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

from google.colab import files

print('Učitaj datoteku: Primjer_1.xlsx')
uploaded = files.upload()

''' 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']
m_ref = Data['m_refluks']
lr_benzin = Data['Benzin_LR']

''' 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


'''  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'''

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()
```