Fakultet kemijskog inženjerstva i tehnologije
Zavod za mjerenja i automatsko vođenje procesa

**Metode umjetne inteligencije u kemijskom inženjerstvu**

Statistička analiza i predobrada podataka
Vrednovanje rezultata modela

Osnovni primjeri funkcija u Pythonu

# **Instalacija distribucije *Anaconda* i Python paketa**

Distribucija Anaconda daje cjeloviti i jednostavni pristup programiranju u programskom jeziku Python za početnike. Distribuciju Anaconda besplatno preuzmite ovdje: <https://www.anaconda.com/products/individual>

Pokretanjem **Anaconda Navigatora** nakon instalacije, otvara se okruženje u kojemu su prikazani instalirani programi i oni koji se mogu instalirati u okviru Anaconda Navigatora.
Utom setu programa, dolazi i interaktivno razvojno okruženje (IDE) **Spyder** koji jednostavnim i intuitivnim izgledom olakšava pisanje i pokretanje skripti u Pythonu.

Spyder dolazi sa nekolicinom već instaliranim Python paketima poput NumPy, SciPy i drugih, dok će se ostatak potrebnih paketa za praćenje ovog seminara preuzeti na sljedeći način:

U Anaconda Navigatoru pokrenut će se CMD.exe Prompt što je crni prozor poznatiji kao *Command Prompt* u kojemu će se upisati sljedeća sintaksa:

* *pip install ime\_paketa*

**na primjer:** pip install sklearn

Nakon instalacije paketa upisuje se sljedeći, ako je potrebno.

Instalirajte sljedeće pakete:

**numpy, scipy, sklearn, matplotlib, pyts, hampel, seaborn, sklearn**

Često se na osnovi nekih paketa grade besplatni paketi koji za svoj rad zahtjevaju određene verzije paketa na kojima su izgrađeni. Ako se primjerice dogodi da imamo stariju verziju nekog paketa, sljedećom sintaksom možemo ažurirati paket:

*pip install --upgrade ime\_paketa*

**na primjer:** pip install --upgrade numpy

Možemo instalirati i određenu verziju paketa(brojevi verzija dostupni u dokumentaciji paketa na web stranicama):

*pip install Ime\_paketa==verzija*

**na primjer:** pip install numpy==1.20

# **Deskriptivna statistika**

import pandas as pd

from pandas import Series

import numpy as np

import matplotlib.pyplot as plt

from scipy import stats

from hampel import hampel

''' unos podataka u python pomoću Pandas paketa'''

Data = pd.read\_excel('Primjer\_1.xlsx', header = 0)

T\_ul = Data['Tul']

T\_sred = Data['T\_sred']

p\_sred = Data['p\_sredina']

m\_ref = Data['m\_refluks']

lr\_benzin = Data['Benzin\_LR']

# Izbacivanje stupca iz podataka

Data\_bez\_Tul = Data.drop('Tul', axis = 1)

# Definiranje novoga stupca

Data\_bez\_Tul['Novi\_Tul'] = Data['Tul']

''' Deskriptivna statistika '''

# Pandas - sum, median, mean, mode, max, min, std, var, describe

print(Data['Tul'].describe())

# numpy - sum, median, mean, max, min, std, var, quantile

print(np.mean(Data['Tul']))

print(np.quantile(Data['Tul'], 0.25))

# stats scipy - kurtosis, skew, var - varijanca, describe, iqr - interkvartil

print(stats.iqr(Data['Tul']))

''' Vizualna analiza podataka - matplotlib '''

#plt.boxplot(T\_ul)

#plt.hist(lr\_benzin, bins = 200)

#plt.plot(T\_sred)

# **Predobrada podataka**

''' Konverzija podataka - predobrada podataka za strojno učenje'''

# Pandas

T\_ul = T\_ul.values

lr\_benzin = lr\_benzin.values

# Numpy

T\_sred = np.array(T\_sred)

''' Imputacija - predobrada nedostajućih vrijednosti unutar podataka'''

#Pandas - fillna(), interpolate()

m\_ref\_imp\_lin = m\_ref.interpolate(method='linear') # Opcije argumenta method:'polynomial', 'spline', 'barycentric', 'krogh', 'piecewise\_polynomial', 'spline', 'pchip'

m\_ref\_imp\_cubSp = m\_ref.interpolate(method='spline', order = 2) # cubic spline primjer (order 2 - kvadratna)

''' Detekcija ekstremnih vrijednosti i imputacija '''

A = Data['Tul']

# Detekcija

outliers = hampel(A, window\_size=5, n=3) # window\_size - veličina kliznoga prozora za detekciju outliera, n - sigma: broj standardnih devijacija za detekciju outliera

print("Outliers: ", outliers)

# Izmjena outlera vrijednostima dobivenim roling medianom

ts\_imputation = hampel(A, window\_size=5, n=1, imputation=True)

# Vizualni prikaz uklanjanja outliera

# A.plot(style="k-")

# ts\_imputation.plot(style="g-")

# plt.show()

''' Sigmaclip - agresivno uklanjanje ekstremnih vrijednosti'''

# sigmaclip kompletno briše red iz niza za razliku od navedene metode

Niz\_podataka = T\_ul

filtr\_niz, donja\_rubna\_vrijednost, gornja\_rubna\_vrijednost = stats.sigmaclip(Niz\_podataka, 3, 3) # sigmaclip kao rezultat vraća tri vrijednosti

''' Zaglađivanje (Smoothing) podataka Sawitzky-Golay'''

from scipy.signal import savgol\_filter

smooth\_niz = savgol\_filter(T\_ul, 21, 5) #argumenti: niz za zaglađivanje, window size , polynomial order

# Vizualni prikaz zaglađivanja

# plt.plot(smooth\_niz,'-k' )

plt.plot(T\_ul, '--r')

plt.show()

''' Analiza i odabir utjecajnih varijabli '''

# scipy

from scipy.stats import pearsonr, spearmanr

Corr, \_ = pearsonr(T\_ul,T\_sred)

print('Korelacija ulatne temperature i temperature dna: %.3f'%Corr)

CorrS, \_ = spearmanr(T\_ul,T\_sred)

print('Korelacija ulatne temperature i temperature dna: %.3f'%CorrS)

# Vizualni prikaz korelacije T\_ul i T\_sred

#plt.scatter(T\_sred,T\_ul)

# Pandas

Corr\_all = Data.corr(method = 'pearson') # metode - pearson, spearman, kendall

Corr\_Tul\_LrBenz = Data[['Tul','Benzin\_LR']].corr(method = 'spearman')

# Vizualna analiza korelacija među varijablama

# import seaborn as sns

# sns.heatmap(Corr\_all)

''' Detrendiranje varijabli'''

from scipy.signal import detrend

T\_ul\_det = detrend(T\_ul, type = 'linear') ## type = 'constant' - mean remove opcija

# Vizualna analiza detrendiranja varijable

# plt.plot(T\_ul\_det,'-k' )

'''Odvajanje podataka na trening i test podskupove'''

from sklearn.model\_selection import train\_test\_split

X = T\_ul

Y = T\_sred

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X,Y, test\_size=0.2, shuffle= True, random\_state = 42)

'''Skaliranje i normaliziranje varijabli'''

from sklearn.preprocessing import MinMaxScaler, StandardScaler

T\_ul = np.reshape(T\_ul,(-1,1)) # za neke funkcije potrebno konvertirati u podoban oblik

scaler =StandardScaler()

Scal\_T\_ul = scaler.fit\_transform(T\_ul)

MMscaler = MinMaxScaler()

MMScal\_T\_ul = MMscaler.fit\_transform(T\_ul)

# Vizualna analiza skalirane varijable

#plt.plot(Scal\_T\_ul)

'''Resemliranje se koristi za problem nebalansiranih podataka.

   Resempliranje je potrebno napraviti prije podjele na test i trening te valiadcijski set'''

from sklearn.utils import resample

T\_ul\_resemp = resample(T\_ul, n\_samples = 500, replace = True)

# Vizualna analiza resempliranja varijable

# plt.plot(T\_ul\_resemp)

# plt.plot(T\_ul)

# **Primjer razvoja linearnog modela i vrednovanje rezultata**

Data2 = pd.read\_excel('Primjer\_1\_nepročićeni\_podaci.xlsx', header = 0)

# provjera ima li NaN vrijednosti u Data1 podacima

print(Data2.isna().sum())

# Imputacija nedostajućih podataka - Ako ih ima

Data2 = Data2.interpolate(method='spline', order = 2)

# Odvajanje praćene varijable (Target variable)

Y = Data2['Benzin\_LR']

X = Data2.drop('Benzin\_LR', axis = 1)

# Koverzija podataka u nizove

Y.values

X.values

# Odvajanje podataka na trening i test podskupove

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X,Y, test\_size=0.2, shuffle= True, random\_state = 42)

# Linearna Regresija

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import mean\_squared\_error, mean\_absolute\_error, r2\_score

LRmodel = LinearRegression()

LRmodel.fit(X\_train, y\_train)

y\_pred = LRmodel.predict(X\_test)

MSE = mean\_squared\_error(y\_test,y\_pred)

RMSE = np.sqrt(mean\_squared\_error(y\_test,y\_pred))

MAE = mean\_absolute\_error(y\_test,y\_pred)

R2 = r2\_score(y\_test,y\_pred)

print('\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n')

print('Rezultati - Linearna Regresija')

print('MSE: ', MSE)

print('RMSE: ', RMSE)

print('MAE: ', MAE)

print('R2: ', R2)

print('\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_')

# Stablo odlučivanja / Šuma stabala odlučivanja (Decision Tree / Random Forest)

from sklearn.tree import DecisionTreeRegressor

DTRmodel = DecisionTreeRegressor()

DTRmodel.fit(X\_train, y\_train)

y\_pred = DTRmodel.predict(X\_test)

MSE = mean\_squared\_error(y\_test,y\_pred)

RMSE = np.sqrt(mean\_squared\_error(y\_test,y\_pred))

MAE = mean\_absolute\_error(y\_test,y\_pred)

R2 = r2\_score(y\_test,y\_pred)

print('\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n')

print('Rezultati - Stablo odlučivanja')

print('MSE: ', MSE)

print('RMSE: ', RMSE)

print('MAE: ', MAE)

print('R2: ', R2)

print('\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_')

from sklearn.ensemble import RandomForestRegressor

RFRmodel = RandomForestRegressor()

RFRmodel.fit(X\_train, y\_train)

y\_pred = RFRmodel.predict(X\_test)

MSE = mean\_squared\_error(y\_test,y\_pred)

RMSE = np.sqrt(mean\_squared\_error(y\_test,y\_pred))

MAE = mean\_absolute\_error(y\_test,y\_pred)

R2 = r2\_score(y\_test,y\_pred)

print('\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n')

print('Rezultati - Šuma stabala')

print('MSE: ', MSE)

print('RMSE: ', RMSE)

print('MAE: ', MAE)

print('R2: ', R2)

print('\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_')