Proyecto final Inteligencia Artificial.

AUTORES:

Julian David Betancourt Marin

PRESENTADO A:
Ing. Francisco Carlos Calderón Bocanegra PH. D

PONTIFICIA UNIVERSIDAD JAVERIANA
FACULTAD DE INGENIERÍA
DEPARTAMENTO DE ELECTRÓNICA
BOGOTÁ D.C. 2022

El presente proyecto se realizó a partir de un proyecto de grado en curso, el cual es análisis de termogramas para detección de diabetes temprana, en donde sabemos que un pie que presenta síntomas de pie diabético, y ulceración, tendera a estar mas caliente que los pies de pacientes sanos, por lo que este es un problema de clasificación no supervisado ya que esta forma de clasificación está basada en píxeles y es esencialmente una clasificación automatizada por computadora.

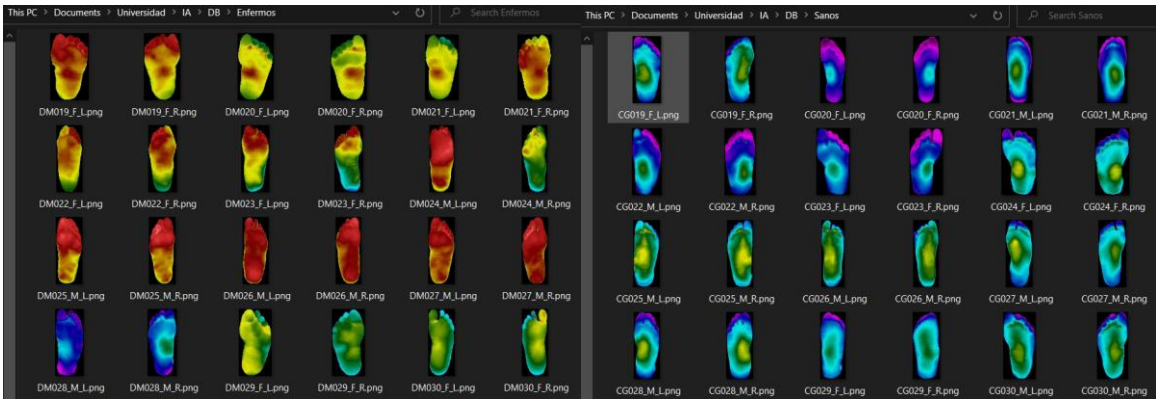
A partir de una bace de datos de pies proveniente de IEEE Data Port, se realizó un primer algoritmo capaz de separar pacientes sanos de pacientes diabéticos utilizando termogramas.

El objetivo de este proyecto es obtener un F1 score(estadística que me da la medida de precisión que tiene el código para clasificar correctamente a los pacientes) de mas de 90%, Las librerías utilizadas se muestran a continuación

```
#%% Librerias
import numpy as np
import cv2
import pandas as pd
from keras.models import Sequential
from keras.layers import Dense
from IPython import get_ipython
from sklearn import model_selection
from sklearn.metrics import confusion_matrix, f1_score,accuracy_score
Run Cell | Run Above | Debug Cell
#%% Limpiar pantalla y variables
print('\014')
Run Cell | Run Above | Debug Cell
#%%get_ipython().magic('reset -sf')
```

Posteriormente se importa la base de datos de imágenes desde una carpeta del computador, colocando la Ruta de la carpeta que contiene las imágenes.

```
#%% Importar Base de datos
Sanos=Leer_DB(r'C:\Users\Julian\OneDrive - Pontificia Universidad Javeriana\Documents\Universidad\IA\DB\Sanos', 'png', 'Pacientes_Sanos',65,168)
Enfermos=Leer_DB(r'C:\Users\Julian\OneDrive - Pontificia Universidad Javeriana\Documents\Universidad\IA\DB\Enfermos', 'png', 'Pacientes_Sanos',65,168)
Run Cell | Run Above | Debug Cell
```



Se identifican las Roi’s que son las regiones de interés propuestas por la literatura para los pies diabéticos

```python
def IdentificacionROI(Imagen,h,w,areaRoi,CteProp=1,Tmin=0,Umbral=0):

    import cv2
    from numpy import mean,setdiff1d
    hsv = cv2.cvtColor(Imagen,cv2.COLOR_BGR2HSV)
    H = hsv[:,:,0]
    H=H[h-areaRoi:h+areaRoi+1,w-areaRoi:w+areaRoi+1]
    H=setdiff1d(H,0)
    H=mean(H)
    return (H-Umbral)*CteProp+Tmin
```

Run Cell | Run Above | Debug Cell

Procedemos a realizar el etiquetado de los datos

```python
# Conjuntos Entrenamiento, Validacion y Prueba por clase 70/15/15
# Creacion Etiquetas
YSanos = np.zeros((len(PSanos), 1))
YEnfermos = np.ones((len(PEnfermos),1))
# Concatenacion Etiquetas
PSanos=np.concatenate((PSanos,YSanos),axis=1)
PEnfermos=np.concatenate((PEnfermos,YEnfermos),axis=1)
# Conjuntos Lo mas balanceados que se puede
TrainingS, ValidS = model_selection.train_test_split(PSanos, test_size = int(0.3*len(PSanos)), train_size = int(0.7*len(PSanos)))
ValidationS, TestingS = model_selection.train_test_split(ValidS, test_size = int(0.5*len(ValidS)), train_size = int(0.5*len(ValidS)))

TrainingE, ValidE = model_selection.train_test_split(PEnfermos, test_size = int(0.3*len(PEnfermos)), train_size = int(0.7*len(PEnfermos)))
ValidationE, TestingE = model_selection.train_test_split(ValidE, test_size = int(0.5*len(ValidE)), train_size = int(0.5*len(ValidE)))

Training=np.concatenate((TrainingS,TrainingE),axis=0)
Testing=np.concatenate((TestingS,TestingE),axis=0)
Validation=np.concatenate((ValidationS,ValidationE),axis=0)
# Se borran los conjuntos sobrantes
del TrainingS,TrainingE
del TestingS,TestingE
del ValidationS,ValidationE
del ValidS,ValidE
# Etiquetas Categoricas
Y_Train = Training[:,np.size(Training,axis=1)-1]
Y_Valid = Validation[:,np.size(Training,axis=1)-1]
Y_Test = Testing[:,np.size(Training,axis=1)-1]
Y_Test = np.float64(Y_Test)
Y_Train_Dummies = pd.get_dummies(Y_Train)
Y_Valid_Dummies = pd.get_dummies(Y_Valid)
# Matrices Ya acondicionadas
Train=Training[:,0:np.size(Training,axis=1)-1]
Valid=Validation[:,0:np.size(Training,axis=1)-1]
Test=Testing[:,0:np.size(Training,axis=1)-1]
```

Posteriormente realizamos el entrenamiento de los datos.

```
# Entrenamiento
Red.fit(Train,Y_Train_Dummies, epochs = 250,
                verbose = 1 , workers = 4 , use_multiprocessing=True,
                validation_data = (Valid,Y_Valid_Dummies))

Out_Prob = Red.predict(Test)
Out_Testing = Out_Prob.round()

Out_Testing = pd.DataFrame(Out_Testing)
Out_Testing = Out_Testing.values.argmax(1)

MatrC=confusion_matrix(Out_Testing,Y_Test)
print(f1_score(Out_Testing,Y_Test),
      '\n',
      MatrC)
# Se obtienen los pesos
class_weights = {0:len(PSanos)/(len(PSanos) + len(PEnfermos)),
                 1:len(PEnfermos)/(len(PSanos) + len(PEnfermos))}


# Datos globales (C-means)
Datos=np.concatenate((PSanos,PEnfermos),axis=0)
Labels=np.concatenate((np.zeros((len(PSanos), 1)), np.ones((len(PEnfermos),1))),axis=0)
Datos=np.concatenate((Datos,Labels),axis=1)
```

Una ves realizado, obtenemos un F1 sore del 94%, siendo que de de los 19 pacientes enfermos se clasificaron a 17 correctamente.