SOPRONI EGYETEM
SIMONYI KÁROLY MŰSZAKI, FAANYAGTUDOMÁNYI ÉS MŰVÉSZETI KAR
INFORMATIKAI ÉS GAZDASÁGI INTÉZET

ÚJ NEMZETI KIVÁLÓSÁG PROGRAM
ÚNKP-18-1-I-SOE-27

DOKUMENTÁCIÓ

CSANAKI RICHÁRD
DR. PÖDÖR ZOLTÁN, DR. JEREB LÁSZLÓ
KZEADR

SOPRON,
2019. ÁPRILIS 10.

Ösztöndíjas időszak
2018. november 1. – 2019. március 31.
Vállalások

1. Kutatási tervben megfogalmazott célok teljesítése
2. Havonta legalább egy magyar vagy idegen nyelvű szakmai anyag feldolgozása
3. Egyetemi ÚNKP rendezvényen való részvétel

Feladat megfogalmazás
A Soproni Egyetem SKK Informatikai és Gazdasági Intézménye által működtetett SensorHUB [1] infrastruktúrában tárolt szenzoradatok automatizált elemzése adatbányászati, Big Data megoldásokkal [2]. Valamint informatív adatvizualizáció az adathalmazokból kinyert ténylegesen új információk szakmai szempontú megjelenítésére [3]. Ezeken az információkon alapuló döntéshozatal támogatása, majd a döntéshozatali ciklusból a humánfaktor részvételének minimalizálása, gépi tanulás (machine learning), öntanuló algoritmusok és az emberi agy (intelligencia forrása, neuronhálózatok rendszere) modellezési lehetőségeinek vizsgálata, a lehetséges megoldások illesztése az adott feladathoz.

Feldolgozott irodalom, anyagok, kurzusok

1. Life 3.0 – Max Tegmark, 2017, ISBN: 978-0-141-98179-6
2. Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython – Wes McKinney, 2017, ISBN: 978-1-491-95766-0
3. Python in 24 Hours – Katie Cunningham, 2014, ISBN: 978-0-672-33687-4
4. Tidy Data – Hadley Wickham, < https://vita.had.co.nz/papers/tidy-data.pdf >, utolsó megtekintés: 2019. április 9.
5. Hands-On Machine Learning with Scikit-Learn & TensorFlow – Aurélien Géron, 2017, ISBN: 978-1-491-96229-9
6. Think Stats (Exploratory Data Analysis in Python) – Allen B. Downey, 2014
7. Pandas for Data Analysis, SciPy 2017 Tutorial – Daniel Chen, utolsó megtekintés: 2019. április 9.
8. Introduction to Probability (Harvard Statistics 110) – Joseph K. Blitzstein, utolsó megtekintés: 2019. április 9.

Felhasznált technológiák

- Python
    - Interpretált programozási nyelv, Big Data és ML területen standard
- Anaconda
    - Könyvtár és virtuális környezet csomag Python nyelven való fejlesztéshez
- CDS API
    - Copernicus adatbázisok lekérdező felülete
- SensorHUB
    - Intézeti adattároló környezet
- ecCodes
    - Terminál eszköztár GRIB és NetCDF formátumú fájlok kezelésére
- CMake
    - C alapú build eszköztár
- QGIS
    - GRIB, netCDF és további formátumok megjelenítésére képes open source alkalmazás
- Pandas
    - Tabuláris adatelemző könyvtár
- Matplotlib
    - Tudományos adatvizualizációs könyvtár
- NumPy
    - Nagy teljesítményű mátrix könyvtár
- Scikit-Learn
    - ML (Machine Learning) eszköztár
- Jupyter Notebook
    - Interaktív Python interpreter (IPython Shell)
- XArray
    - N-D dimenziójú tabuláris adatszerkezet
- CDO
    - Climate Data Operator – Parancssoros eszköztár netCDF fájlok kezelésére
    - Ugyanilyen névvel Python könyvtár hasonló funkciókkal

- GitHub és GitHub Desktop
    - Felhőalapú verziókövető szolgáltatás

Felhasznált technológiák – bővebben(*)

Python

A Python egy általános célú, high-level interpretált programozási nyelv. Főbb ismérvei a könnyű olvashatóság és az élénk fejlesztői közösség. Webfejlesztés, adatelemzés és gépi tanulás területeken standardnak számít.

Anaconda

Data Science területen standard Python disztribúció, könyvtár gyűjtemény. Több, mint 1000 open source csomagot tartalmaz, többek között a Pandas, NumPy, Matplotlib, Scikit-Learn és TensorFlow könyvtárakat.

Pandas

Python könyvtár könnyen használható, nagy teljesítményű adatstruktúrákkal, adatelemző eszközökkel.

NumPy

Python könyvtár N-dimenziós tömbökkel és ezek transzformációs eszközeivel. A hagyományos adatszerkezeteknél könnyebb és gyorsabb feldolgozhatóságot biztosít tetszőleg adattípusokkal.

Scikit-Learn

Klasszikus machine learning Python könyvtár, regressziós, klaszterezős és klasszifikációs modulokkal. Támogatja az ún. Supervised és Unsupervised gépi tanulás módszereket is.

Jupyter Notebook

IPython Shell alapú böngészős Python fejlesztői környezet, mely több integrált könyvtár segítségével és megoldással könnyíti meg a projektek dokumentálását: Markdown markup nyelv támogatása, Matplotlib grafikonok kirajzolása, adatszerkezetek automatikus megjelenítése.

SensorHUB [1]

Az Informatikai és Gazdasági Intézet által működtetett adattároló technológia, az országszerte kihelyezett 6 mérőállomás (Püspökladány, Sárvár, Kecel, Napkor, Karád, Pilisszentlélek) által küldött adatok tárolási helye, az elkészült általános Gateway alkalmazás átmeneti futtató környezete. Az általános Gateway alkalmazás gondoskodik a SensorHUB rendszerbe érkező adatok validációjával, illetve felhasználói igények kielégítésével. Új fejlesztés.

CDS API

Copernicus Climate Data Store lekérdező felülete, Python környezetből elérhető, az egész Földre kiterjedő meteorológiai adatokat tárolja, könnyen hozzáférhető programozói interfészt biztosítva.

ecCodes
Az ECMWF által fejlesztett parancssori eszköztár, melynek segítségével a GRIB és BUFR kiterjesztésű fájlok feldolgozhatóak, átalakíthatóak más programok által is feldolgozható formátumba. A klasszikus GRIB-API bővített verziója.

CMake
C alapú build alkalmazás, hordozható tudományos alkalmazások telepíthetőek vele.

QGIS
Grafikus felhasználói felülettel rendelkező alkalmazás mely automatikusan képes GRIB és netCDF kiterjesztésű fájlokat megjeleníteni, elemezni.

GitHub és GitHub Desktop
Web alapú, a git verziókövető rendszert használó szolgáltatás, melynek nem csak parancsoros eszközei vannak, hanem grafikus felhasználói felületet támogató alkalmazásai is. Ilyen alkalmazás a GitHub Desktop is, melynek segítségével a forráskódban és a dokumentációban eszközölt változások követhetők figyelemmel.

A projekt forráskódját és dokumentációját tartalmazó GitHub repository itt található meg:
< https://github.com/csana23/UNKP >

(*)
A felsorolt alkalmazások, programok között nem szerepel a Microsoft által fejlesztett Power BI, attól függetlenül, hogy a kutatási tervben ezt az alkalmazást jelöltem meg az elsőszámú adatelemző technológiaként. A tervhez képest azért nem ezt használtam, hiszen nagyvállalati környezetben, folyamatos és standardizált dataflow-k elemzésére kiváló eszköz, azonban mi, ezen projekt folyamán nem ilyen adatforrásokkal dolgoztunk, hanem sokrétű, egymástól független és különböző szerkezetű adathalmazokkal. Így egy ilyen eszköz használata nem volt célravezető, továbbá a meteorológiai jellegű adatok elemzéséhez standard eszközök mind open source szoftverek, melyek szépen el vannak látva Python könyvtárakkal, API-kal. Természetesen ennek a megközelítésnek is megvannak a hátrányai: mindegyik fájl formátumhoz külön parancssori eszköz telepítése, új API-k kezelésének elsajátítása...

Továbbá: a dokumentációban található program kódrészletek saját munka eredményei.

A projektben részt vevő személyek

Attól függetlenül, hogy a pályázati időszakban még úgy tűnt, hogy a munka során egyedül Dr. Pödör Zoltán szakmai konzulensemre számíthatok, rajta keresztül több, a szakmai kérdésekben rendkívül sok segítséget nyújtó személlyel dolgozhattam együtt.

Dr. Pödör Zoltán, szakmai konzulens
Az ÚNKP projekt koordinátora, adatbányászati technikák és ezek elsajátításához szükséges anyagok forrása.

Dr. Gálos Borbála
Az erdészeti kapcsolatunk, az ő irányadásával nyílt új fejezet a projekt folyásában (indexek, modellek összehasonlítása, új adatforrások).

Molnár András
Doktorandusz hallgató, vele együtt fedeztük fel az új adatforrásokat és dolgoztunk a projekt második felén.

Adatok
Az elemzések első és legfontosabb lépése a sok és jó minőségű adat összegyűjtése.

Milyen mennyiségtől tekintünk egy adathalmazt Big Data-nak?
Tudományos körökön kívül, egyéni felhasználóként akár az 1-től 3 gigabájt (10^9 bájt) méretig terjedő adathalmazok már hatalmasnak számítanak, főleg a személyi számítógépek drága és limitált memóriakapacitása miatt. Az elemzéshez használt adatokat azért kell a memóriában tárolni, hogy az egyes Data Science könyvtárak által alkalmazott algoritmusok lefutása viszonylag gyorsan lefuthasson, pár GB-nyi adattal (1-5 évig terjedő adathalmaz) PC-ken is pillanatok alatt képesek ezek a programok eredményt generálni, főleg azért, mert ezen méreteknél nagyságrendekkel nagyobb mennyiségű adat feldolgozására lettek felkészítve.
Viszonytásképp a szuperszámítógépekkel felszerelt intézetek és vállalatok napi szinten petabájt (10^{15} bájt) nagyságrendű adatbázisokkal dolgoznak. Ebből következik, hogy maga a „Big Data" elnevezés erősen szituáció függő.

Mitől lesz jó minőségű az adat?
Az adatok összegyűjtése, azok szűrése és manipulálása külön tudományággá vált. Erről szól Hadley Wickham [4] tanulmánya is, mely kiinduló standarddá vált adat elemző körökben. Az ebben az anyagban megfogalmazott alapelvek:

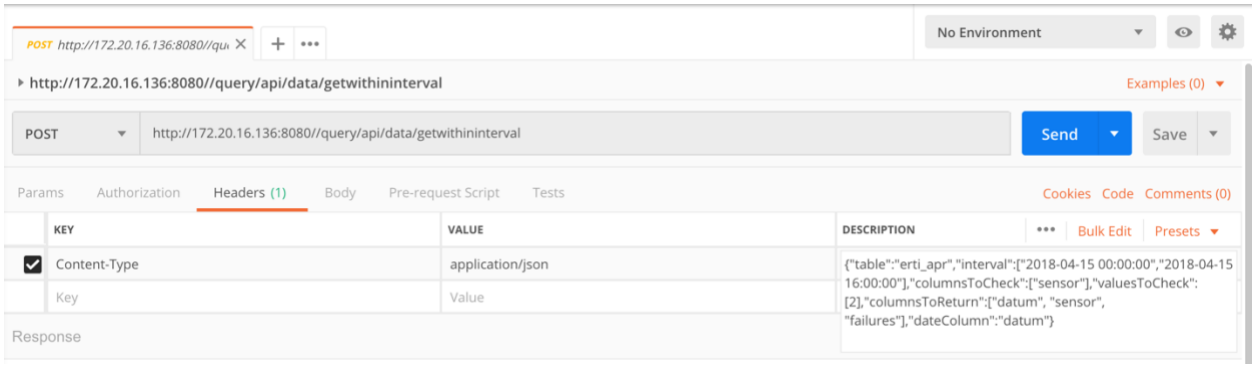
1. Az egyes jelenségeket leíró adathalmazok feloszthatók tulajdonságokra és megvalósulásokra, azaz attribútumokra és ezek alapján rekordokra.
2. Az attribútumok a jelenség elemi tulajdonságait jelentik, így elemi adattípusokkal leírhatók.
3. A jelenséget leíró adathalmazok nem feltétlenül tabulárisak (táblázatos felépítésűek), bár a különböző adatbázis elmező szoftverek a tabuláris adatbázisok elemzésire már jól optimalizáltak.
4. Ne féljünk belenyúlni a megkapott adathalmazba! Ez az úgynevezett „data wrangling" folyamat, melyet jó közelítéssel adatmanipulációnak nevezhetünk. Ezen lépés teszi ki az adatelemzői procedúra oroszlánrészét, nem véletlenül: a hibás adatok (rossz mérések, duplikátumok stb.) használata rossz és elvi hibás következtetések levonását eredményezik.
5. Ügyeljünk arra, hogy minimalizáljuk az adatmanipuláció során elvesztett adat mennyiségét, ugyanis a hiányzó adat sokszor hiányzó információt jelent.

A projektben használt adatforrások

- Az intézeti SensorHUB környezet 2015 év szeptemberéig visszamenőleg tárol adatokat, melyek forrása az Erdészeti Tudományos Intézet (ERTI) által működtetett 6 mérőállomás (Püspökladány, Sárvár, Kecel, Napkor, Karád, Pilisszentlélek). Itt a Query API segítségével JSON formátumban kérhetők le az adatok.
- Copernicus Climate Data Store – bárki által elérhető klímaadatok online tárháza, mely a bemutatott CDS API-val kezelhető. Főleg az ERA5 adatbázissal dolgoztunk (egész Földre kiterjedő mérési adatok), mely GRIB formátumban tárolja az eredményeket.
- ESGF Node – A CDS-hez hasonlóan ez is Európai Úniós kezdeményezés, innen netCDF formátumú fájlok érhetőek el, szintén globális lefedettséggel.

Az adatok összegyűjtése

- SensorHUB



Egyszerű HTTP lekérdezéssel

```
from cassandra.cluster import Cluster
from cassandra.metadata import KeyspaceMetadata

import pandas as pd
import matplotlib.pyplot as plt
from matplotlib import style
from matplotlib.dates import DateFormatter
style.use("ggplot")

# connect to Cassandra cluster
cluster = Cluster(["172.20.16.137"], port=30241)
keyspace_name = "sensorhub"
session = cluster.connect(keyspace_name)

# create pandas dataframe
query = "SELECT sensor, datum, temperatureat2meter FROM erti_new WHERE datum >= '2017-01-01 00:00:00' AND  datum <= '2017-01-31 23:59:00' AND sensor=1"
df = pd.DataFrame(list(session.execute(query)))
```

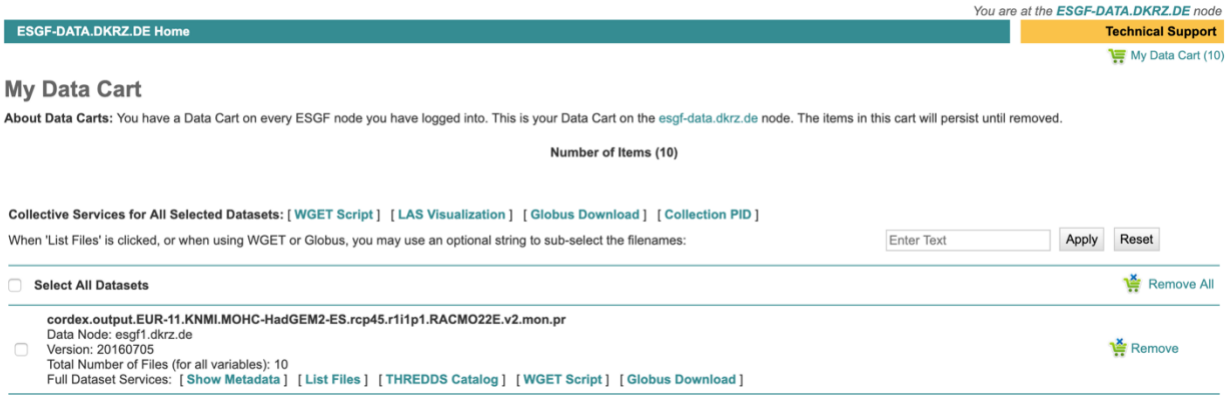
Pythonból a cassandra-driver könyvtár függvényeivel

- Copernicus Climate Data Store

```
import cdsapi

c = cdsapi.Client()

c.retrieve("reanalysis-era5-pressure-levels",
    {
        "variable": "temperature",
        "pressure_level": "1000",
        "product_type": "reanalysis",
        "year": "2008",
        "month": "01",
        "day": "01",
        "time": "12:00",
        "format": "grib"
    },
        "download.grib")
```

ERA5 adatok lekérdezése CDS API segítségével

- ESGF Node



ESGF Node Data Cart egyik eleme

```
(base) ~$ ./wget-20190313103147.sh -H https://esgf-data.dkrz.de/esgf-idp/openid/csana23
```

Data Cart kicsomagolása wget parancssori eszközzel és OpenID-val

Az adatok feldolgozása, előkészítése elemzésre

- SensorHUB

Getting data from more sensors

Include all sensors.

```
In [3]: query2 = "SELECT sensor, datum, temperatureat2meter FROM erti_new WHERE datum >= '2017-01-01 00:00:00' AND  datum
        <= '2017-01-31 23:59:59' ALLOW FILTERING"
        df2 = pd.DataFrame(list(session.execute(query2)))

In [4]: df2.head()
```

Out[4]:

	sensor	datum	temperatureat2meter
0	5	2017-01-01 00:00:03	-1.7
1	5	2017-01-01 00:10:03	-1.6
2	5	2017-01-01 00:20:03	-1.8
3	5	2017-01-01 00:40:13	-1.9
4	5	2017-01-01 00:50:03	-1.7

Cleaning up the DataFrame

```
In [5]: df2['temperatureat2meter'] = df2['temperatureat2meter'].replace(to_replace=[-888.8, -777.7, -999.9], value=np.NaN)

In [6]: df2.isnull().values.any()
Out[6]: True
```

A lekérdezés eredményét egy tabuláris adatszerkezetté konvertáljuk, majd a hibás adatokat NaN-al jelöljük

- Copernicus Climate Data Store
  - GRIB fájlok értelmezése ecCodes parancssori eszközzel

```
16  cd CMakeFiles/
17  cd grib_ls.dir/
18  ls
19  cmake cmake_clean.cmake
20  cd ..
21  cmake grib_ls.dir/
```

ecCodes telepítése CMake parancsori eszközzel

```
(base) src$ grib_ls download.grib
download.grib
edition      centre       typeOfLevel  level        dataDate     stepRange    dataType     shortName    packingType  gridType
1            ecmf         isobaricInhPa  1000         20080101     0            an           t            grid_simple  regular_ll
1 of 1 messages in download.grib

1 of 1 total messages in 1 files
```

ecCodes grib_ls parancs a GRIB fájl metaadatainak listázására

- GRIB fájl átkonvertálása CSV formátumba ecCodes segítségével

```
(base) src$  grib_get_data -w time=1200 download.grib > download.csv
```

ecCodes parancs CSV formátumba konvertáláshoz

- Kapott CSV fájl formázása Python-nal

```
#!/usr/bin/env python

with open('download.csv', 'r') as f_in, open('formatted_download.csv', 'w') as f_out:
    f_out.write(next(f_in))
    [f_out.write(','.join(line.split()) + '\n') for line in f_in]
```

Utolsó simítások a CSV fájlon

- CSV fájl betöltése pandas DataFrame-be (tabuláris adataszerkezet)

```
In [2]: import pandas as pd

        import matplotlib.pyplot as plt
        from matplotlib import style
        from matplotlib.dates import DateFormatter

        import numpy as np

In [3]: df = pd.read_csv('formatted_download.csv')

In [4]: # Value is in Kelvin
        df.head()

Out[4]:
```

	Latitude	Longitude	Value
0	90.0	0.00	249.149124
1	90.0	0.25	249.149124
2	90.0	0.50	249.149124
3	90.0	0.75	249.149124
4	90.0	1.00	249.149124

CSV fájl (GRIB-ből átalakítva) betöltése memóriába Jupyter Notebook-ban

- ESGF Node

```
In [1]: from netCDF4 import Dataset
        import pandas as pd
        import numpy
        from cdo import *
        import xarray as xr
        import cartopy.crs as ccrs
        import matplotlib.pyplot as plt

In [2]: dataset = Dataset("testCDF.nc", "r")

In [3]: print(dataset.dimensions)

        OrderedDict([('bnds', <class 'netCDF4._netCDF4.Dimension'>: name = 'bnds', size = 2
        ), ('rlon', <class 'netCDF4._netCDF4.Dimension'>: name = 'rlon', size = 424
        ), ('rlat', <class 'netCDF4._netCDF4.Dimension'>: name = 'rlat', size = 412
        ), ('time', <class 'netCDF4._netCDF4.Dimension'> (unlimited): name = 'time', size = 60
        )])

In [4]: for i in dataset.variables:
            print(i)

        rotated_pole
        rlon
        lon
        rlat
        lat
        time
        time_bnds
        pr
```

Az ESGF Node Data Cart-ból letöltött és kicsomagolt netCDF fájlok betöltése memóriába Jupyter notebook-ban

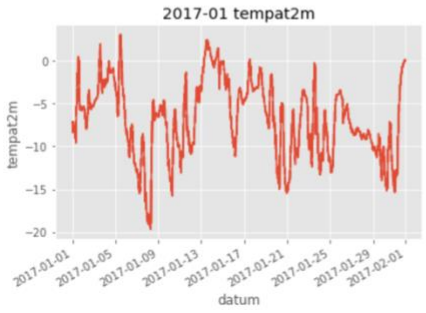
Adatvizualizáció, az adathalmazok felfedezése, „data wrangling”

- SensorHUB

How does our DataFrame look like?

```
In [6]: style.use('ggplot')

        my_dates = df['datum']

        plt.figure()
        fig, ax = plt.subplots()

        ax.plot(my_dates, df['temperatureat2meter'])

        fig.autofmt_xdate()
        ax.set_title('2017-01 tempat2m')

        plt.xlabel('datum')
        plt.ylabel('tempat2m')

        plt.grid(True)
        plt.show()
```

```
<Figure size 432x288 with 0 Axes>
```



A memóriába betöltött SensorHUB adatok megjelenítése

Cleaning up the DataFrame

```
In [14]: df2['temperatureat2meter'] = df2['temperatureat2meter'].replace(to_replace=[-888.8, -777.7, -999.9], value=np.NaN)

In [15]: df2.isnull().values.any()
Out[15]: True
```

A hibás adatok megjelölése NaN (null) értékkel

Making sense of the dataset - separating datetime

```
In [11]: df['time'] = df['datum'].dt.time

         print(df.head())
```

```
   sensor               datum  temperatureat2meter  MA_500       date  week  \
0       1 2017-01-01 00:00:01                 -7.2     NaN 2017-01-01    52
1       1 2017-01-01 00:10:01                 -7.2     NaN 2017-01-01    52
2       1 2017-01-01 00:20:01                 -7.1     NaN 2017-01-01    52
3       1 2017-01-01 00:40:11                 -8.4     NaN 2017-01-01    52
4       1 2017-01-01 00:50:01                 -7.9     NaN 2017-01-01    52

       time
0  00:00:01
1  00:10:01
2  00:20:01
3  00:40:11
4  00:50:01
```

Attribútum halmaz bővítése az idő adatok felbontásával

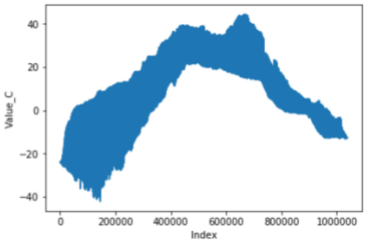
- Copernicus Climate Data Store

```
In [5]: df.columns.values
Out[5]: array(['Latitude', 'Longitude', 'Value'], dtype=object)

In [6]: df['Value_C'] = df['Value'] - 273.15
```

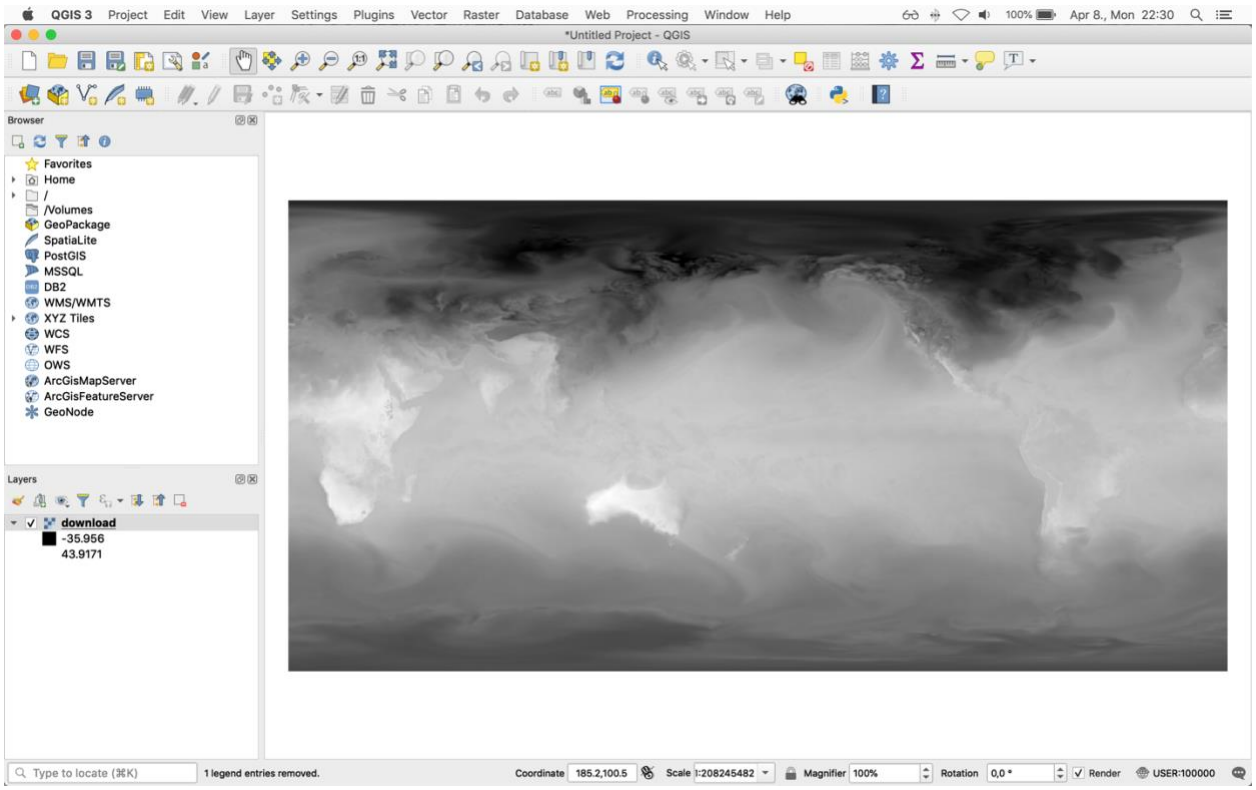
A GRIB fájl hőmérséklet adatainak átváltása Kelvinből Celsiusba

Plotting the data

```
In [14]: plt.subplot()

x = df.index

plt.plot(x, df['Value_C'])
plt.xlabel('Index')
plt.ylabel('Value_C')
plt.show()
```



A fájl tartalmának ábrázolása

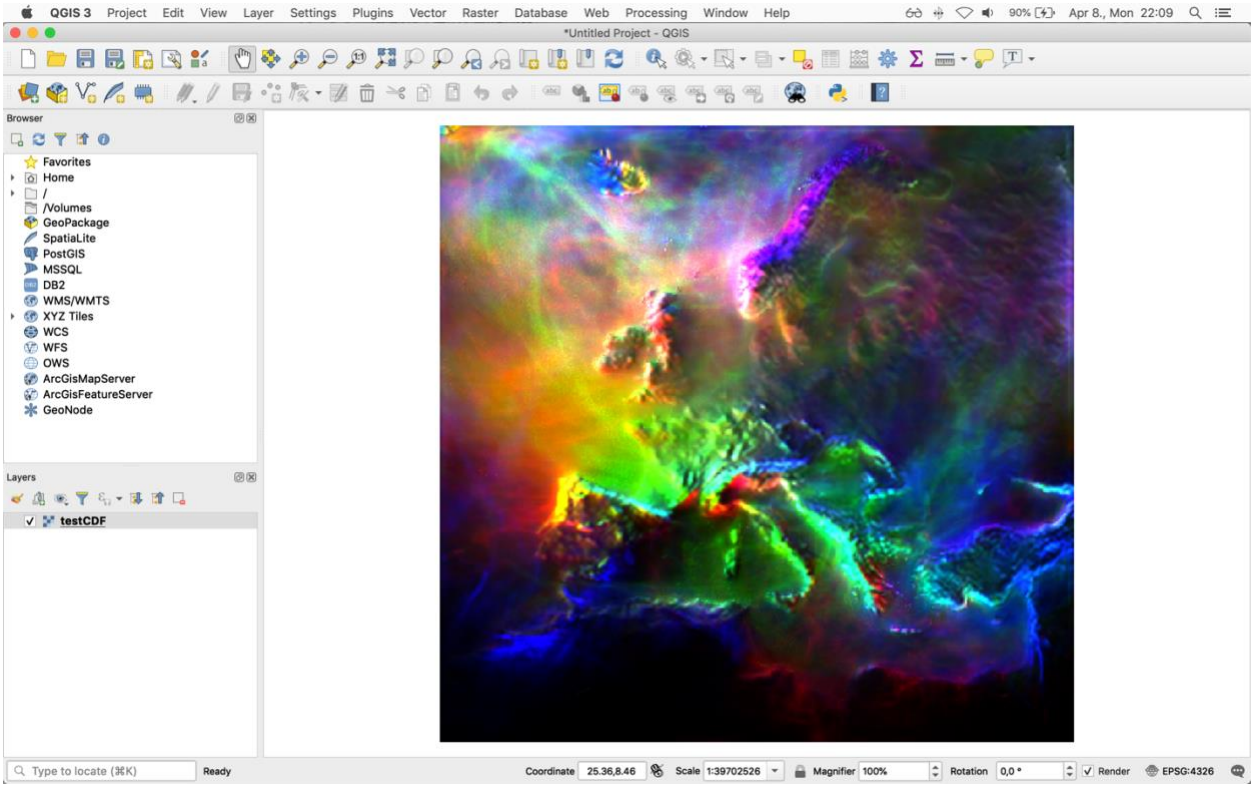


GRIB fájl megjelenítése a QGIS alkalmazásban

- ESGF Node

```
In [6]: cdo = Cdo()

In [7]: cdo.infov(input='testCDF.nc')

Out[7]: ['-1 :       Date     Time   Level Gridsize    Miss :     Minimum        Mean     Maximum : Parameter name',
         '1 : 2006-01-16 00:00:00       0   174688       0 :      0.0000  3.1469e-05  0.00042979 : pr',
         '2 : 2006-02-16 00:00:00       0   174688       0 :      0.0000  2.9225e-05  0.00055680 : pr',
         '3 : 2006-03-16 00:00:00       0   174688       0 :      0.0000  2.1257e-05  0.00034298 : pr',
         '4 : 2006-04-16 00:00:00       0   174688       0 :      0.0000  1.5059e-05  0.00023509 : pr',
         '5 : 2006-05-16 00:00:00       0   174688       0 :      0.0000  1.5268e-05  0.00015464 : pr',
         '6 : 2006-06-16 00:00:00       0   174688       0 :      0.0000  1.7751e-05  0.00019486 : pr',
         '7 : 2006-07-16 00:00:00       0   174688       0 :      0.0000  1.3065e-05  0.00018736 : pr',
         '8 : 2006-08-16 00:00:00       0   174688       0 :      0.0000  1.7605e-05  0.00016702 : pr',
         '9 : 2006-09-16 00:00:00       0   174688       0 :      0.0000  2.3436e-05  0.00025481 : pr',
         '10 : 2006-10-16 00:00:00      0   174688       0 :      0.0000  2.7215e-05  0.00028551 : pr',
         '11 : 2006-11-16 00:00:00      0   174688       0 :      0.0000  3.3106e-05  0.00043728 : pr',
         '12 : 2006-12-16 00:00:00      0   174688       0 :      0.0000  3.4009e-05  0.00044165 : pr',
         '13 : 2007-01-16 00:00:00      0   174688       0 :      0.0000  2.8870e-05  0.00038074 : pr',
         '14 : 2007-02-16 00:00:00      0   174688       0 :      0.0000  2.8055e-05  0.00026122 : pr',
         '15 : 2007-03-16 00:00:00      0   174688       0 :  3.7606e-14  2.5631e-05  0.00045462 : pr',
         '16 : 2007-04-16 00:00:00      0   174688       0 :      0.0000  2.1457e-05  0.00035148 : pr',
         '17 : 2007-05-16 00:00:00      0   174688       0 :      0.0000  1.6183e-05  0.00016404 : pr',
         '18 : 2007-06-16 00:00:00      0   174688       0 :      0.0000  1.6490e-05  0.00024719 : pr',
         '19 : 2007-07-16 00:00:00      0   174688       0 :      0.0000  1.5748e-05  0.00016690 : pr',
         '20 : 2007-08-16 00:00:00      0   174688       0 :      0.0000  1.8088e-05  0.00017776 : pr',
         '21 : 2007-09-16 00:00:00      0   174688       0 :      0.0000  2.3927e-05  0.00026538 : pr',
         '22 : 2007-10-16 00:00:00      0   174688       0 :      0.0000  2.6807e-05  0.00028133 : pr',
         '23 : 2007-11-16 00:00:00      0   174688       0 :      0.0000  3.2165e-05  0.00026027 : pr',
         '24 : 2007-12-16 00:00:00      0   174688       0 :      0.0000  3.0696e-05  0.00055932 : pr',
         '25 : 2008-01-16 00:00:00      0   174688       0 :      0.0000  2.9947e-05  0.00044463 : pr',
         '26 : 2008-02-16 00:00:00      0   174688       0 :      0.0000  2.8732e-05  0.00034564 : pr',
         '27 : 2008-03-16 00:00:00      0   174688       0 :      0.0000  2.4070e-05  0.00033787 : pr',
         '28 : 2008-04-16 00:00:00      0   174688       0 :      0.0000  1.8678e-05  0.00024261 : pr',
         '29 : 2008-05-16 00:00:00      0   174688       0 :      0.0000  1.8342e-05  0.00018769 : pr',
         '30 : 2008-06-16 00:00:00      0   174688       0 :      0.0000  1.4781e-05  0.00016467 : pr',
         '31 : 2008-07-16 00:00:00      0   174688       0 :      0.0000  1.4768e-05  0.00014919 : pr',
```

CDO Python könyvtár használata netCDF fájl részleteinek feltárására



netCDF fájl megjelenítése a QGIS programban

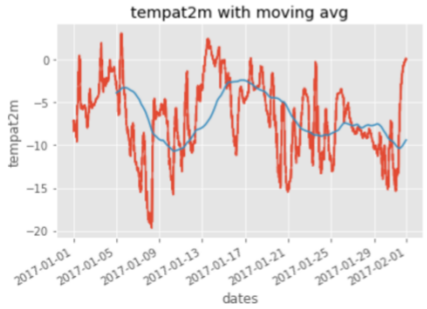
Adatelemzés, értelmezés

- SensorHUB
  - Itt első körben egy 2017 januárjában, az első szenzor által mért hőmérsékleti adatokat tartalmazó halmazt vizsgálunk. Az adathalmaz relatíve kicsi, az eredmények gyors kinyerése (és értelmezhetőségének) érdekében.

With moving average (per 500 data)

We have to insert another column into our dataset, which will hold the moving averages calculated after every 500 records.

```
In [7]: df['MA_500'] = df['temperatureat2meter'].rolling(500).mean()

        plt.figure()
        fig, ax = plt.subplots()

        plt.plot(my_dates, df['temperatureat2meter'], label="tempat2m")
        plt.plot(my_dates, df['MA_500'], label="MA_500")

        plt.xlabel('dates')
        plt.ylabel('tempat2m')

        myFmt = DateFormatter("%Y-%m-%d")
        ax.xaxis.set_major_formatter(myFmt)
        fig.autofmt_xdate()
        ax.set_title('tempat2m with moving avg')
Out[7]: Text(0.5, 1.0, 'tempat2m with moving avg')

        <Figure size 432x288 with 0 Axes>
```

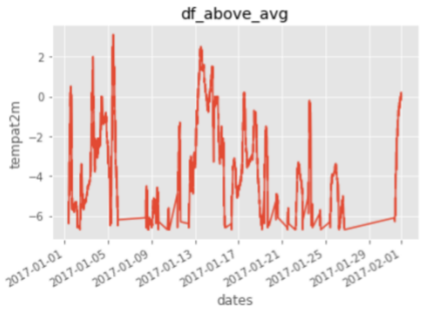


Mozgóátlag számolása és megjelenítése

Above average temperatures

```
In [8]: df_above_avg = df.loc[df['temperatureat2meter'] > tempat2mAvg, :]

        plt.figure()
        fig, ax = plt.subplots()

        dates = df_above_avg['datum'].tolist()

        ax.plot(dates, df_above_avg['temperatureat2meter'])

        plt.xlabel('dates')
        plt.ylabel('tempat2m')

        myFmt = DateFormatter("%Y-%m-%d")
        ax.xaxis.set_major_formatter(myFmt)
        ax.set_title('df_above_avg')
        fig.autofmt_xdate()
```

```
<Figure size 432x288 with 0 Axes>
```



A mintaátlag feletti értékek számolása és megjelenítése

Innentől kezdve bármilyen statisztikai, adatelemző metódust, függvényt alkalmazva nyerhetünk ki információt az adott adathalmazból. A következőkben ezekre láthatunk néhány példát.

Splitting up the dataset into weeks

```
In [9]: df['date'] = df['datum'].dt.date
        df['date'] = pd.to_datetime(df['date'])
        df['week'] = df['date'].dt.week

        print(df.head())
```

```
   sensor              datum  temperatureat2meter  MA_500       date  week
0       1 2017-01-01 00:00:01                 -7.2     NaN 2017-01-01    52
1       1 2017-01-01 00:10:01                 -7.2     NaN 2017-01-01    52
2       1 2017-01-01 00:20:01                 -7.1     NaN 2017-01-01    52
3       1 2017-01-01 00:40:11                 -8.4     NaN 2017-01-01    52
4       1 2017-01-01 00:50:01                 -7.9     NaN 2017-01-01    52
```

Average tempat2m by week

```
In [10]: weekly_avg = df.groupby('week')['temperatureat2meter'].mean()
         print(weekly_avg)
```

```
week
1    -6.779839
2    -5.035640
3    -6.530840
4    -8.407135
5    -8.961446
52   -5.408800
Name: temperatureat2meter, dtype: float64
```

Az adathalmaz attribútumainak bővítése egy heti bontással, továbbá a heti átlaghőmérsékletek kalkulálása

Plotting average tempat2m from each sensor

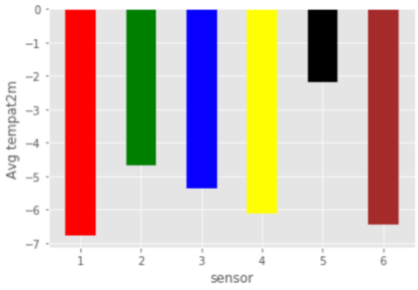
```
In [16]: grouped = df2['temperatureat2meter'].groupby(df2['sensor'])

In [17]: mean_of_grouped = grouped.mean()
         print(mean_of_grouped)
```

```
sensor
1   -6.794749
2   -4.683816
3   -5.370310
4   -6.124344
5   -2.174382
6   -6.450526
Name: temperatureat2meter, dtype: float64
```

It might occur to us that the 5th sensor has measured a significantly lower average temperature... Why is that? (we won't explore the reasons, for now)

Bővítettük az adathalmazt az összes többi szenzor adataival és szenzorokra vetített bontásban számoltunk átlaghőmérsékletet – vegyük észre, hogy az 5. szenzorhoz tartozó érték különösen alacsony

```
In [19]: plt.figure()

         # random_color = np.random.choice(range(256), size=3)
         colors = ['red', 'green', 'blue', 'yellow', 'black', 'brown']

         plt.bar(x, y, width=.5, color=colors)

         plt.xlabel('sensor')
         plt.ylabel('Avg tempat2m')

         plt.grid(True)
         plt.show()
```

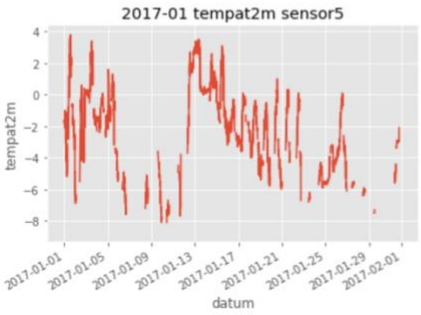


Ezt egyszerűbben levonhatjuk, mint következtetés, hogyha grafikus módon jelenítjük meg a számolt értékeket. De mi lehet ennek az oka?

Próbáljuk meg az előzőekhez hasonlóan vizualizálni az 5. szenzor által mért adatokat!

```
In [20]: style.use('ggplot')

         my_dates = df2[df2.sensor == 5].datum

         plt.figure()
         fig, ax = plt.subplots()

         ax.plot(my_dates, df2[df2.sensor == 5].temperatureat2meter)

         fig.autofmt_xdate()
         ax.set_title('2017-01 tempat2m sensor5')

         plt.xlabel('datum')
         plt.ylabel('tempat2m')

         plt.grid(True)
         plt.show()
```

```
<Figure size 432x288 with 0 Axes>
```



Szépen kivehető, hogy az 5. szenzortól a vizsgált időszak egy jelentős hányadában nem érkeztek adatok!

Így nem meglepő, hogy a számolt átlaghőmérséklet ennyire eltért a többi szenzor által mért adatokból számolt átlagoktól.

További elemzések:

- Cohen-féle hatásnagyság (Cohen effect size) – Két minta várható értékének különbsége szórásban kifejezve, megadja a standardizált különbséget két átlag között. 0,8-as érték felett a hatásnagyság nagy.

Calculating Cohen's effect size for two groups

```
In [21]: def CohenEffectSize(group1, group2):
             """Computes Cohen's effect size for two groups.

             group1: Series or DataFrame
             group2: Series or DataFrame

             returns: float if the arguments are Series;
                      Series if the arguments are DataFrames
             """
             diff = group1.mean() - group2.mean()

             var1 = group1.var()
             var2 = group2.var()
             n1, n2 = len(group1), len(group2)

             pooled_var = (n1 * var1 + n2 * var2) / (n1 + n2)
             d = diff / np.sqrt(pooled_var)
             return d

In [22]: # creating two groups
         sensor5_temp = df2[df2.sensor == 5].temperatureat2meter
         sensor2_temp = df2[df2.sensor == 2].temperatureat2meter

In [24]: CohenEffectSize(sensor5_temp, sensor2_temp)
Out[24]: 0.7396660964541354
```

Jelentős a hatásnagyság az egyes és a kettes szenzorok által mért adatok között

- Pearson-féle korrelációs együttható

Pearson correlation

Calculating the Pearson correlation between two attributes: tempat2m, and windspeed.

```
In [12]: df3['windspeed'].head()
Out[12]: 0    1.1
         1    0.9
         2    0.9
         3    0.8
         4    0.9
         Name: windspeed, dtype: float64

In [13]: pearsonr(df3['windspeed'], df3['temperatureat2meter'])
Out[13]: (0.9998943946698517, 0.0)
```

Calculating the Pearson correlation between two attributes: tempat2m, and relativehumidity.

```
In [14]: df3['relativehumidity'].head()
Out[14]: 0    73.6
         1    72.6
         2    73.3
         3    72.2
         4    70.8
         Name: relativehumidity, dtype: float64

In [15]: pearsonr(df3['relativehumidity'], df3['temperatureat2meter'])
Out[15]: (0.9993104307407322, 0.0)
```

Pearson-féle korrelációs együttható számolása új adathalmazra, több attribútum között. Úgy tűnik, hogy jó közelítéssel lineáris a kapcsolat a vizsgált attribútumok között.

- Lineáris regresszió
    - Új adathalmaz: januári hőmérséklet adatok az első szenzortól
    - Elvek: adathalmaz felbontása tréning és teszt részhalmazokra (az adathalmaz egy részén „betanítjuk” a modellt, a másik részén – teszt – pedig ellenőrizzük a kész modellt), a kapott modell megjelenítése

Linear regression with scikit-learn

Importing the modules

```
In [15]: import pandas as pd

         import matplotlib.pyplot as plt
         from matplotlib import style
         from matplotlib.dates import DateFormatter

         import numpy as np
         from sklearn import datasets, linear_model
         from sklearn.metrics import mean_squared_error, r2_score

         from cassandra.cluster import Cluster
         from cassandra.metadata import KeyspaceMetadata
```

Connecting to SensorHUB

```
In [3]: cluster = Cluster(["172.20.16.137"], port=30241)
        keyspace_name = "sensorhub"
        session = cluster.connect(keyspace_name)
```

Getting data

```
In [4]: query = "SELECT sensor, datum, temperatureat2meter, windspeed FROM erti_new WHERE datum >= '2017-01-01 00:00:00'
        AND  datum <= '2017-01-31 23:59:00' AND sensor=1"
        df = pd.DataFrame(list(session.execute(query)))
```

Adatok beszerzése a már ismert módon

Clean the dataframe

```
In [27]: df['temperatureat2meter'] = df['temperatureat2meter'].replace(to_replace=[-888.8, -777.7, -999.9], value=np.NaN)

In [28]: df['windspeed'] = df['windspeed'].replace(to_replace=[-888.8, -777.7, -999.9], value=np.NaN)

In [29]: filtered_df = df[df['temperatureat2meter'].notnull()]

In [30]: filtered_df = df[df['windspeed'].notnull()]

In [31]: filtered_df.isnull().values.any()
Out[31]: False
```

Adathalmazból hibás adatok kiszűrése

Splitting up the dataset

```
In [34]: # x train and test - tempat2m
         temp_X_train = pd.DataFrame(df['temperatureat2meter'].iloc[:1500])
         temp_X_test = pd.DataFrame(df['temperatureat2meter'].iloc[1501:])

In [35]: # y train and test - windspeed
         wind_Y_train = pd.DataFrame(df['windspeed'].iloc[:1500])
         wind_Y_test = pd.DataFrame(df['windspeed'].iloc[1501:])
```

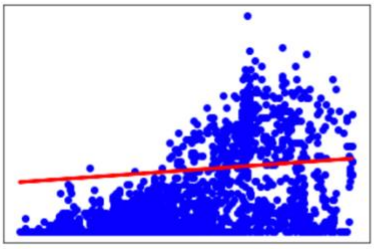
Adathalmaz felosztása training és test részhalmazokra

Instantiate regression model

```
In [36]: # skipped some of the cells above
         regr = linear_model.LinearRegression()

In [37]: # train the model
         regr.fit(temp_X_train, wind_Y_train)

Out[37]: LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,
                  normalize=False)

In [38]: # make predictions using the test set
         wind_y_pred = regr.predict(temp_X_test)
```

Regressziós modell példányosítása, felkészítése a training részhalmazon és előrejelzések kalkulálása a test részhalmazon

```
In [39]: #plot outputs
         plt.scatter(temp_X_test, wind_Y_test, color='blue')
         plt.plot(temp_X_test, wind_y_pred, color='red', linewidth=3)

         plt.xticks(())
         plt.yticks(())

         plt.show()
```



A kapott regressziós egyenes megjelenítése az aktuális adatok felett

Jól látható, hogy a regressziós egyenes nem pontosan becsüli meg a teszt részhalmaz értékeit, attól függetlenül, hogy az általános trendet jól „megtanulta”. Ennek több oka is van:

1. Kis méretű adathalmaz
2. Az adathalmaz rossz felosztása tréning és teszt részhalmazokra

Ezekből is következik, hogy a megfelelő adathalmaz kiválasztása és a részhalmazok kijelölése regressziós problémáknál kritikus.

- Copernicus Data Store, ESGF Node
    - Ezen masszív adatforrások további kutatást igényelnek
- Az elért eredmények:
    - GRIB és netCDF formátumú fájlok összegyűjtése
    - Értelmezése (parancssoros – ecCodes, CDO - vagy programozói eszközökkel)
    - Megjelenítése (Python, QGIS)
- Jelenleg fejlesztés alatt:
    - netCDF fájlokból indexek számolása a Kárpát-medencére: Módosított Ellenberg index, Forestry Aridity Index
    - Ezen indexek alapján az egyes klíma modellek összehasonlítása

A gépi tanulás alapjai

„If it's in Python then it's probably machine learning. On the other hand, if it's in PowerPoint then it's AI."

A gépi tanulás definíciója: a számítógépek azon képessége, hogy anélkül képesek tanulni, hogy erre külön meg lettek volna tanítva. [5]

Különböző ML (Machine Learning) rendszerek típusai, kategorizálási lehetőségek:

- Felügyelt, nem felügyelt, részben felügyelt és megerősített tanulás
    - Felügyelt tanulás esetén a rendszerbe bevitt adatok tartalmazzák a kívánt eredményeket (labels). Például: osztályozás (classification).
    - Regressziós problémák: adott értékek becslése, előrejelzése. (Lásd: lineáris regresszió az adatelemző szekcióban). Fajták: neurális hálók, döntési fák, stb.
    - Nem felügyelt tanulás: nem címkézett adat, azaz a kezdeti adathalmaz nem tartalmazza a kívánt eredményeket. Fajták: Klaszterezés (clustering), dimenzió redukció (dimension reduction), anomália detektálás, stb.
    - Részben felügyelt rendszerek: felügyelt és nem felügyelt megoldások hibridjei
    - Megerősített tanulás (Reinforcement Learning): a rendszer megfigyeli a környezetét és ennek hatására javaslatokat tesz, ezen javaslatok vagy helyesek vagy nem, ennek függvényében a rendszert vagy jutalmazzuk vagy nem. Például: DeepMind AlphaGo
- Online és batch tanulás („on the fly" tanulás és előre elkészített, konstans adathalmazon végzett tanulás)
    - Batch tanulás: offline tanulás, a rendszer a működése közben képtelen új adatot magába fogadni, az induláskor az elérhető összes adatot be kell tölteni
    - Online tanulás: szekvenciális adatbetöltés lehetősége, korlátozott memóriával rendelkező rendszerek estében preferált megoldás
- Példány alapú és modell alapú tanulás
    - Példány alapú tanulás: adott adatok között hasonlóságokat fedez fel a rendszer és ezen hasonlóságok alapján dönt egy még nem látott adatról
    - Modell alapú tanulás: attribútumhalmazok egybefűzésével tár fel kapcsolatot az egyes attribútumok között

Természetesen a gyakorlatban használatos ML rendszerek kategorizálása nem ennyire egyértelmű, hiszen ezen rendszerek legtöbbször a felsorolt megoldások keverékei.

Gépi tanulás eszközök
Mára rengeteg ML könyvtár, megoldás és keretrendszer létezik, álljon itt egy felsorolás a legismertebbekről. Ezen eszközök mindegyike nyílt forráskódú.

- Scikit-Learn
    - Python könyvtár
    - Könnyű használhatóság (lásd: lineáris regresszió az adatelemző szekcióban)
- Google TensorFlow
    - Komplexebb könvytár, mely támogatja a párhuzamosított számításokat (akár videókártyás rendszerekben)
- Keras
    - Magas szintű API
    - Többek között TensorFlow támogatással
    - Deep Learning
- PyTorch
    - Python könyvtár
    - TensorFlow-hoz hasonlóan elosztott számítások támogatása

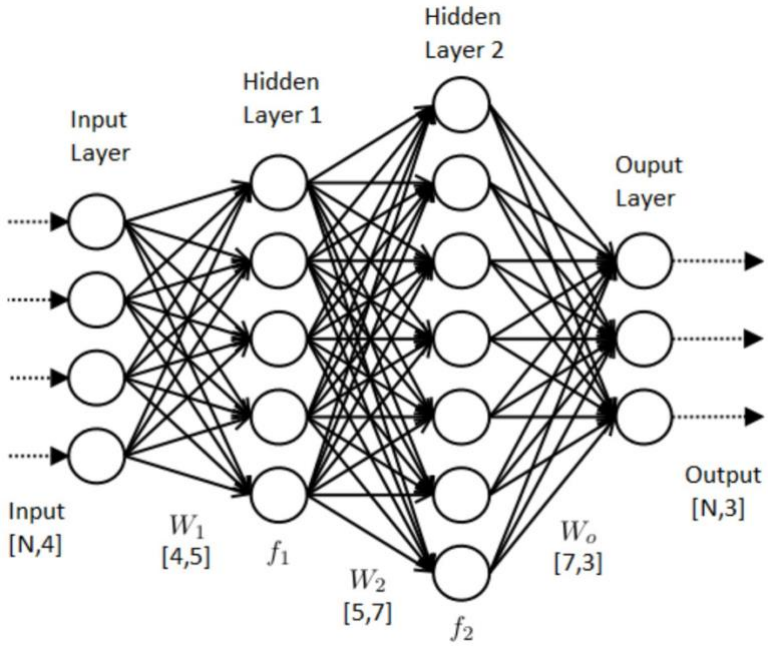
Neurális hálók

Mindezen megoldások működési elve a mesterséges neurális háló. Ennek koncepcionális alapja az emberi agy modellje: neuronok és ezek végei, valamint az őket összekötő inger átvivő anyagok, a neurotranszmitterek, szinapszisok.

Felépítés:

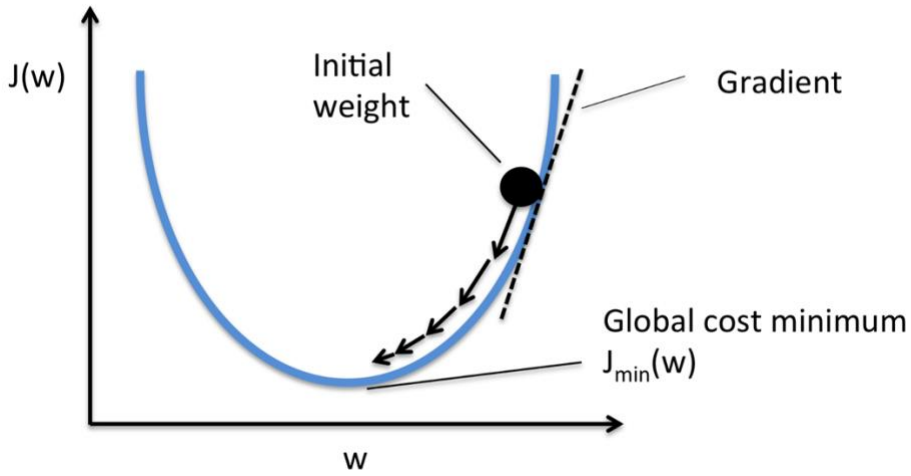
- Neuronok
    - Aktiváló függvény alapján vagy felébrednek vagy inaktívak maradnak.
- Kapcsolatok, prioritások és súlyok
    - A neuronok közötti kapcsolatokat definiáló függvények.
- Layer, réteg
    - A neuronok és ezek kapcsolatai rétegeket alkotnak. Két kötelező layer: input és output layer. Ha a bemeneti és kimeneti réteg között több, mint 1 réteg helyezkedik el, akkor a neurális hálót Deep Learning Neural Networknek nevezzük.

Neurális háló ábrázolása [6]:



Neurális háló vázlatos rajza. A körök a neuronok, idegsejtek, a nyilak az egyes neuronokat összekötő szinapszisok, kapcsoló függvények. Ezekhez definiálunk súlyokat, prioritásokat. Az egyes neuronok kimenete a következő rétegbeli neuronok bemenete.

A neurális háló szerkezete, az emberi agy idegsejt kapcsolódásaihoz hasonlóan, nem állandó. Címkézett (labeled) adathalmaz esetén tudjuk követni, hogy a neurális háló a bementi adatokból jó következtetést vont-e le, azaz sikeres volt-e a betanítási fázis (pl. Klímamodellek esetén a háló jól jósolta meg a következő napi időjárást). Így a helyes és rossz válaszokból létre tudunk hozni egy error (hiba) függvényt. Ezen hibafüggvény minimiumának megtalálásával (gradient descent) elérhetjük, hogy a modellünk egyre pontosabban működjön és nagyobb arányban hozzon jó döntéseket. Ezen minimumkeresés során alakul, változik (automatikusan) a neurális háló szerkezete. Ezt nevezzük backpropagation (visszacsatolási) elvnek.



Súlyok és prioritások átalakítása ez error függvény minimumának elérése közben

Összefoglalás és kitekintés

Az ösztöndíjas periódus alatt számos, az adatelemzéssel kapcsolatos irodalmat sikerült feldolgozni és ezek alapján egy komprehenzív adatelemzési workflowt kialakítani open source környezetben. Megismertem a tudományos körökben standardnak számító adatforrásokat, formátumokat és elemzési eszközöket, a Big Data világának matematikai, statisztikai alapjait. A gépi tanulás elméleti hátterét feltérképeztem, a gyakorlati implementációt kollégáimmal megkezdtük, jelenleg ezen dolgozunk TensorFlow és Scikit-Learn környezetekben.

Továbbá köszönöm a munkában részt vevő személyek támogatását, együttműködését és iránymutatását.

Sopron, 2019. április 10.

Hivatkozások

[1] SensorHUB, < https://www.aut.bme.hu/Pages/Research/SensorHUB >, utolsó megtekintés: 2019. április 9.
[2] Mark Stacey, Joe Salvatore and Adam Jorgensen – Visual Intelligence: Microsoft Tools and Techniques for Visualizing Data
[3] Adam Aspin – Hight Impact Data Visualization with Power View, Power Map and PowerBI
[4] Tidy Data – Hadley Wickham, < https://vita.had.co.nz/papers/tidy-data.pdf >, utolsó megtekintés: 2019. április 9.
[5] Hands-On Machine Learning with Scikit-Learn & TensorFlow – Aurélien Géron, 2017, ISBN: 978-1-491-96229-9
[6] Artificial neural networks < https://medium.com/coinmonks/the-artificial-neural-networks-handbook-part-1-f9ceb0e376b4 >, utolsó megtekintés: 2019. április 9.