt.plot(model.error)
        plt.xlabel("Iteration")
        plt.ylabel("Objective function")

Out[5]: Text(0,0.5,'Objective function')
```



Let's compare the model and the original data.

```
In [6]: fig, ax = plt.subplots(nrows=1, ncols=2)
        ax[0].pcolor(X)
        ax[0].set_title("Original")

        ax[1].pcolor(model.predict_all())
        ax[1].set_title("Model")
        plt.show()
```



We calculate the explained variance.

```
In [7]: Xp = model.predict_all()
        expl_var = (np.var(X) - np.var(X-Xp))/np.var(X)
        expl_var

Out[7]: -0.38198039813109452
```

Question 6-1-2 How does the explained variance change with the rang of the model, no. of iterations?

```
In [8]: for rank in range(3, 10):
            model = NMF(rank=rank, max_iter=20, eta=0.001)
            model.fit(X)
            Xp = model.predict_all()
            expl_var = (np.var(X) - np.var(X-Xp))/np.var(X)
            print(rank, expl_var)
```

```
3 -0.30690999008
4 -0.433325233176
5 -0.446298698026
6 -0.577543124238
7 -0.630121526449
8 -0.753749766289
9 -1.036617355
```

Question 6-1-3 Test the NMF method on the Jester database. The data are divided into a learning and test set, where a share p of ratings is present in the learning set. Run the model on the learning set and calculate the test error (RMSE, explained variance) on estimates that were not used for learning. Calculate how the test error varies depending on: * the share of learning estimates of p, * rank matrix of the model (number r, parameter `rank`)

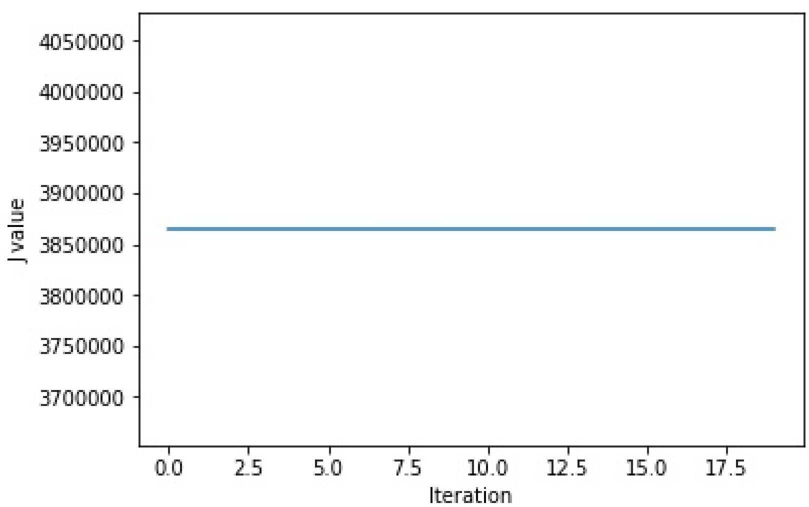
```
In [9]: # Naložimo podatkovno zbirko Jester z 1% upoštevanih ocen
        def load_jester(p=0.05):
            """
            :param p: Probability of rating appearing in the training set.
            :return
                X training grades (retining with probability p)
                Y test grades (whole dataset)
            """

            Y = np.genfromtxt("podatki/jester-data.csv", delimiter=",", dtype=float, )
            Y = Y[:, 1:]
            Y[Y == 99] = 0
            Y[Y != 0]  = Y[Y!=0] + abs(Y[Y!=0].min())

            # Separate data in test/train with probability p
            M = np.random.rand(*Y.shape)
            M_tr = M < p
            M_te = M > p
            X = Y * M_tr
            Y = Y * M_te

            return X, Y

        # X: 1% podatkov, Y ostalih 99%
        X, Y = load_jester(p=0.5)

        X = X[:1000, :]
        Y = Y[:1000, :]
        print("X shape:", X.shape)
        print("Y shape:", Y.shape)

        print("X, Nonzeros:", np.sum(X>0), "Total:", X.shape[0]*X.shape[1])
```

```
        print("Y, Nonzeros:", np.sum(Y>0), "Total:", Y.shape[0]*Y.shape[1])
```

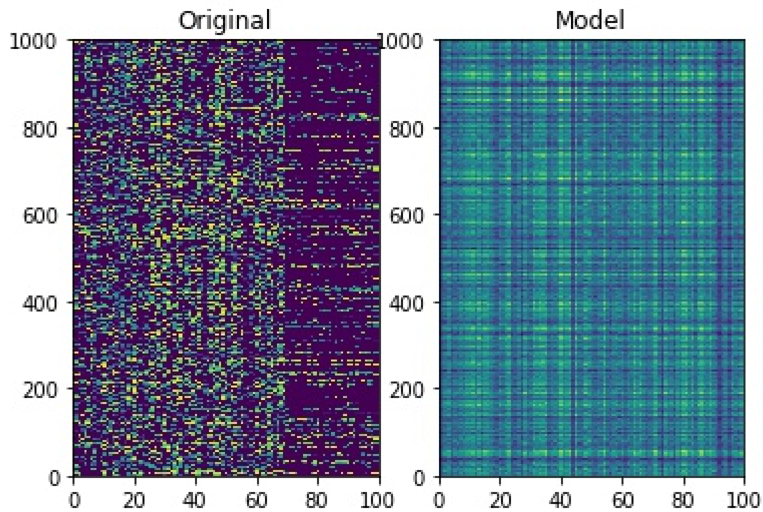
```
X shape: (1000, 100)
Y shape: (1000, 100)
X, Nonzeros: 35573 Total: 100000
Y, Nonzeros: 35772 Total: 100000
```

```
In [10]: model = NMF(rank=7, max_iter=20, eta=0.001)
         model.fit(X)
         Yp = model.predict_all()
```

```
In [11]: plt.figure()
         plt.plot(model.error)
         plt.xlabel("Iteration")
         plt.ylabel("J value")
         plt.show()
```




```
In [12]: fig, ax = plt.subplots(nrows=1, ncols=2)
         ax[0].pcolor(Y)
         ax[0].set_title("Original")

         ax[1].pcolor(Yp)
         ax[1].set_title("Model")
         plt.show()
```




```
In [13]: expl_var = (np.var(Y[Y>0]) - np.var(Y[Y>0] - Yp[Y>0])) / np.var(Y[Y>0])
         print(expl_var)
```

```
-0.00695500063601
```

Question 6-1-4 On the Jester database, select one cell with a value other than 0, and set it to 0. Factorize and predict the value of this cell.

Question 6-1-5 Locate cells where the difference between the approximate and the original matrix is greatest.

Question 6-1-6 Create a recommended system. Choose a few users and for each of them output five yet unrated jokes that they will like the most, according to the prediction.

Chapter 7

Networks

7.1 networkx library

Simple handling of graph data in Python.

```
In [1]: import networkx as nx
        import numpy as np
        import matplotlib.pyplot as plt
        %matplotlib inline
        %config InlineBackend.figure_formats = ['jpg']
        import matplotlib
        matplotlib.figure.Figure.__repr__ = lambda self: (
            f"<{self.__class__.__name__} size {self.bbox.size[0]:g}"
            f"x{self.bbox.size[1]:g} with {len(self.axes)} Axes>")
```

7.1.1 Graph creation

Create a simple graph.

```
In [2]: G = nx.Graph()    # Undirected
        # G = nx.DiGraph()  # Directed

        G.add_node("Ana")
        G.add_nodes_from(["Bojan", "Cene", "Danica"])

        G.add_edge("Ana", "Bojan")
        G.add_edge("Ana", "Cene")
        G.add_edge("Ana", "Danica")
        G.add_edge("Bojan", "Danica")
```

```
In [3]: G.nodes
```

```
Out[3]: NodeView(('Ana', 'Bojan', 'Cene', 'Danica'))
```

```
In [4]: G.edges
```

```
Out[4]: EdgeView([('Ana', 'Bojan'), ('Ana', 'Cene'), ('Ana', 'Danica'), ('Bojan', 'Danica')])
```

Write the graph into a file.

```
In [5]: nx.write_pajek(G, 'podatki/mreza-primer.net')
```

Read the `.net` file in a Graph structure.

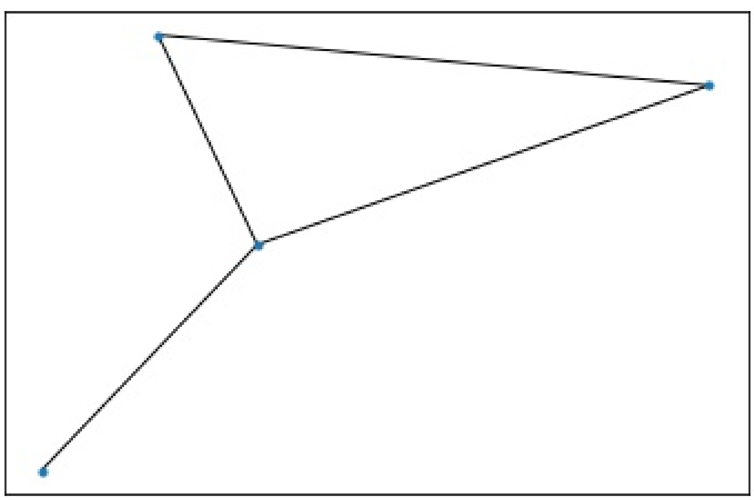
```
In [6]: G = nx.read_pajek('podatki/mreza-primer.net')
```

7.1.2 Drawing the graph

Draw the graph structure using `matplotlib`.

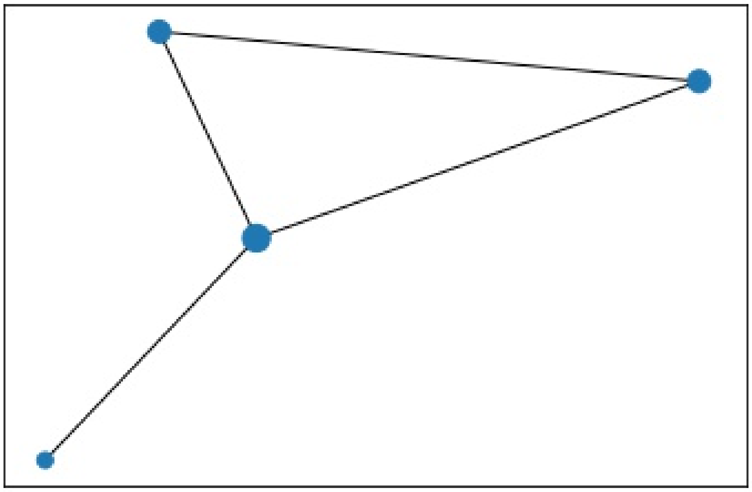
For more options, see the documentation.

```
In [7]: plt.figure()
        np.random.seed(42)
        nx.draw_networkx(G, with_labels=False, node_size=10)
        plt.show()
```



Compute node sizes proportional to the number of edges for a node. Plot using `draw.networkx(...,node_size=node_size)`

```
In [8]: node_size = [50 * G.degree(ky) for ky in G.node]
        node_size

        plt.figure()
        np.random.seed(42)
        nx.draw_networkx(G, with_labels=False, node_size=node_size)
        plt.show()
```



7.1.3 Network segmentation

Finding strongly connected components inside a network.

First, we load the data. As this is the network of email correspondents for a given address, we remove the central node (why?).

In [9]:
```
H = nx.read_pajek("podatki/email.net")
H = nx.Graph(H)

# Remove central node
myself = "rok0"
H.remove_node(myself)
```

```
---------------------------------------------------------------------------

FileNotFoundError                         Traceback (most recent call last)

<ipython-input-9-fe2cbc88c35d> in <module>
----> 1 H = nx.read_pajek("podatki/email.net")
      2 H = nx.Graph(H)
      3
      4 # Remove central node
      5 myself = "rok0"


</Users/tomazc/anaconda3/lib/python3.6/site-packages/decorator.py:decorator-gen-717> in read_pa


~/anaconda3/lib/python3.6/site-packages/networkx/utils/decorators.py in _open_file(func_to_be_d
    212         if is_string_like(path):
    213             ext = splitext(path)[1]
--> 214             fobj = _dispatch_dict[ext](path, mode=mode)
    215             close_fobj = True
    216         elif hasattr(path, 'read'):
```

```
FileNotFoundError: [Errno 2] No such file or directory: 'podatki/email.net'
```

Next, we find the k-connected components. A k-connected component is a connected subgraph, for which we need to remove at least k nodes to break it into more components. Intuitively, subgraphs with large value of k are harder to break and thus more strongly connected.

```
In [10]: from networkx.algorithms import approximation as apxa
         k_components = apxa.k_components(H)

         k_components
```

```
---------------------------------------------------------------------------

NameError                                 Traceback (most recent call last)

<ipython-input-10-593d9d9e0cd7> in <module>
      1 from networkx.algorithms import approximation as apxa
----> 2 k_components = apxa.k_components(H)
      3
      4 k_components


NameError: name 'H' is not defined
```

Let's look at solutions for a given k and look at the number of nodes on each connected component.

```
In [11]: k = 2                     # Subgraphs of connectivity k
         sol = k_components[k]     # Multiple solutions of k_components
         list(map(len, sol))       # Each component breaks a graph
```

```
---------------------------------------------------------------------------

NameError                                 Traceback (most recent call last)

<ipython-input-11-eec503e45b4e> in <module>
      1 k = 2                     # Subgraphs of connectivity k
----> 2 sol = k_components[k]     # Multiple solutions of k_components
      3 list(map(len, sol))       # Each component breaks a graph


NameError: name 'k_components' is not defined
```

For each connected component, assign the black color to its corresponding nodes and white to all other nodes.

```
In [12]: colors_groups = list()
         for gi, group in enumerate(sol):
             colors_arr = ["red" if (n in group) else "gray" for n in H.node]
             colors_groups.append(colors_arr)
```

```
---------------------------------------------------------------------------
```

```
NameError                                 Traceback (most recent call last)

<ipython-input-12-8dab3a9a22b1> in <module>
  1 colors_groups = list()
----> 2 for gi, group in enumerate(sol):
  3     colors_arr = ["red" if (n in group) else "gray" for n in H.node]
  4     colors_groups.append(colors_arr)


NameError: name 'sol' is not defined
```

Plot a selected component.

```
In [13]: comp_index = 1
         plt.figure()
         nx.draw_networkx(H, with_labels=False,
                          node_color=colors_groups[comp_index],)
         plt.show()
```

```
---------------------------------------------------------------------------

NameError                                 Traceback (most recent call last)

<ipython-input-13-a60994b2ce76> in <module>
  1 comp_index = 1
  2 plt.figure()
----> 3 nx.draw_networkx(H, with_labels=False,
  4                   node_color=colors_groups[comp_index],)
  5 plt.show()


NameError: name 'H' is not defined
```

```
<Figure size 432x288 with 0 Axes>
```

7.2 Primer: analiza in vizualizacija omrežja elektronskih sporočil

V tej kratki vaji bomo spoznali osnove analize omrežij, format `.net` in funkcionalnosti modula Orange - Networks.

Pripravili smo funkcijo, ki iz email računa (protokol IMAP) prebere vse naslovnike sporočil. Tako zgradimo omrežje so-naslovnikov danega računa.

```
In [1]: from get_email import generate_addressee_network
        help(generate_addressee_network)
```

```
---------------------------------------------------------------------------

ModuleNotFoundError                       Traceback (most recent call last)
```

```
    <ipython-input-1-92545730b825> in <module>
----> 1 from get_email import generate_addressee_network
      2 help(generate_addressee_network)


    ModuleNotFoundError: No module named 'get_email'
```

S spodnjo funkcijo zgradimo podatkovne datoteke `.txt` (seznam sonaslovnikov), `.tab` (atributi vozlišč), `.net` (graf omrežja). Oglej si kodo in format datotek.

```
In [2]: generate_addressee_network("someone@gmail.com", imap='imap.gmail.com', max_tuples=10000,
                                   email_folder="INBOX", file_prefix="ds",
                                   min_tuple_length=2, max_tuple_length=15)
```

```
    ---------------------------------------------------------------------------

    NameError                                 Traceback (most recent call last)

    <ipython-input-2-fa8231f156e0> in <module>
----> 1 generate_addressee_network("someone@gmail.com", imap='imap.gmail.com', max_tuples=10000,
      2                                email_folder="INBOX", file_prefix="ds",
      3                                min_tuple_length=2, max_tuple_length=15)


    NameError: name 'generate_addressee_network' is not defined
```

Uporabimo programski paket Orange oz. dodatek "Networks".

Podatke naložimo z vtičnikom Network File, ki prebere podatke o vozliščih in povezavah.

Z uporabo algoritmov v vtičniku Network Clustering poiščemo močno povezane komponente v omrežju.

Naredi sam/a. Ponovi zgornjo analizo za podatke iz svojega e-poštnega računa.

```
In [3]: # ...
```

Naredi sam/a. Zgradi omrežje igralcev v podatkovni zbirki MovieLens.

```
In [4]: # ...
```

Naredi sam/a. Zgradi omrežje uporabnikov v podatkovni zbirki MovieLens.

```
In [5]: # ...
```

Chapter 8

Zaporedja

8.1 Skriti Markovi modeli

Skriti markov model (ang. Hidden Markov model - HMM) je generativni model, ki ponazarja zaporedje diskretnih podatkov. Je razširitev Markovih verig (ang. Markov chain), na način da so opazovane spremenljivke odvisne od trenutnega skritega stanja.

Denimo, da opazujemo mete kovanca, ki jih izvaja druga oseba. Na voljo ima dva kovanca: pošten (F - fair) in utežen (L - loaded). Pri vsakem metu lahko opazujemo le izid (o ali -), ne pa tudi kovanca. Skriti Markov model je zapis tovrstnega problema, s poljubnim končnim številom tako skritih stanj in kot tudi opazovanih spremenljivk (abecede).

Primer zaporedja skritih stanj in opazovanih spremenljivk:

```
S: FFFFFFLLLLLFFFFFLLLLFFFFLL...
X: -o-o-ooooo-o--o-ooo-oo-ooo...
```

Celoten model je podan z naborom verjetnosti. Te predstavljajo parametre modela.

Verjenosto opazovanih spremenljivk X v koraku i glede trenutno stanje S:

$$P(X_i = o \mid S_i = F) = \frac{1}{2},\ P(X_i = - \mid S_i = F) = \frac{1}{2}$$

$$P(X_i = o \mid S_i = L) = \frac{19}{20},\ P(X_i = - \mid S_i = L) = \frac{1}{20}$$

Za vsako skrito stanje je torej definirana verjetnostna porazdelitev opazovanih spremenljivk.

V praktičnih primerih uporabe HMM se stanja ohranjajo. Verjetnost ohranitve stanja je torej navadno večja od zamenjave stanja. Verjetnosti prehodov podajajo drugo skupino parametrov.

$$P(S_{i+1} = F|S_i = F) = \frac{19}{20},\ P(S_{i+1} = L|S_i = F) = \frac{1}{20}$$

$$P(S_{i+1} = L|S_i = L) = \frac{19}{20},\ P(S_{i+1} = F|S_i = L) = \frac{1}{20}$$

Navadno definiramo tudi začetne verjetnosti skritih stanj (verjetnost v koraku $i = 0$):

$$P(S_0 = F) = \frac{1}{2},\ P(S_0 = L) = \frac{1}{2}$$

Tako definiran model uporabljamo za praktične naloge, kot so: * generiranje zaporedij iz danega modela,

- učenje parametrov modela iz danih podatkov:
  - podana so skrita stanja in opazovane spremenljivke (štetje pojavitev)
  - podane so samo opazovanje spremenljivke in število skritih stanj (algoritem Baum-Welch)
- napoved skritih stanj za dano zaporedje opazovanih spremeljivk pri danem modelu (algoritma Viterbi ter Posterior-decoding)

Primeri praktičnih problemov, ki jih rešujemo z uporabo Skritih Markovih modelov: * prepoznavanje in generiranje govora, * strojno prevajanje, * prepoznavanje pisave, * segmentacija besedil (prepoznavanje besednih vrst), * analiza biološki zaporedij (iskanje genov, poravnava zaporedij), * kriptoanaliza, * …

Model lahko zapišemo s slovarjem slovarjev. Na primer, za metanje kovancev:

```
In [1]: # Transition matrix
        T = {"F": {"F": 0.95, "L": 0.05},
             "L": {"F": 0.05, "L": 0.95}}

        # Emission matrix
        E = {"F": {"o": 0.5, "-": 0.5},
             "L": {"o": 0.95, "-": 0.05}}
        start = "F"
```

8.1.1 Generiranje zaporedij

Naredi sam/a. Zapiši funkcijo `generate_hmm_sequence`, ki sprejme skriti Markov model in vrne zaporedje dolžine n (skrito in vidno zaporedje).

Še prej zapišite funkcijo `weighted_choice`, ki na podlagi uteži (v vrednosti) naključno izbere vrednost (v ključu slovarja).

```
In [2]: import random
        random.seed(42)

        def weighted_choice(weighted_items):
            """Random choice given the list of elements and their weights
                example weighted_items: {"F": 0.95, "L": 0.05}
            """

            pass
```

Zdaj pa funkcijo generate_hmm_sequence:

```
In [3]: def generate_hmm_sequence(h, T, E, n):
            """
            h: given start state,
            T: transition probabilities
            E: emission probabilities
            n: sequence length

            return:
                hidden_sequence
                observable_sequence
```

```
        """
        pass
```

Rešitev lahko pogledate v 208-1.ipynb.

```
In [4]: %run '208-1.ipynb'
```

```
ERROR:root:File `'208-1.ipynb.py'` not found.
```

Generiraj nekaj zaporedij različnih dolžin.

```
In [5]: list(generate_hmm_sequence('F', T, E, 5))
```

```
        ---------------------------------------------------------------------------

        TypeError                                 Traceback (most recent call last)

        <ipython-input-5-b7ac19a359e0> in <module>
    ----> 1 list(generate_hmm_sequence('F', T, E, 5))


        TypeError: 'NoneType' object is not iterable
```

```
In [6]: list(generate_hmm_sequence('F', T, E, 20))
```

```
        ---------------------------------------------------------------------------

        TypeError                                 Traceback (most recent call last)

        <ipython-input-6-d419da9b7f30> in <module>
    ----> 1 list(generate_hmm_sequence('F', T, E, 20))


        TypeError: 'NoneType' object is not iterable
```

Model poskusite uporabiti tudi na primeru goljufive igralnice. Kovanec smo zamenjali z igralno kocko, ki vrača vrednosti 1-6.

```
In [7]: # Alphabet
        A = ["1", "2", "3", "4", "5", "6"]

        # Emission probabilities
        E = {"F": {a: 1/6. for a in A},
             "L": {a: 1/10. if a != "6" else 0.5 for a in A}}

        # Transition probabilities
        T =  {0:  {0: 0, "F": 0.5, "L": 0.5},
             "F": {0: 0, "F": 0.95, "L": 0.05},
             "L": {0: 0, "F": 0.1, "L": 0.9}}
        start = "F"
```

```
In [8]: list(generate_hmm_sequence('F', T, E, 5))
```

```
        ---------------------------------------------------------------------------
```

```
    TypeError                                 Traceback (most recent call last)

    <ipython-input-8-b7ac19a359e0> in <module>
----> 1 list(generate_hmm_sequence('F', T, E, 5))


    TypeError: 'NoneType' object is not iterable
```

```
In [9]: list(generate_hmm_sequence('F', T, E, 20))
```

```
    ---------------------------------------------------------------------------

    TypeError                                 Traceback (most recent call last)

    <ipython-input-9-d419da9b7f30> in <module>
----> 1 list(generate_hmm_sequence('F', T, E, 20))


    TypeError: 'NoneType' object is not iterable
```

8.1.2 Učenje parametrov modela iz podatkov

Napišite funkcijo `learn_hmm`, ki bo sprejela vidno in skrito zaporedje, ter vrnila parametre skritega Markovega modela (slovarja `T` in `E`).

```
In [10]: from collections import Counter

         def normalize(dic, eps=1e-8):
             """
             Normalize probabilities of items in a dictionary `dic`.
             Correct probabilities with a small constant to prevent probability 0.

             dic = {"o": 90, "-": 10}

             return
                 dic = {"o": 0.9, "-": 0.1}
             """
             pass


         def learn_hmm(h, x):
             """
             h: hidden sequence
             x: observable sequence
             """

             return T, E
```

Rešitev lahko pogledate v 208-1.ipynb.

```
In [11]: %run '208-1.ipynb'
```

```
ERROR:root:File `'208-1.ipynb.py'` not found.
```

```
In [12]: n = 40
         h, x = zip(*list(generate_hmm_sequence('F', T, E, n)))
```

```
        ---------------------------------------------------------------------------

        TypeError                                 Traceback (most recent call last)

        <ipython-input-12-731639424f0c> in <module>
          1 n = 40
    ----> 2 h, x = zip(*list(generate_hmm_sequence('F', T, E, n)))


        TypeError: 'NoneType' object is not iterable
```

```
In [13]: # Estimated parameters from data
         T_est, E_est = learn_hmm(h, x)
```

```
        ---------------------------------------------------------------------------

        NameError                                 Traceback (most recent call last)

        <ipython-input-13-88a310a8e988> in <module>
          1 # Estimated parameters from data
    ----> 2 T_est, E_est = learn_hmm(h, x)


        NameError: name 'h' is not defined
```

```
In [14]: T_est
```

```
        ---------------------------------------------------------------------------

        NameError                                 Traceback (most recent call last)

        <ipython-input-14-86299011f15e> in <module>
    ----> 1 T_est


        NameError: name 'T_est' is not defined
```

```
In [15]: E_est
```

```
        ---------------------------------------------------------------------------

        NameError                                 Traceback (most recent call last)

        <ipython-input-15-865fe85fe9c1> in <module>
    ----> 1 E_est
```

```
NameError: name 'E_est' is not defined
```

Primerjamo z dejanskimi:

```
In [16]: T
```

```
Out[16]: {0: {0: 0, 'F': 0.5, 'L': 0.5},
          'F': {0: 0, 'F': 0.95, 'L': 0.05},
          'L': {0: 0, 'F': 0.1, 'L': 0.9}}
```

```
In [17]: E
```

```
Out[17]: {'F': {'1': 0.16666666666666666,
           '2': 0.16666666666666666,
           '3': 0.16666666666666666,
           '4': 0.16666666666666666,
           '5': 0.16666666666666666,
           '6': 0.16666666666666666},
          'L': {'1': 0.1, '2': 0.1, '3': 0.1, '4': 0.1, '5': 0.1, '6': 0.5}}
```

8.1.3 Viterbijev algoritem

Algoritem za iskanje najverjetnejšega zaporedja skritih stanj (Viterbi).

Zaporedja, s katerimi delamo, so lahko zelo dolga. Množenje (majhnih) verjetnosti nas lahko hitro privede do napake *underflow*. Težavi se izognemo tako, da namesto množenja verjetnosti, seštevamo logaritme verjetnosti.

```
In [18]: import math
         def logmv(a):
             min_val = 0.0000000001
             return math.log(max(a, min_val))

         def viterbi_log(s, hmm):
             t, e = hmm

             # seznam skritih stanj
             zh = set()
             for h, tmpd in e.items():
                 zh.add(h)
             zh = [0] + list(zh)

             # Create table V
             V = [{} for i in range(len(s)+1)]
             ptr = [{} for i in range(len(s)+1)]

             # Initialize i = 0; V(0, 0) = 1; V(k, 0) = 0 for k > 0
             for k in zh:
                 V[0][k] = logmv(0.0) #t[0][k]*e[k][s[0]]
             V[0][0] = logmv(1.0)

             # for 1 = 1 : n, compute
             for i in range(1, len(s)+1):
                 for l in zh:
```

```
                vals = [(V[i-1][k] + logmv(t[k].get(l, 0.0)), k) for k in zh]
                max_val, max_k = max(vals)
                V[i][l] = logmv(e.get(l, {}).get(s[i-1], 0.0)) + max_val
                ptr[i][l] = max_k

        # trace back
        pi = []
        pi_L = max([(V[-1][k], k) for k in zh])[1]
        pi.append(pi_L)

        for p in ptr[-1:1:-1]:
            pi.append(p[pi[-1]])

        pi.reverse()
        return V, zh, ptr, "".join(pi)
```

Pokliči funkcijo, ki za dano zaporedje x in za dani model (T in E) vrne najbolj verjetno skrito pot (h_najv).

Primerjaj jo z dejansko skrito potjo.

```
In [19]: # Alphabet
         A = ["1", "2", "3", "4", "5", "6"]

         # Emission probabilities
         E = {"F": {a: 1/6. for a in A},
              "L": {a: 1/10. if a != "6" else 0.5 for a in A}}

         # Transition probabilities
         T =  {0:  {0: 0, "F": 0.5, "L": 0.5},
              "F": {0: 0, "F": 0.95, "L": 0.05},
              "L": {0: 0, "F": 0.1, "L": 0.9}}


         hmm = (T, E)
         #s = "1233516266666666666663561253612365611213231524112666666666611666666666612"

         random.seed(442)
         skrito, vidno = zip(*generate_hmm_sequence('L', T, E, 71))
         skrito = "".join(skrito)
         vidno = "".join(vidno)

         _, _, _, napoved = viterbi_log(vidno, hmm)

         print(vidno)
         print(skrito)
         print(napoved)
```

```
        ---------------------------------------------------------------------------

        TypeError                                 Traceback (most recent call last)

        <ipython-input-19-7ee266dc12ce> in <module>
         16
         17 random.seed(442)
    ---> 18 skrito, vidno = zip(*generate_hmm_sequence('L', T, E, 71))
```

```
 19 skrito = "".join(skrito)
 20 vidno = "".join(vidno)


TypeError: type object argument after * must be an iterable, not NoneType
```

Izračunaj delež ujemanja:

```
In [20]: sum(pi == pj for pi, pj in zip(skrito, napoved))/len(skrito)
```

```
---------------------------------------------------------------------------

NameError                                 Traceback (most recent call last)

<ipython-input-20-f814a227f9fd> in <module>
----> 1 sum(pi == pj for pi, pj in zip(skrito, napoved))/len(skrito)


NameError: name 'skrito' is not defined
```

8.2 Časovne vrste

Modeliranje časovnih vrst je pomeben del ekonomskih modelov, borznega posredništva, analize časovnih meritev v fiziki, biologiji, kemiji, ipd.

Podatki v obliki časovnih vrst se od dosedanjih scenarijev razlikujejo po pomebni lastnosti: vzorci niso neodvisni med seboj. Podatke predstavlja vektor časovnih točk (ki niso nujno enako oddaljene):

$$\mathbf{t} = (t_1, t_2, ..., t_n)$$

Običajno nas zanima funkcija oz. signal v vsaki časovni točki.

$$x(\mathbf{t}) = (x(t_1), x(t_2), ..., x(t_n))$$

```
In [1]: import os
        import sys
        import mlpy

        %matplotlib inline
        %config InlineBackend.figure_formats = ['jpg']
        import matplotlib
        matplotlib.figure.Figure.__repr__ = lambda self: (
            f"<{self.__class__.__name__} size {self.bbox.size[0]:g}"
            f"x{self.bbox.size[1]:g} with {len(self.axes)} Axes>")

        import matplotlib.pyplot as plt
        plt.style.use('PR.mplstyle')
        import matplotlib.cm as cm
        import numpy as np
        import scipy
        import os
```

```
import GPy

np.random.seed(42)
```

```
---------------------------------------------------------------------------

ModuleNotFoundError                       Traceback (most recent call last)

<ipython-input-1-86e0c314098b> in <module>
      1 import os
      2 import sys
----> 3 import mlpy
      4
      5 get_ipython().run_line_magic('matplotlib', 'inline')


ModuleNotFoundError: No module named 'mlpy'
```

8.2.1 Primerjava časovnih vrst

Oglejmo si preprost primer dveh podobnih signalov $x(\mathbf{t})$ in $y(\mathbf{t})$.

In [2]:
```
resolution = 100
t = np.linspace(-5, 5, resolution)
x = np.sin(t)    * np.cos(2*t) + 0.2*np.random.rand(1, resolution).ravel()
y = -np.sin(t-2) * np.cos(2*t + 4) + 0.2*np.random.rand(1, resolution).ravel()

from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(t.reshape((len(t), 1)), x)
z = model.predict(t.reshape((len(t), 1)))

plt.figure()
plt.plot(x, "b.-", label="$x(t)$")
# plt.plot(z, "r-", label="$x(t)$")
plt.plot(y, "r.-", label="$y(t)$")
plt.gca().set_xticklabels(np.linspace(-5, 5, 6))
plt.xlabel("t")
plt.legend()
plt.show()
```

```
---------------------------------------------------------------------------

NameError                                 Traceback (most recent call last)

<ipython-input-2-14cb7c2528c3> in <module>
      1 resolution = 100
----> 2 t = np.linspace(-5, 5, resolution)
      3 x = np.sin(t)    * np.cos(2*t) + 0.2*np.random.rand(1, resolution).ravel()
      4 y = -np.sin(t-2) * np.cos(2*t + 4) + 0.2*np.random.rand(1, resolution).ravel()
      5
```

```
NameError: name 'np' is not defined
```

Opazimo, da sta si signala zelo podobna, vendar so vrhovi zelo oddaljeni med seboj. Kako izmeriti to razdaljo (npr. za potrebe hierarhičnega gručenja)? Evklidska razdalja vrne navidez zelo visoko vrednost.

```
In [3]: np.linalg.norm(x[:-2]-x[2:], ord=2)
```

```
---------------------------------------------------------------------------

NameError                                 Traceback (most recent call last)

<ipython-input-3-a6d5c6a3198d> in <module>
----> 1 np.linalg.norm(x[:-2]-x[2:], ord=2)


NameError: name 'np' is not defined
```

Korelacija med signaloma je nizka oz. celo obratna.

```
In [4]: scipy.stats.pearsonr(x, y)[0]
```

```
---------------------------------------------------------------------------

NameError                                 Traceback (most recent call last)

<ipython-input-4-f38797f4b1fb> in <module>
----> 1 scipy.stats.pearsonr(x, y)[0]


NameError: name 'scipy' is not defined
```

8.2.2 Dinamična poravnava signalov

V splošnem imamo dva različno dolga signala:

$$x(\mathbf{t}) = (x(t_1), x(t_2), ..., x(t_n))$$

$$y(\mathbf{t}) = (y(t_1), y(t_2), ..., y(t_m))$$

Dinamična poravnava signalov (ang. Dynamic time warping, DTW) je algoritem dinamičnega programiranja, ki poišče signala $x_w(\mathbf{t})$ in $y_w(\mathbf{t})$, tako, da je razdalja med vrednostmi signalov $|x_w(t_k) - y_w(t_k)|$ čim manjša. Dovoljeno je lokalno raztezanje in krčenje obeh signalov.

Algoritem DTW sestavi matriko $\mathbf{W}$ velikosti $m \times n$ ki hrani razdalje, tako da

$$w_{ij} = |x(t_i) - y(t_j)|$$

Cilj algoritma DTW je iskanje poti dolžine $max(m, n)$, ki gre iz levega spodnjega v desni zgornji kot matrike $\mathbf{W}$, tako da zmanjšamo skupno razdaljo

$$\min \sum_k \sqrt{w_k}$$

Rezultat algoritma je matrika poravnave, optimalna pot in skupna razdalja med signaloma.

```
In [5]: dist, cost, path = mlpy.dtw_std(x, y, dist_only=False)
        print("Oddaljenost med x in y", dist)
```

```
        ---------------------------------------------------------------------------

        NameError                                 Traceback (most recent call last)

        <ipython-input-5-68b036f3cefd> in <module>
    ----> 1 dist, cost, path = mlpy.dtw_std(x, y, dist_only=False)
          2 print("Oddaljenost med x in y", dist)


        NameError: name 'mlpy' is not defined
```

```
In [6]: fig = plt.figure(2)
        ax = fig.add_subplot(111)
        plot1 = plt.imshow(cost.T, origin='lower', cmap=cm.gray, interpolation='nearest')
        plot2 = plt.plot(path[0], path[1], 'y', label="Pot w_k")
        xlim = ax.set_xlim((-0.5, cost.shape[0]-0.5))
        ylim = ax.set_ylim((-0.5, cost.shape[1]-0.5))
        plt.xlabel("Indeks $y$")
        plt.ylabel("Indeks $x$")
        plt.title("Optimalna matrika poravnave",)
        plt.legend( loc=4)
        plt.show()
```

```
        ---------------------------------------------------------------------------

        NameError                                 Traceback (most recent call last)

        <ipython-input-6-6b4a84ec98cc> in <module>
    ----> 1 fig = plt.figure(2)
          2 ax = fig.add_subplot(111)
          3 plot1 = plt.imshow(cost.T, origin='lower', cmap=cm.gray, interpolation='nearest')
          4 plot2 = plt.plot(path[0], path[1], 'y', label="Pot w_k")
          5 xlim = ax.set_xlim((-0.5, cost.shape[0]-0.5))


        NameError: name 'plt' is not defined
```

Poravnana signala dobimo s poravnavo vrednosti na ustreznih mestih v zaporedju.

```
In [7]: xw = x[path[0]]
        yw = y[path[1]]
```

```
        ---------------------------------------------------------------------------
```

```
NameError                                 Traceback (most recent call last)

<ipython-input-7-8f2a52d1f936> in <module>
----> 1 xw = x[path[0]]
      2 yw = y[path[1]]


NameError: name 'x' is not defined
```

Korelacija med signaloma je bistveno večja!

```
In [8]: scipy.stats.pearsonr(xw, yw)[0]
```

```
---------------------------------------------------------------------------

NameError                                 Traceback (most recent call last)

<ipython-input-8-edc467517e3d> in <module>
----> 1 scipy.stats.pearsonr(xw, yw)[0]


NameError: name 'scipy' is not defined
```

Oba signala sta lokalno deformirana.

```
In [9]: plt.figure()
        plt.plot(xw, label="$x_w(t)$")
        plt.plot(yw, label="$y_w(t)$")
        plt.legend()
        plt.show()
```

```
---------------------------------------------------------------------------

NameError                                 Traceback (most recent call last)

<ipython-input-9-d80ac1b5a14f> in <module>
----> 1 plt.figure()
      2 plt.plot(xw, label="$x_w(t)$")
      3 plt.plot(yw, label="$y_w(t)$")
      4 plt.legend()
      5 plt.show()


NameError: name 'plt' is not defined
```

Naredi sam/a. Poišči slovenske občine s podobnimi trendi spreminjanja gostote prebivalstva. Oglej si trende nekaj najpodobnejših krajev.

```
In [10]: x = np.loadtxt('podatki/ages/Maribor_starost-20-24_let.txt')
         y = np.loadtxt('podatki/ages/Ljubljana_starost-20-24_let.txt')
```

```
---------------------------------------------------------------------------
```

```
NameError                                 Traceback (most recent call last)

<ipython-input-10-875ce3b25d70> in <module>
----> 1 x = np.loadtxt('podatki/ages/Maribor_starost-20-24_let.txt')
      2 y = np.loadtxt('podatki/ages/Ljubljana_starost-20-24_let.txt')


NameError: name 'np' is not defined
```

```
In [11]: # ... load all cities and store them into a matrix
         import glob
         labels = []
         X = []

         for f in glob.glob('podatki/ages/*_starost-20-24_let.txt'):
             city = os.path.basename(f).split("_")[0]
             labels.append(city)
             data = np.loadtxt(f)
             d = scipy.stats.zscore(data)
             X.append(d)
         X = np.array(X)
         X.shape
```

```
---------------------------------------------------------------------------

NameError                                 Traceback (most recent call last)

<ipython-input-11-c3983a4792d4> in <module>
      7     city = os.path.basename(f).split("_")[0]
      8     labels.append(city)
----> 9     data = np.loadtxt(f)
     10     d = scipy.stats.zscore(data)
     11     X.append(d)


NameError: name 'np' is not defined
```

Namig. Uporabi hierarhično razvšanje, kjer namesto funkcije za razdaljo (`sch.linkage(metric=...)`) podaš razdaljo izmerjeno po DTW.

```
In [12]: import scipy.cluster.hierarchy as sch
         # TODO: your code here
```

8.3 Napovedovanje trendov

8.3.1 Gaussovi procesi

Gaussovi procesi so paradni konj družine modelov, ki ji pravimo neparametrična regresija. Napovedni model tokrat ne bo predstavljen kot vektor uteži, temveč bo vsa informacija za napovedovanje vsebovana v učnem vzorcu. Prednosti pristopa sta: * predpostavka, da so primeri neodvisni med seboj ne drži več, * model se posodobi, ko se pojavijo novi primeri.

Glana predpostavka je naslednja. Funkcija $x(\mathbf{t})$ je porazdeljena po multivariatni normalni porazdelitvi. To ne pomeni, da je vsaka vrednost $(x(t_i))$ porazdeljena normalno, temveč da celoten vektor $x(\mathbf{t})$ prihaja iz skupne noramalne porazdelitve , kjer so posamezne vrednosti $(x(t_i))$ lahko odvisne med sabo!

Torej:

$$x(\mathbf{t}) \sim \mathcal{N}(m(\mathbf{t}), k(\mathbf{t}, \mathbf{t}))$$

Funkcija $m(\mathbf{t})$ je funkcija povprečja, funkcija $k(\mathbf{t}, \mathbf{t})$ pa funkcija kovariance. Večinoma funkcijo povprečja nastavimo na 0, na obliko modela pa bistveno vpliva struktura kovariance. Zapišemo

$$x(\mathbf{t}) \sim \mathcal{N}(\mathbf{0}, k(\mathbf{t}, \mathbf{t}))$$

V praksi to pomeni, da za vsak končen učni vzorec $(x(t_1), x(t_2), ..., x(t_n))$ lahko statistično sklepamo o vsaki drugi časovni točki. Ob predpostavki normalne porazdelitve tako lahko analitično izračunamo naslednjo pogojno verjetnost

$$p(x(t_*)|x(t_1), x(t_2), ..., x(t_n))$$

Za vsako časovno točko t_*. Kje je torej skrita informacija o podobnosti med primeri? V kovariančni funkciji!

8.3.2 Primer

Oglejmo si spreminjanje bruto državnega proizvoda v Združenih državah amerike med leti 1970 in 2012.

```
In [1]: data = np.genfromtxt("podatki/GDP-USD-countries.csv", delimiter=",")

        i = 205
        x = data[0, 1:]
        y = data[i, 1:]
        n = len(x)

        plt.figure()
        plt.plot(x, y)
        plt.show()
```

```
        ---------------------------------------------------------------------------

        NameError                                 Traceback (most recent call last)

        <ipython-input-1-e6615157aac7> in <module>
    ----> 1 data = np.genfromtxt("podatki/GDP-USD-countries.csv", delimiter=",")
          2
          3 i = 205
          4 x = data[0, 1:]
          5 y = data[i, 1:]


        NameError: name 'np' is not defined
```

Uporabili bomo tipično funkcijo kovariance, eksponentno-kvadratno funkcijo (ang. "Exponentiated-quadratic" oz. "RBF"), dano z izrazom

$$k(t, t') = exp\{-\frac{\|t - t'\|^2}{2\ell^2}\}$$

kjer parametru ℓ pravimo dolžina vpliva (ang. lengthscale).

In [2]:
```
resolution = 10

t = np.linspace(-5, 5, resolution)
x = np.sin(t)    * np.cos(2*t) + 0.2*np.random.rand(1, resolution).ravel()

t = t.reshape((len(t), 1))[0::1]
x = x.reshape((len(x), 1))[0::1]

x = x - x.mean()

# Gaussian kernel, RBF, sq exp, exponentiated quadratic, stationary kernel
kernel = GPy.kern.RBF(input_dim=1, lengthscale=1)
model = GPy.models.GPRegression(t, x, kernel, noise_var=0.1) # noise_var=10.0

# model.optimize(messages=True)
model.plot(lower=5, upper=95)
plt.gca().set_xlabel("t")
plt.gca().set_ylabel("GDP($)")
plt.xlim(-5, 10)
plt.show()
```

```
---------------------------------------------------------------------------

NameError                                 Traceback (most recent call last)

<ipython-input-2-0a9721c61fe7> in <module>
      1 resolution = 10
      2
----> 3 t = np.linspace(-5, 5, resolution)
      4 x = np.sin(t)    * np.cos(2*t) + 0.2*np.random.rand(1, resolution).ravel()
      5


NameError: name 'np' is not defined
```

Dobimo model neparametrične regresije, ki nam omogoča ekstrapolacijo v naslednja leta. Opazimo, da se negotovost (varianca) napovedi povečuje, s tem ko se oddaljujemo od podatkov.

8.3.3 Kovariančne funkcije

Kovariačne funkcije bistveno vplivajo na obliko družine funkcij, ki jih vzorčimo iz Gaussovega Procesa. Oglejmo si nekaj tipičnih primerov kovariačnih funkcij. Bodite pozorni na lastnosti družine funkcij.

In [3]:
```
kernels = [ GPy.kern.Linear, GPy.kern.RBF, GPy.kern.Brownian, GPy.kern.PeriodicExponential, GPy
names = ["linear", "exp", "brownian", "per_exp", "poly"]
```

```
fig, axes = plt.subplots(nrows=2, ncols=5, figsize=(14, 4))
for i, k, name in zip(range(5), kernels, names):

    # Narisi funkcijo kovariance
    knl = k(input_dim=1)
    knl.plot(x=1, ax=axes[0][i])
    axes[0][i].set_xlabel("t, t'")
    axes[0][i].set_ylabel("k(t, t')")
    axes[0][i].set_title(name)

    # Narisi vzorce iz druzine funkcij
    X = np.linspace(0, 10, 100).reshape((100, 1))
    mu = np.zeros((100, ))
    C = knl.K(X,X)
    Z = np.random.multivariate_normal(mu,C,5)
    for z in Z:
        axes[1][i].plot(z)

fig.tight_layout()
plt.show()
```

```
---------------------------------------------------------------------------

NameError                                 Traceback (most recent call last)

<ipython-input-3-0549116d34f6> in <module>
----> 1 kernels = [ GPy.kern.Linear, GPy.kern.RBF, GPy.kern.Brownian, GPy.kern.PeriodicExponential,
      2 names = ["linear", "exp", "brownian", "per_exp", "poly"]
      3 
      4 fig, axes = plt.subplots(nrows=2, ncols=5, figsize=(14, 4))
      5 for i, k, name in zip(range(5), kernels, names):


NameError: name 'GPy' is not defined
```

Oglejmo si nekoliko bolj zanimiv signal. Spodnji podatki prikazujejo koncentracijo ogljikovega diosksida (CO_2) v ozračju od leta 1960.

```
In [4]: co2 = np.genfromtxt("podatki/co2.csv", delimiter=",", skip_header=1)


        n = len(co2)

        t = co2[:, 0].reshape((n, 1))
        x = co2[:, 2].reshape((n, 1))
        x = x - x.mean()
        co2
```

```
---------------------------------------------------------------------------

NameError                                 Traceback (most recent call last)
```

```
    <ipython-input-4-0a9320767844> in <module>
----> 1 co2 = np.genfromtxt("podatki/co2.csv", delimiter=",", skip_header=1)
      2
      3
      4 n = len(co2)
      5


    NameError: name 'np' is not defined
```

Opazimo sezonsko periodično spreminjanje signala, v kombinaciji z naraščajočim trendom.

```
In [5]: plt.figure(figsize=(10, 4))
        plt.plot(t, x, ".")
        plt.ylabel("$CO_2$")
        plt.xlabel("t (mesec)")
        plt.show()


    ---------------------------------------------------------------------------

    NameError                                 Traceback (most recent call last)

    <ipython-input-5-f3205e50f927> in <module>
----> 1 plt.figure(figsize=(10, 4))
      2 plt.plot(t, x, ".")
      3 plt.ylabel("$CO_2$")
      4 plt.xlabel("t (mesec)")
      5 plt.show()


    NameError: name 'plt' is not defined
```

Naredi sam/a. S seštevanjem kovariančnih funkcij poizkušaj modelirati podatke o koncetraciji CO_2. Poizkusi najto kombinacijo funkcij, ki najbolje ekstrapolirajo koncentracijo CO_2 v prihodnja leta.

```
In [6]: kernel = GPy.kern.RBF(1, lengthscale=1)
        model  = GPy.models.GPRegression(t, x, kernel, noise_var=0.1)
        model.optimize(messages=True)
        model.plot()
        plt.gca().set_xlim(200, 600)


    ---------------------------------------------------------------------------

    NameError                                 Traceback (most recent call last)

    <ipython-input-6-be96130dff9b> in <module>
----> 1 kernel = GPy.kern.RBF(1, lengthscale=1)
      2 model  = GPy.models.GPRegression(t, x, kernel, noise_var=0.1)
      3 model.optimize(messages=True)
      4 model.plot()
      5 plt.gca().set_xlim(200, 600)
```

```
NameError: name 'GPy' is not defined
```

Odgovori

1.1 Knjižnica numpy

Odgovor 1-1-1

```
In [1]: import numpy as np

        X = np.array([
            [[1, 2, 3, 4], [2, 3, 4, 5]],
            [[3, 4, 5, 6], [4, 5, 6, 7]],
            [[5, 6, 7, 8], [6, 7, 8, 9]]
        ])
        X

Out[1]: array([[[1, 2, 3, 4],
                [2, 3, 4, 5]],

               [[3, 4, 5, 6],
                [4, 5, 6, 7]],

               [[5, 6, 7, 8],
                [6, 7, 8, 9]]])

In [2]: X.shape

Out[2]: (3, 2, 4)

In [3]: X.size

Out[3]: 24
```

Odgovor 1-1-2

```
In [4]: A = np.array([[n+m*10 for n in range(5)] for m in range(5)])
        A

Out[4]: array([[ 0,  1,  2,  3,  4],
               [10, 11, 12, 13, 14],
               [20, 21, 22, 23, 24],
               [30, 31, 32, 33, 34],
               [40, 41, 42, 43, 44]])

In [5]: A[A[:, 0]>10, 0:2]
```

```
Out[5]: array([[20, 21],
               [30, 31],
               [40, 41]])

In [6]: A[:, A[0, :]>2]

Out[6]: array([[ 3,  4],
               [13, 14],
               [23, 24],
               [33, 34],
               [43, 44]])
```

1.2 Primer: statistika temperatur na severu

Naložimo podatke o dnevnih temperaturah v Stockholmu.

```
In [1]: import numpy as np

        data = np.loadtxt('podatki/stockholm.csv', delimiter=",", skiprows=1)
```

Odgovor 1-2-1

```
In [2]: data[(data[:, 0] == 1817) * (data[:, 1] == 12) * (data[:, 2] == 5), :]

Out[2]: array([[ 1817. ,    12. ,     5. ,    -5.8]])
```

Odgovor 1-2-2

```
In [3]: np.mean(data[(data[:, 1] == 1), 3])

Out[3]: -3.0447656725502132
```

Odgovor 1-2-3

```
In [4]: odkloni = [(np.std(data[(data[:, 1] == mesec), 3]), mesec) for mesec in range(1, 13)]
        odkloni

Out[4]: [(4.9892658658329561, 1),
         (5.0903907687662713, 2),
         (4.2923064618199263, 3),
         (3.76783651629394, 4),
         (4.029747854809286, 5),
         (3.5320797349808082, 6),
         (2.995916472129954, 7),
         (2.8473127640895139, 8),
         (3.0389674027350599, 9),
         (3.4875394481813999, 10),
         (3.8200293557907226, 11),
         (4.5026210415550008, 12)]

In [5]: max(odkloni)

Out[5]: (5.0903907687662713, 2)
```

Največji odklon znaša 5.1 C, pojavi se v februarju.

Odgovor 1-2-4

```
In [6]: povprečja = []
        for leto in range(1800, 2012):
            for mesec in range(1, 13):
                t = np.mean(data[(data[:, 1] == mesec) * (data[:, 0] == leto), 3])
                povprečja.append((t, (leto, mesec)))
        max(povprečja)

Out[6]: (21.532258064516132, (1994, 7))
```

Odgovor 1-2-5

```
In [7]: # Izračunajmo povprečno temperaturo za vsako leto posebej
        letna_povprečja = dict()

        for leto in range(1800, 2012):
            # Uporabimo pogojno naslavljanje polja
            letna_povprečja[leto] = data[data[:, 0] == leto, 3].mean()
```

```
In [8]: # Izpiši vsako leto, ki ima večjo povprečno temperaturo od prejšnjega
        leto_t = sorted(letna_povprečja.items())
        večji_od_lani = [leto_t[i][0] for i in range(1, len(leto_t)) if leto_t[i-1][1] < leto_t[i][1]]
        večji_od_lani[-10:]  # izpišimo le nekaj letnic
```

```
Out[8]: [1992, 1994, 1997, 1999, 2000, 2002, 2005, 2006, 2008, 2011]
```

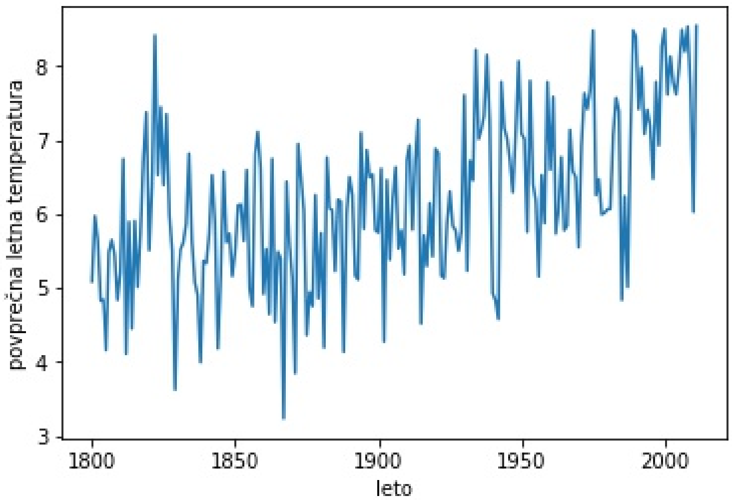
```
In [9]: # Poišči 10 najtoplejših let
        t_leto = sorted(((t, leto) for leto, t in leto_t))
        t_leto[-10:]
```

```
Out[9]: [(8.2189041095890421, 1934),
         (8.2657534246575342, 1999),
         (8.3997260273972589, 1990),
         (8.4134246575342466, 1822),
         (8.4797260273972608, 1975),
         (8.4808219178082194, 1989),
         (8.4882191780821916, 2006),
         (8.4978142076502738, 2000),
         (8.5330601092896181, 2008),
         (8.5394520547945199, 2011)]
```

Odgovor 1-2-6

```
In [10]: %matplotlib inline
         %config InlineBackend.figure_formats = ['jpg']
         import matplotlib
         matplotlib.figure.Figure.__repr__ = lambda self: (
             f"<{self.__class__.__name__} size {self.bbox.size[0]:g}"
             f"x{self.bbox.size[1]:g} with {len(self.axes)} Axes>")

         import matplotlib.pyplot as plt
```

```
In [11]: # Pomagaj si s letna_povprečja.
         # Os x: leto
         # Os y: povprečna letna temperatura

         plt.figure()

         # Narišimo izvirne podatke
         leta, temperature = zip(*sorted(letna_povprečja.items()))
         plt.plot(leta, temperature)

         # Vedno označimo osi.
         plt.xlabel("leto")
         plt.ylabel("povprečna letna temperatura")
         plt.show()
```



povprečna letna temperatura
leto
3
4
5
6
7
8
1800
1850
1900
1950
2000

2.1 Library matplotlib

```
In [1]: %matplotlib inline
        import matplotlib
        %config InlineBackend.figure_format = 'jpg'
        matplotlib.figure.Figure.__repr__ = lambda self: (
            f"<{self.__class__.__name__} size {self.bbox.size[0]:g}"
            f"x{self.bbox.size[1]:g} with {len(self.axes)} Axes>")
        import matplotlib.pyplot as plt
        plt.style.use('PR.mplstyle')

        import numpy as np
        np.random.seed(42)
```

```
        ---------------------------------------------------------------------------

        FileNotFoundError                         Traceback (most recent call last)

        ~/anaconda3/lib/python3.6/site-packages/matplotlib/style/core.py in use(style)
        112             try:
    --> 113                 rc = rc_params_from_file(style, use_default_template=False)
        114                 _apply_style(rc)


        ~/anaconda3/lib/python3.6/site-packages/matplotlib/__init__.py in rc_params_from_file(fname, fa
       1028     """
    -> 1029     config_from_file = _rc_params_in_file(fname, fail_on_error)
       1030


        ~/anaconda3/lib/python3.6/site-packages/matplotlib/__init__.py in _rc_params_in_file(fname, fai
        944     rc_temp = {}
    --> 945     with _open_file_or_url(fname) as fd:
        946         try:


        ~/anaconda3/lib/python3.6/contextlib.py in __enter__(self)
         80         try:
    ---> 81             return next(self.gen)
         82         except StopIteration:


        ~/anaconda3/lib/python3.6/site-packages/matplotlib/__init__.py in _open_file_or_url(fname)
        929             encoding = "utf-8"
    --> 930         with io.open(fname, encoding=encoding) as f:
        931             yield f


        FileNotFoundError: [Errno 2] No such file or directory: 'PR.mplstyle'


    During handling of the above exception, another exception occurred:
```

```
OSError                                   Traceback (most recent call last)

<ipython-input-1-4709b06cc9e5> in <module>
      6     f"x{self.bbox.size[1]:g} with {len(self.axes)} Axes>")
      7 import matplotlib.pyplot as plt
----> 8 plt.style.use('PR.mplstyle')
      9
     10 import numpy as np


~/anaconda3/lib/python3.6/site-packages/matplotlib/style/core.py in use(style)
    117                          "not a valid URL or path. See `style.available` for "
    118                          "list of available styles.")
--> 119                 raise IOError(msg % style)
    120
    121


OSError: 'PR.mplstyle' not found in the style library and input is not a valid URL or path. See
```

Answer 2-1-1

```
In [2]: plt.figure()
        x = np.linspace(1, 5, 10)

        for exp in range(-3, 4):
            y = x ** exp
            plt.plot(x, y)

        plt.xlabel('x')
        plt.ylabel('y');
```

```
---------------------------------------------------------------------------

NameError                                 Traceback (most recent call last)

<ipython-input-2-ad75e5174fb6> in <module>
      1 plt.figure()
----> 2 x = np.linspace(1, 5, 10)
      3
      4 for exp in range(-3, 4):
      5     y = x ** exp


NameError: name 'np' is not defined

<Figure size 432x288 with 0 Axes>
```

Answer 2-1-2

```
In [3]: N = 10000
        data = np.random.randn(N)

        plt.hist(data, bins=25);

        ---------------------------------------------------------------------------

        NameError                                 Traceback (most recent call last)

        <ipython-input-3-203c78b08417> in <module>
          1 N = 10000
    ----> 2 data = np.random.randn(N)
          3
          4 plt.hist(data, bins=25);


        NameError: name 'np' is not defined
```

Answer 2-1-3

```
In [4]: data = np.random.randn(1000) / 2 + 5
        plt.hist(data, bins=10);

        ---------------------------------------------------------------------------

        NameError                                 Traceback (most recent call last)

        <ipython-input-4-4c0984270b17> in <module>
    ----> 1 data = np.random.randn(1000) / 2 + 5
          2 plt.hist(data, bins=10);


        NameError: name 'np' is not defined
```

2.2 Example: Winter Olympics, Sochi 2014

In [1]:
```
%matplotlib inline
%config InlineBackend.figure_formats = ['jpg']
import matplotlib
matplotlib.figure.Figure.__repr__ = lambda self: (
    f"<{self.__class__.__name__} size {self.bbox.size[0]:g}"
    f"x{self.bbox.size[1]:g} with {len(self.axes)} Axes>")

import matplotlib.pyplot as plt

import Orange
from Orange.data.filter import SameValue
from Orange.data import Table
data = Table('podatki/athletes.tab')

# barve medalj
gold_color = "#FFDF00"
silv_color = "#C0C0C0"
bron_color = "#CD7F32"

sports = data.domain["sport"].values
```

Answer 2-2-1

In [2]:

Answer 2-2-2

In [2]:

Answer 2-2-3

In [2]:
```
import numpy as np
def pearson(x, y):
    return np.mean(((x - np.mean(x))*(y-np.mean(y)))/(np.std(x)*np.std(y)))
```

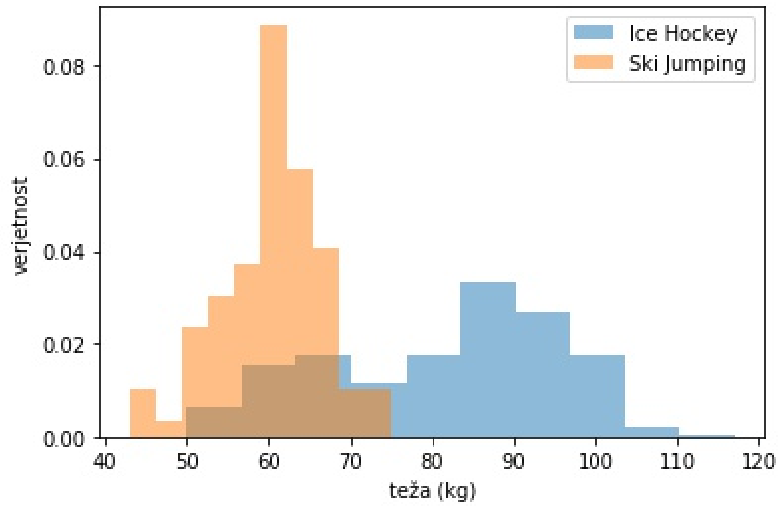
In [3]:
```
x = data.X[:, 2]    # višina
y = data.X[:, 3]    # teža
pearson(x, y)
```

Out[3]: 0.83074658486272401

Answer 2-2-4

In [4]:
```
filt = SameValue(data.domain['sport'], 'Ice Hockey')
data_subset = filt(data)
weights = data_subset.X[:, 3]
plt.hist(weights, normed=True, bins=10, label="Ice Hockey", alpha=0.5)

filt = SameValue(data.domain['sport'], 'Ski Jumping')
data_subset = filt(data)
weights = data_subset.X[:, 3]
plt.hist(weights, normed=True, bins=10, label="Ski Jumping", alpha=0.5)
```

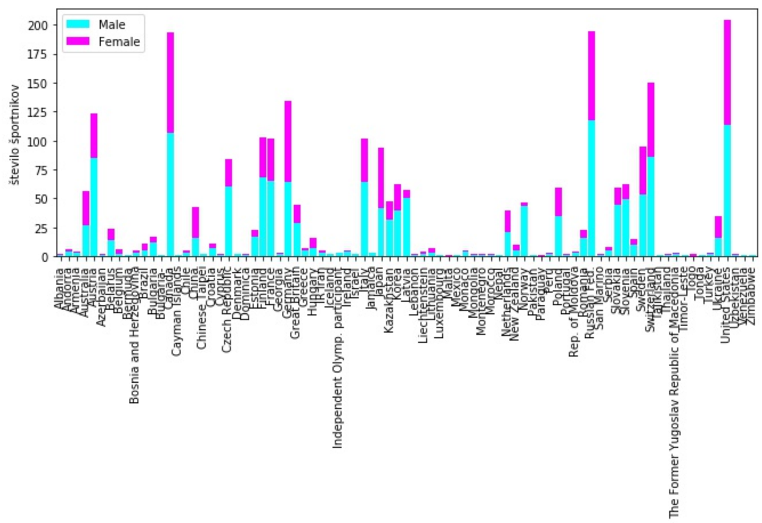
```
plt.xlabel('teža (kg)')
plt.ylabel('verjetnost')
plt.legend();
```



Answer 2-2-5

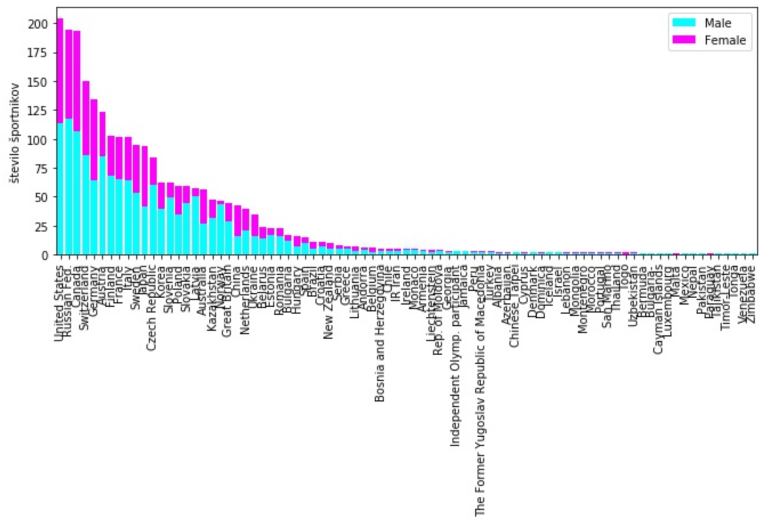
```
In [5]: countries = data.domain['country'].values
        gender_by_country = dict()

        for country in countries:
            # Filter by countries
            filt        = SameValue(data.domain['country'], country)
            data_subset = filt(data)

            # Filter males
            filt        = SameValue(data.domain['gender'], 'Male')
            data_subset_male = filt(data_subset)

            # Filter females
            filt        = SameValue(data.domain['gender'], 'Female')
            data_subset_female = filt(data_subset)

            # Store gender counts
            gender_by_country[country] = {
                'Male': len(data_subset_male),
                'Female': len(data_subset_female),
            }

In [6]: m = [gender_by_country[country]['Male'] for country in countries]
        f = [gender_by_country[country]['Female'] for country in countries]
        x = range(len(countries))

        plt.figure(figsize=(11, 4))
        plt.bar(x, m, color='cyan', align='center', label="Male")
        plt.bar(x, f, bottom=m, color='magenta', align='center', label="Female")
        plt.xlim(-0.5, len(countries)-0.5)
```

```
plt.xticks(x)
plt.gca().set_xticklabels(countries, rotation=90)
plt.ylabel('število športnikov')

plt.legend();
```



Answer 2-2-6

```
In [7]: countries = filter(lambda c: sum([gender_by_country[c][m] for m in gender_by_country[c].keys()]
        countries = sorted(countries, key=lambda c: -sum([gender_by_country[c][m] for m in gender_by_cou


        m = [gender_by_country[country]['Male'] for country in countries]
        f = [gender_by_country[country]['Female'] for country in countries]
        x = range(len(countries))

        plt.figure(figsize=(11, 4))
        plt.bar(x, m, color='cyan', align='center', label="Male")
        plt.bar(x, f, bottom=m, color='magenta', align='center', label="Female")
        plt.xlim(-0.5, len(countries)-0.5)
        plt.xticks(x)
        plt.gca().set_xticklabels(countries, rotation=90)
        plt.ylabel('število športnikov')

        plt.legend();
```



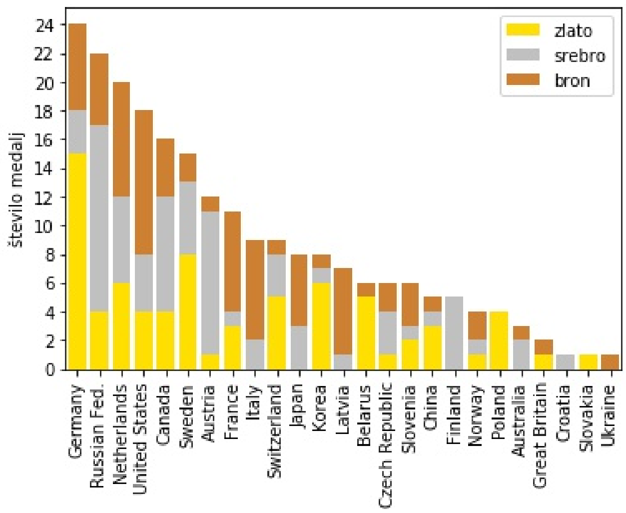
Answer 2-2-7 We first calculate the distribution of the values.

```
In [8]: # poišči indekse
        gold_inx = data.domain.index("gold_medals")
        silv_inx = data.domain.index("silver_medals")
        bron_inx = data.domain.index("bronze_medals")

        # pripravi podatke ; shrani št. medalj za vsako državo in šport
        countries = data.domain["country"].values


        # preštej medalje
        medals_by_country = dict()
        for country in countries:
            medals_by_country[country] = dict()
            filt        = SameValue(data.domain["country"], country)
            data_subset = filt(data)
            medals_by_country[country] = {
                "gold":   data_subset.X[:, gold_inx].sum(),
                "silver": data_subset.X[:, silv_inx].sum(),
                "bronze": data_subset.X[:, bron_inx].sum(),
        }
```

Then we draw the distribution.

```
In [9]: import numpy

        countries = filter(lambda c: sum([medals_by_country[c][m] for m in medals_by_country[c].keys()])
        countries = sorted(countries, key=lambda c: -sum([medals_by_country[c][m] for m in medals_by_cou
        gx = numpy.array([medals_by_country[c]["gold"] for c in countries])
        sx = numpy.array([medals_by_country[c]["silver"] for c in countries])
        bx = numpy.array([medals_by_country[c]["bronze"] for c in countries])
        x = range(len(countries))

        plt.bar(x, gx, align="center", color=gold_color, label="zlato")
        plt.bar(x, sx, align="center", bottom=gx, color=silv_color, label="srebro")
        plt.bar(x, bx, align="center", bottom=gx+sx, color=bron_color, label="bron")
```

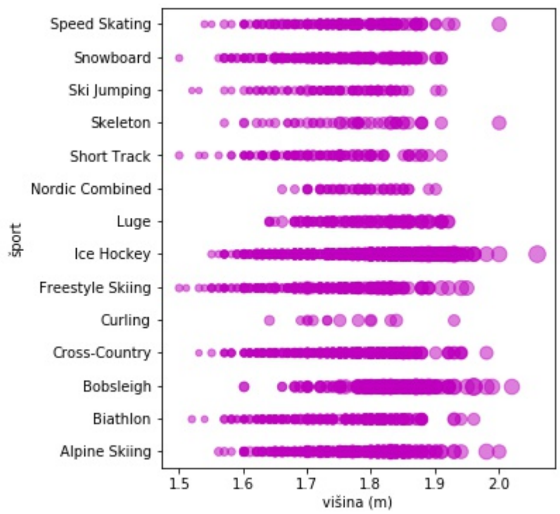
```
plt.xlim(-0.5, len(x)-0.5)
plt.legend()
plt.xticks(x)
plt.yticks(range(0, 25, 2))
plt.gca().set_xticklabels(countries, rotation=90)
plt.ylabel("število medalj")
plt.savefig('slike/odgovori/2-2-7.png', bbox_inches='tight')
```



Answer 2-2-8 Preparation of data - weight and height according to the sport (the sport is located in the 8th column)

```
In [10]: sports = data.domain["sport"].values
         weights_by_sport = dict()
         heights_by_sport = dict()
         ages_by_sport = dict()

         for sport in sports:
             filt = SameValue(data.domain["sport"], sport)
             data_subset = filt(data)

             w = data_subset[:, data.domain.index("weight")].X.ravel()
             h = data_subset[:, data.domain.index("height")].X.ravel()
             a = data_subset[:, data.domain.index("age")].X.ravel()

             weights_by_sport[sport] = w
             heights_by_sport[sport] = h
             ages_by_sport[sport]    = a

In [11]: plt.figure(figsize=(5, 6))

         for si, sport in enumerate(sports):
             xs = heights_by_sport[sport]    # x os
             ys = [si for x in xs]           # y os je v visini sporta
             zs = weights_by_sport[sport]    # velikost točke je premosorazmerna s tezo

             for x, y, z in zip(xs, ys, zs): # rišemo točko po točko
                 plt.plot(x, y, "m.", alpha=0.5, markersize=z/5)
```

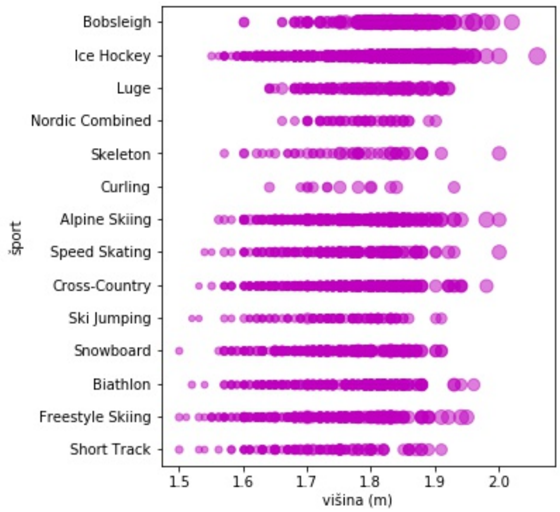
```
plt.yticks(range(len(sports)))
plt.ylim(-0.5, len(sports)-0.5)
plt.gca().set_yticklabels(sports)

plt.xlabel("višina (m)")
plt.ylabel("šport");
plt.savefig('slike/odgovori/2-2-8.png', bbox_inches='tight')
```



Answer 2-2-9

```
In [12]: plt.figure(figsize=(5, 6))

         sport_order = []
         for si, sport in enumerate(sports):
             xs = heights_by_sport[sport]     # x os
             sport_order.append((numpy.average(xs), si))
         sport_order.sort()

         sport_label = []
         for nsi, (avg_xs, si) in enumerate(sport_order):
             sport = sports[si]
             sport_label.append(sport)

             xs = heights_by_sport[sport]     # x os
             ys = [nsi for x in xs]           # y os je v visini sporta
             zs = weights_by_sport[sport]     # velikost točke je premosorazmerna s tezo

             for x, y, z in zip(xs, ys, zs): # rišemo točko po točko
                 plt.plot(x, y, "m.", alpha=0.5, markersize=z/5)

             plt.plot(avg_xs, nsi, 'k', markersize=1)

         plt.yticks(range(len(sports)))
         plt.ylim(-0.5, len(sports)-0.5)
         plt.gca().set_yticklabels(sport_label)
```

```
plt.xlabel("višina (m)")
plt.ylabel("šport");
```



3.1 Pogoste verjetnostne porazdelitve

Odgovor 3-1-1

```
In [1]:
```

Odgovor 3-1-2

```
In [1]:
```

Odgovor 3-1-3

```
In [1]:
```

Odgovor 3-1-4

```
In [1]:
```

3.2 Primer: iskanje neslanih šal

Odgovor 3-2-1

```
In [1]:
```

Odgovor 3-2-2

```
In [1]:
```

Odgovor 3-2-3

```
In [1]:
```

Odgovor 3-2-4

```
In [1]:
```

4.1 Group detection

Answer 4-1-1

```
In [1]: import numpy as np
        np.random.seed(42)

        class KMeans:

            def __init__(self, k=10, max_iter=100):
                """
                Initialize KMeans clustering model.

                :param k
                    Number of clusters.
                :param max_iter
                    Maximum number of iterations.
                """
                self.k         = k
                self.max_iter  = max_iter

            def fit(self, X):
                """
                Fit the Kmeans model to data.

                :param X
                    Numpy array of shape (n, p)
                    n: number of data examples
                    p: number of features (attributes)

                :return
                    labels: array of shape (n, ), cluster labels (0..k-1)
                    centers: array of shape (p, )
                """

                n, p    = X.shape
                labels  = np.random.choice(range(self.k), size=n, replace=True)
                centers = np.random.rand(self.k, p)

                ### Your code here ###
                centers = np.min(X, axis=0) + centers * (np.max(X, axis=0) - np.min(X, axis=0))

                i = 0
                while i < self.max_iter:

                    # Find nearest cluster
                    for j, x in enumerate(X):
                        ki = np.argmin(np.sum((centers - x) ** 2, axis=1))
                        labels[j] = ki

                    # Move centroid
                    for ki in range(self.k):
                        centers[ki] = X[labels == ki].mean(axis=0)
                    i += 1
```

```
    ### Your code here ###
    return labels, centers
```

Answer 4-1-2

```
In [2]:
```

Answer 4-1-3

```
In [2]:
```

Answer 4-1-4

```
In [2]:
```

Answer 4-1-5

```
In [2]:
```

Answer 4-1-6

```
In [2]:
```

4.2 Hierarhično gručenje

Answer 4-2-1

In [1]:

Answer 4-2-2

In [1]:

Answer 4-2-3

In [1]:

4.3 Example: genomic data in the form of character strings

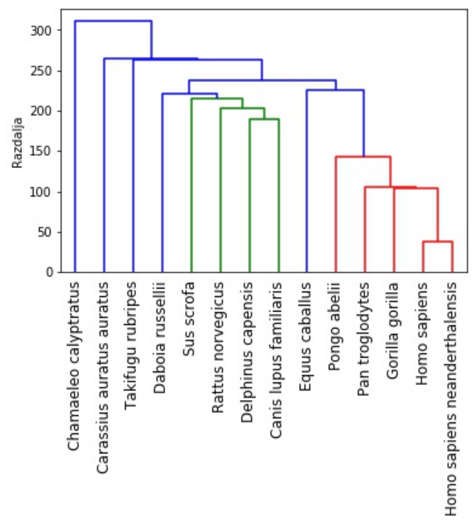
```
In [1]: import json
        sequences = json.load(open("podatki/seqs.json"))
```

Answer 4-3-1

```
In [2]: from itertools import product
        import numpy as np
        import scipy.cluster.hierarchy as sch
        %matplotlib inline
        %config InlineBackend.figure_formats = ['jpg']
        import matplotlib
        matplotlib.figure.Figure.__repr__ = lambda self: (
            f"<{self.__class__.__name__} size {self.bbox.size[0]:g}"
            f"x{self.bbox.size[1]:g} with {len(self.axes)} Axes>")

        import matplotlib.pyplot as plt

        def seq_to_kmer_count(seq, k=4):
            ktuples = list(zip(*[seq[i:] for i in range(k)]))      # razbijemo niz na k-terke
            kmers   = list(product(*(k*[["A", "C", "T", "G"]])))  # vse mozne k-terke

            x = np.zeros((len(kmers), ))

            for ki, kmer in enumerate(kmers):
                x[ki] = ktuples.count(kmer)
            return x
```

```
In [3]: # ...k = 4
        k = 4
        keys = sequences.keys()
        X    = np.zeros((len(keys), 4**k))
        for ki, ky in enumerate(keys):
            seq    = sequences[ky]
            X[ki]  = seq_to_kmer_count(seq, k=k)

        print(X)
        print(X.shape)
        H = sch.linkage(X)
        D = sch.dendrogram(H, labels=list(sequences.keys()), leaf_rotation=90)
        plt.ylabel("Razdalja")
        plt.show()
```

```
[[ 182.  157.  110. …,   22.   18.   15.]
 [ 187.  149.  120. …,   14.   13.   12.]
 [ 174.  159.  124. …,   18.   13.   14.]
 …,
 [ 158.  125.  120. …,   22.   31.   27.]
 [ 238.  160.  158. …,   12.   18.   14.]
 [ 184.  156.  110. …,   25.   18.   19.]]
(14, 256)
```


Razdalja
300
250
200
150
100
50
0
Chamaeleo calyptratus
Carassius auratus auratus
Takifugu rubripes
Daboia russellii
Sus scrofa
Rattus norvegicus
Delphinus capensis
Canis lupus familiaris
Equus caballus
Pongo abelii
Pan troglodytes
Gorilla gorilla
Homo sapiens
Homo sapiens neanderthalensis

5.1 Linear regression

Answer 5-1-1

```
In [1]: explained_var = 100.0 * ( np.var(y) - np.var(hx-y) ) /  np.var(y)
        print("Explained variance: %.2f " % explained_var + "%" )
```

```
---------------------------------------------------------------------------

NameError                                 Traceback (most recent call last)

<ipython-input-1-d7f1e86fae41> in <module>
----> 1 explained_var = 100.0 * ( np.var(y) - np.var(hx-y) ) /  np.var(y)
      2 print("Explained variance: %.2f " % explained_var + "%" )


NameError: name 'np' is not defined
```

Answer 5-1-2

```
In [2]: explained_var = 100.0 * ( np.var(y) - np.var(hx-y) ) /  np.var(y)
        print("Explained variance: %.2f " % explained_var + "%" )
```

```
---------------------------------------------------------------------------

NameError                                 Traceback (most recent call last)

<ipython-input-2-d7f1e86fae41> in <module>
----> 1 explained_var = 100.0 * ( np.var(y) - np.var(hx-y) ) /  np.var(y)
      2 print("Explained variance: %.2f " % explained_var + "%" )


NameError: name 'np' is not defined
```

Answer 5-1-3

```
In [3]: D = 20 # stopnja polinoma

        # Ustvarimo ustrezen prostor
        X = np.zeros((len(x), D))
        for d in range(0, D):
            X[:, d] = x.ravel()**d

        model = Ridge(alpha=0.1)
        model.fit(X, y)

        hx = model.predict(X)

        plot_fit_residual(X[:, 1], y, hx)
        plot_coefficients(model.coef_)
        model.coef_
```

```
---------------------------------------------------------------------------

NameError                                 Traceback (most recent call last)

<ipython-input-3-8e0a57f74d20> in <module>
      2
      3 # Ustvarimo ustrezen prostor
----> 4 X = np.zeros((len(x), D))
      5 for d in range(0, D):
      6     X[:, d] = x.ravel()**d


NameError: name 'np' is not defined
```

Answer 5-1-4

```
In [4]: explained_var = 100.0 * ( np.var(y) - np.var(hx-y) ) /  np.var(y)
        print("Explained variance: %.2f " % explained_var + "%" )
```

```
---------------------------------------------------------------------------

NameError                                 Traceback (most recent call last)

<ipython-input-4-d7f1e86fae41> in <module>
----> 1 explained_var = 100.0 * ( np.var(y) - np.var(hx-y) ) /  np.var(y)
      2 print("Explained variance: %.2f " % explained_var + "%" )


NameError: name 'np' is not defined
```

Answer 5-1-5

```
In [5]: model = Lasso(alpha=0.1)
        model.fit(X, y)

        hx = model.predict(X_test)

        print("MSE: %.2f " %mean_squared_error(hx, y_test))
        explained_var = 100.0 * ( np.var(y_test) - np.var(hx-y_test) ) /  np.var(y_test)
        print("Explained variance: %.2f" % explained_var + "%" )
```

```
---------------------------------------------------------------------------

NameError                                 Traceback (most recent call last)

<ipython-input-5-748a51cac842> in <module>
----> 1 model = Lasso(alpha=0.1)
      2 model.fit(X, y)
      3
      4 hx = model.predict(X_test)
```

```
  5

NameError: name 'Lasso' is not defined
```

Answer 5-1-6

In [6]:

5.2 Naivni Bayesov klasifikator

```
In [1]: class NaiveBayes:
            """
            Naive Bayes classifier.

            :attribute self.probabilities
                Dictionary that stores
                    - prior class probabilities P(Y)
                    - attribute probabilities conditional on class P(X|Y)

            :attribute self.class_values
                All possible values of the class.

            :attribute self.variables
                Variables in the data.

            :attribute self.trained
                Set to True after fit is called.
            """

            def __init__(self):
                self.trained       = False
                self.probabilities = dict()


            def fit(self, data):
                """
                Fit a NaiveBayes classifier.

                :param data
                    Orange data Table.
                """
                class_variable      = data.domain.class_var    # class variable (Y)
                self.class_values   = class_variable.values    # possible class values
                self.variables      = data.domain.attributes    # all other variables (X)

                n = len(data) # number of all data points

                # Compute P(Y)
                for y in self.class_values:

                    # A not too smart guess (INCORRECT)
                    self.probabilities[y] = 1/len(self.class_values)

                    # <your code here>
                    # Compute class probabilities and correctly fill
                    #   probabilities[y] = ...
                    # Select all examples (rows) with class = y
                    filt = SameValue(data.domain.class_var, y)
                    data_subset = filt(data)
                    m = len(data_subset)

                    self.probabilities[y] = m/n
```

```
            # </your code here>

        # Compute P(X|Y)
        for y in self.class_values:

            # Select all examples (rows) with class = y
            filty = SameValue(class_variable, y)

            for variable in self.variables:
                for x in variable.values:

                    # A not too smart guess (INCORRECT)
                    p = 1 / (len(self.variables) * len(variable.values) * len(self.class_values)

                    # P(variable=x|Y=y)
                    self.probabilities[variable, x, y] = p

                    # <your code here>
                    # Compute correct conditional class probability
                    #   probabilities[x, value, c] = ...
                    #
                    # Select all examples with class == y AND
                    # variable x == value
                    # Hint: use SameValue filter twice
                    filtx = SameValue(variable, x)
                    data_subset = filtx(filty(data))
                    m = len(data_subset)

                    data_subset = filty(data)
                    p = len(data_subset)

                    self.probabilities[variable, x, y] = m/p
                    # </your code here>

        self.trained = True

    def predict_instance(self, row):
        """
        Predict a class value for one row.

        :param row
            Orange data Instance.
        :return
            Class prediction.
        """
        curr_p = float("-inf")   # Current highest "probability" (unnormalized)
        curr_c = None            # Current most probable class

        for y in self.class_values:
            p = np.log(self.probabilities[y])
            for x in self.variables:
                p = p + np.log(self.probabilities[x, row[x].value, y])

            if p > curr_p:
```

```
                curr_p = p
                curr_c = y

        return curr_c, curr_p

    def predict(self, data):
        """
        Predict class labels for all rows in data.

        :param data
            Orange data Table.
        :return y
            NumPy vector with predicted classes.
        """

        n = len(data)
        predictions = list()
        confidences = np.zeros((n, ))

        for i, row in enumerate(data):
            pred, cf = self.predict_instance(row)
            predictions.append(pred)
            confidences[i] = cf

        return predictions, confidences
```

6 Nenegativna matrična faktorizacija in priporočilni sistemi

In [1]:
```
import numpy as np
import itertools
import time

np.random.seed(42)

class NMF:

    """
    Fit a matrix factorization model for a matrix X with missing values.
    such that
        X = W H.T + E
    where
        X is of shape (m, n)    - data matrix
        W is of shape (m, rank) - approximated row space
        H is of shape (n, rank) - approximated column space
        E is of shape (m, n)    - residual (error) matrix
    """

    def __init__(self, rank=10, max_iter=100, eta=0.01):
        """
        :param rank: Rank of the matrices of the model.
        :param max_iter: Maximum nuber of SGD iterations.
        :param eta: SGD learning rate.
        """
        self.rank = rank
        self.max_iter = max_iter
        self.eta = eta


    def fit(self, X, verbose=False):
        """
        Fit model parameters W, H.
        :param X:
            Non-negative data matrix of shape (m, n)
            Unknown values are assumed to take the value of zero (0).
        """
        m, n = X.shape

        W = np.random.rand(m, self.rank)
        H = np.random.rand(n, self.rank)

        # Indices to model variables
        w_vars = list(itertools.product(range(m), range(self.rank)))
        h_vars = list(itertools.product(range(n), range(self.rank)))

        # Indices to nonzero rows/columns
        nzcols = dict([(j, X[:, j].nonzero()[0]) for j in range(n)])
        nzrows = dict([(i, X[i, :].nonzero()[0]) for i in range(m)])

        # Errors
        self.error = np.zeros((self.max_iter,))
```

```
        for t in range(self.max_iter):
            t1 = time.time()
            np.random.shuffle(w_vars)
            np.random.shuffle(h_vars)

            for i, k in w_vars:
                wgrad   = sum([(X[i, j] - W[i, :].dot(H[j, :]))*W[i, k] for j in nzrows[i]])
                W[i, k] = max(0, W[i, k] + self.eta * wgrad)

            for j, k in h_vars:
                hgrad   = sum([(X[i, j] - W[i, :].dot(H[j, :]))*H[j, k] for i in nzcols[j]])
                H[j, k] = max(0, H[j, k] + self.eta * hgrad)

            self.error[t] = sum([sum([(X[i, j] - W[i, :].dot(H[j, :]))**2 for j in nzrows[i]])
                                 for i in range(X.shape[0])])

            if verbose: print(t, self.error[t])

        self.W = W
        self.H = H


    def predict(self, i, j):
        """
        Predict score for row i and column j
        :param i: Row index.
        :param j: Column index.
        """
        return self.W[i, :].dot(self.H[:, j])


    def predict_all(self):
        """
        Return approximated matrix for all
        columns and rows.
        """
        return self.W.dot(self.H.T)
```

7.1 Knjižica `networkx`

7.2 Primer: analiza in vizualizacija omrežja elektronskih sporočil

8.1 Skriti Markovi modeli

In [1]:
```
import random
random.seed(42)

def weighted_choice(weighted_items):
    """Random choice given the list of elements and their weights"""
    rnd = random.random() * sum(weighted_items.values())
    for i, w in weighted_items.items():
        rnd -= w
        if rnd < 0:
            return i

def generate_hmm_sequence(h, T, E, n):
    """
    HMM sequence given start state,
    transition, emission matrix and sequence length

    return zip(hidden_path, visible_sequence)
    """

    s = weighted_choice(E[h])
    yield h, s
    for _ in range(n-1):
        h = weighted_choice(T[h])
        yield h, weighted_choice(E[h])

from collections import Counter

def normalize(dic):
    s = sum(dic.values())
    return {k: dic[k]/s for k in dic}

def learn_hmm(h, x):
    t = {}
    for (i, j), cn in Counter(zip(h, h[1:])).items():
        t.setdefault(i, {}).setdefault(j, cn)
    T = {}
    for i, d in t.items():
        T[i] = normalize(d)

    c = Counter(zip(h, x))
    E = {}
    for h in T.keys():
        E[h] = normalize({xi: c[(pi, xi)] for pi, xi in c if pi == h})
    return T, E
```

8.2 Modeliranje časovnih vrst

8.3 Neparametrična regresija ali napovedovanje trendov

Assignments

Data preparation, basic statistics and visualization

Data mining, assignment, `<INSERT DATE>`
`<INSERT Name and surname>`

An inevitable part of every project in the field of data mining is searching for, editing and preparing data. In this task, you will get acquainted with a dataset and use procedures for converting data into the appropriate format and do overview and display of basic statistics.

Data

In the task you will review and prepare Hollywood movie ratings from the MovieLens collection from the period **1995-2016**.

The same data is used in all assignments, so you should get to know the data well. This is a database for evaluating recommendations systems that include viewers and their ratings on a scale of 1 to 5. In addition to the basic user and rating matrix, it includes also movie information (e.g., genre, date, tags, players).

The dataset is in folder `./podatki/ml-latest-small`. The database contains the following files:

- ratings.csv: user data and ratings,
- movies.csv: movie genre information,
- cast.csv: player information,
- tags.csv: tag information (*tags*),
- links.csv: links to related databases.

Before starting to solve the task, take a good look at the data and read the **README.txt** file. You can learn about the details on the website.

Prepare methods for loading data into the appropriate data structures. They will come in handy also for further tasks. Pay attention to the size of the data.

Write down the code to read the files and prepare the appropriate matrices (and other structures) of the data that you will use to answer the questions below.

You can split the code into multiple cells.

```
In [ ]:
```

Questions

The main purpose of data mining is *knowledge discovery from data*, i.e., answering questions using computational approaches.

By using the principles you have learned on the exercises and lectures, answer the questions below. For each question, think carefully about the way you will best give, show or justify the answer. The essential part is the answers to questions rather than the implemention of your solution.

Question 1 (15Which movies are the best on average? Prepare a list of

movies and their average ratings and print 10 movies from the top of the list. Do you see any problems with such an assessment? How could you solve it? What are they? results of that?

You can split the code into multiple cells.

```
In [ ]:
```

Answer: **write down the answer and explain it**

Question 2 (15Each film belongs to one or more genres.

How many genres are there? Show the distribution of genres using appropriate visualization.

You can split the code into multiple cells.

In [ ]:

Answer: **write down the answer and explain it**

Question 3 (20The number of ratings is different for each film. But is there a relationship between the number of ratings and the average movie rating? Describe the procedure that you used to answer the question.

You can split the code into multiple cells.

In [ ]:

Answer: **write down the answer and explain it**

Question 4 (30Each rating was entered on a specific date (column

timestamp). Does the popularity of individual films change over time? Solve the problem by allocating ratings for a given film by time and at any time point calculate the average for the last 30, 50, or 100 ratings. Draw a graph, how the rating changes and show it for two interesting examples of movies.

You can split the code into multiple cells.

In [ ]:

Answer: **write down the answer and explain it**

Question 5 (20How would you rate the popularity of individual actors? Describe the procedure

for evaluating and print the 10 most popular actors.

You can split the code into multiple cells.

In [ ]:

Answer: **write down the answer and explain it**

Bonus question (5

What's your favorite movie? Why?

Answer: **write down the answer and explain it**

Notes

You can use the built-in `csv` module to load data.

In [1]:
```
from csv import DictReader

reader = DictReader(open('podatki/ml-latest-small/ratings.csv', 'rt', encoding='utf-8'))
for row in reader:
    user = row["userId"]
```

```
            movie = row["movieId"]
            rating = row["rating"]
            timestamp = row["timestamp"]
```

Data in the last line of the file:

```
In [2]: user, movie, rating, timestamp
```

```
Out[2]: ('671', '6565', '3.5', '1074784724')
```

Convert the time format (*Unix time*). Code about the structure is listed in documentation of the module datetime.

```
In [3]: from datetime import datetime

        t = 1217897793 # Unix-time
        ts = datetime.fromtimestamp(t).strftime('%Y-%m-%d %H:%M')
        ts
```

```
Out[3]: '2008-08-05 02:56'
```

Search for structure in data

Data mining, assignment, `<INSERT DATE>`
`<INSERT Name and surname>`

By modeling, we try to find structure in the data. Using unsupervised modeling methods, we try to find groups of similar data or cases.

In this assignment you will use modeling of probability distributions to search for outliers and methods for finding groups of similar cases (clustering).

Data

The description of the MovieLens database remains the same as for the first assignment.

Questions

By using the principles you have learned on exercises and lectures, answer the following questions. For each question, think carefully about the best way to give, show or justify the answer. The essential part is the answers to the questions and not so much the implementation of your solution.

1. Finding outliers (50About the ratings of which movies are the users the least unified? In other words, for which films are the corresponding scores the most dispersed?

Formulate the problem as modeling the probability distribution. Think about the following questions, make the appropriate experiments and answers.

```
In [1]: # kodo lahko razdelite v več celic
```

Answer: **You can write the answer in multiple cells**

1.1. question:

What is the appropriate random variable (quantity) in the data that answers the question?

```
In [2]: # kodo lahko razdelite v več celic
```

Answer: **You can write the answer in multiple cells**

1.2. question:

Draw its distribution, for example, using a histogram.

```
In [3]: # kodo lahko razdelite v več celic
```

Answer: **You can write the answer in multiple cells**

1.3. question:

Does the distribution remind you of a known distribution? Is the distribution possibly normal or some other?

```
In [4]: # kodo lahko razdelite v več celic
```

Answer: **You can write the answer in multiple cells**

1.4. question:

Assess the parameters of this distribution by means of the procedures we have learned at the exercises. From the distributions we have learned at the exercises, choose the one that best fits the data.

```
In [5]: # kodo lahko razdelite v več celic
```

Answer: **You can write the answer in multiple cells**

1.5. question:

Print movies with the value of a random variable that falls in the top 5% of the statistically significant cases.

```
In [6]: # kodo lahko razdelite v več celic
```

Answer: **You can write the answer in multiple cells**

2. Clustering films (50

Recommendation systems often detect groups of objects (in our example films), which are of high similarity.

Find the 100 most watched movies. Are there groups among them? Use the appropriate clustering algorithm. We can watch the film as a vector where the number of components is equal to the number of users.

Vectors also contain *unknown values*. An example of the vectors for ten films is shown in the table below.

Clustering algorithms can be performed in the original space (the coordinate system films-users), or we can compare the films with the similarities we have learned on the exercises. Think about which method is more appropriate in terms of the data format.

x	Movie	u_0	u_1	u_2	$\cdots$
$\vec{x}_0$	Fight Club (1999)	?	?	?	$\cdots$
$\vec{x}_1$	Twelve Monkeys (a.k.a. 12 Monkeys) (1995)	?	?	2.5	$\cdots$
$\vec{x}_2$	Independence Day (a.k.a. ID4) (1996)	?	?	?	$\cdots$
$\vec{x}_3$	Dances with Wolves (1990)	4.0	?	?	$\cdots$
$\vec{x}_4$	Fargo (1996)	?	?	?	$\cdots$
$\vec{x}_5$	Speed (1994)	?	?	?	$\cdots$
$\vec{x}_6$	Apollo 13 (1995)	?	2.0	?	$\cdots$
$\vec{x}_7$	Seven (a.k.a. Se7en) (1995)	?	?	?	$\cdots$
$\vec{x}_8$	Sixth Sense, The (1999)	3.0	?	4.0	$\cdots$
$\vec{x}_9$	Aladdin (1992)	?	?	?	$\cdots$
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$

In doing so, answer the following questions.

2.1. question:

Justify the choice of algorithm and similarity measures.

```
In [7]: # kodo lahko razdelite v več celic
```

Answer: **You can write the answer in multiple cells**

2.2. question:

How many groups of films are among the selected? Do we know the quantitative estimates for the various grouping options?

```
In [8]: # kodo lahko razdelite v več celic
```

Answer: **You can write the answer in multiple cells**

2.3. question:

Display results using an appropriate visualization.

```
In [9]: # kodo lahko razdelite v več celic
```

Answer: **You can write the answer in multiple cells**

2.4. question:

Comment on the validity of the results obtained.

```
In [10]: # kodo lahko razdelite v več celic
```

Answer: **You can write the answer in multiple cells**

Predicting values

Data mining, assignment, `<INSERT DATE>`
`<INSERT Name and surname>`

We will get to know the practical use of simple methods of supervised modeling or forecasting. The common property of all of these methods is that with the help of random variables (attributes) they model the values of a specific variable, which we call *class* (in the context of classifying, classifying) or *response* (in the context of regression). We learnt about the basic differences between contexts in lectures and tutorials.

The practical goals we will pursue are: * modeling of individual user's responses (responses) with the help of all other users, * Comparison of supervised modeling methods.

Data

The description of the MovieLens database remains the same as for the first assignment.

Preparation of data

For the purposes of this task we will prepare the data as follows: 1. Select m movies with at least 100 views. 2. Select n users who have watched at least 100 movies. 3. Prepare matrix X in the size of $m \times n$, where the lines represent movies and columns represent users. Replace unknown values with 0.

For each of the selected n users, a regression model will be built, which aims to predict film ratings.

$y^{(0)}$

$X^{(0)}$

Film/uporabnik

u_0

u_1

u_2

...

f_1

Twelve Monkeys (a.k.a. 12 Monkeys) (1995)

0

0

2.5

...

f_2

Dances with Wolves (1990)

4

0

0

...

f_3

Apollo 13 (1995)

0

2

0

...

f_4

Sixth Sense, The (1999)

3

0

4

...

...

...

...

...

...

...

$y^{(1)}$

$X^{(1)}$

Film/uporabnik

u_1

u_0

u_2

...

f_1

Twelve Monkeys (a.k.a. 12 Monkeys) (1995)

0

0

2.5

...

f_2

Dances with Wolves (1990)

0

4

0

. . .

f_3

Apollo 13 (1995)

2

0

0

. . .

f_4

Sixth Sense, The (1999)

0

3

4

. . .

. . .

. . .

. . .

. . .

. . .

. . .

Data distribution for the model user u_0 (top matrix) and the user u_1 (bottom matrix).

Questions

1. Regression (100Set up a regression model for each user. Use one or more methods for learning regression models (linear regression, Ridge, Lasso, etc.).

For each of the n users, select the appropriate column in the data matrix. Therefore, fot the i user we have:

- Response vector $y^{(i)}$,
- Data matrix $\mathrm{X}^{(i)}$ containing all columns *except i*. For an easier understanding see the above tables. Repeat the test procedure several times (e.g., three times) with the help of the learning and test sets:
- Divide the films that the user viewed *randomly* into 75(learning set) and 25% (test set). * Learn the regression model on learning set (select the appropriate rows in X and y).
- Evaluate the model on the test set (select the appropriate rows in X and y).

Then divide the evaluation score with the number of experiments to get the final score.

Report on the performance of your model. Focus on the following questions:

- Justify an appropriate evaluation score. Does the predict the scores well?
- Rate the models for all n users with the selected evaluation score.

You can split the code needed for each answer into multiple cells.

```
In [ ]:
```

Bonus question (15Create a new user who represents your movie ratings. Rate some movies after your own taste and see how models

evaluate non-selected movies. Do you find the predictions appropriate?

You can split the code needed for each answer into multiple cells.

```
In [ ]:
```

Notes

Implementation, description, and evaluation of supervised learning methods are included in libraries `skleaarn` or`Orange`.

Predicting by matrix factorization

Data mining, assignment, `<INSERT DATE>`
`<INSERT Name and surname>`

V prejšnji domači nalogi smo uporabili metode nadzorovanega modeliranja na problemu napovedovanja ocen neocenjenih filmov. Ker smo za vsakega od m uporabnikov zgradili svoj model, dobimo m modelov, ki si med seboj ne delijo nobene informacije.

Metode matrične faktorizacije so pomemben gradnik sodobnih priporočilnih sistemov. Omogočajo nam, da vsakega uporabnika in vsak izdelek (film) modeliramo s pomočjo r regresijskih modelov, kar vodi v enoten model, ki omogoča napoved ocene za poljubno kombinacijo uporabnika in filma.

Model *matrične faktorizacije* matriko podatkov $X \in \mathbb{R}^{m \times n}$ oceni s produktov dveh matrik nižjega ranga $W \in \mathbb{R}^{m \times r}$ in $H \in \mathbb{R}^{n \times r}$, tako da

$$X = WH^T + E \tag{8.1}$$

kjer je $E \in \mathbb{R}^{m \times n}$ matrika napak oz. ostankov. Matriki modela W in H lahko poiščemo tudi, če nekatere vrednosti v X niso znane, kar velja za priporočilne sisteme. Model omogoča *napoved* vseh omenjenih neznanih vrednosti.

Vrednotenje priporočilnih sistemov se razlikuje od običajnih regresijskih modelov, saj na napovedne vrednosti gledamo kot na *seznam priporočil*, kjer nas zanima samo nekaj vrhnjih elementov tega seznama oz. ali se med njimi nahajajo relevantna priporočila.

Podatki

Opis podatkovne zbirke MovieLens 1996-2016 ostaja enak prvi nalogi.

Predpriprava podatkov

Za potrebe te naloge podatke pripravite na naslednji način:

1. Izberite n filmov, ki imajo vsaj 20 ocen.
2. Izberite m uporabnikov, ki je ocenilo vsaj 20 filmov. Upoštevajte samo filme, izbrane v prejšnjem koraku.
3. Sestavite matriko X velikosti $m \times n$ (v vsaki vrstici vsebuje vsaj 20 ocen).

Nato sestavite učno in testno množico, kot je prikazano na sliki. Za vsakega uporabnika (vrstico v X) izberite k (npr. $k = 5$) visoko ocenjenih filmov (z ocenami 5 ali 4). Učno matriko X_U sestavite tako, da izbrane filme odstranite, in jih shranite v testno matriko X_T.

Vprašanja

1. (30 NMF, predstavljenega na laboratorijskih vajah. Pri izračunu gradienta (odvoda) za vsako spremenljivko upoštevajte samo znane ocene. Na kratko opišite, kateri parametri vplivajo na učenje modela in kako? Kakšne kompromise predstavljajo?
2. (50 testno množico v skladu z opisom na Sliki~**??**a. Za vsakega uporabnika naključno odstranite $k = 5$ visoko ocenjenih filmov (z ocenami 4 ali 5). Omenjeni filmi predstavljajo *testno množco*.

   ```
   S pomočjo algoritma poiščite matriki $W$ in $H$, ki modelirata učno matriko $X_U$,
   kot je prikazano na Sliki~\ref{f:nmf-shema}b.

   Za vsakega uporabnika $i$ nato napovejte ocene za vse neocenjene filme. Vektor
   ```

```
ocen pretvorite v seznam priporočil tako, da ocene uredite po padajočem vrstnem
redu (višje napovedane ocene se nahajajo v vrhu seznama). Postopek je prikazan
na Sliki~\ref{f:nmf-shema}c.

Ocenite, ali se filmi, ki ste jih odstranili za uporabnika $i$ v povprečju
pojavljajo bližje vrhu seznama, kot bi to pričakovali po naključju. Na ta način
ugotovite, ali model smiselno priporoča filme. Opišite, kako ste izvedli
postopek vrednotenja in komentirajte rezultate.
\item (20 \%) Kako parametri modela NMF vplivajo na uspešnost napovedi? Preizkusite
    npr. nekaj različnih vrednosti za rang ($r$) matrik $W$ in $H$ in preverite,
kako različne nastavitve vplivajo na napoved.
\item (Bonus 10 \%) Ustvarite novega uporabnika, ki predstavlja vaše ocene
filmov. Ocenite nekaj filmov po lastnem okusu in ponovite analizo.
```

Komentirajte ustreznost predlogov.

Zapiski

Pri implementaciji, uporabi in opisu algoritma za reševanje matrične faktorizacije si lahko pomagate z zapiski laboratorijskih vaj, ki jih najdete na spletni učilnici.

Viri

1. Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," Computer (Long. Beach. Calif)., no. 8, pp. 30–37, 2009. [Povezava].

Implementation of a recommendation system

Data mining, assignment, `<VPIŠI DATUM ODDAJE>`
`<VPIŠI Ime in priimek>`

Develop an application in which you combine all the acquired knowledge. The application should be implemented using any technology and it should allow the new user to sign in, evaluate movies and recommend not yet rated films.

Data

The description of the MovieLens database remains the same as for the first assignment.

Questions

1. Prototype (100

Implement a prototype recommendation system. You can use any data mining method. Wherever the implementation of the functional requirements is not specified, you will take the decisions you need to implement yourself. The application should have the following functionalities:

* User interface. Plan a simple user interface of your choice. The interface can be a command line, a graphical interface or a web interface. It should be easy to start the system and be accessible for use. The interface must support the communication between the user and the recommendation system. * Registering a new user. The system should allow to add new users and input ratings through the user interface. Users can be identified, for example, with usernames, numbers, etc. The entered ratings are saved when exiting the system and are used when restarted. The user rates the films in the data with scores between 1 (insufficient) and 5 (excellent). * At the request of the user, generate a list of five most appropriate films that the user has not yet rated. The method for recommending films can be selected from a set of methods that we have learned during the course (methods of supervised modeling, matrix factorization, …) or others. In the report, describe which method you are using.

Bonus question (20Add different visualizations of the current data that can be started from the user interface to the application. This can include user clusters, trends of different genres over time, distribution (average ratings), currently the most popular movies, etc.

Describe in the report the examples of using your application - a screen image with a text or commands and results if the interface is a command line. In doing so, display at least one example for each functionality that you have implemented.

Results

Present your application briefly in a report.

The results of the task are: * a report that consists of brief instructions for using the application. Submit the `.tex` and`.pdf` file of the report, * source code of your program (.ipynb, .py, …).

Bibliography

[1] S. van der Walt, S. C. Colbert, and G. Varoquaux. The numpy array: A structure for efficient numerical computation. *Computing in Science Engineering*, 13(2):22–30, March 2011.

[2]

[3]

[4]