Fakultet kemijskog inženjerstva i tehnologije
Zavod za mjerenja i automatsko vođenje procesa
Metode umjetne inteligencije u kemijskom inženjerstvu

Statistička analiza i predobrada podataka
Razvoj viševeličinskog linearnog modela, modela stabla odlučivanja i modela slučajne šume.
Vrednovanje rezultata modela

Osnovni primjeri funkcija u Pythonu

Studeni, 2025.

Instalacija distribucije *Anaconda* i Python paketa

Distribucija **Anaconda** pruža cjeloviti i jednostavan pristup programiranju u programskom jeziku Python, posebno prilagođena početnicima. Distribuciju Anaconda možete besplatno preuzeti na službenoj stranici: https://www.anaconda.com/products/individual.

Pokretanjem **Anaconda Navigatora** nakon instalacije, otvara se grafičko sučelje (okruženje) u kojemu su prikazani instalirani programi i oni koji se mogu instalirati u okviru Anaconda Navigatora.
Među ponuđenim alatima nalazi se i interaktivno razvojno okruženje (IDE) **Spyder,** koje svojim jednostavnim i intuitivnim izgledom olakšava pisanje, uređivanje i pokretanje Python skripti.

Spyder dolazi s nizom već instaliranih Python paketa, poput **NumPy**, **SciPy** i drugih, dok će se ostatak potrebnih paketa za praćenje ovog seminara preuzeti na način prikazan u nastavku.

Za instalaciju potrebnih paketa u Anaconda Navigatoru pokrenut će se **CMD.exe Prompt** što je crni prozor poznatiji kao ***Command Prompt*** u kojem se upisuje 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

Ako radite s paketima koji zahtijevaju određene verzije drugih paketa, moguće je da ćete naići na situaciju u kojoj je potrebna nadogradnja postojećih paketa. Za ažuriranje paketa možete koristiti sljedeće naredbe:

pip install --upgrade ime_paketa

na primjer: pip install --upgrade numpy

Može se instalirati i određena verzija paketa (brojevi verzija dostupni su u dokumentaciji paketa na web stranicama):

pip install Ime paketa==verzija

na primjer: pip install numpy==1.20

Google Colab i Jupyter Notebook

Jupyter Notebook predstavlja interaktivno okruženje koje kombinira programski kôd, vizualizacije i tekst u jednom dokumentu. Pogodno je za eksplorativnu analizu podataka i edukaciju jer omogućava pokretanje kôda u ćelijama, što znači da možete pokretati dijelove kôda, analizirati rezultate i pisati bilješke unutar istog dokumenta.

Google Colab je besplatna platforma u oblaku koja omogućava izvršavanje Python skripti direktno u pregledniku. Posebno je korisna za rad s Jupyter Notebook datotekama, što omogućava kombinaciju kôda, teksta i vizualizacija u interaktivnom okruženju. S obzirom na to da ga pokreće Googleova infrastruktura, idealan je alat za situacije kada baratamo slabijim računalima ili moramo pristupiti kôdu online.

Za pokretanje Google Colab potreban je samo Google račun s kojim se prijavljuje na stranicu https://colab.research.google.com/. Za kreiranje novog notebooka potrebno je kliknuti na **File > New Notebook**

Nakon otvaranja notebooka, Python kôd se unosi u ćelije i izvršava pritiskom na tipku Play (ili kombinacijom tipki Shift + Enter). Podrška za većinu Python biblioteka dolazi unaprijed instalirana, dok se dodatni paketi mogu instalirati pomoću *pip* komande direktno u notebooku:

!pip install ime_paketa

na primjer: !pip install matplotlib

Deskriptivna statistika

```
# UČITAVANJE EXCEL DATOTEKE IZ RAČUNALA U GOOGLE COLAB (ako radite u Colabu)

from google.colab import files
import pandas as pd
uploaded = files.upload()


import pandas as pd
from pandas import Series
import numpy as np
import matplotlib.pyplot as plt
from scipy import stats


''' Unos podataka iz Excel datoteke 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.figure()
plt.boxplot(T_ul)
plt.title('Kutijasti dijagram: Tul')
plt.show()

plt.figure()
plt.hist(lr_benzin, bins = 20)
plt.title('Histogram: Benzin_LR')
plt.xlabel('Vrijednost')
plt.ylabel('Frekvencija')
plt.show()

plt.figure()
plt.plot(T_sred)
plt.title('Linijski graf: T_sred')
plt.xlabel('Indeks')
plt.ylabel('T_sred')
plt.show()
```

```
''' Konverzija podataka – priprema za funkcije za strojno učenje'''

# Pandas

T_ul = T_ul.values  #pretvara Pandas Series u NumPy array, što mnoge SciPy/sklearn funkcije
#očekuju
lr_benzin = lr_benzin.values

# Numpy

T_sred = np.array(T_sred)


'''  IMPUTACIJA – popunjavanje nedostajućih vrijednosti u podacima'''

m_ref_imp_lin = m_ref.interpolate(method='linear')  #
method:'polynomial', 'spline', 'barycentric', 'piecewise_polynomial',
'spline'

print(m_ref)
print(m_ref_imp_lin)

m_ref_imp_cubSp = m_ref.interpolate(method='spline', order = 2) # order
2 – kvadratni spline – glađa interpolacija


'''  SIGMACLIP - uklanjanje ekstremnih vrijednosti'''

Niz_podataka = T_ul
filtr_niz, donja_rubna_vrijednost, gornja_rubna_vrijednost =
stats.sigmaclip(Niz_podataka, 3, 3)

# Indeksi outliera
outlier_mask = (Niz_podataka < donja_rubna_vrijednost) | (Niz_podataka
> gornja_rubna_vrijednost)
outlier_idx  =  np.where(outlier_mask)[0]   #  indeksi  svih  elemenata  u
nizu koji su outlieri

# --- Graf ---
plt.figure
plt.plot(Niz_podataka, label="Originalni niz")

# Outlieri naglašeni crvenim kružićima
plt.scatter(outlier_idx, Niz_podataka[outlier_idx],
            color='red', label="Outlieri")
```

```
plt.title(" Sigmaclip outlieri")
plt.legend()
plt.show()


'''  ZAGLAĐIVANJE PODATAKA SAVITZKY-GOLAY FILTROM'''

from scipy.signal import savgol_filter
smooth_niz = savgol_filter(T_ul, 21, 5) # argumenti: niz za
#zaglađivanje, window size, polynomial order

# Grafički prikaz zaglađivanja
plt.plot(smooth_niz,'-k' )
plt.plot(T_ul, '--r')
plt.title ('Savitzky-Golay filtriranje')
plt.show()


''' KORELACIJSKA ANALIZA I ODABIR UTJECAJNIH VARIJABLI '''

# scipy

from scipy.stats import pearsonr, spearmanr
Corr, _ = pearsonr(T_ul,T_sred)  # pearsonr vraća 2 vrijednosti, uzmi samo
#korelaciju, odbaci p-vrijednost _placeholder za “nepotrebnu” vrijednost
print('Korelacija ulazne temperature i temperature dna: %.3f'%Corr)

CorrS, _ = spearmanr(T_ul,T_sred)
print('Korelacija ulazne temperature i temperature dna: %.3f'%CorrS)

# Grafički prikaz korelacije T_ul i T_sred
plt.scatter(T_sred,T_ul)
plt.show()

# Pandas korelacijska matrica

Corr_all = Data.corr(method = 'pearson') # method - pearson, spearman
print(Corr_all)

Corr_Tul_LrBenz = Data[['Tul','Benzin_LR']].corr(method = 'spearman')

# Vizualna analiza korelacija među varijablama (korelacijska matrica)

import seaborn as sns
sns.heatmap(Corr_all)
plt.show()
```

```
'''SKALIRANJE I NORMALIZIRANJE VARIJABLI'''

from sklearn.preprocessing import MinMaxScaler, StandardScaler

T_ul = np.reshape(T_ul,(-1,1)) # pretvara niz iz oblika (n,) u oblik (n, 1)
jer sklearn očekuje 2D matrice; (-1, 1) napravi onoliko redaka koliko treba i
jedan stupac

scaler = StandardScaler()    # standardizacija: sredina = 0, std = 1
Scal_T_ul = scaler.fit_transform(T_ul)

MMscaler = MinMaxScaler()  # preslikava na [0, 1] (default)
MMScal_T_ul = MMscaler.fit_transform(T_ul)

# Vizualna analiza skalirane varijable
plt.plot(Scal_T_ul)
plt.show()

plt.plot(MMScal_T_ul)
plt.show()


''' HAMPEL DETEKCIJA EKSTREMNIH VRIJEDNOSTI i Imputacija podataka'''

# informativno, jer je složenije

# definiranje funkcije hampel

def hampel(series, window_size=5, n_sigmas=3, imputation=False):
    """
    series      : ulazni niz (pandas Series)
    window_size : broj susjednih točaka s obje strane
    n_sigmas    : prag u broju MAD-ova (koliko puta Median Absolute Deviation treba
premašiti da bi točka bila outlier) ako je točka puno dalje od medijana od n_sigmas * MAD → outlier
    imputation  : False -> samo detektira outliere
                  True  -> zamjenjuje outlier lokalnim medijanom
    """
    series = series.astype(float)
    new_series = series.copy()
    k = 1.4826  # konstanta za pretvaranje MAD u procjenu stand. devij.
    N = len(series)  # broj točaka u signalu

    outlier_idx = []     # sprema indekse svih detektiranih outliera

    for i in range(N):   # petlja – prolazak kroz svaku točku signala
        start = max(i - window_size, 0)
        end = min(i + window_size + 1, N)
```

```
        window = series.iloc[start:end]

        median = window.median()
        mad = k * np.median(np.abs(window - median)) # MAD = medijan |točka – medijan|

        if mad == 0:
            continue    # Preskače se taj korak.

        if np.abs(series.iloc[i] - median) > n_sigmas * mad: # Ako je razlika od medijana > od (3 × MAD) → outlier

            outlier_idx.append(series.index[i])    # dodajemo indeks u listu outliera
            if imputation:
                new_series.iloc[i] = median

    if imputation:          # Ako imputation=False: → vraća listu indeksa gdje su outlieri
        return new_series
    else:
        return outlier_idx        # Ako imputation=True: → vraća novi "popravljen" signal

A = Data['Tul']

#  Detekcija outliera (pozivamo funkciju Hampel da detektira outliere)
outliers_idx = hampel(A, window_size=5, n_sigmas=3, imputation=False)
print("Indeksi outliera: ", outliers_idx)

# Imputacija (zamjena outliera rolling medianom / lokalnim medijanom)
A_imputed = hampel(A, window_size=5, n_sigmas=3, imputation=True)

# Grafički prikaz uklanjanja outliera (po želji)
A.plot(style="k-", label="original")
A_imputed.plot(style="g-", label="Hampel filtrirano")
plt.legend()
plt.show()
```

Primjer razvoja linearnog viševeličinskog modela, modela stabla odlučivanja i modela šuma stabala odlučivanja. Vrednovanje dobivenih rezultata

```
Data2 = pd.read_excel('Primjer_1.xlsx', header = 0)

# provjera ima li NaN vrijednosti u Data2 podacima
print(Data2.isna().sum())

# Imputacija nedostajućih podataka - Ako ih ima
Data2 = Data2.interpolate(method='spline', order = 2)

# Odvajanje izlazne varijable modela (Target variable)

Y = Data2['Benzin_LR']
X = Data2.drop('Benzin_LR', axis = 1)

# Konverzija podataka u nizove
Y = Y.values
X = X.values


# Odvajanje podataka na skup za treniranje i test

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)


# Model viševeličinske linearne regresije

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 (engl. Decision Tree)

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('_____________________________')


# Slučajna šuma (engl. Random Forest)

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('_____________________________')
```