Lab Notes DB/IM-ML

Author: Koen B
Owner N/A
Customer: N/A

Document History

Document Location

This is a snapshot of an on-line document. Paper copies are valid only on the day they are printed. Refer to the author if you are in any doubt about the currency of this document.

Revision History

Date of this revision: **Error! Reference source not found.**	Date of next revision

Revision Number	Revision Date	Summary of Changes	Changes marked
0.01	2021-09-16	Initial version	
0.02	2021-09-21	Updated version	(N)
			(N)
			(N)
			(N)

Approvals

This document requires following approvals.

Name	Title

Distribution

This document has been distributed to

Name	Title

Contents

1. Introduction

This document provides detailed instructions for installing the required software that accompanies the summer school DB/MI-ML session, as well as the source code of the scripts and/or programs.

1.1 Structure of the document

The document comprises the following major parts

- Installation of the tools required
- An overview of the sample data
- The scripts used

2. Preparing the visualization environment

2.1 Summary

This section describes the installation of 2 visualization libraries

- PlotLy is a library for data visualization https://plotly.com/python/
- Pyvis is a library for visualizing interactive dependencies https://pyvis.readthedocs.io/en/latest/#

Both libraries have the same feature, pyvis is particularly good for displaying dependencies.

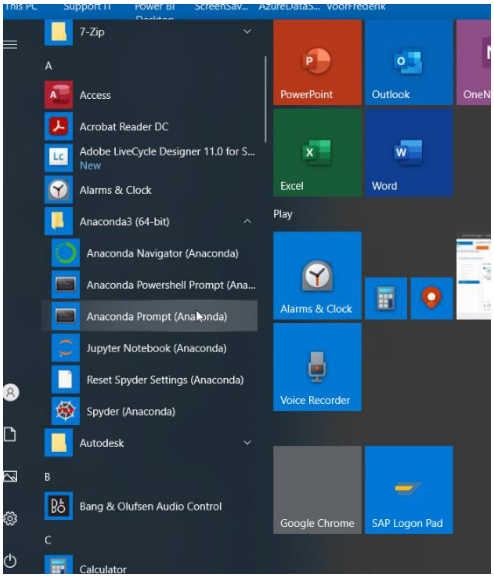
NOTE – There is also a library PlotPy, which is not the subject of this note.

2.2 Installation of anaconda

Just follow the instructions on https://docs.anaconda.com/anaconda/install/index.html

Upon successful install you should be able to launch anaconda from the Windows start bar.

This lab will primarily use the command line version of anaconda (anaconda prompt)



2.3 Anaconda tips and tricks

Have a quick look at

- "Getting started with anaconda" https://docs.anaconda.com/anaconda/user-guide/getting-started/
- The conda cheat sheet : https://docs.conda.io/projects/conda/en/latest/user-guide/tasks/manage-environments.html

Basic commands are

List the environments	conda env list
List the environments	conda info –envs
remove environment	conda remove –name <name> --all
Activate an environment	activate <name>
Deactivate an environment	deactivate <name>

NOTE – to ensure that a 64-bit environment is created unset the following environment parameter
set CONDA_FORCE_32BIT =

2.4 Installation in anaconda environment

2.4.1 Creation of a dedicated environment

Open an anaconda command prompt screen and issue the following set of commands

`conda env list`	shows the environments already present
`set CONDA_FORCE_32BIT =`	force 64 bit
`conda create -n summerschool python=3.8`	create the “summerschool” environment
`pip –version`	Just verify whether pip is available

Screenshot

```
base) c:\temp>conda create -n summerschool python=3.8
Collecting package metadata (current_repodata.json): done
Solving environment: done
==> WARNING: A newer version of conda exists. <==
  current version: 4.8.3
  latest version: 4.10.3
Please update conda by running
    $ conda update -n base -c defaults conda
## Package Plan ##
  environment location: C:\temp\devTools\Anaconda\envs\summerschool
  added / updated specs:
    - python=3.8
The following packages will be downloaded:
    package                    |            build
    ---------------------------|-----------------
    ca-certificates-2021.7.5   |       haa95532_1         113 KB
    certifi-2021.5.30          |   py38haa95532_0         140 KB
    openssl-1.1.1l             |       h2bbff1b_0         4.8 MB
    pip-21.0.1                 |   py38haa95532_0         1.8 MB
    python-3.8.11              |       h6244533_1        16.0 MB
    setuptools-52.0.0          |   py38haa95532_0         726 KB
    sqlite-3.36.0              |       h2bbff1b_0         780 KB
    vc-14.2                    |       h21ff451_1           8 KB
    vs2015_runtime-14.27.29016 |       h5e58377_2        1007 KB
```

```
    wheel-0.37.0               |     pyhd3eb1b0_1          33 KB
    ------------------------------------------------------------
                                           Total:        25.4 MB
The following NEW packages will be INSTALLED:
  ca-certificates    pkgs/main/win-64::ca-certificates-2021.7.5-haa95532_1
  certifi            pkgs/main/win-64::certifi-2021.5.30-py38haa95532_0
  openssl            pkgs/main/win-64::openssl-1.1.1l-h2bbff1b_0
  pip                pkgs/main/win-64::pip-21.0.1-py38haa95532_0
  python             pkgs/main/win-64::python-3.8.11-h6244533_1
  setuptools         pkgs/main/win-64::setuptools-52.0.0-py38haa95532_0
  sqlite             pkgs/main/win-64::sqlite-3.36.0-h2bbff1b_0
  vc                 pkgs/main/win-64::vc-14.2-h21ff451_1
  vs2015_runtime     pkgs/main/win-64::vs2015_runtime-14.27.29016-h5e58377_2
  wheel              pkgs/main/noarch::wheel-0.37.0-pyhd3eb1b0_1
  wincertstore       pkgs/main/win-64::wincertstore-0.2-py38_0
Proceed ([y]/n)? y
Downloading and Extracting Packages
certifi-2021.5.30    | 140 KB    |
################################################################################ | 100%
vs2015_runtime-14.27 | 1007 KB    |
################################################################################ | 100%
python-3.8.11        | 16.0 MB    |
################################################################################ | 100%
pip-21.0.1           | 1.8 MB     |
################################################################################ | 100%
setuptools-52.0.0    | 726 KB     |
################################################################################ | 100%
ca-certificates-2021 | 113 KB     |
################################################################################ | 100%
wheel-0.37.0         | 33 KB      |
################################################################################ | 100%
vc-14.2              | 8 KB       |
################################################################################ | 100%
openssl-1.1.1l       | 4.8 MB     |
################################################################################ | 100%
sqlite-3.36.0        | 780 KB     |
################################################################################ | 100%
Preparing transaction: done
Verifying transaction: done
Executing transaction: done
#
# To activate this environment, use
#     $ conda activate summerschool
# To deactivate an active environment, use
#     $ conda deactivate
```

2.4.2 Install PlotLy

Detailed instructions on https://plotly.com/python/

```
pip install plotly==4.14.3
```

```
(summerschool) c:\temp>pip --version
pip 21.0.1 from C:\temp\devTools\Anaconda\envs\summerschool\lib\site-packages\pip (python 3.8)

(summerschool) c:\temp>pip install plotly==4.14.3
Collecting plotly==4.14.3
  Using cached plotly-4.14.3-py2.py3-none-any.whl (13.2 MB)
Collecting six
  Downloading six-1.16.0-py2.py3-none-any.whl (11 kB)
Collecting retrying>=1.3.3
  Using cached retrying-1.3.3-py3-none-any.whl
Installing collected packages: six, retrying, plotly
Successfully installed plotly-4.14.3 retrying-1.3.3 six-1.16.0

(summerschool) c:\temp>
```

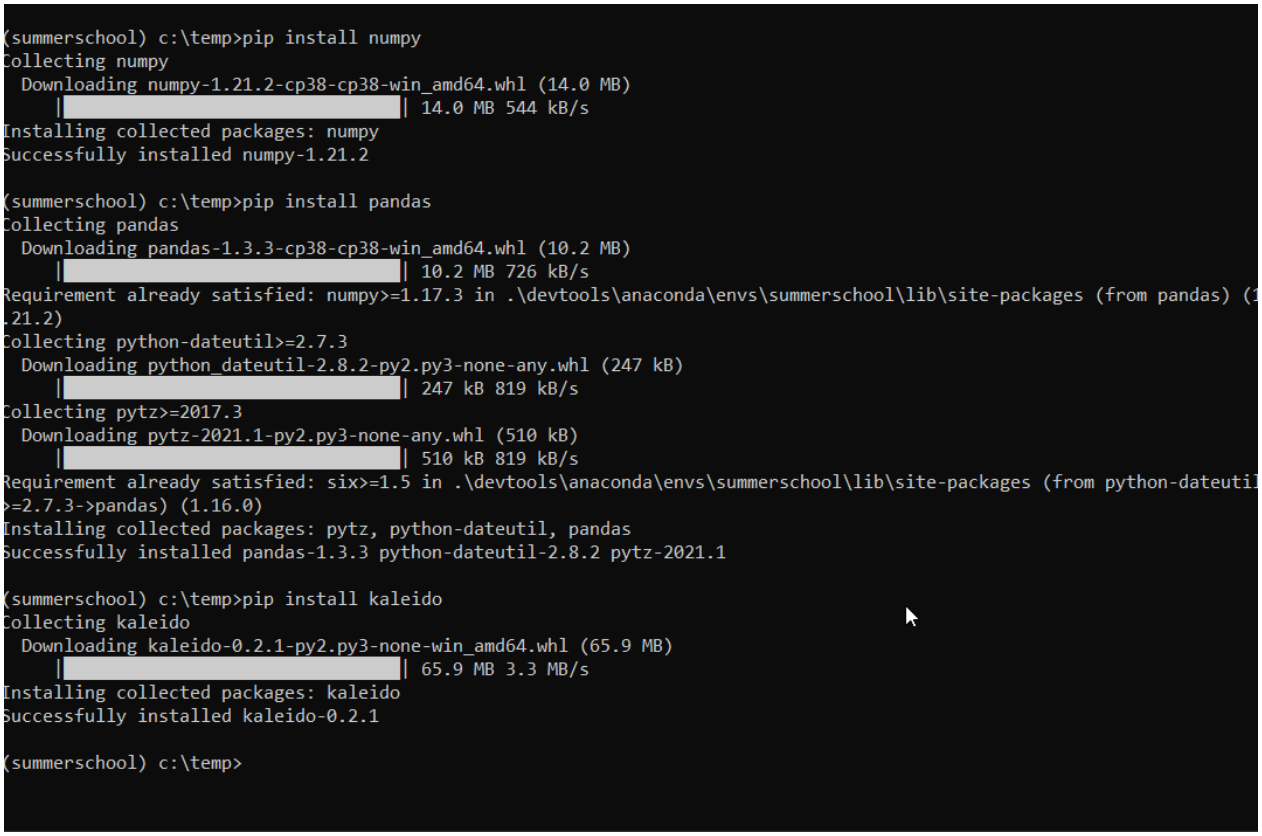
2.4.3 Install additional required packages

```
pip install numpy
pip install pandas
pip install kaleido
pip install statsmodels (only if you want to draw regressionline)
```

Kaleido is required for “static image export”, i.e. to export PlotPy diagrams to PNG of JPG format.

You might see a firewall message kaleido is asking to access internet, this is ok. Kaleido is used for rendering images in a browser? See following article

https://medium.com/plotly/introducing-kaleido-b03c4b7b1d81



2.4.4 Support for reading Excel files (openpyxl)

This library is required is you plan to read Excel files via Pandas + you need at least 2.5.7, so just pick a recent version.

```
pip install openpyxl==3.0.0
```

```
(summerschool) c:\temp>pip install openpyxl==3.0.0
Collecting openpyxl==3.0.0
  Downloading openpyxl-3.0.0.tar.gz (172 kB)
     |████████████████████████████████| 172 kB 409 kB/s
Collecting jdcal
  Using cached jdcal-1.4.1-py2.py3-none-any.whl (9.5 kB)
Collecting et_xmlfile
  Downloading et_xmlfile-1.1.0-py3-none-any.whl (4.7 kB)
Building wheels for collected packages: openpyxl
  Building wheel for openpyxl (setup.py) ... done
  Created wheel for openpyxl: filename=openpyxl-3.0.0-py2.py3-none-any.whl size=241188 sha256=8a43926fb0414ec8aa30a6f03c
e68a132474c7e62e787f0d5065aa68a3d9688e
  Stored in directory: c:\users\koen.berton\appdata\local\pip\cache\wheels\3a\fa\7c\72b59cf291697c7161eab1edadcb6bfcd1cc
b0a3b5b98a50e1
Successfully built openpyxl
Installing collected packages: jdcal, et-xmlfile, openpyxl
Successfully installed et-xmlfile-1.1.0 jdcal-1.4.1 openpyxl-3.0.0

(summerschool) c:\temp>
```

NOTE - You might also want to install xlrd if you plan to read XLS files (and not the XML based versions, e.g. XLSX)

2.4.5 Install PYVIS

This is a straightforward pip install (https://pyvis.readthedocs.io/en/latest/install.html

```
pip install pyvis
```

```
summerschool) c:\temp\SummerSchool\Scripts>pip install pyvis
Collecting pyvis
  Downloading pyvis-0.1.9-py3-none-any.whl (23 kB)
Collecting jinja2>=2.9.6
  Downloading Jinja2-3.0.1-py3-none-any.whl (133 kB)
     |████████████████████████████████| 133 kB 819 kB/s
Collecting ipython>=5.3.0
  Downloading ipython-7.27.0-py3-none-any.whl (787 kB)
     |████████████████████████████████| 787 kB 1.3 MB/s
Collecting jsonpickle>=1.4.1
  Downloading jsonpickle-2.0.0-py2.py3-none-any.whl (37 kB)
Collecting networkx>=1.11
  Downloading networkx-2.6.3-py3-none-any.whl (1.9 MB)
     |████████████████████████████████| 1.9 MB 3.3 MB/s
Collecting decorator
  Downloading decorator-5.1.0-py3-none-any.whl (9.1 kB)
Requirement already satisfied: setuptools>=18.5 in
c:\temp\devtools\anaconda\envs\summerschool\lib\site-packages (from ipython>=5.3.0-
>pyvis) (52.0.0.post20210125)
Collecting colorama
  Using cached colorama-0.4.4-py2.py3-none-any.whl (16 kB)
Collecting prompt-toolkit!=3.0.0,!=3.0.1,<3.1.0,>=2.0.0
  Downloading prompt_toolkit-3.0.20-py3-none-any.whl (370 kB)
     |████████████████████████████████| 370 kB 1.7 MB/s
Collecting jedi>=0.16
  Downloading jedi-0.18.0-py2.py3-none-any.whl (1.4 MB)
     |████████████████████████████████| 1.4 MB 819 kB/s
Collecting pickleshare
  Using cached pickleshare-0.7.5-py2.py3-none-any.whl (6.9 kB)
Collecting pygments
  Downloading Pygments-2.10.0-py3-none-any.whl (1.0 MB)
     |████████████████████████████████| 1.0 MB 1.3 MB/s
Collecting backcall
```

```
  Using cached backcall-0.2.0-py2.py3-none-any.whl (11 kB)
Collecting traitlets>=4.2
  Downloading traitlets-5.1.0-py3-none-any.whl (101 kB)
     |████████████████████████████████| 101 kB 3.3 MB/s
Collecting matplotlib-inline
  Downloading matplotlib_inline-0.1.3-py3-none-any.whl (8.2 kB)
Collecting parso<0.9.0,>=0.8.0
  Downloading parso-0.8.2-py2.py3-none-any.whl (94 kB)
     |████████████████████████████████| 94 kB 1.0 MB/s
Collecting MarkupSafe>=2.0
  Downloading MarkupSafe-2.0.1-cp38-cp38-win_amd64.whl (14 kB)
Collecting wcwidth
  Using cached wcwidth-0.2.5-py2.py3-none-any.whl (30 kB)
Installing collected packages: wcwidth, traitlets, parso, pygments, prompt-toolkit,
pickleshare, matplotlib-inline, MarkupSafe, jedi, decorator, colorama, backcall,
networkx, jsonpickle, jinja2, ipython, pyvis
Successfully installed MarkupSafe-2.0.1 backcall-0.2.0 colorama-0.4.4 decorator-5.1.0
ipython-7.27.0 jedi-0.18.0 jinja2-3.0.1 jsonpickle-2.0.0 matplotlib-inline-0.1.3
networkx-2.6.3 parso-0.8.2 pickleshare-0.7.5 prompt-toolkit-3.0.20 pygments-2.10.0
pyvis-0.1.9 traitlets-5.1.0 wcwidth-0.2.5
```

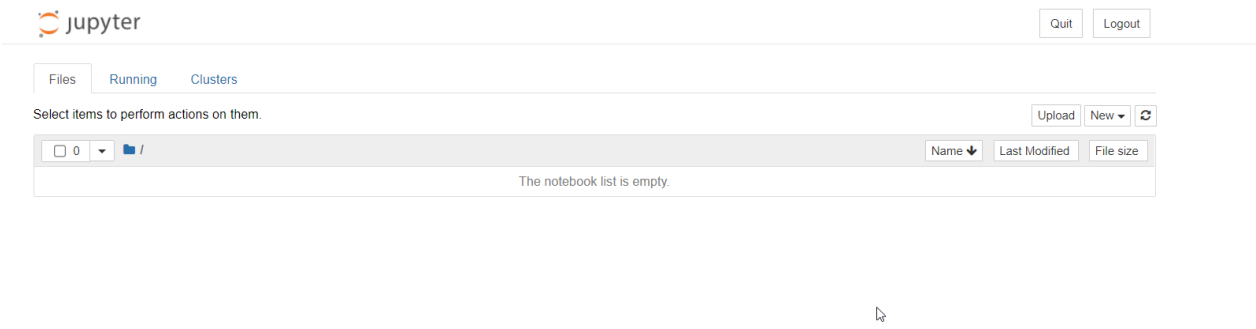
2.4.6 Install Jupyter notebooks

```
pip install jupyter
```

NOTE - "pip install jupyterlab" and "pip install notebook" can alternatively be used. The subtle differences between these options are yet unknown to me.

Start jupyter as follows (your default browser will open and display this page)

```
Jupyter notebook
```



2.4.7 Smoke tests

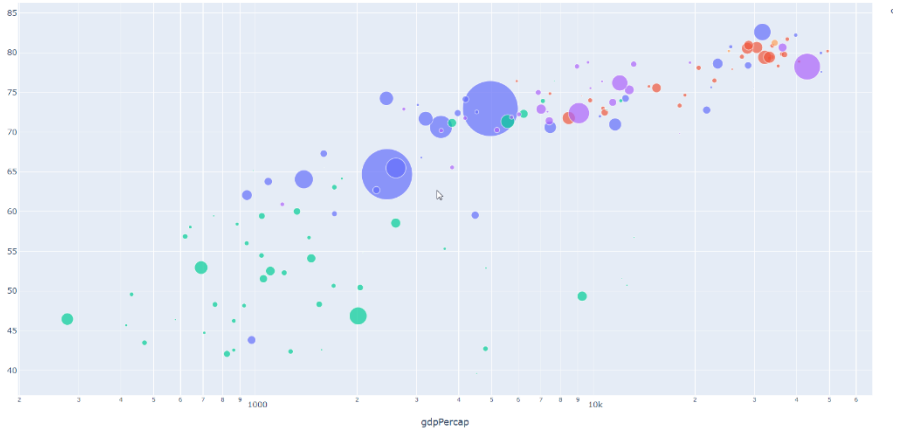
Smoke tests are very simple tests that merely assess the correct functioning of the infrastructure and/or environment.

2.4.8 Plotly smoke test

Create the following python scrip (test.txt) and run it (python test.txt)

```
import plotly.express as px
df = px.data.gapminder()

fig = px.scatter(df.query("year==2007"), x="gdpPercap", y="lifeExp",
                 size="pop", color="continent",
hover_name="country", log_x=True, size_max=60)
fig.show()
```

This will open your web browser and display this picture (hopefully)



Change the script to redirect its output to an image file: fig.write_image, e.g. "diagram.png".

```
import plotly.express as px
df = px.data.gapminder()

fig = px.scatter(df.query("year==2007"), x="gdpPercap", y="lifeExp",
                 size="pop", color="continent",
                   hover_name="country", log_x=True, size_max=60)
#fig.show()
fig.write_image("diagram.png")
quit()
```

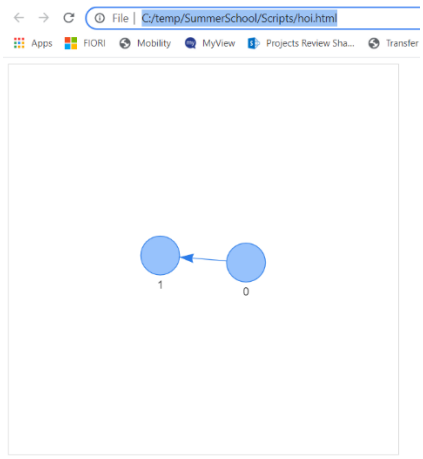
2.4.9 pyvis smoke test

Create the following script (testpyvis.txt)

```
from pyvis.network import Network
net = Network(directed=True)
net.add_node(0)
net.add_node(1)
net.add_edge(0,1)
net.show('hoi.html')
quit()
```

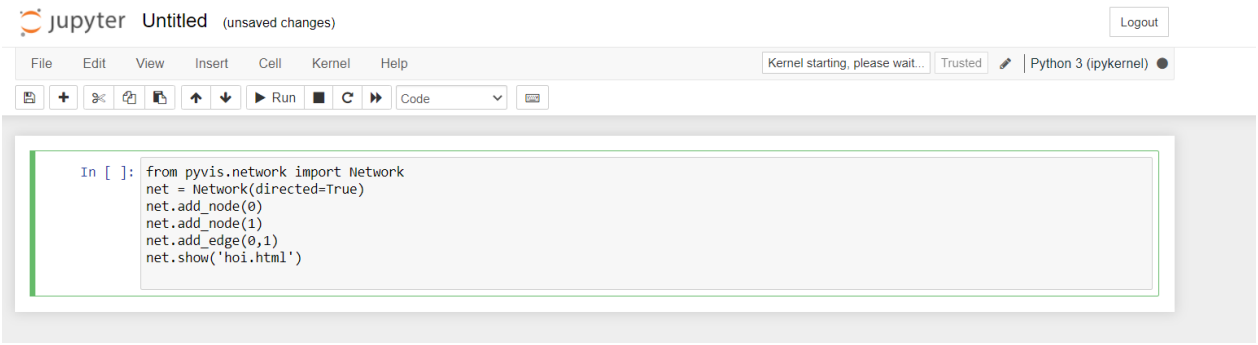
run the script as follows :python testpyvis.txt

Your default browser will open, and an interactive diagram will be present.



The script can (obviously) also be run via Jupyter notebook.

from within the jupyter landing page select new > python and the following screen will open, then copy the above script in the first line, put the cursor right before the first instruction, press run and be very patient.



2.5 Installation standalone visualization test environment

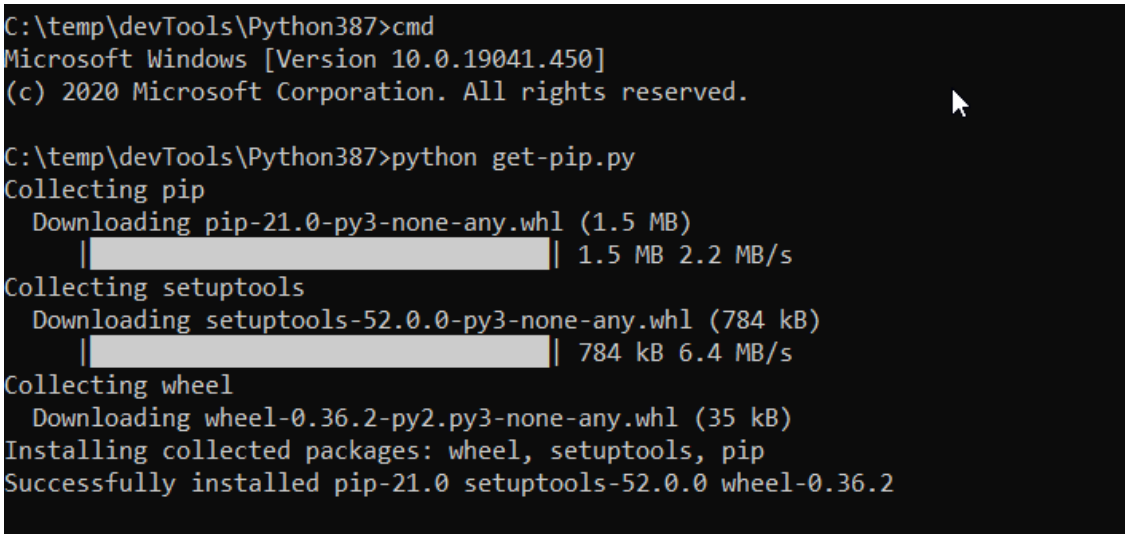
2.5.1 Download Python

- Download an embeddable version of Python for Windows, e.g. version 3.8.7, from https://www.python.org/downloads/release/python-387/
- Unzip in c:\temp\devtoool\Python387
- Create this bat file to start python (make sure to include the Scripts folder on the path

```
SET JAVA_HOME=C:\temp\devTools\jdk-9.0.1
SET PYTHON3_HOME=C:\temp\devTools\Python387
SET PYTHON3_SCRIPTS=%PYTHON3_HOME%\Scripts
SET PATH=%PYTHON3_HOME%;%PYTHON3_SCRIPTS%;%PATH%;%JAVA_HOME%\bin
CD %PYTHON3_HOME%
cmd
```

2.5.2 Install PIP

- Create a folder %PYTHON%Scripts
- Create a script get-pip.py in the %PYTHON% folder, this is a bootstrap script for installing PIP. You need to download or copy the contents of this webpage https://bootstrap.pypa.io/get-pip.py in the script.
- Run the script: python get-pip.py



Try to run pip (pip –version). Pip should not work at this stage (ModuleNotFoundError)

2.5.3 Configure PYTHONPATH

Python requires an environment variable PYTHONPATH

HOWEVER, this environment variable is overruled by the settings in the python38. __pth file (which is in the %PYTHON% folder and might be named differently depending of python version you installed e.g. python36.__pth)

There are 2 options

Option 1

Modify the content of the python38.__path file as follows

```
C:\temp\devtools\python387
C:\temp\devtools\python387\DLLs
C:\temp\devtools\python387\lib
C:\temp\devtools\python387\lib\plat-win
C:\temp\devtools\python387\lib\site-packages
```

DO NOT USE THIS OPTION – I experienced some uses when installing openpylxl, ie. missing of. directory. 2nd option is better.

Option 2

Rename python38.__pth to python38._pth_OLD

And set the environment variable

Set PYTHONPATH=%PYTHON_HOME%;%PYTHON_HOME%\DLLS;%PYTHON_HOME%\lib;%PYTHON_HOME%\lib\plat-win;%PYTHON_HOME%\lib\site-packages

I prefer the above and create a batch file and clearly mention to rename the _PTH file

```
REM DO NOT FORGET to rename pythonNN._pth to pythonNN.__pth.OLD to
activate
SET JAVA_HOME=C:\temp\devTools\jdk-9.0.1
SET PYTHON3_HOME=C:\temp\devTools\Python387
SET PYTHON3_SCRIPTS=%PYTHON3_HOME%\Scripts
SET
PYTHONPATH=%PYTHON3_HOME%;%PYTHON3_HOME%\DLLs;%PYTHON3_HOME%\lib;%PYTHO
N3_HOME%\lib\plat-win;%PYTHON3_HOME%\lib\site-packages
SET PATH=%PYTHON3_HOME%;%PYTHON3_SCRIPTS%;%PATH%;%JAVA_HOME%\bin
CD %PYTHON3_HOME%
cmd
```

Test pip

```
pip –version
```

2.5.4 Install PlotLy

Detailed instructions on https://plotly.com/python/

```
pip install plotly==4.14.3
```

```
C:\temp\devTools\Python387>pip install plotly==4.14.3
Collecting plotly==4.14.3
  Downloading plotly-4.14.3-py2.py3-none-any.whl (13.2 MB)
     |████████████████████████████████| 13.2 MB 6.4 MB/s
Collecting retrying>=1.3.3
  Downloading retrying-1.3.3.tar.gz (10 kB)
Collecting six
  Using cached six-1.15.0-py2.py3-none-any.whl (10 kB)
Building wheels for collected packages: retrying
  Building wheel for retrying (setup.py) ... done
  Created wheel for retrying: filename=retrying-1.3.3-py3-none-any.whl size=11429 sha256
2a6d4041ecf6faecb9f97e925b9987d9f
  Stored in directory: c:\users\koen.berton\appdata\local\pip\cache\wheels\c4\a7\48\0a43
0792f67b476e56
Successfully built retrying
Installing collected packages: six, retrying, plotly
Successfully installed plotly-4.14.3 retrying-1.3.3 six-1.15.0

C:\temp\devTools\Python387>_
```

NOTE – This might take a while (2 minutes)

Install additional required packages

```
pip install numpy
pip install pandas
pip install kaleido
```

Kaleido is required for “static image export”, i.e. to export PlotPy diagrams to PNG of JPG format.

You might see a firewall message kaleido is asking to access internet, this is ok. Kaleido is used for rendering images in a browser? See following article

https://medium.com/plotly/introducing-kaleido-b03c4b7b1d81

2.5.5 Install openpyxl

This library is required is you plan to read Excel files via Pandas + you need at least 2.5.7, so just pick a recent version.

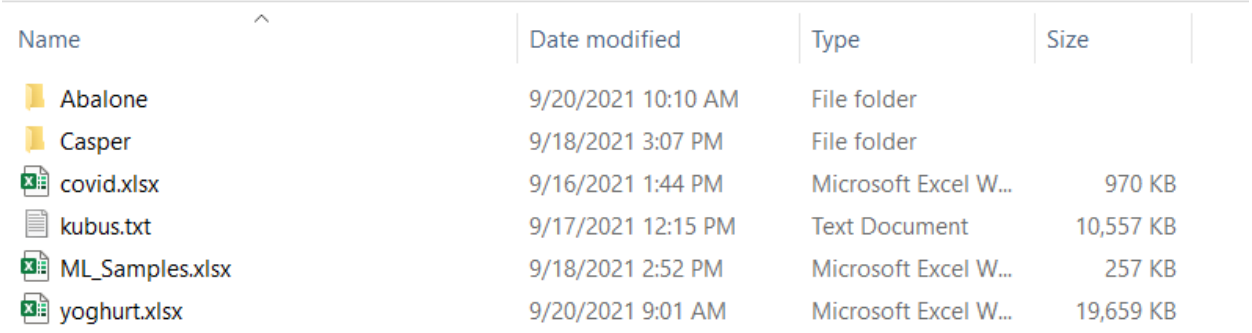
```
pip install openpyxl==3.0.0
```

NOTE - You might also want to install xlrd if you plan to read XLS files (and not the XML based versions, e.g. XLSX)

3. Sample data

3.1 Introduction

We will use the following sample data during this education session.



Name	Date modified	Type	Size
Abalone	9/20/2021 10:10 AM	File folder	
Casper	9/18/2021 3:07 PM	File folder	
covid.xlsx	9/16/2021 1:44 PM	Microsoft Excel W...	970 KB
kubus.txt	9/17/2021 12:15 PM	Text Document	10,557 KB
ML_Samples.xlsx	9/18/2021 2:52 PM	Microsoft Excel W...	257 KB
yoghurt.xlsx	9/20/2021 9:01 AM	Microsoft Excel W...	19,659 KB

- Covid.xlsx comprises information on the admissions and releases for Belgian hospitals during the first months of the COVID-19 pandemic.
- Kubus.txt can be ignored. This is denormalized data from the yoghurt sample data. It is created by the Python script dairy04.txt.
- ML_Samples.xlsx comprises the abalone and Casper sample data in Excel format
- The Abalone folder has the Abalone.ARFF training set enabling to predict the gender of the abalone seashell in Attribute Relation File Format, as well as a copy of all scatterplots and histograms.
- The Casper folder contains the pictures and data for the Casper clustering exercise.
- Yohurt.xlsx is sample sales, customer and product information. Courtesy of Jan V/Jonas M.

4. PlotLy fundamentals

4.1 Usage

- Create a folder c:\temp\summerschool
- And the following subfolders
- C:\temp\summerschool\diagrams
- C:\temp\summeschool\scripts
- C:\temp\summerschool\sampledata

Get the sampledata and scritps from github and put those in the corresponding subfolder

Open an anaconda command window and go to the folder c:\temp\summerschool\scripts yo can run any scrip by issuing the command python <scriptname>. In mostg cases this will create a diagram in the ..\diagram subfolder.

4.2 Script overview

See the slide deck for screenshots

ScriptName	Comment
Abalone01	Reads Excel and crated scatterplot (shell weight / length)
Abalone02	Adds regression lines to the above diagram
Abalone03	Creates a histogram instead of scatterplot
Abalone04	Creates a covariance diagram for the abalone sample data
Casper01	K-means example
Casper03	Pyvis example
Covid01	Reads excel and creates boxplot (hospital admissions and releases per province), shows all data points (outliers), also demonstrates how to perform aggregation (group by) in Pandas.
Covid02	Scatterplot (not very useful)
Covid03	Idem (also not very inspiring)
Covid04	Stacked bar plot of the hospital admission data (also demonstrated filtering)
Covid05	Bar diagram
Covid06	Cluster diagram (not very useful)
Dairy01	Sunburst diagram regions and subregions of the yogurt test data
Dairy02	Sunburst diagram products and product category of the yogurt test data
Dairy03	Sunburst yogurt sales, demonstrates joining of Pandas dataframes and export to CSV file format.

Dairy04	Scatterplot yogurt sales projected the region X and Y coordinates
Dairy05	Idem
Kmeans	K-means clustering of the customers on geo-location in the yogurt test data
Pyvis02	Pyvis example (Game of Thrones characters)
Testpyvis	Pvysi smoke test (2 nodes and 1 edge)

5. Clustering (K-Means) example

5.1 Introduction

The Python library scikit-learn comprises an implementation of the K-Means algoritm.

Use pip to Install

- scikit-learn, and
- matplotlib

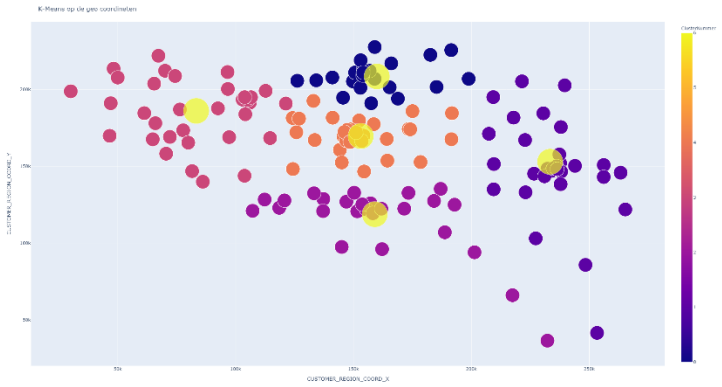
```
C:\temp\SummerSchool\Scripts>pip install scikit-learn
Collecting scikit-learn
  Downloading scikit_learn-0.24.2-cp38-cp38-win_amd64.whl (6.9 MB)
     |████████████████████████████████| 6.9 MB 6.4 MB/s
Collecting joblib>=0.11
  Downloading joblib-1.0.1-py3-none-any.whl (303 kB)
     |████████████████████████████████| 303 kB 1.3 MB/s
Collecting threadpoolctl>=2.0.0
  Downloading threadpoolctl-2.2.0-py3-none-any.whl (12 kB)
Requirement already satisfied: scipy>=0.19.1 in
c:\temp\devtools\anaconda\envs\summerschool\lib\site-packages (from scikit-
learn) (1.7.1)
Requirement already satisfied: numpy>=1.13.3 in
c:\temp\devtools\anaconda\envs\summerschool\lib\site-packages (from scikit-
learn) (1.21.2)
Installing collected packages: threadpoolctl, joblib, scikit-learn
Successfully installed joblib-1.0.1 scikit-learn-0.24.2 threadpoolctl-2.2.0
```

5.2 Scripts

5.2.1 Clustering on geo-coordinates of yogurt customers

This script clusters the customers from the Yogurt sample data based on their geographical X and Y coordinates.

You should be able to detect the shape of Belgium, 5 clusters and their centroids.



The core of the script is the "*kmeans = KMeans (n_clusters=n, random_state=0).fit(Pandas dataframe)*" instruction. The remainder of the script just reads the data, transforming arrays intgo Pandas dataframes and readying the date for displaying in a scatterplot.

```
import pandas as pd
import numpy as np
import plotly.express as px
import os
from sklearn.cluster import KMeans


AantalKlusters=5

#
WorkingDirCDrive='C:/temp/summerschool'
if os.path.exists( WorkingDirCDrive ):
  WorkingDir = WorkingDirCDrive
#
DiagramFileName= WorkingDir + '/diagrams/dairy-kmeans.png'
ExcelFileName=WorkingDir + '/sampledata/yoghurt.xlsx'
print ( DiagramFileName )

#
df0 = pd.read_excel( ExcelFileName  , sheet_name='Customer' , header=0 ,
engine='openpyxl'  )
print( df0.head() )

# alles
df1 = df0[ df0["CUSTOMER_POSTAL_CODE"] != "0000" ]

# sunburst does not support nodes which are NULL or NaN
df2 = df1.replace( np.nan, '', regex=True)
#

#
df3 =
df2[['CUSTOMER_ID','CUSTOMER_REGION_COORD_X','CUSTOMER_REGION_COORD_Y']]
print ( df3.head )


#
```

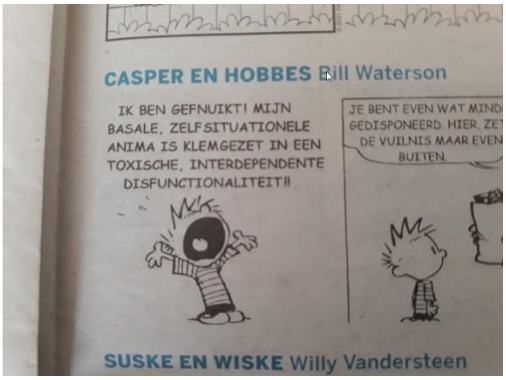
```
kmeans = KMeans (n_clusters=AantalKlusters, random_state=0).fit( df3 )
print( kmeans )

# Get the cluster labels
print("dit zijn de labels")
print(kmeans.labels_)

# we gaan de input data nu proberen te mergen met de labels
# kmeans result is een array, dus omzetten naar pandas dataframe
dflabels = pd.DataFrame( kmeans.labels_ , columns = ['ClusterNummer'])
print( dflabels.head() )

# plak het rijnummer aan iedere rij
df3['RijNummer'] = df3.reset_index().index
print( df3.head() )
dflabels['RijNummer'] = dflabels.reset_index().index
print( dflabels.head() )

# joinen
df5 = pd.merge( df3 , dflabels , on="RijNummer" )
print( df5.head() )

# Voeg een kolom toe die de grootte van de dot definieert
df5['Grootte'] = 15

# CUSTOMER_ID  CUSTOMER_REGION_COORD_X  CUSTOMER_REGION_COORD_Y  RijNummer
ClusterNummer Grootte

#
print("Dit zijn de centroids")
print(kmeans.cluster_centers_)
dfcentroids = pd.DataFrame( kmeans.cluster_centers_ , columns =
['SILLY','CUSTOMER_REGION_COORD_X','CUSTOMER_REGION_COORD_Y'])
print( dfcentroids.head() )

# herwerk de centroids datafram zodat die past op de df5 data frame - zet de
grootte
df10 = dfcentroids[['CUSTOMER_REGION_COORD_X','CUSTOMER_REGION_COORD_Y']]
df10['ClusterNummer'] = AantalKlusters + 1  # zet hoog
df10['Grootte'] = 50
print( df10.head() )

# achteraan toevoegen -
df100 = pd.concat([df5, df10], ignore_index=True, sort=False)
print( df100 )


#
fig = px.scatter( df100 , x="CUSTOMER_REGION_COORD_X",
y="CUSTOMER_REGION_COORD_Y" , color="ClusterNummer", size_max=50 ,
size="Grootte" )
fig.update_layout( title='K-Means op de geo coordinaten')
#fig.update_traces(marker={'size': 15})

fig.write_image( DiagramFileName , width=1980, height=1080 )


quit()
```

5.2.2 Clustering of connected components

5.2.2.1 CASPER01.txt

This testcase also shows how to apply the elbow optimization method for finding the optimal number of clusters.

A phot was taken from the following Casper cartoon and cropped to only contain the first frame.

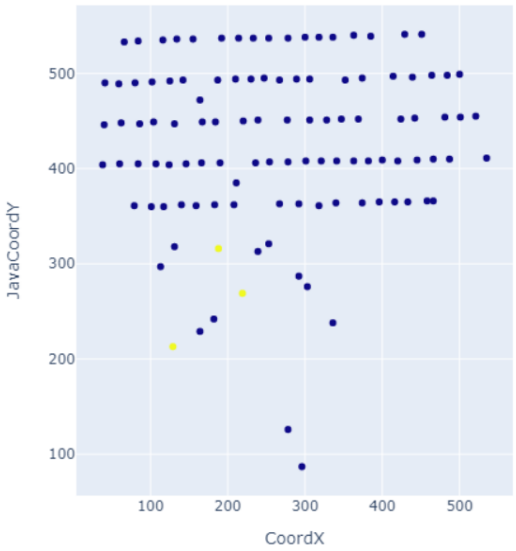


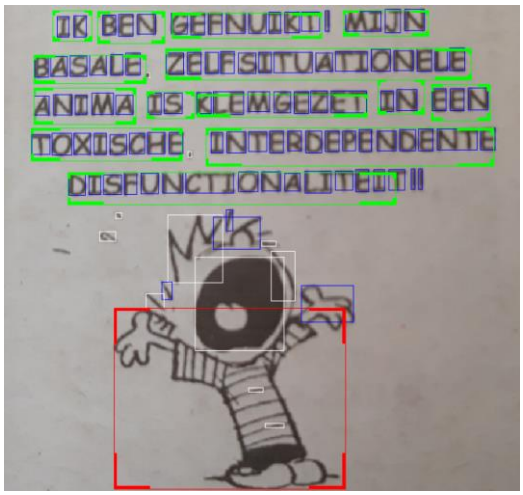
The connected components on the photo were extracted and stored in Excel. A connected component is a collection of pixels (black dots) adjacent to one another. There are approx. 120 connected components in this picture.

UID	ParentUID	CoordX	CoordY	Width	Height	Content	Tipe
5277563950925	-1	429	13	17	25	?	ConCom
5277563950926	-1	451	13	21	23	?	ConCom
5277563950927	-1	363	14	4	23	?	ConCom
5277563950928	111	385	15	39	22	mijn	ConCom
5277563950929	-1	300	16	13	23	?	ConCom
5277563950930	-1	318	16	16	22	?	ConCom
5277563950931	-1	336	16	20	21	?	ConCom
5277563950932	-1	192	17	18	23	?	ConCom
5277563950933	110	214	17	15	24	g	ConCom
5277563950934	110	233	17	15	23	e	ConCom
5277563950935	110	253	17	21	22	f	ConCom
5277563950936	110	278	17	19	22	nuikt	ConCom
5277563950937	109	134	18	16	24	b	ConCom
5277563950938	109	155	18	20	22	e	ConCom
5277563950939	109	116	19	15	22	n	ConCom
5277563950940	108	84	20	14	23	i	ConCom
5277563950941	108	66	21	13	22	k	ConCom
5277563950942	-1	500	55	17	25	?	ConCom
5277563950943	113	464	56	16	25	z	ConCom

The connected components look as follows (blue boxes identify the connected components which are predicted to be characters)

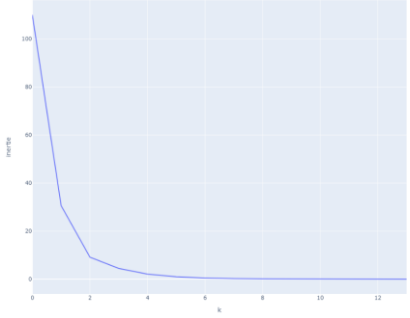
By applying a K-Means algorithm for K = 3 on the height of the connected components., the following clusters are found. By plotting the x/y coordinates in a scatterplot in plotpy you get this picture.





Observe that the characters on the lower part of the picture are not clustered correctly (left picture, there are components, such as Casper's spiky hair) which are also part of the character cluster. Additional processing is needed to correctly identify characters.

The script will also create a diagram with the result of the Elbow optimization. Optimal number of clusters is 2.



6. Pyvis example

6.1.1.1 CASPER03.txt

There is also a pyvis example script, which shows the dependencies of the various connected components, the words, paragraphs and entire file.

The script crate the casper03.html file. Just click on the HTML to experience the interaction.