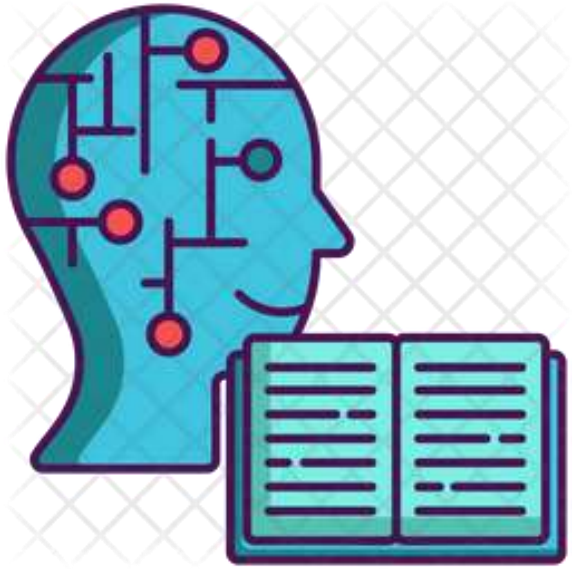
ProyectAL

Daniel Eduardo Ortiz Celis Cod: 2171469 – H2

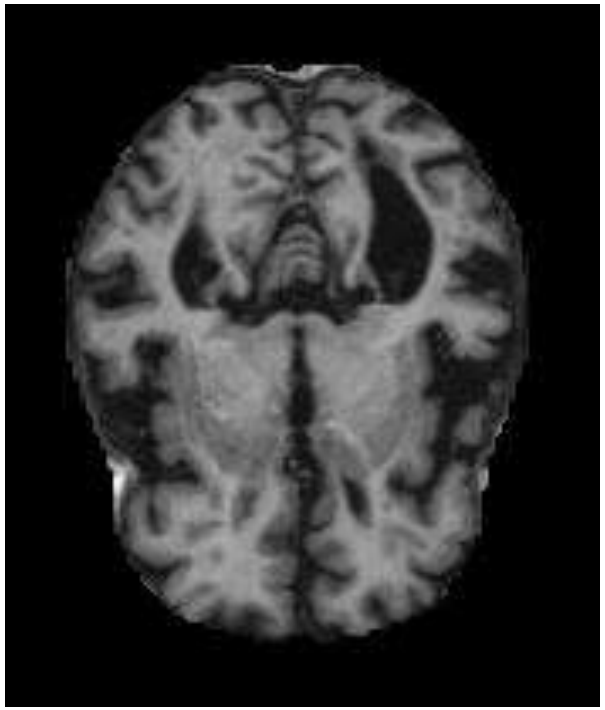
Camilo Eduardo González Guerrero Cod: 2180065 – H1

Introducción del proyecto

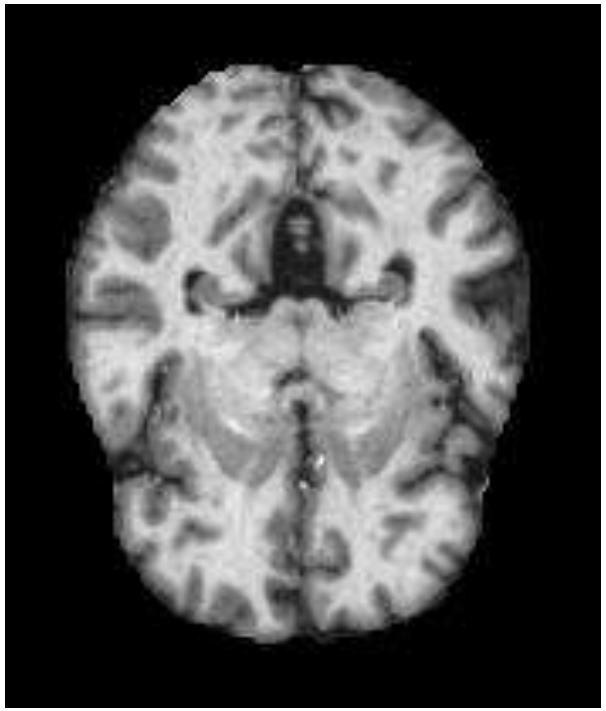


Dataset

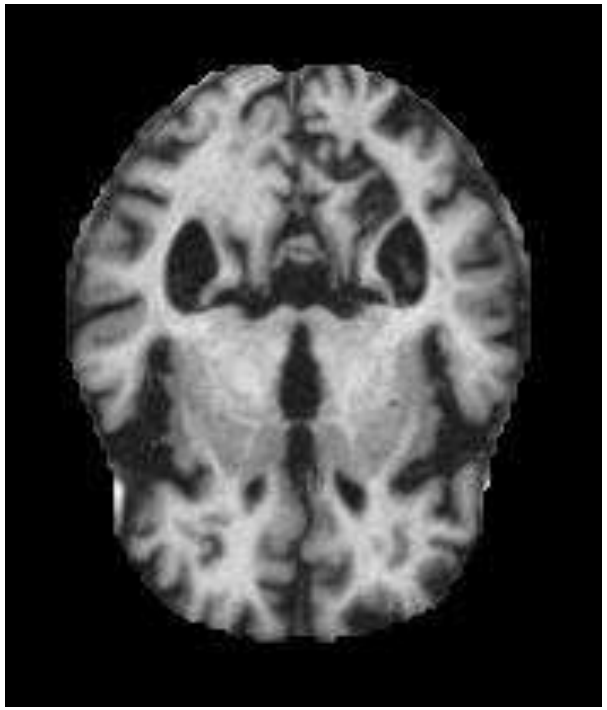
NonDemented



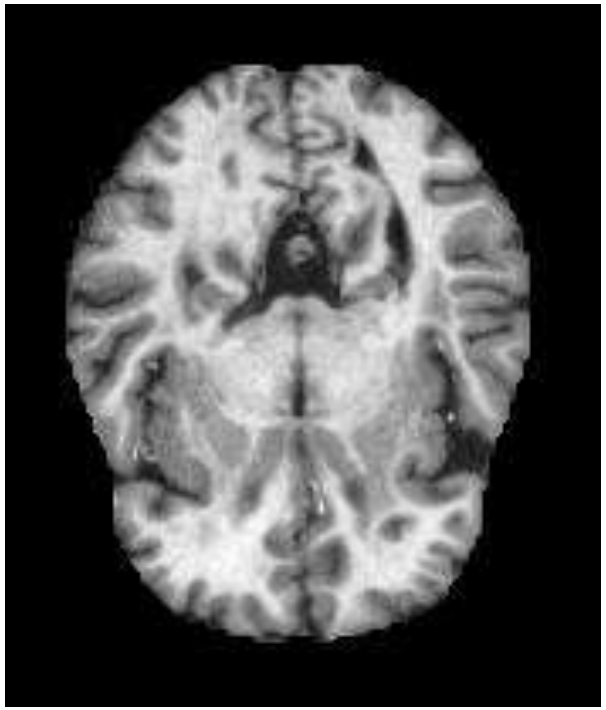
VeryMildDemented



MildDemented



ModerateDemented

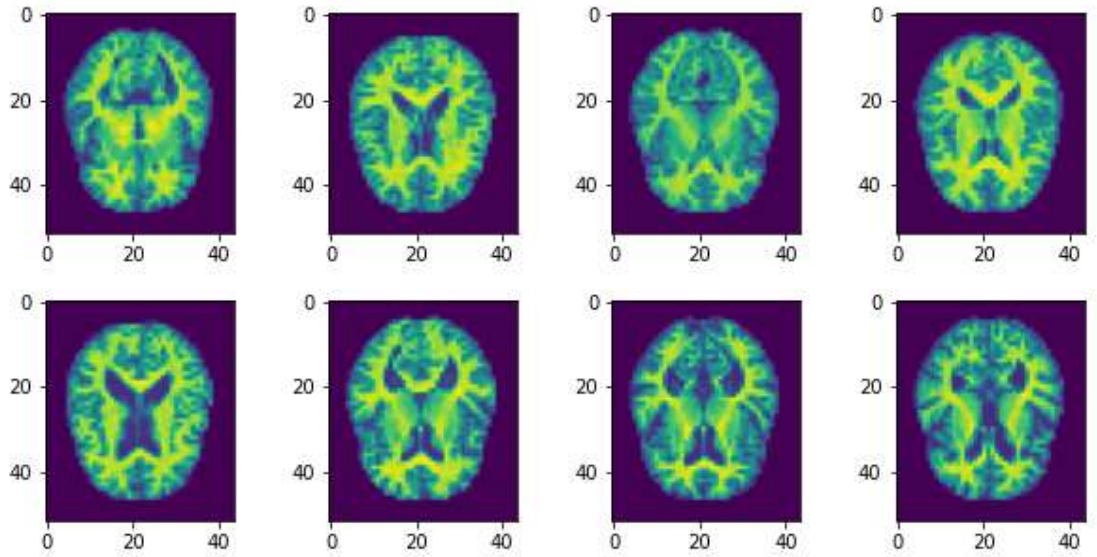


https://www.kaggle.com/tourist55/alzheimers-dataset-4-class-of-images

```
#@title Cargamos las imagenes de los NonDemented
NonDemented = "/content/gdrive/MyDrive/Proyecto IA/Dataset_/data/Alzheimer_s Dataset/train/NonDemented"
images_NonDemented = []
image_size_a = 44
image_size_b = 52

for image in os.listdir(NonDemented):
  image = cv2.imread(os.path.join(NonDemented, image))
  image_gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
  image_gray_resize = cv2.resize(image_gray, (image_size_a, image_size_b))
  images_NonDemented.append([image_gray_resize])
```

Imágenes NonDemented

Mostrar código



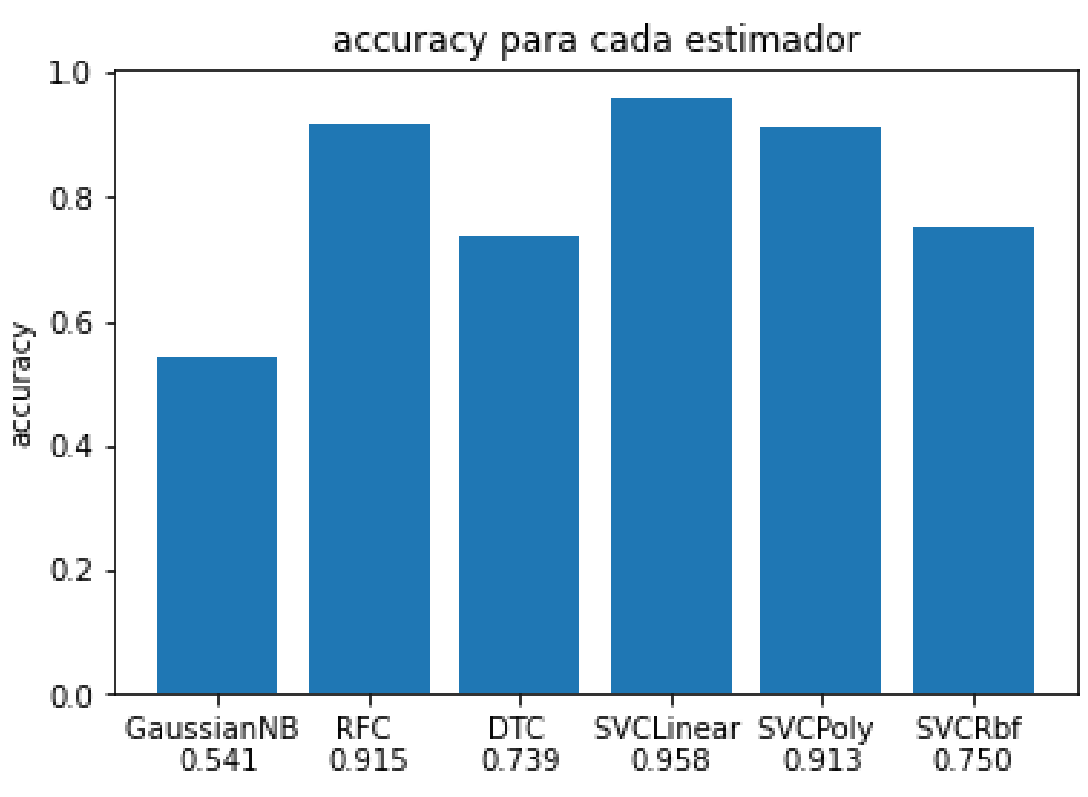
Escala grises

Proceso

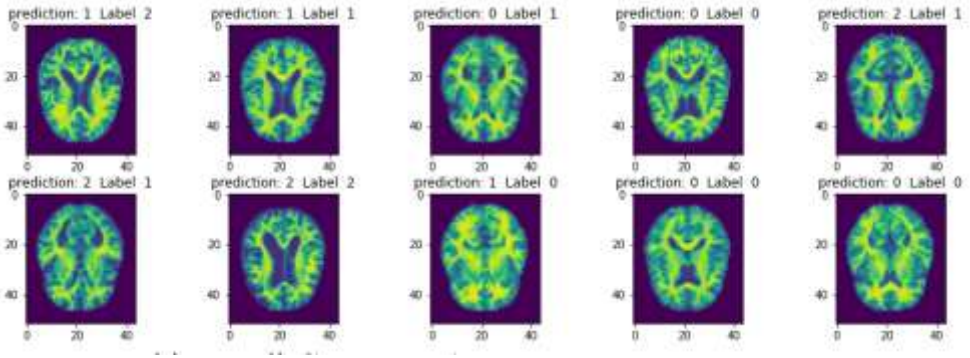
```python
#@title Creamos el dataset y se redimensiona
images_NonDemented_resize = np.resize(images_NonDemented, (2560, 2288))
images_MildDemented_resize = np.resize(images_MildDemented, (717, 2288))
images_ModerateDemented_resize = np.resize(images_ModerateDemented, (52, 2288))
images_VeryMildDemented_resize = np.resize(images_VeryMildDemented, (1792, 2288))

#asignamos las etiquetas y concatenamos los dataset
images_NonDemented_resize = pd.DataFrame(images_NonDemented_resize)
images_MildDemented_resize = pd.DataFrame(images_MildDemented_resize)
images_ModerateDemented_resize = pd.DataFrame(images_ModerateDemented_resize)
images_VeryMildDemented_resize = pd.DataFrame(images_VeryMildDemented_resize)


# 0 -> NonDemented
# 1 -> VeryMildDemented
# 2 -> MildDemented
# 3 -> ModerateDemented

images_NonDemented_resize['label'] = 0
images_VeryMildDemented_resize['label'] = 1
images_MildDemented_resize['label'] = 2
images_ModerateDemented_resize['label'] = 3

Datas = images_NonDemented_resize
Datas = Datas.append(images_VeryMildDemented_resize, ignore_index = True)
Datas = Datas.append(images_MildDemented_resize, ignore_index = True)
Datas = Datas.append(images_ModerateDemented_resize, ignore_index = True)

#Se revuelven los datos
Datas = np.random.permutation(np.random.permutation(np.random.permutation(Datas.values)))

print(Datas.shape)
pd.DataFrame(Datas).tail(25)
```

	0	1	2	3	4	5	6	7	8	9	...	2279	2280	2281	2282	2283	2284	2285	2286	2287	2288
5096	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	2
5097	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	1
5098	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
5099	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
5100	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	1
5101	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	1
5102	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
5103	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	1
5104	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	2
5105	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
5106	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
5107	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
5108	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	1
5109	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	1
5110	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
5111	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
5112	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	1
5113	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	1
5114	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	1
5115	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	2
5116	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
5117	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	3
5118	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
5119	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	1
5120	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0

Metodos de clasificacion Usando ML (Resultados)




```
GaussianNB:  0.5415 STD 0.4983
Random Forest:  0.9151 STD 0.2787
Decision Tree:  0.7385 STD 0.4394
SVCLinear:0.9580 STD 0.2005
SVCPoly:0.9132 STD 0.2816
SVCRbf:0.7502 STD 0.4329
```



Modelo Red neuronal (Entrenamiento y resultados)

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 flatten_2 (Flatten)         (None, 2288)              0

 dense_11 (Dense)            (None, 1144)              2618616

 dense_12 (Dense)            (None, 572)               654940

 dense_13 (Dense)            (None, 286)               163878

 dense_14 (Dense)            (None, 143)               41041

 dense_15 (Dense)            (None, 4)                 576

=================================================================
Total params: 3,479,051
Trainable params: 3,479,051
Non-trainable params: 0
_________________________________________________________________
```

```
Epoch 1/15
128/128 [==============================] - 1s 3ms/step - loss: 1.0373 - accuracy: 0.5022
Epoch 2/15
128/128 [==============================] - 0s 3ms/step - loss: 0.9070 - accuracy: 0.5525
Epoch 3/15
128/128 [==============================] - 0s 3ms/step - loss: 0.8666 - accuracy: 0.5811
Epoch 4/15
128/128 [==============================] - 0s 3ms/step - loss: 0.8293 - accuracy: 0.6028
Epoch 5/15
128/128 [==============================] - 0s 3ms/step - loss: 0.8165 - accuracy: 0.6140
Epoch 6/15
128/128 [==============================] - 0s 3ms/step - loss: 0.7484 - accuracy: 0.6565
Epoch 7/15
128/128 [==============================] - 0s 3ms/step - loss: 0.7125 - accuracy: 0.6775
Epoch 8/15
128/128 [==============================] - 0s 3ms/step - loss: 0.6407 - accuracy: 0.7007
Epoch 9/15
128/128 [==============================] - 0s 3ms/step - loss: 0.6093 - accuracy: 0.7351
Epoch 10/15
128/128 [==============================] - 0s 3ms/step - loss: 0.6233 - accuracy: 0.7334
Epoch 11/15
128/128 [==============================] - 0s 3ms/step - loss: 0.5406 - accuracy: 0.7690
Epoch 12/15
128/128 [==============================] - 0s 3ms/step - loss: 0.5172 - accuracy: 0.7820
Epoch 13/15
128/128 [==============================] - 0s 3ms/step - loss: 0.5007 - accuracy: 0.7886
Epoch 14/15
128/128 [==============================] - 0s 3ms/step - loss: 0.4007 - accuracy: 0.8342
Epoch 15/15
128/128 [==============================] - 0s 3ms/step - loss: 0.4246 - accuracy: 0.8281
<keras.callbacks.History at 0x7f4862761a90>
```

```
33/33 [==============================] - 0s 3ms/step - loss: 0.7612 - accuracy: 0.7200
Test accuracy: 0.7200000286102295
```

Modelo Red convolucional (Entrenamiento y resultados)

```
Model: "sequential_6"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_8 (Conv2D)           (None, 44, 52, 32)        320

 conv2d_9 (Conv2D)           (None, 44, 52, 32)        9248

 max_pooling2d_4 (MaxPooling  (None, 22, 26, 32)       0
 2D)

 flatten_6 (Flatten)         (None, 18304)             0

 dense_34 (Dense)            (None, 2288)              41881840

 dense_35 (Dense)            (None, 1144)              2618616

 dense_36 (Dense)            (None, 572)               654940

 dense_37 (Dense)            (None, 286)               163878

 dense_38 (Dense)            (None, 143)               41041

 dense_39 (Dense)            (None, 4)                 576

=================================================================
Total params: 45,370,459
Trainable params: 45,370,459
Non-trainable params: 0
```

```
Epoch 1/5
128/128 [==============================] - 3s 15ms/step - loss: 1.0581 - accuracy: 0.4983
Epoch 2/5
128/128 [==============================] - 2s 15ms/step - loss: 0.9123 - accuracy: 0.5510
Epoch 3/5
128/128 [==============================] - 2s 15ms/step - loss: 0.7740 - accuracy: 0.6292
Epoch 4/5
128/128 [==============================] - 2s 15ms/step - loss: 0.5701 - accuracy: 0.7468
Epoch 5/5
128/128 [==============================] - 2s 15ms/step - loss: 0.3125 - accuracy: 0.8762
<keras.callbacks.History at 0x7f48f0156e50>
```

```
33/33 [==============================] - 0s 5ms/step - loss: 0.3399 - accuracy: 0.8800
Test accuracy: 0.8799999952316284
```