Erick Alexander Velasquez de León
Carné: 21005129

# Laboratorio 1

Arquitectura 1: 0.6679

```
1 accuracy_score(y_test, y_preds)
0.667910447761194
```

```
[93] 1 #KFolds
     2 from keras.wrappers.scikit_learn import KerasClassifier
     3 from sklearn.model_selection import cross_val_score
     4 from sklearn.model_selection import KFold
```

```
1 def red_v1():
2   #base secuencial para definir la estructura de la Red (SIEMPRE INICIAR CON SQUENTIAL)
3   clasificador = Sequential()
4
5   #primera capa oclta
6   clasificador.add(Dense(input_dim = 11, units=6, activation='relu', kernel_initializer='uniform'))
7
8   #segunda capa oculta
9   clasificador.add(Dense(units=6, activation='relu', kernel_initializer='uniform'))
10
11  #capa de salida
12  clasificador.add(Dense(units=1, activation='sigmoid', kernel_initializer='uniform'))
13
14  #parametros de optimización
15  clasificador.compile(optimizer='SGD', loss='binary_crossentropy', metrics=['accuracy'])
16
17  return clasificador
```

Keras: 0.5939

```
1 accs.mean()
0.5939324200153351
```

**Descripción de Arquitectura de Red con Keras**

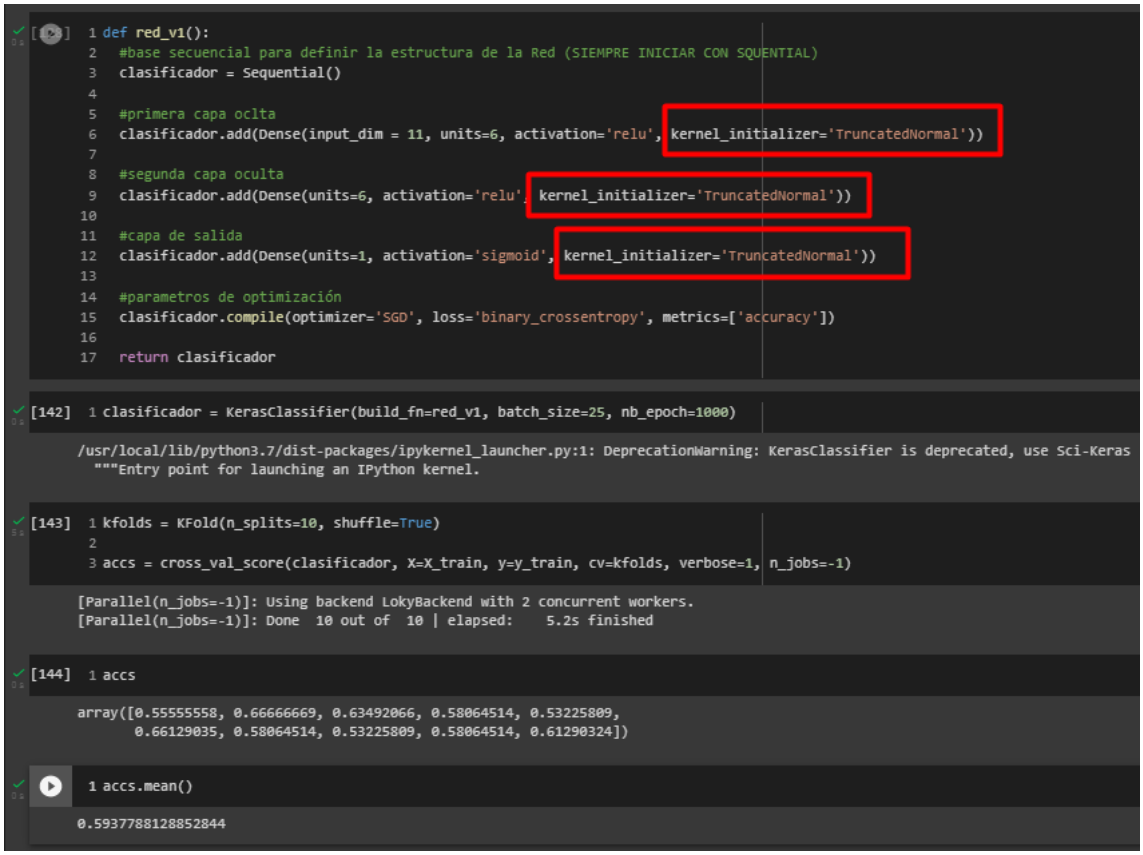
```
[82] 1 import keras
     2 from keras.models import Sequential
     3 from keras.layers import Dense
```

```
1 inputs = X_train.shape[1]
2
3 #base secuencial para definir la estructura de la Red (SIEMPRE INICIAR CON SQUENTIAL)
4 clasificador = Sequential()
5
6 #primera capa oclta
7 clasificador.add(Dense(input_dim = inputs, units=6, activation='relu', kernel_initializer='uniform'))
8
9 #segunda capa oculta
10 clasificador.add(Dense(units=6, activation='relu', kernel_initializer='uniform'))
11
12 #capa de salida
13 clasificador.add(Dense(units=1, activation='sigmoid', kernel_initializer='uniform'))
```

```
[84] 1 #parametros de optimización
     2 clasificador.compile(optimizer='SGD', loss='binary_crossentropy', metrics=['accuracy'])
     3
     4 #parametros de entrenamiento.
     5 clasificador.fit(X_train, y_train, batch_size=25, epochs=150)
```

Arquitectura 2:

Modificamos el Kernel Initializer a TruncateNormal, dio 0.5937



```
def red_v1():
  #base secuencial para definir la estructura de la Red (SIEMPRE INICIAR CON SQUENTIAL)
  clasificador = Sequential()

  #primera capa oclta
  clasificador.add(Dense(input_dim = 11, units=6, activation='relu', kernel_initializer='TruncatedNormal'))

  #segunda capa oculta
  clasificador.add(Dense(units=6, activation='relu', kernel_initializer='TruncatedNormal'))

  #capa de salida
  clasificador.add(Dense(units=1, activation='sigmoid', kernel_initializer='TruncatedNormal'))

  #parametros de optimización
  clasificador.compile(optimizer='SGD', loss='binary_crossentropy', metrics=['accuracy'])

  return clasificador
```

```
[142] clasificador = KerasClassifier(build_fn=red_v1, batch_size=25, nb_epoch=1000)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: DeprecationWarning: KerasClassifier is deprecated, use Sci-Keras
  """Entry point for launching an IPython kernel.
```

```
[143] kfolds = KFold(n_splits=10, shuffle=True)

accs = cross_val_score(clasificador, X=X_train, y=y_train, cv=kfolds, verbose=1, n_jobs=-1)
```

```
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 2 concurrent workers.
[Parallel(n_jobs=-1)]: Done  10 out of  10 | elapsed:    5.2s finished
```

```
[144] accs
```

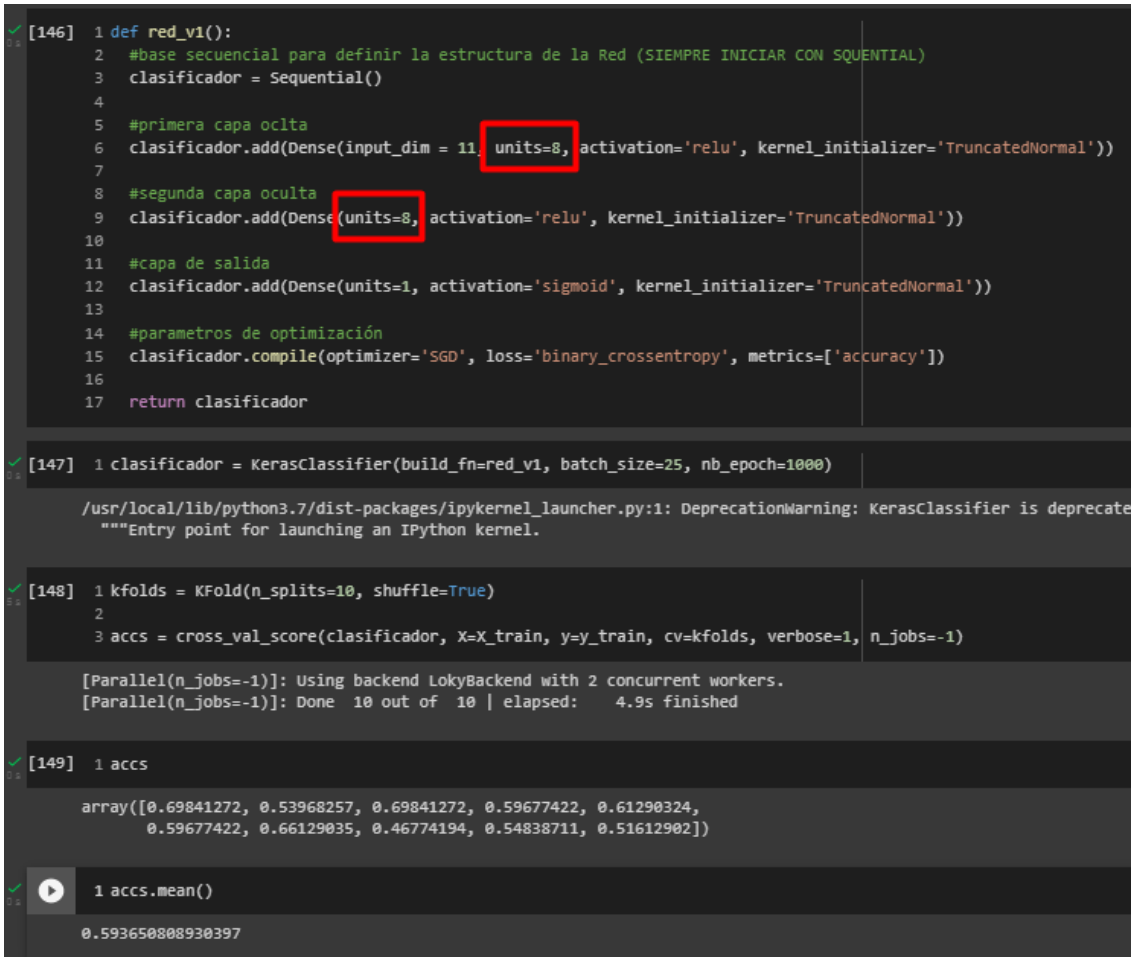
```
array([0.55555558, 0.66666669, 0.63492066, 0.58064514, 0.53225809,
       0.66129035, 0.58064514, 0.53225809, 0.58064514, 0.61290324])
```

```
accs.mean()
```

```
0.5937788128852844
```

Arquitectura 3:

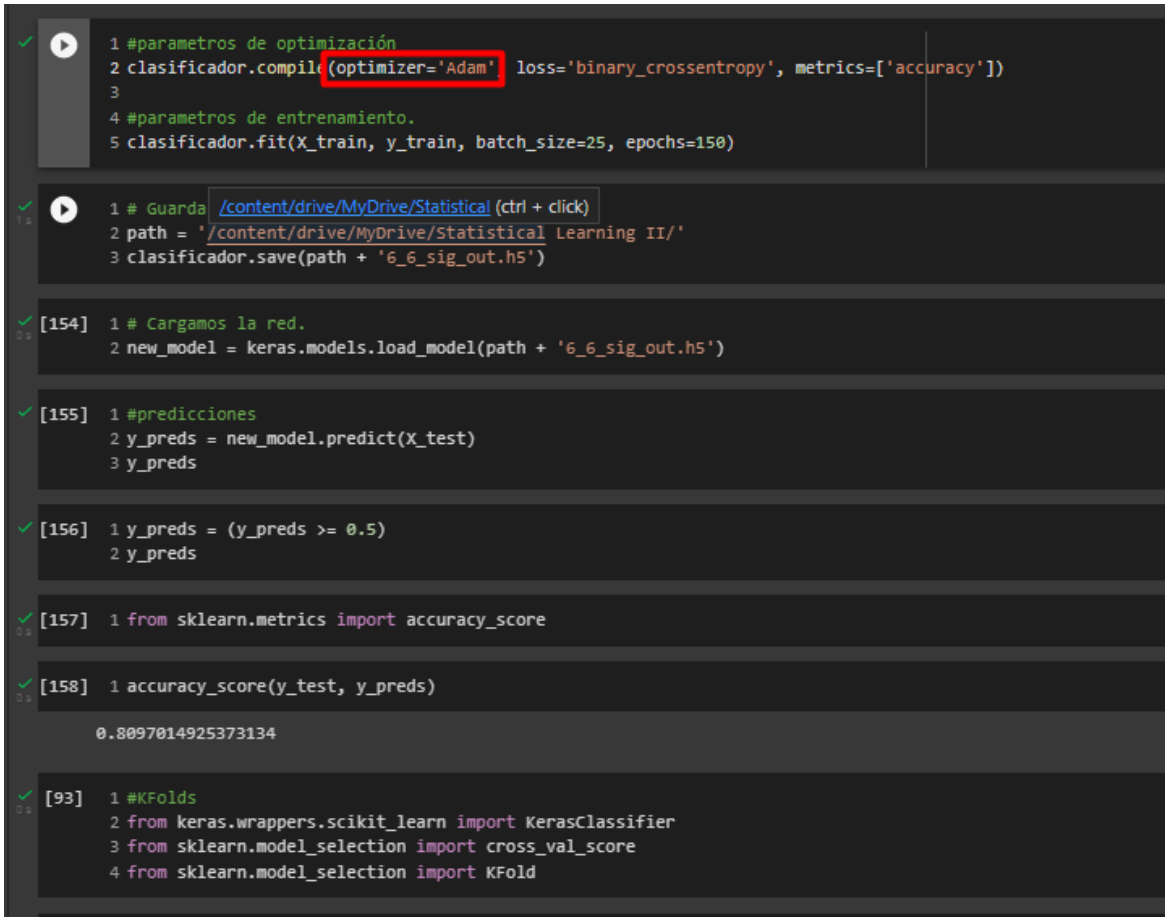
Modifificamos las unidades de nuestra primer y segunda capa oculta de 6 a 8. Bajo a 0.5936



```
[146] 1 def red_v1():
      2   #base secuencial para definir la estructura de la Red (SIEMPRE INICIAR CON SQUENTIAL)
      3   clasificador = Sequential()
      4
      5   #primera capa oclta
      6   clasificador.add(Dense(input_dim = 11, units=8, activation='relu', kernel_initializer='TruncatedNormal'))
      7
      8   #segunda capa oculta
      9   clasificador.add(Dense(units=8, activation='relu', kernel_initializer='TruncatedNormal'))
     10
     11   #capa de salida
     12   clasificador.add(Dense(units=1, activation='sigmoid', kernel_initializer='TruncatedNormal'))
     13
     14   #parametros de optimización
     15   clasificador.compile(optimizer='SGD', loss='binary_crossentropy', metrics=['accuracy'])
     16
     17   return clasificador

[147] 1 clasificador = KerasClassifier(build_fn=red_v1, batch_size=25, nb_epoch=1000)

/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: DeprecationWarning: KerasClassifier is deprecate
  """Entry point for launching an IPython kernel.

[148] 1 kfolds = KFold(n_splits=10, shuffle=True)
      2
      3 accs = cross_val_score(clasificador, X=X_train, y=y_train, cv=kfolds, verbose=1, n_jobs=-1)

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 2 concurrent workers.
[Parallel(n_jobs=-1)]: Done  10 out of  10 | elapsed:    4.9s finished

[149] 1 accs

array([0.69841272, 0.53968257, 0.69841272, 0.59677422, 0.61290324,
       0.59677422, 0.66129035, 0.46774194, 0.54838711, 0.51612902])

      1 accs.mean()

0.593650808930397
```

Arquitectura 4:

Cambiamos nuestro optimizar a Adam, acá subió a 0.8097



```
#parametros de optimización
clasificador.compile(optimizer='Adam', loss='binary_crossentropy', metrics=['accuracy'])

#parametros de entrenamiento.
clasificador.fit(X_train, y_train, batch_size=25, epochs=150)
```

```
# Guarda /content/drive/MyDrive/Statistical (ctrl + click)
path = '/content/drive/MyDrive/Statistical Learning II/'
clasificador.save(path + '6_6_sig_out.h5')
```

```
[154] # Cargamos la red.
new_model = keras.models.load_model(path + '6_6_sig_out.h5')
```

```
[155] #predicciones
y_preds = new_model.predict(X_test)
y_preds
```

```
[156] y_preds = (y_preds >= 0.5)
y_preds
```

```
[157] from sklearn.metrics import accuracy_score
```

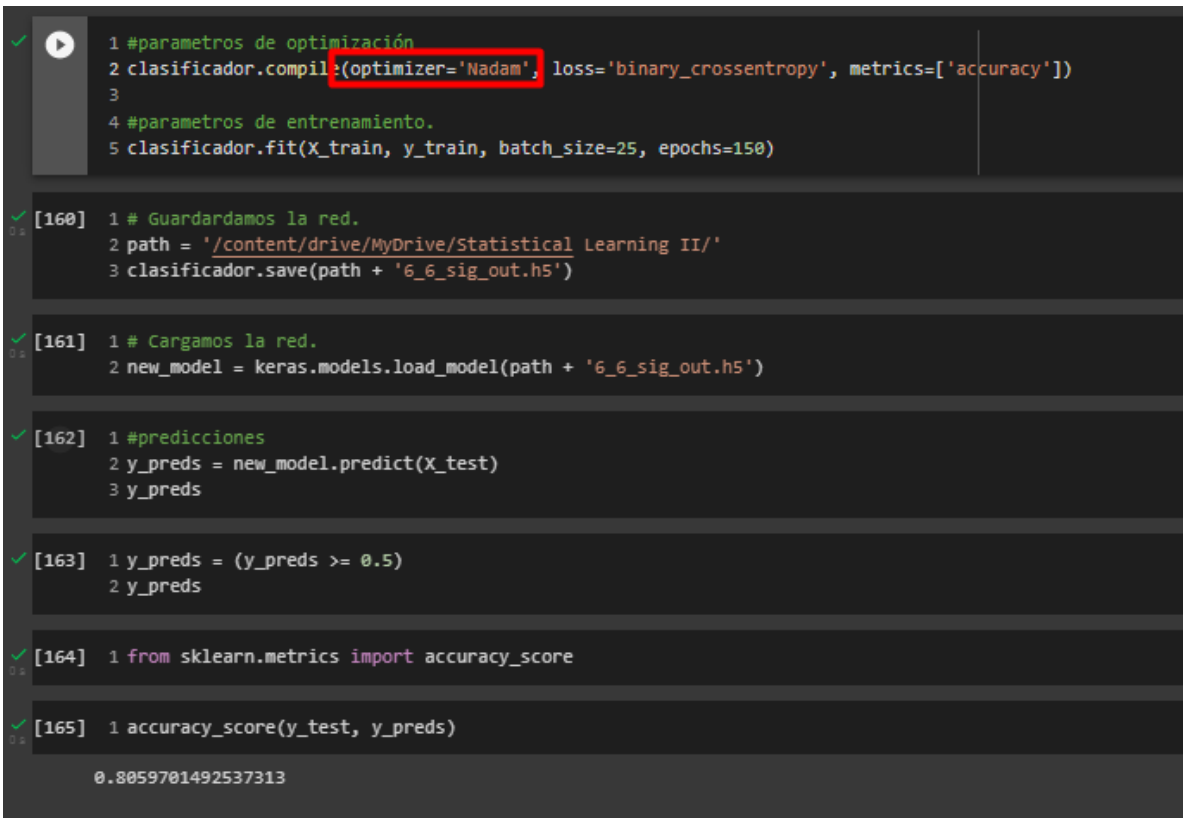
```
[158] accuracy_score(y_test, y_preds)
```

```
0.8097014925373134
```

```
[93] #KFolds
from keras.wrappers.scikit_learn import KerasClassifier
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold
```

Arquitectura 5:

Probamos con el optimizador Nadam, bajo a 0.8059.



```
#parametros de optimización
clasificador.compile(optimizer='Nadam', loss='binary_crossentropy', metrics=['accuracy'])

#parametros de entrenamiento.
clasificador.fit(X_train, y_train, batch_size=25, epochs=150)
```

```
[160]
# Guardardamos la red.
path = '/content/drive/MyDrive/Statistical Learning II/'
clasificador.save(path + '6_6_sig_out.h5')
```

```
[161]
# Cargamos la red.
new_model = keras.models.load_model(path + '6_6_sig_out.h5')
```

```
[162]
#predicciones
y_preds = new_model.predict(X_test)
y_preds
```

```
[163]
y_preds = (y_preds >= 0.5)
y_preds
```

```
[164]
from sklearn.metrics import accuracy_score
```

```
[165]
accuracy_score(y_test, y_preds)
```

```
0.8059701492537313
```

**La mejor métrica fue utilizando el optimizador de Adam en donde se alcanzo un 0.8097.**