MODELO DE REDES NEURONALES (RNA) – LABORATORIO 4

Se tomo un modelo de redes neuronales RNA y se entrenó con datos de pacientes con RCV (Riesgo Cardio Vascular). Actualmente tiene una exactitud del 82% esto se logra solo con 1 capa oculta y 1 neurona de la siguiente manera.

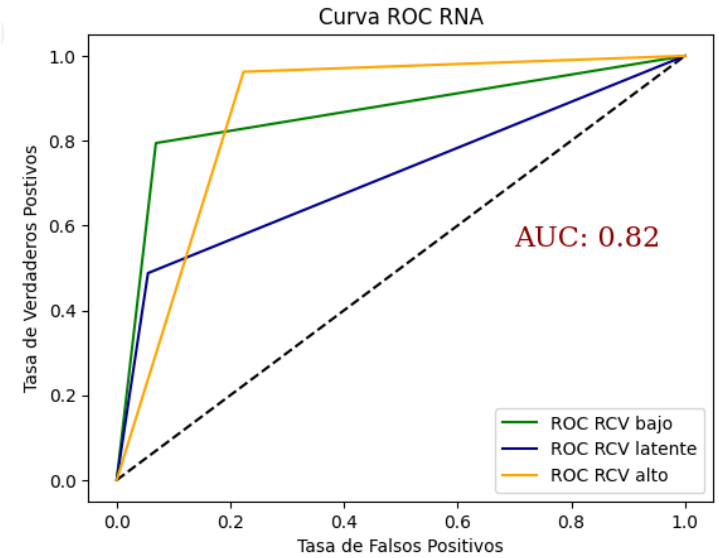
```
# Definir la arquitectura del modelo de la RNA ..................TENSORFL
modelRNA = models.Sequential() # se declara el modelo secuanbcial
#Primer nummero es el numero de neuronas en la capa oculta y el segundo n
modelRNA.add(Dense(1, batch_input_shape=(None, 35), activation='relu')) #
#modelRNA.add(Dense(4, activation='PReLU')) # Este linea crea una nueva c
# modelRNA.add(Dense(3, activation='PReLU'))
# activation='softmax' en las salidas pero existen mas funciones de activ
modelRNA.add(Dense(3, activation='softmax')) # definir el numero de categ
```

Creamos una variable para declarar el modelo en este caso nuestra variable la llamamos **modelRNA,** Una vez este declarado el modelo procedemos agregar las capas y las neuronas. En este ejemplo vemos que el numero 1 es la capa oculta y el 35 es el numero de variables que se tienen. Activamos 2 funciones de activación 1 de entrada y 1 de salida, para la función de entrada utilizamos **relu** y para la función de salida utilizamos **softmax**

ReLU (Rectified Linear Unit): La función de activación ReLU es una de las más populares y ampliamente utilizadas en redes neuronales, especialmente en redes profundas. Su popularidad se debe a su simplicidad y eficiencia computacional. La función retorna 0 si el valor de entrada es negativo y retorna el valor de entrada mismo si es positivo y su rango [0, hasta infinito).

Softmax: La función de activación Softmax se utiliza principalmente en la capa de salida de las redes neuronales para problemas de clasificación multiclase. Convierte un vector de valores en una distribución de probabilidad. Softmax toma un vector de valores reales y los convierte en probabilidades que suman a 1. Cada valor en la salida de Softmax está entre 0 y 1. Y su rango va de (0, 1).

Ahora vamos a evaluar la curva ROC con el ejercicio que viene predefinido.



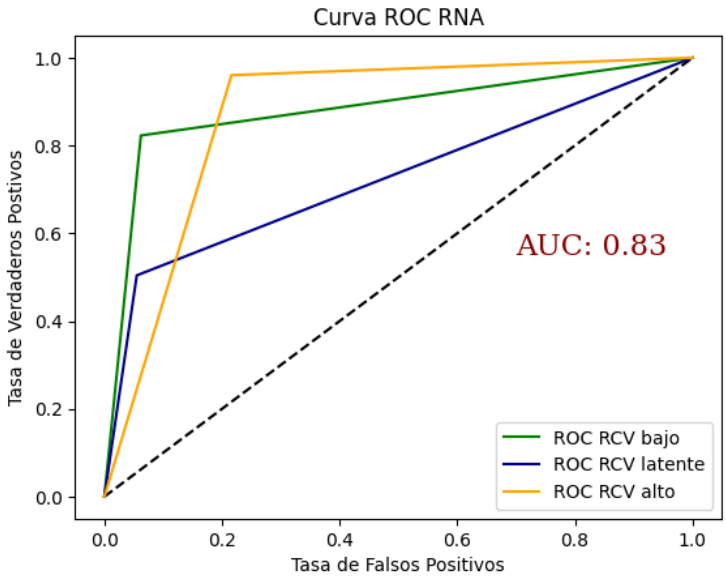
Entonces podemos decir que para la primera prueba con **1 capa oculta** y **1 neurona** se tiene una exactitud del **0.82** se debe tener en cuenta que entre más se acerque a 1 es porque el modelo va estar más exacto.

PRUEBA NUEMERO 1

Para la prueba numero 1 agregamos una nueva capa oculta y una nueva neurona, esto nos dejó un total de **2 x 2,** es decir, un total de **2 neuronas con 2 capas ocultas**.

```
# Definir la arquitectura del modelo de la RNA ..................TENSORFLOW...
modelRNA = models.Sequential() # se declara el modelo secuanbcial
#Primer nummero es el numero de neuronas en la capa oculta y el segundo numero
modelRNA.add(Dense(1, batch_input_shape=(None, 35), activation='relu')) ## neu
modelRNA.add(Dense(1, activation='relu'))
#modelRNA.add(Dense(4, activation='PReLU')) # Este linea crea una nueva capa c
# activation='softmax' en las salidas pero existen mas funciones de activacion
modelRNA.add(Dense(3, activation='softmax')) # definir el numero de categorias
```

Se tiene en cuenta que no se cambió la función de activación en este caso seguimos utilizando **relu.**

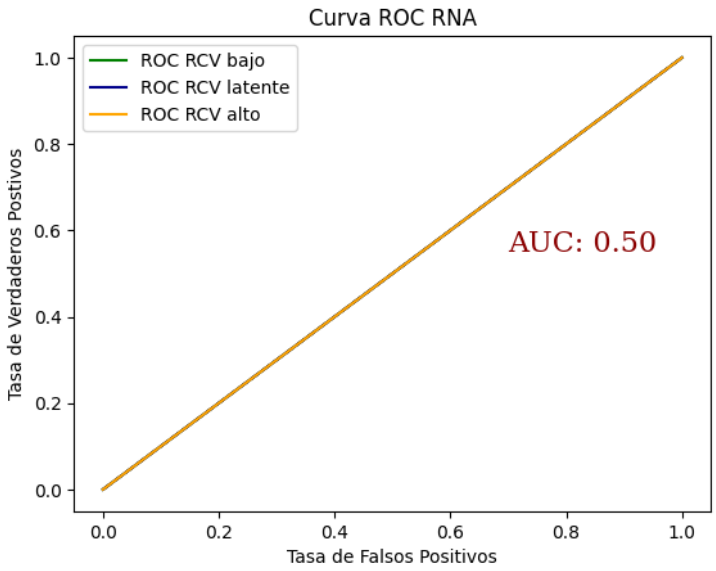


La exactitud de nuestro modelo ha subido a **0.83.** Evidenciamos el aumento de un **0.01,** esto sucedió cuando agregamos una neurona y una capa oculta más, eso quiere decir, **¿Si agrego 4 capas ocultas con 4 neuronas subirá un 0.04?**, vamos a intentarlo.

PRUEBA NUMERO 2

Agregamos 4 capas ocultas y 4 neuronas para un total de 4 x 4 y dejamos predefinida la función de activación relu.

```
# Definir la arquitectura del modelo de la RNA ..................TENSORFLOW...
modelRNA = models.Sequential() # se declara el modelo secuanbcial
#Primer nummero es el numero de neuronas en la capa oculta y el segundo numero
modelRNA.add(Dense(1, batch_input_shape=(None, 35), activation='relu')) ## neu
modelRNA.add(Dense(1, activation='relu'))
modelRNA.add(Dense(1, activation='relu'))
modelRNA.add(Dense(1, activation='relu'))
modelRNA.add(Dense(1, activation='relu'))
#modelRNA.add(Dense(4, activation='PReLU')) # Este linea crea una nueva capa o
# activation='softmax' en las salidas pero existen mas funciones de activacion
modelRNA.add(Dense(3, activation='softmax')) # definir el numero de categorias
```

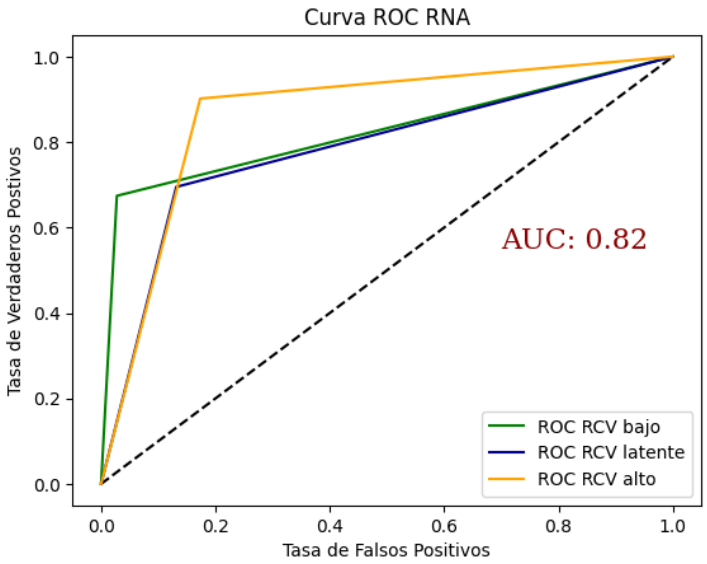


Se evidencia que la respuesta a nuestra pregunta es un NO, la exactitud bajo considerable mente hasta **0.50** eso quiere decir que el aumento de capas ocultas no significa que le de mas exactitud al modelo. Ahora vamos a volver a las primeras 2 capas que hicimos en la prueba 1 pero vamos aumentas las neuronas en esas capas eso quiere decir que **tenemos 2 capas con 4 neuronas 2 neuronas en cada capa, 2 x 4.**

PRUEBA NUMERO 3

```
[24] # Definir la arquitectura del modelo de la RNA ..................TENSORFLOW....
     modelRNA = models.Sequential() # se declara el modelo secuanbcial
     #Primer nummero es el numero de neuronas en la capa oculta y el segundo numero
     modelRNA.add(Dense(2, batch_input_shape=(None, 35), activation='relu')) ## neur
     modelRNA.add(Dense(2, activation='relu'))
     #modelRNA.add(Dense(4, activation='PReLU')) # Este linea crea una nueva capa oc
     # activation='softmax' en las salidas pero existen mas funciones de activacion
     modelRNA.add(Dense(3, activation='softmax')) # definir el numero de categorias
```

Entonces agregamos 2 neuronas por capa y seguimos dejando predefinida la función relu.

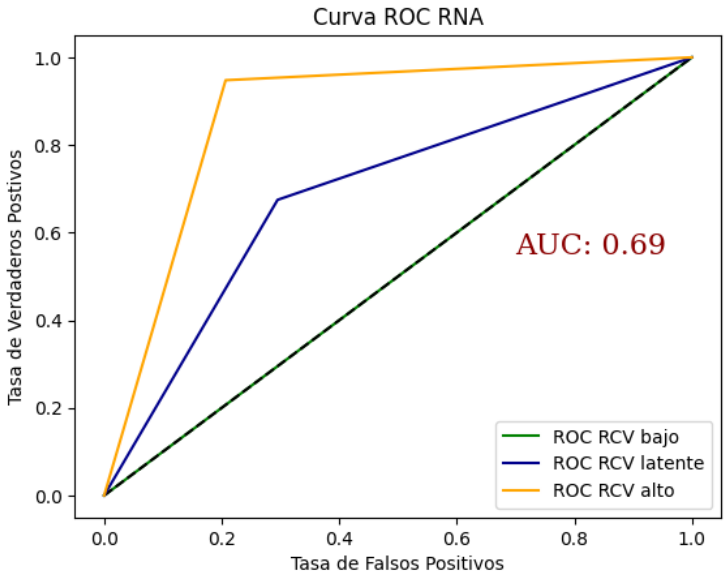


Evidenciamos que con el 2 x 4 nos devolvió al mismo **0.82** de exactitud, es decir, que podemos trabajar con 2 capa ocupas pero aumentando las neuronas a ver si vemos mejoría en la exactitud de nuestro modelo.

PRUEBA NUMERO 4

Seguimos con las mismas 2 capas ocultas, pero ahora vamos agregar de 3 en cada una, para dejar un **2 x 9.**

```
# Definir la arquitectura del modelo de la RNA ..................TENSORFLOW.
modelRNA = models.Sequential() # se declara el modelo secuanbcial
#Primer nummero es el numero de neuronas en la capa oculta y el segundo nume
modelRNA.add(Dense(3, batch_input_shape=(None, 35), activation='relu')) ## n
modelRNA.add(Dense(3, activation='relu'))
#modelRNA.add(Dense(4, activation='PReLU')) # Este linea crea una nueva capa
# activation='softmax' en las salidas pero existen mas funciones de activaci
modelRNA.add(Dense(3, activation='softmax')) # definir el numero de categori
```

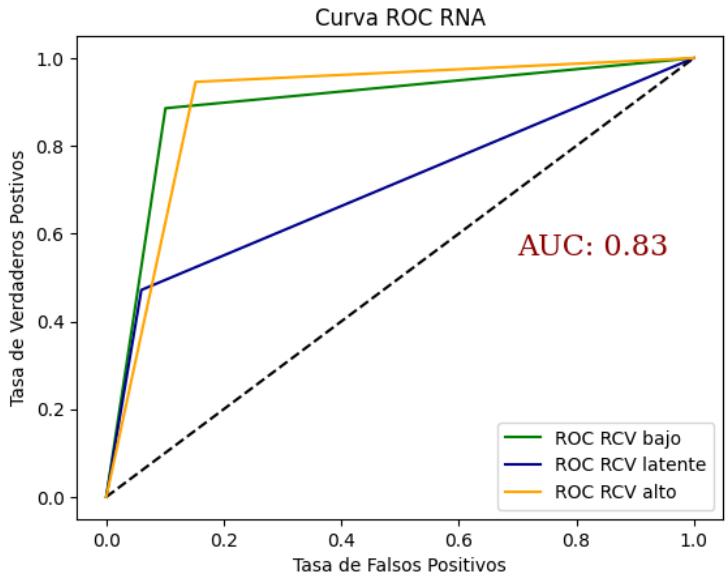


Se evidencia que la exactitud volvió a bajar considerablemente de **0.82 a 0.69** pero logre evidenciar lo siguiente, he puesto siempre un numero de neuronas similar, es decir, 3 por capa o 2 por capa o 1 por capa. ¿Qué pasa si ponemos número diferentes?

PRUEBA NUMERO 5

Entonces vamos a dejar las mimas 2 capas ocultas, pero ahora vamos a dejar 3 en la primera y 2 en la segunda.

```
# Definir la arquitectura del modelo de la RNA ..................TENSORFLOW.
modelRNA = models.Sequential() # se declara el modelo secuanbcial
#Primer nummero es el numero de neuronas en la capa oculta y el segundo nume
modelRNA.add(Dense(3, batch_input_shape=(None, 35), activation='relu')) ## n
modelRNA.add(Dense(2, activation='relu'))
#modelRNA.add(Dense(4, activation='PReLU')) # Este linea crea una nueva capa
# activation='softmax' en las salidas pero existen mas funciones de activaci
modelRNA.add(Dense(3, activation='softmax')) # definir el numero de categori
```

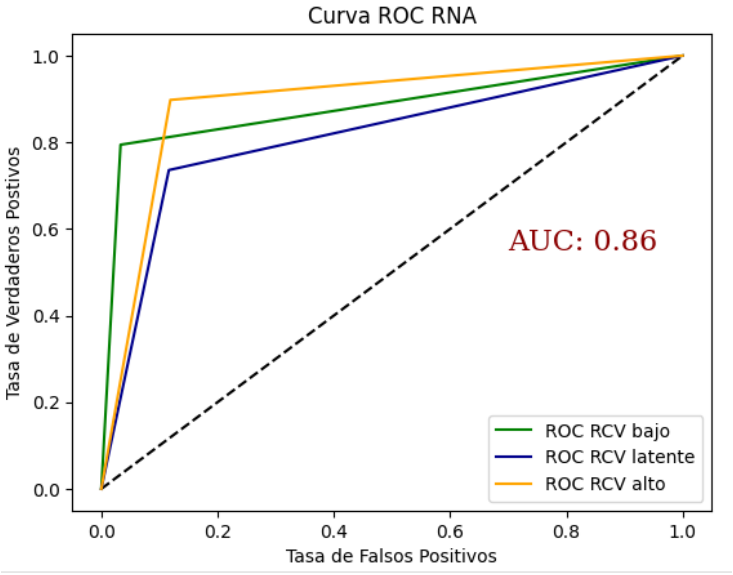


Evidenciamos una mejora en la exactitud del modelo del **0.83** eso quiere decir que podría funcionar agregando números de neuronas deferentes por capa.

PRUEBA NUMERO 6

```
# Definir la arquitectura del modelo de la RNA ..................TENSORFLOW..
modelRNA = models.Sequential() # se declara el modelo secuanbcial
#Primer nummero es el numero de neuronas en la capa oculta y el segundo numer
modelRNA.add(Dense(6, batch_input_shape=(None, 35), activation='relu')) ## ne
modelRNA.add(Dense(3, activation='relu'))
#modelRNA.add(Dense(4, activation='PReLU')) # Este linea crea una nueva capa
# activation='softmax' en las salidas pero existen mas funciones de activacio
modelRNA.add(Dense(3, activation='softmax')) # definir el numero de categoria
```

Ahora dejamos un total de 6 neuronas en la primera capa y un total de 3 neuronas en la segunda.



Evidenciamos una gran mejoría del **0.83** al **0.86** siempre predefiniendo la función de activación **relu.** Pero como se habló anteriormente esta función no es la única tenemos bastantes funciones de activación de entrada, en este caso haremos 2 pruebas en una.

PReLU: Esta función de activación nos dice que tiene una pendiente ajustable para los valores negativos, la cual es aprendida durante el entrenamiento. Esto permite que la función de activación se adapte a los datos de manera más efectiva. También no dice que podemos combinarla con **Relu** para que el modelo sea más eficiente.

ReLU: En las primeras capas puede proporcionar una activación rápida y eficiente.

PReLU: En las capas posteriores puede ajustar mejor los valores negativos, mejorando la capacidad de la red para aprender representaciones más complejas.

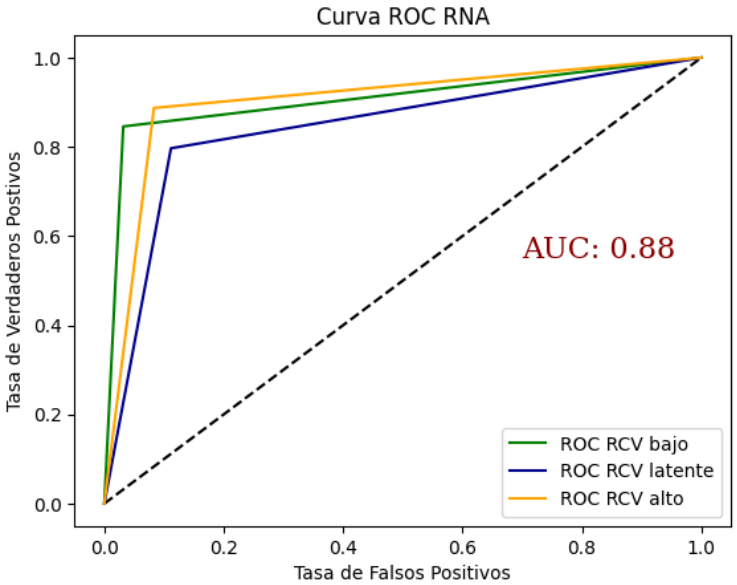
PRUEBA FINAL

```
# Definir la arquitectura del modelo de la RNA ..................TENSORFLOW..
modelRNA = models.Sequential() # se declara el modelo secuanbcial
#Primer nummero es el numero de neuronas en la capa oculta y el segundo numer
modelRNA.add(Dense(6, batch_input_shape=(None, 35), activation='relu')) ## ne
modelRNA.add(Dense(4, activation='PReLU'))
#modelRNA.add(Dense(4, activation='PReLU')) # Este linea crea una nueva capa
# activation='softmax' en las salidas pero existen mas funciones de activacio
modelRNA.add(Dense(3, activation='softmax')) # definir el numero de categoria
```

He añadido una neurona más a la red neuronal para la capa 2 y he combinado las funciones de activación. La primera capa utiliza **ReLU** y la segunda capa utiliza **PReLU**. Según la documentación consultada, esta combinación tiene las siguientes ventajas:

La activación **ReLU** permite que las primeras capas realicen cálculos de manera rápida y eficiente. lo que significa que las neuronas se activan solo si la entrada es positiva, ayudando a mitigar el problema del desvanecimiento del gradiente y mejorando la velocidad del entrenamiento. **PReLU** introduce flexibilidad adicional, ya que aprende la pendiente de los valores

negativos, ayudando a reducir el problema de las neuronas muertas y mejorando la capacidad de la red para capturar relaciones complejas en los datos. En conjunto, esta combinación aprovecha la eficiencia de **ReLU** en las primeras capas y la adaptabilidad de **PReLU** en las capas posteriores, lo que puede llevar a una mejora en la precisión del modelo.



Este fue mi resultado final una exactitud del **0.88** no se si sea posible subir la exactitud con la cantidad de datos que se tienen actualmente, pero combine nuevamente el numero de neuronas y agregue otras capas, pero esta combinación fue la que me dio la exactitud más alta.

CONCLUCIÓN

Para que un modelo de Red Neuronal (RNA) sea mas eficiente es correcto combinar las funciones de activación para el momento que se estén procesando los datos, puede que si se necesite un mayor número de neuronas que de capas o por lo menos para este ejercicio me funciono con mas neuronas que capas.