REDES NEURONALES ARTIFICIALES PARA PREDECIR RIESGO CARDIOVASCULAR

Presentado por:

JUAN CAMILO SANCHEZ FERNANDEZ

TANIA JULIET HURTADO RAMÍREZ

Presentado a:

Ing ELIAS BUITRAGO BOLIVAR

Universidad ECCI

Ingeniería en Sistemas

Seminario Big Data & Gerencia de datos

Bogotá

2024

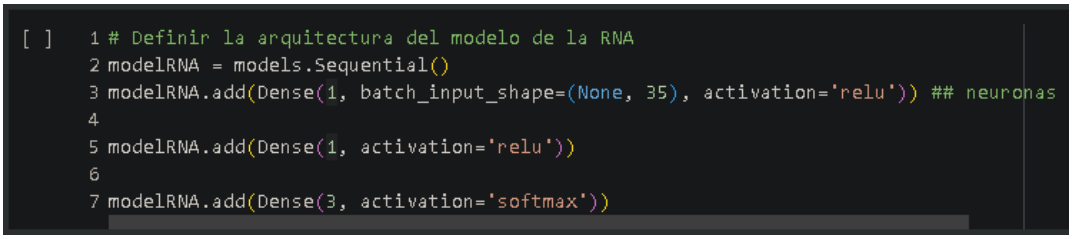
Introducción

En este trabajo exploraremos a fondo elementos clave como funciones de activación, estructura de capas, cantidad óptima de neuronas, diseño y arquitectura de la red, el concepto de Epoch para entrenamiento eficiente, la plataforma Google Colab para colaboración en tiempo real y diseño avanzado de redes neuronales. Además, nos enfocaremos en evaluar la eficacia del modelo utilizando métricas como el Área Bajo la Curva (AUC).

Optimización Modelo Clasificación Redes Neuronales

Pruebas

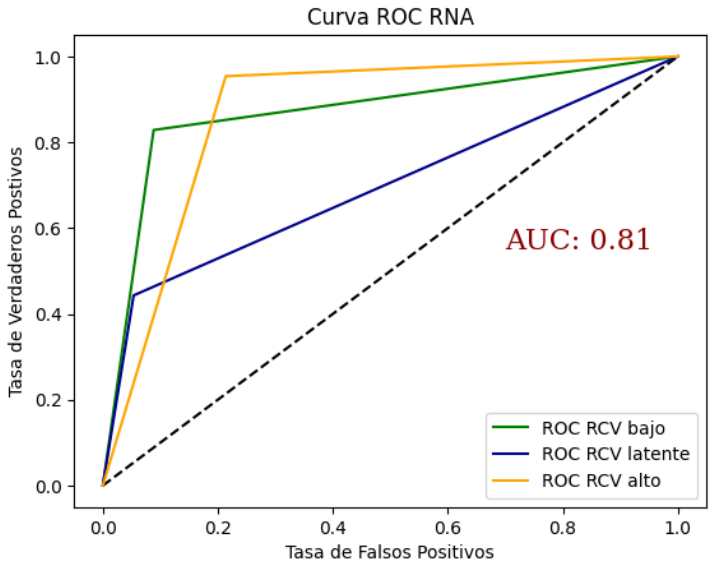
1. Se lleva a cabo una prueba del modelo utilizando la siguiente arquitectura:



```
[ ]  1 # Definir la arquitectura del modelo de la RNA
     2 modelRNA = models.Sequential()
     3 modelRNA.add(Dense(1, batch_input_shape=(None, 35), activation='relu')) ## neuronas
     4
     5 modelRNA.add(Dense(1, activation='relu'))
     6
     7 modelRNA.add(Dense(3, activation='softmax'))
```

```
1 training_log = modelRNA.fit(X_train,
2                   Y_train,
3                   epochs=150,
4                   batch_size=32,
5                   validation_data=(X_valid, Y_valid),
6                   verbose=1)
```

```
53/53 [==============================] - 0s 3ms/step - loss: 0.3738 - accuracy: 0.6940 - val_loss: 0.3792 - val_accu
Epoch 123/150
53/53 [==============================] - 0s 3ms/step - loss: 0.3736 - accuracy: 0.6935 - val_loss: 0.3790 - val_accu
Epoch 124/150
53/53 [==============================] - 0s 4ms/step - loss: 0.3735 - accuracy: 0.6946 - val_loss: 0.3792 - val_accu
Epoch 125/150
53/53 [==============================] - 0s 3ms/step - loss: 0.3735 - accuracy: 0.6935 - val_loss: 0.3794 - val_accu
Epoch 126/150
53/53 [==============================] - 0s 3ms/step - loss: 0.3734 - accuracy: 0.6929 - val_loss: 0.3790 - val_accu
Epoch 127/150
53/53 [==============================] - 0s 3ms/step - loss: 0.3733 - accuracy: 0.6935 - val_loss: 0.3787 - val_accu
Epoch 128/150
53/53 [==============================] - 0s 3ms/step - loss: 0.3731 - accuracy: 0.6929 - val_loss: 0.3785 - val_accu
Epoch 129/150
53/53 [==============================] - 0s 4ms/step - loss: 0.3730 - accuracy: 0.6940 - val_loss: 0.3780 - val_accu
Epoch 130/150
53/53 [==============================] - 0s 4ms/step - loss: 0.3728 - accuracy: 0.6935 - val_loss: 0.3781 - val_accu
Epoch 131/150
53/53 [==============================] - 0s 3ms/step - loss: 0.3726 - accuracy: 0.6935 - val_loss: 0.3778 - val_accu
```

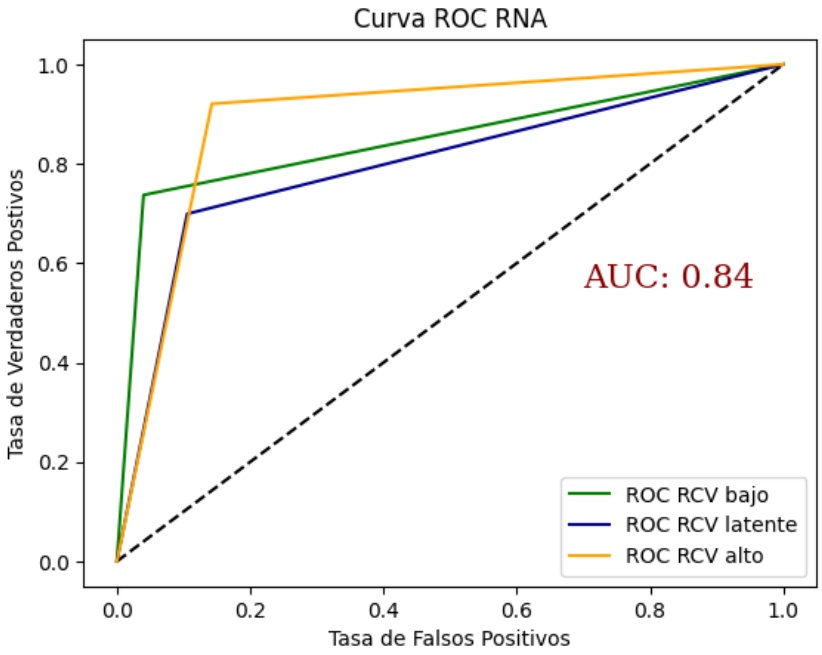


2. Para esta decidimos aumentar el número de capas ocultas a 5 con una cantidad variable de neuronas en cada una, como se muestra en la imagen a continuación:

```
[53]  1 # Definir la arquitectura del modelo de la RNA
      2 modelRNA = models.Sequential()
      3 modelRNA.add(Dense(1, batch_input_shape=(None, 35), activation='relu')) ## neuronas en
      4
      5 modelRNA.add(Dense(64, activation='relu'))
      6 modelRNA.add(Dense(32, activation='relu'))
      7 modelRNA.add(Dense(16, activation='relu'))
      8
      9 modelRNA.add(Dense(3, activation='softmax'))
```

```
[56]  1 training_log = modelRNA.fit(X_train,
      2                   Y_train,
      3                   epochs=150,
      4                   batch_size=32,
      5                   validation_data=(X_valid, Y_valid),
      6                   verbose=1)

53/53 [==============================] - 0s 3ms/step - loss: 0.2890 - accuracy: 0.8280 - val_loss: 0.2585 - val_accu
Epoch 123/150
53/53 [==============================] - 0s 3ms/step - loss: 0.2913 - accuracy: 0.8208 - val_loss: 0.2591 - val_accu
Epoch 124/150
53/53 [==============================] - 0s 3ms/step - loss: 0.2911 - accuracy: 0.8190 - val_loss: 0.2600 - val_accu
Epoch 125/150
53/53 [==============================] - 0s 3ms/step - loss: 0.2910 - accuracy: 0.8208 - val_loss: 0.2583 - val_accu
Epoch 126/150
53/53 [==============================] - 0s 3ms/step - loss: 0.2921 - accuracy: 0.8179 - val_loss: 0.2566 - val_accu
Epoch 127/150
53/53 [==============================] - 0s 4ms/step - loss: 0.2906 - accuracy: 0.8232 - val_loss: 0.2571 - val_accu
Epoch 128/150
53/53 [==============================] - 0s 3ms/step - loss: 0.2911 - accuracy: 0.8232 - val_loss: 0.2572 - val_accu
```



3. Después de ver el rendimiento del modelo con la arquitectura anterior, decidimos incrementar el número de capas ocultas a 7.

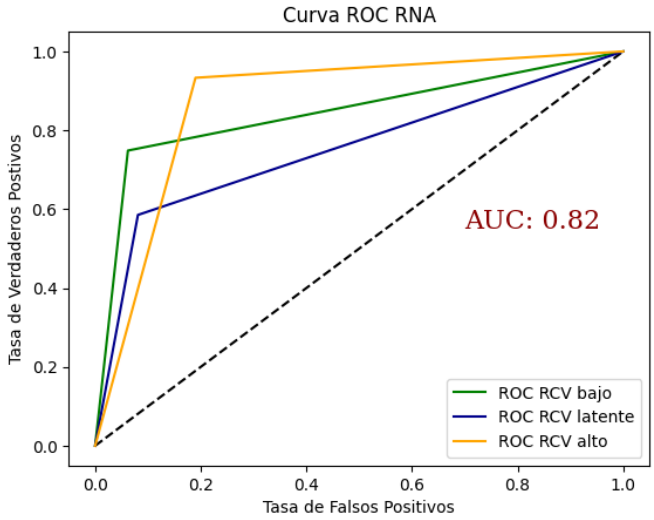
```
# Definir la arquitectura del modelo de la RNA
modelRNA = models.Sequential()
modelRNA.add(Dense(1, batch_input_shape=(None, 35), activation='relu')) ## neuronas

modelRNA.add(Dense(128, activation='relu'))
modelRNA.add(Dense(64, activation='relu'))
modelRNA.add(Dense(32, activation='relu'))
modelRNA.add(Dense(16, activation='relu'))
modelRNA.add(Dense(8, activation='relu'))

modelRNA.add(Dense(3, activation='softmax'))
```

```
[77]  training_log = modelRNA.fit(X_train,
                    Y_train,
                    epochs=150,
                    batch_size=32,
                    validation_data=(X_valid, Y_valid),
                    verbose=1)

53/53 [==============================] - 0s 6ms/step - loss: 0.3147 - accuracy: 0.8161 - val_loss: 0.3114 - val_accu
Epoch 123/150
53/53 [==============================] - 0s 6ms/step - loss: 0.3150 - accuracy: 0.8125 - val_loss: 0.3068 - val_accu
Epoch 124/150
53/53 [==============================] - 0s 6ms/step - loss: 0.3144 - accuracy: 0.8107 - val_loss: 0.3074 - val_accu
Epoch 125/150
53/53 [==============================] - 0s 6ms/step - loss: 0.3145 - accuracy: 0.8167 - val_loss: 0.3133 - val_accu
Epoch 126/150
53/53 [==============================] - 0s 6ms/step - loss: 0.3150 - accuracy: 0.8167 - val_loss: 0.3093 - val_accu
Epoch 127/150
53/53 [==============================] - 0s 6ms/step - loss: 0.3149 - accuracy: 0.8107 - val_loss: 0.3112 - val_accu
Epoch 128/150
53/53 [==============================] - 0s 6ms/step - loss: 0.3155 - accuracy: 0.8131 - val_loss: 0.3086 - val_accu
Epoch 129/150
53/53 [==============================] - 0s 6ms/step - loss: 0.3131 - accuracy: 0.8143 - val_loss: 0.3100 - val_accu
Epoch 130/150
53/53 [==============================] - 0s 7ms/step - loss: 0.3143 - accuracy: 0.8161 - val_loss: 0.3049 - val_accu
```



4. Consideramos que la cantidad de capas ocultas en la configuración anterior no era eficiente ya que bajo el AUC a 0.82.

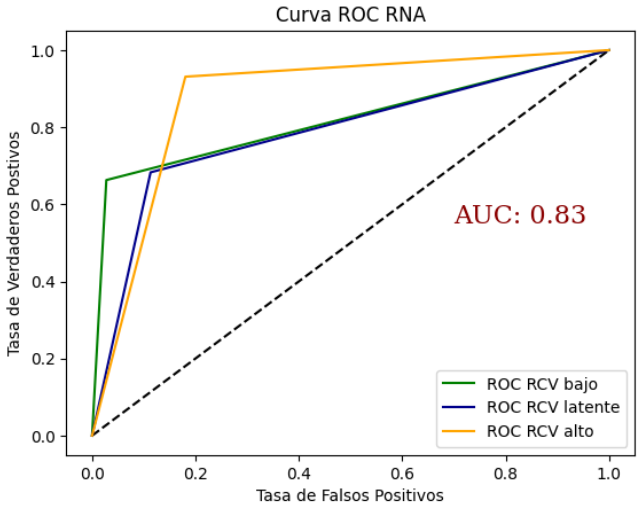
```python
# Definir la arquitectura del modelo de la RNA
modelRNA = models.Sequential()
modelRNA.add(Dense(1, batch_input_shape=(None, 35), activation='relu')) ## neuronas

modelRNA.add(Dense(32, activation='relu'))
modelRNA.add(Dense(32, activation='relu'))
modelRNA.add(Dense(16, activation='relu'))
modelRNA.add(Dense(16, activation='relu'))
modelRNA.add(Dense(8, activation='relu'))
modelRNA.add(Dense(8, activation='relu'))

modelRNA.add(Dense(3, activation='softmax'))
```

```python
training_log = modelRNA.fit(X_train,
                    Y_train,
                    epochs=100,
                    batch_size=32,
                    validation_data=(X_valid, Y_valid),
                    verbose=1)
```

```
53/53 [==============================] - 0s 3ms/step - loss: 0.2925 - accuracy: 0.8208 - val_loss: 0.3522 - val_accu
Epoch 45/150
53/53 [==============================] - 0s 4ms/step - loss: 0.2931 - accuracy: 0.8244 - val_loss: 0.3408 - val_accu
Epoch 46/150
53/53 [==============================] - 0s 3ms/step - loss: 0.2916 - accuracy: 0.8244 - val_loss: 0.3440 - val_accu
Epoch 47/150
53/53 [==============================] - 0s 3ms/step - loss: 0.2931 - accuracy: 0.8274 - val_loss: 0.3420 - val_accu
Epoch 48/150
53/53 [==============================] - 0s 5ms/step - loss: 0.2936 - accuracy: 0.8214 - val_loss: 0.3399 - val_accu
Epoch 49/150
53/53 [==============================] - 0s 5ms/step - loss: 0.2927 - accuracy: 0.8232 - val_loss: 0.3461 - val_accu
Epoch 50/150
53/53 [==============================] - 0s 6ms/step - loss: 0.2931 - accuracy: 0.8179 - val_loss: 0.3522 - val_accu
Epoch 51/150
53/53 [==============================] - 0s 5ms/step - loss: 0.2924 - accuracy: 0.8339 - val_loss: 0.3601 - val_accu
Epoch 52/150
53/53 [==============================] - 0s 5ms/step - loss: 0.2932 - accuracy: 0.8220 - val_loss: 0.3433 - val_accu
Epoch 53/150
```



5. Decidimos mantener la arquitectura, pero cambiamos la activación.

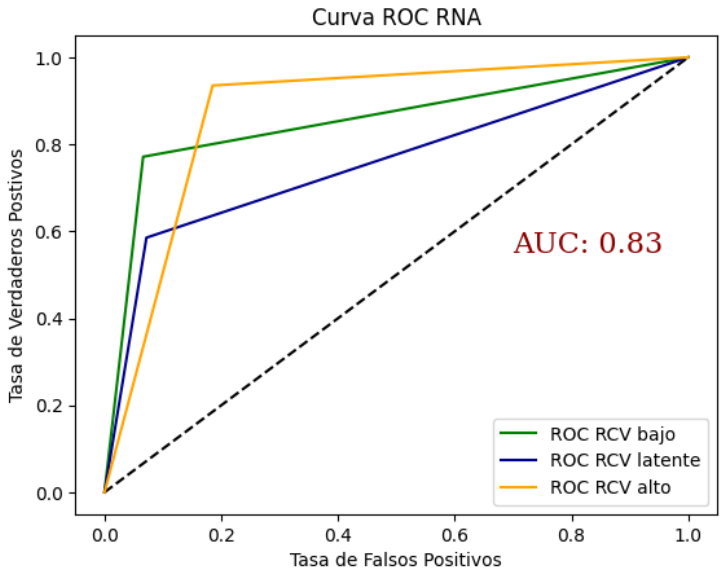
```
# Definir la arquitectura del modelo de la RNA
modelRNA = models.Sequential()
modelRNA.add(Dense(1, batch_input_shape=(None, 35), activation='relu')) ## neuron

modelRNA.add(Dense(32, activation='tanh'))
modelRNA.add(Dense(32, activation='tanh'))
modelRNA.add(Dense(16, activation='tanh'))
modelRNA.add(Dense(16, activation='tanh'))
modelRNA.add(Dense(8, activation='tanh'))
modelRNA.add(Dense(8, activation='tanh'))

modelRNA.add(Dense(3, activation='softmax'))
```

```
[119] training_log = modelRNA.fit(X_train,
                                  Y_train,
                                  epochs=100,
                                  batch_size=32,
                                  validation_data=(X_valid, Y_valid),
                                  verbose=1)
```

```
53/53 [==============================] - 0s 7ms/step - loss: 0.3098 - accuracy: 0.8113 - val_loss: 0.3343 - val_accu
Epoch 73/100
53/53 [==============================] - 0s 3ms/step - loss: 0.3103 - accuracy: 0.8190 - val_loss: 0.3333 - val_accu
Epoch 74/100
53/53 [==============================] - 0s 3ms/step - loss: 0.3098 - accuracy: 0.8155 - val_loss: 0.3353 - val_accu
Epoch 75/100
53/53 [==============================] - 0s 4ms/step - loss: 0.3104 - accuracy: 0.8161 - val_loss: 0.3354 - val_accu
Epoch 76/100
53/53 [==============================] - 0s 4ms/step - loss: 0.3090 - accuracy: 0.8179 - val_loss: 0.3341 - val_accu
Epoch 77/100
53/53 [==============================] - 0s 4ms/step - loss: 0.3087 - accuracy: 0.8190 - val_loss: 0.3327 - val_accu
Epoch 78/100
53/53 [==============================] - 0s 3ms/step - loss: 0.3085 - accuracy: 0.8149 - val_loss: 0.3358 - val_accu
Epoch 79/100
53/53 [==============================] - 0s 3ms/step - loss: 0.3085 - accuracy: 0.8131 - val_loss: 0.3341 - val_accu
Epoch 80/100
53/53 [==============================] - 0s 3ms/step - loss: 0.3084 - accuracy: 0.8131 - val_loss: 0.3339 - val_accu
Epoch 81/100
53/53 [==============================] - 0s 4ms/step - loss: 0.3059 - accuracy: 0.8208 - val_loss: 0.3469 - val_accu
```



6. Se mantuvo el mismo AUC, vamos a eliminar unas capas a ver cómo se comporta.

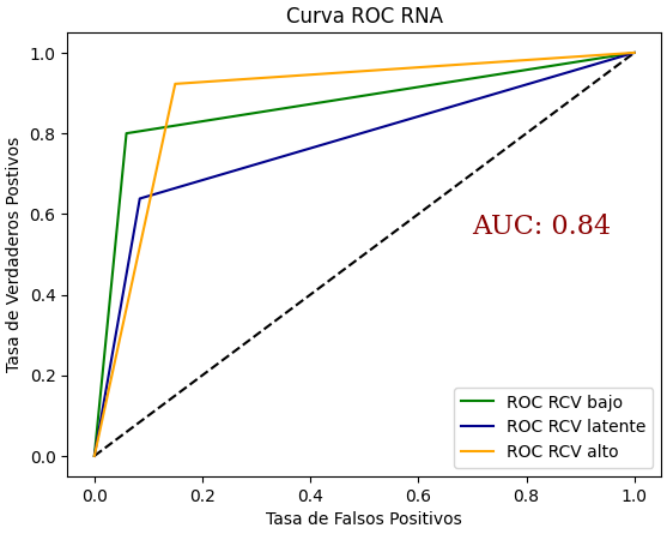
```
# Definir la arquitectura del modelo de la RNA
modelRNA = models.Sequential()
modelRNA.add(Dense(1, batch_input_shape=(None, 35), activation='relu')) ## neur

modelRNA.add(Dense(32, activation='tanh'))
modelRNA.add(Dense(16, activation='tanh'))
modelRNA.add(Dense(8, activation='tanh'))

modelRNA.add(Dense(3, activation='softmax'))
```

```
training_log = modelRNA.fit(X_train,
                  Y_train,
                  epochs=100,
                  batch_size=32,
                  validation_data=(X_valid, Y_valid),
                  verbose=1)
```

```
53/53 [==============================] - 0s 4ms/step - loss: 0.3059 - accuracy: 0.8054 - val_loss: 0.3342 - val_accu
Epoch 73/100
53/53 [==============================] - 0s 6ms/step - loss: 0.3053 - accuracy: 0.8101 - val_loss: 0.3336 - val_accu
Epoch 74/100
53/53 [==============================] - 0s 6ms/step - loss: 0.3028 - accuracy: 0.8089 - val_loss: 0.3346 - val_accu
Epoch 75/100
53/53 [==============================] - 0s 6ms/step - loss: 0.3019 - accuracy: 0.8113 - val_loss: 0.3294 - val_accu
Epoch 76/100
53/53 [==============================] - 0s 6ms/step - loss: 0.3009 - accuracy: 0.8125 - val_loss: 0.3279 - val_accu
Epoch 77/100
53/53 [==============================] - 0s 5ms/step - loss: 0.3000 - accuracy: 0.8202 - val_loss: 0.3261 - val_accu
Epoch 78/100
53/53 [==============================] - 0s 6ms/step - loss: 0.2977 - accuracy: 0.8179 - val_loss: 0.3279 - val_accu
Epoch 79/100
53/53 [==============================] - 0s 5ms/step - loss: 0.2979 - accuracy: 0.8155 - val_loss: 0.3250 - val_accu
Epoch 80/100
53/53 [==============================] - 0s 4ms/step - loss: 0.2973 - accuracy: 0.8196 - val_loss: 0.3254 - val_accu
Epoch 81/100
53/53 [==============================] - 0s 5ms/step - loss: 0.2964 - accuracy: 0.8214 - val_loss: 0.3316 - val_accu
```



7. Mejoró un 0.01, vamos agregarle unas capas.

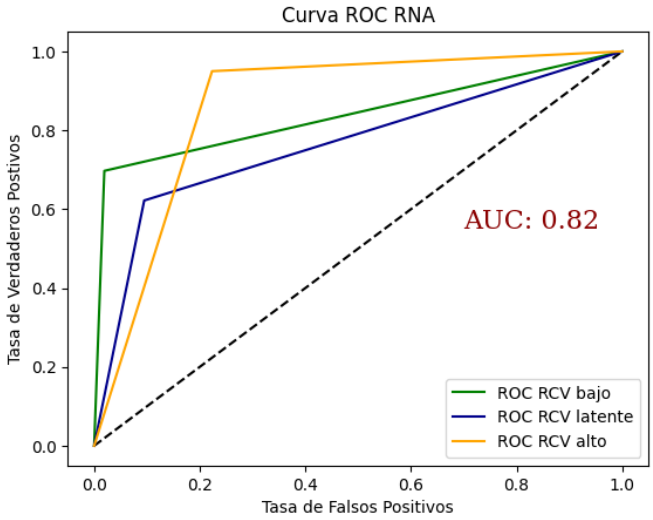
```python
# Definir la arquitectura del modelo de la RNA
modelRNA = models.Sequential()
modelRNA.add(Dense(1, batch_input_shape=(None, 35), activation='relu')) ## neuron

modelRNA.add(Dense(128, activation='tanh'))
modelRNA.add(Dense(64, activation='tanh'))
modelRNA.add(Dense(32, activation='tanh'))
modelRNA.add(Dense(16, activation='tanh'))
modelRNA.add(Dense(8, activation='tanh'))

modelRNA.add(Dense(3, activation='softmax'))
```

```python
training_log = modelRNA.fit(X_train,
                  Y_train,
                  epochs=100,
                  batch_size=32,
                  validation_data=(X_valid, Y_valid),
                  verbose=1)
```

```
53/53 [==============================] - 0s 3ms/step - loss: 0.2886 - accuracy: 0.8280 - val_loss: 0.3570 - val_accu
Epoch 65/100
53/53 [==============================] - 0s 5ms/step - loss: 0.2902 - accuracy: 0.8220 - val_loss: 0.3594 - val_accu
Epoch 66/100
53/53 [==============================] - 0s 3ms/step - loss: 0.2903 - accuracy: 0.8238 - val_loss: 0.3604 - val_accu
Epoch 67/100
53/53 [==============================] - 0s 4ms/step - loss: 0.2892 - accuracy: 0.8190 - val_loss: 0.3582 - val_accu
Epoch 68/100
53/53 [==============================] - 0s 3ms/step - loss: 0.2883 - accuracy: 0.8268 - val_loss: 0.3593 - val_accu
Epoch 69/100
53/53 [==============================] - 0s 4ms/step - loss: 0.2890 - accuracy: 0.8274 - val_loss: 0.3532 - val_accu
Epoch 70/100
53/53 [==============================] - 0s 4ms/step - loss: 0.2880 - accuracy: 0.8244 - val_loss: 0.3576 - val_accu
Epoch 71/100
53/53 [==============================] - 0s 4ms/step - loss: 0.2915 - accuracy: 0.8202 - val_loss: 0.3601 - val_accu
```



8. La arquitectura anterior no fue muy optima, por lo tanto, decidimos dejar menor número de capas.

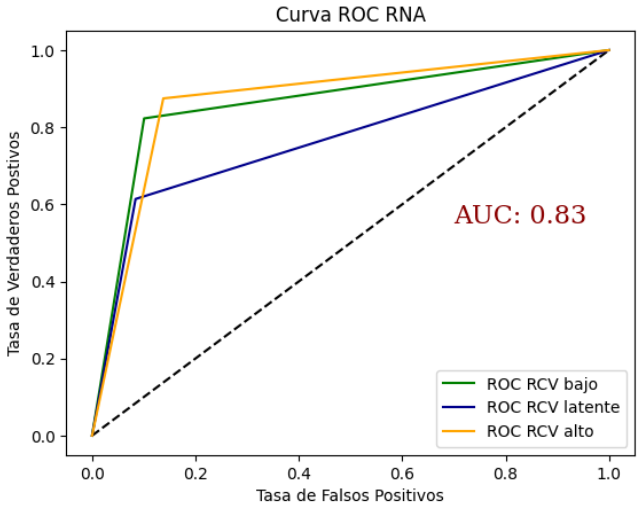
```
# Definir la arquitectura del modelo de la RNA
modelRNA = models.Sequential()
modelRNA.add(Dense(1, batch_input_shape=(None, 35), activation='relu')) ## neuro

modelRNA.add(Dense(16, activation='tanh'))
modelRNA.add(Dense(8, activation='tanh'))

modelRNA.add(Dense(3, activation='softmax'))
```

```
[203]  training_log = modelRNA.fit(X_train,
                    Y_train,
                    epochs=100,
                    batch_size=32,
                    validation_data=(X_valid, Y_valid),
                    verbose=1)
```

```
53/53 [==============================] - 0s 5ms/step - loss: 0.3331 - accuracy: 0.7869 - val_loss: 0.3435 - val_accu
Epoch 73/100
53/53 [==============================] - 0s 5ms/step - loss: 0.3327 - accuracy: 0.7857 - val_loss: 0.3427 - val_accu
Epoch 74/100
53/53 [==============================] - 0s 5ms/step - loss: 0.3324 - accuracy: 0.7887 - val_loss: 0.3406 - val_accu
Epoch 75/100
53/53 [==============================] - 0s 6ms/step - loss: 0.3317 - accuracy: 0.7881 - val_loss: 0.3408 - val_accu
Epoch 76/100
53/53 [==============================] - 0s 5ms/step - loss: 0.3320 - accuracy: 0.7899 - val_loss: 0.3407 - val_accu
Epoch 77/100
53/53 [==============================] - 0s 5ms/step - loss: 0.3311 - accuracy: 0.7857 - val_loss: 0.3409 - val_accu
Epoch 78/100
53/53 [==============================] - 0s 5ms/step - loss: 0.3306 - accuracy: 0.7881 - val_loss: 0.3416 - val_accu
Epoch 79/100
53/53 [==============================] - 0s 6ms/step - loss: 0.3299 - accuracy: 0.7923 - val_loss: 0.3397 - val_accu
Epoch 80/100
53/53 [==============================] - 0s 5ms/step - loss: 0.3291 - accuracy: 0.7917 - val_loss: 0.3402 - val_accu
Epoch 81/100
53/53 [==============================] - 0s 5ms/step - loss: 0.3300 - accuracy: 0.7917 - val_loss: 0.3391 - val_accu
```



9. Cómo ultima prueba, decidimos dejar la siguiente arquitectura.

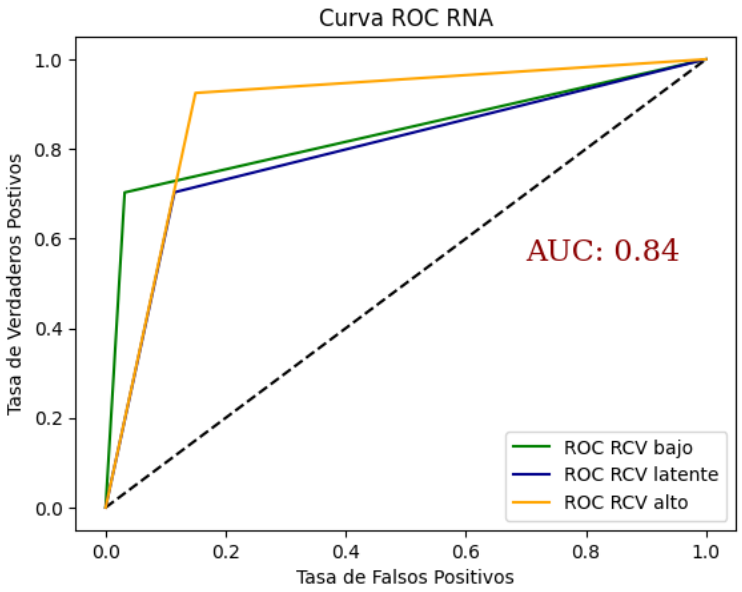
```
# Definir la arquitectura del modelo de la RNA
modelRNA = models.Sequential()
modelRNA.add(Dense(1, batch_input_shape=(None, 35), activation='relu')) ## neuro

modelRNA.add(Dense(20, activation='tanh'))
modelRNA.add(Dense(10, activation='tanh'))

modelRNA.add(Dense(3, activation='softmax'))
```

```
[224]  training_log = modelRNA.fit(X_train,
                                   Y_train,
                                   epochs=100,
                                   batch_size=32,
                                   validation_data=(X_valid, Y_valid),
                                   verbose=1)
```

```
53/53 [==============================] - 0s 4ms/step - loss: 0.2800 - accuracy: 0.8286 - val_loss: 0.3124 - val_acc
Epoch 73/100
53/53 [==============================] - 0s 3ms/step - loss: 0.2800 - accuracy: 0.8214 - val_loss: 0.3113 - val_acc
Epoch 74/100
53/53 [==============================] - 0s 3ms/step - loss: 0.2798 - accuracy: 0.8304 - val_loss: 0.3110 - val_acc
Epoch 75/100
53/53 [==============================] - 0s 3ms/step - loss: 0.2796 - accuracy: 0.8238 - val_loss: 0.3115 - val_acc
Epoch 76/100
53/53 [==============================] - 0s 4ms/step - loss: 0.2793 - accuracy: 0.8226 - val_loss: 0.3117 - val_acc
Epoch 77/100
53/53 [==============================] - 0s 3ms/step - loss: 0.2791 - accuracy: 0.8268 - val_loss: 0.3087 - val_acc
Epoch 78/100
53/53 [==============================] - 0s 3ms/step - loss: 0.2792 - accuracy: 0.8226 - val_loss: 0.3100 - val_acc
Epoch 79/100
53/53 [==============================] - 0s 4ms/step - loss: 0.2792 - accuracy: 0.8214 - val_loss: 0.3108 - val_acc
Epoch 80/100
53/53 [==============================] - 0s 3ms/step - loss: 0.2789 - accuracy: 0.8262 - val_loss: 0.3098 - val_acc
Epoch 81/100
53/53 [==============================] - 0s 3ms/step - loss: 0.2787 - accuracy: 0.8256 - val_loss: 0.3100 - val_acc
```



CONCLUSIONES

1. Se observó que incrementar el número de capas ocultas no siempre mejora el desempeño del modelo. En nuestro caso, aumentar el número de capas de 5 a 7 redujo el AUC a 0.82. Sin embargo, ajustar el número de capas y neuronas, así como las funciones de activación, permitió optimizar el rendimiento del modelo.
2. La optimización de hiperparámetros, como la elección de funciones de activación y la configuración de capas ocultas, es crucial para el desempeño del modelo. Cambios sutiles en estos parámetros pueden mejorar o empeorar significativamente los resultados, como se evidenció en nuestras pruebas iterativas.
3. El número de epochs, o épocas de entrenamiento, es un parámetro fundamental que afecta el rendimiento de las redes neuronales. A través de nuestras pruebas, se observó que un número adecuado de epochs permite al modelo aprender de manera efectiva sin caer en el sobreajuste.