ANEXO I

```
df = pd.read_csv (r'C:\Users\usuario\Desktop\Universidad\#tfg\Debernardi et al 2020 data.csv')

df = df[["sex","age","plasma_CA19_9","creatinine","LYVE1","REG1B","TFF1","diagnosis"]]
df = df.dropna(axis='rows')                              # Quitar filas con NaNs
df = df.reset_index()
df = df.drop(["index"], axis=1)

for i in range(len(df)) :                               # Hombres = 1, Mujeres = 0
  if (df.iloc[i, 0] == "M"):
    df.iloc[i, 0] = 1
  else:
    df.iloc[i, 0] = 0

df_nums = df[["age","plasma_CA19_9","creatinine","LYVE1","REG1B","TFF1"]]

z = np.abs(stats.zscore(df_nums))
df = df[(z < 4).all(axis = 1)]
df_nums = df_nums[(z < 4).all(axis = 1)]

normalized_df = (df_nums-df_nums.min()) / (df_nums.max() - df_nums.min())

sex_df = df["sex"]
diagnosis = df["diagnosis"]
                                                        # Trabajamos con los valores numéricos
final_df = pd.concat([sex_df,normalized_df,diagnosis],axis = 1)


del(sex_df)
del(diagnosis)

for i in range(len(final_df)):                          # Maligno = 1, Benigno = -1, Control/Sano = 0
  if (final_df.iloc[i, 7] == 1):
    final_df.iloc[i, 7] = 0
  elif (final_df.iloc[i, 7] == 2):
    final_df.iloc[i, 7] = -1
  else:
    final_df.iloc[i, 7] = 1


#################### TEST/TRAIN ####################################
msk = np.random.rand(len(final_df)) <= 0.8

train = final_df[msk]
test = final_df[~msk]
                                                         # Trabajamos solo con C/M
train = train[train.diagnosis != -1]
train = train.reset_index()
train = train.drop(["index"], axis = 1)

test = test[test.diagnosis != -1]
test = test.reset_index()
test = test.drop(["index"], axis = 1)

#################### CONTAR E IGUALAR ##############################
def contar():
  diagnosis = list(train["diagnosis"])
  diagnosis_test = list(test["diagnosis"])

  global maligno, sano
  maligno, sano = 0, 0

  for k in range(len(diagnosis)):
    if(diagnosis[k] == 1):
      maligno += 1

    elif(diagnosis[k] == 0):
      sano += 1

  print (" Sanos: %s \n Malignos: %s \n " %(sano, maligno))
```

```
############## IGUALAR ###############
def igualar(tr,ts):
  global tst, trn, maligno, sano

  peq = min(sano,maligno)

  diferencia_e = maligno - peq
  rows_m = tr.iloc[len(tr) - diferencia_e:len(tr)]
  ts = pd.concat([ts,rows_m], ignore_index = True)
  tr = tr.drop(tr.index[len(tr) - diferencia_e:len(tr)])

  diferencia_s = sano-peq
  rows_s = tr.iloc[1:diferencia_s]
  tst = pd.concat([ts,rows_s], ignore_index = True)
  trn = tr.drop(train.index[1:diferencia_s])

#############

contar()

igualar(train,test)
train = trn
test = tst

contar()

diagnosis = list(train["diagnosis"])
diagnosis_test = list(test["diagnosis"])
```

Figura 15. Procesamiento de datos para la creación del archivo usado para los algoritmos diseñados para la clasificación en C y M. Escrito en Python utilizando RStudio.

```
def f_apt(w):
  a = 0
  for k in range(len(train)):
    g_exp = 0

    for j in range(len(w)):
      g = w[j] * train.iloc[k, j]
      g_exp += g

    sigmoide = (1 / (1 + np.exp(-g_exp)))
    a_parcial = 1 - abs (diagnosis[k] - sigmoide)

    a += a_parcial

  return a
```

Figura 16. Función de aptitud utilizada en el AG01. Escrito en Python utilizando RStudio.

```
def f_apt(w):
  a = 0
  for k in range(len(train)):
    g_exp = 0

    for j in range(len(w)):
      g = w[j] * train.iloc[k, j]
      g_exp += g

    sigmoide = (1 / (1 + np.exp(-g_exp)))

    if(math.isclose(sigmoide, diagnosis[k], abs_tol = u)):
        a += 1

  return a
```

Figura 17. Función de aptitud utilizada en el AG02. Escrito en Python utilizando RStudio.

```
def f_apt(w1,w2):
  a = 0
  for k in range(len(train)):
    g_exp = 0

    for j in range(atribs):
      g = w1[j] * ((pob1.iloc[k, j]) ** (w1[j + atribs]))
      g_exp += g

    sigmoide1 = 1 / (1 + np.exp(-w1[atribs * 2] * g_exp))

    g_exp = 0

    for j in range(atribs):
      g = w2[j] * ((pob2.iloc[k, j]) ** (w2[j + atribs]))
      g_exp += g

    sigmoide2 = 1 / (1 + np.exp(-w2[atribs * 2] * g_exp))

    if (math.isclose(sigmoide1, 0, abs_tol = u) and math.isclose(sigmoide2, 1, abs_tol = u)):
      a = 0
      k = len(train) + 1

    else:
      y = sigmoide1 + sigmoide2
      a_parcial = 2 - abs (diagnosis[k] - y)
      a += a_parcial

  return a
```

Figura 18. Función de aptitud utilizada en el AG05 utilizando valores de *r* y α. Véase la pág. 12 para comprobar el orden de los elementos dentro del individuo *w*. Escrito Python utilizando RStudio.