CUARTO TALLER

Presentado por:

VALENTINA PIEDRAHITA GIL

JHOJAN STIVEN SANCHEZ PALADINES

Presentado a:

Ing ELIAS BUITRAGO BOLIVAR

Universidad ECCI

Ingeniería en Sistemas

Seminario Big Data & Gerencia de datos

Bogotá

2024

INTRODUCCIÓN

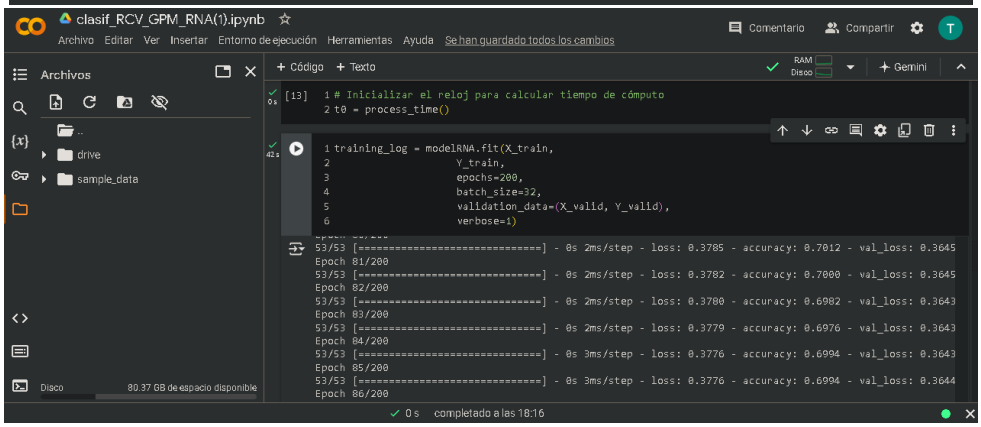
Nos enfocaremos en elementos fundamentales como el diseño de capas ocultas, la elección de funciones de activación adecuadas y la optimización del parámetro epochs para un entrenamiento eficiente y así ver con cuales la curva ROC funciona mejor.

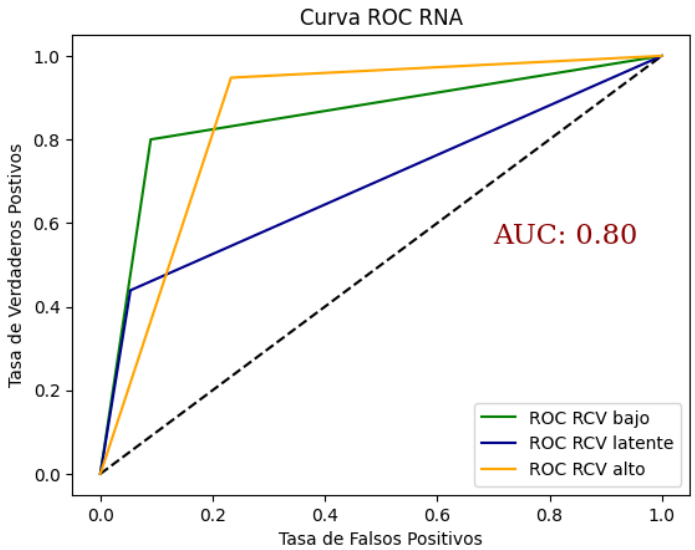
DESARROLLO

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

```
[11]  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 la capa de entra
      4 # modelRNA.add(Dense(1, activation='relu'))
      5 modelRNA.add(Dense(3, activation='softmax'))

[12]  1 # compile the keras (tensorflow) flow graph
      2 modelRNA.compile(optimizer=optimizers.RMSprop(learning_rate=0.001),
      3                  loss='binary_crossentropy',
      4                  metrics=['accuracy'])
```



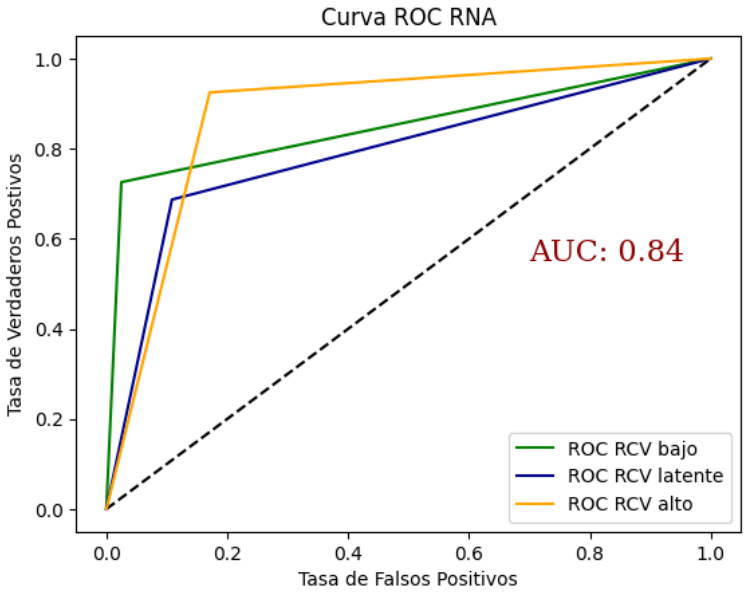


2. En la segunda prueba, decidimos aumentar el número de capas ocultas a 5 con una cantidad variable de neuronas en cada una y el epochs a 100:

```
[74]  1 # Definir la arquitectura del modelo de la RNA
      2 modelRNA = models.Sequential()
      3 modelRNA.add(Dense(3, batch_input_shape=(None, 35), activation='relu')) ## neuronas en la capa de entra
      4
      5 modelRNA.add(Dense(64, activation='relu'))
      6 modelRNA.add(Dense(32, activation='relu'))
      7 modelRNA.add(Dense(16, activation='relu'))
      8
      9
     10 modelRNA.add(Dense(3, activation='softmax'))
```

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

```
53/53 [==============================] - 0s 3ms/step - loss: 0.2561 - accuracy: 0.8423 - val_loss: 0.2582
Epoch 73/100
53/53 [==============================] - 0s 3ms/step - loss: 0.2556 - accuracy: 0.8446 - val_loss: 0.2567
Epoch 74/100
53/53 [==============================] - 0s 2ms/step - loss: 0.2565 - accuracy: 0.8488 - val_loss: 0.2576
Epoch 75/100
53/53 [==============================] - 0s 3ms/step - loss: 0.2559 - accuracy: 0.8429 - val_loss: 0.2612
Epoch 76/100
53/53 [==============================] - 0s 3ms/step - loss: 0.2536 - accuracy: 0.8524 - val_loss: 0.2628
Epoch 77/100
53/53 [==============================] - 0s 3ms/step - loss: 0.2543 - accuracy: 0.8512 - val_loss: 0.2547
Epoch 78/100
53/53 [==============================] - 0s 3ms/step - loss: 0.2519 - accuracy: 0.8589 - val_loss: 0.2641
Epoch 79/100
53/53 [==============================] - 0s 3ms/step - loss: 0.2517 - accuracy: 0.8512 - val_loss: 0.2643
```

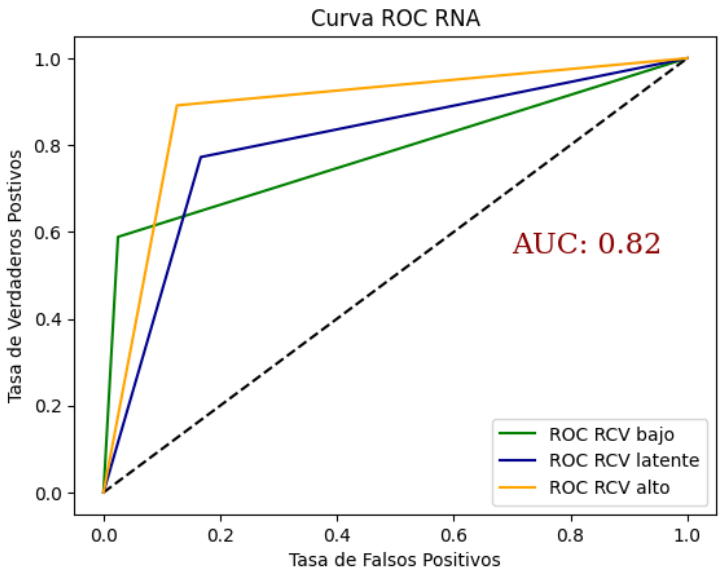


3. Después de ver que el rendimiento del modelo con la arquitectura aumento, decidimos aumentar el número de capas ocultas a 7.

```
[74]  1 # Definir la arquitectura del modelo de la RNA
      2 modelRNA = models.Sequential()
      3 modelRNA.add(Dense(3, batch_input_shape=(None, 35), activation='relu')) ## neuronas en la capa de entrada (batch
      4
      5 modelRNA.add(Dense(128, activation='relu'))
      6 modelRNA.add(Dense(64, activation='relu'))
      7 modelRNA.add(Dense(32, activation='relu'))
      8 modelRNA.add(Dense(16, activation='relu'))
      9 modelRNA.add(Dense(8, activation='relu'))
     10
     11
     12 modelRNA.add(Dense(3, activation='softmax'))
```

```
[98]  1 training_log = modelRNA.fit(X_train,
      2                     Y_train,
      3                     epochs=100,
      4                     batch_size=32,
      5                     validation_data=(X_valid, Y_valid),
      6                     verbose=1)
```

```
53/53 [==============================] - 0s 5ms/step - loss: 0.2512 - accuracy: 0.8488 - val_loss: 0.2687 - val_accu
Epoch 73/100
53/53 [==============================] - 0s 4ms/step - loss: 0.2494 - accuracy: 0.8554 - val_loss: 0.2877 - val_accu
Epoch 74/100
53/53 [==============================] - 0s 6ms/step - loss: 0.2523 - accuracy: 0.8536 - val_loss: 0.2695 - val_accu
Epoch 75/100
53/53 [==============================] - 0s 5ms/step - loss: 0.2498 - accuracy: 0.8518 - val_loss: 0.2714 - val_accu
Epoch 76/100
53/53 [==============================] - 0s 5ms/step - loss: 0.2480 - accuracy: 0.8542 - val_loss: 0.2699 - val_accu
Epoch 77/100
53/53 [==============================] - 0s 5ms/step - loss: 0.2498 - accuracy: 0.8500 - val_loss: 0.2802 - val_accu
Epoch 78/100
53/53 [==============================] - 0s 6ms/step - loss: 0.2474 - accuracy: 0.8583 - val_loss: 0.2695 - val_accu
Epoch 79/100
53/53 [==============================] - 0s 6ms/step - loss: 0.2499 - accuracy: 0.8554 - val_loss: 0.2677 - val_accu
Epoch 80/100
53/53 [==============================] - 0s 5ms/step - loss: 0.2488 - accuracy: 0.8577 - val_loss: 0.2709 - val_accu
Epoch 81/100
53/53 [==============================] - 0s 6ms/step - loss: 0.2475 - accuracy: 0.8536 - val_loss: 0.2683 - val_accu
```

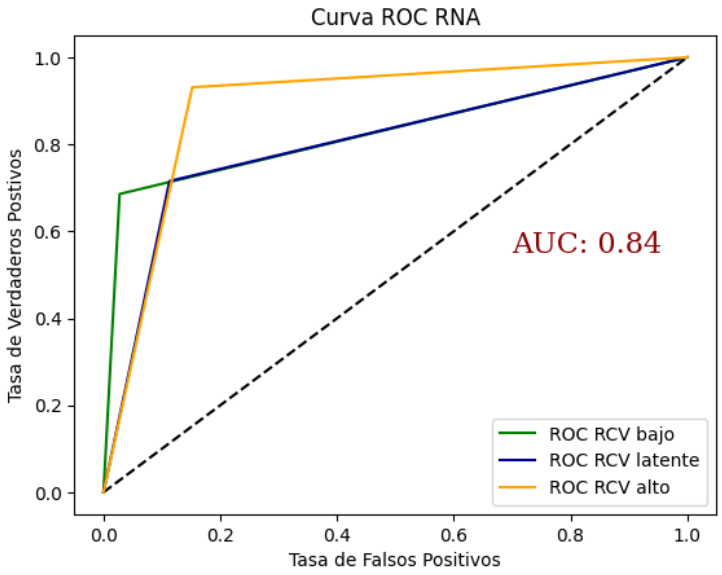


4. Como el AUC disminuyó, decidimos quitarle una capa

```
[116] 1 # Definir la arquitectura del modelo de la RNA
      2 modelRNA = models.Sequential()
      3 modelRNA.add(Dense(3, batch_input_shape=(None, 35), activation='relu')) ## neuronas en la capa de entrada (batch_i
      4
      5 modelRNA.add(Dense(64, activation='relu'))
      6 modelRNA.add(Dense(32, activation='relu'))
      7 modelRNA.add(Dense(16, activation='relu'))
      8 modelRNA.add(Dense(8, activation='relu'))
      9
     10
     11 modelRNA.add(Dense(3, activation='softmax'))
```

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

```
53/53 [==============================] - 0s 3ms/step - loss: 0.2659 - accuracy: 0.8298 - val_loss: 0.3098 - val_accu
Epoch 73/100
53/53 [==============================] - 0s 3ms/step - loss: 0.2668 - accuracy: 0.8304 - val_loss: 0.3010 - val_accu
Epoch 74/100
53/53 [==============================] - 0s 4ms/step - loss: 0.2648 - accuracy: 0.8310 - val_loss: 0.3020 - val_accu
Epoch 75/100
53/53 [==============================] - 0s 4ms/step - loss: 0.2656 - accuracy: 0.8310 - val_loss: 0.3005 - val_accu
Epoch 76/100
53/53 [==============================] - 0s 3ms/step - loss: 0.2651 - accuracy: 0.8304 - val_loss: 0.2999 - val_accu
Epoch 77/100
53/53 [==============================] - 0s 3ms/step - loss: 0.2646 - accuracy: 0.8310 - val_loss: 0.2985 - val_accu
Epoch 78/100
53/53 [==============================] - 0s 5ms/step - loss: 0.2648 - accuracy: 0.8315 - val_loss: 0.3015 - val_accu
Epoch 79/100
53/53 [==============================] - 0s 4ms/step - loss: 0.2643 - accuracy: 0.8256 - val_loss: 0.2979 - val_accu
Epoch 80/100
53/53 [==============================] - 0s 5ms/step - loss: 0.2635 - accuracy: 0.8310 - val_loss: 0.2997 - val_accu
Epoch 81/100
53/53 [==============================] - 0s 4ms/step - loss: 0.2626 - accuracy: 0.8375 - val_loss: 0.3005 - val_accu
```

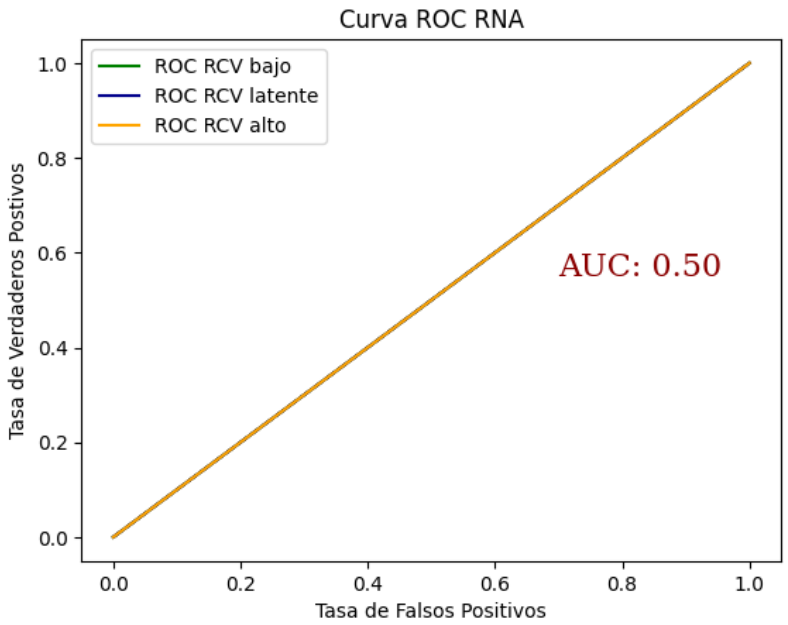


5. Observamos que, entre menos capas, mejor funciona, vamos a cambiarle el tipo de activación:

```
[179]  1 # Definir la arquitectura del modelo de la RNA
       2 modelRNA = models.Sequential()
       3 modelRNA.add(Dense(3, batch_input_shape=(None, 35), activation='softmax')) ## neuronas en la capa de entrada (ba
       4
       5 modelRNA.add(Dense(64, activation='softmax'))
       6 modelRNA.add(Dense(32, activation='softmax'))
       7 modelRNA.add(Dense(16, activation='softmax'))
       8 modelRNA.add(Dense(8, activation='softmax'))
       9
      10
      11 modelRNA.add(Dense(3, activation='softmax'))
```

```
[182]  1 training_log = modelRNA.fit(X_train,
       2                     Y_train,
       3                     epochs=100,
       4                     batch_size=32,
       5                     validation_data=(X_valid, Y_valid),
       6                     verbose=1)
```

```
Epoch 1/100
53/53 [==============================] - 1s 7ms/step - loss: 0.6853 - accuracy: 0.5143 - val_loss: 0.6739 - val_accu
Epoch 2/100
53/53 [==============================] - 0s 3ms/step - loss: 0.6645 - accuracy: 0.5321 - val_loss: 0.6552 - val_accu
Epoch 3/100
53/53 [==============================] - 0s 4ms/step - loss: 0.6477 - accuracy: 0.5321 - val_loss: 0.6396 - val_accu
Epoch 4/100
53/53 [==============================] - 0s 3ms/step - loss: 0.6333 - accuracy: 0.5321 - val_loss: 0.6267 - val_accu
Epoch 5/100
53/53 [==============================] - 0s 4ms/step - loss: 0.6216 - accuracy: 0.5321 - val_loss: 0.6163 - val_accu
Epoch 6/100
53/53 [==============================] - 0s 3ms/step - loss: 0.6125 - accuracy: 0.5321 - val_loss: 0.6082 - val_accu
Epoch 7/100
53/53 [==============================] - 0s 3ms/step - loss: 0.6056 - accuracy: 0.5321 - val_loss: 0.6023 - val_accu
Epoch 8/100
53/53 [==============================] - 0s 3ms/step - loss: 0.6005 - accuracy: 0.5321 - val_loss: 0.5979 - val_accu
Epoch 9/100
53/53 [==============================] - 0s 3ms/step - loss: 0.5968 - accuracy: 0.5321 - val_loss: 0.5950 - val_accu
```

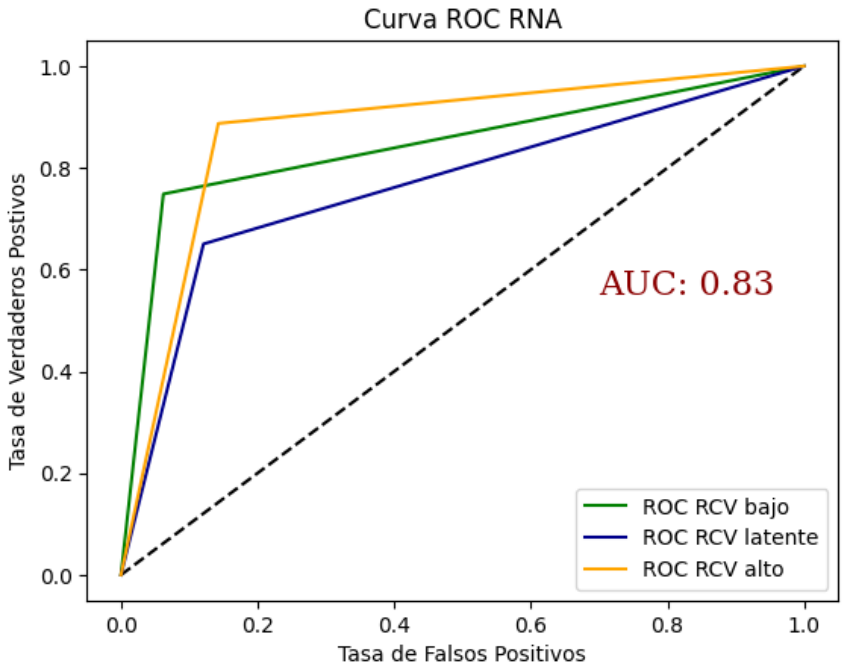


6. La activación softmax no dio buenos resultados, por lo tanto, volvemos a la anterior.

```
[221]  1 # Definir la arquitectura del modelo de la RNA
       2 modelRNA = models.Sequential()
       3 modelRNA.add(Dense(3, batch_input_shape=(None, 35), activation='relu')) ## neuronas en la capa de entrada (batch_i
       4
       5 modelRNA.add(Dense(64, activation='relu'))
       6 modelRNA.add(Dense(32, activation='relu'))
       7 modelRNA.add(Dense(16, activation='relu'))
       8 modelRNA.add(Dense(8, activation='relu'))
       9
      10
      11 modelRNA.add(Dense(3, activation='softmax'))
```

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

```
53/53 [==============================] - 0s 3ms/step - loss: 0.2539 - accuracy: 0.8476 - val_loss: 0.2838 - val_accu
Epoch 73/100
53/53 [==============================] - 0s 4ms/step - loss: 0.2547 - accuracy: 0.8518 - val_loss: 0.2658 - val_accu
Epoch 74/100
53/53 [==============================] - 0s 3ms/step - loss: 0.2533 - accuracy: 0.8542 - val_loss: 0.2671 - val_accu
Epoch 75/100
53/53 [==============================] - 0s 3ms/step - loss: 0.2534 - accuracy: 0.8470 - val_loss: 0.2624 - val_accu
Epoch 76/100
53/53 [==============================] - 0s 3ms/step - loss: 0.2511 - accuracy: 0.8512 - val_loss: 0.2641 - val_accu
Epoch 77/100
53/53 [==============================] - 0s 3ms/step - loss: 0.2509 - accuracy: 0.8458 - val_loss: 0.2625 - val_accu
Epoch 78/100
53/53 [==============================] - 0s 3ms/step - loss: 0.2504 - accuracy: 0.8494 - val_loss: 0.2666 - val_accu
Epoch 79/100
53/53 [==============================] - 0s 3ms/step - loss: 0.2489 - accuracy: 0.8542 - val_loss: 0.2665 - val_accu
Epoch 80/100
53/53 [==============================] - 0s 4ms/step - loss: 0.2478 - accuracy: 0.8500 - val_loss: 0.2719 - val_accu
Epoch 81/100
```



7. Realizamos una ultima prueba, cambiando el número de neuronas.

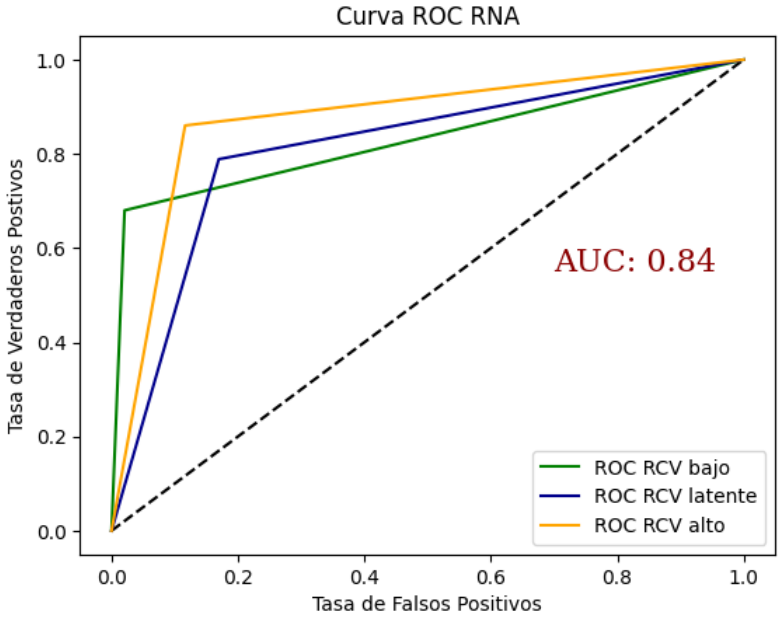
```
# Definir la arquitectura del modelo de la RNA
modelRNA = models.Sequential()
modelRNA.add(Dense(3, batch_input_shape=(None, 35), activation='relu')) ## neuronas en la capa de entrada (batch_i

modelRNA.add(Dense(256, activation='relu'))
modelRNA.add(Dense(128, activation='relu'))
modelRNA.add(Dense(64, activation='relu'))
modelRNA.add(Dense(32, activation='relu'))


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

```
[245]  1 training_log = modelRNA.fit(X_train,
       2                     Y_train,
       3                     epochs=100,
       4                     batch_size=32,
       5                     validation_data=(X_valid, Y_valid),
       6                     verbose=1)
```

```
53/53 [==============================] - 0s 3ms/step - loss: 0.2279 - accuracy: 0.8732 - val_loss: 0.3262 - val_accu
Epoch 73/100
53/53 [==============================] - 0s 4ms/step - loss: 0.2276 - accuracy: 0.8750 - val_loss: 0.3158 - val_accu
Epoch 74/100
53/53 [==============================] - 0s 3ms/step - loss: 0.2233 - accuracy: 0.8804 - val_loss: 0.3160 - val_accu
Epoch 75/100
53/53 [==============================] - 0s 3ms/step - loss: 0.2259 - accuracy: 0.8732 - val_loss: 0.3051 - val_accu
Epoch 76/100
53/53 [==============================] - 0s 4ms/step - loss: 0.2233 - accuracy: 0.8780 - val_loss: 0.3012 - val_accu
Epoch 77/100
53/53 [==============================] - 0s 3ms/step - loss: 0.2211 - accuracy: 0.8786 - val_loss: 0.3501 - val_accu
Epoch 78/100
53/53 [==============================] - 0s 3ms/step - loss: 0.2252 - accuracy: 0.8798 - val_loss: 0.3216 - val_accu
Epoch 79/100
53/53 [==============================] - 0s 3ms/step - loss: 0.2228 - accuracy: 0.8750 - val_loss: 0.3164 - val_accu
Epoch 80/100
53/53 [==============================] - 0s 3ms/step - loss: 0.2228 - accuracy: 0.8762 - val_loss: 0.3418 - val_accu
Epoch 81/100
53/53 [==============================] - 0s 3ms/step - loss: 0.2248 - accuracy: 0.8762 - val_loss: 0.3370 - val_accu
```



CONCLUSIONES

1. El aumento del número de capas ocultas a 7 resultó en un menor AUC, lo cual indica que una arquitectura más profunda no siempre es beneficiosa. Reducir el número de capas mejoró el rendimiento, lo que sugiere que un modelo más sencillo puede generalizar mejor.
2. Probamos diferentes funciones de activación y encontramos que la función de activación original nos dio mejores resultados comparada con la activación softmax para las capas ocultas.
3. Ajustar el número de epochs a 100 permitió un entrenamiento más exhaustivo del modelo, proporcionando un balance adecuado entre sobreajuste y subajuste.