FACULTATEA DE CIBERNETICA,STATISTICA SI INFORMATICA ECONOMICA, BUCURESTI

**PROIECT LA ANALIZA DATELOR**

RICHTER CRISTINA

GRUPA 1120, AN 3 ID

INFORMATICA ECONOMICA

PROF. COORDONATOR: FURTUNA TITUS FELIX

CUPRINS

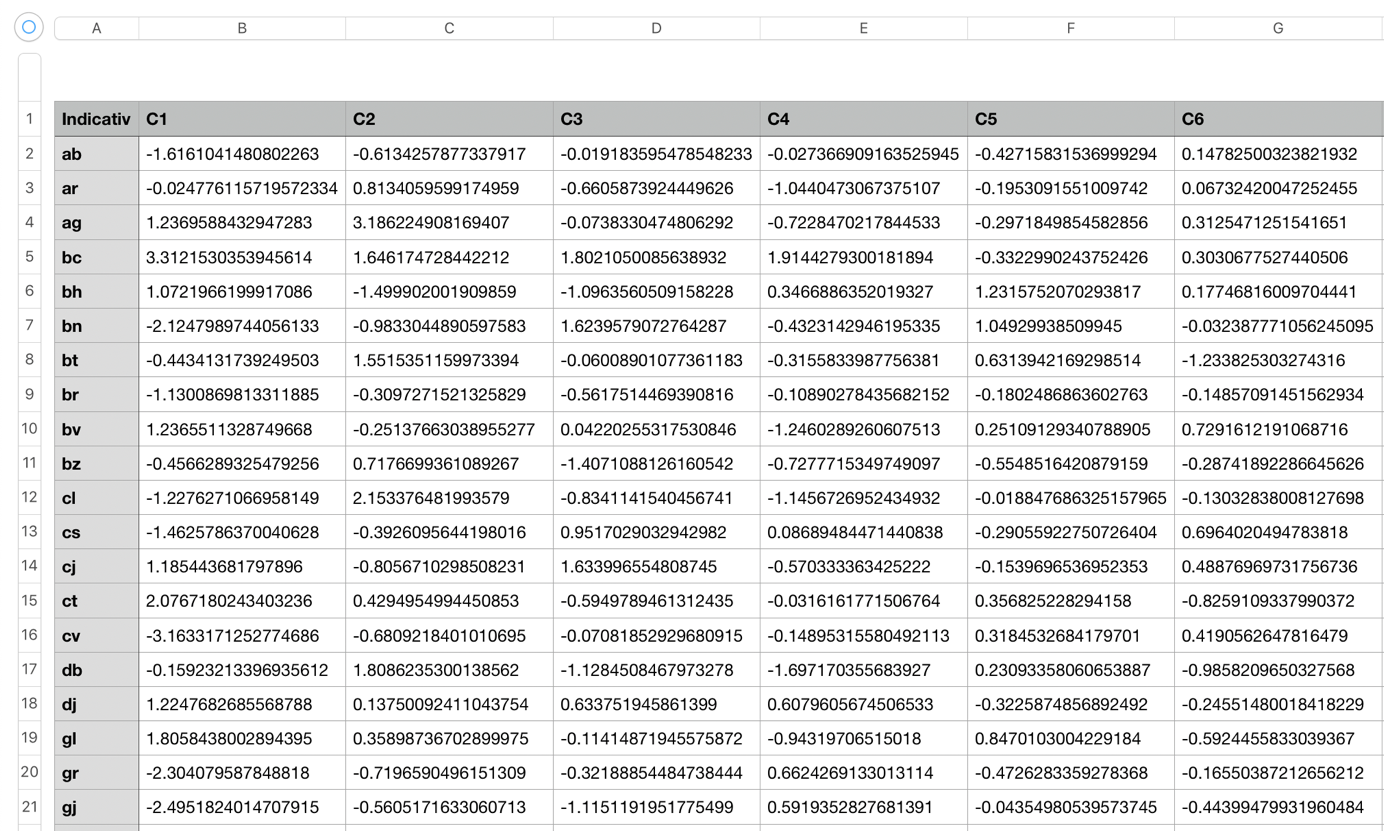
* Prezentare date
* Analiza componentelor principale
* Bibliografie
* Concluzii

Preluare date:

Datele care au fost folosite in elaborarea acestui proiect au fost preluate de pe site-ul Tempo Online: [*http://statistici.insse.ro:8077/tempo-online/#/pages/tables/insse-table*](http://statistici.insse.ro:8077/tempo-online/#/pages/tables/insse-table)

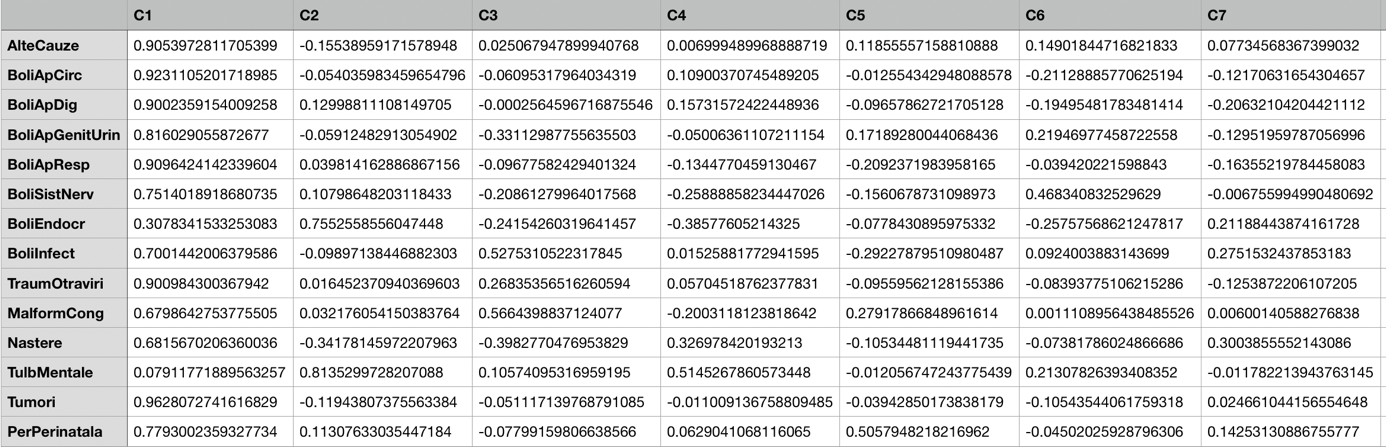
* A. STATISTICA SOCIALA
  + A.2 MISCAREA NATURALA A POPULATIEI
    - 2. MORTALITATE
      * POP2016D – Decedați pe județe si localități

Au fost utilizați 14 indicatori, si anume, cauzele deceselor in pe perioada anului 2017 pe întreg teritoriul României.



Coloanele C1 pana la C14 reprezintă indicatorii (cauzele deceselor, grupate pe județe)

Cauzele deceselor in ordine alfabetica: Alte cauze, Boli ale aparatului circulator, Boli ale aparatului digestiv, Boli ale aparatului genital si urinar, Boli ale aparatului respirator, Boli ale sistemului nervos, Boli endocrinologice, Boli infecțioase, Traume/Otrăviri, Malformații congenitale, Naștere, Tulburări mentale, Tumori, Boli in perioada perinatala.



2. ANALIZA COMPONENTELOR PRINCIPALE

Reprezintă o analiza multidimensionala care are ca scop sintetizarea informației conținuta in variabilele inițiale prin intermediul unor noi variabile in număr mai mic ,care sunt corelate doua cate doua, fără ca aceasta forma sa conțină redundante informaționale. Aceste componente exprima atribute noi si sunt construite in așa fel încât sa fie necorelate intre ele, fiecare fiind o combinație liniara de variabile originale.

Astfel, are loc reducerea dimensionalității spațiului cauzal inițiala si înlăturarea redundantei informaționale. Caracteristicile rezultate in urma transformării caracteristicilor inițiale se nume component principale si sunt exprimate sub forma combinațiilor liniare de variabile originale ,ele ,de asemenea sunt caracterizate de o variabilitate maxima.

Scopul acestei analize este ca pentru matricea noastră de date sa se identifice noi variabile care sa exprime sintetic vechile variabile astfel încât cantitatea totala de informative sa se piardă in mod controlat.

Mortalitatea

In proiectul de fata se încearcă identificarea unei posibile legături intre tipurile de deces si diferitele regiuni ale României.

Tehnologia folosita este:

* PyCharm ca si editor
* Python ca si limbaj de programare.

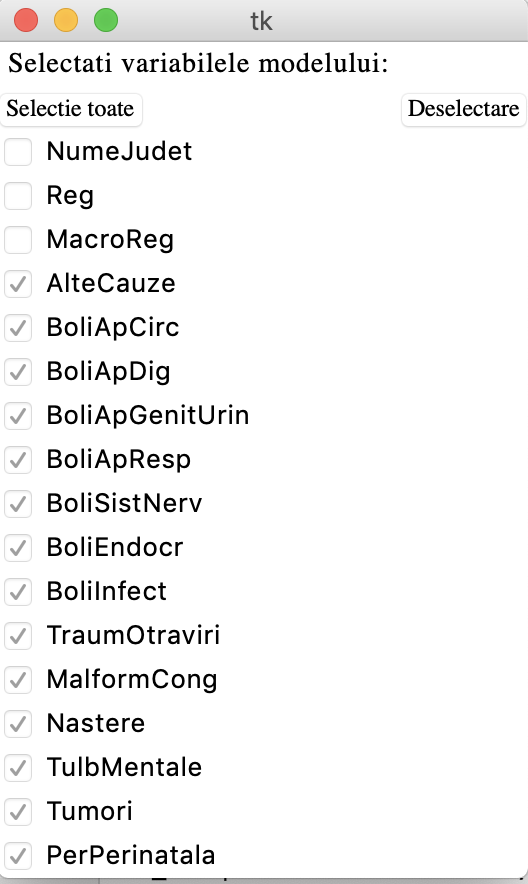
Mortalitatea măsoară totalitatea deceselor în cadrul unei populaţii pe parcursul unei perioade definite de timp. Variaţia ratelor de mortalitate, în mare măsură, determină nivelul sporului natural şi al speranţei de viaţă. La rândul ei, mortalitatea este indicatorul cel mai sensibil influenţat de factori socio-economici şi biologici (mediul ambiant, stilul de viaţă), precum și de serviciile de sănătate.

În anul 2017, numărul deceselor a fost de 247.086, ceea ce corespunde unei rate brute de mortalitate de 11,7‰ locuitori.

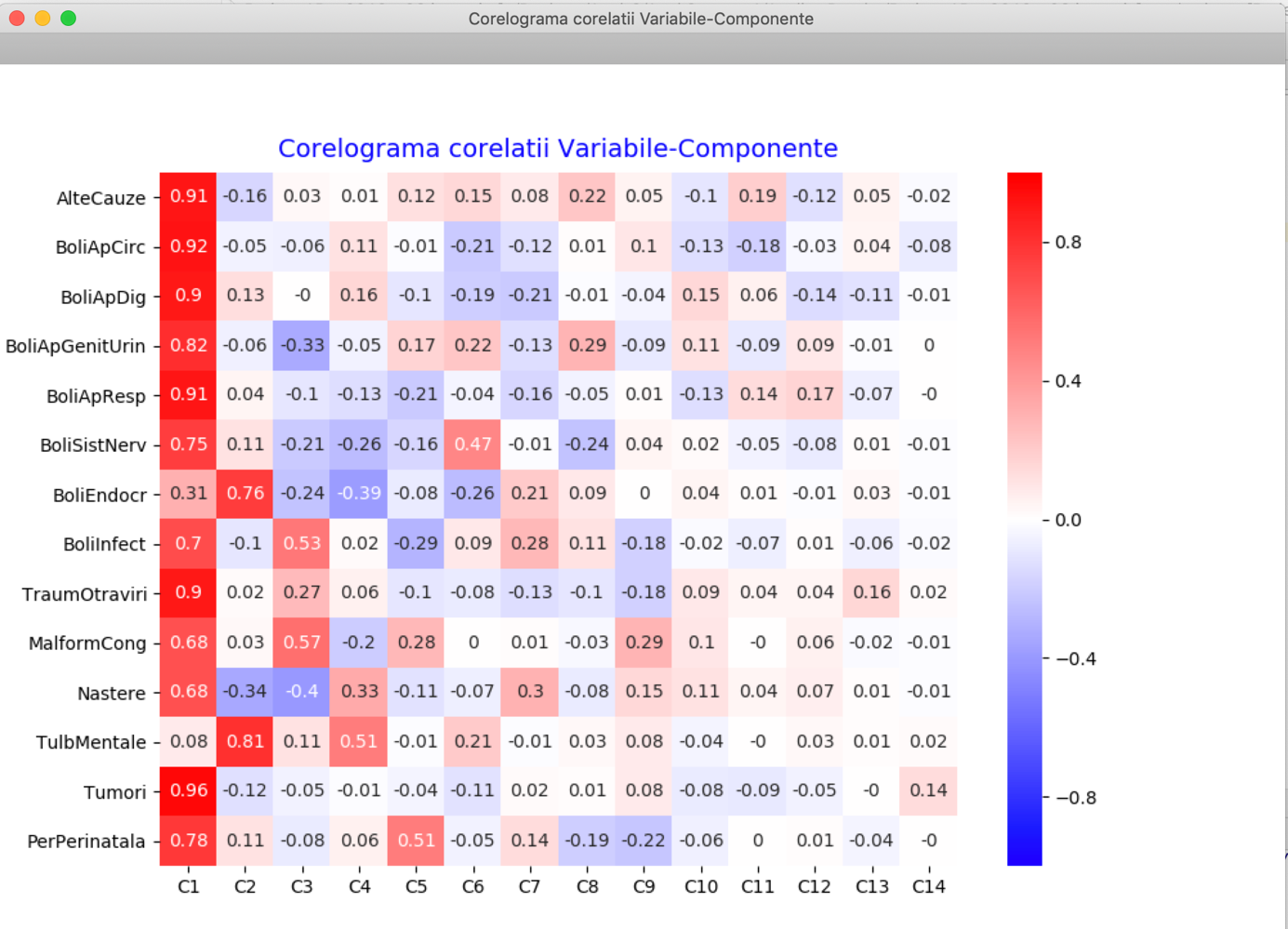
Valori mai mari ale ratei brute de mortalitate în 2017 se constată în judeţele

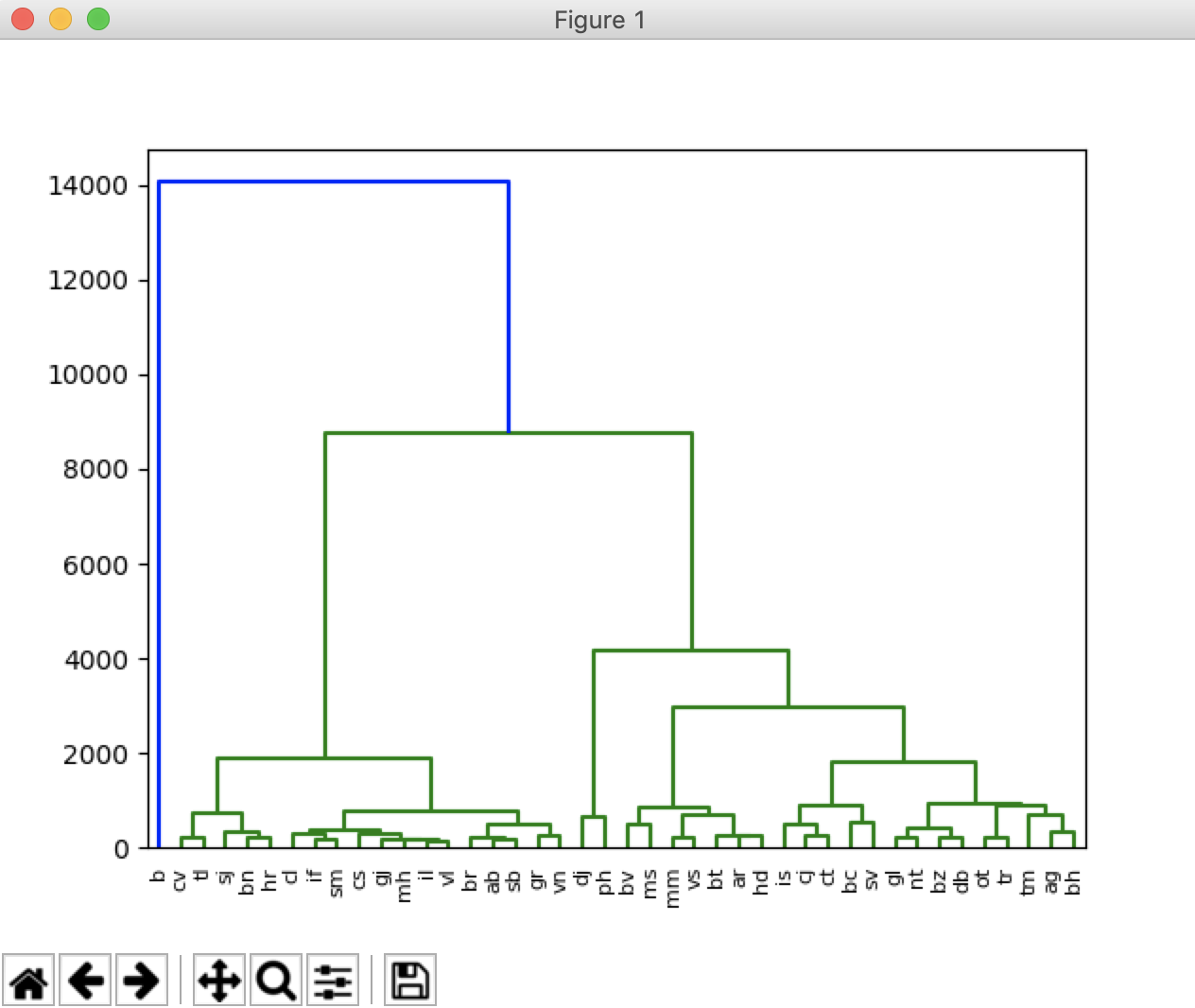
|  |  |
| --- | --- |
| **Municipiul Bucuresti** | 22142 |
| **Prahova** | 10447 |
| **Dolj** | 9403 |
| **Iasi** | 8779 |
| **Bacau** | 8668 |
| **Constanta** | 8115 |

Pentru a putea face selectia datelor dand „Run” pe main.py apare un pop-up, care contine 2 butoane (Selectie toate si Deselectare). Pentru a face selectia datelor, sunt selectate campurile care contin valori numerice.



Odată alese doar câmpurile cu valori numerice, se închide popup-ul si se generează „Corelograma corelații Variabile-Componente.





Main.py

# Import biblioteci si module externe

import Functii.functii as functii

import pandas as pd

import numpy as np

import Gui.gui as gui

import Grafice.grafice as grafice

import scipy.cluster.hierarchy as clusth

import matplotlib.pyplot as plt

# Citire date

t = pd.read\_csv('Teritorial/Teritorial2016/Mortalitate.csv', index\_col=0)

# print(t)

# preluare nume variabile

nume\_variabile = np.array(t.columns)

# selectie variabile model

variabile\_model = gui.Check(nume\_variabile, "Selectati variabilele modelului:")

x = t[variabile\_model].values

# print(x)

functii.inlocuire\_na(x)

R, alpha, a, Rxc, coloana = functii.acp(x)

nume\_componente = ['C' + str(i) for i in range(1, len(variabile\_model) + 1)] #ca sa apara C1, C2

rxc\_t = pd.DataFrame(data=Rxc, index=variabile\_model, columns=nume\_componente)

rxc\_t.to\_csv("Rxc.csv")

C\_t = pd.DataFrame(data=coloana, index=t.index, columns=nume\_componente)

C\_t.to\_csv("C.csv")

grafice.corelograma(rxc\_t, title="Corelograma corelatii Variabile-Componente")

grafice.t\_scatter(coloana[:, 0], coloana[:, 1], np.array(t.index), 'C1', 'C2', "Plot componente - axele 1x2")

grafice.show()

# cluster

functii.inlocuire\_na\_df(t)

z = clusth.linkage(t[variabile\_model].values,method='ward')

print(z)

clusth.dendrogram(z,labels=t.index)

plt.show()

Gui.py

import tkinter as tk

import tkinter.ttk as ttk

import tkinter.filedialog

import numpy as np

import matplotlib

# proiect facut pe un mac

# pentru ca imi dadea eroare am urmat pasii de aici: https://github.com/MTG/sms-tools/issues/36

from sys import platform as sys\_pf

if sys\_pf == 'darwin':

import matplotlib

matplotlib.use("TkAgg")

# END adaugat pentru ca am facut proiect pe mac

def ListBox(optiuni, mesaj="Selectati o optiune"):

selectii = []

def selectie(event):

k = mylist.curselection()

selectii.clear()

for i in k:

selectii.append(i)

def selectie\_toate():

mylist.selection\_set(0, len(optiuni))

def deselectare():

mylist.select\_clear(0, len(optiuni))

root = tk.Tk()

tk.Label(master=root, text=mesaj, font=('Times', '10')).grid(row=0, sticky=tk.W)

tk.Button(master=root, command=selectie\_toate, text="Selectie toate",

font=('Times', '10')).grid(row=1, column=0, sticky=tk.W)

tk.Button(master=root, command=deselectare, text="Deselectare",

font=('Times', '10')).grid(row=1, column=1, sticky=tk.W)

mylist = tk.Listbox(root, selectmode=tk.MULTIPLE, font=('Times', '12'))

mylist.grid(row=2, sticky=tk.W)

for optiune in optiuni:

mylist.insert(tk.END, optiune)

v\_optiuni = np.array(optiuni)

mylist.bind('<<ListboxSelect>>', selectie)

root.mainloop()

return v\_optiuni[selectii]

def Combo(optiuni, mesaj="Selectati o optiune"):

optiuni\_str = [str(o) for o in optiuni]

frame = tk.Tk()

frame.geometry("350x150")

frame.title("Selector Combobox")

ttk.Label(master=frame, text=mesaj, font=('Times', '14')).grid(row=1, sticky=tk.W)

variabila\_continut = tk.StringVar()

cb = ttk.Combobox(master=frame, textvariable=variabila\_continut,

values=optiuni\_str, font=('Times', '14'))

cb.grid(row=3, sticky=tk.W)

cb.current(0)

frame.mainloop()

return variabila\_continut.get()

def Check(optiuni, mesaj="Selectati una sau mai multe optiuni"):

def selectie\_toate():

for o\_chk in variabile\_continut:

o\_chk.set(1)

def deselectare():

for o\_chk in variabile\_continut:

o\_chk.set(0)

frame = tk.Tk()

tk.Label(master=frame, text=mesaj, font=('Times', '14')).grid(row=0, sticky=tk.W)

tk.Button(master=frame, command=selectie\_toate, text="Selectie toate", font=('Times', '12')).grid(row=1, column=0,sticky=tk.W)

tk.Button(master=frame, command=deselectare, text="Deselectare", font=('Times', '12')).grid(row=1, column=1, sticky=tk.W)

variabile\_continut = []

for i in range(len(optiuni)):

variabile\_continut.append(tk.IntVar())

chk\_b = tk.Checkbutton(master=frame, text=optiuni[i],

variable=variabile\_continut[i]).grid(row=i + 2, sticky=tk.W)

tk.mainloop()

selectii = []

for i in range(len(optiuni)):

if variabile\_continut[i].get() == 1:

selectii.append(optiuni[i])

return selectii

def FileDialog(extensie, mesaj='Fisier input '):

frame = tk.Tk()

frame.geometry("200x150+30+30")

nume = tk.StringVar()

def callback():

nume\_fisier = tk.filedialog.askopenfilename(parent=frame, initialdir=".", filetypes=[("Fisiere csv", extensie)])

nume.set(nume\_fisier)

tk.Button(text=mesaj, command=callback, font=('Times', '14')).pack(fill=tk.BOTH, padx=40, pady=40)

frame.mainloop()

return nume.get()

grafice.py

import seaborn as sb

import matplotlib.pyplot as plt

import numpy as np

def corelograma(t,title=None,valmin=-1,valmax=1):

f = plt.figure(title)

f1 = f.add\_subplot(1,1,1)

f1.set\_title(title,fontsize=14,color='b',verticalalignment='bottom')

sb.heatmap(np.round(t,2),cmap='bwr',vmin=valmin,vmax=valmax,annot=True)

def plot\_varianta(alpha, titlu='Plot varianta'):

n = len(alpha)

f = plt.figure(titlu,figsize=(10,7))

f1 = f.add\_subplot(1,1,1)

f1.set\_title(titlu,fontsize=10,color='b',verticalalignment='bottom')

f1.set\_xticks(np.arange(1,n+1))

f1.set\_xlabel('Componenta',fontsize=10,color='r',verticalalignment='top')

f1.set\_ylabel('Varianta',fontsize=10,color='r',verticalalignment='bottom')

f1.plot(np.arange(1, n + 1),alpha,'ro-')

f1.axhline(1,c='g')

j\_Kaiser = np.where(alpha < 1)[0][0]

eps = alpha[:n - 1] - alpha[1:]

d = eps[:n-2]-eps[1:]

j\_Cattel = np.where(d<0)[0][0]

f1.axhline(alpha[j\_Cattel + 1], c='m')

return j\_Cattel+2,j\_Kaiser

def t\_scatter(x,y,label=None,tx="",ty="",titlu='Scatterplot'):

f = plt.figure(titlu, figsize=(10, 7))

f1 = f.add\_subplot(1, 1, 1)

f1.set\_title(titlu,fontsize=10,color='b',verticalalignment='bottom')

f1.set\_xlabel(tx,fontsize=10,color='r',verticalalignment='top')

f1.set\_ylabel(ty,fontsize=10,color='r',verticalalignment='bottom')

f1.scatter(x=x,y=y,c='r')

if label is not None:

n = len(label)

for i in range(n):

f1.text(x[i],y[i],label[i])

def t\_scatter\_s(x,y,x1,y1,label=None,label1=None,tx="",ty="",titlu='Scatterplot set suplimentar'):

f = plt.figure(titlu, figsize=(10, 7))

f1 = f.add\_subplot(1, 1, 1)

f1.set\_title(titlu,fontsize=16,color='b',verticalalignment='bottom')

f1.set\_xlabel(tx,fontsize=12,color='r',verticalalignment='top')

f1.set\_ylabel(ty,fontsize=12,color='r',verticalalignment='bottom')

f1.scatter(x=x,y=y,c='r')

f1.scatter(x=x1,y=y1,c='b')

if label is not None:

n = len(label);p = len(label1)

for i in range(n):

f1.text(x[i],y[i],label[i],color='k')

for i in range(p):

f1.text(x1[i],y1[i],label1[i],color='k')

def cercul\_corelatiilor(R,k1,k2,titlu="Cercul corelatiilor"):

plt.figure(titlu,figsize=(6,6))

plt.title(titlu, fontsize=16, color='b',verticalalignment='bottom')

x =[v for v in np.arange(0,np.math.pi\*2,0.01)]

cosx = np.cos(x);sinx = np.sin(x)

plt.plot(cosx,sinx)

plt.axhline(0,color='g');plt.axvline(0,color='g')

plt.scatter(R.iloc[:,k1],R.iloc[:,k2],c='r')

plt.xlabel(R.columns[k1],fontsize=12,color='r',verticalalignment='top')

plt.ylabel(R.columns[k2],fontsize=12,color='r',verticalalignment='bottom')

for i in range(len(R)):

plt.text(R.iloc[i,k1],R.iloc[i,k2],R.index[i])

def show():

plt.show()

functii.py

import numpy as np

import pandas as pd

import pandas.api.types as pdt

def standardizare(X):

medii = np.mean(X, axis=0)

abaterestd = np.std(X, axis=0)

Xstd = (X - medii) / abaterestd

return Xstd

def inversare(t, y=None):

if type(t) is pd.DataFrame:

for c in t.columns:

minim = t[c].min()

maxim = t[c].max()

if abs(minim) > abs(maxim):

t[c] = -t[c]

if y is not None:

k = t.columns.get\_loc(c)

y[:, k] = -y[:, k]

else:

for i in range(np.shape(t)[1]):

minim = np.min(t[:, i])

maxim = np.max(t[:, i])

if np.abs(minim) > np.abs(maxim):

t[:, i] = -t[:, i]

def acp(X):

R = np.corrcoef(X, rowvar=False)

# calcul vector si valori proprii

valp, vecp = np.linalg.eig(R)

# sortare valori proprii si vectori proprii

k\_inv = [k for k in reversed(np.argsort(valp))]

alpha = valp[k\_inv]

a = vecp[:, k\_inv]

inversare(a)

# calcul corelatii factoriale

Rxc = a \* np.sqrt(alpha)

# calcul componente

# standardizare X

medii = np.mean(X, axis=0)

abaterestd = np.std(X, axis=0)

Xstd = (X - medii) / abaterestd

coloana = Xstd @ a

return R, alpha, a, Rxc, coloana

# Functie pentru inlocuirea valorilor lipsa

# prin medie/modul

def inlocuire\_na\_df(t):

for c in t.columns:

if pdt.is\_numeric\_dtype(t[c]):

if t[c].isna().any():

medie = t[c].mean()

t[c] = t[c].fillna(medie)

else:

if t[c].isna().any():

modul = t[c].mode()

t[c] = t[c].fillna(modul[0])

def inlocuire\_na(X):

medii = np.nanmean(X, axis=0)

k\_nan = np.where(np.isnan(X))

X[k\_nan] = medii[k\_nan[1]]