UNIVERSIDAD ECCI

Harold Shneider Martinez Tapiero - 77999
Luis Felipe Sanchez Sanchez - 90613

Elias Buitrago Bolivar

Facultad de Ingenieria
Seminario BigData
Bogota D.C.
2.024

1. En la prueba numero 1: se utilizan 2 capas ocultas con 7 y 12 neuronas respectivamente y 1 capa de salida con 3 neuronas.

Compilar el modelo creado

```
# Definir la arquitectura del modelo de la RNA
modelRNA = models.Sequential()
modelRNA.add(Dense(7, batch_input_shape=(None, 35), activation='relu')) ## neuronas en la capa de entrada (batch_input_shape) y #neuronas en la primera capa oculta
modelRNA.add(Dense(12, activation='relu'))
modelRNA.add(Dense(3, activation='softmax'))
```

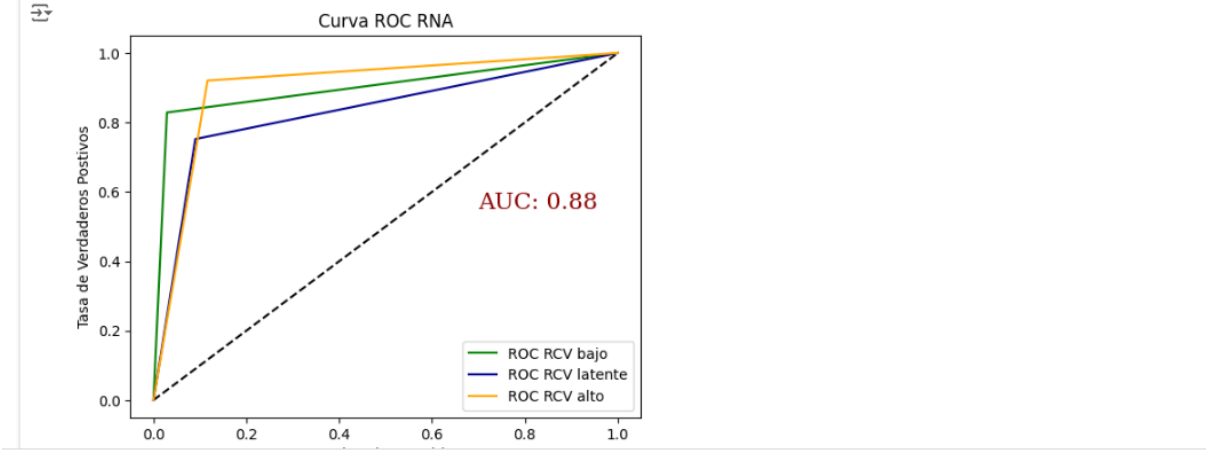
```
[13]
# compile the keras (tensorflow) flow graph
modelRNA.compile(optimizer=optimizers.RMSprop(learning_rate=0.001),
                 loss='binary_crossentropy',
                 metrics=['accuracy'])
```

Entrenamiento del modelo de RNA

```
[14]
# Inicializar el reloj para calcular tiempo de cómputo
t0 = process_time()
```

```
[15]
training_log = modelRNA.fit(X_train,
                            Y_train,
                            epochs=200,
                            batch_size=32,
                            validation_data=(X_valid, Y_valid),
                            verbose=1)
```

```
# Curva ROC
# #https://stackabuse.com/understanding-roc-curves-with-python/
Y_pred = np_utils.to_categorical(y_pred)
auc = roc_auc_score(Y_test, Y_pred)
fpr, tpr, threshold = roc_curve(Y_test[:,0], Y_pred[:,0])
font = {'family': 'serif',
    'color':  'darkred',
    'weight': 'normal',
    'size': 16,
    }
plt.plot(fpr, tpr, color='green', label='ROC RCV bajo')
plt.plot([0, 1], [0, 1], color='black', linestyle='--')
plt.xlabel('Tasa de Falsos Positivos')
plt.ylabel('Tasa de Verdaderos Postivos')
plt.title('Curva ROC RNA')
fpr, tpr, threshold = roc_curve(Y_test[:,1], Y_pred[:,1])
plt.plot(fpr, tpr, color='darkblue', label='ROC RCV latente')
fpr, tpr, threshold = roc_curve(Y_test[:,2], Y_pred[:,2])
plt.plot(fpr, tpr, color='orange', label='ROC RCV alto')
plt.text(0.7, 0.55, 'AUC: %.2f' % auc, fontdict=font)
plt.legend()
plt.show()
```



2. En la segunda Prueba se utilizan 2 capas ocultas con 4 y 20 neuronas y 1 capa de salida con 3 neuronas.

Compilar el modelo creado

```
[72]  1 # Definir la arquitectura del modelo de la RNA
      2 modelRNA = models.Sequential()
      3 modelRNA.add(Dense(4, batch_input_shape=(None, 35), activation='relu')) ## neuronas en la capa de entrada (batch_input_shape) y #neuronas en la primera capa oculta
      4 modelRNA.add(Dense(20, activation='relu'))
      5 modelRNA.add(Dense(3, activation='softmax'))
```

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

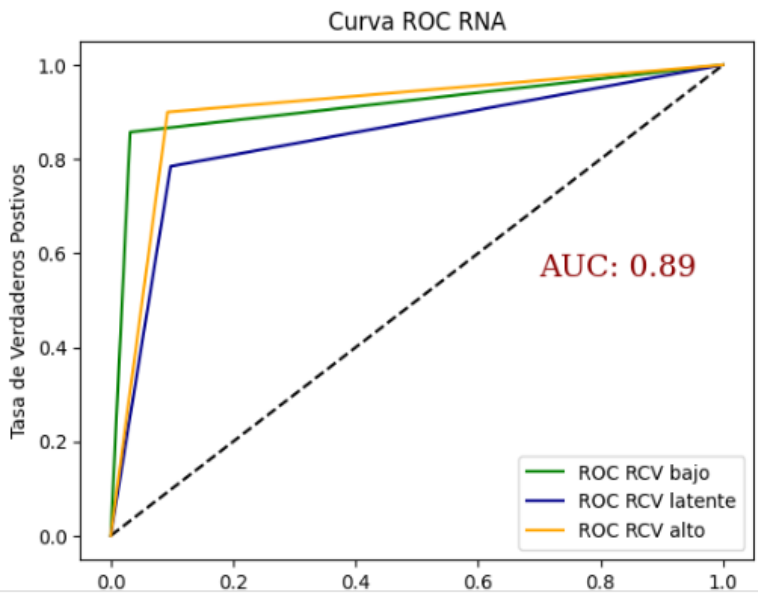
Entrenamiento del modelo de RNA

```
[74]  1 # Inicializar el reloj para calcular tiempo de cómputo
      2 t0 = process_time()
```

```
[75]  1 training_log = modelRNA.fit(X_train,
      2                   Y_train,
      3                   epochs=200,
      4                   batch_size=32,
      5                   validation_data=(X_valid, Y_valid),
      6                   verbose=1)
```

Curva ROC

```
 1 # Curva ROC
 2 # #https://stackabuse.com/understanding-roc-curves-with-python/
 3 Y_pred = np_utils.to_categorical(y_pred)
 4 auc = roc_auc_score(Y_test, Y_pred)
 5 fpr, tpr, threshold = roc_curve(Y_test[:,0], Y_pred[:,0])
 6 font = {'family': 'serif',
 7     'color':  'darkred',
 8     'weight': 'normal',
 9     'size': 16,
10     }
11 plt.plot(fpr, tpr, color='green', label='ROC RCV bajo')
12 plt.plot([0, 1], [0, 1], color='black', linestyle='--')
13 plt.xlabel('Tasa de Falsos Positivos')
14 plt.ylabel('Tasa de Verdaderos Postivos')
15 plt.title('Curva ROC RNA')
16 fpr, tpr, threshold = roc_curve(Y_test[:,1], Y_pred[:,1])
17 plt.plot(fpr, tpr, color='darkblue', label='ROC RCV latente')
18 fpr, tpr, threshold = roc_curve(Y_test[:,2], Y_pred[:,2])
19 plt.plot(fpr, tpr, color='orange', label='ROC RCV alto')
20 plt.text(0.7, 0.55, 'AUC: %.2f' % auc, fontdict=font)
21 plt.legend()
22 plt.show()
```



3. La prueba numero 3, se realizo con 3 capas ocultas que contaban con 4, 7, 20 neuronas y 1 capa de salida con 3 neuronas.

Compilar el modelo creado

```
[92] 1 # Definir la arquitectura del modelo de la RNA
     2 modelRNA = models.Sequential()
     3 modelRNA.add(Dense(4, batch_input_shape=(None, 35), activation='relu')) ## neuronas en la capa de entrada (batch_input_shape) y #neuronas en la primera capa oculta
     4 modelRNA.add(Dense(7, activation='relu'))
     5 modelRNA.add(Dense(20, activation='relu'))
     6 modelRNA.add(Dense(3, activation='softmax'))
```

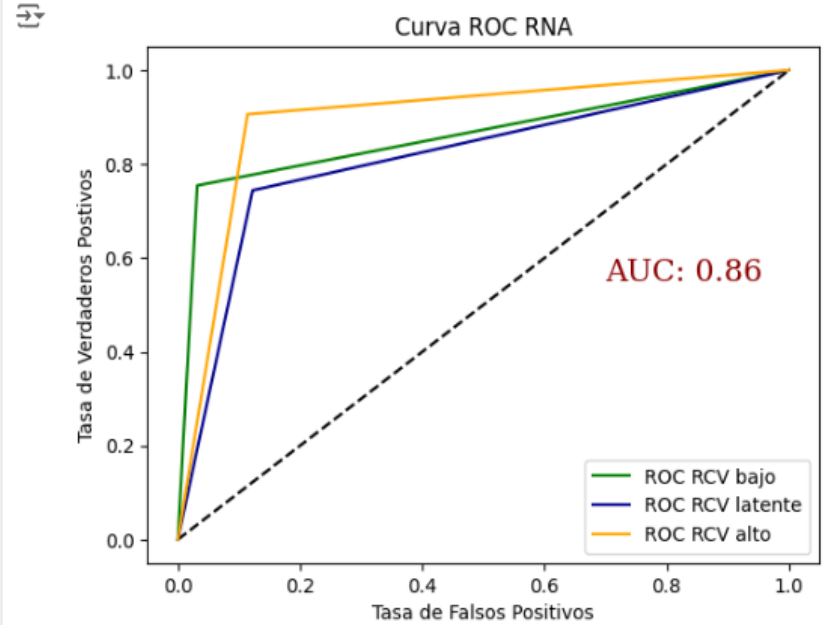
```
     1 # compile the keras (tensorflow) flow graph
     2 modelRNA.compile(optimizer=optimizers.RMSprop(learning_rate=0.001),
     3               loss='binary_crossentropy',
     4               metrics=['accuracy'])
```

Entrenamiento del modelo de RNA

```
[94] 1 # Inicializar el reloj para calcular tiempo de cómputo
     2 t0 = process_time()
```

```
[95] 1 training_log = modelRNA.fit(X_train,
     2                   Y_train,
     3                   epochs=200,
     4                   batch_size=32,
     5                   validation_data=(X_valid, Y_valid),
     6                   verbose=1)
```

```
 1 # Curva ROC
 2 # #https://stackabuse.com/understanding-roc-curves-with-python/
 3 Y_pred = np_utils.to_categorical(y_pred)
 4 auc = roc_auc_score(Y_test, Y_pred)
 5 fpr, tpr, threshold = roc_curve(Y_test[:,0], Y_pred[:,0])
 6 font = {'family': 'serif',
 7     'color':  'darkred',
 8     'weight': 'normal',
 9     'size': 16,
10     }
11 plt.plot(fpr, tpr, color='green', label='ROC RCV bajo')
12 plt.plot([0, 1], [0, 1], color='black', linestyle='--')
13 plt.xlabel('Tasa de Falsos Positivos')
14 plt.ylabel('Tasa de Verdaderos Postivos')
15 plt.title('Curva ROC RNA')
16 fpr, tpr, threshold = roc_curve(Y_test[:,1], Y_pred[:,1])
17 plt.plot(fpr, tpr, color='darkblue', label='ROC RCV latente')
18 fpr, tpr, threshold = roc_curve(Y_test[:,2], Y_pred[:,2])
19 plt.plot(fpr, tpr, color='orange', label='ROC RCV alto')
20 plt.text(0.7, 0.55, 'AUC: %.2f' % auc, fontdict=font)
21 plt.legend()
22 plt.show()
```



4. La cuarta prueba se realizo con 3 capas ocultas que contaban con 4, 12, 20 neuronas y la capa de salida con 3 neuronas.

```
[112] 1 # Definir la arquitectura del modelo de la RNA
      2 modelRNA = models.Sequential()
      3 modelRNA.add(Dense(4, batch_input_shape=(None, 35), activation='relu')) ## neuronas en la capa de entrada (batch_input_shape) y #neuronas en la primera capa oculta
      4 modelRNA.add(Dense(12, activation='relu'))
      5 modelRNA.add(Dense(20, activation='relu'))
      6 modelRNA.add(Dense(3, activation='softmax'))
```

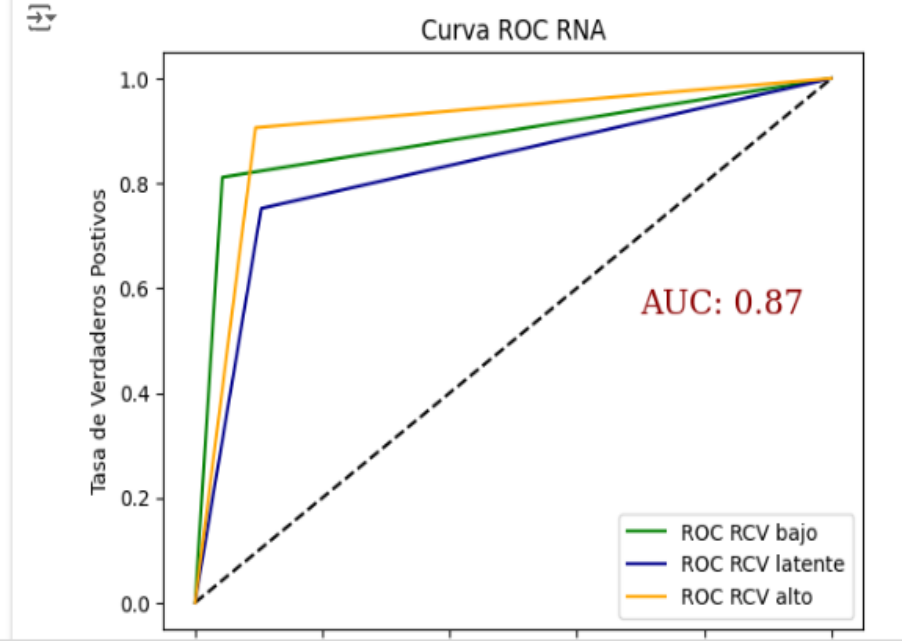
```
      1 # compile the keras (tensorflow) flow graph
      2 modelRNA.compile(optimizer=optimizers.RMSprop(learning_rate=0.001),
      3                  loss='binary_crossentropy',
      4                  metrics=['accuracy'])
```

Entrenamiento del modelo de RNA

```
[114] 1 # Inicializar el reloj para calcular tiempo de cómputo
      2 t0 = process_time()
```

```
[115] 1 training_log = modelRNA.fit(X_train,
      2                             Y_train,
      3                             epochs=200,
      4                             batch_size=32,
      5                             validation_data=(X_valid, Y_valid),
      6                             verbose=1)
```

```
 1 # Curva ROC
 2 # #https://stackabuse.com/understanding-roc-curves-with-python/
 3 Y_pred = np_utils.to_categorical(y_pred)
 4 auc = roc_auc_score(Y_test, Y_pred)
 5 fpr, tpr, threshold = roc_curve(Y_test[:,0], Y_pred[:,0])
 6 font = {'family': 'serif',
 7     'color':  'darkred',
 8     'weight': 'normal',
 9     'size': 16,
10     }
11 plt.plot(fpr, tpr, color='green', label='ROC RCV bajo')
12 plt.plot([0, 1], [0, 1], color='black', linestyle='--')
13 plt.xlabel('Tasa de Falsos Positivos')
14 plt.ylabel('Tasa de Verdaderos Postivos')
15 plt.title('Curva ROC RNA')
16 fpr, tpr, threshold = roc_curve(Y_test[:,1], Y_pred[:,1])
17 plt.plot(fpr, tpr, color='darkblue', label='ROC RCV latente')
18 fpr, tpr, threshold = roc_curve(Y_test[:,2], Y_pred[:,2])
19 plt.plot(fpr, tpr, color='orange', label='ROC RCV alto')
20 plt.text(0.7, 0.55, 'AUC: %.2f' % auc, fontdict=font)
21 plt.legend()
22 plt.show()
```

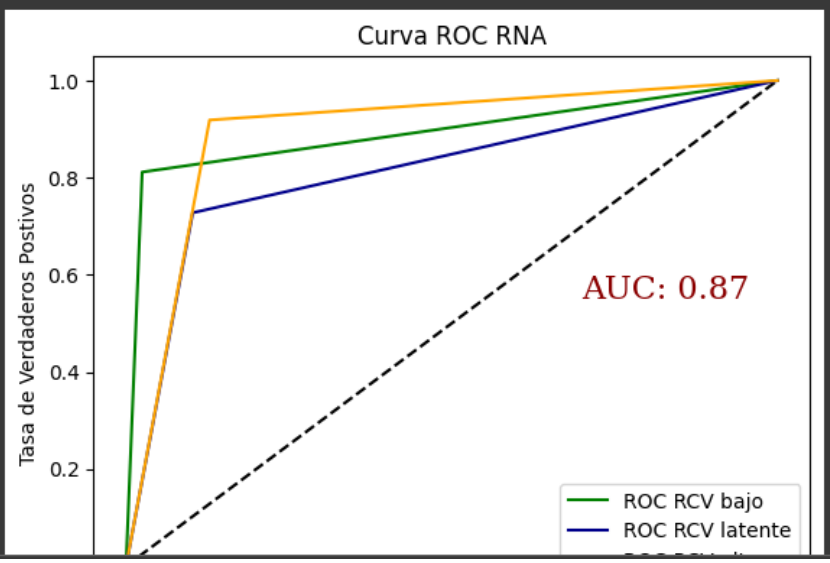


5. La quinta prueba se realizo con 4 capas ocultas que tienen 4, 15, 9, 21 neuronas y una capa de salida con 3 neuronas.

```python
# Definir la arquitectura del modelo de la RNA
modelRNA = models.Sequential()
modelRNA.add(Dense(4, batch_input_shape=(None, 35), activation='relu')) #
modelRNA.add(Dense(15, activation='relu'))
modelRNA.add(Dense(9, activation='relu'))
modelRNA.add(Dense(21, activation='relu'))
modelRNA.add(Dense(3, activation='softmax'))
```

```python
[24]
# compile the keras (tensorflow) flow graph
modelRNA.compile(optimizer=optimizers.RMSprop(learning_rate=0.001),
                 loss='binary_crossentropy',
                 metrics=['accuracy'])
```

```python
# Curva ROC
# #https://stackabuse.com/understanding-roc-curves-with-python/
Y_pred = np_utils.to_categorical(y_pred)
auc = roc_auc_score(Y_test, Y_pred)
fpr, tpr, threshold = roc_curve(Y_test[:,0], Y_pred[:,0])
font = {'family': 'serif',
    'color':  'darkred',
    'weight': 'normal',
    'size': 16,
    }
plt.plot(fpr, tpr, color='green', label='ROC RCV bajo')
plt.plot([0, 1], [0, 1], color='black', linestyle='--')
plt.xlabel('Tasa de Falsos Positivos')
plt.ylabel('Tasa de Verdaderos Postivos')
plt.title('Curva ROC RNA')
fpr, tpr, threshold = roc_curve(Y_test[:,1], Y_pred[:,1])
plt.plot(fpr, tpr, color='darkblue', label='ROC RCV latente')
fpr, tpr, threshold = roc_curve(Y_test[:,2], Y_pred[:,2])
plt.plot(fpr, tpr, color='orange', label='ROC RCV alto')
plt.text(0.7, 0.55, 'AUC: %.2f' % auc, fontdict=font)
plt.legend()
plt.show()
```



CONCLUSIONES

En la practica realizada con el fin de obtener un mejor resultado en la Curva ROC RNA, se realizaron diferentes pruebas (5 de ellas mostradas en el actual documento), en las cuales se evidencio que entre más capas ocultas el resultado es mas próximo a 1, cabe resaltar un caso en el que, con dos capas, pero altas neuronas se obtiene también este porcentaje.

Se probo con la función de activación “tahn”, mostrando resultados menores, dando asi evidencia que la función “relu” es la mas optima para este caso, ya que alcanzo a arrojar un numero mayor el cual fue de 0.89.

Si se tomaran mas capas ocultas con altas cantidades de neuronas es posible llegar a superar el 0.90 esto con el fin de dar un mejor resultado en casos de predicción que puedan generarse.