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

Prosinac, 2024.

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 preuzmite 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 pisanjem 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 sljedeći način:

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

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

Google Colab i Jupyter Notebook

Jupyter Notebook prestavlja interaktivno okruženje koje kombinira kod, vizualizacije i tekst u jednom dokumentu. Pogodno je za eksplorativnu analizu podataka i edukaciju jer omogućava pokretanje koda u ćelijama, što znači da možete pokretati dijelove koda, 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 koda, 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 kod 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

```
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 (outlier-a 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 (engl.Smoothing) podataka Sawitzky-Golay filtrom'''

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' )

'''Podjela 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 viševeličinskog modela, modela stabla odlučivanja i modela šuma stabala odlučivanja. Vrednovanje dobivenih 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 izlazne varijable (modela) (Target variable)

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

# Konverzija 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)

# 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 / Slučajna šuma (engl. 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('_____________________________')
```