```
#Assignment 1

#Step 1

#Relationship between solar radio flux and sunspot number
#Team 12
#Yaroslav Savotin, Elizaveta Pestova, Selamawit Asfaw, Skoltech, 2023

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import math
%matplotlib inline
#import essential libraries

#Step 2

df = pd.read_csv('D:\Skoltech\Term 1B Courses\Experimental Data Processing\Assignment 1\data_group4.csv')
#create dataframe (df) by reading dataset file in csv format
df
#display dataframe to be sure everything is all right
```

	year	month	monthly solar radio flux at 10.7 cm	monthly sunspot number
0	1964	4	72.6	12.9
1	1964	5	69.5	14.3
2	1964	6	69.0	13.5
3	1964	7	67.0	4.8
4	1964	8	69.3	13.8
...	...	...	...	...
394	1997	2	73.8	11.0
395	1997	3	73.5	12.1
396	1997	4	74.5	23.0
397	1997	5	74.6	25.4
398	1997	6	71.7	20.8

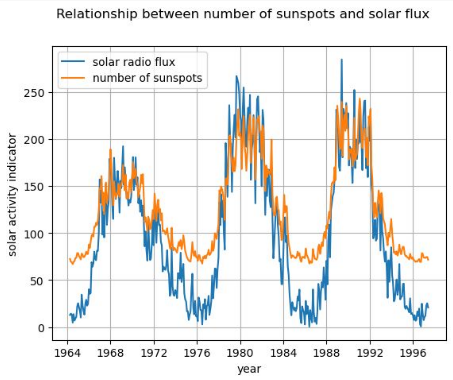
399 rows × 4 columns

```
df['data'] = pd.to_datetime((df['year'].astype ( str ) + df['month'].astype ( str )), format="%Y%m")
#add new column to dataframe names'data' which unites year-month columns for easy work with plots
df
#display dataframe to be sure everything is all right
```

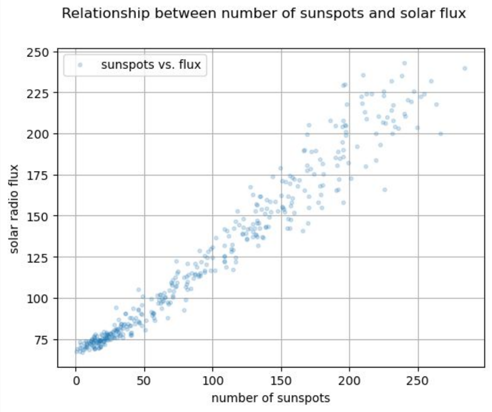
	year	month	monthly solar radio flux at 10.7 cm	monthly sunspot number	data
0	1964	4	72.6	12.9	1964-04-01
1	1964	5	69.5	14.3	1964-05-01
2	1964	6	69.0	13.5	1964-06-01
3	1964	7	67.0	4.8	1964-07-01
4	1964	8	69.3	13.8	1964-08-01
...	...	...	...	...	...
394	1997	2	73.8	11.0	1997-02-01
395	1997	3	73.5	12.1	1997-03-01
396	1997	4	74.5	23.0	1997-04-01
397	1997	5	74.6	25.4	1997-05-01
398	1997	6	71.7	20.8	1997-06-01

399 rows × 5 columns

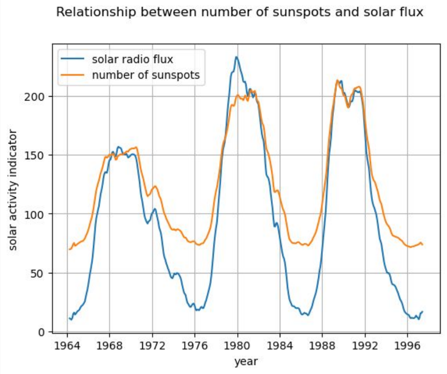
```
#Step 3 - create the plot of every dataset for visible comparison

y1 = df["monthly sunspot number"]
y2 = df["monthly solar radio flux at 10.7 cm"]
#chose two dependent variables from dataframe to be displayed on y-axis
x = df["data"]
#chose independent variable from dataframe to be displayed on x-axis
plt.plot(x,y1,label = 'solar radio flux')
plt.plot(x,y2,label = 'number of sunspots')
#create one plot for two dependent variables to be displaud simultaneously
plt.xlabel('year')
plt.ylabel('solar activity indicator')
#sign axis of the plot
plt.suptitle('Relationship between number of sunspots and solar flux')
plt.legend()
plt.grid(True)
#name the plot and legends for each variable, add the grid for more comfortable work
```



```
#Conclusion: relationship between number of sunspots and solar flux is visible as the dots on plot have a tendency to form some kind of linear dependency

#Step 4 - create the scatter plot for clear observation of relationship

sunspots=df['monthly sunspot number']
flux=df['monthly solar radio flux at 10.7 cm']
#rename two columns in dataframe for easier work
plt.plot(sunspots, flux , 'o', alpha=0.2, markersize=3, label = 'sunspots vs. flux')
#create and adjust the scatter plot for sunspots on x axis and flux in y axis, prepare the legend
plt.legend()
plt.xlabel('number of sunspots')
plt.ylabel('solar radio flux')
plt.suptitle('Relationship between number of sunspots and solar flux')
plt.grid(True)
#create the legend, sign axis  and the plot itself, turn on the grid
```

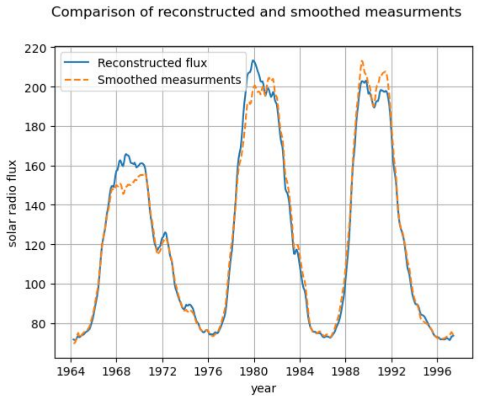


```
#Conclusion: relationship between number of sunspots and solar flux is visible as the dots on plot have a tendency to form some kind of linear dependency

#Step 5

#smooth the data for number of sunspots
R = df['monthly sunspot number'].values[:]
#rebuild our data frame into form for numpy
numrows = len(sunspots)
#create new variable 'numrows' wich is equal to length of 'sunspots' data frame
R1 = [[0] for _ in range(numrows)]
#this cycle must be run 398 times
for i in range(0,399):
    #create the borders of cycle
    if i in range(0,6):
        R1[i] = (R[i] + R[i+5] + R[i+4] + R[i+3] + R[i+2] + R[i+1])/6
        #create suborders and conditions for them - here we counting the mean for first six numbers in dataset
    if i in range(392,399):
        R1[i] = (R[i-5] + R[i-4] + R[i-3] + R[i-2] + R[i-1] + R[i])/6
        #subborders and formula for 6 last members of sunspots data frame
    if i in range(6,392):
        R1[i] = (R[i-6])/24 + (R[i-5] + R[i-4] + R[i-3] + R[i-2] + R[i-1] + R[i] + R[i+5] + R[i+4] + R[i+3] + R[i+2] + R[i+1])/12 + (R[i+6])/24
#calculate all members of smoothed data frame

#monthly solar radio flux at 10.7 cm
R = df['monthly solar radio flux at 10.7 cm'].values[:]
#rebuild our dataframe into form for numpy
numrows = len(flux)
#create new variable 'numrows' wich is equal to length of 'flux' data frame
R2 = [[0] for _ in range(numrows)]
for i in range(0,399):
#create the borders of cycle
    if i in range(0,6):
        R2[i] = (R[i] + R[i+5] + R[i+4] + R[i+3] + R[i+2] + R[i+1])/6
        #create suborders and conditions for them - here we counting the mean for first six numbers in dataset
    if i in range(392,399):
        R2[i] = (R[i-5] + R[i-4] + R[i-3] + R[i-2] + R[i-1] + R[i])/6
        #subborders and formula for 6 last members of sunspots dataframe
    if i in range(6,392):
        R2[i] = (R[i-6])/24 + (R[i-5] + R[i-4] + R[i-3] + R[i-2] + R[i-1] + R[i] + R[i+5] + R[i+4] + R[i+3] + R[i+2] + R[i+1])/12 + (R[i+6])/24
#calculate all members of smoothed data frame

#plot the results of smoothing our data with matplotlib
y1 = R1
y2 = R2
x = df['data'].values[:]
plt.plot(x,y1,label = 'solar radio flux')
plt.plot(x,y2,label = 'number of sunspots')
plt.xlabel('year')
plt.ylabel('solar activity indicator')
plt.suptitle('Relationship between number of sunspots and solar flux')
plt.legend()
plt.grid(True)
#plot smoothed data on the same plot to see how could we keep relationship between postprocessed datasets
```



```
#Conclusion: smoothed datasets are easier to analyze and they look nice
```

```
#Step 6 - begining of construction of multi-dimensional linear regression

F = np.array(df['monthly solar radio flux at 10.7 cm'].values[:])
#create vector of dependent variables, regressand, solar radio flux at 10.7 cm with numpy
R = np.ones((399, 4))
#create a sample matrix with requared sizes from ones for future work

#Step 7 - we alredy have  vector of dependent variables and going to create matrix of independent variables

v = R1
for i in range (0,399): #Matrix of independent variables, regressors,
    R[i,1] = v[i]
    R[i,2] = v[i]**2
    R[i,3] = v[i]**3

#build matrix of independent variables by replacing columns in sample matrix

#Step 8 - determine vector of coefficients by LSM

Beta = np.matmul(np.linalg.inv(np.matmul(np.transpose(R),R)),np.matmul(np.transpose(R),F))

print(Beta)

#calculate and display Beta - vector of coefficients by LSM
[ 6.82637784e+01  2.60862668e-01  3.86118378e-03 -9.90126915e-06]

#Step 9 - actual construction of multi-dimensional linear regression

F = np.ones((399,1))
for i in range (0,399):

    F[i] = Beta[0] + Beta[1]*R[i,1] + Beta[2]*R[i,2] + Beta[3]*R[i,3]
#border conditions and calculation of predicted numbers of solar flux based on sunspots measurements

#Step 10 - determine the variance of estimation error of solar radio flux
N = len(F)
error = 0
for i in range(0,399):
    error += (F[i] - R2[i])**2
error = error/(N-1)
#following the formula reproduce it in Python using predicted data set and real measurments after smoothing procedure
sigma = math.sqrt(error)
print(error)
print(sigma)
#display the sigma square and sigma
[28.42855268]
5.331843272083315

y1 = F
y2 = R2
x = df["data"]
plt.plot(x,y1,label = 'Reconstructed flux')
plt.plot(x,y2,label='Smoothed measurments', linestyle='--')
plt.xlabel('year')
plt.ylabel('solar radio flux')
plt.suptitle('Comparison of reconstructed and smoothed measurments')
plt.legend()
plt.grid(True)
#create the plot which shows how predicted data corresponds to real measuarments - looks pretty similar
```



```
#Conclusion: this plot clearly shows us how small is the error of reconstruction as the lines almost identical
#Step 11 - Learning log

#Assignment conclusion: This task is a great example of data analysis work because it provides  basic understanding
#of establishment of relationship between two datasets. Using plots as visualisation tool for future analysis
#and representation of results also provideds useful experience. Besides this task let us try one of simplification methods
#for rough data: smoothing of mean, we even try powerful method of reeconstruction of dataset: Construction of multi-dimensional
#linear regression.

#Personal learning log.
#   Yaroslav: being a newbie in programming I consider most of the work done as new experience.
#Briefly running through basics of Python I got into building plots with help of pandas and matplotlib libraries.
#Then recreation of statistics equations took my time  and at the end of work I tried operating with arrays in Python.
#I can say that I almost performed to complete this task by myself, however most of code in this report is work
#of my teammate, Lisa, who is skillful in coding and her code is much more pleasant to see. But let her tell it by herself.

#   Lisa: During the task, I remembered how to create plots and work with data frames in python..
#During the data analysis, I found a relationship between number of sunspots and solar radio flux with the help of the data
#presented in the plot. I learned how to make calculations of multi-dimensional linear regression and data smoothing.
```

```
#   Selamawit Asfaw: During the lecture, I learned about the multi-dimensional linear regression technique
#for processing data and identifying the relationship between different parameters. Additionally, I discovered that there
#is a direct correlation between the number of sunspots and the solar radio flux at 10.7 cm (2800 MHz), which are both used
#to describe solar activity. We created a multi-dimensional linear
#regression model with the solar radio flux at 10.7 cm as the regressand and the sunspot number
#as the regressor. We calculated the relationship between these parameters and determined the
#variance of the estimation error for this method. Additionally, we gained experience using
#the least-squares approach to determine the regression coefficients.
```