**Data analysis (Python): Randon forest regressors (Figure 4)**

# -\*- coding: utf-8 -\*-

"""

Created on Wed Mar 3 08:00:24 2021

@author: Gush

"""

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import statistics

dataset = pd.read\_csv(r'C:\Users\gusta\Desktop\Quarantene work 28.05.21\ASIA conference - 2022 - New Orleans\Abstract 1 - Recovery profiles FEST\Recovery profile X DST for heatmap.txt', delimiter ='\t')

X = dataset.iloc[:, 0].values

y = dataset.iloc[:, 1:3].values

print(y)

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.5, random\_state=0)

from sklearn.ensemble import RandomForestRegressor

regressor = RandomForestRegressor(n\_estimators=100, random\_state=0)

regressor.fit(X, y)

y\_pred = regressor.predict(X\_test)

plt.scatter(X, y, color = 'blue')

X\_grid = np.arange(min(X), max(X), 0.01)

X\_grid = X\_grid.reshape((len(X\_grid), 1))

plt.plot(X\_grid, regressor.predict(X\_grid),

color = 'red')

plt.title('Random Forest Regression')

plt.xlabel('Initial impairment (5 - baseline MS)')

plt.ylabel('Change (final - baseline MS)')

plt.show()

from sklearn.metrics import r2\_score

R2 = r2\_score(y\_test, y\_pred)

Pred\_error = y\_test-y\_pred

Abs\_Pred\_error = abs(Pred\_error)

Average\_Prediction\_error = sum(Abs\_Pred\_error)/len(Abs\_Pred\_error)

SD\_Average\_Prediction\_error = statistics.stdev(Abs\_Pred\_error)

print (Abs\_Pred\_error)

print ('R2 score =', R2)

print('Average prediction error =', Average\_Prediction\_error)

print ('Error SD =', SD\_Average\_Prediction\_error)

**Data analysis (Python): Leave one muscle out cross-validation code (Figures 5 and 6)**

# -\*- coding: utf-8 -\*-

"""

Created on Thu Jan 21 14:05:36 2021

@author: Gush

"""

import pandas as pd

import numpy as np

dataset = pd.read\_csv(r'C: path', delimiter ='\t')

print(dataset)

X = dataset.iloc[:, 0:4].values

print(X)

y = dataset.iloc[:, 4].values

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.5, random\_state=0)

from sklearn.ensemble import RandomForestClassifier

regressor = RandomForestClassifier(n\_estimators=100, random\_state=0)

regressor.fit(X\_train, y\_train)

y\_pred = regressor.predict(X\_test)

from sklearn.metrics import classification\_report, confusion\_matrix, accuracy\_score

print(confusion\_matrix(y\_train,y\_pred))

print(classification\_report(y\_train,y\_pred))

print(accuracy\_score(y\_train, y\_pred))

print(y\_test)

print(y\_pred)

target\_names = ['No rec', 'Rec'];

#output\_dict make the report saved as a dictionary

A=classification\_report(y\_test, y\_pred, target\_names=target\_names, output\_dict=True)

B=confusion\_matrix(y\_test, y\_pred)

#Read values as

class\_0\_f1score=A['No rec']['f1-score']

print(A)

print(B)

#Leave one out cross validation

from sklearn.model\_selection import LeaveOneOut

X = dataset.iloc[:, 0:4].values

y = dataset.iloc[:, 4].values

loo = LeaveOneOut()

loo.get\_n\_splits(X)

print(loo)

LeaveOneOut()

LOOCV\_pred = []

for train\_index, test\_index in loo.split(X):

print("TRAIN:", train\_index, "TEST:", test\_index)

X\_LOOCV\_train, X\_LOOCV\_test = X[train\_index], X[test\_index]

y\_LOOCV\_train, y\_LOOCV\_test = y[train\_index], y[test\_index]

print(X\_LOOCV\_train, X\_LOOCV\_test, y\_LOOCV\_train, y\_LOOCV\_test)

LOOCV\_pred.append(y\_LOOCV\_test)

print('y\_test =', y)

print('y\_pred =', LOOCV\_pred)

import csv

LOOCV\_output = open(r'C: path', 'w+', newline ='')

write = csv.writer(LOOCV\_output)

write.writerow('y\_test')

write.writerow(y)

write.writerow('y\_pred')

write.writerow(LOOCV\_pred)