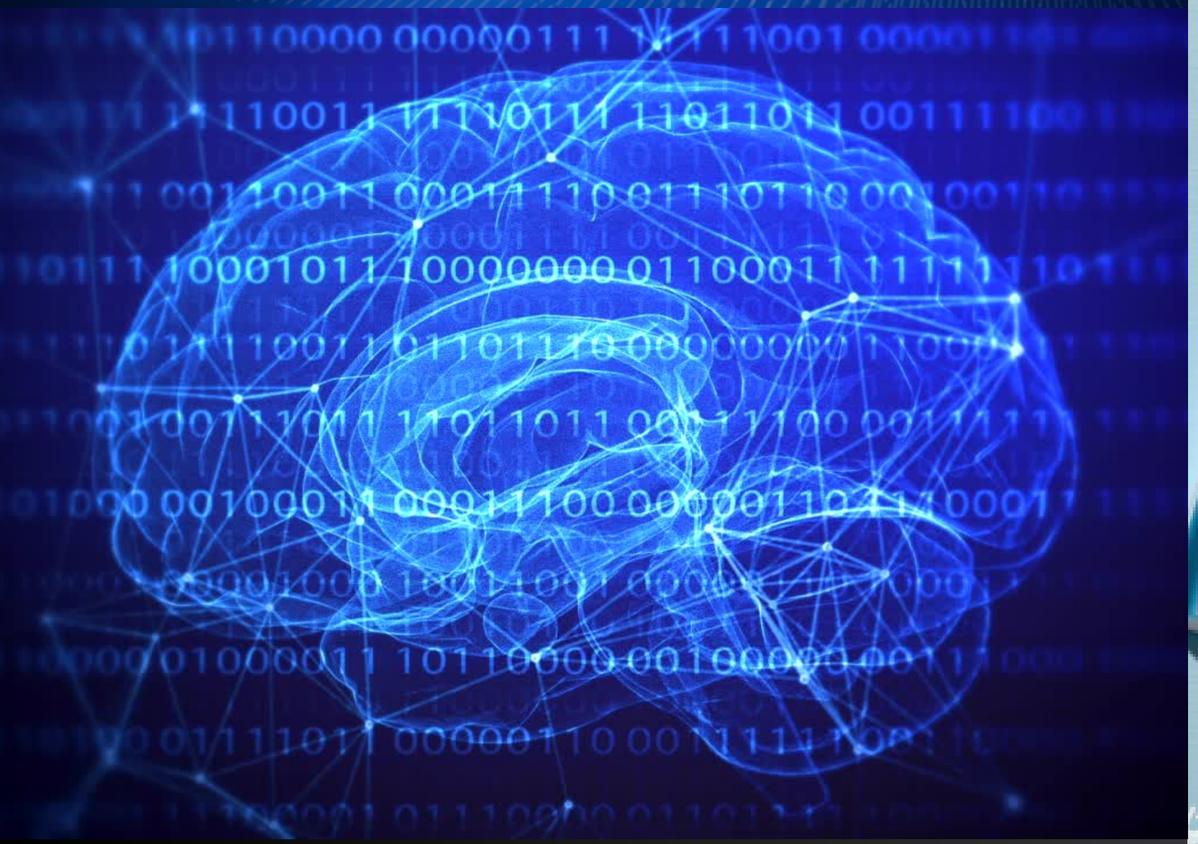
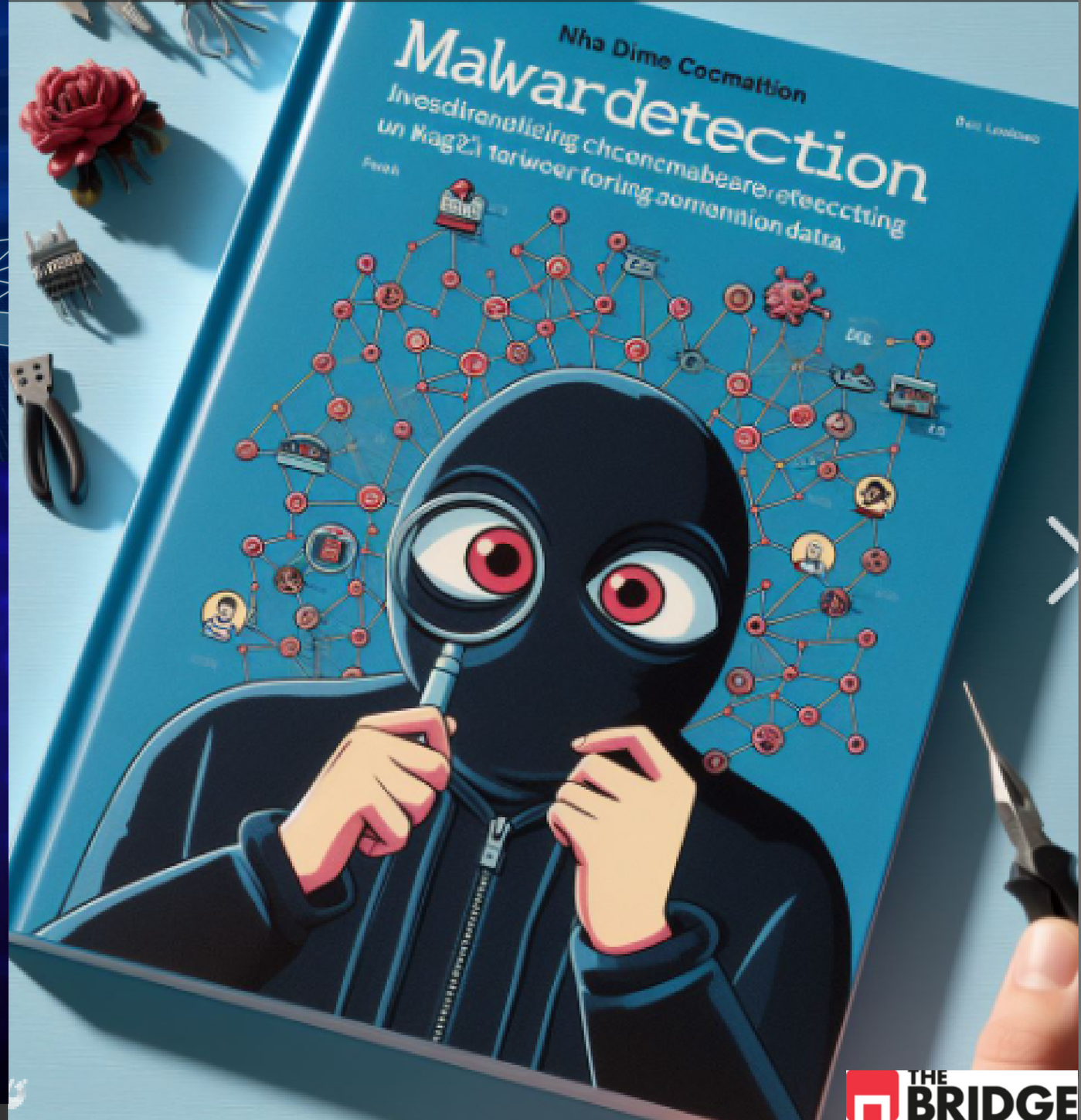
EL APRENDIZAJE AUTOMATICO

ESTUDIO DE LA DETECCIÓN

ATAQUES MALWARE

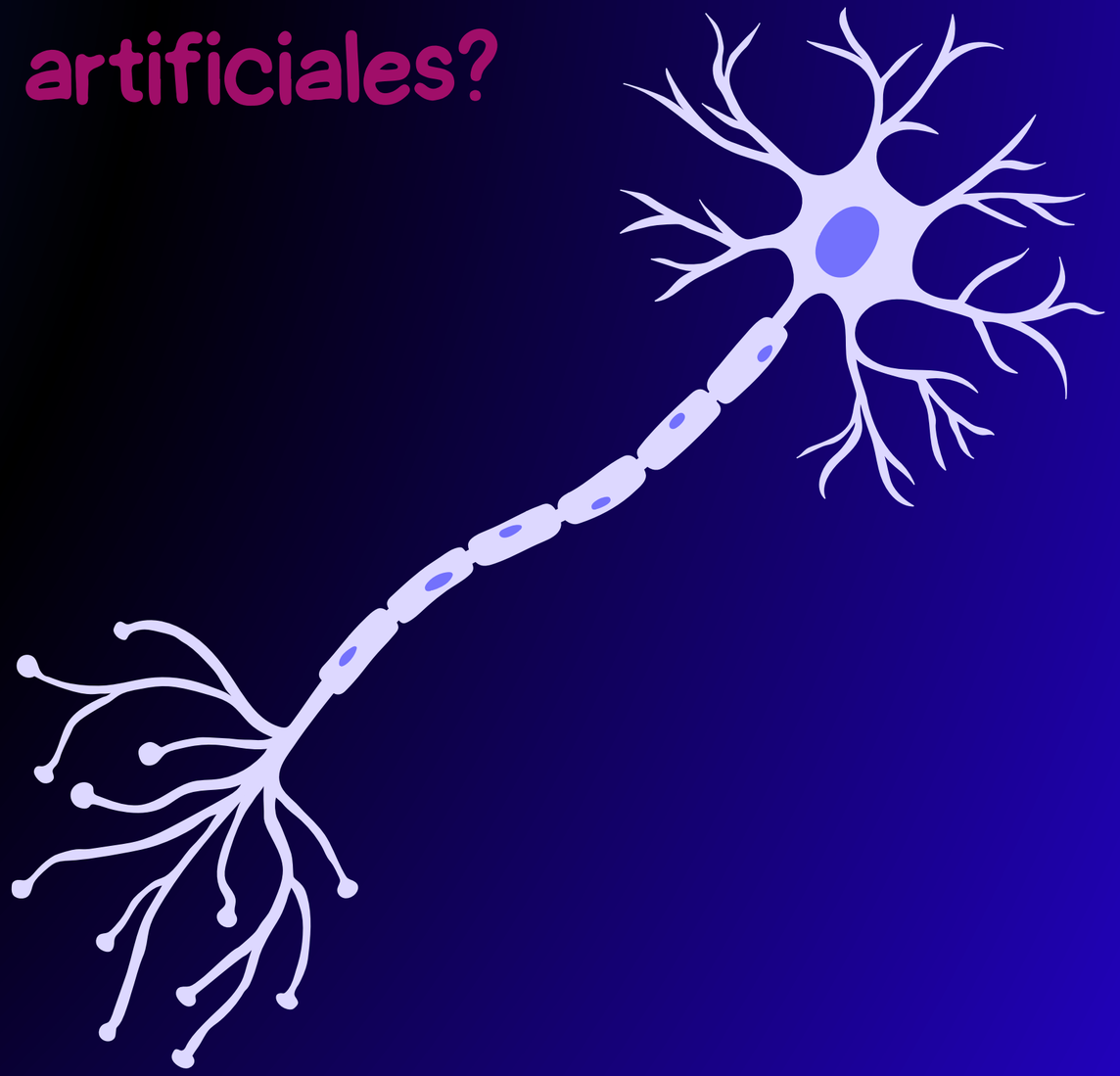




INTRODUCCIÓN

¿Qué son las neuronas?

¿Qué son las neuronas artificiales?



Breve historia de la IA

PROCESO

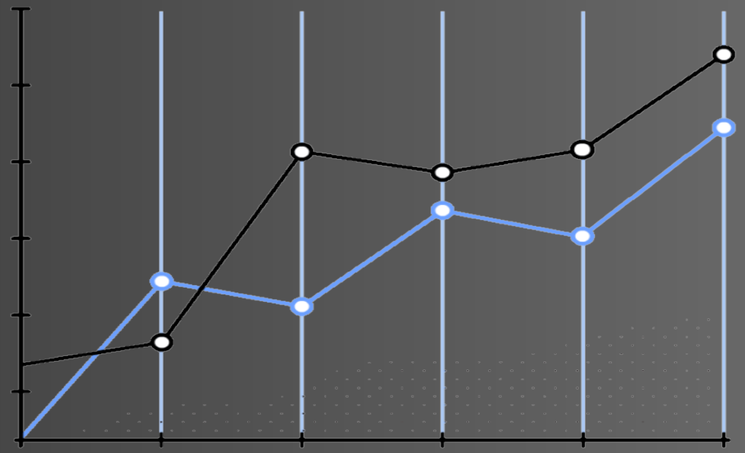
RECOPILACIÓN, PREPARACIÓN, ANÁLISIS Y MODELADO DE DATOS

COMENZAMOS

EL DATASET

Columna:	Descripción:	Tipo:	Orden de entrada:	Información posterior a la etiqueta "label"
id.resp_h	Identificador único del host de destino	Categórico	1	No
proto	Protocolo utilizado (TCP, UDP, ICMP, etc.)	Categórico	2	No
conn_state	Estado de la conexión (SYN, ACK, FIN, etc.)	Categórico	3	No
orig_pkts	Número de paquetes enviados por el host de origen	Numérico	4	No
orig_ip_bytes	Número de bytes enviados por el host de origen	Numérico	5	No
resp_pkts	Número de paquetes recibidos por el host de destino	Numérico	6	No
resp_ip_bytes	Número de bytes recibidos por el host de destino	Numérico	7	No
anyo	Año de la conexión	Numérico	8	No
mes	Mes de la conexión	Numérico	9	No
dia	Día de la conexión	Numérico	10	No
hora	Hora de la conexión	Numérico	11	No
id.resp_p	Identificador único del puerto de destino	Categórico	12	No
missed_bytes	Número de bytes perdidos en la conexión	Numérico	13	No
id.orig_p	Identificador único del puerto de origen	Categórico	14	No
hostory	conexiones y patrones detectados de conexiones anteriores	Categórico	15	No
label	Etiqueta que indica si la conexión es normal o anómala	Categórica	Target	

MINI- EDA

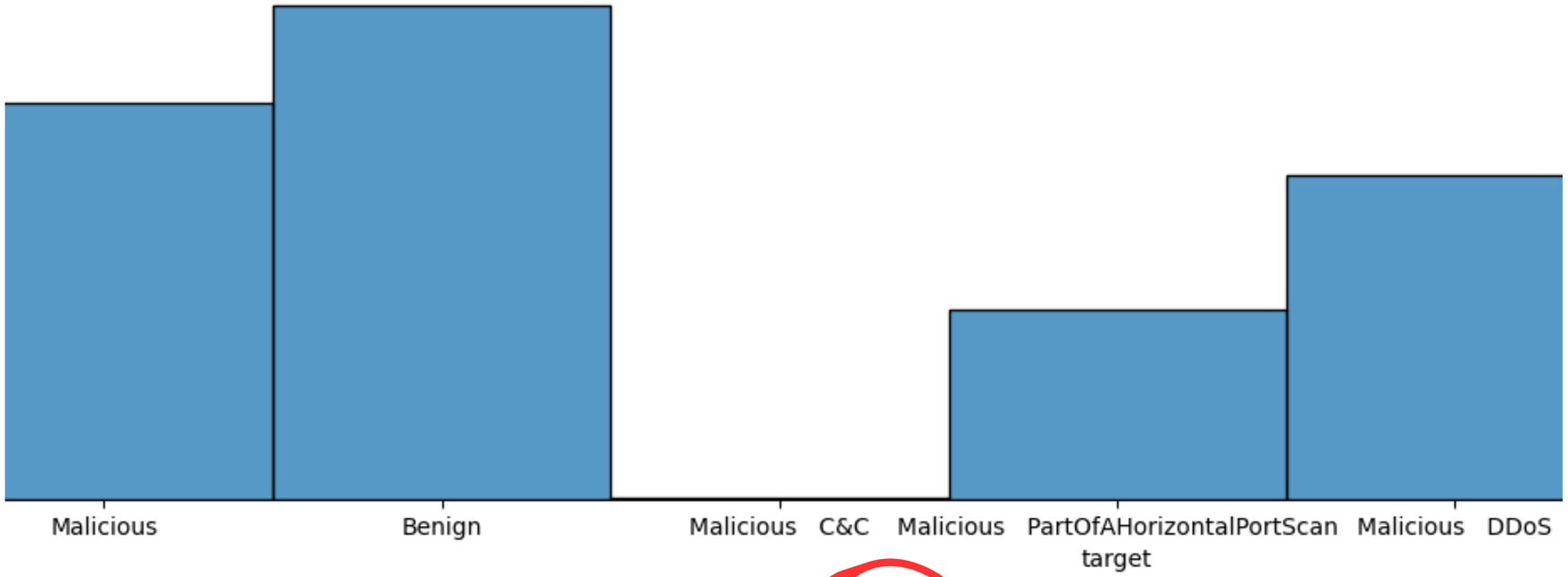


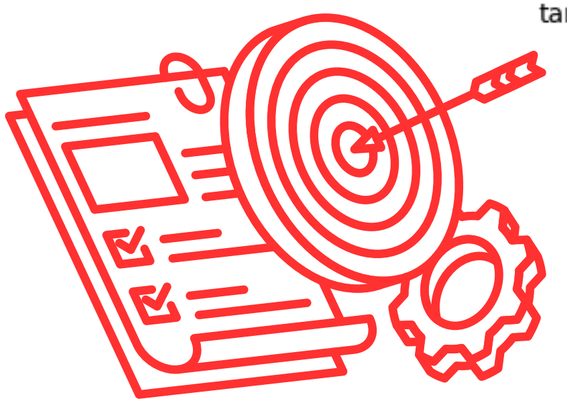
TARTAMIENTO DE NULOS

COLUMNA	NULOS
ts	0
uid	0
id.orig_h	0
id.orig_p	0
id.resp_h	0
id.resp_p	0
proto	0
service	250
duration	1595
orig_bytes	1595
resp_bytes	1595
conn_state	0
local_orig	0
local_resp	0
missed_bytes	0
history	25260
orig_pkts	0
orig_ip_bytes	0
resp_pkts	0
resp_ip_bytes	0
tunnel_parents	0
label	0
detailed-label	664504

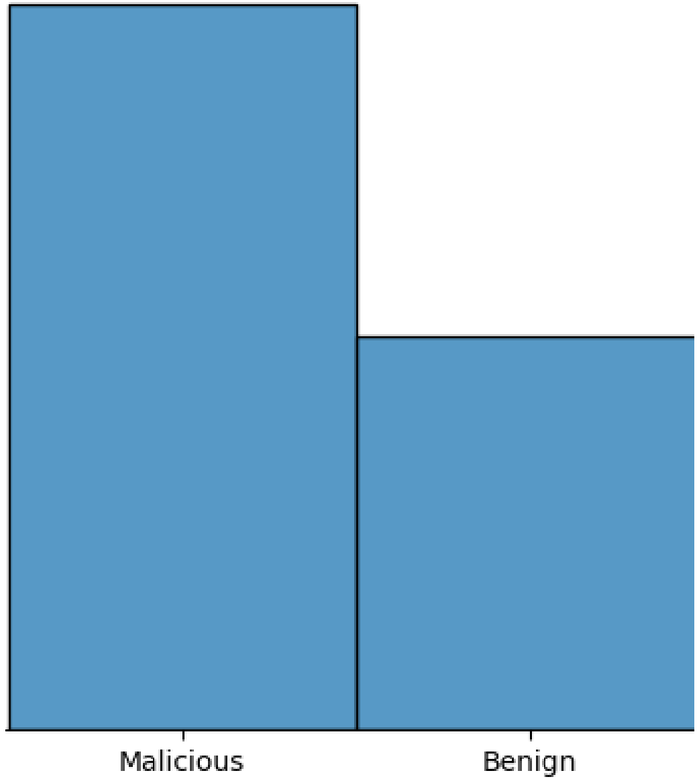
TARGET

SIN TRATAR

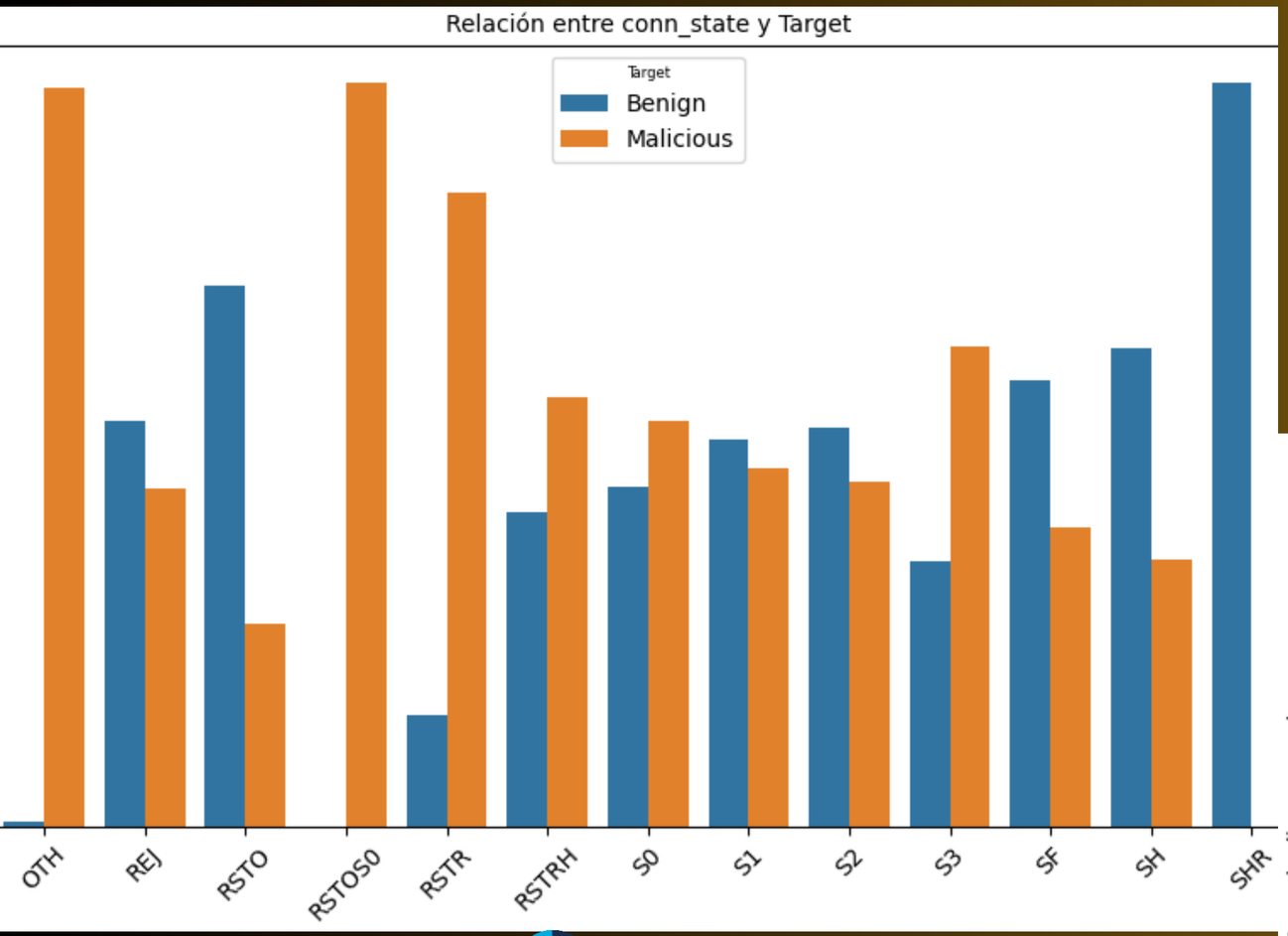


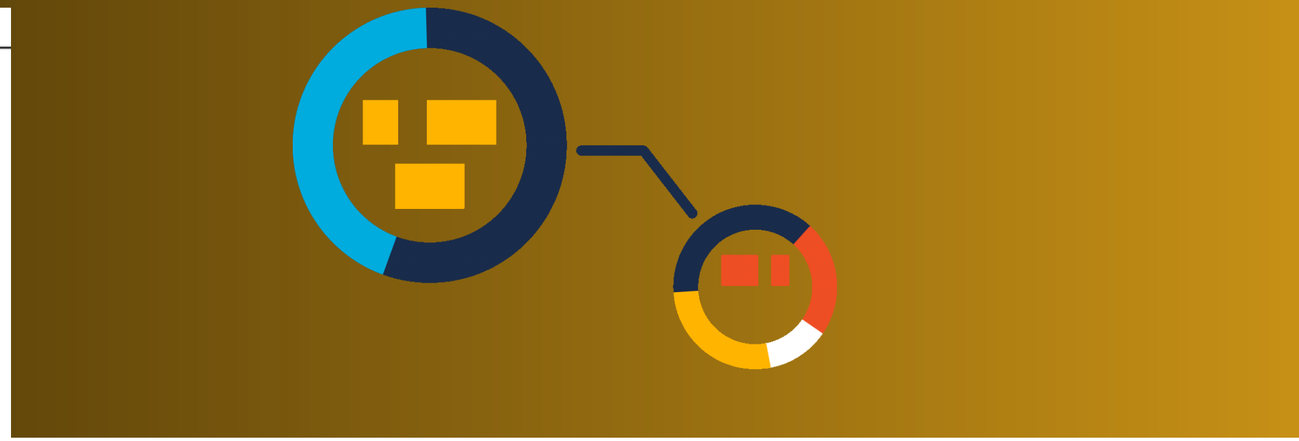


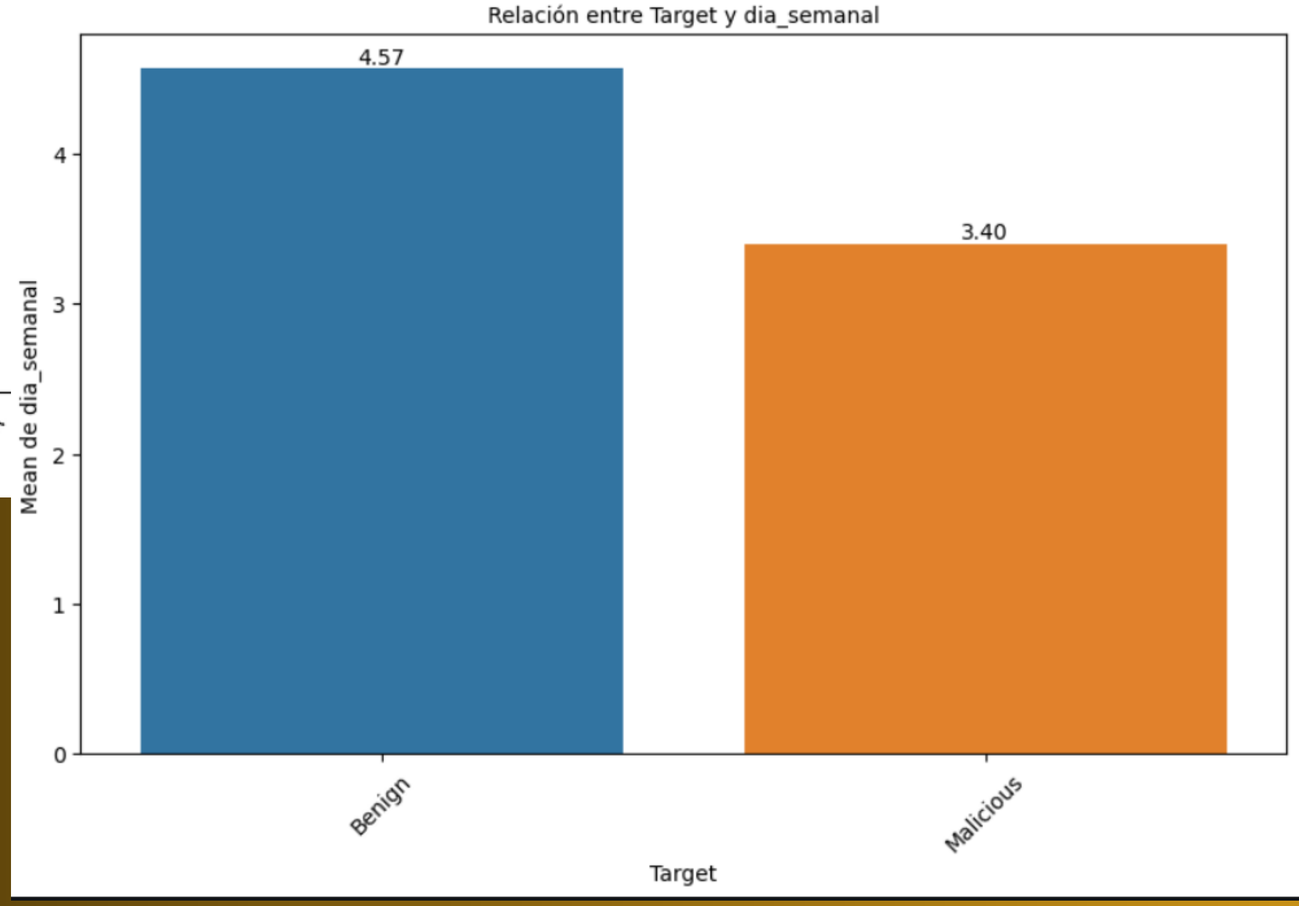
TRATADA



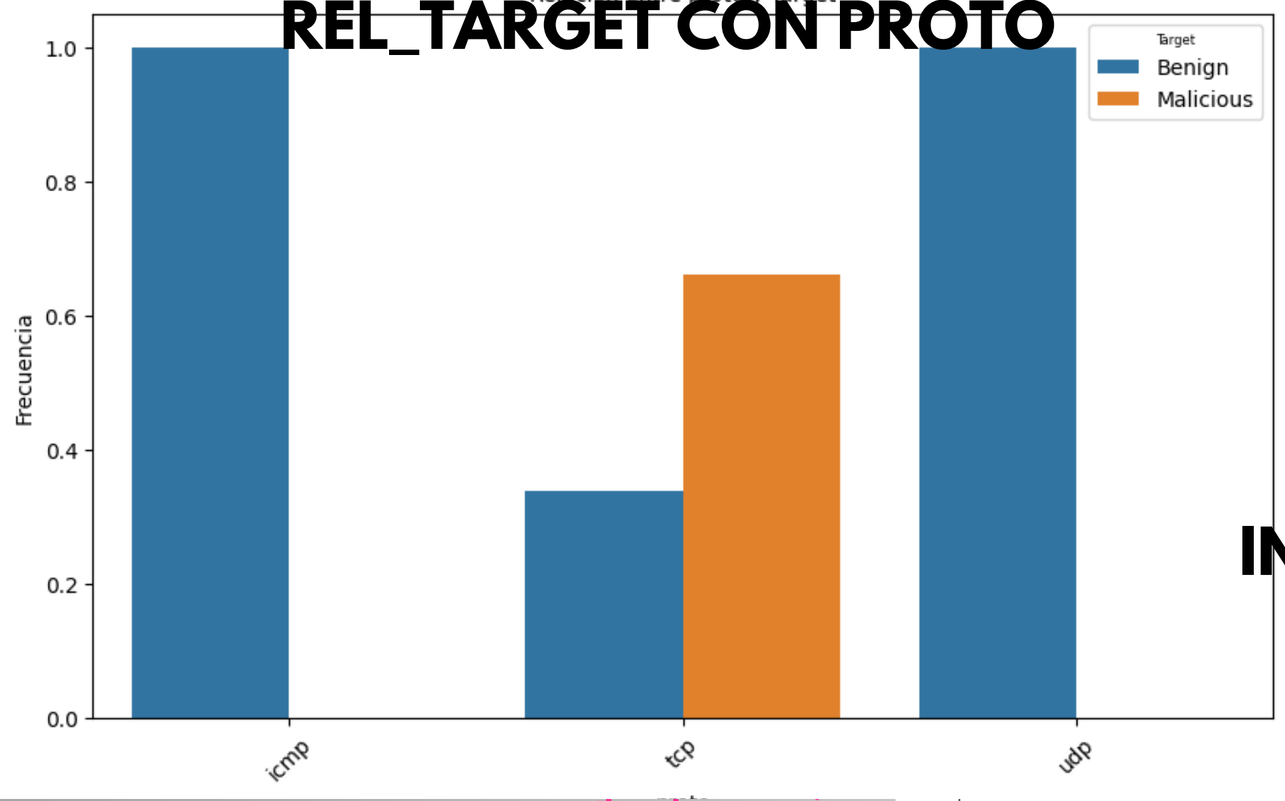
RELACIONES TARGET CON COLUMNAS DIA_SEMANAL Y CONN_STATE



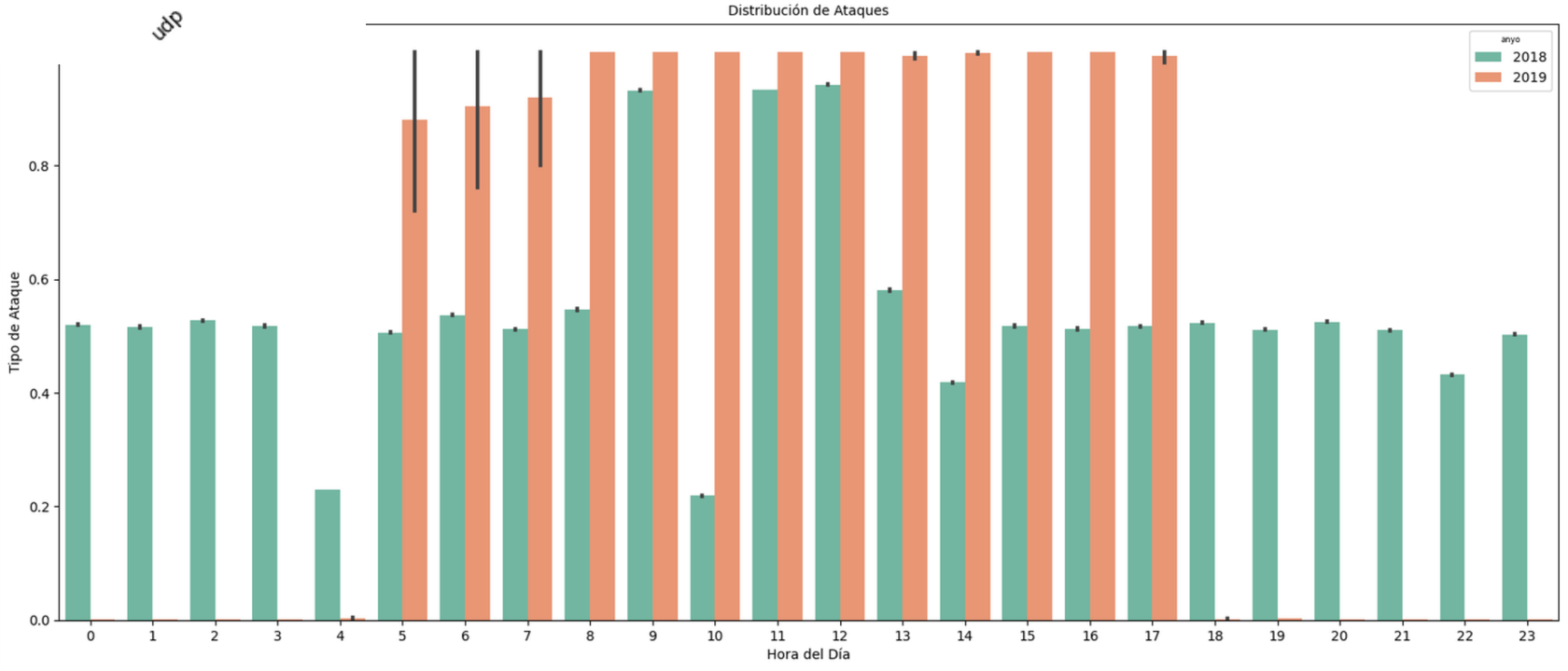




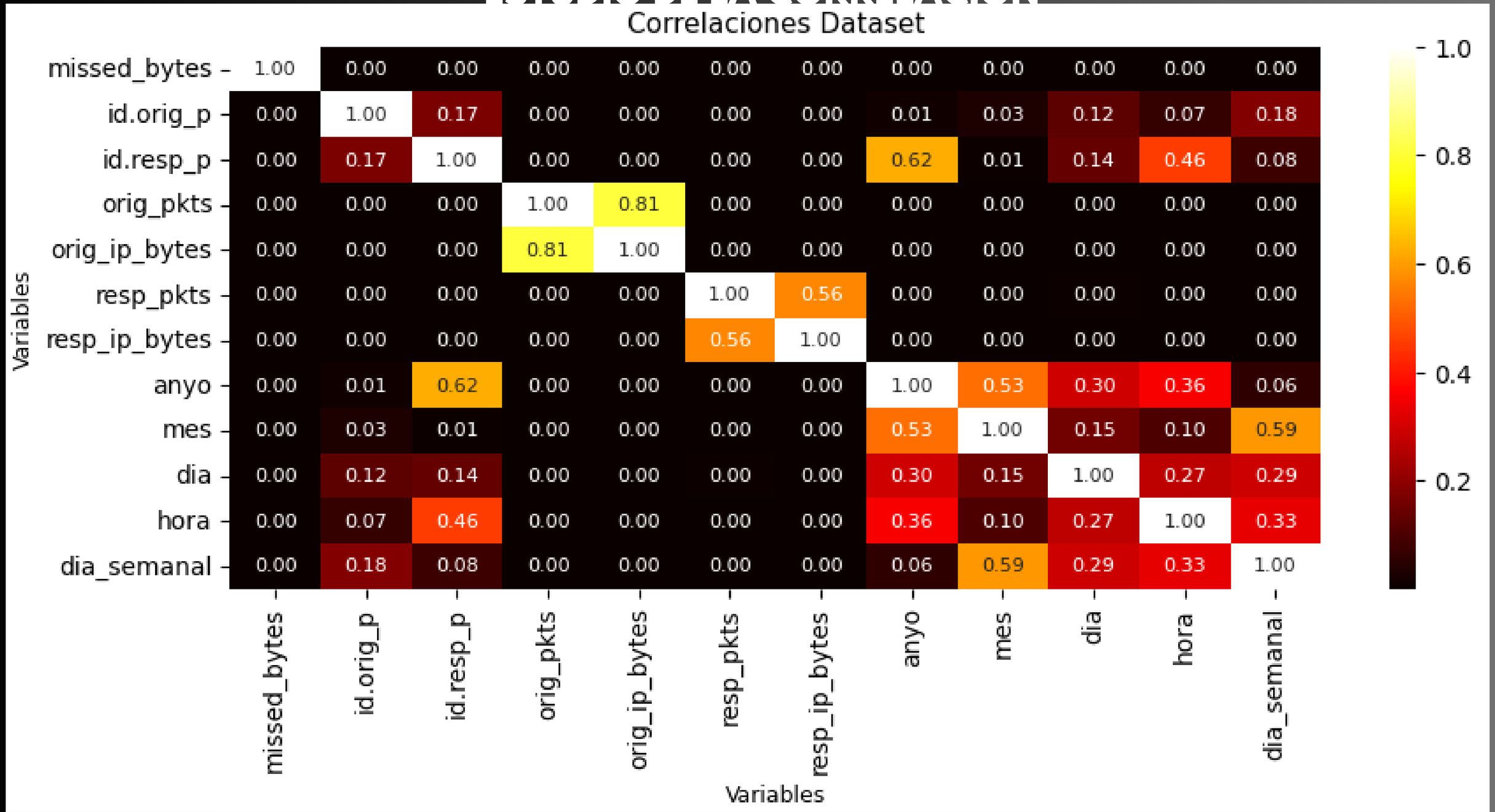
REL_TARGET CON PROTO

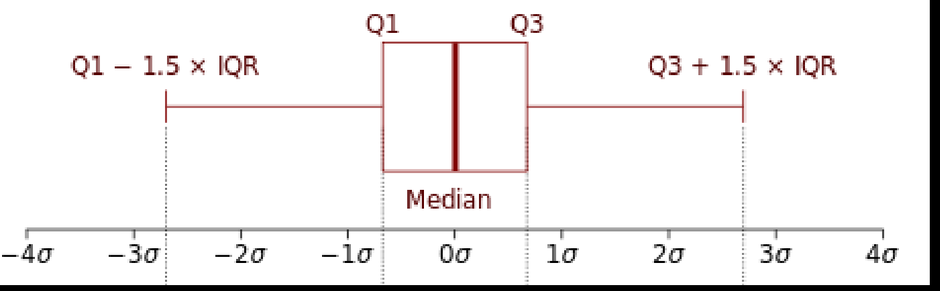


INTERVALO DE ATAQUES POR HORA Y AÑO



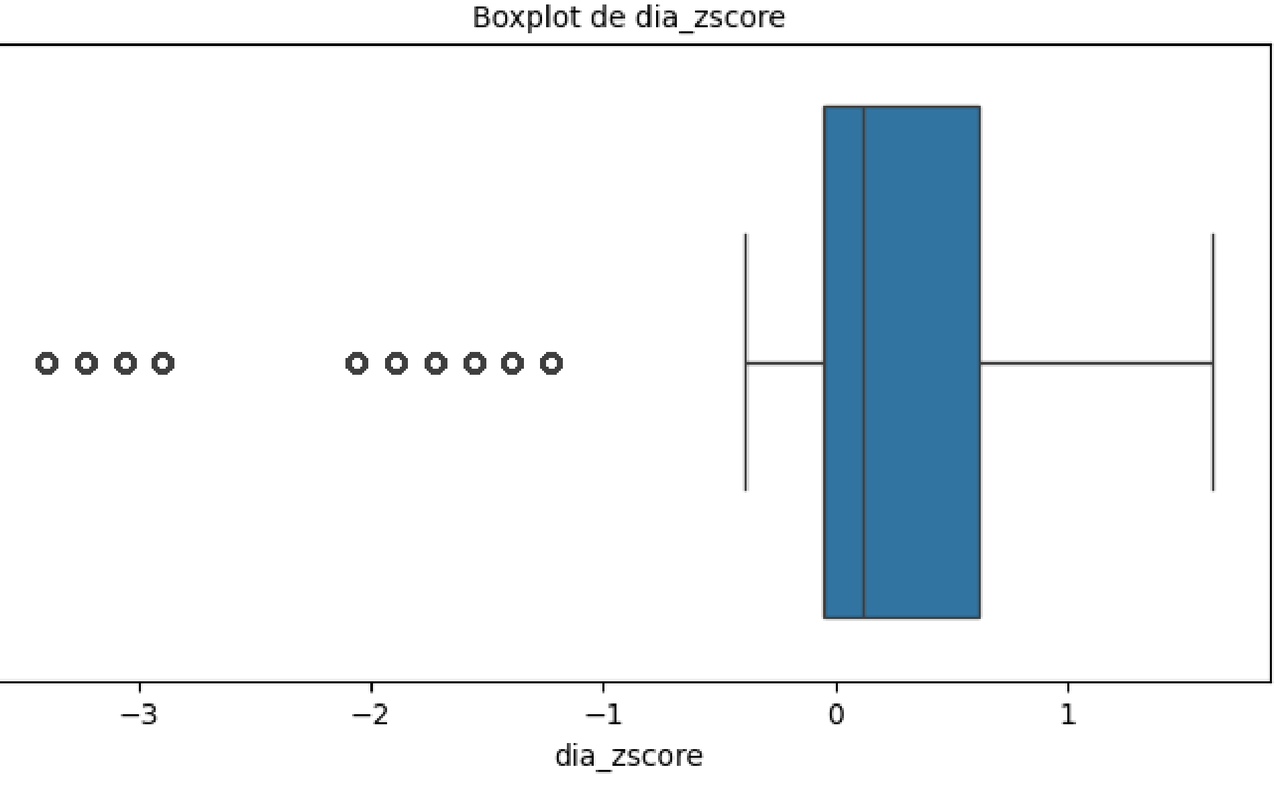
ESTUDIO DE LA CORRELACION



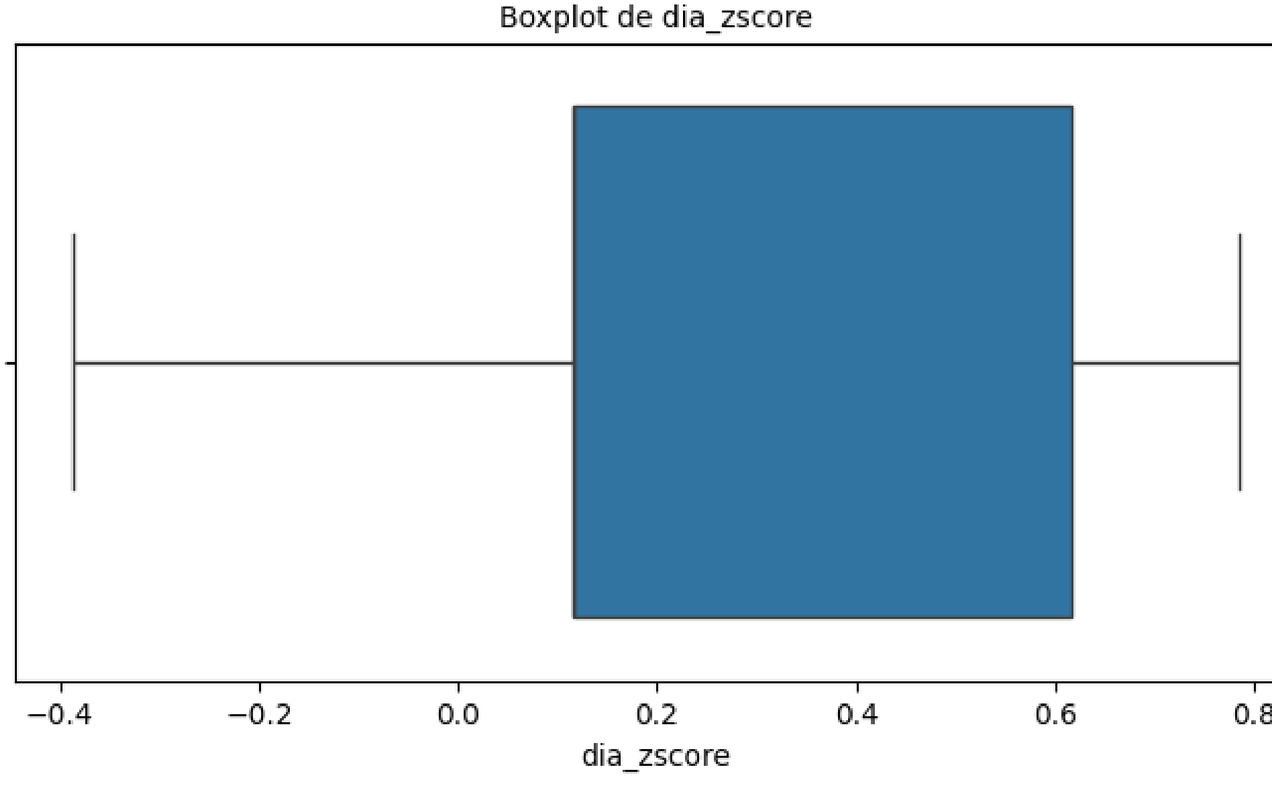


BUSQUEDA Y ELIMINACION DE OUTLIERS

ANTES DE ELIMINACION DE OUTLIERS



TRAS LA ELIMINACION DE OUTLIERS



FIN DEL EDA Y CUADRO DE COLUMNAS FINALES

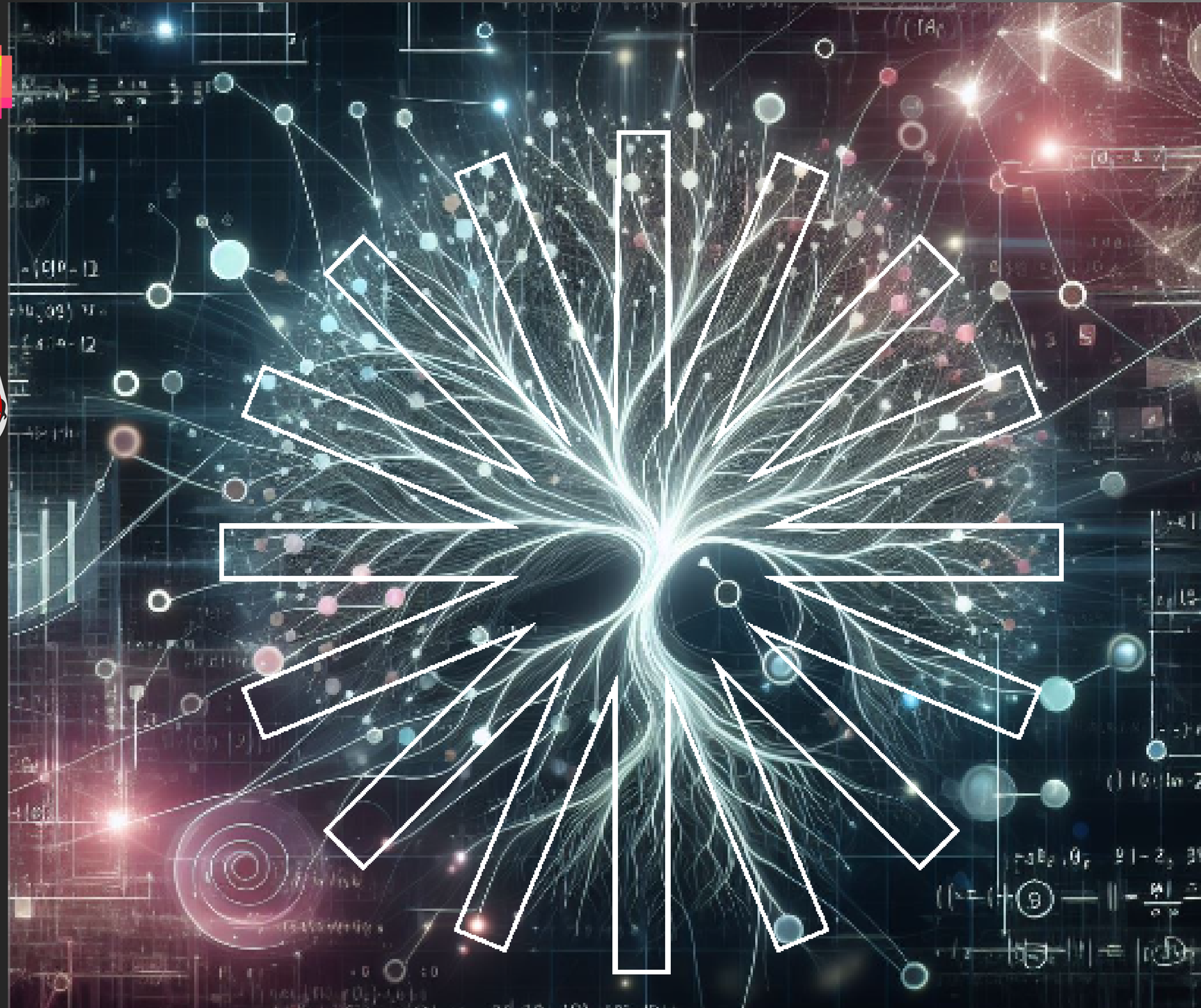
The end

Columna:	Descripción:	Tipo:	Orden de entrada:	Información posterior a la etiqueta "label"
id.resp_h	Identificador único del host de destino	Categórico	1	No
proto	Protocolo utilizado (TCP, UDP, ICMP, etc.)	Categórico	2	No
conn_state	Estado de la conexión (SYN, ACK, FIN, etc.)	Categórico	3	No
history	Historial de tráfico relacionado con esta conexión o sesión, pudiendo obtener patrones de actividad	Categórico	4	No
id.orig_p	Mes de la conexión	Numérico	5	No
orig_pkts	Número de paquetes enviados por el host de origen	Numérico	6	No
resp_pkts	Número de paquetes recibidos por el host de destino	Numérico	7	No
anyo	Año de la conexión	Numérico	8	No
hora	Hora de la conexión	Numérico	9	No
Target	Etiqueta que indica si la conexión es normal o anómala	Categórica	10	Target

2

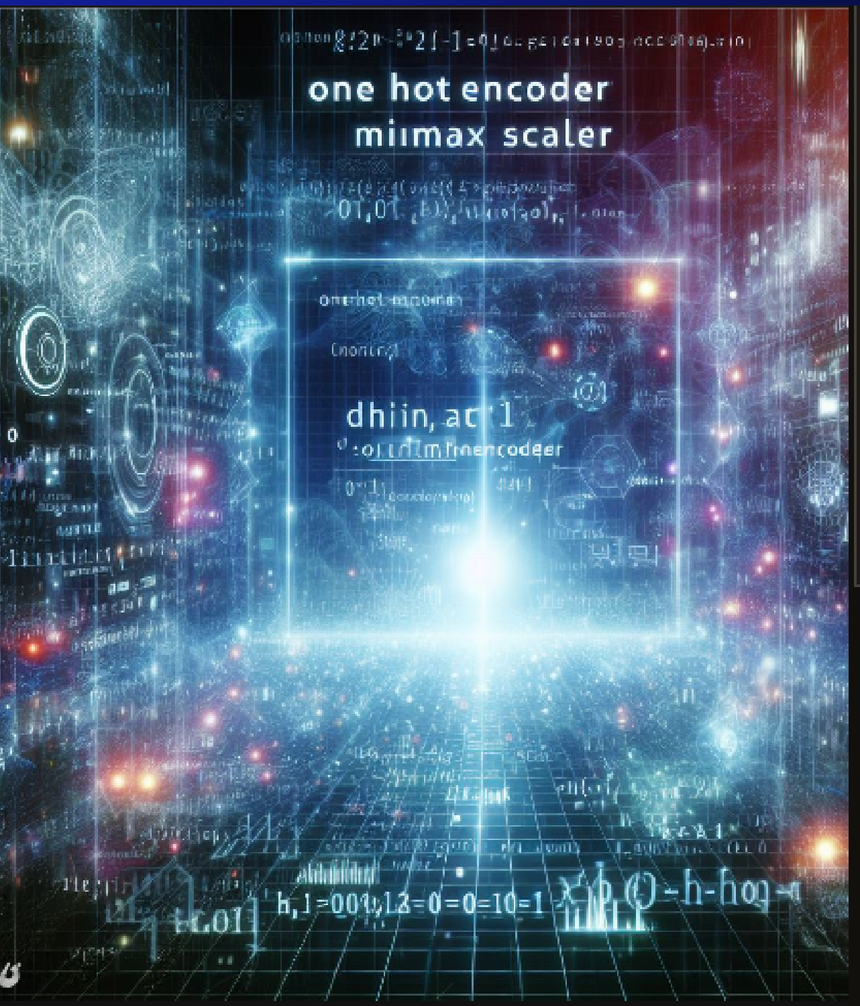
START

MACHINE LEARNING



1.- MODELO ML RANDONFOREST CON 115 COLUMNAS

PROCESO TRANSFORMACION



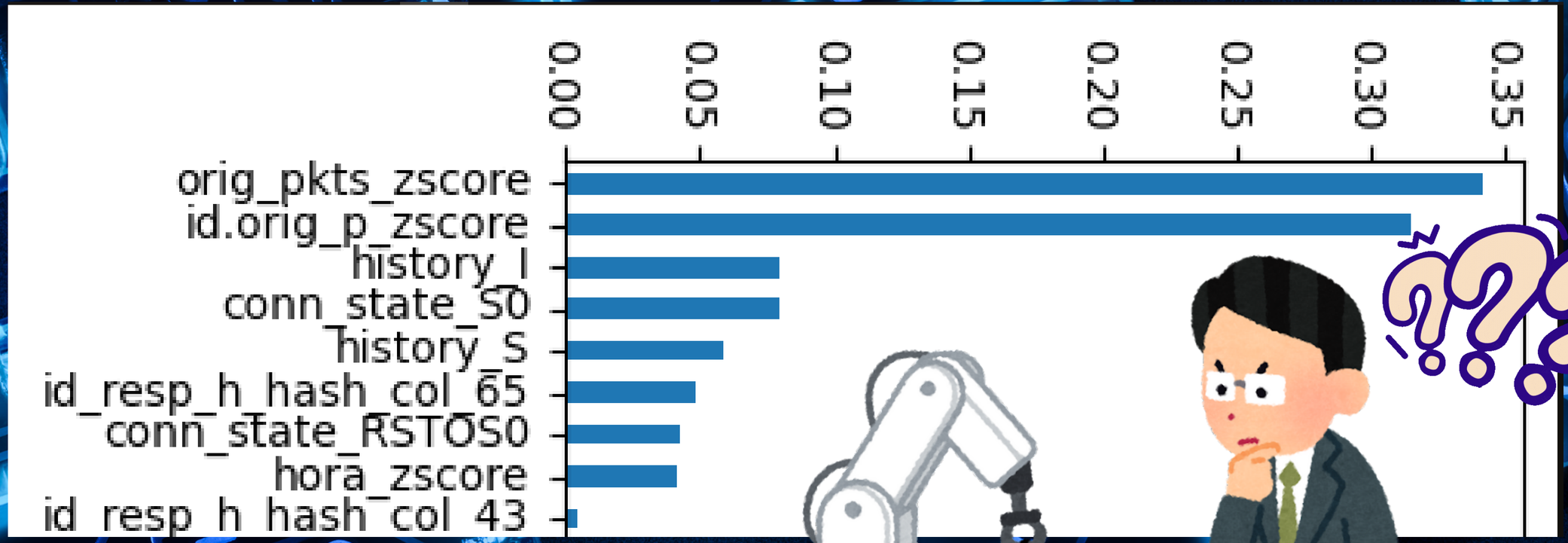
RESULTADO

```
model = RandomForestClassifier(n_estimators=100, max_depth=10, random_state=42)

# Entrenar el modelo
model.fit(X_train, y_train)

# Evaluar el modelo
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)

print(classification_report(y_pred, y_test))
```

✓ 22m 42.0s

	precision	recall	f1-score	support
0	0.59	1.00	0.74	812036
1	1.00	0.74	0.85	2224292
accuracy			0.81	3036328
macro avg	0.79	0.87	0.80	3036328
weighted avg	0.89	0.81	0.82	3036328

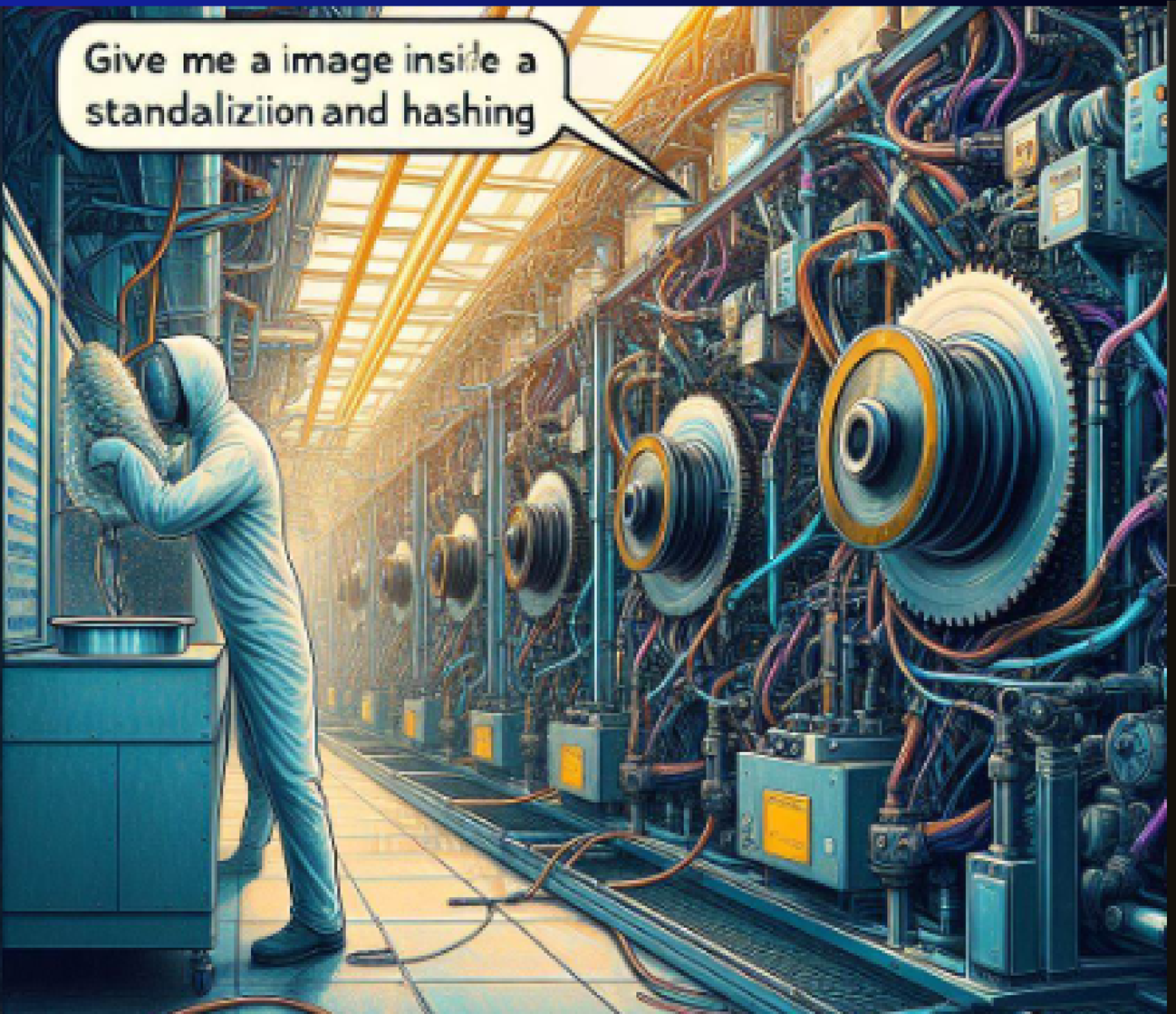
CARACT. MÁS INFLUYENTES DURANTE EL R.FOREST.



....DE 115 COLUMNAS....

2.- MODELO ML LOGISTICREGRESSION CON 115 COLUMNAS

PROCESO TRANSFORMACION



RESULTADO

```python
# Definir el modelo logisctic con C para regule el sobre
model = LogisticRegression(C=10, solver="sag")

# Entrenar el modelo
model.fit(X_train, y_train)

# Evaluar el modelo
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)

print(classification_report(y_pred, y_test))
```

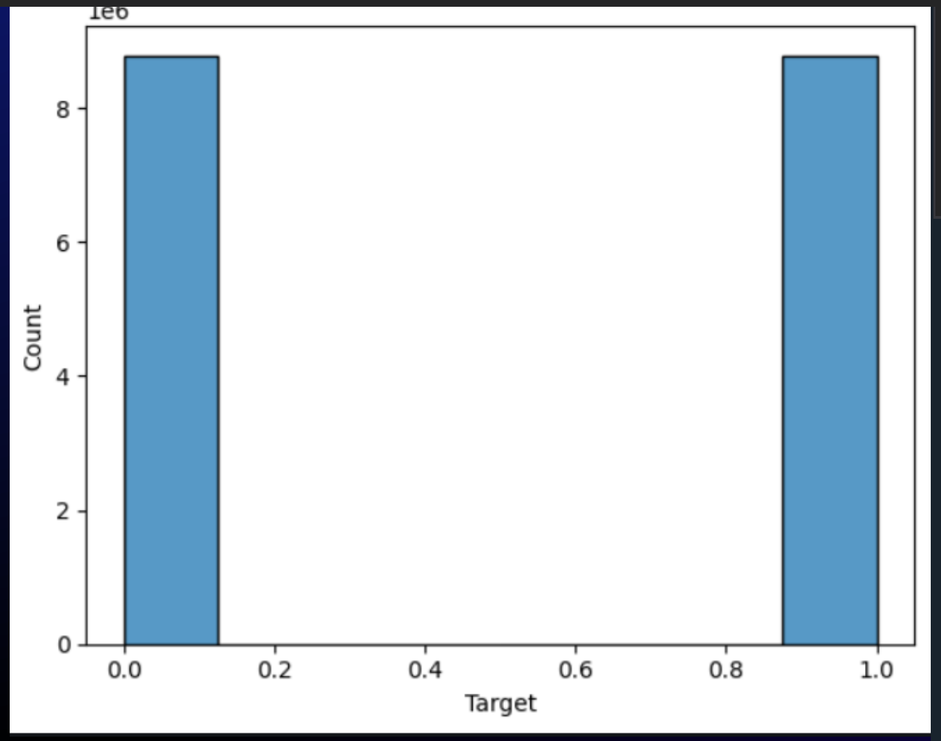
✓ 5m 11.2s

	precision	recall	f1-score	support
0.0	0.59	0.71	0.64	1159710
1.0	0.79	0.70	0.74	1876618
accuracy			0.70	3036328
macro avg	0.69	0.70	0.69	3036328
weighted avg	0.72	0.70	0.70	3036328

2.-SUBMUESTREO DEL DATASET Y CLASE MINORITARIA

RESAMPLED

```
# Aplica submuestro, para reducir el número de obser
X=df_out.drop(["Target"], axis=1)
y=df_out["Target"]

rus = RandomUnderSampler(random_state=42)
X_resampled, y_resampled = rus.fit_resample(X, y)
```



MODELO LOGISITC REGRESSION

```
# Definir el modelo
model = LogisticRegression(solver='sag', max_iter=100, n_jobs=-1)

# Entrenar el modelo
model.fit(X_train, y_train)

# Predecir
y_pred = model.predict(X_test)

# Evaluar el modelo
print(classification_report(y_test, y_pred))
```

✓ 23m 52.3s

```
c:\Users\victo\anaconda3\envs\gpu-dask\lib\site-packages\sklearn\linear
  warnings.warn(
              precision    recall  f1-score   support

           0       0.73      0.62      0.67   1384044
           1       0.67      0.77      0.72   1385471

    accuracy                           0.70   2769515
   macro avg       0.70      0.70      0.69   2769515
weighted avg       0.70      0.70      0.69   2769515
```

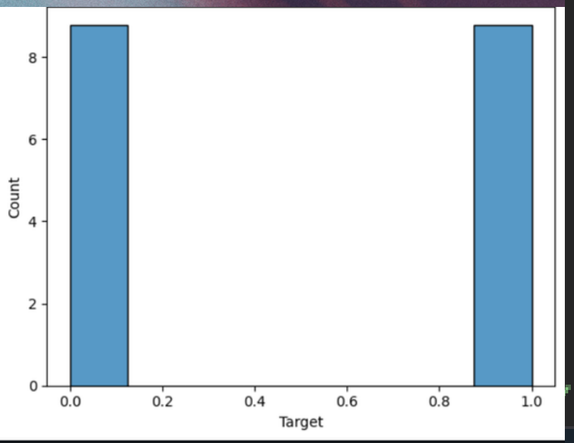
3.- SUBMUESTREO DEL DATASET Y CLASE MINORITARIA

```
C:\Users\...\Anaconda3\envs\geo\lib\site-packages\
  warnings.warn(
              precision    recall  f1-score   support

           0       0.73      0.62      0.67   1384044
           1       0.67      0.77      0.72   1385471

    accuracy                           0.69   2769515
   macro avg       0.70      0.69      0.69   2769515
weighted avg       0.70      0.69      0.69   2769515
```

```
# Aplica submuestro, para reducir el número de obser
X=df_out.drop(["Target"], axis=1)
y=df_out["Target"]
```

RESAMPLED

```
rus = RandomUnderSampler(random_state=42)
X_resampled, y_resampled = rus.fit_resample(X, y)
```




```
# Definir el modelo
model_rf = RandomForestClassifier(n_estimators=100, max_depth=10, random_state=42, n_jobs=-1)

# Entrenar el modelo
# Usar parallel_backend para aprovechar la multiprocesamiento
with parallel_backend('threading'):
    model.fit(X_train, y_train)

# Predecir
y_pred = model.predict(X_test)

# Evaluar el modelo
print(classification_report(y_test, y_pred))
```

25m 20.4s

MODELO RANDOMFOREST

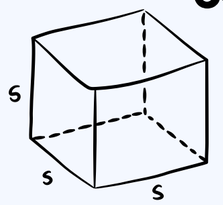
APLICACION DE TECNICAS DE DEEP-LEARNING

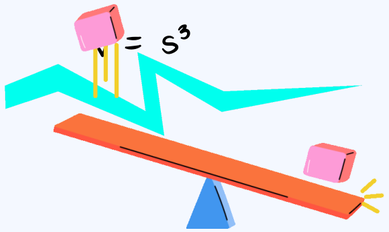
3.

MI MODELO DE RED NEURONAL 1

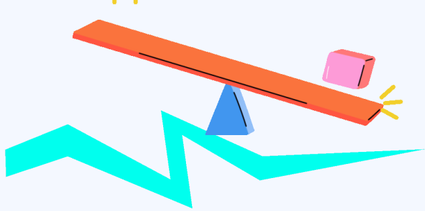
CAPAS OCULTAS
64-32-16-8 Y L2 1 CAPA Y 2 DENSA

CAPA DE ENTRADA
64 UNITS

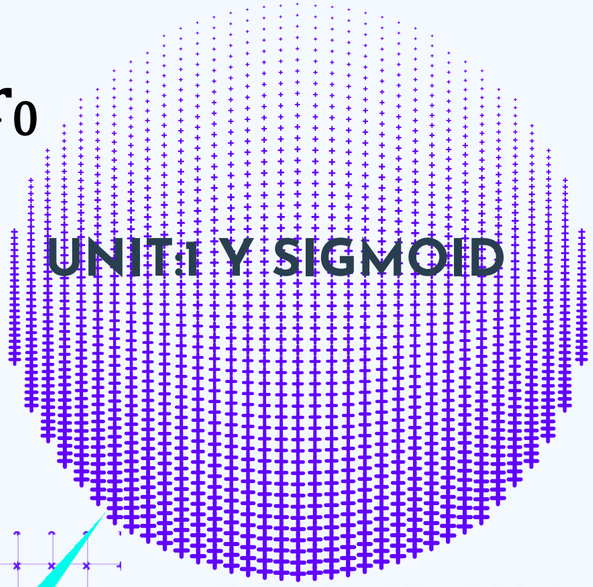




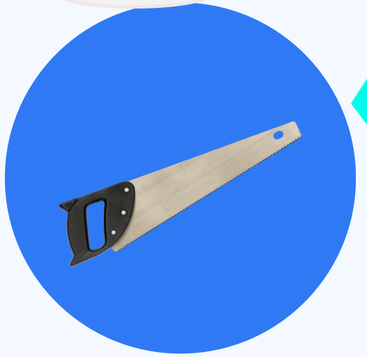
$$\mathbf{r} = \frac{1}{2}\mathbf{a}t^2 + \mathbf{v_0}t + \mathbf{r_0}$$

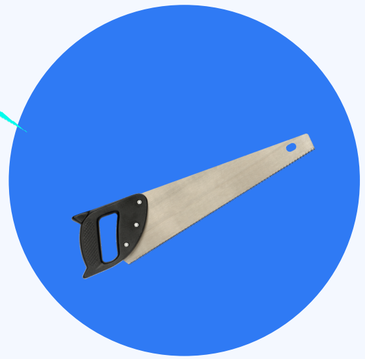


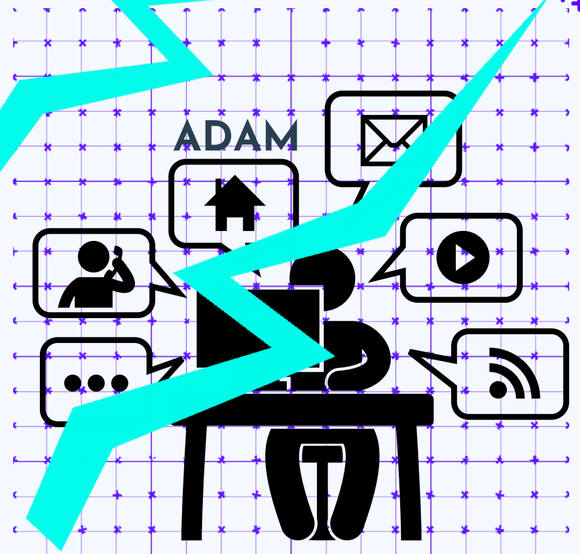
CAPA DE SALIDA



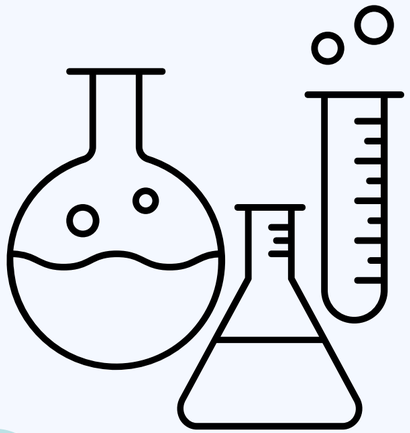
20%







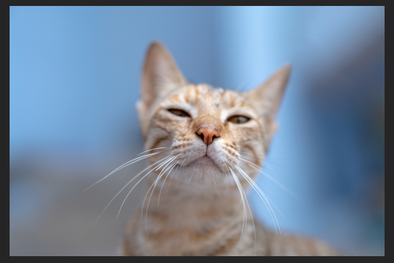
EARLY_STOPPING



REDUCE_LR

START

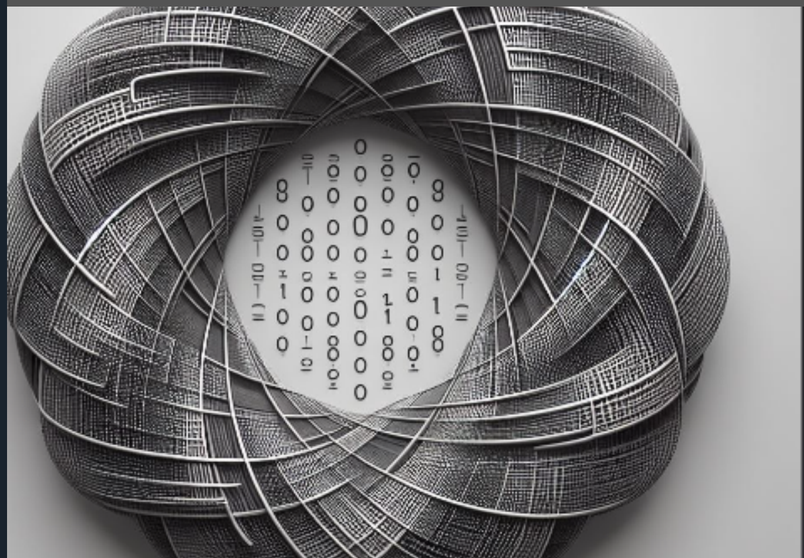
```python
modelo = tf.keras.Sequential()
# Definir las 4 entradas
modelo.add(tf.keras.layers.Dense(units=64, activation='relu', input_shape=(115,), kernel_regularizer=l2(0.01)))
modelo.add(tf.keras.layers.BatchNormalization())#mejora la tasa de parendizaje, evitando sobreajuste, ya que tb regulariza la funcion

# Capas ocultas
modelo.add(tf.keras.layers.Dense(units=32, activation="relu", input_dim=115))#relu, la mas comun,rapida y eficiente
modelo.add(tf.keras.layers.BatchNormalization())#mejora la tasa de parendizaje, evitando sobreajuste, ya que tb regulariza la funcion

modelo.add(Dropout(0.2))

modelo.add(tf.keras.layers.Dense(units=16, activation="elu", kernel_regularizer=l2(0.01)))
modelo.add(tf.keras.layers.BatchNormalization())

modelo.add(tf.keras.layers.Dense(units=8, activation='elu'))#eLU mejora la convergencia y la velocidad de entrenamiento
modelo.add(tf.keras.layers.BatchNormalization())

# Capa de salida
modelo.add(tf.keras.layers.Dense(units=1, activation='sigmoid'))  # sigmoide para clasificacion binaria
#optimizador
optimizador=Adam( learning_rate=0.01, # rebajo el learning rate por ser mas adecuado para un ajuste fino aunque tarde mas
                  beta_1=0.6, # influye en el gradiente pasado (1 mas al gradiente y 0 nada)
                  beta_2=0.4, #influye en los cuadrados de los gradientes pasados(1 mas a la varianza y 0 nada)
                  epsilon=1e-09, #es para prevenior divisiones entre 0
                  amsgrad=False)#es una variante de Adam y ayuda con la convergencia de ambos a evitar oscilaciones de los pesos durante la optimizacion"""
# Compilar el modelo
modelo.compile(optimizer='adam',
               loss='binary_crossentropy',
               metrics=['accuracy', 'Precision', 'Recall'])
tf.keras.metrics.AUC(name='auc')

#detiene el entrenamiento si la metrica no mejora
early_stopping_callbacks = tf.keras.callbacks.EarlyStopping(patience=5,restore_best_weights=True)

#reduce la tasa de aprendizaje cuando el rendimiento del conj. val no mejora
reduce_lr_callbacks = ReduceLROnPlateau(monitor='val_loss', patience=2, verbose=1)
#Guarda el modelo cuando mejora la métrica de validación
filepath = r'D:\Cursos\REPOSITORIOS\DATASET\malware_total\logs\modelo.h5'
monitor = 'val_loss'
checkpoint_callbacks = ModelCheckpoint(filepath=filepath, monitor=monitor, verbose=1, save_best_only=True)
#visionado_tensorboard
tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir=log_dir)

callbacks_list = [early_stopping_callbacks, reduce_lr_callbacks, checkpoint_callbacks, tensorboard_callback]
# Entrenar el modelo
historia = modelo.fit(X_train_a, y_train_a, batch_size=32, epochs=10,verbose=1,validation_split=0.1,callbacks=callbacks_list)
```

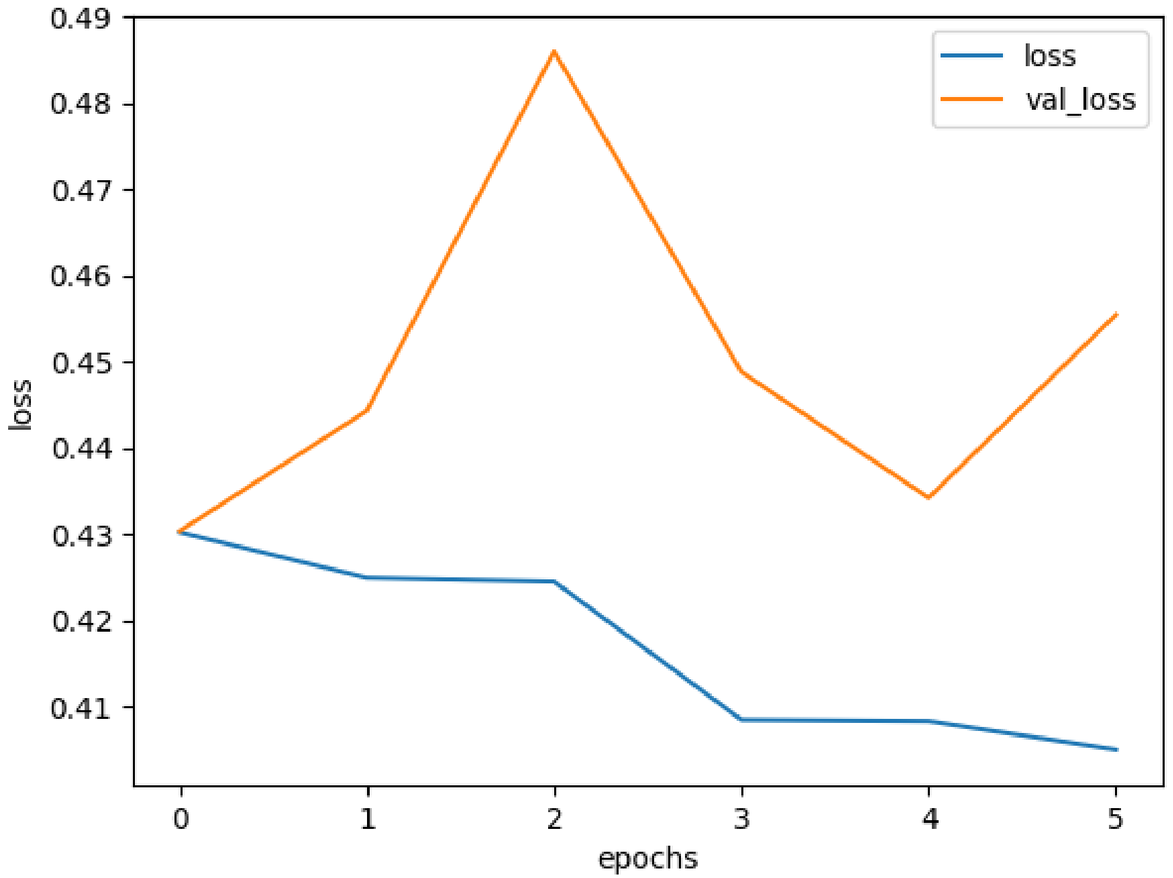


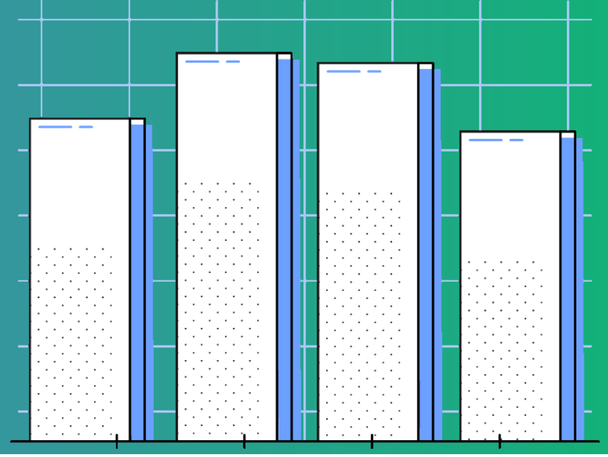
modelo de red neuronal 1

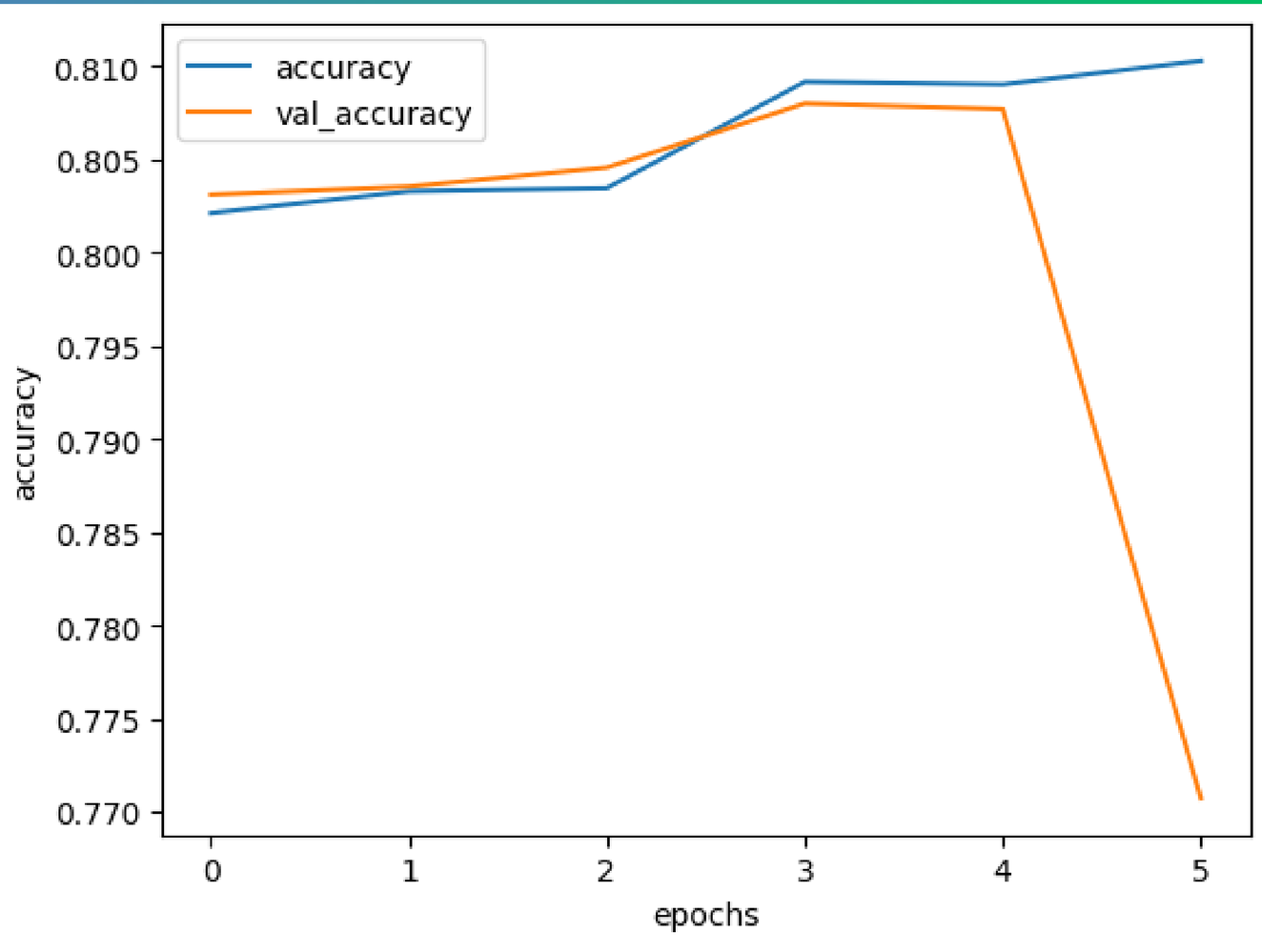
Adam y desbalaceada

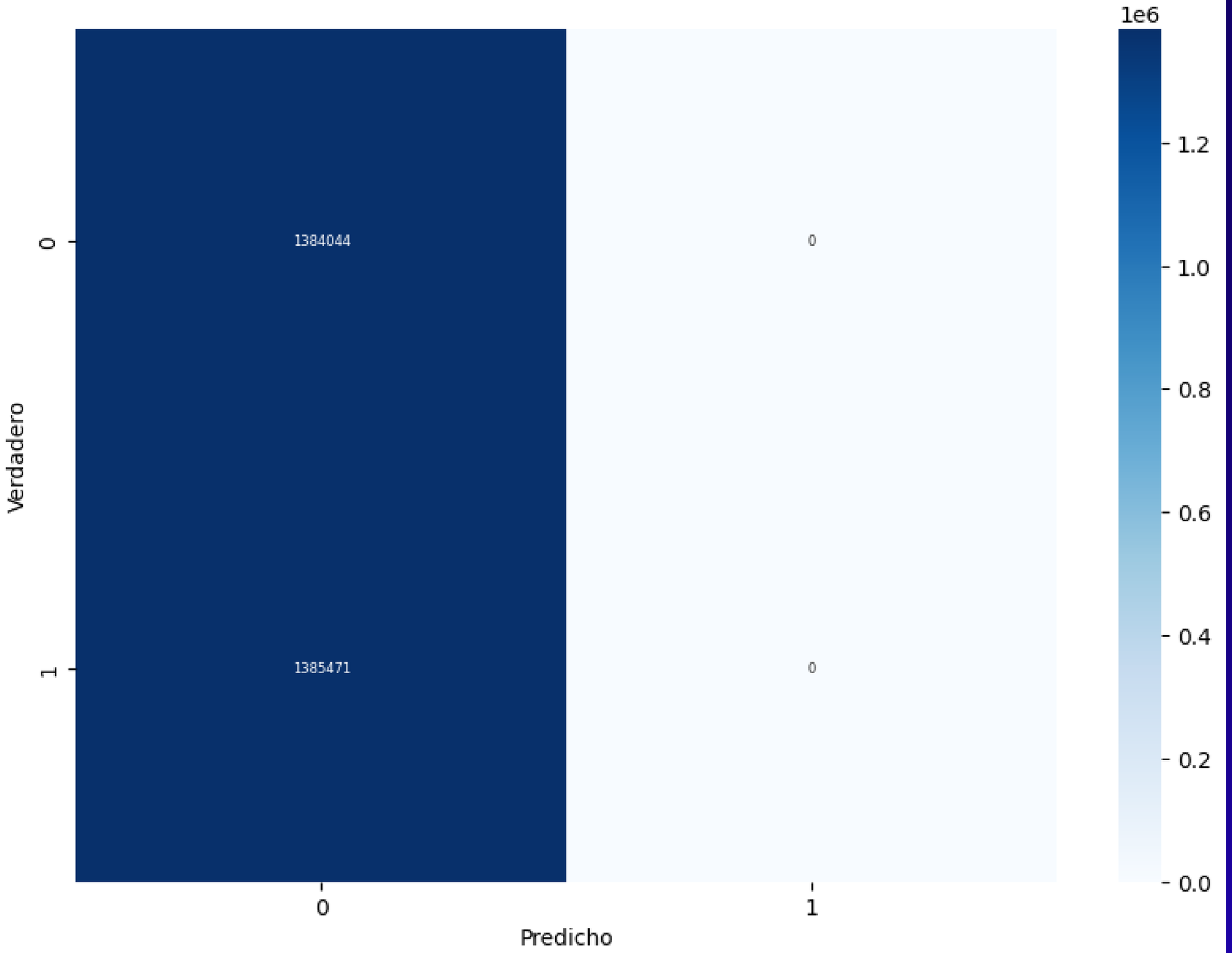
```
94886/94886 [==============================] - 93s 976u
Loss: 0.4293183386325836
Accuracy: 0.8039270043373108
precision: 0.7358649373054504
recall: 0.9977438449859619
94886/94886 [==============================] - 62s 650u
c:\Users\victo\anaconda3\envs\tf-gpu\lib\site-packages\
  _warn_prf(average, modifier, f"{metric.capitalize()}
c:\Users\victo\anaconda3\envs\tf-gpu\lib\site-packages\
  _warn_prf(average, modifier, f"{metric.capitalize()}
              precision    recall  f1-score   support

         0.0       0.46      1.00      0.63   1384396
         1.0       0.00      0.00      0.00   1651932

    accuracy                           0.46   3036328
   macro avg       0.23      0.50      0.31   3036328
weighted avg       0.21      0.46      0.29   3036328
```




loss
val_loss
0.49
0.48
0.47
0.46
0.45
0.44
0.43
0.42
0.41
loss
0
1
2
3
4
5
epochs

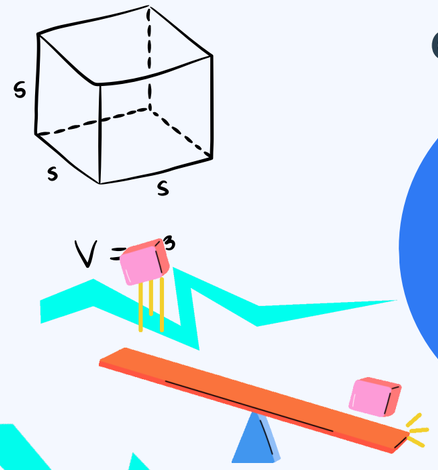



accuracy
val_accuracy
0.810
0.805
0.800
0.795
0.790
0.785
0.780
0.775
0.770
accuracy
0
1
2
3
4
5
epochs

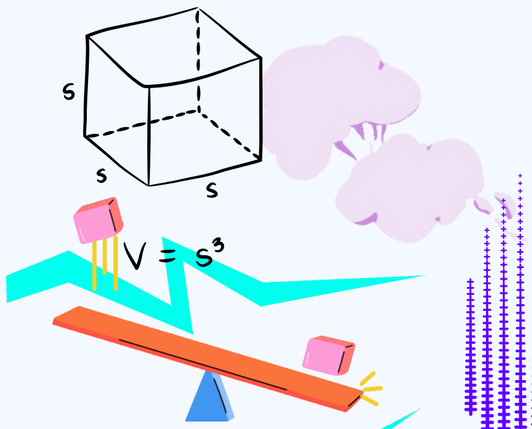

1e6
1.2
1.0
0.8
0.6
0.4
0.2
0.0
1384044
0
1385471
0
0
1
Verdadero
0
1
Predicho

MI MODELO DE RED NEURONAL 2

CAPA DE ENTRADA



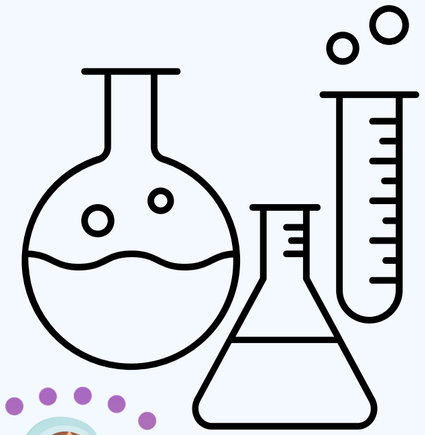
CAPAS OCULTAS



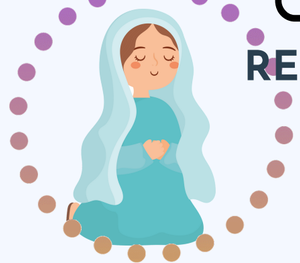
CAPA DE SALIDA

UNIT 1 Y SIGMOID

EARLY_STOPPING



REDUCE_LR



TODAS LAS CAPAS
TIENEN 64 UNITS

SGD

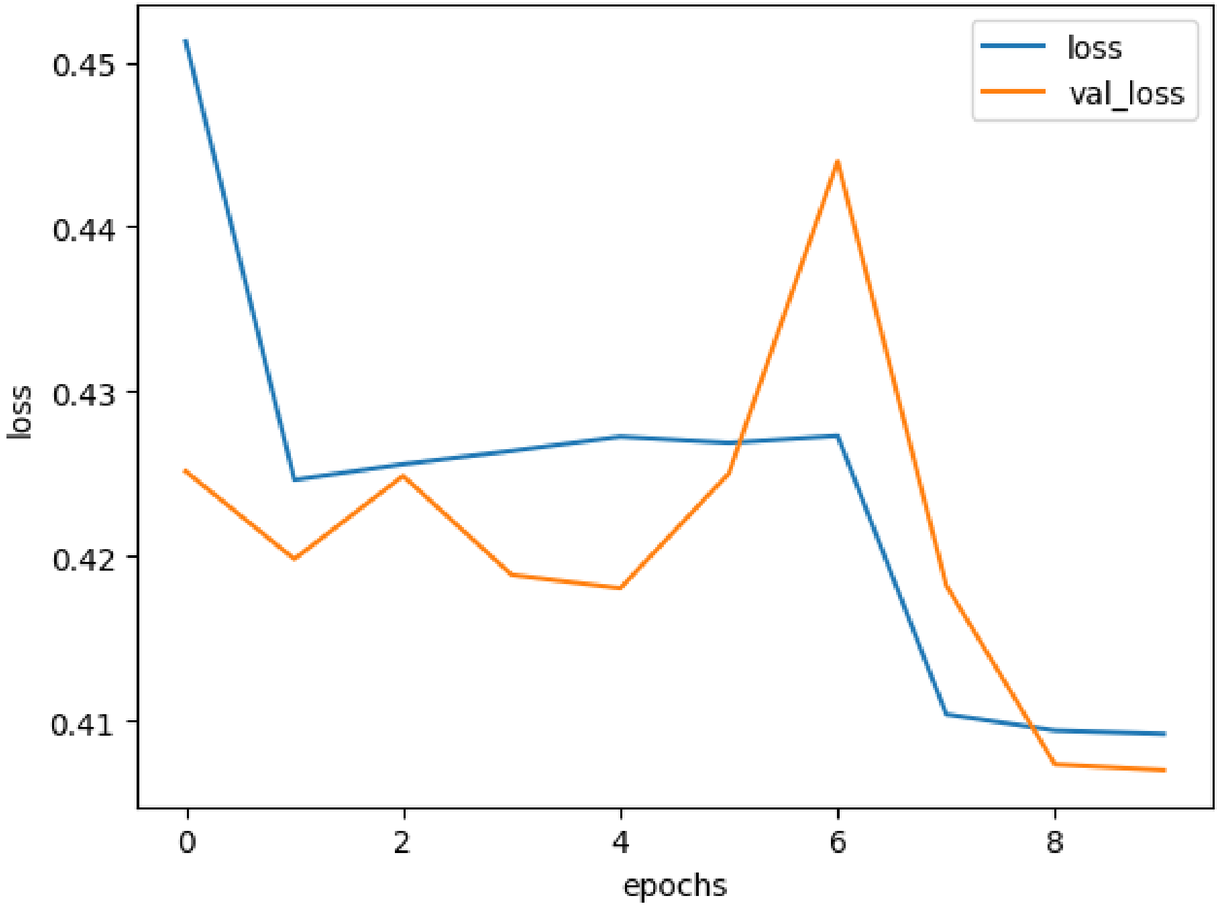
START

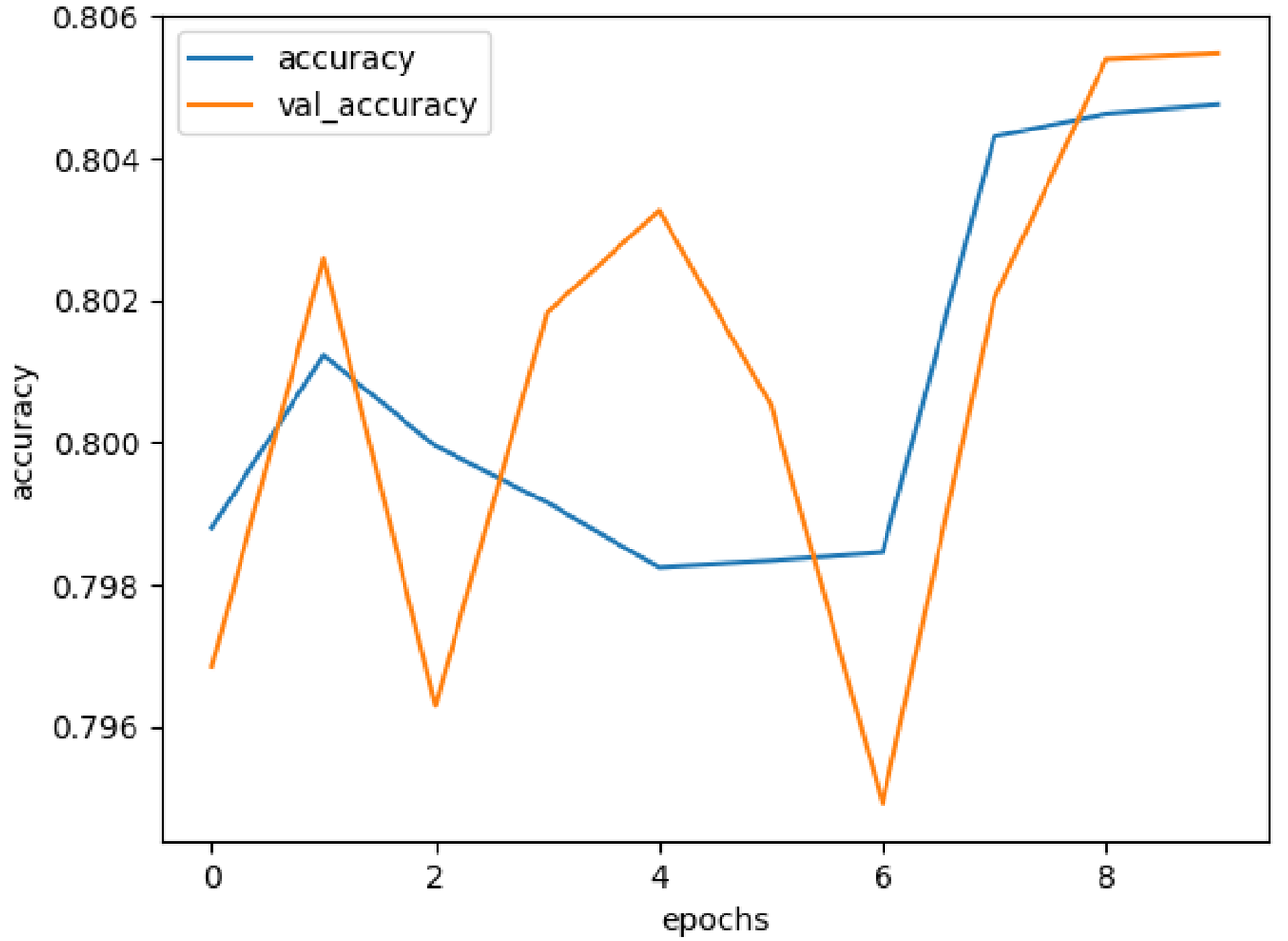
segundo modelo de red neuronal

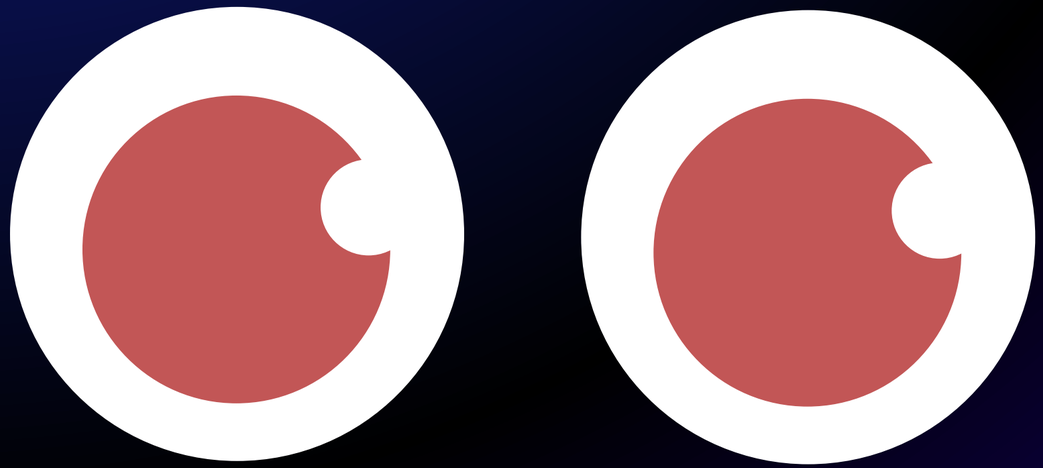
```
modelo = tf.keras.Sequential()
# Definir las 4 entradas

modelo.add(tf.keras.layers.Dense(units=64, activation='relu', input_shape=(115,), kernel_regularizer=l2(0.01)))
modelo.add(tf.keras.layers.BatchNormalization())#mejora la tasa de parendizaje, evitando sobreajuste, ya que tb regulariza la funcion

# Capas ocultas
modelo.add(tf.keras.layers.Dense(units=64, activation="relu", input_dim=115))#relu, la mas comun,rapida y eficiente
modelo.add(tf.keras.layers.BatchNormalization())#mejora la tasa de parendizaje, evitando sobreajuste, ya que tb regulariza la funcion

modelo.add(Dropout(0.2))

modelo.add(tf.keras.layers.Dense(units=64, activation="elu", kernel_regularizer=l2(0.01)))
modelo.add(tf.keras.layers.BatchNormalization())

modelo.add(tf.keras.layers.Dense(units=64, activation='elu'))#eLU mejora la convergencia y la velocidad de entrenamiento
modelo.add(tf.keras.layers.BatchNormalization())

# Capa de salida
modelo.add(tf.keras.layers.Dense(units=1, activation='sigmoid')) # sigmoide para clasificacion binaria
#optimizador
optimizador= tf.keras.optimizers.SGD(learning_rate=0.005,
                                     momentum=0.9, #aporta inercia al proceso de optimizacion, acelerando la convergencia
                                     nesterov=True)# en True actualiza momentum demanera anticipada al gradiente, mejorando la estaboiilidad

# Compilar el modelo
modelo.compile(optimizer='SGD',
               loss='binary_crossentropy',
               metrics=['accuracy', 'Precision', 'Recall'])
tf.keras.metrics.AUC(name='auc')

#detiene el entrenamiento si la metrica no mejora
early_stopping_callbacks = tf.keras.callbacks.EarlyStopping(patience=5,restore_best_weights=True)

#reduce la tasa de aprendizaje cuando el rendimiento del conj. val no mejora
reduce_lr_callbacks = ReduceLROnPlateau(monitor='val_loss', patience=2, verbose=1)

#Guarda el modelo cuando mejora la métrica de validación
filepath = r'D:\Cursos\REPOSITORIOS\DATASET\malware_total\logs\modelo_SGD.h5'
monitor = 'val_loss'
checkpoint_callbacks = ModelCheckpoint(filepath=filepath, monitor=monitor, verbose=1, save_best_only=True)

#visionado_tensorboard
tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir=log_dir)
```

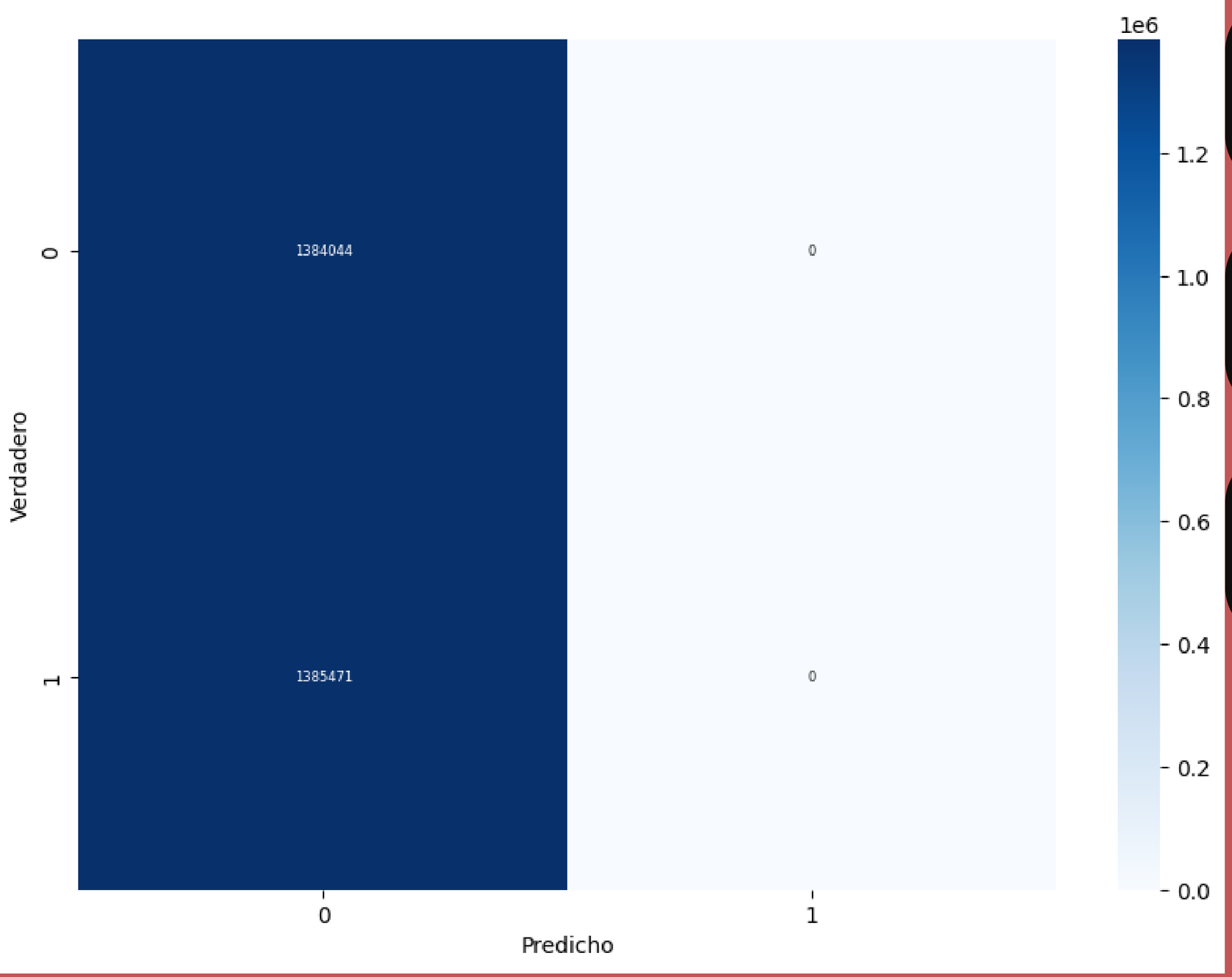
modelo de red neuronal 2
SGD y desbalaceada

```
94886/94886 [==============================] - 95s 997us
Loss: 0.4060222084999084
Accuracy: 0.806392133235931
precision: 0.738439321517944
recall: 0.997440576553344
94886/94886 [==============================] - 74s 777us
c:\Users\victo\anaconda3\envs\tf-gpu\lib\site-packages\s
  _warn_prf(average, modifier, f"{metric.capitalize()} i
c:\Users\victo\anaconda3\envs\tf-gpu\lib\site-packages\s
  _warn_prf(average, modifier, f"{metric.capitalize()} i
              precision    recall  f1-score   support

         0.0       0.46      1.00      0.63   1384396
         1.0       0.00      0.00      0.00   1651932

    accuracy                           0.46   3036328
   macro avg       0.23      0.50      0.31   3036328
weighted avg       0.21      0.46      0.29   3036328
```


loss
val_loss
0.45
0.44
0.43
0.42
0.41
loss
0
2
4
6
8
epochs


accuracy
val_accuracy
0.806
0.804
0.802
0.800
0.798
0.796
accuracy
0
2
4
6
8
epochs

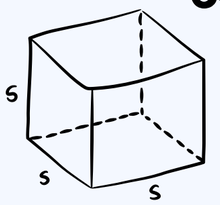


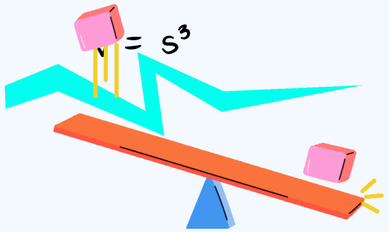

1e6
1.2
1.0
0.8
0.6
0.4
0.2
0.0
1384044
0
1385471
0
0
1
Verdadero
Predicho

MODELO DE RED NEURONAL RESSEMPLED 1

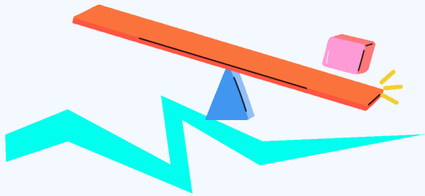
CAPAS OCULTAS
64-32-16-8 Y L2 1 CAPA Y 2 DENSA

CAPA DE ENTRADA
64 UNITS





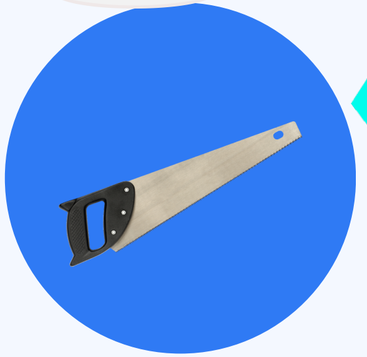
$$\mathbf{r} = \frac{1}{2}\mathbf{a}t^2 + \mathbf{v_0}t + \mathbf{r_0}$$

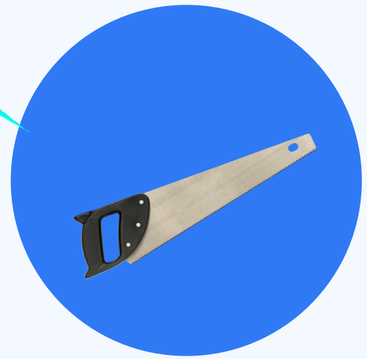


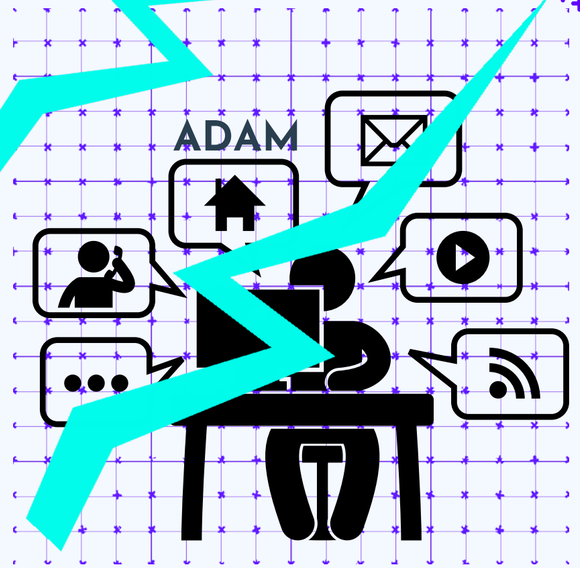
CAPA DE SALIDA

UNIT:1 Y SIGMOID

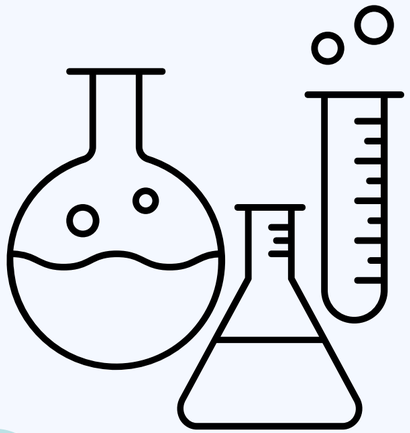
20%







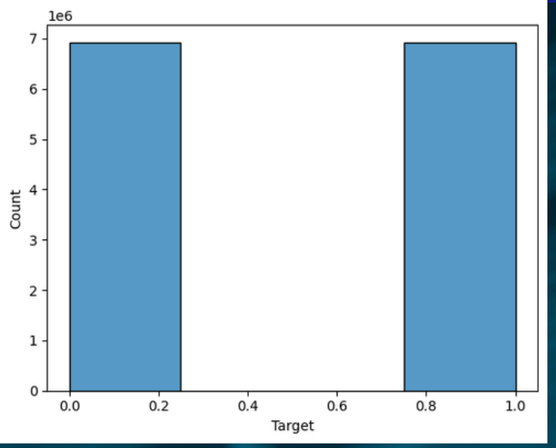
EARLY_STOPPING



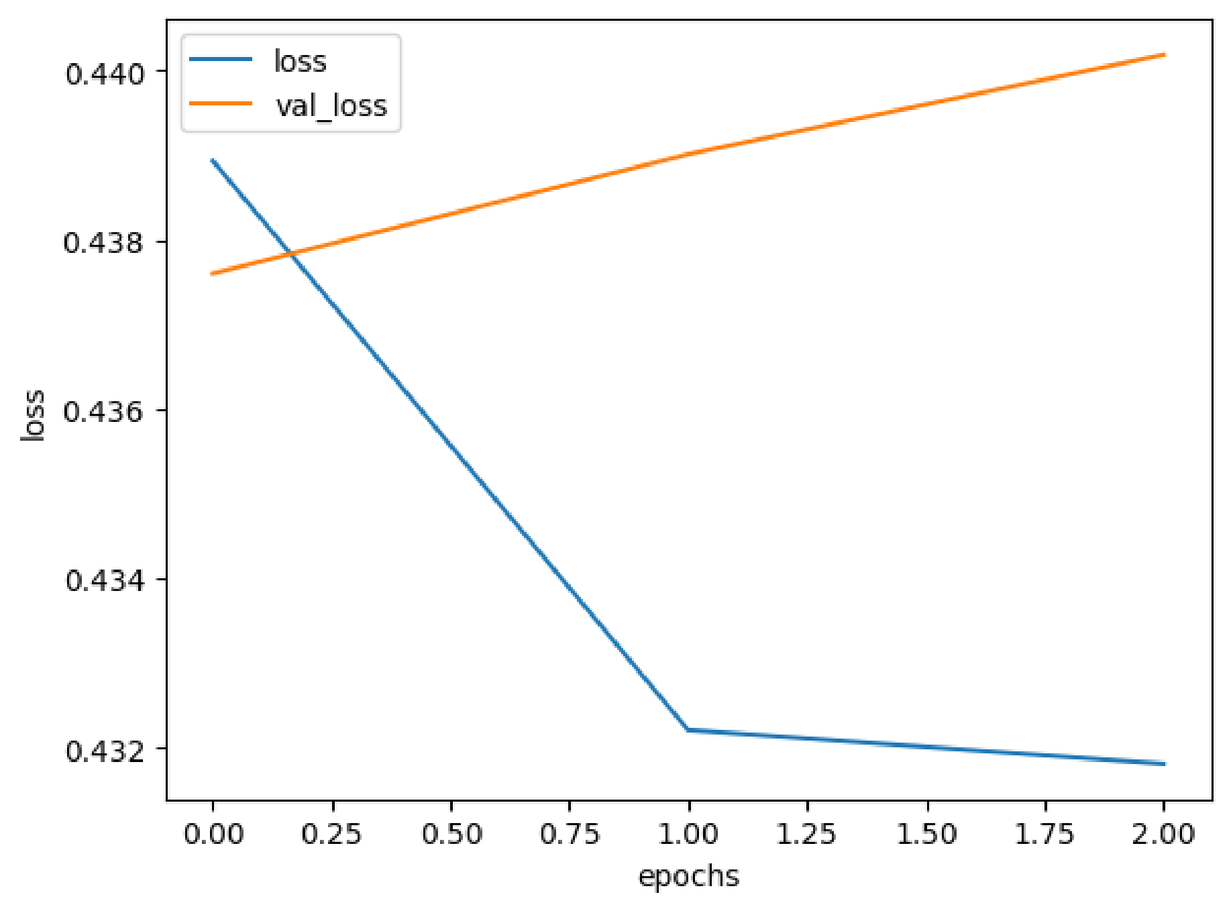
REDUCE_LR

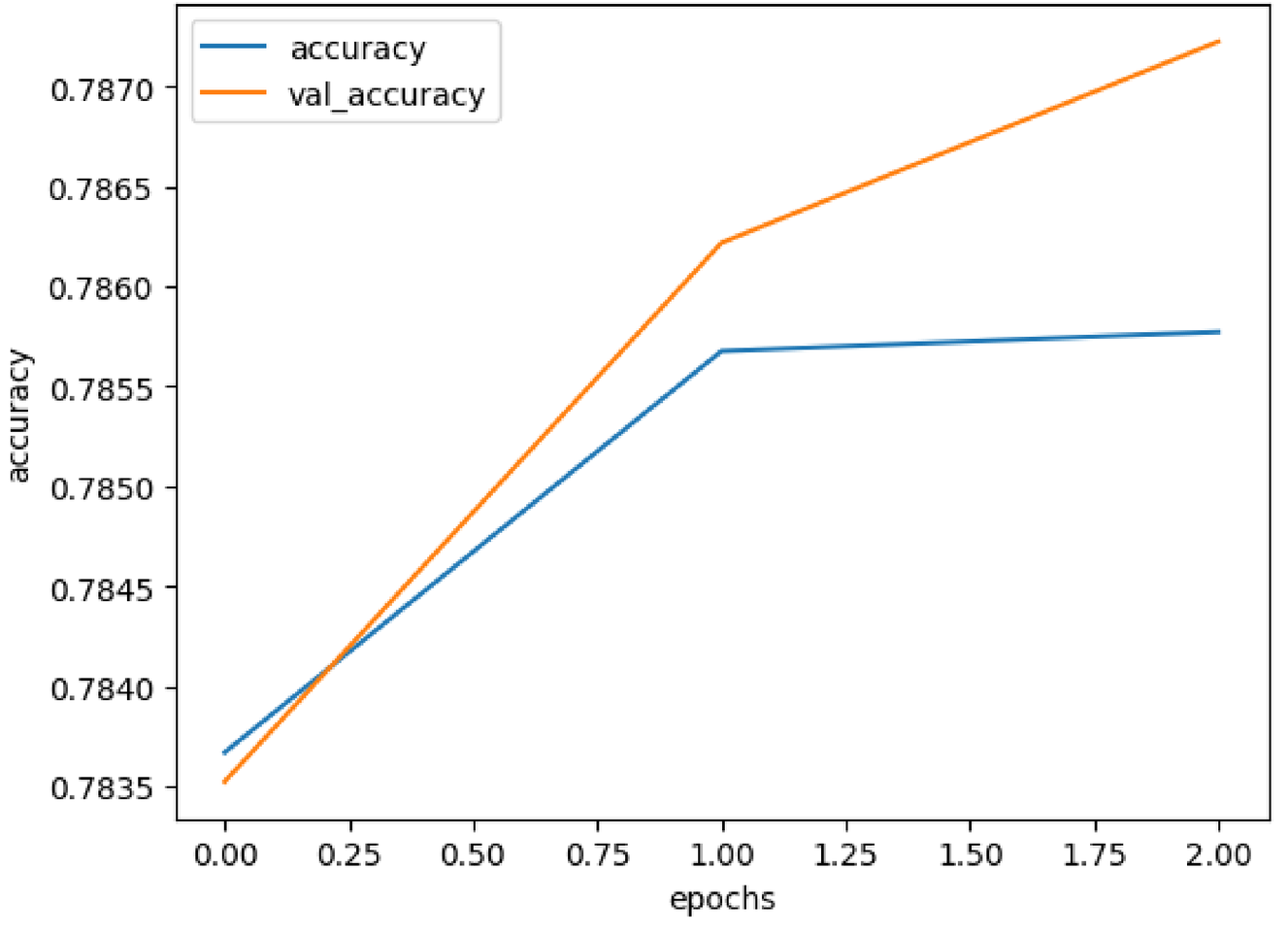
START

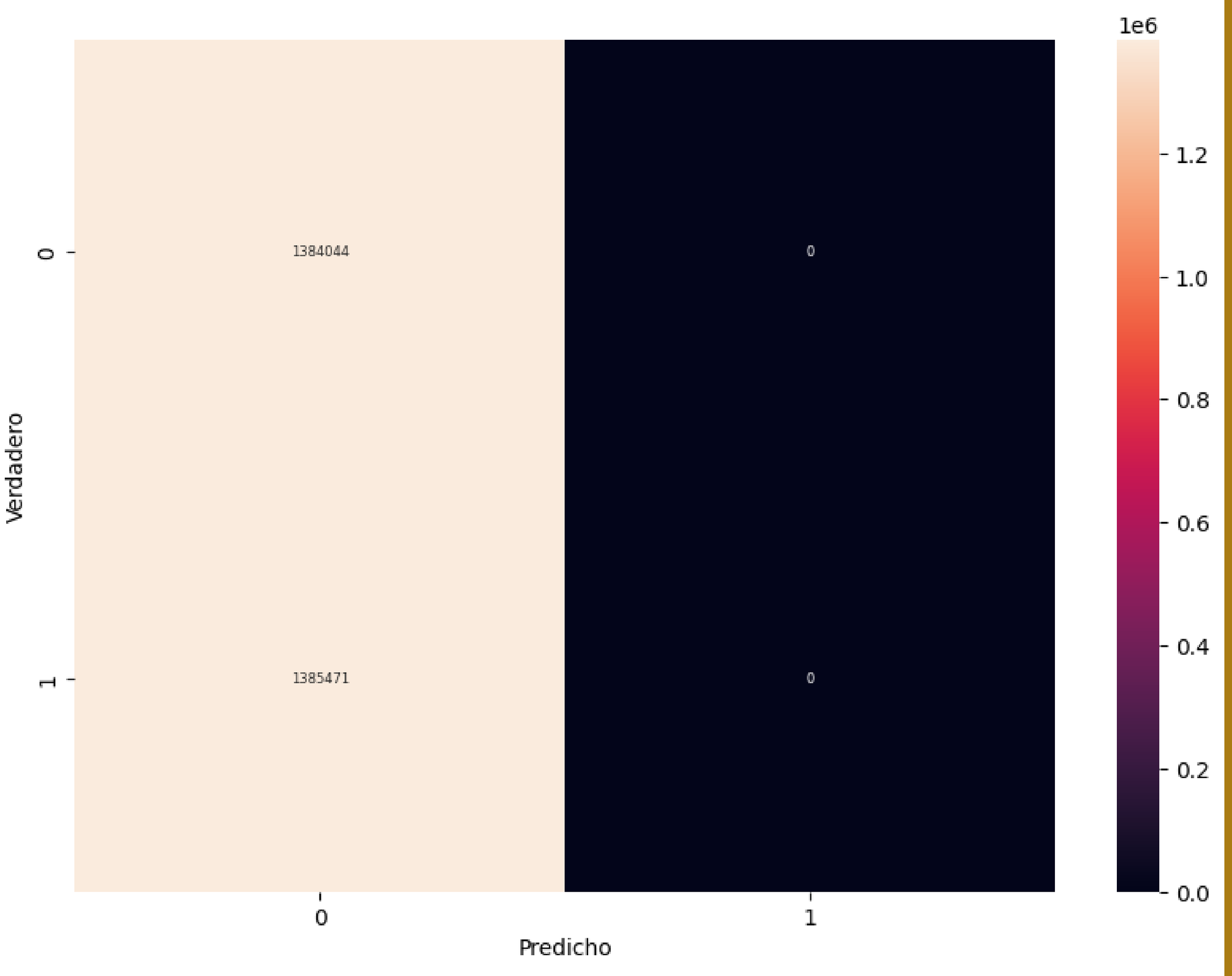
modelo de red neuronal 1




```
Loss: 0.437042325735092l6
Accuracy: 0.7845828533172607
precision: 0.7084441184997559
recall: 0.9675958752632141
86548/86548 [==============================] - 63s 731u
c:\Users\victo\anaconda3\envs\tf-gpu\lib\site-packages\
  _warn_prf(average, modifier, f"{metric.capitalize()}
c:\Users\victo\anaconda3\envs\tf-gpu\lib\site-packages\
  _warn_prf(average, modifier, f"{metric.capitalize()}
              precision    recall  f1-score   support

         0.0       0.50      1.00      0.67   1384044
         1.0       0.00      0.00      0.00   1385471

    accuracy                           0.50   2769515
   macro avg       0.25      0.50      0.33   2769515
weighted avg       0.25      0.50      0.33   2769515
```

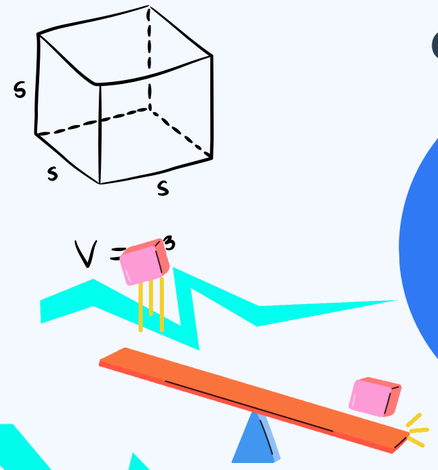

loss
val_loss
0.440
0.438
0.436
0.434
0.432
loss
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
epochs


accuracy
val_accuracy
0.7870
0.7865
0.7860
0.7855
0.7850
0.7845
0.7840
0.7835
accuracy
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
epochs

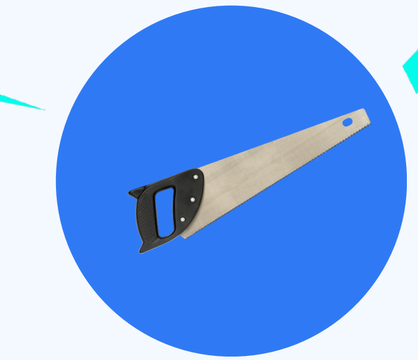

1e6
1.2
1.0
0.8
0.6
0.4
0.2
0.0
1384044
0
1385471
0
0
1
Verdadero
Predicho

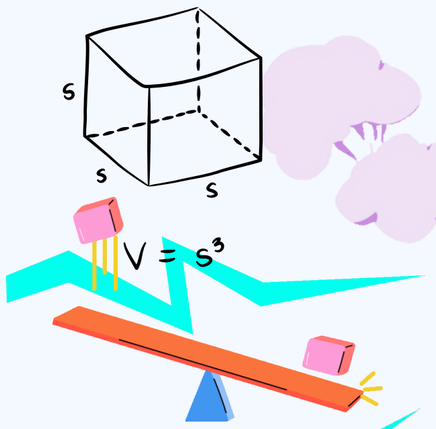
MI MODELO DE RED NEURONAL RESSEMPLED2

CAPA DE ENTRADA



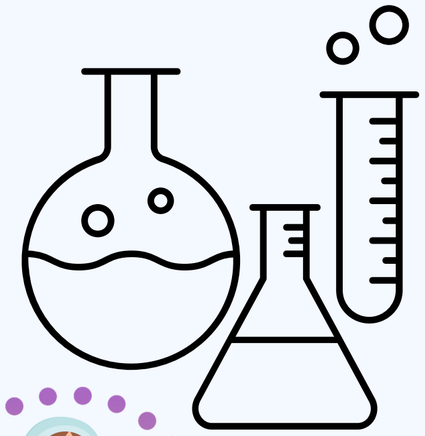
CAPAS OCULTAS





CAPA DE SALIDA

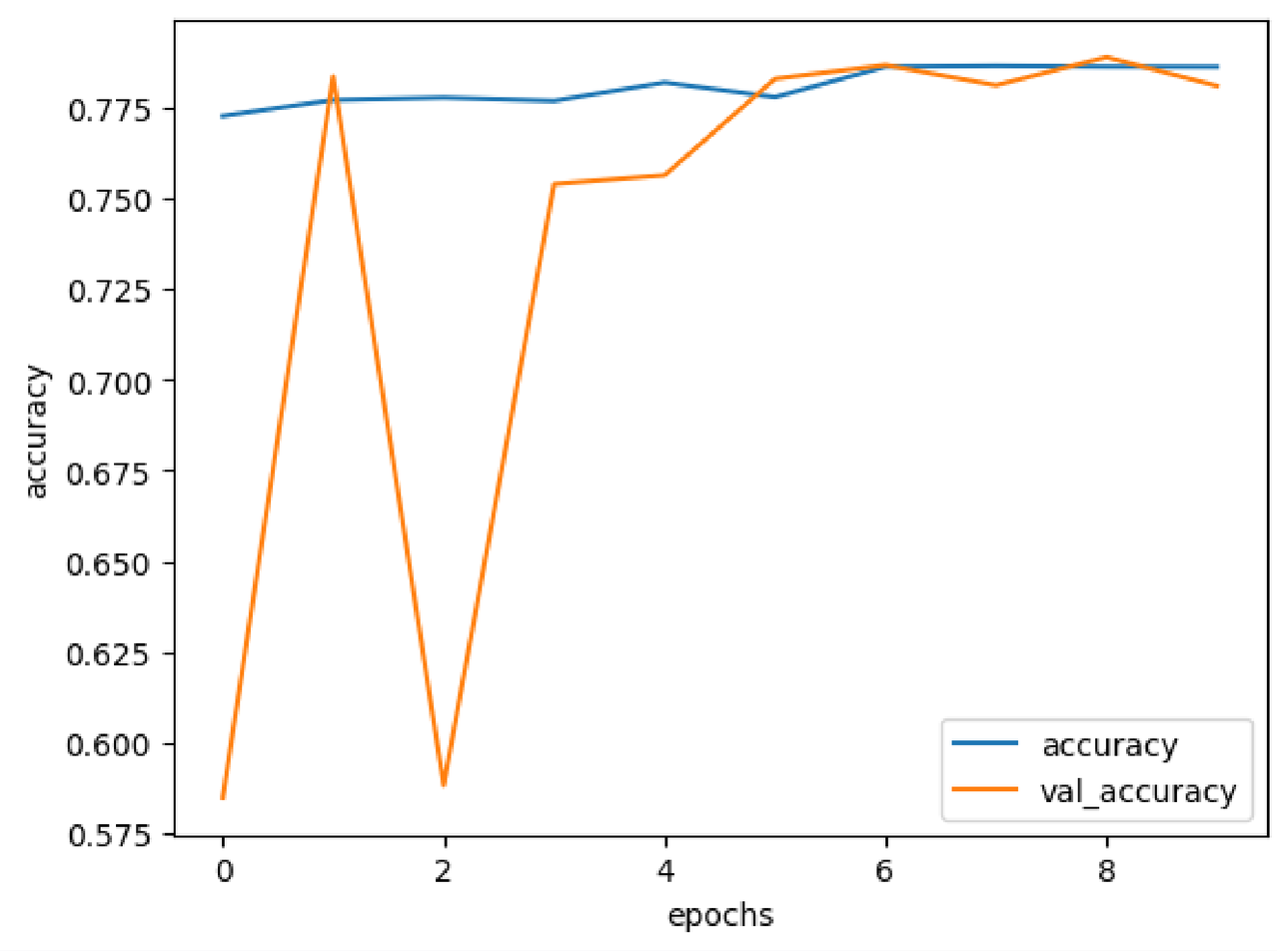
EARLY_STOPPING

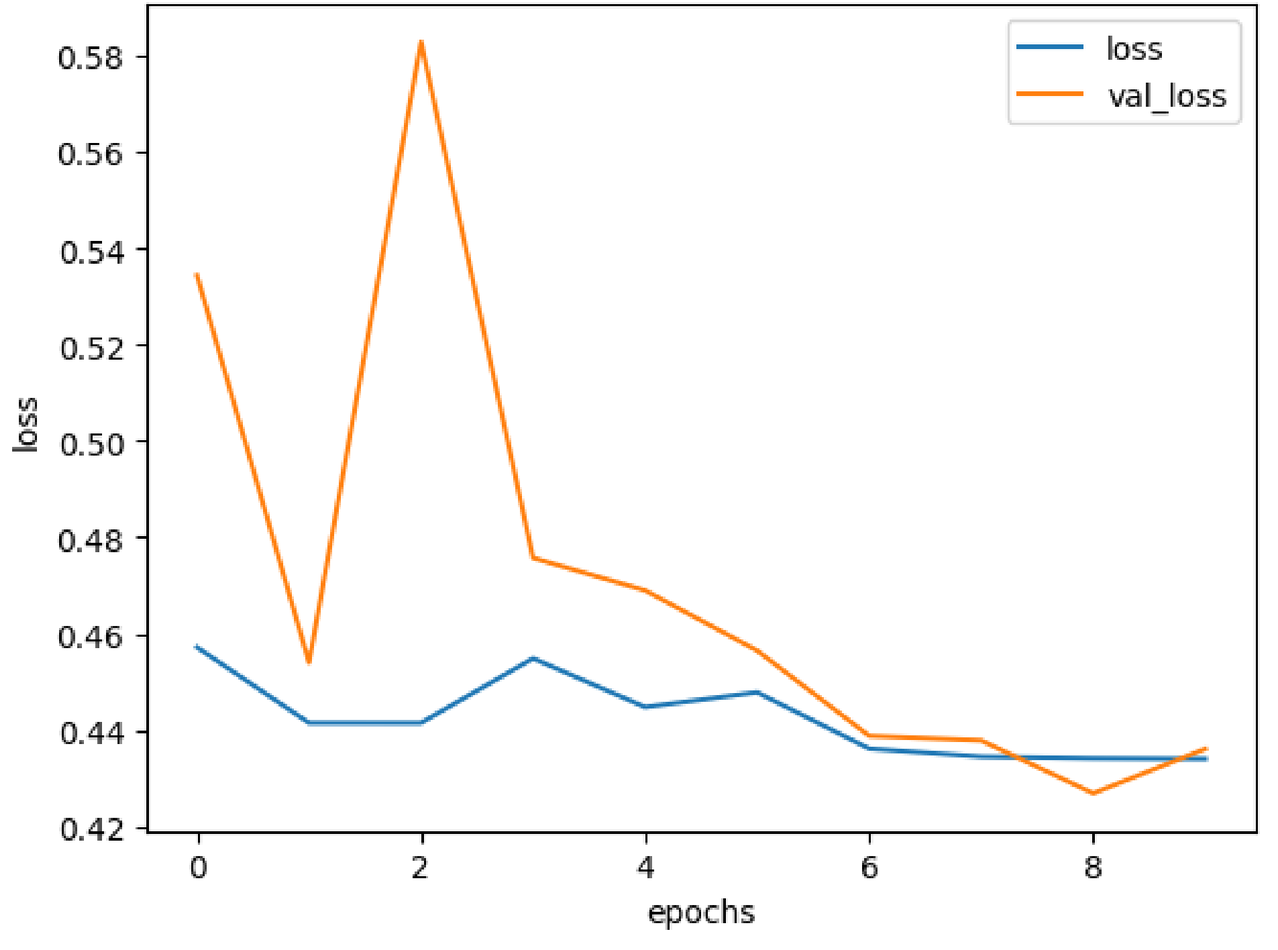


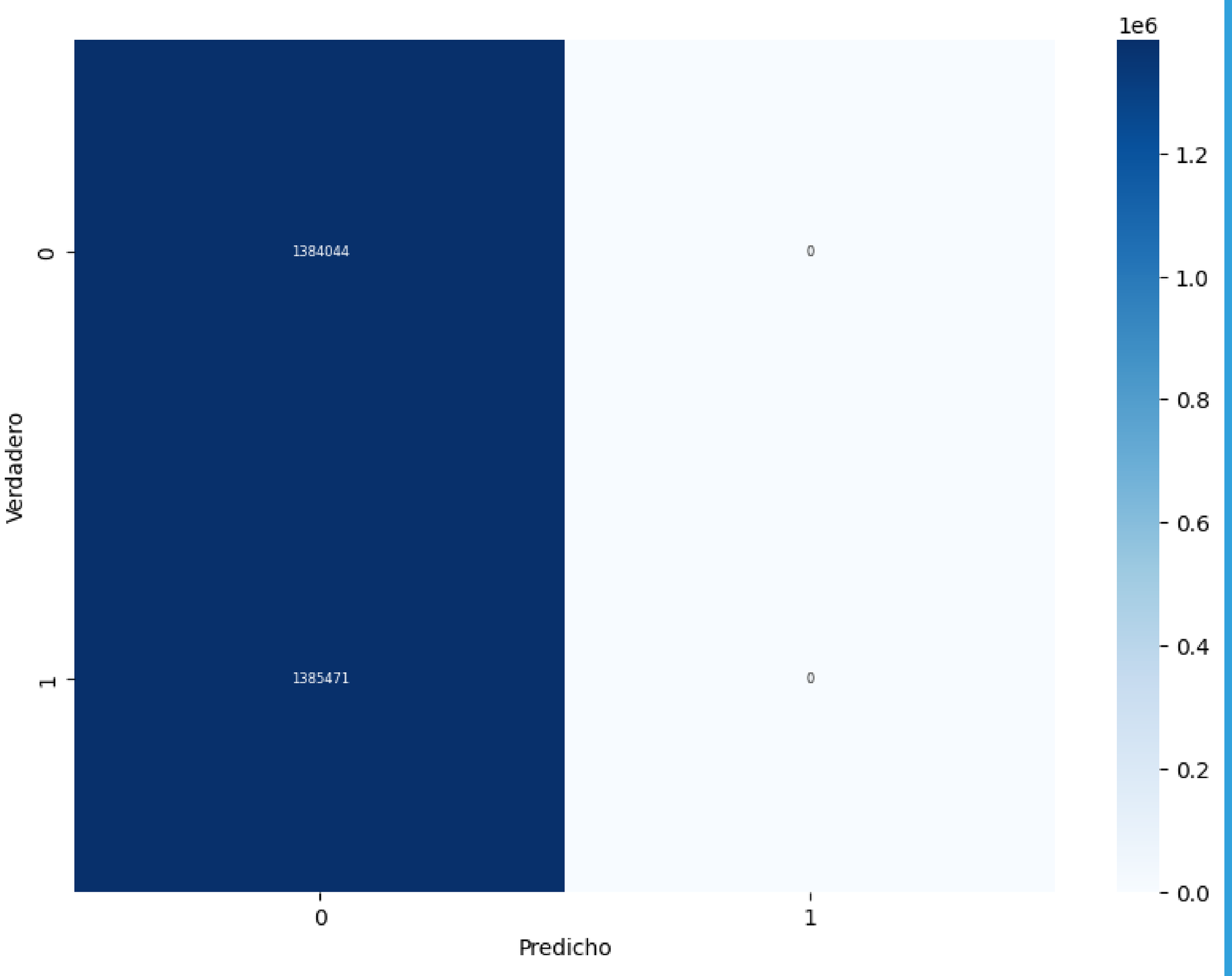
SGD

TODAS LAS CAPAS TIENEN 64 UNITS

```
Loss: 0.43704232573509216
Accuracy: 0.7845828533172607
precision: 0.7084441184997559
recall: 0.9675958752632141
86548/86548 [==============================] - 63s 731us/step
c:\Users\victo\anaconda3\envs\tf-gpu\lib\site-packages\sklearn\
  _warn_prf(average, modifier, f"{metric.capitalize()} is", le
c:\Users\victo\anaconda3\envs\tf-gpu\lib\site-packages\sklearn\
  _warn_prf(average, modifier, f"{metric.capitalize()} is", le
              precision    recall  f1-score   support

         0.0       0.50      1.00      0.67   1384044
         1.0       0.00      0.00      0.00   1385471

    accuracy                           0.50   2769515
   macro avg       0.25      0.50      0.33   2769515
weighted avg       0.25      0.50      0.33   2769515
```


accuracy
0.775
0.750
0.725
0.700
0.675
0.650
0.625
0.600
0.575
0
2
4
6
8
epochs
accuracy
val_accuracy


loss
0.58
0.56
0.54
0.52
0.50
0.48
0.46
0.44
0.42
0
2
4
6
8
epochs
loss
val_loss


1e6
1.2
1.0
0.8
0.6
0.4
0.2
0.0
1384044
0
1385471
0
0
1
Verdadero
Predicho

finally

FINISHED

THANK YOU SO MUCH