


```
!pip install tensorflow
!pip install scikit-learn
!pip install pandas
```



```
Requirement already satisfied: tensorflow in /usr/local/lib/python3.10/dist-packages (2.17.0)
Requirement already satisfied: absl-py>=1.0.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.4.0)
Requirement already satisfied: astunparse>=1.6.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.6.3)
Requirement already satisfied: flatbuffers>=24.3.25 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (24.3.25)
Requirement already satisfied: gast!=0.5.0,!0.5.1,!0.5.2,>=0.2.1 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (0.6.0)
Requirement already satisfied: google-pasta>=0.1.1 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (0.2.0)
Requirement already satisfied: h5py>=3.10.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (3.11.0)
Requirement already satisfied: libclang>=13.0.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (18.1.1)
Requirement already satisfied: ml-dtypes<0.5.0,>=0.3.1 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (0.4.1)
Requirement already satisfied: opt-einsum>=2.3.2 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (3.4.0)
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-packages (from tensorflow) (24.1)
Requirement already satisfied: protobuf!=4.21.0,!4.21.1,!4.21.2,!4.21.3,!4.21.4,!4.21.5,<5.0.0dev,>=3.20.3 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (2.32.3)
Requirement already satisfied: requests<3,>=2.21.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (2.32.3)
Requirement already satisfied: setuptools in /usr/local/lib/python3.10/dist-packages (from tensorflow) (75.1.0)
Requirement already satisfied: six>=1.12.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.16.0)
Requirement already satisfied: termcolor>=1.1.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (2.5.0)
Requirement already satisfied: typing-extensions>=3.6.6 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (4.12.2)
Requirement already satisfied: wrapt>=1.11.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.16.0)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.64.1)
Requirement already satisfied: tensorboard<2.18,>=2.17 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (2.17.0)
Requirement already satisfied: keras>=3.2.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (3.4.1)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (0.37.1)
Requirement already satisfied: numpy<2.0.0,>=1.23.5 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.26.4)
Requirement already satisfied: wheel<1.0,>=0.23.0 in /usr/local/lib/python3.10/dist-packages (from astunparse>=1.6.0->tensorflow) (0.44.0)
Requirement already satisfied: rich in /usr/local/lib/python3.10/dist-packages (from keras>=3.2.0->tensorflow) (13.9.2)
Requirement already satisfied: namex in /usr/local/lib/python3.10/dist-packages (from keras>=3.2.0->tensorflow) (0.0.8)
Requirement already satisfied: optree in /usr/local/lib/python3.10/dist-packages (from keras>=3.2.0->tensorflow) (0.13.0)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0->tensorflow) (3.4.0)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0->tensorflow) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0->tensorflow) (2.2.3)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0->tensorflow) (2024.8.30)
Requirement already satisfied: markdown>=2.6.8 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.18,>=2.17->tensorflow) (3.7)
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.18,>=2.17->tensorflow) (0.17.0)
Requirement already satisfied: werkzeug>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.18,>=2.17->tensorflow) (3.0.4)
Requirement already satisfied: MarkupSafe>=2.1.1 in /usr/local/lib/python3.10/dist-packages (from werkzeug>=1.0.1->tensorboard<2.18,>=2.17->tensorflow) (2.1.1)
Requirement already satisfied: markdown-it-py>=2.2.0 in /usr/local/lib/python3.10/dist-packages (from rich->keras>=3.2.0->tensorflow) (3.0.0)
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in /usr/local/lib/python3.10/dist-packages (from rich->keras>=3.2.0->tensorflow) (2.18.0)
Requirement already satisfied: mdurl~=0.1 in /usr/local/lib/python3.10/dist-packages (from markdown-it-py>=2.2.0->rich->keras>=3.2.0->tensorflow) (0.1.2)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.10/dist-packages (1.5.2)
Requirement already satisfied: numpy>=1.19.5 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.26.4)
Requirement already satisfied: scipy>=1.6.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.13.1)
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.4.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (3.5.0)
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packages (2.2.2)
Requirement already satisfied: numpy>=1.22.4 in /usr/local/lib/python3.10/dist-packages (from pandas) (1.26.4)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.10/dist-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas) (2024.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.10/dist-packages (from pandas) (2024.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.8.2->pandas) (1.16.0)
```

✓ Curso: Redes Neurais e Deep Learning

Prof. Denilson Alves Pereira <https://sites.google.com/ufla.br/denilsonpereira/> Departamento de Ciência da Computação - Instituto de Ciências Exatas e Tecnológicas - Universidade Federal de Lavras

Atividade Prática 01

Instruções:

1. Siga os passos indicados em cada célula abaixo para completar a atividade.
2. Você deve inserir código somente entre as linhas marcadas com **INICIE O CÓDIGO AQUI** e **TERMINE O CÓDIGO AQUI**. Há uma indicação de quantas linhas de código são necessárias.
3. Em alguns pontos, confira o resultado esperado conforme marcado com **SAÍDA ESPERADA**.

Tempo estimado para execução: 1 hora

Versão: Junho, 2021

O Problema a ser Resolvido

O objetivo da atividade é elaborar uma rede neural para predizer se um paciente tem ou não diabetes, com base nas medidas diagnósticas contidas no *dataset* disponível em <https://www.kaggle.com/uciml/pima-indians-diabetes-database>.

Os dados são de pacientes do sexo feminino, com pelo menos 20 anos de idade. Os atributos das condições médicas incluem o número de gestações que a paciente teve, seu IMC, nível de insulina, idade e outros. A classe a ser predita é o atributo "Outcome", cujos valores são 0 (não tem diabetes) ou 1 (tem diabetes). Portanto, é um problema de classificação binária.

Você vai praticar as seguintes habilidades:

- Efetuar o pré-processamento dos dados, separando-os em conjuntos de treino e teste.
- Configurar uma rede neural simples para um problema de classificação binária.

✓ Pacotes

```
import numpy as np # package for scientific computing
import tensorflow as tf # package for numerical computation using data flow graphs
from tensorflow import keras # package for deep learning
import pandas as pd # package for working with structured data
```

Pré-Processamentos dos Dados de Treino e de Teste

```
# Read dataset
data = pd.read_csv('diabetes.csv')
data.head() # display dataset first lines
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

```
# Separate the class from other attributes
X = data.drop("Outcome", axis=1)
Y = data["Outcome"]
```

Documentação de *train_test_split*: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html

A função divide os dados em partições de treino e teste, de acordo com a proporção especificada pelo parâmetro *test_size*.

O parâmetro *random_state* é usado para deixar os resultados reproduzíveis para fins de avaliação do exercício.

```
# Preparing the dataset for training and test
from sklearn.model_selection import train_test_split
train_set_X, test_set_X, train_set_Y, test_set_Y = train_test_split(X, Y, test_size=0.20, random_state=7)
```

Padronize os atributos usando a média e a variância dos dados

Dica: use a função *fit_transform*: <https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html>

```
# Standardize features by removing the mean and scaling to unit variance
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
```

```
### INICIE O CÓDIGO AQUI ### (2 linhas de código)
train_set_X = scaler.fit_transform(train_set_X)
test_set_X = scaler.fit_transform(test_set_X)
### TERMINE O CÓDIGO AQUI ###
```

```
# Checking
print("train_set_X:\n", train_set_X[:2,:])
print("\ntest_set_X:\n", test_set_X[:2,:])
```

```
train_set_X:
[[ 0.35483802 -0.370418    0.16624635  1.40032403 -0.01279543  0.49863797
  -0.61786951  0.01064085]
 [-0.54794048 -0.55620696  0.9024924   0.96548604  0.40069886  1.70739891
  -1.0345765  -0.86049025]]

test_set_X:
[[-0.83168001 -1.14367855 -0.39723824 -0.57849223 -0.36848171 -0.47210798
   0.22800925 -0.84284936]
 [ 0.8647269   1.8633592  0.67258938  0.00976179  0.72124393  0.58740671
   0.24525553  1.28629259]]
```

SAÍDA ESPERADA:

```
train_set_X:
[[ 0.35483802 -0.370418 0.16624635 1.40032403 -0.01279543 0.49863797 -0.61786951 0.01064085]
 [-0.54794048 -0.55620696 0.9024924 0.96548604 0.40069886 1.70739891 -1.0345765 -0.86049025]]
```

```
test_set_X:
[[-0.83168001 -1.14367855 -0.39723824 -0.57849223 -0.36848171 -0.47210798 0.22800925 -0.84284936]
 [ 0.8647269 1.8633592 0.67258938 0.00976179 0.72124393 0.58740671 0.24525553 1.28629259]]
```

Obtenha o número de atributos e o número de exemplos de treinamento

Dica: use a função *shape*: <https://numpy.org/devdocs/reference/generated/numpy.shape.html>

```
### INICIE O CÓDIGO AQUI ### (2 linhas de código)
n = train_set_X.shape[1]# number of attributes
m =train_set_X.shape[0]# number of training examples
### TERMINE O CÓDIGO AQUI ###
```

```
print ("Number of attributes: n = " + str(n))
print ("Number of training examples: m = " + str(m))
print ("Train set X shape: " + str(train_set_X.shape))
print ("Train set Y shape: " + str(train_set_Y.shape))
print ("Test set X shape: " + str(test_set_X.shape))
print ("Test set Y shape: " + str(test_set_Y.shape))
```

```
Number of attributes: n = 8
Number of training examples: m = 614
Train set X shape: (614, 8)
Train set Y shape: (614,)
Test set X shape: (154, 8)
Test set Y shape: (154,)
```

SAÍDA ESPERADA:

```
Number of attributes: n = 8
Number of training examples: m = 614
```

Train set X shape: (614, 8)
Train set Y shape: (614,)
Test set X shape: (154, 8)
Test set Y shape: (154,)

Definição do Modelo

Crie um modelo em Keras com a seguinte configuração:

- Camada de entrada: no formato dos dados de entrada do problema
- Camada 1: 3 neurônios, função de ativação *Tanh*
- Camada 2: 5 neurônios, função de ativação *Tanh*
- Camada 3: 3 neurônios, função de ativação *Tanh*
- Camada 4 (saída): 1 neurônio, função de ativação *Sigmoid*

Dica 1: use a classe *Model*: <https://keras.io/api/models/model/>

Dica 2: veja as funções de ativação disponíveis: <https://keras.io/api/layers/activations/>

```
### INICIE O CÓDIGO AQUI ### (6 linhas de código)
inputs = keras.Input(shape=(train_set_X.shape[1],))
x = keras.layers.Dense(units=3, activation='tanh')(inputs)
x = keras.layers.Dense(units=5, activation='tanh')(x)
x = keras.layers.Dense(units=3, activation='tanh')(x)
outputs = keras.layers.Dense(units=1, activation='sigmoid')(x)
model = keras.Model(inputs=inputs, outputs=outputs)
### TERMINE O CÓDIGO AQUI ###
```

```
# Checking
processed_data = model(train_set_X)
print(processed_data.shape)
```

(614, 1)

SAÍDA ESPERADA:

(614, 1)

```
# Prints a summary of the network, showing its architecture and parameters.
model.summary()
```

Model: "functional"

Layer (type)	Output Shape	Param #
input_layer (InputLayer)	(None, 8)	0
dense (Dense)	(None, 3)	27
dense_1 (Dense)	(None, 5)	20
dense_2 (Dense)	(None, 3)	18
dense_3 (Dense)	(None, 1)	4

Total params: 69 (276.00 B)
Trainable params: 69 (276.00 B)
Non-trainable params: 0 (0.00 B)

SAÍDA ESPERADA:

Confira a configuração da rede e o total de parâmetros = 69

Compilação do Modelo

Compile o model usando os seguintes parâmetros:

- Função de perda: mean_absolute_error
- Otimizador: RMSprop
- Métricas: accuracy, Precision, Recall

Dica 1: use a função *compile*: https://keras.io/api/models/model_training_apis/

Dica 2: relação de funções de perda: https://www.tensorflow.org/api_docs/python/tf/keras/losses

Dica 3: relação de otimizadores: https://www.tensorflow.org/api_docs/python/tf/keras/optimizers

Dica 4: relação de métricas: <https://keras.io/api/metrics/>

```
### INICIE O CÓDIGO AQUI ### (1 linha de código)
model.compile(loss='mean_absolute_error', optimizer='RMSprop', metrics=['accuracy', keras.metrics.Precision(), keras.metrics.Recall()])
### TERMINE O CÓDIGO AQUI ###
```

Treinamento do Modelo

Ajusta o modelo aos dados de treinamento. Devem ser fornecidos os dados de treinamento, o número de épocas (iterações) e o tamanho do lote (batch). Uma época é composta por uma única passagem por todos os exemplos do conjunto de treino. O tamanho do lote define o número de amostras (exemplos) a serem consideradas pelo modelo antes de atualizar os pesos. Assim, uma época é composta por um ou mais lotes.

Efetue o treinamento do modelo usando os seguintes parâmetros:

- Tamanho do lote: 64
- Número de épocas: 1000

Dica: use a função *fit*: https://keras.io/api/models/model_training_apis/

5/5 ————— 0s 14ms/step
Predictions: [0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0]

Correct: [0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0]

Valores aproximados:

[illegible]

Correct: [0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0,
1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0,
0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0]

Modifique a configuração da sua rede e/ou os parâmetros dos métodos com o objetivo de melhorar os resultados das métricas no conjunto de teste. Experimente várias opções.

Adicione abaixo a nova sequência de código que alcançou o melhor resultado.

Fim

Parabéns! Você efetuou todos os passos para criar uma rede neural com várias camadas para um problema de classificação binária.