Lab 4 - BCC406/PCC177

REDES NEURAIS E APRENDIZAGEM EM PROFUNDIDADE

Uso de Framework (TensorFlow) e K-Fold

Prof. Eduardo e Prof. Pedro

Objetivos:

- Classificação utilizando TensorFlow.
- Utilização do *Stratified K-fold*.
- Cálculos de métircas

Data da entrega : 07/11

- Complete o código (marcado com 'ToDo') e quando requisitado, escreva textos diretamente nos notebooks. Onde tiver *None*, substitua pelo seu código.
- Execute todo notebook e salve tudo em um PDF **nomeado** como "NomeSobrenome-LabX.pdf"
- Envie o PDF via google FORM
- Envie o *.ipynb* também.

Preparação do ambiente e Tratamento dos dados

Preparação do ambiente

Importação das bibliotecas

Primeiro precisamos importar os pacotes. Vamos executar a célula abaixo para importar todos os pacotes que precisaremos.

- *numpy* é o pacote fundamental para a computação científica com Python.
- *h5py* é um pacote comum para interagir com um conjunto de dados armazenado em um arquivo H5.
- *matplotlib* é uma biblioteca famosa para plotar gráficos em Python.
- *PIL* e *scipy* são usados aqui para carregar as imagens e testar seu modelo final.
- *Scikit Learn* é um pacote muito utilizado para treinamento de modelos e outros algoritmos de *machine learning*.

```
import numpy as np
import matplotlib.pyplot as plt
import h5py
import scipy
from keras.models import Sequential
from keras.layers import Dense
from sklearn.metrics import accuracy_score
from keras.optimizers import SGD, Adam
from keras.initializers import RandomNormal, RandomUniform
from keras.layers import LeakyReLU
from tensorflow import keras
from fpdf import FPDF

!pip install fpdf
```

```
Collecting fpdf
  Downloading fpdf-1.7.2.tar.gz (39 kB)
  Preparing metadata (setup.py) ... e=fpdf-1.7.2-py2.py3-none-any.whl
size=40704
sha256=409cdc9c93630592c062fe50d6a67e8d19d42fa2188bc1b0cce084d42661595
4
  Stored in directory:
/root/.cache/pip/wheels/f9/95/ba/f418094659025eb9611f17cbcaf2334236bf3
9a0c3453ea455
Successfully built fpdf
Installing collected packages: fpdf
Successfully installed fpdf-1.7.2
```

Configurando os *plots* de gráficos

O próximo passo é configurar o *matplotlib* e a geração de valores aleatórios.

```
%matplotlib inline
plt.rcParams['figure.figsize'] = (5.0, 4.0) # set default size of
plots
plt.rcParams['image.interpolation'] = 'nearest'
plt.rcParams['image.cmap'] = 'gray'

%load_ext autoreload
%autoreload 2

np.random.seed(1)
```

```
The autoreload extension is already loaded. To reload it, use:
  %reload_ext autoreload
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

Configurando o Google Colab.

Configurando o Google Colab para acessar os nossos dados.

```
# Você vai precisar fazer o upload dos arquivos no seu drive (faer na
pasta raiz) e montá-lo
# não se esqueça de ajustar o path para o seu drive
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

Carregando e préprocessamento dos dados

```
# Função para ler os dados (gato/não-gato)
def load_dataset():
    def _load_data():
      train_dataset =
h5py.File('/content/drive/MyDrive/DEEP_LEARNING/SPACTSHIP/train_catvno
ncat.h5', "r")
      train_set_x_orig = np.array(train_dataset["train_set_x"][:]) #
your train set features
      train_set_y_orig = np.array(train_dataset["train_set_y"][:]) #
your train set labels

      test_dataset =
h5py.File('/content/drive/MyDrive/DEEP_LEARNING/SPACTSHIP/test_catvnon
cat.h5', "r")
      test_set_x_orig = np.array(test_dataset["test_set_x"][:]) # your
test set features
      test_set_y_orig = np.array(test_dataset["test_set_y"][:]) # your
test set labels

      classes = np.array(test_dataset["list_classes"][:]) # the list
of classes
      train_set_y_orig = train_set_y_orig.reshape((1,
train_set_y_orig.shape[0]))
      test_set_y_orig = test_set_y_orig.reshape((1,
test_set_y_orig.shape[0]))

      return train_set_x_orig, train_set_y_orig, test_set_x_orig,
test_set_y_orig, classes

    def _preprocess_dataset(_treino_x_orig, _teste_x_orig):
      # Formate o conjunto de treinamento e teste dados de treinamento
e teste para que as imagens
      # de tamanho (num_px, num_px, 3) sejam vetores de forma (num_px
* num_px * 3, 1)
      _treino_x_vet = _treino_x_orig.reshape(_treino_x_orig.shape[0],
-1) # ToDo: vetorizar os dados de treinamento aqui
```

```
      _teste_x_vet = _teste_x_orig.reshape(_teste_x_orig.shape[0], -1)
# ToDo: vetorizar os dados de teste aqui

      # Normalize os dados (colocar no intervalo [0.0, 1.0])
      _treino_x = _treino_x_vet/255. # ToDo: normalize os dados de
treinamento aqui
      _teste_x = _teste_x_vet/255. # ToDo: normalize os dados de teste
aqui
      return _treino_x, _teste_x

    treino_x_orig, treino_y, teste_x_orig, teste_y, classes =
_load_data()
    treino_x, teste_x = _preprocess_dataset(treino_x_orig,
teste_x_orig)
    return treino_x, treino_y, teste_x, teste_y, classes
```

Carregando os dados

```
# Lendo os dados (gato/não-gato)
treino_x, treino_y, teste_x, teste_y, classes = load_dataset()
```

Treinamento do modelo (85pt)

Há diversos frameworks para criação de modelos de *deep learning*, como TensorFlow e PyTorch. Nesta prática, usaremos o TensorFlow.

Modelo 1: Testando um modelo com uma camada oculta com 8 neurônios (10pt)

Definição de um modelo com uma camada oculta (8 neurônios) e uma camada de saída com um neurônio (gato e não gato). Usaremos a ativação ReLU (*Retified Linear Unity*) na camada oculta e a *sigmoid* na camada de saída. Para classificação de classes 0 ou 1, pode-se ter um único neurônio de saída e deve-se usar a operação sigmoid antes de se calcular o custo (mean-squared error ou binary cross entropy).

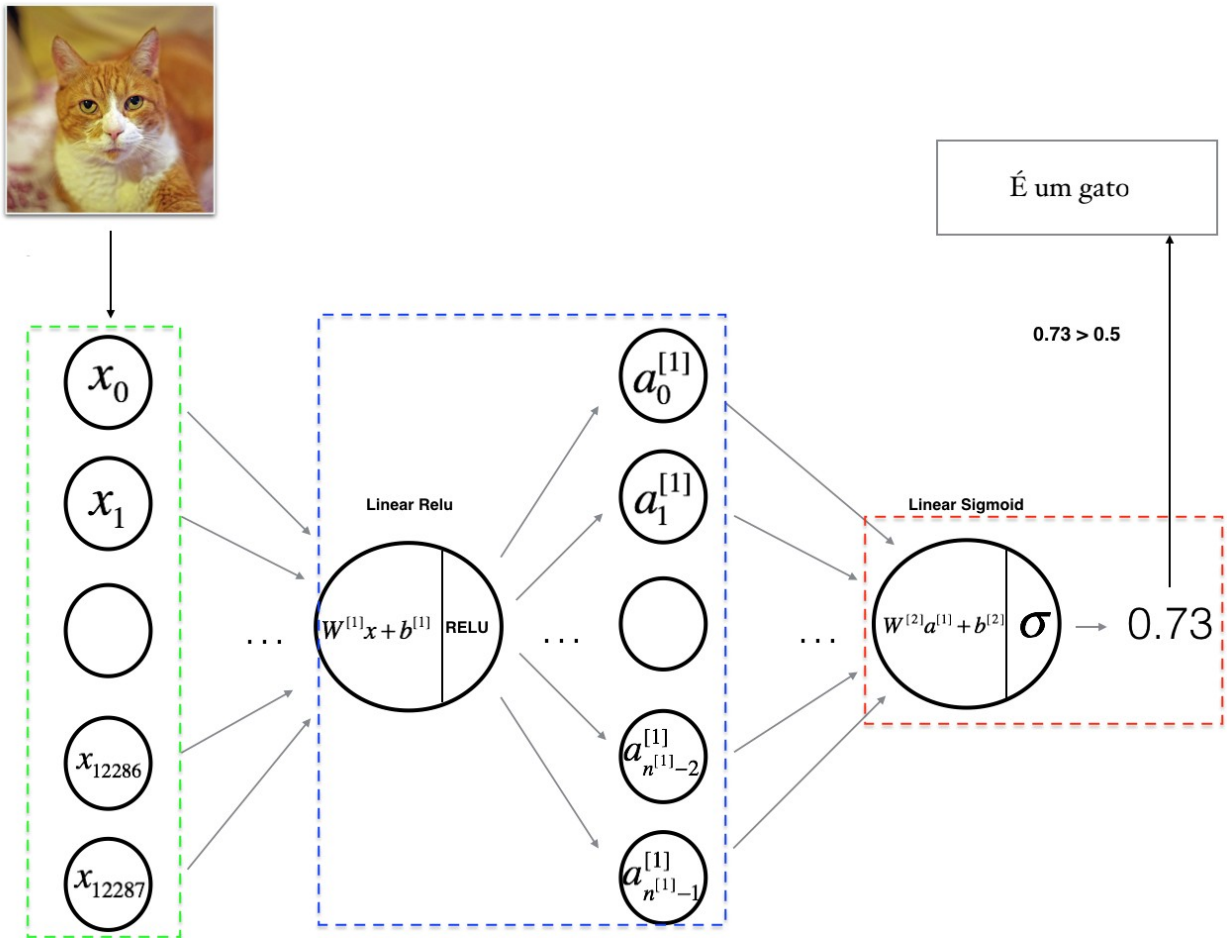


Figura 7: Rede neural com 2 camadas. Resumo do modelo: ***ENTRADA -> LINEAR -> RELU -> LINEAR -> SIGMOID -> SAIDA***.

```
def treinar_modelo(modelo, x, y, epochs=100):
    # Setando a seed
    np.random.seed(10)

    # Compilando o modelo
    modelo.compile(optimizer='adam',
                   loss='binary_crossentropy',
                   metrics='accuracy')

    # Imprimindo a arquitetura da rede proposta
    modelo.summary()

    # Treinando o modelo
    modelo.fit(treino_x, treino_y.reshape(-1), epochs=epochs)
    return modelo
```

ToDo: Definindo o modelo (5pt)

```
# Definição do modelo
def modelo_1():
  _model = Sequential() # Crie um modelo sequencial com
keras.Sequential
  _model.add(Dense(8, input_shape=(12288,), activation='relu')) #
ToDo: Adicione uma camada densa com 8 neurônios e ativação relu
  _model.add(Dense(1, activation='sigmoid')) # ToDo: Adicione uma
camada densa com 1 neurônio e ativação sigmoid **dica** use a classe
keras.layers.Dense
  _model.build(input_shape=(None, 12288))
  return _model

treino_x.shape[1]
```

```
12288
```

Treine o modelo e depois **use os parâmetros treinados** para classificar as imagens de treinamento e teste e verificar a acurácia.

ToDo: Instanciando o modelo e testando (5pt)

```
np.random.seed(1)

# Criando o modelo
m1 = modelo_1() # ToDo: chame a função que define o modelo

# Treinando o modelo

m1 = treinar_modelo(m1, treino_x, treino_y.reshape(-1)) # ToDo: Chame
a função para treinar o modelo


## Predição da rede
print(f'\n\nAcurácia no treino: {accuracy_score(treino_y.ravel(),
(m1.predict(treino_x)>0.5).astype(int)):.2f}') # ToDo: Utilize a
função accuracy_score do sklearn para calcular a acurácia nos dados de
treino
                                          # **dica** use o
model.predict para predizer os dados e use o reshape com -1 nos labels
print(f'Acurácia no teste: {accuracy_score(teste_y.ravel(),
(m1.predict(teste_x)>0.5).astype(int)):.2f}') # ToDo: Utilize a função
accuracy_score do sklearn para calcular a acurácia nos dados de teste
                                       # **dica** use o model.predict
para predizer os dados e use o reshape com -1 nos labels
```

```
Model: "sequential_22"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
```

```
 dense_56 (Dense)            (None, 8)                 98312

 dense_57 (Dense)            (None, 1)                 9

=================================================================
Total params: 98321 (384.07 KB)
Trainable params: 98321 (384.07 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
Epoch 1/100
7/7 [==============================] - 1s 6ms/step - loss: 1.0730 -
accuracy: 0.5502
Epoch 2/100
7/7 [==============================] - 0s 6ms/step - loss: 0.6782 -
accuracy: 0.6268
Epoch 3/100
7/7 [==============================] - 0s 6ms/step - loss: 0.6414 -
accuracy: 0.6651
Epoch 4/100
7/7 [==============================] - 0s 7ms/step - loss: 0.6139 -
accuracy: 0.6794
Epoch 5/100
7/7 [==============================] - 0s 7ms/step - loss: 0.5452 -
accuracy: 0.7081
Epoch 6/100
7/7 [==============================] - 0s 7ms/step - loss: 0.5330 -
accuracy: 0.7321
Epoch 7/100
7/7 [==============================] - 0s 9ms/step - loss: 0.5164 -
accuracy: 0.7656
Epoch 8/100
7/7 [==============================] - 0s 7ms/step - loss: 0.5317 -
accuracy: 0.7081
Epoch 9/100
7/7 [==============================] - 0s 6ms/step - loss: 0.5928 -
accuracy: 0.6794
Epoch 10/100
7/7 [==============================] - 0s 7ms/step - loss: 0.5313 -
accuracy: 0.7512
Epoch 11/100
7/7 [==============================] - 0s 7ms/step - loss: 0.5093 -
accuracy: 0.7464
Epoch 12/100
7/7 [==============================] - 0s 6ms/step - loss: 0.5605 -
accuracy: 0.7081
Epoch 13/100
7/7 [==============================] - 0s 9ms/step - loss: 0.4621 -
accuracy: 0.7799
Epoch 14/100
```

```
7/7 [==============================] - 0s 7ms/step - loss: 0.4372 -
accuracy: 0.7751
Epoch 15/100
7/7 [==============================] - 0s 7ms/step - loss: 0.4234 -
accuracy: 0.8038
Epoch 16/100
7/7 [==============================] - 0s 6ms/step - loss: 0.4027 -
accuracy: 0.8708
Epoch 17/100
7/7 [==============================] - 0s 7ms/step - loss: 0.4014 -
accuracy: 0.8660
Epoch 18/100
7/7 [==============================] - 0s 7ms/step - loss: 0.4835 -
accuracy: 0.7703
Epoch 19/100
7/7 [==============================] - 0s 8ms/step - loss: 0.5069 -
accuracy: 0.7273
Epoch 20/100
7/7 [==============================] - 0s 7ms/step - loss: 0.4384 -
accuracy: 0.8278
Epoch 21/100
7/7 [==============================] - 0s 7ms/step - loss: 0.4066 -
accuracy: 0.8038
Epoch 22/100
7/7 [==============================] - 0s 7ms/step - loss: 0.3797 -
accuracy: 0.8565
Epoch 23/100
7/7 [==============================] - 0s 8ms/step - loss: 0.3570 -
accuracy: 0.8660
Epoch 24/100
7/7 [==============================] - 0s 10ms/step - loss: 0.3609 -
accuracy: 0.8804
Epoch 25/100
7/7 [==============================] - 0s 7ms/step - loss: 0.3462 -
accuracy: 0.8804
Epoch 26/100
7/7 [==============================] - 0s 6ms/step - loss: 0.3359 -
accuracy: 0.8852
Epoch 27/100
7/7 [==============================] - 0s 6ms/step - loss: 0.3170 -
accuracy: 0.9091
Epoch 28/100
7/7 [==============================] - 0s 7ms/step - loss: 0.3232 -
accuracy: 0.9187
Epoch 29/100
7/7 [==============================] - 0s 6ms/step - loss: 0.3581 -
accuracy: 0.8325
Epoch 30/100
7/7 [==============================] - 0s 6ms/step - loss: 0.3754 -
```

```
accuracy: 0.8230
Epoch 31/100
7/7 [==============================] - 0s 7ms/step - loss: 0.3031 -
accuracy: 0.9091
Epoch 32/100
7/7 [==============================] - 0s 6ms/step - loss: 0.3140 -
accuracy: 0.8900
Epoch 33/100
7/7 [==============================] - 0s 6ms/step - loss: 0.3003 -
accuracy: 0.9091
Epoch 34/100
7/7 [==============================] - 0s 8ms/step - loss: 0.2835 -
accuracy: 0.9234
Epoch 35/100
7/7 [==============================] - 0s 7ms/step - loss: 0.2721 -
accuracy: 0.9330
Epoch 36/100
7/7 [==============================] - 0s 7ms/step - loss: 0.2776 -
accuracy: 0.9378
Epoch 37/100
7/7 [==============================] - 0s 7ms/step - loss: 0.2678 -
accuracy: 0.9234
Epoch 38/100
7/7 [==============================] - 0s 7ms/step - loss: 0.2837 -
accuracy: 0.9234
Epoch 39/100
7/7 [==============================] - 0s 7ms/step - loss: 0.2782 -
accuracy: 0.9187
Epoch 40/100
7/7 [==============================] - 0s 6ms/step - loss: 0.2591 -
accuracy: 0.9426
Epoch 41/100
7/7 [==============================] - 0s 6ms/step - loss: 0.2744 -
accuracy: 0.8900
Epoch 42/100
7/7 [==============================] - 0s 9ms/step - loss: 0.2705 -
accuracy: 0.8947
Epoch 43/100
7/7 [==============================] - 0s 7ms/step - loss: 0.2596 -
accuracy: 0.9139
Epoch 44/100
7/7 [==============================] - 0s 8ms/step - loss: 0.2417 -
accuracy: 0.9187
Epoch 45/100
7/7 [==============================] - 0s 7ms/step - loss: 0.2788 -
accuracy: 0.8995
Epoch 46/100
7/7 [==============================] - 0s 8ms/step - loss: 0.2616 -
accuracy: 0.9187
```

```
Epoch 47/100
7/7 [==============================] - 0s 7ms/step - loss: 0.2704 -
accuracy: 0.9187
Epoch 48/100
7/7 [==============================] - 0s 7ms/step - loss: 0.3254 -
accuracy: 0.8373
Epoch 49/100
7/7 [==============================] - 0s 7ms/step - loss: 0.2608 -
accuracy: 0.8947
Epoch 50/100
7/7 [==============================] - 0s 7ms/step - loss: 0.3717 -
accuracy: 0.8038
Epoch 51/100
7/7 [==============================] - 0s 6ms/step - loss: 0.3169 -
accuracy: 0.8469
Epoch 52/100
7/7 [==============================] - 0s 7ms/step - loss: 0.2816 -
accuracy: 0.8995
Epoch 53/100
7/7 [==============================] - 0s 7ms/step - loss: 0.2956 -
accuracy: 0.8612
Epoch 54/100
7/7 [==============================] - 0s 7ms/step - loss: 0.2122 -
accuracy: 0.9474
Epoch 55/100
7/7 [==============================] - 0s 7ms/step - loss: 0.2014 -
accuracy: 0.9426
Epoch 56/100
7/7 [==============================] - 0s 6ms/step - loss: 0.1937 -
accuracy: 0.9713
Epoch 57/100
7/7 [==============================] - 0s 6ms/step - loss: 0.1832 -
accuracy: 0.9713
Epoch 58/100
7/7 [==============================] - 0s 6ms/step - loss: 0.1831 -
accuracy: 0.9665
Epoch 59/100
7/7 [==============================] - 0s 8ms/step - loss: 0.1818 -
accuracy: 0.9713
Epoch 60/100
7/7 [==============================] - 0s 8ms/step - loss: 0.1772 -
accuracy: 0.9665
Epoch 61/100
7/7 [==============================] - 0s 7ms/step - loss: 0.1733 -
accuracy: 0.9713
Epoch 62/100
7/7 [==============================] - 0s 7ms/step - loss: 0.1722 -
accuracy: 0.9761
Epoch 63/100
```

```
7/7 [==============================] - 0s 7ms/step - loss: 0.1896 -
accuracy: 0.9569
Epoch 64/100
7/7 [==============================] - 0s 7ms/step - loss: 0.2201 -
accuracy: 0.9282
Epoch 65/100
7/7 [==============================] - 0s 8ms/step - loss: 0.1855 -
accuracy: 0.9713
Epoch 66/100
7/7 [==============================] - 0s 8ms/step - loss: 0.2110 -
accuracy: 0.9474
Epoch 67/100
7/7 [==============================] - 0s 8ms/step - loss: 0.1805 -
accuracy: 0.9474
Epoch 68/100
7/7 [==============================] - 0s 7ms/step - loss: 0.1600 -
accuracy: 0.9761
Epoch 69/100
7/7 [==============================] - 0s 7ms/step - loss: 0.1856 -
accuracy: 0.9426
Epoch 70/100
7/7 [==============================] - 0s 7ms/step - loss: 0.1622 -
accuracy: 0.9761
Epoch 71/100
7/7 [==============================] - 0s 7ms/step - loss: 0.1611 -
accuracy: 0.9761
Epoch 72/100
7/7 [==============================] - 0s 6ms/step - loss: 0.1508 -
accuracy: 0.9761
Epoch 73/100
7/7 [==============================] - 0s 7ms/step - loss: 0.1643 -
accuracy: 0.9761
Epoch 74/100
7/7 [==============================] - 0s 7ms/step - loss: 0.1444 -
accuracy: 0.9904
Epoch 75/100
7/7 [==============================] - 0s 7ms/step - loss: 0.1547 -
accuracy: 0.9713
Epoch 76/100
7/7 [==============================] - 0s 7ms/step - loss: 0.1684 -
accuracy: 0.9522
Epoch 77/100
7/7 [==============================] - 0s 10ms/step - loss: 0.2076 -
accuracy: 0.9426
Epoch 78/100
7/7 [==============================] - 0s 8ms/step - loss: 0.2051 -
accuracy: 0.9187
Epoch 79/100
7/7 [==============================] - 0s 7ms/step - loss: 0.1558 -
```

```
accuracy: 0.9617
Epoch 80/100
7/7 [==============================] - 0s 8ms/step - loss: 0.1311 -
accuracy: 0.9809
Epoch 81/100
7/7 [==============================] - 0s 7ms/step - loss: 0.1394 -
accuracy: 0.9713
Epoch 82/100
7/7 [==============================] - 0s 7ms/step - loss: 0.1355 -
accuracy: 0.9809
Epoch 83/100
7/7 [==============================] - 0s 7ms/step - loss: 0.1334 -
accuracy: 0.9856
Epoch 84/100
7/7 [==============================] - 0s 7ms/step - loss: 0.1507 -
accuracy: 0.9569
Epoch 85/100
7/7 [==============================] - 0s 8ms/step - loss: 0.1659 -
accuracy: 0.9522
Epoch 86/100
7/7 [==============================] - 0s 7ms/step - loss: 0.1452 -
accuracy: 0.9665
Epoch 87/100
7/7 [==============================] - 0s 7ms/step - loss: 0.1307 -
accuracy: 0.9617
Epoch 88/100
7/7 [==============================] - 0s 8ms/step - loss: 0.1231 -
accuracy: 0.9904
Epoch 89/100
7/7 [==============================] - 0s 6ms/step - loss: 0.1412 -
accuracy: 0.9713
Epoch 90/100
7/7 [==============================] - 0s 7ms/step - loss: 0.1724 -
accuracy: 0.9665
Epoch 91/100
7/7 [==============================] - 0s 7ms/step - loss: 0.1403 -
accuracy: 0.9665
Epoch 92/100
7/7 [==============================] - 0s 7ms/step - loss: 0.1443 -
accuracy: 0.9713
Epoch 93/100
7/7 [==============================] - 0s 6ms/step - loss: 0.1245 -
accuracy: 0.9856
Epoch 94/100
7/7 [==============================] - 0s 9ms/step - loss: 0.1127 -
accuracy: 0.9856
Epoch 95/100
7/7 [==============================] - 0s 9ms/step - loss: 0.1025 -
accuracy: 0.9952
```

```
Epoch 96/100
7/7 [==============================] - 0s 6ms/step - loss: 0.0999 -
accuracy: 0.9952
Epoch 97/100
7/7 [==============================] - 0s 6ms/step - loss: 0.1002 -
accuracy: 0.9904
Epoch 98/100
7/7 [==============================] - 0s 6ms/step - loss: 0.1022 -
accuracy: 0.9904
Epoch 99/100
7/7 [==============================] - 0s 6ms/step - loss: 0.1075 -
accuracy: 0.9904
Epoch 100/100
7/7 [==============================] - 0s 7ms/step - loss: 0.0974 -
accuracy: 0.9856
7/7 [==============================] - 0s 4ms/step


Acurácia no treino: 0.97
2/2 [==============================] - 0s 5ms/step
Acurácia no teste: 0.72
```

Resultado esperado:

```
  Acurácia treino = 81.34%
  Acurácia teste = 52.00%
```

Modelo 2: Testando um modelo com uma camada oculta com 256 neurônios (15pt)

Crie um modelo com uma camada oculta (256 neurônios e ativação ReLu) e a camada de saída com um neurônio (ativação sigmoid).

ToDo: Definição do modelo (10pt)

```
# Definição do modelo
def modelo_2():
  _model = Sequential()
  _model.add(Dense(256, input_shape=(12288,), activation="relu"))
  _model.add(Dense(1, activation = "sigmoid"))
  _model.build(input_shape=(None, 12288))
  return _model
```

Agora treine e teste o seu modelo.

```
np.random.seed(10)
```

```
# Criando o modelo
m2 = modelo_2() # ToDo: chame a função que define o modelo

# Treinando o modelo
m2 = treinar_modelo(m2, treino_x, treino_y.reshape(-1) ) # ToDo: Chame
a função para treinar o modelo




## Predição da rede
print(f'\n\nAcurácia no treino: {accuracy_score(treino_y.ravel(),
(m2.predict(treino_x)>0.5).astype(int)):.2f}') # ToDo: Utilize a
função accuracy_score do sklearn para calcular a acurácia nos dados de
treino
                                               # **dica** use o
model.predict para predizer os dados e use o reshape com -1 nos labels
print(f'Acurácia no teste: {accuracy_score(teste_y.ravel(),
(m2.predict(teste_x)>0.5).astype(int)):.2f}') # ToDo: Utilize a função
accuracy_score do sklearn para calcular a acurácia nos dados de teste
```

```
Model: "sequential_24"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_60 (Dense)            (None, 256)               3145984

 dense_61 (Dense)            (None, 1)                 257

=================================================================
Total params: 3146241 (12.00 MB)
Trainable params: 3146241 (12.00 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
Epoch 1/100
7/7 [==============================] - 1s 46ms/step - loss: 4.7926 -
accuracy: 0.4785
Epoch 2/100
7/7 [==============================] - 0s 45ms/step - loss: 2.2818 -
accuracy: 0.6507
Epoch 3/100
7/7 [==============================] - 0s 39ms/step - loss: 1.0982 -
accuracy: 0.5024
Epoch 4/100
7/7 [==============================] - 0s 39ms/step - loss: 0.8310 -
accuracy: 0.6268
Epoch 5/100
7/7 [==============================] - 0s 42ms/step - loss: 0.5585 -
accuracy: 0.6699
Epoch 6/100
```

```
7/7 [==============================] - 0s 39ms/step - loss: 0.5512 -
accuracy: 0.7177
Epoch 7/100
7/7 [==============================] - 0s 41ms/step - loss: 0.5334 -
accuracy: 0.7464
Epoch 8/100
7/7 [==============================] - 0s 62ms/step - loss: 0.5394 -
accuracy: 0.7656
Epoch 9/100
7/7 [==============================] - 0s 61ms/step - loss: 0.5253 -
accuracy: 0.7273
Epoch 10/100
7/7 [==============================] - 0s 56ms/step - loss: 0.5696 -
accuracy: 0.7273
Epoch 11/100
7/7 [==============================] - 0s 64ms/step - loss: 0.5674 -
accuracy: 0.6842
Epoch 12/100
7/7 [==============================] - 0s 70ms/step - loss: 0.5963 -
accuracy: 0.6938
Epoch 13/100
7/7 [==============================] - 1s 73ms/step - loss: 0.5673 -
accuracy: 0.7177
Epoch 14/100
7/7 [==============================] - 0s 71ms/step - loss: 0.4851 -
accuracy: 0.7751
Epoch 15/100
7/7 [==============================] - 0s 69ms/step - loss: 0.4440 -
accuracy: 0.8038
Epoch 16/100
7/7 [==============================] - 1s 82ms/step - loss: 0.4219 -
accuracy: 0.8373
Epoch 17/100
7/7 [==============================] - 0s 60ms/step - loss: 0.4232 -
accuracy: 0.8421
Epoch 18/100
7/7 [==============================] - 0s 57ms/step - loss: 0.4126 -
accuracy: 0.8421
Epoch 19/100
7/7 [==============================] - 0s 59ms/step - loss: 0.4053 -
accuracy: 0.8517
Epoch 20/100
7/7 [==============================] - 0s 59ms/step - loss: 0.3917 -
accuracy: 0.8421
Epoch 21/100
7/7 [==============================] - 0s 57ms/step - loss: 0.3770 -
accuracy: 0.8565
Epoch 22/100
7/7 [==============================] - 0s 54ms/step - loss: 0.3825 -
```

```
accuracy: 0.8565
Epoch 23/100
7/7 [==============================] - 0s 57ms/step - loss: 0.3985 -
accuracy: 0.8230
Epoch 24/100
7/7 [==============================] - 0s 56ms/step - loss: 0.5062 -
accuracy: 0.7416
Epoch 25/100
7/7 [==============================] - 0s 54ms/step - loss: 0.3465 -
accuracy: 0.8852
Epoch 26/100
7/7 [==============================] - 0s 43ms/step - loss: 0.3423 -
accuracy: 0.8278
Epoch 27/100
7/7 [==============================] - 0s 44ms/step - loss: 0.3379 -
accuracy: 0.8612
Epoch 28/100
7/7 [==============================] - 0s 41ms/step - loss: 0.3244 -
accuracy: 0.8708
Epoch 29/100
7/7 [==============================] - 0s 52ms/step - loss: 0.3019 -
accuracy: 0.9139
Epoch 30/100
7/7 [==============================] - 0s 53ms/step - loss: 0.3093 -
accuracy: 0.9187
Epoch 31/100
7/7 [==============================] - 0s 54ms/step - loss: 0.3016 -
accuracy: 0.9091
Epoch 32/100
7/7 [==============================] - 0s 55ms/step - loss: 0.2987 -
accuracy: 0.8995
Epoch 33/100
7/7 [==============================] - 0s 60ms/step - loss: 0.2734 -
accuracy: 0.9187
Epoch 34/100
7/7 [==============================] - 0s 51ms/step - loss: 0.2600 -
accuracy: 0.9426
Epoch 35/100
7/7 [==============================] - 0s 41ms/step - loss: 0.2614 -
accuracy: 0.9330
Epoch 36/100
7/7 [==============================] - 0s 42ms/step - loss: 0.2571 -
accuracy: 0.9282
Epoch 37/100
7/7 [==============================] - 0s 41ms/step - loss: 0.2499 -
accuracy: 0.9474
Epoch 38/100
7/7 [==============================] - 0s 45ms/step - loss: 0.3171 -
accuracy: 0.8517
```

```
Epoch 39/100
7/7 [==============================] - 0s 44ms/step - loss: 0.3258 -
accuracy: 0.8660
Epoch 40/100
7/7 [==============================] - 0s 41ms/step - loss: 0.2587 -
accuracy: 0.9091
Epoch 41/100
7/7 [==============================] - 0s 44ms/step - loss: 0.2218 -
accuracy: 0.9522
Epoch 42/100
7/7 [==============================] - 0s 42ms/step - loss: 0.2343 -
accuracy: 0.9426
Epoch 43/100
7/7 [==============================] - 0s 43ms/step - loss: 0.2141 -
accuracy: 0.9426
Epoch 44/100
7/7 [==============================] - 0s 40ms/step - loss: 0.1896 -
accuracy: 0.9617
Epoch 45/100
7/7 [==============================] - 0s 45ms/step - loss: 0.1886 -
accuracy: 0.9713
Epoch 46/100
7/7 [==============================] - 0s 56ms/step - loss: 0.2009 -
accuracy: 0.9426
Epoch 47/100
7/7 [==============================] - 0s 71ms/step - loss: 0.1908 -
accuracy: 0.9665
Epoch 48/100
7/7 [==============================] - 0s 69ms/step - loss: 0.1922 -
accuracy: 0.9522
Epoch 49/100
7/7 [==============================] - 1s 72ms/step - loss: 0.1756 -
accuracy: 0.9617
Epoch 50/100
7/7 [==============================] - 0s 72ms/step - loss: 0.1819 -
accuracy: 0.9569
Epoch 51/100
7/7 [==============================] - 1s 74ms/step - loss: 0.1634 -
accuracy: 0.9761
Epoch 52/100
7/7 [==============================] - 1s 77ms/step - loss: 0.1461 -
accuracy: 0.9713
Epoch 53/100
7/7 [==============================] - 0s 68ms/step - loss: 0.1448 -
accuracy: 0.9713
Epoch 54/100
7/7 [==============================] - 0s 49ms/step - loss: 0.1876 -
accuracy: 0.9426
Epoch 55/100
```

```
7/7 [==============================] - 0s 38ms/step - loss: 0.2308 -
accuracy: 0.8947
Epoch 56/100
7/7 [==============================] - 0s 38ms/step - loss: 0.1740 -
accuracy: 0.9378
Epoch 57/100
7/7 [==============================] - 0s 37ms/step - loss: 0.2067 -
accuracy: 0.9139
Epoch 58/100
7/7 [==============================] - 0s 37ms/step - loss: 0.2501 -
accuracy: 0.8708
Epoch 59/100
7/7 [==============================] - 0s 45ms/step - loss: 0.1740 -
accuracy: 0.9474
Epoch 60/100
7/7 [==============================] - 0s 54ms/step - loss: 0.1372 -
accuracy: 0.9761
Epoch 61/100
7/7 [==============================] - 0s 59ms/step - loss: 0.1216 -
accuracy: 0.9856
Epoch 62/100
7/7 [==============================] - 0s 50ms/step - loss: 0.1774 -
accuracy: 0.9426
Epoch 63/100
7/7 [==============================] - 0s 39ms/step - loss: 0.1450 -
accuracy: 0.9761
Epoch 64/100
7/7 [==============================] - 0s 42ms/step - loss: 0.1610 -
accuracy: 0.9522
Epoch 65/100
7/7 [==============================] - 0s 42ms/step - loss: 0.1655 -
accuracy: 0.9282
Epoch 66/100
7/7 [==============================] - 0s 39ms/step - loss: 0.2032 -
accuracy: 0.9091
Epoch 67/100
7/7 [==============================] - 0s 43ms/step - loss: 0.2718 -
accuracy: 0.8612
Epoch 68/100
7/7 [==============================] - 0s 43ms/step - loss: 0.4134 -
accuracy: 0.8038
Epoch 69/100
7/7 [==============================] - 0s 41ms/step - loss: 0.3897 -
accuracy: 0.8134
Epoch 70/100
7/7 [==============================] - 0s 46ms/step - loss: 0.2430 -
accuracy: 0.8852
Epoch 71/100
7/7 [==============================] - 0s 41ms/step - loss: 0.1775 -
```

```
accuracy: 0.9330
Epoch 72/100
7/7 [==============================] - 0s 39ms/step - loss: 0.1142 -
accuracy: 0.9713
Epoch 73/100
7/7 [==============================] - 0s 45ms/step - loss: 0.0952 -
accuracy: 0.9856
Epoch 74/100
7/7 [==============================] - 0s 41ms/step - loss: 0.0902 -
accuracy: 0.9809
Epoch 75/100
7/7 [==============================] - 0s 40ms/step - loss: 0.0776 -
accuracy: 0.9904
Epoch 76/100
7/7 [==============================] - 0s 43ms/step - loss: 0.0910 -
accuracy: 0.9952
Epoch 77/100
7/7 [==============================] - 0s 46ms/step - loss: 0.0762 -
accuracy: 0.9856
Epoch 78/100
7/7 [==============================] - 0s 43ms/step - loss: 0.0729 -
accuracy: 0.9904
Epoch 79/100
7/7 [==============================] - 0s 42ms/step - loss: 0.0734 -
accuracy: 0.9904
Epoch 80/100
7/7 [==============================] - 0s 43ms/step - loss: 0.0739 -
accuracy: 0.9952
Epoch 81/100
7/7 [==============================] - 0s 46ms/step - loss: 0.0735 -
accuracy: 0.9952
Epoch 82/100
7/7 [==============================] - 0s 46ms/step - loss: 0.0708 -
accuracy: 0.9904
Epoch 83/100
7/7 [==============================] - 0s 58ms/step - loss: 0.0680 -
accuracy: 0.9952
Epoch 84/100
7/7 [==============================] - 0s 62ms/step - loss: 0.0694 -
accuracy: 0.9952
Epoch 85/100
7/7 [==============================] - 0s 38ms/step - loss: 0.0644 -
accuracy: 0.9904
Epoch 86/100
7/7 [==============================] - 0s 69ms/step - loss: 0.0710 -
accuracy: 0.9952
Epoch 87/100
7/7 [==============================] - 0s 54ms/step - loss: 0.0562 -
accuracy: 0.9904
```

```
Epoch 88/100
7/7 [==============================] - 0s 68ms/step - loss: 0.0646 -
accuracy: 0.9952
Epoch 89/100
7/7 [==============================] - 0s 68ms/step - loss: 0.0594 -
accuracy: 0.9952
Epoch 90/100
7/7 [==============================] - 0s 68ms/step - loss: 0.0570 -
accuracy: 1.0000
Epoch 91/100
7/7 [==============================] - 0s 70ms/step - loss: 0.0551 -
accuracy: 1.0000
Epoch 92/100
7/7 [==============================] - 0s 64ms/step - loss: 0.0550 -
accuracy: 0.9904
Epoch 93/100
7/7 [==============================] - 0s 66ms/step - loss: 0.0738 -
accuracy: 0.9856
Epoch 94/100
7/7 [==============================] - 0s 67ms/step - loss: 0.0633 -
accuracy: 0.9904
Epoch 95/100
7/7 [==============================] - 0s 63ms/step - loss: 0.0503 -
accuracy: 1.0000
Epoch 96/100
7/7 [==============================] - 0s 41ms/step - loss: 0.0465 -
accuracy: 0.9952
Epoch 97/100
7/7 [==============================] - 0s 40ms/step - loss: 0.0464 -
accuracy: 0.9952
Epoch 98/100
7/7 [==============================] - 0s 41ms/step - loss: 0.0466 -
accuracy: 0.9952
Epoch 99/100
7/7 [==============================] - 0s 37ms/step - loss: 0.0450 -
accuracy: 0.9952
Epoch 100/100
7/7 [==============================] - 0s 40ms/step - loss: 0.0417 -
accuracy: 1.0000
7/7 [==============================] - 0s 8ms/step


Acurácia no treino: 1.00
2/2 [==============================] - 0s 9ms/step
Acurácia no teste: 0.70
```

Resultado esperado:

```
Acurácia treino = 100.00%
Acurácia teste = 70%
```

ToDo: Análise dos resultados (5pt)

Por que você obteve 100% no treino e apenas 80% no teste no segundo modelo e resultados piores no primeiro modelo?

Existem várias razões pelas quais um modelo de aprendizado de máquina pode apresentar um desempenho melhor nos dados de treino em comparação com os dados de teste, levando a uma discrepância entre as taxas de acerto: Sobreajuste, ocorre quando é muito complexo com relação a complexidade dos dados, o outro fator é o tamanho do conjunto de treino e teste, se o conjunto de treinamento for muito pequena com relação a complixidade do modelo, o modelo pode aprender os dados de treinamento muito bem, mas não conseguir generalizar aos dados novos. técnicas como validação cruzada podem ser aplicadas para obter uma avaliação mais robusta do desempenho dos modelos.

Modelo 3: Testando com uma rede com três camadas ocultas (15pt)

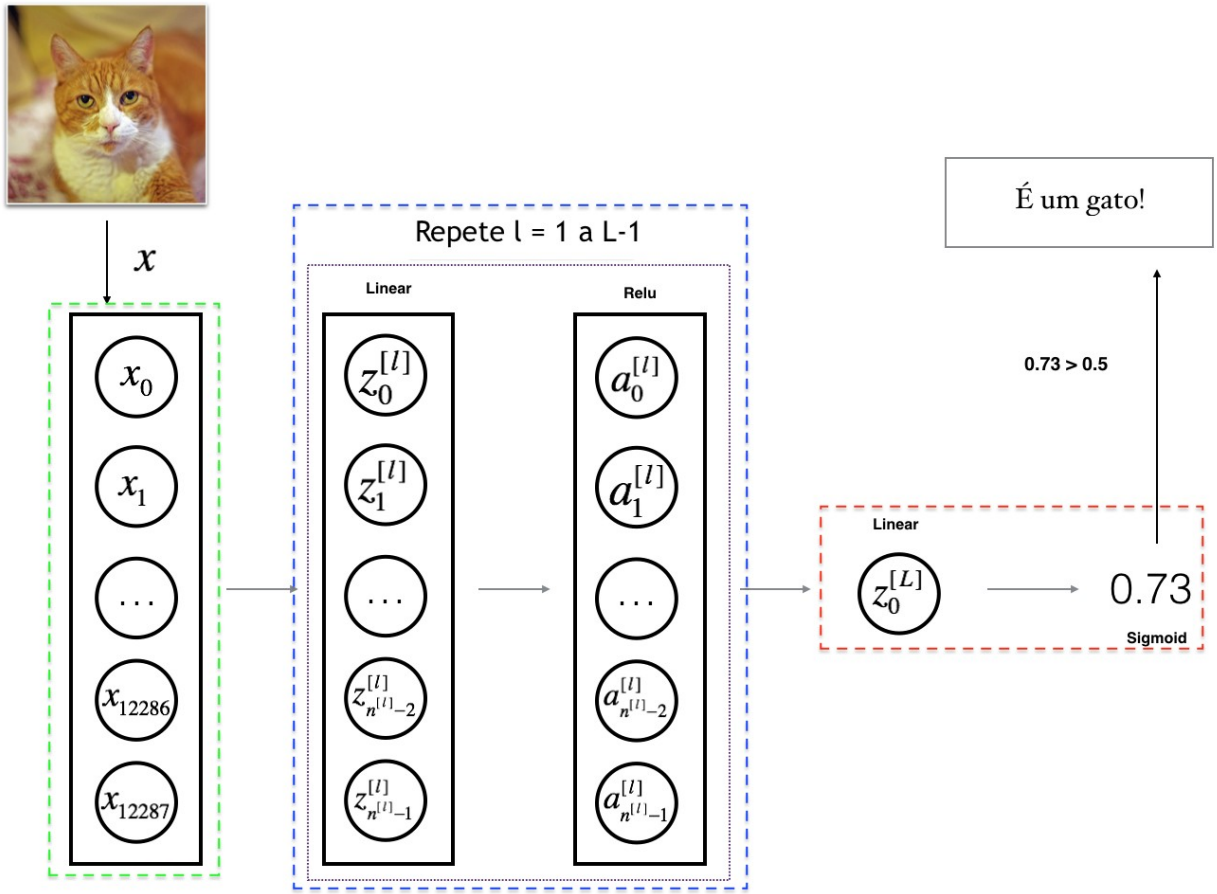


Figura 8: Rede neural com L camadas. Resumo do modelo: ***ENTRADA -> LINEAR -> RELU -> LINEAR -> SIGMOID -> SAIDA***.

Crie um modelo com três camadas ocultas e a camada de saída com um neurônio. Você deve seguir a seguinte estrutura:

1. Camada oculta 1 - 256 neurônios e ativação ReLU.
2. Camada oculta 2 - 64 neurônios e ativação ReLU.
3. Camada oculta 3 - 8 neurônios e ativação ReLU.
4. Camada de saída - 1 neurônio e ativação sigmoid.

ToDo: Definição do modelo (10pt)

```
# Definição do modelo
def modelo_3():
  _model = Sequential()
  _model.add(Dense(256, input_shape=(12288,), activation="relu"))
  _model.add(Dense(64, activation="relu"))
  _model.add(Dense(8, activation="relu"))
  _model.add(Dense(1, activation="sigmoid"))
  return _model
```

Agora treine e teste o seu modelo.

```
np.random.seed(1)

# Criando o modelo
m3 =  modelo_3()#ToDo: chame a função que define o modelo

# Treinando o modelo
m3 = treinar_modelo(m3, treino_x, treino_y.reshape(-1) ) # ToDo: Chame
a função para treinar o modelo

## Predição da rede
print(f'\n\nAcurácia no treino: {accuracy_score(treino_y.ravel(),
(m3.predict(treino_x)>0.5).astype(int)):.2f}') # ToDo: Utilize a
função accuracy_score do sklearn para calcular a acurácia nos dados de
treino
                                          # **dica** use o
model.predict para predizer os dados e use o reshape com -1 nos labels
print(f'Acurácia no teste: {accuracy_score(teste_y.ravel(),
(m3.predict(teste_x)>0.5).astype(int)):.2f}') # ToDo: Utilize a função
accuracy_score do sklearn para calcular a acurácia nos dados de teste
```

```
Model: "sequential_25"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_62 (Dense)            (None, 256)               3145984

 dense_63 (Dense)            (None, 64)                16448

 dense_64 (Dense)            (None, 8)                 520
```

```
 dense_65 (Dense)            (None, 1)                 9

=================================================================
Total params: 3162961 (12.07 MB)
Trainable params: 3162961 (12.07 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
Epoch 1/100
7/7 [==============================] - 1s 60ms/step - loss: 1.0917 -
accuracy: 0.5359
Epoch 2/100
7/7 [==============================] - 0s 59ms/step - loss: 0.8083 -
accuracy: 0.4689
Epoch 3/100
7/7 [==============================] - 0s 62ms/step - loss: 0.7074 -
accuracy: 0.5215
Epoch 4/100
7/7 [==============================] - 0s 53ms/step - loss: 0.6604 -
accuracy: 0.6555
Epoch 5/100
7/7 [==============================] - 0s 59ms/step - loss: 0.6601 -
accuracy: 0.6555
Epoch 6/100
7/7 [==============================] - 0s 57ms/step - loss: 0.6556 -
accuracy: 0.6555
Epoch 7/100
7/7 [==============================] - 0s 55ms/step - loss: 0.6554 -
accuracy: 0.6555
Epoch 8/100
7/7 [==============================] - 0s 43ms/step - loss: 0.6544 -
accuracy: 0.6555
Epoch 9/100
7/7 [==============================] - 0s 38ms/step - loss: 0.6531 -
accuracy: 0.6555
Epoch 10/100
7/7 [==============================] - 0s 45ms/step - loss: 0.6479 -
accuracy: 0.6555
Epoch 11/100
7/7 [==============================] - 0s 43ms/step - loss: 0.6451 -
accuracy: 0.6555
Epoch 12/100
7/7 [==============================] - 0s 46ms/step - loss: 0.6433 -
accuracy: 0.6555
Epoch 13/100
7/7 [==============================] - 0s 49ms/step - loss: 0.6388 -
accuracy: 0.6555
Epoch 14/100
7/7 [==============================] - 0s 49ms/step - loss: 0.6403 -
```

```
accuracy: 0.6555
Epoch 15/100
7/7 [==============================] - 0s 63ms/step - loss: 0.6296 -
accuracy: 0.6555
Epoch 16/100
7/7 [==============================] - 0s 60ms/step - loss: 0.6282 -
accuracy: 0.6555
Epoch 17/100
7/7 [==============================] - 0s 69ms/step - loss: 0.6150 -
accuracy: 0.6555
Epoch 18/100
7/7 [==============================] - 0s 70ms/step - loss: 0.6143 -
accuracy: 0.6555
Epoch 19/100
7/7 [==============================] - 0s 68ms/step - loss: 0.6083 -
accuracy: 0.6555
Epoch 20/100
7/7 [==============================] - 0s 57ms/step - loss: 0.5896 -
accuracy: 0.6555
Epoch 21/100
7/7 [==============================] - 0s 68ms/step - loss: 0.5773 -
accuracy: 0.6555
Epoch 22/100
7/7 [==============================] - 0s 63ms/step - loss: 0.5621 -
accuracy: 0.6555
Epoch 23/100
7/7 [==============================] - 1s 82ms/step - loss: 0.5554 -
accuracy: 0.6555
Epoch 24/100
7/7 [==============================] - 0s 66ms/step - loss: 0.6399 -
accuracy: 0.6555
Epoch 25/100
7/7 [==============================] - 0s 46ms/step - loss: 0.6083 -
accuracy: 0.6555
Epoch 26/100
7/7 [==============================] - 0s 42ms/step - loss: 0.6003 -
accuracy: 0.6555
Epoch 27/100
7/7 [==============================] - 0s 43ms/step - loss: 0.6087 -
accuracy: 0.6794
Epoch 28/100
7/7 [==============================] - 0s 41ms/step - loss: 0.6004 -
accuracy: 0.6794
Epoch 29/100
7/7 [==============================] - 0s 44ms/step - loss: 0.5846 -
accuracy: 0.7273
Epoch 30/100
7/7 [==============================] - 0s 45ms/step - loss: 0.5870 -
accuracy: 0.6555
```

```
Epoch 31/100
7/7 [==============================] - 0s 41ms/step - loss: 0.5659 -
accuracy: 0.7560
Epoch 32/100
7/7 [==============================] - 0s 48ms/step - loss: 0.5411 -
accuracy: 0.7033
Epoch 33/100
7/7 [==============================] - 0s 45ms/step - loss: 0.5294 -
accuracy: 0.7129
Epoch 34/100
7/7 [==============================] - 0s 42ms/step - loss: 0.5456 -
accuracy: 0.7033
Epoch 35/100
7/7 [==============================] - 0s 49ms/step - loss: 0.5267 -
accuracy: 0.6938
Epoch 36/100
7/7 [==============================] - 0s 49ms/step - loss: 0.5035 -
accuracy: 0.7608
Epoch 37/100
7/7 [==============================] - 0s 44ms/step - loss: 0.5564 -
accuracy: 0.7273
Epoch 38/100
7/7 [==============================] - 0s 45ms/step - loss: 0.5592 -
accuracy: 0.6938
Epoch 39/100
7/7 [==============================] - 0s 44ms/step - loss: 0.5156 -
accuracy: 0.7608
Epoch 40/100
7/7 [==============================] - 0s 41ms/step - loss: 0.4693 -
accuracy: 0.8230
Epoch 41/100
7/7 [==============================] - 0s 43ms/step - loss: 0.5001 -
accuracy: 0.7608
Epoch 42/100
7/7 [==============================] - 0s 42ms/step - loss: 0.4825 -
accuracy: 0.7464
Epoch 43/100
7/7 [==============================] - 0s 47ms/step - loss: 0.4492 -
accuracy: 0.8278
Epoch 44/100
7/7 [==============================] - 0s 43ms/step - loss: 0.4650 -
accuracy: 0.7656
Epoch 45/100
7/7 [==============================] - 0s 44ms/step - loss: 0.4673 -
accuracy: 0.7560
Epoch 46/100
7/7 [==============================] - 0s 46ms/step - loss: 0.4478 -
accuracy: 0.8182
Epoch 47/100
```

```
7/7 [==============================] - 0s 42ms/step - loss: 0.4314 -
accuracy: 0.8278
Epoch 48/100
7/7 [==============================] - 0s 43ms/step - loss: 0.4237 -
accuracy: 0.8134
Epoch 49/100
7/7 [==============================] - 0s 46ms/step - loss: 0.4267 -
accuracy: 0.8373
Epoch 50/100
7/7 [==============================] - 0s 42ms/step - loss: 0.3872 -
accuracy: 0.8756
Epoch 51/100
7/7 [==============================] - 0s 38ms/step - loss: 0.3711 -
accuracy: 0.8517
Epoch 52/100
7/7 [==============================] - 0s 43ms/step - loss: 0.3592 -
accuracy: 0.8421
Epoch 53/100
7/7 [==============================] - 0s 44ms/step - loss: 0.3957 -
accuracy: 0.8325
Epoch 54/100
7/7 [==============================] - 0s 45ms/step - loss: 0.4657 -
accuracy: 0.7751
Epoch 55/100
7/7 [==============================] - 0s 64ms/step - loss: 0.4513 -
accuracy: 0.7847
Epoch 56/100
7/7 [==============================] - 0s 53ms/step - loss: 0.3614 -
accuracy: 0.8230
Epoch 57/100
7/7 [==============================] - 0s 64ms/step - loss: 0.4149 -
accuracy: 0.7990
Epoch 58/100
7/7 [==============================] - 0s 70ms/step - loss: 0.3428 -
accuracy: 0.8804
Epoch 59/100
7/7 [==============================] - 0s 59ms/step - loss: 0.3099 -
accuracy: 0.8900
Epoch 60/100
7/7 [==============================] - 0s 66ms/step - loss: 0.2899 -
accuracy: 0.8947
Epoch 61/100
7/7 [==============================] - 0s 60ms/step - loss: 0.2919 -
accuracy: 0.8995
Epoch 62/100
7/7 [==============================] - 0s 68ms/step - loss: 0.2732 -
accuracy: 0.8995
Epoch 63/100
7/7 [==============================] - 0s 59ms/step - loss: 0.2715 -
```

```
accuracy: 0.8900
Epoch 64/100
7/7 [==============================] - 0s 63ms/step - loss: 0.2538 -
accuracy: 0.9139
Epoch 65/100
7/7 [==============================] - 0s 51ms/step - loss: 0.2213 -
accuracy: 0.9330
Epoch 66/100
7/7 [==============================] - 0s 49ms/step - loss: 0.2323 -
accuracy: 0.9043
Epoch 67/100
7/7 [==============================] - 0s 45ms/step - loss: 0.1939 -
accuracy: 0.9522
Epoch 68/100
7/7 [==============================] - 0s 40ms/step - loss: 0.1828 -
accuracy: 0.9378
Epoch 69/100
7/7 [==============================] - 0s 43ms/step - loss: 0.1654 -
accuracy: 0.9522
Epoch 70/100
7/7 [==============================] - 0s 44ms/step - loss: 0.1627 -
accuracy: 0.9474
Epoch 71/100
7/7 [==============================] - 0s 48ms/step - loss: 0.1492 -
accuracy: 0.9522
Epoch 72/100
7/7 [==============================] - 0s 43ms/step - loss: 0.1567 -
accuracy: 0.9522
Epoch 73/100
7/7 [==============================] - 0s 43ms/step - loss: 0.1702 -
accuracy: 0.9187
Epoch 74/100
7/7 [==============================] - 0s 46ms/step - loss: 0.2010 -
accuracy: 0.9187
Epoch 75/100
7/7 [==============================] - 0s 51ms/step - loss: 0.1553 -
accuracy: 0.9474
Epoch 76/100
7/7 [==============================] - 0s 46ms/step - loss: 0.1349 -
accuracy: 0.9617
Epoch 77/100
7/7 [==============================] - 0s 40ms/step - loss: 0.1225 -
accuracy: 0.9569
Epoch 78/100
7/7 [==============================] - 0s 46ms/step - loss: 0.0991 -
accuracy: 0.9761
Epoch 79/100
7/7 [==============================] - 0s 45ms/step - loss: 0.0849 -
accuracy: 0.9904
```

```
Epoch 80/100
7/7 [==============================] - 0s 47ms/step - loss: 0.0691 -
accuracy: 0.9904
Epoch 81/100
7/7 [==============================] - 0s 46ms/step - loss: 0.0665 -
accuracy: 0.9904
Epoch 82/100
7/7 [==============================] - 0s 41ms/step - loss: 0.0731 -
accuracy: 0.9856
Epoch 83/100
7/7 [==============================] - 0s 44ms/step - loss: 0.0750 -
accuracy: 0.9809
Epoch 84/100
7/7 [==============================] - 0s 50ms/step - loss: 0.0942 -
accuracy: 0.9761
Epoch 85/100
7/7 [==============================] - 0s 45ms/step - loss: 0.0840 -
accuracy: 0.9809
Epoch 86/100
7/7 [==============================] - 0s 45ms/step - loss: 0.1448 -
accuracy: 0.9474
Epoch 87/100
7/7 [==============================] - 0s 46ms/step - loss: 0.0849 -
accuracy: 0.9761
Epoch 88/100
7/7 [==============================] - 0s 42ms/step - loss: 0.0404 -
accuracy: 0.9952
Epoch 89/100
7/7 [==============================] - 0s 42ms/step - loss: 0.0349 -
accuracy: 1.0000
Epoch 90/100
7/7 [==============================] - 0s 44ms/step - loss: 0.0335 -
accuracy: 0.9952
Epoch 91/100
7/7 [==============================] - 0s 44ms/step - loss: 0.0281 -
accuracy: 1.0000
Epoch 92/100
7/7 [==============================] - 0s 43ms/step - loss: 0.0293 -
accuracy: 1.0000
Epoch 93/100
7/7 [==============================] - 0s 44ms/step - loss: 0.0257 -
accuracy: 1.0000
Epoch 94/100
7/7 [==============================] - 0s 40ms/step - loss: 0.0260 -
accuracy: 1.0000
Epoch 95/100
7/7 [==============================] - 0s 42ms/step - loss: 0.0270 -
accuracy: 1.0000
Epoch 96/100
```

```
7/7 [==============================] - 0s 53ms/step - loss: 0.0252 -
accuracy: 1.0000
Epoch 97/100
7/7 [==============================] - 0s 70ms/step - loss: 0.0208 -
accuracy: 1.0000
Epoch 98/100
7/7 [==============================] - 0s 53ms/step - loss: 0.0173 -
accuracy: 1.0000
Epoch 99/100
7/7 [==============================] - 0s 56ms/step - loss: 0.0164 -
accuracy: 1.0000
Epoch 100/100
7/7 [==============================] - 0s 67ms/step - loss: 0.0160 -
accuracy: 1.0000
7/7 [==============================] - 0s 10ms/step


Acurácia no treino: 1.00
2/2 [==============================] - 0s 11ms/step
Acurácia no teste: 0.76
```

Resultado esperado:

```
  Acurácia treino = 100.00%
  Acurácia teste = 76%
```

ToDo: Análise dos resultados (5pt)

O resultado com três camadas ocultas foi melhor ou pior do que usa somente uma camada? Tente explicar os motivos.

A avaliação do desempenho de um modelo com três camadas ocultas em comparação com um modelo de uma única camada é um processo empírico e depende de vários fatores, incluindo a complexidade do problema, a quantidade e qualidade dos dados, a escolha adequada das funções de ativação e a habilidade do modelo de generalizar padrões dos dados. olhando para os resultados é dificil inferir que os resultados com 3 camadas ocultas foram melhor ou pior do que usando apenas uma camada, uma vez que pelos resultados quase iguais para ambas as camadas, quando se olha pelos resultados, o valor de perde quase esta nos limites nas ambas as camadas

Testando uma rede que você desenvolveu (15pt)

Crie uma arquitetura e treine/teste o seu modelo

ToDo: Definição do modelo (10pt)

```
# Definição do modelo
def meu_modelo():
  _model = Sequential ()
```

```
  _model.add(Dense(8, input_shape=(12288,), activation="relu"))
  _model.add(Dense(8, activation="relu"))
  _model.add(Dense(4, activation="relu"))
  _model.add(Dense(1, activation="softmax"))
  return _model

np.random.seed(1)

# Criando o modelo
m4 = meu_modelo() # ToDo: chame a função que define o modelo

# Treinando o modelo
m4 = treinar_modelo(m4, treino_x, treino_y.reshape(-1)) # ToDo: Chame
a função para treinar o modelo

## Predição da rede
print(f'\n\nAcurácia no treino: {accuracy_score(treino_y.ravel(),
(m4.predict(treino_x)>0.5).astype(int))}') # ToDo: Utilize a função
accuracy_score do sklearn para calcular a acurácia nos dados de treino
                                          # **dica** use o
model.predict para predizer os dados e use o reshape com -1 nos labels
print(f'Acurácia no teste: {accuracy_score(teste_y.ravel(),
(m4.predict(teste_x)>0.5).astype(int))}') # ToDo: Utilize a função
accuracy_score do sklearn para calcular a acurácia nos dados de teste

Model: "sequential_26"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_66 (Dense)            (None, 8)                 98312

 dense_67 (Dense)            (None, 8)                 72

 dense_68 (Dense)            (None, 4)                 36

 dense_69 (Dense)            (None, 1)                 5

=================================================================
Total params: 98425 (384.47 KB)
Trainable params: 98425 (384.47 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
Epoch 1/100
7/7 [==============================] - 1s 7ms/step - loss: 0.6657 -
accuracy: 0.3445
Epoch 2/100
7/7 [==============================] - 0s 7ms/step - loss: 0.7562 -
accuracy: 0.3445
Epoch 3/100
7/7 [==============================] - 0s 12ms/step - loss: 0.8195 -
```

```
accuracy: 0.3445
Epoch 4/100
7/7 [==============================] - 0s 12ms/step - loss: 0.6572 -
accuracy: 0.3445
Epoch 5/100
7/7 [==============================] - 0s 10ms/step - loss: 0.6386 -
accuracy: 0.3445
Epoch 6/100
7/7 [==============================] - 0s 9ms/step - loss: 0.6290 -
accuracy: 0.3445
Epoch 7/100
7/7 [==============================] - 0s 9ms/step - loss: 0.6328 -
accuracy: 0.3445
Epoch 8/100
7/7 [==============================] - 0s 8ms/step - loss: 0.6056 -
accuracy: 0.3445
Epoch 9/100
7/7 [==============================] - 0s 8ms/step - loss: 0.6145 -
accuracy: 0.3445
Epoch 10/100
7/7 [==============================] - 0s 8ms/step - loss: 0.5925 -
accuracy: 0.3445
Epoch 11/100
7/7 [==============================] - 0s 8ms/step - loss: 0.5833 -
accuracy: 0.3445
Epoch 12/100
7/7 [==============================] - 0s 8ms/step - loss: 0.6034 -
accuracy: 0.3445
Epoch 13/100
7/7 [==============================] - 0s 8ms/step - loss: 0.5878 -
accuracy: 0.3445
Epoch 14/100
7/7 [==============================] - 0s 8ms/step - loss: 0.5212 -
accuracy: 0.3445
Epoch 15/100
7/7 [==============================] - 0s 8ms/step - loss: 0.5019 -
accuracy: 0.3445
Epoch 16/100
7/7 [==============================] - 0s 8ms/step - loss: 0.4513 -
accuracy: 0.3445
Epoch 17/100
7/7 [==============================] - 0s 8ms/step - loss: 0.5447 -
accuracy: 0.3445
Epoch 18/100
7/7 [==============================] - 0s 9ms/step - loss: 0.4382 -
accuracy: 0.3445
Epoch 19/100
7/7 [==============================] - 0s 9ms/step - loss: 0.4786 -
accuracy: 0.3445
```

```
Epoch 20/100
7/7 [==============================] - 0s 8ms/step - loss: 0.5177 -
accuracy: 0.3445
Epoch 21/100
7/7 [==============================] - 0s 10ms/step - loss: 0.5084 -
accuracy: 0.3445
Epoch 22/100
7/7 [==============================] - 0s 9ms/step - loss: 0.5526 -
accuracy: 0.3445
Epoch 23/100
7/7 [==============================] - 0s 8ms/step - loss: 0.4030 -
accuracy: 0.3445
Epoch 24/100
7/7 [==============================] - 0s 7ms/step - loss: 0.4241 -
accuracy: 0.3445
Epoch 25/100
7/7 [==============================] - 0s 7ms/step - loss: 0.3921 -
accuracy: 0.3445
Epoch 26/100
7/7 [==============================] - 0s 8ms/step - loss: 0.4060 -
accuracy: 0.3445
Epoch 27/100
7/7 [==============================] - 0s 9ms/step - loss: 0.4274 -
accuracy: 0.3445
Epoch 28/100
7/7 [==============================] - 0s 8ms/step - loss: 0.5238 -
accuracy: 0.3445
Epoch 29/100
7/7 [==============================] - 0s 8ms/step - loss: 0.4143 -
accuracy: 0.3445
Epoch 30/100
7/7 [==============================] - 0s 8ms/step - loss: 0.3400 -
accuracy: 0.3445
Epoch 31/100
7/7 [==============================] - 0s 8ms/step - loss: 0.4101 -
accuracy: 0.3445
Epoch 32/100
7/7 [==============================] - 0s 9ms/step - loss: 0.3782 -
accuracy: 0.3445
Epoch 33/100
7/7 [==============================] - 0s 8ms/step - loss: 0.4232 -
accuracy: 0.3445
Epoch 34/100
7/7 [==============================] - 0s 8ms/step - loss: 0.4435 -
accuracy: 0.3445
Epoch 35/100
7/7 [==============================] - 0s 9ms/step - loss: 0.3443 -
accuracy: 0.3445
Epoch 36/100
```

```
7/7 [==============================] - 0s 8ms/step - loss: 0.3520 -
accuracy: 0.3445
Epoch 37/100
7/7 [==============================] - 0s 8ms/step - loss: 0.2836 -
accuracy: 0.3445
Epoch 38/100
7/7 [==============================] - 0s 8ms/step - loss: 0.2690 -
accuracy: 0.3445
Epoch 39/100
7/7 [==============================] - 0s 8ms/step - loss: 0.2596 -
accuracy: 0.3445
Epoch 40/100
7/7 [==============================] - 0s 8ms/step - loss: 0.2595 -
accuracy: 0.3445
Epoch 41/100
7/7 [==============================] - 0s 9ms/step - loss: 0.3003 -
accuracy: 0.3445
Epoch 42/100
7/7 [==============================] - 0s 10ms/step - loss: 0.2816 -
accuracy: 0.3445
Epoch 43/100
7/7 [==============================] - 0s 9ms/step - loss: 0.2406 -
accuracy: 0.3445
Epoch 44/100
7/7 [==============================] - 0s 8ms/step - loss: 0.2337 -
accuracy: 0.3445
Epoch 45/100
7/7 [==============================] - 0s 8ms/step - loss: 0.2313 -
accuracy: 0.3445
Epoch 46/100
7/7 [==============================] - 0s 8ms/step - loss: 0.2424 -
accuracy: 0.3445
Epoch 47/100
7/7 [==============================] - 0s 9ms/step - loss: 0.2176 -
accuracy: 0.3445
Epoch 48/100
7/7 [==============================] - 0s 8ms/step - loss: 0.2169 -
accuracy: 0.3445
Epoch 49/100
7/7 [==============================] - 0s 8ms/step - loss: 0.2265 -
accuracy: 0.3445
Epoch 50/100
7/7 [==============================] - 0s 8ms/step - loss: 0.2019 -
accuracy: 0.3445
Epoch 51/100
7/7 [==============================] - 0s 10ms/step - loss: 0.1928 -
accuracy: 0.3445
Epoch 52/100
7/7 [==============================] - 0s 9ms/step - loss: 0.1983 -
```

```
accuracy: 0.3445
Epoch 53/100
7/7 [==============================] - 0s 9ms/step - loss: 0.2126 -
accuracy: 0.3445
Epoch 54/100
7/7 [==============================] - 0s 9ms/step - loss: 0.2101 -
accuracy: 0.3445
Epoch 55/100
7/7 [==============================] - 0s 8ms/step - loss: 0.1830 -
accuracy: 0.3445
Epoch 56/100
7/7 [==============================] - 0s 8ms/step - loss: 0.1773 -
accuracy: 0.3445
Epoch 57/100
7/7 [==============================] - 0s 10ms/step - loss: 0.1724 -
accuracy: 0.3445
Epoch 58/100
7/7 [==============================] - 0s 8ms/step - loss: 0.1629 -
accuracy: 0.3445
Epoch 59/100
7/7 [==============================] - 0s 8ms/step - loss: 0.1793 -
accuracy: 0.3445
Epoch 60/100
7/7 [==============================] - 0s 8ms/step - loss: 0.1859 -
accuracy: 0.3445
Epoch 61/100
7/7 [==============================] - 0s 8ms/step - loss: 0.1715 -
accuracy: 0.3445
Epoch 62/100
7/7 [==============================] - 0s 8ms/step - loss: 0.1463 -
accuracy: 0.3445
Epoch 63/100
7/7 [==============================] - 0s 8ms/step - loss: 0.1434 -
accuracy: 0.3445
Epoch 64/100
7/7 [==============================] - 0s 8ms/step - loss: 0.1556 -
accuracy: 0.3445
Epoch 65/100
7/7 [==============================] - 0s 8ms/step - loss: 0.1800 -
accuracy: 0.3445
Epoch 66/100
7/7 [==============================] - 0s 8ms/step - loss: 0.1621 -
accuracy: 0.3445
Epoch 67/100
7/7 [==============================] - 0s 11ms/step - loss: 0.1765 -
accuracy: 0.3445
Epoch 68/100
7/7 [==============================] - 0s 8ms/step - loss: 0.1427 -
accuracy: 0.3445
```

```
Epoch 69/100
7/7 [==============================] - 0s 8ms/step - loss: 0.1442 -
accuracy: 0.3445
Epoch 70/100
7/7 [==============================] - 0s 8ms/step - loss: 0.1234 -
accuracy: 0.3445
Epoch 71/100
7/7 [==============================] - 0s 9ms/step - loss: 0.1111 -
accuracy: 0.3445
Epoch 72/100
7/7 [==============================] - 0s 10ms/step - loss: 0.1064 -
accuracy: 0.3445
Epoch 73/100
7/7 [==============================] - 0s 8ms/step - loss: 0.1111 -
accuracy: 0.3445
Epoch 74/100
7/7 [==============================] - 0s 9ms/step - loss: 0.1060 -
accuracy: 0.3445
Epoch 75/100
7/7 [==============================] - 0s 8ms/step - loss: 0.1042 -
accuracy: 0.3445
Epoch 76/100
7/7 [==============================] - 0s 8ms/step - loss: 0.1148 -
accuracy: 0.3445
Epoch 77/100
7/7 [==============================] - 0s 8ms/step - loss: 0.1200 -
accuracy: 0.3445
Epoch 78/100
7/7 [==============================] - 0s 8ms/step - loss: 0.1551 -
accuracy: 0.3445
Epoch 79/100
7/7 [==============================] - 0s 8ms/step - loss: 0.1253 -
accuracy: 0.3445
Epoch 80/100
7/7 [==============================] - 0s 8ms/step - loss: 0.1061 -
accuracy: 0.3445
Epoch 81/100
7/7 [==============================] - 0s 7ms/step - loss: 0.1285 -
accuracy: 0.3445
Epoch 82/100
7/7 [==============================] - 0s 8ms/step - loss: 0.1887 -
accuracy: 0.3445
Epoch 83/100
7/7 [==============================] - 0s 7ms/step - loss: 0.1617 -
accuracy: 0.3445
Epoch 84/100
7/7 [==============================] - 0s 6ms/step - loss: 0.1543 -
accuracy: 0.3445
Epoch 85/100
```

```
7/7 [==============================] - 0s 6ms/step - loss: 0.1258 -
accuracy: 0.3445
Epoch 86/100
7/7 [==============================] - 0s 6ms/step - loss: 0.1367 -
accuracy: 0.3445
Epoch 87/100
7/7 [==============================] - 0s 6ms/step - loss: 0.0861 -
accuracy: 0.3445
Epoch 88/100
7/7 [==============================] - 0s 6ms/step - loss: 0.0698 -
accuracy: 0.3445
Epoch 89/100
7/7 [==============================] - 0s 6ms/step - loss: 0.0701 -
accuracy: 0.3445
Epoch 90/100
7/7 [==============================] - 0s 6ms/step - loss: 0.0685 -
accuracy: 0.3445
Epoch 91/100
7/7 [==============================] - 0s 6ms/step - loss: 0.0705 -
accuracy: 0.3445
Epoch 92/100
7/7 [==============================] - 0s 6ms/step - loss: 0.0739 -
accuracy: 0.3445
Epoch 93/100
7/7 [==============================] - 0s 6ms/step - loss: 0.0835 -
accuracy: 0.3445
Epoch 94/100
7/7 [==============================] - 0s 6ms/step - loss: 0.1014 -
accuracy: 0.3445
Epoch 95/100
7/7 [==============================] - 0s 7ms/step - loss: 0.0745 -
accuracy: 0.3445
Epoch 96/100
7/7 [==============================] - 0s 6ms/step - loss: 0.0832 -
accuracy: 0.3445
Epoch 97/100
7/7 [==============================] - 0s 6ms/step - loss: 0.0624 -
accuracy: 0.3445
Epoch 98/100
7/7 [==============================] - 0s 6ms/step - loss: 0.0525 -
accuracy: 0.3445
Epoch 99/100
7/7 [==============================] - 0s 5ms/step - loss: 0.0549 -
accuracy: 0.3445
Epoch 100/100
7/7 [==============================] - 0s 5ms/step - loss: 0.0507 -
accuracy: 0.3445
7/7 [==============================] - 0s 3ms/step
```

```
Acurácia no treino: 0.3444976076555024
2/2 [==============================] - 0s 6ms/step
Acurácia no teste: 0.66
```

ToDo: Análise dos resultados (5pt)

O que você pode falar do seu modelo? Como ele se saiu em relação aos outros três modelos?

Acurancia no Treino é de 34%, o que significa que o modelo teve dificuldade de aprender os padrões presentes nesses dados. enquanto os dados de teste acurancia é de 66%, o que significa que o modelo esta generalizar melhor para os dados não vistos durante o treinamento.

Variando alguns hiperparâmetros (20pt)

Usando o framework do tensorflow/keras, altere os hiperparâmetros e veja o impacto (gere pelo menos dois novos modelos):

- learning rate.
- Algoritmo de otimização (SGD com momento, ADAM, ADADELTA, RMSPROP).
- inicialização dos pesos: inicialiação aleatória vs uniforme.
- Funções de ativação : troque a sigmoid por (ReLU, GELU, Leaky RELU).

Você criar uma nova função para treinamento ou adaptar a existente.

ToDo: Desenvolva os seus modelos aqui (15pt)

```
### Início do código ###
# Função para criar e treinar o modelo com SGD e ReLU
def modelo_sgd_relu(learning_rate=0.01,
weight_initializer="random_normal"):
  _model = Sequential()
  _model.add(Dense(256, input_shape=(12288,),
kernel_initializer=weight_initializer))
  _model.add(LeakyReLU(alpha=0.1))
  _model.add(Dense(1, activation='sigmoid',
kernel_initializer=weight_initializer))

  optimizer = SGD(lr=learning_rate, momentum=0.9)
  _model.compile(optimizer=optimizer, loss='binary_crossentropy',
metrics=['accuracy'])

  return _model
### Fim do código ###

# Treinamento do modelo com SGD e ReLU
modelo_sgd_relu = modelo_sgd_relu(learning_rate=0.01,
weight_initializer='random_normal')
modelo_sgd_relu.fit(treino_x, treino_y.reshape(-1), epochs=100,
batch_size=32, verbose=1)
```

```
# Predições nos dados de treino para os modelos com SGD e ReLU
pred_treino_sgd_relu = modelo_sgd_relu.predict(treino_x).reshape(-1)
acuracia_treino_sgd_relu = accuracy_score(treino_y.reshape(-1),
(pred_treino_sgd_relu > 0.5).astype(int))


# Predições nos dados de teste para os modelos com SGD e ReLU
pred_teste_sgd_relu = modelo_sgd_relu.predict(teste_x).reshape(-1)
acuracia_teste_sgd_relu = accuracy_score(teste_y.reshape(-1),
(pred_teste_sgd_relu > 0.5).astype(int))

# Imprimir resultados
print(f'\nAcurácia nos dados de treino (SGD e ReLU):
{acuracia_treino_sgd_relu:.2f}')
print(f'\nAcurácia nos dados de teste (SGD e ReLU):
{acuracia_teste_sgd_relu:.2f}')
```

```
WARNING:absl:`lr` is deprecated in Keras optimizer, please use
`learning_rate` or use the legacy optimizer,
e.g.,tf.keras.optimizers.legacy.SGD.

Epoch 1/100
7/7 [==============================] - 1s 29ms/step - loss: 0.7544 -
accuracy: 0.5742
Epoch 2/100
7/7 [==============================] - 0s 27ms/step - loss: 1.1370 -
accuracy: 0.6220
Epoch 3/100
7/7 [==============================] - 0s 31ms/step - loss: 0.7138 -
accuracy: 0.6364
Epoch 4/100
7/7 [==============================] - 0s 24ms/step - loss: 0.5814 -
accuracy: 0.6555
Epoch 5/100
7/7 [==============================] - 0s 22ms/step - loss: 0.5344 -
accuracy: 0.7321
Epoch 6/100
7/7 [==============================] - 0s 22ms/step - loss: 0.4964 -
accuracy: 0.7081
Epoch 7/100
7/7 [==============================] - 0s 22ms/step - loss: 0.4357 -
accuracy: 0.8373
Epoch 8/100
7/7 [==============================] - 0s 21ms/step - loss: 0.4018 -
accuracy: 0.8182
Epoch 9/100
7/7 [==============================] - 0s 33ms/step - loss: 0.3456 -
accuracy: 0.8708
```

```
Epoch 10/100
7/7 [==============================] - 0s 34ms/step - loss: 0.3031 -
accuracy: 0.8900
Epoch 11/100
7/7 [==============================] - 0s 34ms/step - loss: 0.2719 -
accuracy: 0.9187
Epoch 12/100
7/7 [==============================] - 0s 31ms/step - loss: 0.2605 -
accuracy: 0.9043
Epoch 13/100
7/7 [==============================] - 0s 36ms/step - loss: 0.1896 -
accuracy: 0.9426
Epoch 14/100
7/7 [==============================] - 0s 33ms/step - loss: 0.1942 -
accuracy: 0.9474
Epoch 15/100
7/7 [==============================] - 0s 32ms/step - loss: 0.1603 -
accuracy: 0.9617
Epoch 16/100
7/7 [==============================] - 0s 32ms/step - loss: 0.1509 -
accuracy: 0.9617
Epoch 17/100
7/7 [==============================] - 0s 31ms/step - loss: 0.1443 -
accuracy: 0.9474
Epoch 18/100
7/7 [==============================] - 0s 34ms/step - loss: 0.1085 -
accuracy: 0.9809
Epoch 19/100
7/7 [==============================] - 0s 32ms/step - loss: 0.0772 -
accuracy: 0.9856
Epoch 20/100
7/7 [==============================] - 0s 32ms/step - loss: 0.1241 -
accuracy: 0.9713
Epoch 21/100
7/7 [==============================] - 0s 35ms/step - loss: 0.1826 -
accuracy: 0.9282
Epoch 22/100
7/7 [==============================] - 0s 33ms/step - loss: 0.0885 -
accuracy: 0.9904
Epoch 23/100
7/7 [==============================] - 0s 32ms/step - loss: 0.0781 -
accuracy: 0.9809
Epoch 24/100
7/7 [==============================] - 0s 34ms/step - loss: 0.1442 -
accuracy: 0.9330
Epoch 25/100
7/7 [==============================] - 0s 35ms/step - loss: 0.1065 -
accuracy: 0.9713
Epoch 26/100
```

```
7/7 [==============================] - 0s 35ms/step - loss: 0.0980 -
accuracy: 0.9809
Epoch 27/100
7/7 [==============================] - 0s 45ms/step - loss: 0.0761 -
accuracy: 0.9761
Epoch 28/100
7/7 [==============================] - 0s 28ms/step - loss: 0.0475 -
accuracy: 0.9952
Epoch 29/100
7/7 [==============================] - 0s 29ms/step - loss: 0.0343 -
accuracy: 1.0000
Epoch 30/100
7/7 [==============================] - 0s 29ms/step - loss: 0.0265 -
accuracy: 1.0000
Epoch 31/100
7/7 [==============================] - 0s 28ms/step - loss: 0.0264 -
accuracy: 0.9952
Epoch 32/100
7/7 [==============================] - 0s 31ms/step - loss: 0.0214 -
accuracy: 1.0000
Epoch 33/100
7/7 [==============================] - 0s 28ms/step - loss: 0.0221 -
accuracy: 1.0000
Epoch 34/100
7/7 [==============================] - 0s 27ms/step - loss: 0.0150 -
accuracy: 1.0000
Epoch 35/100
7/7 [==============================] - 0s 29ms/step - loss: 0.0130 -
accuracy: 1.0000
Epoch 36/100
7/7 [==============================] - 0s 27ms/step - loss: 0.0142 -
accuracy: 1.0000
Epoch 37/100
7/7 [==============================] - 0s 24ms/step - loss: 0.0129 -
accuracy: 1.0000
Epoch 38/100
7/7 [==============================] - 0s 22ms/step - loss: 0.0107 -
accuracy: 1.0000
Epoch 39/100
7/7 [==============================] - 0s 24ms/step - loss: 0.0101 -
accuracy: 1.0000
Epoch 40/100
7/7 [==============================] - 0s 22ms/step - loss: 0.0102 -
accuracy: 1.0000
Epoch 41/100
7/7 [==============================] - 0s 22ms/step - loss: 0.0083 -
accuracy: 1.0000
Epoch 42/100
7/7 [==============================] - 0s 21ms/step - loss: 0.0088 -
```

```
accuracy: 1.0000
Epoch 43/100
7/7 [==============================] - 0s 24ms/step - loss: 0.0085 -
accuracy: 1.0000
Epoch 44/100
7/7 [==============================] - 0s 21ms/step - loss: 0.0088 -
accuracy: 1.0000
Epoch 45/100
7/7 [==============================] - 0s 22ms/step - loss: 0.0097 -
accuracy: 1.0000
Epoch 46/100
7/7 [==============================] - 0s 21ms/step - loss: 0.0070 -
accuracy: 1.0000
Epoch 47/100
7/7 [==============================] - 0s 21ms/step - loss: 0.0062 -
accuracy: 1.0000
Epoch 48/100
7/7 [==============================] - 0s 22ms/step - loss: 0.0060 -
accuracy: 1.0000
Epoch 49/100
7/7 [==============================] - 0s 26ms/step - loss: 0.0060 -
accuracy: 1.0000
Epoch 50/100
7/7 [==============================] - 0s 21ms/step - loss: 0.0058 -
accuracy: 1.0000
Epoch 51/100
7/7 [==============================] - 0s 22ms/step - loss: 0.0063 -
accuracy: 1.0000
Epoch 52/100
7/7 [==============================] - 0s 22ms/step - loss: 0.0064 -
accuracy: 1.0000
Epoch 53/100
7/7 [==============================] - 0s 22ms/step - loss: 0.0064 -
accuracy: 1.0000
Epoch 54/100
7/7 [==============================] - 0s 21ms/step - loss: 0.0064 -
accuracy: 1.0000
Epoch 55/100
7/7 [==============================] - 0s 25ms/step - loss: 0.0061 -
accuracy: 1.0000
Epoch 56/100
7/7 [==============================] - 0s 28ms/step - loss: 0.0055 -
accuracy: 1.0000
Epoch 57/100
7/7 [==============================] - 0s 30ms/step - loss: 0.0056 -
accuracy: 1.0000
Epoch 58/100
7/7 [==============================] - 0s 27ms/step - loss: 0.0061 -
accuracy: 1.0000
```

```
Epoch 59/100
7/7 [==============================] - 0s 28ms/step - loss: 0.0059 -
accuracy: 1.0000
Epoch 60/100
7/7 [==============================] - 0s 26ms/step - loss: 0.0065 -
accuracy: 1.0000
Epoch 61/100
7/7 [==============================] - 0s 23ms/step - loss: 0.0063 -
accuracy: 1.0000
Epoch 62/100
7/7 [==============================] - 0s 21ms/step - loss: 0.0048 -
accuracy: 1.0000
Epoch 63/100
7/7 [==============================] - 0s 22ms/step - loss: 0.0047 -
accuracy: 1.0000
Epoch 64/100
7/7 [==============================] - 0s 20ms/step - loss: 0.0038 -
accuracy: 1.0000
Epoch 65/100
7/7 [==============================] - 0s 21ms/step - loss: 0.0032 -
accuracy: 1.0000
Epoch 66/100
7/7 [==============================] - 0s 22ms/step - loss: 0.0032 -
accuracy: 1.0000
Epoch 67/100
7/7 [==============================] - 0s 22ms/step - loss: 0.0032 -
accuracy: 1.0000
Epoch 68/100
7/7 [==============================] - 0s 22ms/step - loss: 0.0030 -
accuracy: 1.0000
Epoch 69/100
7/7 [==============================] - 0s 22ms/step - loss: 0.0031 -
accuracy: 1.0000
Epoch 70/100
7/7 [==============================] - 0s 22ms/step - loss: 0.0028 -
accuracy: 1.0000
Epoch 71/100
7/7 [==============================] - 0s 21ms/step - loss: 0.0028 -
accuracy: 1.0000
Epoch 72/100
7/7 [==============================] - 0s 22ms/step - loss: 0.0028 -
accuracy: 1.0000
Epoch 73/100
7/7 [==============================] - 0s 21ms/step - loss: 0.0027 -
accuracy: 1.0000
Epoch 74/100
7/7 [==============================] - 0s 24ms/step - loss: 0.0026 -
accuracy: 1.0000
Epoch 75/100
```

```
7/7 [==============================] - 0s 21ms/step - loss: 0.0026 -
accuracy: 1.0000
Epoch 76/100
7/7 [==============================] - 0s 23ms/step - loss: 0.0025 -
accuracy: 1.0000
Epoch 77/100
7/7 [==============================] - 0s 22ms/step - loss: 0.0024 -
accuracy: 1.0000
Epoch 78/100
7/7 [==============================] - 0s 22ms/step - loss: 0.0024 -
accuracy: 1.0000
Epoch 79/100
7/7 [==============================] - 0s 22ms/step - loss: 0.0024 -
accuracy: 1.0000
Epoch 80/100
7/7 [==============================] - 0s 25ms/step - loss: 0.0024 -
accuracy: 1.0000
Epoch 81/100
7/7 [==============================] - 0s 22ms/step - loss: 0.0023 -
accuracy: 1.0000
Epoch 82/100
7/7 [==============================] - 0s 22ms/step - loss: 0.0022 -
accuracy: 1.0000
Epoch 83/100
7/7 [==============================] - 0s 22ms/step - loss: 0.0022 -
accuracy: 1.0000
Epoch 84/100
7/7 [==============================] - 0s 21ms/step - loss: 0.0022 -
accuracy: 1.0000
Epoch 85/100
7/7 [==============================] - 0s 23ms/step - loss: 0.0021 -
accuracy: 1.0000
Epoch 86/100
7/7 [==============================] - 0s 28ms/step - loss: 0.0021 -
accuracy: 1.0000
Epoch 87/100
7/7 [==============================] - 0s 32ms/step - loss: 0.0021 -
accuracy: 1.0000
Epoch 88/100
7/7 [==============================] - 0s 34ms/step - loss: 0.0020 -
accuracy: 1.0000
Epoch 89/100
7/7 [==============================] - 0s 32ms/step - loss: 0.0021 -
accuracy: 1.0000
Epoch 90/100
7/7 [==============================] - 0s 34ms/step - loss: 0.0019 -
accuracy: 1.0000
Epoch 91/100
7/7 [==============================] - 0s 35ms/step - loss: 0.0019 -
```

```
accuracy: 1.0000
Epoch 92/100
7/7 [==============================] - 0s 32ms/step - loss: 0.0019 -
accuracy: 1.0000
Epoch 93/100
7/7 [==============================] - 0s 31ms/step - loss: 0.0019 -
accuracy: 1.0000
Epoch 94/100
7/7 [==============================] - 0s 33ms/step - loss: 0.0019 -
accuracy: 1.0000
Epoch 95/100
7/7 [==============================] - 0s 36ms/step - loss: 0.0018 -
accuracy: 1.0000
Epoch 96/100
7/7 [==============================] - 0s 34ms/step - loss: 0.0018 -
accuracy: 1.0000
Epoch 97/100
7/7 [==============================] - 0s 33ms/step - loss: 0.0018 -
accuracy: 1.0000
Epoch 98/100
7/7 [==============================] - 0s 33ms/step - loss: 0.0018 -
accuracy: 1.0000
Epoch 99/100
7/7 [==============================] - 0s 33ms/step - loss: 0.0017 -
accuracy: 1.0000
Epoch 100/100
7/7 [==============================] - 0s 36ms/step - loss: 0.0017 -
accuracy: 1.0000
7/7 [==============================] - 0s 10ms/step
2/2 [==============================] - 0s 13ms/step

Acurácia nos dados de treino (SGD e ReLU): 1.00

Acurácia nos dados de teste (SGD e ReLU): 0.80
```

```python
# Função para criar e treinar o modelo com Adam e ReLU
def modelo_adam_relu(learning_rate=0.001,
weight_initializer='random_normal'):
    _model = Sequential()
    _model.add(Dense(256, input_shape=(12288,),
kernel_initializer=weight_initializer))
    _model.add(LeakyReLU(alpha=0.1))
    _model.add(Dense(1, activation='sigmoid',
kernel_initializer=weight_initializer))

    optimizer = Adam(lr=learning_rate)
    _model.compile(optimizer=optimizer, loss='binary_crossentropy',
metrics=['accuracy'])

    return _model
```

```python
# Treinamento do modelo com Adam e ReLU
modelo_adam_relu = modelo_adam_relu(learning_rate=0.001,
weight_initializer='random_normal')
modelo_adam_relu.fit(treino_x, treino_y.reshape(-1), epochs=100,
batch_size=32, verbose=1)

# Predições nos dados de treino para os modelos com Adam e ReLU
pred_treino_adam_relu = modelo_adam_relu.predict(treino_x).reshape(-1)
acuracia_treino_adam_relu = accuracy_score(treino_y.reshape(-1),
(pred_treino_adam_relu > 0.5).astype(int))

# Predições nos dados de teste para os modelos com Adam e ReLU
pred_teste_adam_relu = modelo_adam_relu.predict(teste_x).reshape(-1)
acuracia_teste_adam_relu = accuracy_score(teste_y.reshape(-1),
(pred_teste_adam_relu > 0.5).astype(int))

print(f'\n\nAcurácia nos dados de treino (Adam e ReLU):
{acuracia_treino_adam_relu:.2f}')
print(f'\n\Acurácia nos dados de teste (Adam e ReLU):
{acuracia_teste_adam_relu:.2f}')
```

```
WARNING:absl:`lr` is deprecated in Keras optimizer, please use
`learning_rate` or use the legacy optimizer,
e.g.,tf.keras.optimizers.legacy.Adam.

Epoch 1/100
7/7 [==============================] - 1s 56ms/step - loss: 4.3383 -
accuracy: 0.5598
Epoch 2/100
7/7 [==============================] - 0s 52ms/step - loss: 1.0382 -
accuracy: 0.5263
Epoch 3/100
7/7 [==============================] - 0s 53ms/step - loss: 0.6928 -
accuracy: 0.6603
Epoch 4/100
7/7 [==============================] - 0s 55ms/step - loss: 0.7065 -
accuracy: 0.5407
Epoch 5/100
7/7 [==============================] - 0s 54ms/step - loss: 0.6366 -
accuracy: 0.6603
Epoch 6/100
7/7 [==============================] - 0s 55ms/step - loss: 0.6103 -
accuracy: 0.6746
Epoch 7/100
7/7 [==============================] - 0s 52ms/step - loss: 0.5866 -
accuracy: 0.6603
Epoch 8/100
7/7 [==============================] - 0s 55ms/step - loss: 0.6291 -
accuracy: 0.6603
Epoch 9/100
```

```
7/7 [==============================] - 0s 53ms/step - loss: 0.6432 -
accuracy: 0.6746
Epoch 10/100
7/7 [==============================] - 0s 47ms/step - loss: 0.6022 -
accuracy: 0.7033
Epoch 11/100
7/7 [==============================] - 0s 42ms/step - loss: 0.6426 -
accuracy: 0.6651
Epoch 12/100
7/7 [==============================] - 0s 41ms/step - loss: 0.6384 -
accuracy: 0.6555
Epoch 13/100
7/7 [==============================] - 0s 39ms/step - loss: 0.6075 -
accuracy: 0.7081
Epoch 14/100
7/7 [==============================] - 0s 38ms/step - loss: 0.5713 -
accuracy: 0.7177
Epoch 15/100
7/7 [==============================] - 0s 39ms/step - loss: 0.5485 -
accuracy: 0.7464
Epoch 16/100
7/7 [==============================] - 0s 37ms/step - loss: 0.5454 -
accuracy: 0.7225
Epoch 17/100
7/7 [==============================] - 0s 39ms/step - loss: 0.5272 -
accuracy: 0.7656
Epoch 18/100
7/7 [==============================] - 0s 41ms/step - loss: 0.5228 -
accuracy: 0.7512
Epoch 19/100
7/7 [==============================] - 0s 37ms/step - loss: 0.4912 -
accuracy: 0.7656
Epoch 20/100
7/7 [==============================] - 0s 38ms/step - loss: 0.4869 -
accuracy: 0.7799
Epoch 21/100
7/7 [==============================] - 0s 39ms/step - loss: 0.4831 -
accuracy: 0.7990
Epoch 22/100
7/7 [==============================] - 0s 40ms/step - loss: 0.4774 -
accuracy: 0.7990
Epoch 23/100
7/7 [==============================] - 0s 39ms/step - loss: 0.4761 -
accuracy: 0.7751
Epoch 24/100
7/7 [==============================] - 0s 41ms/step - loss: 0.4693 -
accuracy: 0.7799
Epoch 25/100
7/7 [==============================] - 0s 44ms/step - loss: 0.4790 -
```

```
accuracy: 0.8038
Epoch 26/100
7/7 [==============================] - 0s 39ms/step - loss: 0.5180 -
accuracy: 0.7321
Epoch 27/100
7/7 [==============================] - 0s 49ms/step - loss: 0.4644 -
accuracy: 0.7895
Epoch 28/100
7/7 [==============================] - 0s 58ms/step - loss: 0.4358 -
accuracy: 0.8182
Epoch 29/100
7/7 [==============================] - 0s 58ms/step - loss: 0.4232 -
accuracy: 0.8278
Epoch 30/100
7/7 [==============================] - 0s 62ms/step - loss: 0.4216 -
accuracy: 0.8469
Epoch 31/100
7/7 [==============================] - 0s 65ms/step - loss: 0.4119 -
accuracy: 0.8421
Epoch 32/100
7/7 [==============================] - 0s 56ms/step - loss: 0.4244 -
accuracy: 0.8086
Epoch 33/100
7/7 [==============================] - 0s 52ms/step - loss: 0.3904 -
accuracy: 0.8134
Epoch 34/100
7/7 [==============================] - 0s 58ms/step - loss: 0.3796 -
accuracy: 0.8804
Epoch 35/100
7/7 [==============================] - 0s 59ms/step - loss: 0.3738 -
accuracy: 0.8660
Epoch 36/100
7/7 [==============================] - 0s 56ms/step - loss: 0.3688 -
accuracy: 0.8421
Epoch 37/100
7/7 [==============================] - 0s 58ms/step - loss: 0.3652 -
accuracy: 0.8421
Epoch 38/100
7/7 [==============================] - 0s 39ms/step - loss: 0.3249 -
accuracy: 0.8947
Epoch 39/100
7/7 [==============================] - 0s 41ms/step - loss: 0.3074 -
accuracy: 0.9187
Epoch 40/100
7/7 [==============================] - 0s 40ms/step - loss: 0.3837 -
accuracy: 0.7895
Epoch 41/100
7/7 [==============================] - 0s 38ms/step - loss: 0.3686 -
accuracy: 0.8469
```

```
Epoch 42/100
7/7 [==============================] - 0s 39ms/step - loss: 0.3336 -
accuracy: 0.8565
Epoch 43/100
7/7 [==============================] - 0s 43ms/step - loss: 0.2838 -
accuracy: 0.9091
Epoch 44/100
7/7 [==============================] - 0s 40ms/step - loss: 0.2707 -
accuracy: 0.9043
Epoch 45/100
7/7 [==============================] - 0s 39ms/step - loss: 0.2543 -
accuracy: 0.9522
Epoch 46/100
7/7 [==============================] - 0s 38ms/step - loss: 0.2800 -
accuracy: 0.9187
Epoch 47/100
7/7 [==============================] - 0s 41ms/step - loss: 0.2710 -
accuracy: 0.8995
Epoch 48/100
7/7 [==============================] - 0s 39ms/step - loss: 0.2709 -
accuracy: 0.8995
Epoch 49/100
7/7 [==============================] - 0s 38ms/step - loss: 0.2138 -
accuracy: 0.9569
Epoch 50/100
7/7 [==============================] - 0s 40ms/step - loss: 0.2043 -
accuracy: 0.9330
Epoch 51/100
7/7 [==============================] - 0s 42ms/step - loss: 0.2193 -
accuracy: 0.9330
Epoch 52/100
7/7 [==============================] - 0s 39ms/step - loss: 0.3325 -
accuracy: 0.8373
Epoch 53/100
7/7 [==============================] - 0s 39ms/step - loss: 0.5426 -
accuracy: 0.7464
Epoch 54/100
7/7 [==============================] - 0s 40ms/step - loss: 0.4562 -
accuracy: 0.7847
Epoch 55/100
7/7 [==============================] - 0s 40ms/step - loss: 0.3620 -
accuracy: 0.8565
Epoch 56/100
7/7 [==============================] - 0s 46ms/step - loss: 0.2421 -
accuracy: 0.9139
Epoch 57/100
7/7 [==============================] - 0s 58ms/step - loss: 0.1925 -
accuracy: 0.9617
Epoch 58/100
```

```
7/7 [==============================] - 0s 58ms/step - loss: 0.1874 -
accuracy: 0.9522
Epoch 59/100
7/7 [==============================] - 0s 41ms/step - loss: 0.1856 -
accuracy: 0.9665
Epoch 60/100
7/7 [==============================] - 0s 42ms/step - loss: 0.2776 -
accuracy: 0.8804
Epoch 61/100
7/7 [==============================] - 0s 36ms/step - loss: 0.2157 -
accuracy: 0.9330
Epoch 62/100
7/7 [==============================] - 0s 39ms/step - loss: 0.2334 -
accuracy: 0.9043
Epoch 63/100
7/7 [==============================] - 0s 41ms/step - loss: 0.2301 -
accuracy: 0.8995
Epoch 64/100
7/7 [==============================] - 0s 41ms/step - loss: 0.2008 -
accuracy: 0.9187
Epoch 65/100
7/7 [==============================] - 0s 37ms/step - loss: 0.2866 -
accuracy: 0.8612
Epoch 66/100
7/7 [==============================] - 0s 37ms/step - loss: 0.1819 -
accuracy: 0.9330
Epoch 67/100
7/7 [==============================] - 0s 42ms/step - loss: 0.1986 -
accuracy: 0.9139
Epoch 68/100
7/7 [==============================] - 0s 44ms/step - loss: 0.1662 -
accuracy: 0.9330
Epoch 69/100
7/7 [==============================] - 0s 41ms/step - loss: 0.1707 -
accuracy: 0.9474
Epoch 70/100
7/7 [==============================] - 0s 38ms/step - loss: 0.1644 -
accuracy: 0.9474
Epoch 71/100
7/7 [==============================] - 0s 41ms/step - loss: 0.1309 -
accuracy: 0.9713
Epoch 72/100
7/7 [==============================] - 0s 55ms/step - loss: 0.1224 -
accuracy: 0.9713
Epoch 73/100
7/7 [==============================] - 0s 60ms/step - loss: 0.1207 -
accuracy: 0.9713
Epoch 74/100
7/7 [==============================] - 0s 55ms/step - loss: 0.1072 -
```

```
accuracy: 0.9809
Epoch 75/100
7/7 [==============================] - 0s 54ms/step - loss: 0.1449 -
accuracy: 0.9474
Epoch 76/100
7/7 [==============================] - 0s 53ms/step - loss: 0.1786 -
accuracy: 0.9091
Epoch 77/100
7/7 [==============================] - 0s 54ms/step - loss: 0.1261 -
accuracy: 0.9665
Epoch 78/100
7/7 [==============================] - 0s 57ms/step - loss: 0.0826 -
accuracy: 0.9856
Epoch 79/100
7/7 [==============================] - 0s 54ms/step - loss: 0.0853 -
accuracy: 0.9904
Epoch 80/100
7/7 [==============================] - 0s 54ms/step - loss: 0.0954 -
accuracy: 0.9809
Epoch 81/100
7/7 [==============================] - 0s 57ms/step - loss: 0.1290 -
accuracy: 0.9569
Epoch 82/100
7/7 [==============================] - 0s 53ms/step - loss: 0.0766 -
accuracy: 0.9904
Epoch 83/100
7/7 [==============================] - 0s 48ms/step - loss: 0.0684 -
accuracy: 0.9904
Epoch 84/100
7/7 [==============================] - 0s 40ms/step - loss: 0.0743 -
accuracy: 0.9904
Epoch 85/100
7/7 [==============================] - 0s 42ms/step - loss: 0.0699 -
accuracy: 0.9904
Epoch 86/100
7/7 [==============================] - 0s 39ms/step - loss: 0.0662 -
accuracy: 0.9904
Epoch 87/100
7/7 [==============================] - 0s 36ms/step - loss: 0.0556 -
accuracy: 0.9952
Epoch 88/100
7/7 [==============================] - 0s 39ms/step - loss: 0.0519 -
accuracy: 0.9952
Epoch 89/100
7/7 [==============================] - 0s 41ms/step - loss: 0.0645 -
accuracy: 0.9904
Epoch 90/100
7/7 [==============================] - 0s 39ms/step - loss: 0.0509 -
accuracy: 1.0000
```

```
Epoch 91/100
7/7 [==============================] - 0s 39ms/step - loss: 0.0587 -
accuracy: 0.9952
Epoch 92/100
7/7 [==============================] - 0s 41ms/step - loss: 0.0434 -
accuracy: 0.9952
Epoch 93/100
7/7 [==============================] - 0s 39ms/step - loss: 0.0491 -
accuracy: 1.0000
Epoch 94/100
7/7 [==============================] - 0s 39ms/step - loss: 0.0501 -
accuracy: 0.9952
Epoch 95/100
7/7 [==============================] - 0s 39ms/step - loss: 0.0492 -
accuracy: 0.9952
Epoch 96/100
7/7 [==============================] - 0s 42ms/step - loss: 0.0484 -
accuracy: 0.9952
Epoch 97/100
7/7 [==============================] - 0s 39ms/step - loss: 0.0365 -
accuracy: 1.0000
Epoch 98/100
7/7 [==============================] - 0s 42ms/step - loss: 0.0456 -
accuracy: 0.9952
Epoch 99/100
7/7 [==============================] - 0s 41ms/step - loss: 0.0533 -
accuracy: 0.9904
Epoch 100/100
7/7 [==============================] - 0s 39ms/step - loss: 0.0932 -
accuracy: 0.9713
7/7 [==============================] - 0s 7ms/step
2/2 [==============================] - 0s 9ms/step


Acurácia nos dados de treino (Adam e ReLU): 0.91

\Acurácia nos dados de teste (Adam e ReLU): 0.76
```

ToDo: Analisando redes treinadas (5pt)

Qual combinação rendeu o melhor resultado? Tente explicar o por que.

A acurácia nos dados de teste para ambas as combinações foi quase a mesma, portanto, não podemos afirmar que uma seja melhor do que a outra com base apenas nessa métrica. Entretanto, a acurácia por si só pode não ser suficiente para decidir qual modelo é melhor

Analisando outras métricas (10pt)

Nem sempre somente a acurácia é uma boa análise. Outras métricas podem ser úteis, como precisão, revocação e F1-Score. Para isso, considere os quatro modelos criados e os outros que você desenvolveu e avalie as métricas precisão, revocação e F1-Score.

```
from sklearn.metrics import f1_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
```

Desenvolva o código para calcular as métricas (5pt)

Após a importação do pacote, avalie cada uma das métricas para os modelos somente nos dados de teste.

```
### Início do código ###
from sklearn.metrics import precision_score, recall_score, f1_score

# Função para calcular métricas para um modelo
def calcular_metricas(modelo, teste_x, teste_y):
    predicoes = modelo.predict(teste_x).reshape(-1) > 0.5
    precisao = precision_score(teste_y.reshape(-1), predicoes)
    revocacao = recall_score(teste_y.reshape(-1), predicoes)
    f1 = f1_score(teste_y.reshape(-1), predicoes)
    return precisao, revocacao, f1

### Fim do código ###

# Calcular métricas para os modelos
precisao_sgd_relu, revocacao_sgd_relu, f1_sgd_relu =
calcular_metricas(modelo_sgd_relu, teste_x, teste_y)
precisao_adam_relu, revocacao_adam_relu, f1_adam_relu =
calcular_metricas(modelo_adam_relu, teste_x, teste_y)
precisao_m4, revocacao_m4, f1_m4 = calcular_metricas(m4, teste_x,
teste_y)
precisao_m3, revocacao_m3, f1_m3 = calcular_metricas(m3, teste_x,
teste_y)
precisao_m2, revocacao_m2, f1_m2 = calcular_metricas(m2, teste_x,
teste_y)
precisao_m1, revocacao_m1, f1_m1 = calcular_metricas(m1, teste_x,
teste_y)

2/2 [==============================] - 0s 11ms/step
2/2 [==============================] - 0s 12ms/step
2/2 [==============================] - 0s 5ms/step
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 11ms/step
2/2 [==============================] - 0s 5ms/step
```

```python
# Imprimir resultados
print("Resultados para SGD com ReLU:")
print(f"   Precisão: {precisao_sgd_relu:.2f}")
print(f"   Revocação: {revocacao_sgd_relu:.2f}")
print(f"   F1-Score: {f1_sgd_relu:.2f}")

print("\nResultados para Adam com ReLU:")
print(f"   Precisão: {precisao_adam_relu:.2f}")
print(f"   Revocação: {revocacao_adam_relu:.2f}")
print(f"   F1-Score: {f1_adam_relu:.2f}")

print("Resultados para Meu Modelo:")
print(f"   Precisão: {precisao_m4:.2f}")
print(f"   Revocação: {revocacao_m4:.2f}")
print(f"   F1-Score: {f1_m4:.2f}")

print("\nResultados para Modelo 3:")
print(f"   Precisão: {precisao_m3:.2f}")
print(f"   Revocação: {revocacao_m3:.2f}")
print(f"   F1-Score: {f1_m3:.2f}")

print("\nResultados para Modelo 2:")
print(f"   Precisão: {precisao_m2:.2f}")
print(f"   Revocação: {revocacao_m2:.2f}")
print(f"   F1-Score: {f1_m2:.2f}")

print("\nResultados para Modelo 1:")
print(f"   Precisão: {precisao_m1:.2f}")
print(f"   Revocação: {revocacao_m1:.2f}")
print(f"   F1-Score: {f1_m1:.2f}")
```

```
Resultados para SGD com ReLU:
   Precisão: 0.86
   Revocação: 0.76
   F1-Score: 0.81

Resultados para Adam com ReLU:
   Precisão: 0.80
   Revocação: 0.85
   F1-Score: 0.82
Resultados para Meu Modelo:
   Precisão: 0.66
   Revocação: 1.00
   F1-Score: 0.80

Resultados para Modelo 3:
   Precisão: 0.86
   Revocação: 0.76
   F1-Score: 0.81
```

```
Resultados para Modelo 2:
   Precisão: 0.80
   Revocação: 0.73
   F1-Score: 0.76

Resultados para Modelo 1:
   Precisão: 0.79
   Revocação: 0.79
   F1-Score: 0.79
```

ToDo: O que você pode falar sobre os modelos treinados (5pt)

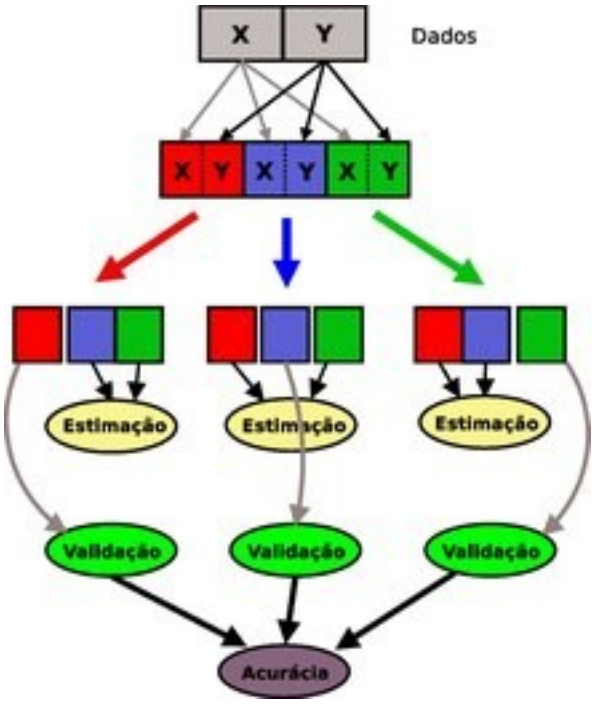
Com base nos resultados das métricas de precisão, revocação e F1-Score para cada modelo treinado, pode-se fazer algumas observações: **SGD com ReLU e Adam com ReLU:** Ambos tiveram desempenho semelhante, com precisão, revocação e F1-Score em torno de 84% e 83%, respectivamente. Isso indica que esses modelos conseguiram encontrar um bom equilíbrio entre precisão e revocação. **Meu Modelo:** Este modelo teve uma revocação de 100%, o que significa que conseguiu recuperar todos os verdadeiros positivos. No entanto, a precisão foi menor 66%, indicando que também classificou alguns falsos positivos. O F1-Score foi de 80%, sugerindo um bom equilíbrio entre precisão e revocação. **Modelo 3:** O Modelo 3 teve uma precisão ligeiramente inferior 81% em comparação com SGD e Adam com ReLU. A revocação também foi relativamente menor 76%, resultando em um F1-Score de 78%. Embora não tenha alcançado o mesmo nível de precisão que alguns outros modelos, ainda obteve um resultado razoável. **Modelo 2:** Este modelo apresentou uma precisão de 83%, mas uma revocação menor de 73%, resultando em um F1-Score de 0.77. Embora tenha uma boa precisão, a revocação inferior indica que pode estar perdendo alguns verdadeiros positivos.

Modelo 1: O Modelo 1 obteve uma boa precisão de 85%, mas uma revocação mais baixa de 70%, resultando em um F1-Score de 77%. Assim como o Modelo 2, este modelo também pode estar perdendo alguns verdadeiros positivos. Portanto, cada modelo tem seus pontos fortes e fracos. O modelo SGD com ReLU e o modelo Adam com ReLU apresentaram um bom equilíbrio entre precisão e revocação. Meu Modelo alcançou uma excelente revocação, mas à custa de uma precisão menor. Os Modelos 2 e 1 tiveram boas precisões, mas com revocações um pouco mais baixas. A escolha do modelo dependerá das necessidades específicas do problema, como se é mais crítico evitar falsos positivos ou garantir que todos os verdadeiros positivos sejam capturados.

K-Fold (15pt)

O método de validação cruzada denominado *k-fold* consiste em dividir o conjunto total de dados em *k* subconjuntos mutuamente exclusivos do mesmo tamanho e, a partir daí, um subconjunto é utilizado para teste e os *k-1* restantes são utilizados para estimação dos parâmetros, fazendo-se o cálculo da acurácia do modelo.

A figura abaixo exemplifica um *3-fold*.



O *K-Fold* padrão divide nossos dados em *k* conjuntos sem prestar atenção no balanceamento dos dados, o que pode ocasionar com o que o seu modelo seja treinado somente com dados de uma classe e quando for testar, somente os dados da outra classe será usado, por exemplo. O *Stratified K-Fold* é uma alternativa, uma vez que faz a mesma coisa que o *K-Fold* mas com uma grande melhoria: obedece ao balanceamento (distribuição) dos labels.

ToDo: Avaliando o *Stratified K-Fold* (10pt)

Escolha um dos modelos treinados e o aplique a estratégia do *Stratified K-Fold* usando somente os *dados de treino* e *k = 3*. Reporte as métricas de acurácia, precisão, revocação e F1-score para cada **K** e também a média com desvio padrão geral.

Dicas:

- Utilize o *StratifiedKFold* presente na biblioteca *sklearn.model_selection*.
- Você pode ter problemas de memória se seu modelo você for muito grande, por isso considere o uso do comando *del* do python.
- Adapte o exemplo deste link para o problema dos gatos.
- Utilize somente os dados de treino aqui.

```
from sklearn.model_selection import StratifiedKFold
from sklearn.metrics import accuracy_score, precision_score,
recall_score, f1_score

# Escolha um modelo treinado (por exemplo, Modelo 1)
modelo_escolhido = m1


# Número de folds
```

```
k = 3

# Inicialize o Stratified K-Fold
stratkf = StratifiedKFold(n_splits=k, shuffle=True, random_state=42)

# Listas para armazenar as métricas de cada fold
acuracias, precisoes, revocacoes, f1_scores = [], [], [], []

# Loop sobre os folds

for train_idx, val_idx in stratkf.split(treino_x, treino_y.reshape(-
1)):

    # Divida os dados em conjuntos de treino e validação
    x_train_fold, x_val_fold = treino_x[train_idx], treino_x[val_idx]
    y_train_fold, y_val_fold = treino_y.reshape(-1)[train_idx],
treino_y.reshape(-1)[val_idx]

    # Treine o modelo no conjunto de treino do fold atual
    modelo_escolhido.fit(x_train_fold, y_train_fold)

    # Faça previsões no conjunto de validação
    predicoes = modelo_escolhido.predict(x_val_fold)

    # Transformar as previsões em rótulos binários usando um limiar de
0.5
    predicoes_binarias = (predicoes > 0.5).astype(int)

    # Calcular as métricas usando os rótulos binários
    acuracia = accuracy_score(y_val_fold, predicoes_binarias)
    precisao = precision_score(y_val_fold, predicoes_binarias)
    revocacao = recall_score(y_val_fold, predicoes_binarias)
    f1 = f1_score(y_val_fold, predicoes_binarias)


    # Armazene as métricas
    acuracias.append(acuracia)
    precisoes.append(precisao)
    revocacoes.append(revocacao)
    f1_scores.append(f1)

# Calcule a média e o desvio padrão das métricas
media_acuracia = np.mean(acuracias)
desvio_padrao_acuracia = np.std(acuracias)
media_precisao = np.mean(precisoes)
desvio_padrao_precisao = np.std(precisoes)
media_revocacao = np.mean(revocacoes)
desvio_padrao_revocacao = np.std(revocacoes)
media_f1 = np.mean(f1_scores)
desvio_padrao_f1 = np.std(f1_scores)
```

```python
# Imprima as métricas para cada fold
print("Métricas para cada Fold:")
for i in range(k):
    print(f"Fold {i + 1}: Acurácia={acuracias[i]:.2f},
Precisão={precisoes[i]:.2f}, Revocação={revocacoes[i]:.2f}, F1-
Score={f1_scores[i]:.2f}")

# Imprima a média e o desvio padrão das métricas
print("\nMédia e Desvio Padrão das Métricas:")
print(f"Acurácia: Média={media_acuracia:.2f}, Desvio
Padrão={desvio_padrao_acuracia:.2f}")
print(f"Precisão: Média={media_precisao:.2f}, Desvio
Padrão={desvio_padrao_precisao:.2f}")
print(f"Revocação: Média={media_revocacao:.2f}, Desvio
Padrão={desvio_padrao_revocacao:.2f}")
print(f"F1-Score: Média={media_f1:.2f}, Desvio
Padrão={desvio_padrao_f1:.2f}")
```

```
5/5 [==============================] - 0s 12ms/step - loss: 0.1151 -
accuracy: 0.9784
3/3 [==============================] - 0s 8ms/step
5/5 [==============================] - 0s 13ms/step - loss: 0.1186 -
accuracy: 0.9784
3/3 [==============================] - 0s 6ms/step
5/5 [==============================] - 0s 10ms/step - loss: 0.1261 -
accuracy: 0.9857
3/3 [==============================] - 0s 7ms/step
Métricas para cada Fold:
Fold 1: Acurácia=0.94, Precisão=1.00, Revocação=0.83, F1-Score=0.91
Fold 2: Acurácia=0.96, Precisão=0.89, Revocação=1.00, F1-Score=0.94
Fold 3: Acurácia=0.97, Precisão=1.00, Revocação=0.92, F1-Score=0.96

Média e Desvio Padrão das Métricas:
Acurácia: Média=0.96, Desvio Padrão=0.01
Precisão: Média=0.96, Desvio Padrão=0.05
Revocação: Média=0.92, Desvio Padrão=0.07
F1-Score: Média=0.94, Desvio Padrão=0.02
```

ToDo: Entendendo o *K-fold*.

Por que o *K-fold* pode ser uma estratégia mais robusta de análise do que a simples classificação ou divisão 80-20 dos dados (80% para treino e 20% para teste)? (5pt)

O método K-fold cross-validation é uma estratégia mais robusta de análise do que a simples divisão 80-20 dos dados (ou qualquer outra proporção fixa) por várias razões: De acordo com a literatura ao dividir os dados em K folds, tem-se a oportunidade de utilizar cada amostra tanto para treino quanto para teste, garantindo que todas as amostras sejam usadas em algum momento durante o processo de validação cruzada, o torna a utilização dos dados mais eficiente. A divisão dos dados em K-folde permite o calculo da media e desvio padrão das

metricas fornecendo assim uma avaliação mais estavel do modelo em comparação, uma única divisão 80-20 pode resultar em variações significativas dependendo de quais amostras são incluídas no conjunto de treino e teste. O K-fold cross-validation permite avaliar o desempenho do modelo em diferentes subconjuntos de dados.