## Keras:

model = keras.Sequential([

layers.BatchNormalization(input\_shape = input\_shape),

layers.Dense(256, activation = "relu"),

layers.BatchNormalization(),

layers.Dropout(0.3),

layers.Dense(256, activation = "relu"),

layers.BatchNormalization(),

layers.Dropout(0.3),

layers.Dense(1, activation = "sigmoid")

])

model.compile(

optimizer='adam',

loss='binary\_crossentropy',

metrics=['binary\_accuracy'],

)

history = model.fit(

X\_train, y\_train,

validation\_data=(X\_valid, y\_valid),

batch\_size=512,

epochs=200,

callbacks=[early\_stopping],

verbose = 2

)

## Early stopping!!

early\_stopping = callbacks.EarlyStopping(

min\_delta=0.001, # minimium amount of change to count as an improvement

patience=20, # how many epochs to wait before stopping

restore\_best\_weights=True,

)

## Dropout

keras.Sequential([

# ...

layers.Dropout(rate=0.3), # apply 30% dropout to the next layer

layers.Dense(16),

# ...

])

## Batch Normalization

layers.Dense(16, activation='relu'),

layers.BatchNormalization(),

ej :

from tensorflow import keras

from tensorflow.keras import layers

model = keras.Sequential([

layers.Dense(1024, activation='relu', input\_shape=[11]),

layers.Dropout(0.3),

layers.BatchNormalization(),

layers.Dense(1024, activation='relu'),

layers.Dropout(0.3),

layers.BatchNormalization(),

layers.Dense(1024, activation='relu'),

layers.Dropout(0.3),

layers.BatchNormalization(),

layers.Dense(1),

])

model.compile(

optimizer='adam',

loss='mae',

)

history = model.fit(

X\_train, y\_train,

validation\_data=(X\_valid, y\_valid),

batch\_size=256,

epochs=100,

verbose=0,

)