UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola d'Enginyeria de Barcelona Est

TREBALL FI DE GRAU

Grau en Enginyeria Biomèdica

TRACTAMENT DE SENYALS PER L'AJUT A LA DIAGNOSI DE L'APNEA DEL SON

Volum II

Annexos

Autor: Arnau Vizcaino Torras
Director: Jordi Solà Soler
Departament: Dept. ESAII
Co-Director: Gerard Escudero Bakx
Concovatòria: Octubre 2017

11. Annex A: Manual d'usuari

Pas0.py. Creació models entrenament i test per l'estudi 1

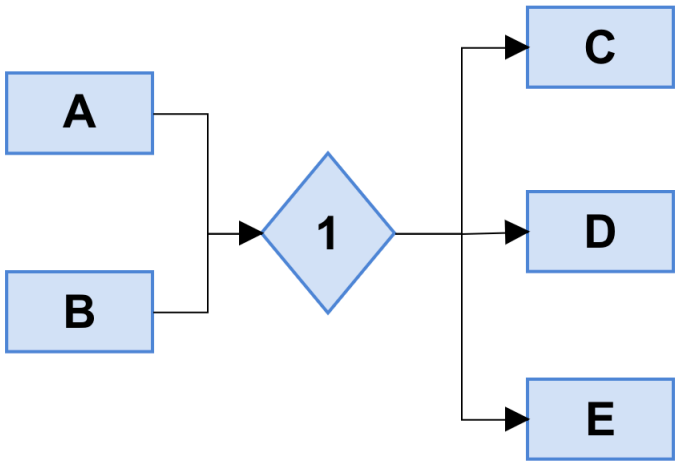

Figura 22. Esquema codi Python pas 0

A	Conjunt d'arxius on cada un conté els roncs d'un pacient
B	Conjunt de característiques clíniques de cada pacient
1	Procés d'homogeneïtzació dels dos tipus de característiques i creació de documents a avaluar
C	Document per realitzar l'entrenament
D	Document per realitzar el test
E	Llista de la repartició dels pacients en el document d'entrenament i el de test

Taula 53. Llegenda de l'esquema pas 0

Pas1.py. Creació del model de l'estudi 2

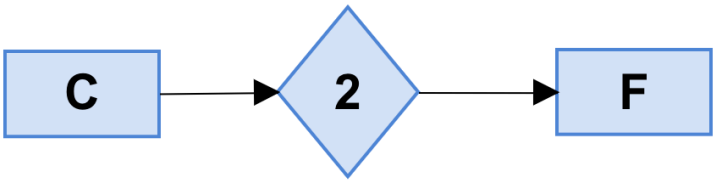

Figura 23. Esquema codi Python pas 1

2	Representació de la distribució dels roncs ens els diferents grups en l'anterior estudi aconseguint el valor "K"
F	Creació d'un arxius amb el model de distribució dels roncs segons el l'algorisme *K-means*

Taula 54. Llegenda de l'esquema pas 1

Pas2.py. Projecció del model de l'estudi 2 per crear un arxiu per avaluar-lo en l'estudi 3:

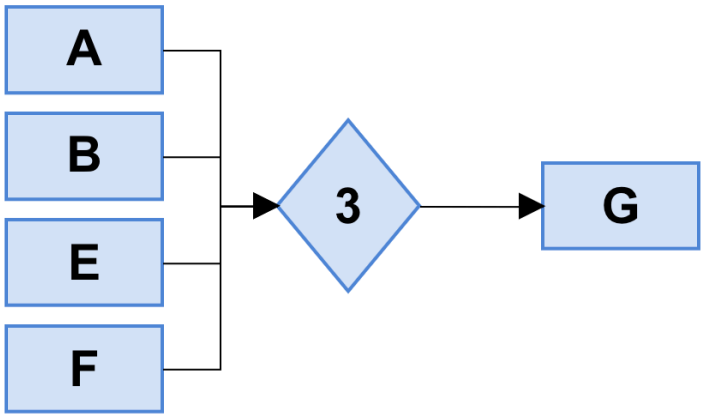

Figura 24. Esquema codi Python pas 2

3	Procés de distribució de dos models, un d'entrenament i un de test, amb el model recent creat, la llista dels pacients i la seva informació mèdica i dels roncs
G	Document amb el model creat per a poder realitzar l'estudi de classificació

Taula 55. Llegenda de l'esquema pas 2

Pas3.py. Implementació i obtenció de resultats de l'estudi 3:

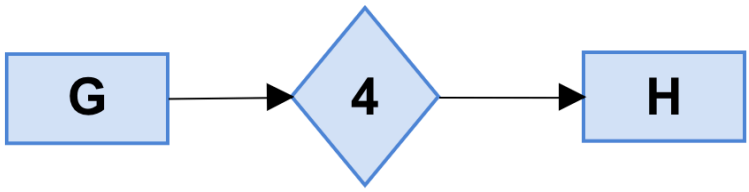

Figura 25. Esquema codi Python pas 3

4	Aplicació de diferents algorismes sobre el documents recent creat
H	Resultats de l'aplicació dels algorismes retornant un seguit de mesures de bondat

Taula 56. Llegenda de l'esquema pas 3

12. Annex B: Codis Programació

En aquest capítol s'adjunten els codis de programació de la iteració número 2, que és la que contempla la millor combinació, però només per el índex de severitat IAH, perquè els següents índexs s'implementen de la mateixa manera.

a) Pas0.py. Creació models entrenament i test per l'estudi 1:

```
from os import listdir
from os.path import isfile, join
from random import shuffle
from math import floor

####        per 5 classes
def crearArxiu(llista, nom):
      gl = []
      for file in llista:
            dt = []
            with open(mypath+"/"+file) as f:
                  for line in f:
                        dt.append(line.strip())

            # el·liminar llegenda
            del(dt[0])

            # construir matriu amb reals i sense 1a columna
            m = [[float(x) for x in s.split(' ')[5:-1]] for s in dt]

            # afegir la matriu a la global
            gl += m
      with open(nom, 'w') as f:
            for l in gl:
                  f.write(','.join([str(x) for x in l])+"\n")

# llista d'arxius
mypath = "caracteristiques"
onlyfiles = [f for f in listdir(mypath) if isfile(join(mypath, f)) and
f[:5] == "DEBUG"]

# dades pacients
dt = []
with open("DadesPacients.csv") as f:
      for l in f:
            dt.append(l.strip().split(','))

del(dt[0])

dp = {}
for temp in dt:
```

```
##      el 8 indica la columna de severitat que agafa per fer la
distribuccio
      dp[temp[0]] = [temp[8]]

for f in onlyfiles:
      k = f.split('_')[1][3:]
      dp[k].append(f)

dc = {}
for k in dp:
      dc.setdefault(dp[k][0], [])
      dc[dp[k][0]].append(dp[k][1])

trn = []
tst = []
for k in dc:
      shuffle(dc[k])
      ll = floor(len(dc[k])*0.67)
      trn += dc[k][:ll]
      tst += dc[k][ll:]

crearArxiu(trn, 'entrenament.weka.csv')
crearArxiu(trn, 'test.weka.csv')

# guardar la llista d'arxius amb l'ordre aleatòri
with open('llistaArxius.txt', 'w') as f:
      for l in trn+tst:
            f.write(l+"\n")

# generar l'arxiu parametres.py
with open('parametres.py', 'w') as f:
      f.write("# parametres\n")
      f.write("llindar = "+str(len(trn))+"\n")
```

b) Pas1.py. Creació del model de l'estudi 2:

```
import numpy as np
from sklearn.cluster import KMeans
from sklearn.cluster import Birch
from sklearn.cluster import SpectralClustering
from sklearn.cluster import DBSCAN
from sklearn.cluster import ward_tree
import pickle


# obrir arxiu
dt = []
with open('entrenament.weka.csv') as f:
      for line in f:
            dt.append(line.strip())
```

```
# construir matriu amb reals i sense 1a columna
X = [[float(x) for x in s.split(',')] for s in dt]


# Clustering KMEANS 8 CLSUTERS

modelClustering = KMeans(n_clusters=8).fit(X)

with open('model.kmeans8clusters.pickled', 'wb') as f:
    pickle.dump(modelClustering, f, pickle.HIGHEST_PROTOCOL)


# obrir arxiu
dt = []
with open('entrenament.weka.csv') as f:
      for line in f:
            dt.append(line.strip())


# construir matriu amb reals i sense 1a columna
X = [[float(x) for x in s.split(',')] for s in dt]


# Clustering KMEANS 7 CLSUTERS

modelClustering = KMeans(n_clusters=7).fit(X)

with open('model.kmeans7clusters.pickled', 'wb') as f:
    pickle.dump(modelClustering, f, pickle.HIGHEST_PROTOCOL)


# obrir arxiu
dt = []
with open('entrenament.weka.csv') as f:
      for line in f:
            dt.append(line.strip())


# construir matriu amb reals i sense 1a columna
X = [[float(x) for x in s.split(',')] for s in dt]


# Clustering KMEANS 9 CLSUTERS

modelClustering = KMeans(n_clusters=9).fit(X)

with open('model.kmeans9clusters.pickled', 'wb') as f:
    pickle.dump(modelClustering, f, pickle.HIGHEST_PROTOCOL)
```

c) Pas2.py. Projecció del model de l'estudi 2 per crear un arxiu per avaluar-lo en l'estudi 3:

```
import pickle
from sklearn.cluster import KMeans
from collections import Counter


############        SEGONS L'INDEX AHI


############        MODEL KMEANS AMB 8 CLSTERS 5 CLASS


##OBRIR UN MODEL O ALTRE
with open('model.kmeans8clusters.pickled','rb') as f:
    modelClustering = pickle.load(f)


# carregar llista de pacients
ordrePacients = []
with open('llistaArxius.txt') as f:
    for line in f:
        ordrePacients.append(line.strip())


############    5 CLASSES DE SEVERITAT
# carregar dades de pacients
temp = []
with open('DadesPacients.csv') as f:
    for line in f:
        temp.append(line.strip().split(','))
del(temp[0])
dadesPacients = {}
for el in temp:
    dadesPacients[el[0]] = el[1:]

# dades de classificació
dades = []
for pacient in ordrePacients:
    idPacient = pacient.split('_')[1][3:]
    dt = []
    with open("caracteristiques/"+pacient) as f:
        for line in f:
            dt.append(line.strip())

    # el·liminar llegenda
    del(dt[0])
```

```
    # construir matriu amb reals i sense 1a columna
    m = [[float(x) for x in s.split(' ')[5:-1]] for s in dt]

    preds = modelClustering.predict(m).tolist()
    d = dict(Counter(preds))
    temp = [idPacient] + dadesPacients[idPacient][:4]
##    EL NUMERO 8 ES EL NUMERO DE RONCS = NUM CLUSTERS
    for cl in range(8):
        if cl in d:
            temp.append(str(d[cl]))
        else:
            temp.append('0')
## la primera es per canviar AHI index apneas hipoapneas(4)
##            HI index dhipoapneas(5), AI index dapneas(6)

##  LA SEGONA ES PER LES CLASSES 5 CLASES (7), 4 CLASSES (8)....

    temp.append(dadesPacients[idPacient][4])
    temp.append(dadesPacients[idPacient][7])
    dades.append(temp)

#Estar atent, per cada classificador(KMEANS o BIRCH) un dades.class_regr
diferent!!!
with open('dades.class_regr8.5AHI.txt', 'w') as f:
    for l in dades:
        f.write(','.join(l)+'\n')



############        MODEL KMEANS AMB 8 CLSTERS 4 CLASS


##OBRIR UN MODEL O ALTRE
with open('model.kmeans8clusters.pickled','rb') as f:
    modelClustering = pickle.load(f)


# carregar llista de pacients
ordrePacients = []
with open('llistaArxius.txt') as f:
    for line in f:
        ordrePacients.append(line.strip())


############    4 CLASSES DE SEVERITAT
# carregar dades de pacients
temp = []
with open('DadesPacients.csv') as f:
    for line in f:
        temp.append(line.strip().split(','))
```

```
del(temp[0])
dadesPacients = {}
for el in temp:
    dadesPacients[el[0]] = el[1:]

# dades de classificació
dades = []
for pacient in ordrePacients:
    idPacient = pacient.split('_')[1][3:]
    dt = []
    with open("caracteristiques/"+pacient) as f:
        for line in f:
            dt.append(line.strip())

    # el·liminar llegenda
    del(dt[0])

    # construir matriu amb reals i sense 1a columna
    m = [[float(x) for x in s.split(' ')[5:-1]] for s in dt]

    preds = modelClustering.predict(m).tolist()
    d = dict(Counter(preds))
    temp = [idPacient] + dadesPacients[idPacient][:4]
    for cl in range(8):
        if cl in d:
            temp.append(str(d[cl]))
        else:
            temp.append('0')
## la primera es per canviar AHI index apneas hipoapneas(4)
##            HI index dhipoapneas(5), AI index dapneas(6)

##  LA SEGONA ES PER LES CLASSES 5 CLAAWA (7), 4 CLASSES (8)....

    temp.append(dadesPacients[idPacient][4])
    temp.append(dadesPacients[idPacient][8])
    dades.append(temp)

#Estar atent, per cada classificador(KMEANS o BIRCH) un dades.class_regr
diferent!!!
with open('dades.class_regr8.4AHI.txt', 'w') as f:
    for l in dades:
        f.write(','.join(l)+'\n')



############        MODEL KMEANS AMB 8 CLSTERS 3 CLASS


##OBRIR UN MODEL O ALTRE
with open('model.kmeans8clusters.pickled','rb') as f:
    modelClustering = pickle.load(f)
```

```
# carregar llista de pacients
ordrePacients = []
with open('llistaArxius.txt') as f:
    for line in f:
        ordrePacients.append(line.strip())


############    3 CLASSES DE SEVERITAT
# carregar dades de pacients
temp = []
with open('DadesPacients.csv') as f:
    for line in f:
        temp.append(line.strip().split(','))
del(temp[0])
dadesPacients = {}
for el in temp:
    dadesPacients[el[0]] = el[1:]

# dades de classificació
dades = []
for pacient in ordrePacients:
    idPacient = pacient.split('_')[1][3:]
    dt = []
    with open("caracteristiques/"+pacient) as f:
        for line in f:
            dt.append(line.strip())

    # el·liminar llegenda
    del(dt[0])

    # construir matriu amb reals i sense 1a columna
    m = [[float(x) for x in s.split(' ')[5:-1]] for s in dt]

    preds = modelClustering.predict(m).tolist()
    d = dict(Counter(preds))
    temp = [idPacient] + dadesPacients[idPacient][:4]
    for cl in range(8):
        if cl in d:
            temp.append(str(d[cl]))
        else:
            temp.append('0')
## la primera es per canviar AHI index apneas hipoapneas(4)
##            HI index dhipoapneas(5), AI index dapneas(6)

##  LA SEGONA ES PER LES CLASSES 5 CLAAWA (7), 4 CLASSES (8)....

    temp.append(dadesPacients[idPacient][4])
    temp.append(dadesPacients[idPacient][9])
    dades.append(temp)
```

```
#Estar atent, per cada classificador(KMEANS o BIRCH) un dades.class_regr
diferent!!!
with open('dades.class_regr8.3AHI.txt', 'w') as f:
    for l in dades:
        f.write(','.join(l)+'\n')



############        MODEL KMEANS AMB 8 CLSTERS 2 CLASS


##OBRIR UN MODEL O ALTRE
with open('model.kmeans8clusters.pickled','rb') as f:
    modelClustering = pickle.load(f)


# carregar llista de pacients
ordrePacients = []
with open('llistaArxius.txt') as f:
    for line in f:
        ordrePacients.append(line.strip())


############    2 CLASSES DE SEVERITAT
# carregar dades de pacients
temp = []
with open('DadesPacients.csv') as f:
    for line in f:
        temp.append(line.strip().split(','))
del(temp[0])
dadesPacients = {}
for el in temp:
    dadesPacients[el[0]] = el[1:]

# dades de classificació
dades = []
for pacient in ordrePacients:
    idPacient = pacient.split('_')[1][3:]
    dt = []
    with open("caracteristiques/"+pacient) as f:
        for line in f:
            dt.append(line.strip())

    # el·liminar llegenda
    del(dt[0])

    # construir matriu amb reals i sense 1a columna
    m = [[float(x) for x in s.split(' ')[5:-1]] for s in dt]

    preds = modelClustering.predict(m).tolist()
    d = dict(Counter(preds))
```

```
    temp = [idPacient] + dadesPacients[idPacient][:4]
##  NUMERO 8 ES EL NUM DE CLUSTERS
    for cl in range(8):
        if cl in d:
            temp.append(str(d[cl]))
        else:
            temp.append('0')
## la primera es per canviar AHI index apneas hipoapneas(4)
##            HI index dhipoapneas(5), AI index dapneas(6)

##  LA SEGONA ES PER LES CLASSES 5 CLAAWA (7), 4 CLASSES (8)....

    temp.append(dadesPacients[idPacient][4])
    temp.append(dadesPacients[idPacient][10])
    dades.append(temp)

#Estar atent, per cada classificador(KMEANS o BIRCH) un dades.class_regr
diferent!!!
with open('dades.class_regr8.2AHI.txt', 'w') as f:
    for l in dades:
        f.write(','.join(l)+'\n')



############        MODEL KMEANS AMB 9 CLSTERS 5 CLASS


##OBRIR UN MODEL O ALTRE
with open('model.kmeans9clusters.pickled','rb') as f:
    modelClustering = pickle.load(f)


# carregar llista de pacients
ordrePacients = []
with open('llistaArxius.txt') as f:
    for line in f:
        ordrePacients.append(line.strip())


############    5 CLASSES DE SEVERITAT
# carregar dades de pacients
temp = []
with open('DadesPacients.csv') as f:
    for line in f:
        temp.append(line.strip().split(','))
del(temp[0])
dadesPacients = {}
for el in temp:
    dadesPacients[el[0]] = el[1:]

# dades de classificació
```

```
dades = []
for pacient in ordrePacients:
    idPacient = pacient.split('_')[1][3:]
    dt = []
    with open("caracteristiques/"+pacient) as f:
        for line in f:
            dt.append(line.strip())

    # el·liminar llegenda
    del(dt[0])

    # construir matriu amb reals i sense 1a columna
    m = [[float(x) for x in s.split(' ')[5:-1]] for s in dt]

    preds = modelClustering.predict(m).tolist()
    d = dict(Counter(preds))
    temp = [idPacient] + dadesPacients[idPacient][:4]
##    EL NUMERO 8 ES EL NUMERO DE RONCS = NUM CLUSTERS
    for cl in range(9):
        if cl in d:
            temp.append(str(d[cl]))
        else:
            temp.append('0')
## la primera es per canviar AHI index apneas hipoapneas(4)
##            HI index dhipoapneas(5), AI index dapneas(6)

##  LA SEGONA ES PER LES CLASSES 5 CLAAWA (7), 4 CLASSES (8)....

    temp.append(dadesPacients[idPacient][4])
    temp.append(dadesPacients[idPacient][7])
    dades.append(temp)

#Estar atent, per cada classificador(KMEANS o BIRCH) un dades.class_regr
diferent!!!
with open('dades.class_regr9.5AHI.txt', 'w') as f:
    for l in dades:
        f.write(','.join(l)+'\n')



############        MODEL KMEANS AMB 9 CLSTERS 4 CLASS


##OBRIR UN MODEL O ALTRE
with open('model.kmeans9clusters.pickled','rb') as f:
    modelClustering = pickle.load(f)


# carregar llista de pacients
ordrePacients = []
with open('llistaArxius.txt') as f:
```

```
    for line in f:
        ordrePacients.append(line.strip())


############    4 CLASSES DE SEVERITAT
# carregar dades de pacients
temp = []
with open('DadesPacients.csv') as f:
    for line in f:
        temp.append(line.strip().split(','))
del(temp[0])
dadesPacients = {}
for el in temp:
    dadesPacients[el[0]] = el[1:]

# dades de classificació
dades = []
for pacient in ordrePacients:
    idPacient = pacient.split('_')[1][3:]
    dt = []
    with open("caracteristiques/"+pacient) as f:
        for line in f:
            dt.append(line.strip())

    # el·liminar llegenda
    del(dt[0])

    # construir matriu amb reals i sense 1a columna
    m = [[float(x) for x in s.split(' ')[5:-1]] for s in dt]

    preds = modelClustering.predict(m).tolist()
    d = dict(Counter(preds))
    temp = [idPacient] + dadesPacients[idPacient][:4]
    for cl in range(9):
        if cl in d:
            temp.append(str(d[cl]))
        else:
            temp.append('0')
## la primera es per canviar AHI index apneas hipoapneas(4)
##            HI index dhipoapneas(5), AI index dapneas(6)

##  LA SEGONA ES PER LES CLASSES 5 CLAAWA (7), 4 CLASSES (8)....

    temp.append(dadesPacients[idPacient][4])
    temp.append(dadesPacients[idPacient][8])
    dades.append(temp)

#Estar atent, per cada classificador(KMEANS o BIRCH) un dades.class_regr
diferent!!!
with open('dades.class_regr9.4AHI.txt', 'w') as f:
    for l in dades:
        f.write(','.join(l)+'\n')
```

```
############        MODEL KMEANS AMB 9 CLSTERS 3 CLASS


##OBRIR UN MODEL O ALTRE
with open('model.kmeans9clusters.pickled','rb') as f:
    modelClustering = pickle.load(f)


# carregar llista de pacients
ordrePacients = []
with open('llistaArxius.txt') as f:
    for line in f:
        ordrePacients.append(line.strip())


############    3 CLASSES DE SEVERITAT
# carregar dades de pacients
temp = []
with open('DadesPacients.csv') as f:
    for line in f:
        temp.append(line.strip().split(','))
del(temp[0])
dadesPacients = {}
for el in temp:
    dadesPacients[el[0]] = el[1:]

# dades de classificació
dades = []
for pacient in ordrePacients:
    idPacient = pacient.split('_')[1][3:]
    dt = []
    with open("caracteristiques/"+pacient) as f:
        for line in f:
            dt.append(line.strip())

    # el·liminar llegenda
    del(dt[0])

    # construir matriu amb reals i sense 1a columna
    m = [[float(x) for x in s.split(' ')[5:-1]] for s in dt]

    preds = modelClustering.predict(m).tolist()
    d = dict(Counter(preds))
    temp = [idPacient] + dadesPacients[idPacient][:4]
    for cl in range(9):
        if cl in d:
            temp.append(str(d[cl]))
        else:
            temp.append('0')
```

```
## la primera es per canviar AHI index apneas hipoapneas(4)
##            HI index dhipoapneas(5), AI index dapneas(6)

##  LA SEGONA ES PER LES CLASSES 5 CLAAWA (7), 4 CLASSES (8)....

    temp.append(dadesPacients[idPacient][4])
    temp.append(dadesPacients[idPacient][9])
    dades.append(temp)

#Estar atent, per cada classificador(KMEANS o BIRCH) un dades.class_regr
diferent!!!
with open('dades.class_regr9.3AHI.txt', 'w') as f:
    for l in dades:
        f.write(','.join(l)+'\n')



############        MODEL KMEANS AMB 9 CLSTERS 2 CLASS


##OBRIR UN MODEL O ALTRE
with open('model.kmeans9clusters.pickled','rb') as f:
    modelClustering = pickle.load(f)


# carregar llista de pacients
ordrePacients = []
with open('llistaArxius.txt') as f:
    for line in f:
        ordrePacients.append(line.strip())


############    2 CLASSES DE SEVERITAT
# carregar dades de pacients
temp = []
with open('DadesPacients.csv') as f:
    for line in f:
        temp.append(line.strip().split(','))
del(temp[0])
dadesPacients = {}
for el in temp:
    dadesPacients[el[0]] = el[1:]

# dades de classificació
dades = []
for pacient in ordrePacients:
    idPacient = pacient.split('_')[1][3:]
    dt = []
    with open("caracteristiques/"+pacient) as f:
        for line in f:
            dt.append(line.strip())
```

```
    # el·liminar llegenda
    del(dt[0])

    # construir matriu amb reals i sense 1a columna
    m = [[float(x) for x in s.split(' ')[5:-1]] for s in dt]

    preds = modelClustering.predict(m).tolist()
    d = dict(Counter(preds))
    temp = [idPacient] + dadesPacients[idPacient][:4]
##  NUMERO 8 ES EL NUM DE CLUSTERS
    for cl in range(9):
        if cl in d:
            temp.append(str(d[cl]))
        else:
            temp.append('0')
## la primera es per canviar AHI index apneas hipoapneas(4)
##            HI index dhipoapneas(5), AI index dapneas(6)

##  LA SEGONA ES PER LES CLASSES 5 CLAAWA (7), 4 CLASSES (8)....

    temp.append(dadesPacients[idPacient][4])
    temp.append(dadesPacients[idPacient][10])
    dades.append(temp)

#Estar atent, per cada classificador(KMEANS o BIRCH) un dades.class_regr
diferent!!!
with open('dades.class_regr9.2AHI.txt', 'w') as f:
    for l in dades:
        f.write(','.join(l)+'\n')
```

```
############        MODEL KMEANS AMB 7 CLSTERS 5 CLASS


##OBRIR UN MODEL O ALTRE
with open('model.kmeans7clusters.pickled','rb') as f:
    modelClustering = pickle.load(f)


# carregar llista de pacients
ordrePacients = []
```

```
with open('llistaArxius.txt') as f:
    for line in f:
        ordrePacients.append(line.strip())


############    5 CLASSES DE SEVERITAT
# carregar dades de pacients
temp = []
with open('DadesPacients.csv') as f:
    for line in f:
        temp.append(line.strip().split(','))
del(temp[0])
dadesPacients = {}
for el in temp:
    dadesPacients[el[0]] = el[1:]

# dades de classificació
dades = []
for pacient in ordrePacients:
    idPacient = pacient.split('_')[1][3:]
    dt = []
    with open("caracteristiques/"+pacient) as f:
        for line in f:
            dt.append(line.strip())

    # el·liminar llegenda
    del(dt[0])

    # construir matriu amb reals i sense 1a columna
    m = [[float(x) for x in s.split(' ')[5:-1]] for s in dt]

    preds = modelClustering.predict(m).tolist()
    d = dict(Counter(preds))
    temp = [idPacient] + dadesPacients[idPacient][:4]
##    EL NUMERO 8 ES EL NUMERO DE RONCS = NUM CLUSTERS
    for cl in range(7):
        if cl in d:
            temp.append(str(d[cl]))
        else:
            temp.append('0')
## la primera es per canviar AHI index apneas hipoapneas(4)
##            HI index dhipoapneas(5), AI index dapneas(6)

##  LA SEGONA ES PER LES CLASSES 5 CLASS (7), 4 CLASSES (8)....

    temp.append(dadesPacients[idPacient][4])
    temp.append(dadesPacients[idPacient][7])
    dades.append(temp)

#Estar atent, per cada classificador(KMEANS o BIRCH) un dades.class_regr
diferent!!!
with open('dades.class_regr7.5AHI.txt', 'w') as f:
    for l in dades:
```

```
        f.write(','.join(l)+'\n')



############        MODEL KMEANS AMB 7 CLSTERS 4 CLASS


##OBRIR UN MODEL O ALTRE
with open('model.kmeans7clusters.pickled','rb') as f:
    modelClustering = pickle.load(f)


# carregar llista de pacients
ordrePacients = []
with open('llistaArxius.txt') as f:
    for line in f:
        ordrePacients.append(line.strip())


############    4 CLASSES DE SEVERITAT
# carregar dades de pacients
temp = []
with open('DadesPacients.csv') as f:
    for line in f:
        temp.append(line.strip().split(','))
del(temp[0])
dadesPacients = {}
for el in temp:
    dadesPacients[el[0]] = el[1:]

# dades de classificació
dades = []
for pacient in ordrePacients:
    idPacient = pacient.split('_')[1][3:]
    dt = []
    with open("caracteristiques/"+pacient) as f:
        for line in f:
            dt.append(line.strip())

    # el·liminar llegenda
    del(dt[0])

    # construir matriu amb reals i sense 1a columna
    m = [[float(x) for x in s.split(' ')[5:-1]] for s in dt]

    preds = modelClustering.predict(m).tolist()
    d = dict(Counter(preds))
    temp = [idPacient] + dadesPacients[idPacient][:4]
    for cl in range(7):
        if cl in d:
            temp.append(str(d[cl]))
```

```
        else:
            temp.append('0')
## la primera es per canviar AHI index apneas hipoapneas(4)
##             HI index dhipoapneas(5), AI index dapneas(6)

##  LA SEGONA ES PER LES CLASSES 5 CLAAWA (7), 4 CLASSES (8)....

    temp.append(dadesPacients[idPacient][4])
    temp.append(dadesPacients[idPacient][8])
    dades.append(temp)

#Estar atent, per cada classificador(KMEANS o BIRCH) un dades.class_regr
diferent!!!
with open('dades.class_regr7.4AHI.txt', 'w') as f:
    for l in dades:
        f.write(','.join(l)+'\n')



############        MODEL KMEANS AMB 8 CLSTERS 3 CLASS


##OBRIR UN MODEL O ALTRE
with open('model.kmeans7clusters.pickled','rb') as f:
    modelClustering = pickle.load(f)


# carregar llista de pacients
ordrePacients = []
with open('llistaArxius.txt') as f:
    for line in f:
        ordrePacients.append(line.strip())


############    3 CLASSES DE SEVERITAT
# carregar dades de pacients
temp = []
with open('DadesPacients.csv') as f:
    for line in f:
        temp.append(line.strip().split(','))
del(temp[0])
dadesPacients = {}
for el in temp:
    dadesPacients[el[0]] = el[1:]

# dades de classificació
dades = []
for pacient in ordrePacients:
    idPacient = pacient.split('_')[1][3:]
    dt = []
    with open("caracteristiques/"+pacient) as f:
```

```
        for line in f:
            dt.append(line.strip())

    # el·liminar llegenda
    del(dt[0])

    # construir matriu amb reals i sense 1a columna
    m = [[float(x) for x in s.split(' ')[5:-1]] for s in dt]

    preds = modelClustering.predict(m).tolist()
    d = dict(Counter(preds))
    temp = [idPacient] + dadesPacients[idPacient][:4]
    for cl in range(7):
        if cl in d:
            temp.append(str(d[cl]))
        else:
            temp.append('0')
## la primera es per canviar AHI index apneas hipoapneas(4)
##            HI index dhipoapneas(5), AI index dapneas(6)

##  LA SEGONA ES PER LES CLASSES 5 CLAAWA (7), 4 CLASSES (8)....

    temp.append(dadesPacients[idPacient][4])
    temp.append(dadesPacients[idPacient][9])
    dades.append(temp)

#Estar atent, per cada classificador(KMEANS o BIRCH) un dades.class_regr
diferent!!!
with open('dades.class_regr7.3AHI.txt', 'w') as f:
    for l in dades:
        f.write(','.join(l)+'\n')



############        MODEL KMEANS AMB 7 CLSTERS 2 CLASS


##OBRIR UN MODEL O ALTRE
with open('model.kmeans7clusters.pickled','rb') as f:
    modelClustering = pickle.load(f)


# carregar llista de pacients
ordrePacients = []
with open('llistaArxius.txt') as f:
    for line in f:
        ordrePacients.append(line.strip())


############    2 CLASSES DE SEVERITAT
# carregar dades de pacients
```

```
temp = []
with open('DadesPacients.csv') as f:
    for line in f:
        temp.append(line.strip().split(','))
del(temp[0])
dadesPacients = {}
for el in temp:
    dadesPacients[el[0]] = el[1:]

# dades de classificació
dades = []
for pacient in ordrePacients:
    idPacient = pacient.split('_')[1][3:]
    dt = []
    with open("caracteristiques/"+pacient) as f:
        for line in f:
            dt.append(line.strip())

    # el·liminar llegenda
    del(dt[0])

    # construir matriu amb reals i sense 1a columna
    m = [[float(x) for x in s.split(' ')[5:-1]] for s in dt]

    preds = modelClustering.predict(m).tolist()
    d = dict(Counter(preds))
    temp = [idPacient] + dadesPacients[idPacient][:4]
##  NUMERO 8 ES EL NUM DE CLUSTERS
    for cl in range(7):
        if cl in d:
            temp.append(str(d[cl]))
        else:
            temp.append('0')
## la primera es per canviar AHI index apneas hipoapneas(4)
##            HI index dhipoapneas(5), AI index dapneas(6)

##  LA SEGONA ES PER LES CLASSES 5 CLAAWA (7), 4 CLASSES (8)....

    temp.append(dadesPacients[idPacient][4])
    temp.append(dadesPacients[idPacient][10])
    dades.append(temp)

#Estar atent, per cada classificador(KMEANS o BIRCH) un dades.class_regr
diferent!!!
with open('dades.class_regr7.2AHI.txt', 'w') as f:
    for l in dades:
        f.write(','.join(l)+'\n')
```

```
############    PER A 4 CLASSES DE SEVERITAT

# carregar dades de pacients
temp = []
with open('DadesPacients4class.csv') as f:
    for line in f:
        temp.append(line.strip().split(','))
del(temp[0])
dadesPacients = {}
for el in temp:
    dadesPacients[el[0]] = el[1:]

# dades de classificació
dades = []
for pacient in ordrePacients:
    idPacient = pacient.split('_')[1][3:]
    dt = []
    with open("caracteristiques/"+pacient) as f:
        for line in f:
            dt.append(line.strip())

    # el·liminar llegenda
    del(dt[0])

    # construir matriu amb reals i sense 1a columna
    m = [[float(x) for x in s.split(' ')[5:-1]] for s in dt]

    preds = modelClustering.predict(m).tolist()
    d = dict(Counter(preds))
    temp = [idPacient] + dadesPacients[idPacient][:4]
```

```
    for cl in range(numRoncs):
        if cl in d:
            temp.append(str(d[cl]))
        else:
            temp.append('0')
## la primera es per canviar AHI(4), HI(5), AI(6)

    temp.append(dadesPacients[idPacient][4])
    temp.append(dadesPacients[idPacient][7])
    dades.append(temp)

#Estar atent, per cada classificador(KMEANS o BIRCH) un dades.class_regr
diferent!!!
with open('dades.class_regr8.4AHI.txt', 'w') as f:
    for l in dades:
        f.write(','.join(l)+'\n')




############    3 CLASSES DE SEVERITAT
# carregar dades de pacients
temp = []
with open('DadesPacients3class.csv') as f:
    for line in f:
        temp.append(line.strip().split(','))
del(temp[0])
dadesPacients = {}
for el in temp:
    dadesPacients[el[0]] = el[1:]

# dades de classificació
dades = []
for pacient in ordrePacients:
    idPacient = pacient.split('_')[1][3:]
    dt = []
    with open("caracteristiques/"+pacient) as f:
        for line in f:
            dt.append(line.strip())

    # el·liminar llegenda
    del(dt[0])

    # construir matriu amb reals i sense 1a columna
    m = [[float(x) for x in s.split(' ')[5:-1]] for s in dt]

    preds = modelClustering.predict(m).tolist()
    d = dict(Counter(preds))
    temp = [idPacient] + dadesPacients[idPacient][:4]
    for cl in range(numRoncs):
        if cl in d:
            temp.append(str(d[cl]))
```

```
        else:
            temp.append('0')
## la primera es per canviar AHI(4), HI(5), AI(6)

    temp.append(dadesPacients[idPacient][4])
    temp.append(dadesPacients[idPacient][7])
    dades.append(temp)

#Estar atent, per cada classificador(KMEANS o BIRCH) un dades.class_regr
diferent!!!
with open('dades.class_regr8.3AHI.txt', 'w') as f:
    for l in dades:
        f.write(','.join(l)+'\n')



############    PER A 2 CLASSES DE SEVERITAT

# carregar dades de pacients
temp = []
with open('DadesPacients2class.csv') as f:
    for line in f:
        temp.append(line.strip().split(','))
del(temp[0])
dadesPacients = {}
for el in temp:
    dadesPacients[el[0]] = el[1:]

# dades de classificació
dades = []
for pacient in ordrePacients:
    idPacient = pacient.split('_')[1][3:]
    dt = []
    with open("caracteristiques/"+pacient) as f:
        for line in f:
            dt.append(line.strip())

    # el·liminar llegenda
    del(dt[0])

    # construir matriu amb reals i sense 1a columna
    m = [[float(x) for x in s.split(' ')[5:-1]] for s in dt]

    preds = modelClustering.predict(m).tolist()
    d = dict(Counter(preds))
    temp = [idPacient] + dadesPacients[idPacient][:4]
    for cl in range(numRoncs):
        if cl in d:
            temp.append(str(d[cl]))
        else:
            temp.append('0')
## la primera es per canviar AHI(4), HI(5), AI(6)
```

```
    temp.append(dadesPacients[idPacient][4])
    temp.append(dadesPacients[idPacient][7])
    dades.append(temp)

#Estar atent, per cada classificador(KMEANS o BIRCH) un dades.class_regr
diferent!!!
with open('dades.class_regr8.2AHI.txt', 'w') as f:
    for l in dades:
        f.write(','.join(l)+'\n')









############        MODEL KMEANS AMB 9 CLSTERS


##OBRIR UN MODEL O ALTRE
with open('model.kmeans9clusters.pickled','rb') as f:
    modelClustering = pickle.load(f)


# carregar llista de pacients
ordrePacients = []
with open('llistaArxius.txt') as f:
    for line in f:
        ordrePacients.append(line.strip())


############    5 CLASSES DE SEVERITAT
# carregar dades de pacients
temp = []
with open('DadesPacients5class.csv') as f:
    for line in f:
        temp.append(line.strip().split(','))
del(temp[0])
dadesPacients = {}
for el in temp:
    dadesPacients[el[0]] = el[1:]

# dades de classificació
dades = []
for pacient in ordrePacients:
    idPacient = pacient.split('_')[1][3:]
    dt = []
    with open("caracteristiques/"+pacient) as f:
```

```
        for line in f:
            dt.append(line.strip())

    # el·liminar llegenda
    del(dt[0])

    # construir matriu amb reals i sense 1a columna
    m = [[float(x) for x in s.split(' ')[5:-1]] for s in dt]

    preds = modelClustering.predict(m).tolist()
    d = dict(Counter(preds))
    temp = [idPacient] + dadesPacients[idPacient][:4]
    for cl in range(numRoncs):
        if cl in d:
            temp.append(str(d[cl]))
        else:
            temp.append('0')
## la primera es per canviar AHI(4), HI(5), AI(6)

    temp.append(dadesPacients[idPacient][4])
    temp.append(dadesPacients[idPacient][7])
    dades.append(temp)

#Estar atent, per cada classificador(KMEANS o BIRCH) un dades.class_regr
diferent!!!
with open('dades.class_regr9.5AHI.txt', 'w') as f:
    for l in dades:
        f.write(','.join(l)+'\n')


############    PER A 4 CLASSES DE SEVERITAT

# carregar dades de pacients
temp = []
with open('DadesPacients4class.csv') as f:
    for line in f:
        temp.append(line.strip().split(','))
del(temp[0])
dadesPacients = {}
for el in temp:
    dadesPacients[el[0]] = el[1:]

# dades de classificació
dades = []
for pacient in ordrePacients:
    idPacient = pacient.split('_')[1][3:]
    dt = []
    with open("caracteristiques/"+pacient) as f:
        for line in f:
            dt.append(line.strip())

    # el·liminar llegenda
    del(dt[0])
```

```
    # construir matriu amb reals i sense 1a columna
    m = [[float(x) for x in s.split(' ')[5:-1]] for s in dt]

    preds = modelClustering.predict(m).tolist()
    d = dict(Counter(preds))
    temp = [idPacient] + dadesPacients[idPacient][:4]
    for cl in range(numRoncs):
        if cl in d:
            temp.append(str(d[cl]))
        else:
            temp.append('0')
## la primera es per canviar AHI(4), HI(5), AI(6)

    temp.append(dadesPacients[idPacient][4])
    temp.append(dadesPacients[idPacient][7])
    dades.append(temp)

#Estar atent, per cada classificador(KMEANS o BIRCH) un dades.class_regr
diferent!!!
with open('dades.class_regr9.4AHI.txt', 'w') as f:
    for l in dades:
        f.write(','.join(l)+'\n')
```

```
############    3 CLASSES DE SEVERITAT
# carregar dades de pacients
temp = []
with open('DadesPacients3class.csv') as f:
    for line in f:
        temp.append(line.strip().split(','))
del(temp[0])
dadesPacients = {}
for el in temp:
    dadesPacients[el[0]] = el[1:]

# dades de classificació
dades = []
for pacient in ordrePacients:
    idPacient = pacient.split('_')[1][3:]
    dt = []
    with open("caracteristiques/"+pacient) as f:
        for line in f:
            dt.append(line.strip())

    # el·liminar llegenda
    del(dt[0])

    # construir matriu amb reals i sense 1a columna
    m = [[float(x) for x in s.split(' ')[5:-1]] for s in dt]
```

```
    preds = modelClustering.predict(m).tolist()
    d = dict(Counter(preds))
    temp = [idPacient] + dadesPacients[idPacient][:4]
    for cl in range(numRoncs):
        if cl in d:
            temp.append(str(d[cl]))
        else:
            temp.append('0')
## la primera es per canviar AHI(4), HI(5), AI(6)

    temp.append(dadesPacients[idPacient][4])
    temp.append(dadesPacients[idPacient][7])
    dades.append(temp)

#Estar atent, per cada classificador(KMEANS o BIRCH) un dades.class_regr
diferent!!!
with open('dades.class_regr9.3AHI.txt', 'w') as f:
    for l in dades:
        f.write(','.join(l)+'\n')


############    PER A 2 CLASSES DE SEVERITAT

# carregar dades de pacients
temp = []
with open('DadesPacients2class.csv') as f:
    for line in f:
        temp.append(line.strip().split(','))
del(temp[0])
dadesPacients = {}
for el in temp:
    dadesPacients[el[0]] = el[1:]

# dades de classificació
dades = []
for pacient in ordrePacients:
    idPacient = pacient.split('_')[1][3:]
    dt = []
    with open("caracteristiques/"+pacient) as f:
        for line in f:
            dt.append(line.strip())

    # el·liminar llegenda
    del(dt[0])

    # construir matriu amb reals i sense 1a columna
    m = [[float(x) for x in s.split(' ')[5:-1]] for s in dt]

    preds = modelClustering.predict(m).tolist()
    d = dict(Counter(preds))
    temp = [idPacient] + dadesPacients[idPacient][:4]
    for cl in range(numRoncs):
```

```
        if cl in d:
            temp.append(str(d[cl]))
        else:
            temp.append('0')
## la primera es per canviar AHI(4), HI(5), AI(6)

    temp.append(dadesPacients[idPacient][4])
    temp.append(dadesPacients[idPacient][7])
    dades.append(temp)

#Estar atent, per cada classificador(KMEANS o BIRCH) un dades.class_regr
diferent!!!
with open('dades.class_regr9.2AHI.txt', 'w') as f:
    for l in dades:
        f.write(','.join(l)+'\n')




############        MODEL KMEANS AMB 7 CLSTERS



##OBRIR UN MODEL O ALTRE
with open('model.kmeans7clusters.pickled','rb') as f:
    modelClustering = pickle.load(f)

##with open('model.birch.pickled','rb') as f:
##    modelClustering = pickle.load(f)



# carregar llista de pacients
ordrePacients = []
with open('llistaArxius.txt') as f:
    for line in f:
        ordrePacients.append(line.strip())


############    5 CLASSES DE SEVERITAT
# carregar dades de pacients
temp = []
with open('DadesPacients5class.csv') as f:
    for line in f:
        temp.append(line.strip().split(','))
del(temp[0])
dadesPacients = {}
for el in temp:
    dadesPacients[el[0]] = el[1:]

# dades de classificació
dades = []
```

```
for pacient in ordrePacients:
    idPacient = pacient.split('_')[1][3:]
    dt = []
    with open("caracteristiques/"+pacient) as f:
        for line in f:
            dt.append(line.strip())

    # el·liminar llegenda
    del(dt[0])

    # construir matriu amb reals i sense 1a columna
    m = [[float(x) for x in s.split(' ')[5:-1]] for s in dt]

    preds = modelClustering.predict(m).tolist()
    d = dict(Counter(preds))
    temp = [idPacient] + dadesPacients[idPacient][:4]
    for cl in range(numRoncs):
        if cl in d:
            temp.append(str(d[cl]))
        else:
            temp.append('0')
## la primera es per canviar AHI(4), HI(5), AI(6)

    temp.append(dadesPacients[idPacient][4])
    temp.append(dadesPacients[idPacient][7])
    dades.append(temp)

#Estar atent, per cada classificador(KMEANS o BIRCH) un dades.class_regr
diferent!!!
with open('dades.class_regr7.5AHI.txt', 'w') as f:
    for l in dades:
        f.write(','.join(l)+'\n')


############    PER A 4 CLASSES DE SEVERITAT

# carregar dades de pacients
temp = []
with open('DadesPacients4class.csv') as f:
    for line in f:
        temp.append(line.strip().split(','))
del(temp[0])
dadesPacients = {}
for el in temp:
    dadesPacients[el[0]] = el[1:]

# dades de classificació
dades = []
for pacient in ordrePacients:
    idPacient = pacient.split('_')[1][3:]
    dt = []
    with open("caracteristiques/"+pacient) as f:
        for line in f:
```

```
            dt.append(line.strip())

    # el·liminar llegenda
    del(dt[0])

    # construir matriu amb reals i sense 1a columna
    m = [[float(x) for x in s.split(' ')[5:-1]] for s in dt]

    preds = modelClustering.predict(m).tolist()
    d = dict(Counter(preds))
    temp = [idPacient] + dadesPacients[idPacient][:4]
    for cl in range(numRoncs):
        if cl in d:
            temp.append(str(d[cl]))
        else:
            temp.append('0')
## la primera es per canviar AHI(4), HI(5), AI(6)

    temp.append(dadesPacients[idPacient][4])
    temp.append(dadesPacients[idPacient][7])
    dades.append(temp)

#Estar atent, per cada classificador(KMEANS o BIRCH) un dades.class_regr
diferent!!!
with open('dades.class_regr7.4AHI.txt', 'w') as f:
    for l in dades:
        f.write(','.join(l)+'\n')




############    3 CLASSES DE SEVERITAT
# carregar dades de pacients
temp = []
with open('DadesPacients3class.csv') as f:
    for line in f:
        temp.append(line.strip().split(','))
del(temp[0])
dadesPacients = {}
for el in temp:
    dadesPacients[el[0]] = el[1:]

# dades de classificació
dades = []
for pacient in ordrePacients:
    idPacient = pacient.split('_')[1][3:]
    dt = []
    with open("caracteristiques/"+pacient) as f:
        for line in f:
            dt.append(line.strip())

    # el·liminar llegenda
```

```
    del(dt[0])

    # construir matriu amb reals i sense 1a columna
    m = [[float(x) for x in s.split(' ')[5:-1]] for s in dt]

    preds = modelClustering.predict(m).tolist()
    d = dict(Counter(preds))
    temp = [idPacient] + dadesPacients[idPacient][:4]
    for cl in range(numRoncs):
        if cl in d:
            temp.append(str(d[cl]))
        else:
            temp.append('0')
## la primera es per canviar AHI(4), HI(5), AI(6)

    temp.append(dadesPacients[idPacient][4])
    temp.append(dadesPacients[idPacient][7])
    dades.append(temp)

#Estar atent, per cada classificador(KMEANS o BIRCH) un dades.class_regr
diferent!!!
with open('dades.class_regr7.3AHI.txt', 'w') as f:
    for l in dades:
        f.write(','.join(l)+'\n')


############    PER A 2 CLASSES DE SEVERITAT

# carregar dades de pacients
temp = []
with open('DadesPacients2class.csv') as f:
    for line in f:
        temp.append(line.strip().split(','))
del(temp[0])
dadesPacients = {}
for el in temp:
    dadesPacients[el[0]] = el[1:]

# dades de classificació
dades = []
for pacient in ordrePacients:
    idPacient = pacient.split('_')[1][3:]
    dt = []
    with open("caracteristiques/"+pacient) as f:
        for line in f:
            dt.append(line.strip())

    # el·liminar llegenda
    del(dt[0])

    # construir matriu amb reals i sense 1a columna
    m = [[float(x) for x in s.split(' ')[5:-1]] for s in dt]
```

```
    preds = modelClustering.predict(m).tolist()
    d = dict(Counter(preds))
    temp = [idPacient] + dadesPacients[idPacient][:4]
    for cl in range(numRoncs):
        if cl in d:
            temp.append(str(d[cl]))
        else:
            temp.append('0')
## la primera es per canviar AHI(4), HI(5), AI(6)

    temp.append(dadesPacients[idPacient][4])
    temp.append(dadesPacients[idPacient][7])
    dades.append(temp)

#Estar atent, per cada classificador(KMEANS o BIRCH) un dades.class_regr
diferent!!!
with open('dades.class_regr7.2AHI.txt', 'w') as f:
    for l in dades:
        f.write(','.join(l)+'\n')
```

d) Pas3.py. Implementació i obtenció de resultats de l'estudi 3:

```
from sklearn.svm import SVR, SVC
from sklearn.ensemble import RandomForestClassifier,RandomForestRegressor,
AdaBoostClassifier, AdaBoostRegressor
from sklearn.neighbors import KNeighborsClassifier, KNeighborsRegressor
from sklearn.naive_bayes import GaussianNB
from sklearn.tree import DecisionTreeClassifier, DecisionTreeRegressor
from sklearn.metrics import r2_score, mean_squared_error, accuracy_score,
confusion_matrix
from sklearn.grid_search import GridSearchCV

# carrega
dades = []

with open('dades.class_regr9.5AHI.txt') as f:
##with open('dades.class_regr9.4AHI.txt') as f:
##with open('dades.class_regr9.3AHI.txt') as f:
##with open('dades.class_regr9.2AHI.txt') as f:
##with open('dades.class_regr8.5AHI.txt') as f:
##with open('dades.class_regr8.4AHI.txt') as f:
##with open('dades.class_regr8.3AHI.txt') as f:
##with open('dades.class_regr8.2AHI.txt') as f:
##with open('dades.class_regr7.5AHI.txt') as f:
##with open('dades.class_regr7.4AHI.txt') as f:
##with open('dades.class_regr7.3AHI.txt') as f:
##with open('dades.class_regr7.2AHI.txt') as f:

##with open('dades.class_regr9.5HI.txt') as f:
##with open('dades.class_regr9.4HI.txt') as f:
##with open('dades.class_regr9.3HI.txt') as f:
##with open('dades.class_regr9.2HI.txt') as f:
##with open('dades.class_regr8.5HI.txt') as f:
```

```
##with open('dades.class_regr8.4HI.txt') as f:
##with open('dades.class_regr8.3HI.txt') as f:
##with open('dades.class_regr8.2HI.txt') as f:
##with open('dades.class_regr7.5HI.txt') as f:
##with open('dades.class_regr7.4HI.txt') as f:
##with open('dades.class_regr7.3HI.txt') as f:
##with open('dades.class_regr7.2HI.txt') as f:


##with open('dades.class_regr9.5AI.txt') as f:
##with open('dades.class_regr9.4AI.txt') as f:
##with open('dades.class_regr9.3AI.txt') as f:
##with open('dades.class_regr9.2AI.txt') as f:
##with open('dades.class_regr8.5AI.txt') as f:
##with open('dades.class_regr8.4AI.txt') as f:
##with open('dades.class_regr8.3AI.txt') as f:
##with open('dades.class_regr8.2AI.txt') as f:
##with open('dades.class_regr7.5AI.txt') as f:
##with open('dades.class_regr7.4AI.txt') as f:
##with open('dades.class_regr7.3AI.txt') as f:
##with open('dades.class_regr7.2AI.txt') as f:

    for line in f:
        dades.append(line.strip().split(','))

#SI NO VOLEM EL IDENTIFICADOR SIG POSSAREM 1:-2, SI NO VOLEM EL ID 0:-2
X = [[float(x) for x in l[1:-2]] for l in dades]
Yregr = [float(l[-2]) for l in dades]
Yclass = [l[-1] for l in dades]


print("")
print("RANDOM FOREST:")
print("")

parametresRF = [
     {'n_estimators': [ 5, 10, 25, 50, 100]}]

##n_estimators=10,
##criterion='gini',
##max_depth=None,
##min_samples_split=2,
##min_samples_leaf=1,
##min_weight_fraction_leaf=0.0,
##max_features='auto',
##max_leaf_nodes=None,
##min_impurity_split=1e-07,
##bootstrap=True,
##oob_score=False,
##n_jobs=1,
##random_state=None,
##verbose=0,
##warm_start=False,
##class_weight=None

# classificacio
algoritme = RandomForestClassifier()
clf = GridSearchCV(algoritme, parametresRF)
```

```
clf.fit(X[:llindar], Yclass[:llindar])
preds = clf.predict(X[llindar:]).tolist()
print("Acc:", accuracy_score(Yclass[llindar:], preds))
print(confusion_matrix(Yclass[llindar:], preds))

#Parametres regr

##n_estimators=10,
##criterion='mse',
##max_depth=None,
##min_samples_split=2,
##min_samples_leaf=1,
##min_weight_fraction_leaf=0.0,
##max_features='auto',
##max_leaf_nodes=None,
##min_impurity_split=1e-07,
##bootstrap=True,
##oob_score=False,
##n_jobs=1,
##random_state=None,
##verbose=0,
##warm_start=False


# regressio
algoritme = RandomForestRegressor()
clf = GridSearchCV(algoritme, parametresRF)
clf.fit(X[:llindar], Yregr[:llindar])
preds = clf.predict(X[llindar:]).tolist()
print("R2:", r2_score(Yregr[llindar:], preds))
print("mse:", mean_squared_error(Yregr[llindar:], preds))



print("")
print("DECISION TREE:")
print("")

#parameters class

##criterion='gini',
##splitter='best',
##learning_rate=0.1,
##max_depth=None,
##min_samples_split=2,
##min_samples_leaf=1,
##min_weight_fraction_leaf=0.0,
##max_features=None,
##random_state=None,
##max_leaf_nodes=None,
##min_impurity_split=1e-07,
##class_weight=None,
##presort=False,

#parameters class
```

```
# classificacio
parametresDECISION1 = [
  {'criterion': ['gini', 'entropy']}]

algoritme = DecisionTreeClassifier()
clf = GridSearchCV(algoritme, parametresDECISION1)
clf.fit(X[:llindar], Yclass[:llindar])
preds = clf.predict(X[llindar:]).tolist()
print("Model:", clf.best_params_)
print("Acc:", accuracy_score(Yclass[llindar:], preds))
print(confusion_matrix(Yclass[llindar:], preds))

# Parametres regrç

##criterion='mse',
##splitter='best',
##max_depth=None,
##min_samples_split=2,
##min_samples_leaf=1,
##min_weight_fraction_leaf=0.0,
##max_features=None,
##random_state=None,
##max_leaf_nodes=None,
##min_impurity_split=1e-07,
##presort=False

parametresDECISION2 = [
  {'criterion': ['mse', 'mae']}]


algoritme = DecisionTreeRegressor()
clf = GridSearchCV(algoritme, parametresDECISION2)
clf.fit(X[:llindar], Yregr[:llindar])
preds = clf.predict(X[llindar:]).tolist()
print("Model:", clf.best_params_)
print("R2:", r2_score(Yregr[llindar:], preds))
print("mse:", mean_squared_error(Yregr[llindar:], preds))



print("")
print("GAUSSIAN NB - nomes class:")
print("")


# classificacio
clf = GaussianNB()
clf.fit(X[:llindar], Yclass[:llindar])
GaussianNB(priors=None)
preds = clf.predict(X[llindar:]).tolist()

print("Acc:", accuracy_score(Yclass[llindar:], preds))
print(confusion_matrix(Yclass[llindar:], preds))


print("")
print("KNEIGHBORS:")
```

```
print("")


parametresKNEIGHBORS1 = [
    {'n_neighbors': [1, 3, 5, 7, 11], 'weights' : ['uniform','distance']},
    {'n_neighbors': [1, 3, 5, 7, 11], 'algorithm' :
['auto','ball_tree','kd_tree', 'brute']}]


##n_neighbors=5,
##weights='uniform',
##algorithm='auto',
##leaf_size=30,
##p=2,
##metric='minkowski',
##metric_params=None,
##n_jobs=1,


## classificacio
algoritme = KNeighborsClassifier()
clf = GridSearchCV(algoritme, parametresKNEIGHBORS1)
clf.fit(X[:llindar], Yclass[:llindar])
preds = clf.predict(X[llindar:]).tolist()
print("Model:", clf.best_params_)
print("Acc:", accuracy_score(Yclass[llindar:], preds))
print(confusion_matrix(Yclass[llindar:], preds))



## regressio
algoritme = KNeighborsRegressor()
clf = GridSearchCV(algoritme, parametresKNEIGHBORS1)
clf.fit(X[:llindar], Yregr[:llindar])
preds = clf.predict(X[llindar:]).tolist()
print("Model:", clf.best_params_)
print("R2:", r2_score(Yregr[llindar:], preds))
print("mse:", mean_squared_error(Yregr[llindar:], preds))




print("")
print("ADABOOST:")
print("")

##base_estimator='DecisionTreeClassifier',
###començar a 100 i anar-lo baixant
##n_estimators=50,
##learning_rate=1.0,
###SAMME', 'SAMME.R',(default='SAMME.R')
##algorithm='SAMME.R',
##random_state=None,

parametresADABOOST1 = [
```

```
     {'n_estimators': [ 5, 10, 25, 50, 100]}]

# classificacio

svm = AdaBoostClassifier()
clf = GridSearchCV(svm, parametresADABOOST1)
clf.fit(X[:llindar], Yclass[:llindar])
preds = clf.predict(X[llindar:]).tolist()
print("Model:", clf.best_params_)
print("Acc:", accuracy_score(Yclass[llindar:], preds))
print(confusion_matrix(Yclass[llindar:], preds))

parametresADABOOST2 = [
     {'n_estimators': [ 5, 10, 25, 50, 100]}]

##base_estimator=None,
##n_estimators=50,
##learning_rate=1.0,
###'linear', 'square', 'exponential'}, optional (default='linear')
##loss='linear',
##random_state=None

# regressio
svm = AdaBoostRegressor()
clf = GridSearchCV(svm, parametresADABOOST2)
clf.fit(X[:llindar], Yregr[:llindar])
preds = clf.predict(X[llindar:]).tolist()
print("Model:", clf.best_params_)
print("R2:", r2_score(Yregr[llindar:], preds))
print("mse:", mean_squared_error(Yregr[llindar:], preds))



##  SVM
print("")
print("SVM:")
print("")

parametersSVM1 = [
   {'C': [1e-3, 1e-2, 0.1, 1, 10, 100, 1000], 'gamma': [0.1, 0.01,0.001,
0.0001]},
   {'C': [1e-3, 1e-2, 0.1, 1, 10, 100, 1000], 'degree': [2, 3]}
  ]

 # classificacio
svm = SVC()
clf = GridSearchCV(svm, parametersSVM1)
clf.fit(X[:llindar], Yclass[:llindar])
preds = clf.predict(X[llindar:]).tolist()
print("Model:", clf.best_params_)
print("Acc:", accuracy_score(Yclass[llindar:], preds))
print(confusion_matrix(Yclass[llindar:], preds))

parametresSVM2 = [
  {'C': [1e-3, 1e-2, 0.1, 1, 10, 100, 1000], 'kernel': ['linear']},
  {'C': [1e-3, 1e-2, 0.1, 1, 10, 100, 1000], 'gamma': [0.1, 0.01,0.001,
0.0001], 'kernel': ['rbf']},
```

```
  {'C': [1e-3, 1e-2, 0.1, 1, 10, 100, 1000], 'degree': [2, 3], 'kernel':
['poly']},
  {'epsilon' : [0.1, 0.15, 0.2, 0.3,0.5, 0.8, 0.9, 1.0, 1.5, 1.8, 2.0]}
 ]

# regressio
svm = SVR()
clf = GridSearchCV(svm, parametresSVM2)
clf.fit(X[:llindar], Yregr[:llindar])
preds = clf.predict(X[llindar:]).tolist()
print("Model:", clf.best_params_)
print("R2:", r2_score(Yregr[llindar:], preds))
print("mse:", mean_squared_error(Yregr[llindar:], preds))
```