Tem Diabetes?

Class= Tem Diabetes

Objectivo:

Usando um conjunto de dados de Diabetes, prever se uma pessoa terá diabetes ou não usando caracteristicas medidas.

Dados

Número total de casos:768, 8 variáveis de entrada, uma variável de saída 0= não tem diabetes : 500 1=tem diabetes : 268

Caracteristicas dos dados:

O conjunto de dados contem apenas pacientes do sexo feminino (indios Pima), com 21 ou mais anos de idade

Todos os atributos são numéricos

Pode haver dados inválidos ou nulos

atributos:

pregnancies, Glucose, BloodPressure, SkinThickness, Insulin, BMI, DiabetesPedigreeFunction, Age, **Outcome**

Exemplos

6,148,72,35,0,33.6,0.627,50,1

1,85,66,29,0,26.6,0.351,31,0

Como funciona?

1. Treina com 80% dos casos dados, escolhidos aleatoriamente
2. Testa com os restantes 20% --> AVALIA (ESTATISTICAMENTE) SE O MODELO É CAPAZ DE DESCOBRIR NOVOS CASOS BEM
3. Se o modelo for bom, posso usá-lo
4. Dado um caso novo, medir as quantidades, formar um caso e submeter --> o sistema dá a sua previsão

   caso novo: 3,154,71,55,0,31.6,0.638,40,**?**

NOTA: vamos correr mocalmente, mas SE QUISER CORRER EM COLAB, TERÁ DE TER O SEGUINTE: import os import numpy as np import pandas as pd import seaborn as sns import matplotlib.pyplot as plt %matplotlib inline #dataset = pd.read_csv('pima-indians-diabetes.data.csv') from google.colab import files uploaded = files.upload() import io dataset = pd.read_csv(io.BytesIO(uploaded['pima-indians-diabetes.data.csv'])) # Dataset is now stored in a Pandas Dataframe dataset.head(2)

In [94]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

dataset = pd.read_csv('C:/Users/guibs/Documents/GitHub/SGD/Labs/lab8_class/data/
```

In [95]:
```
dataset.describe(include='all')
```

Out[95]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	Diabete
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	

P1. Which two attributes have top correlation with class?

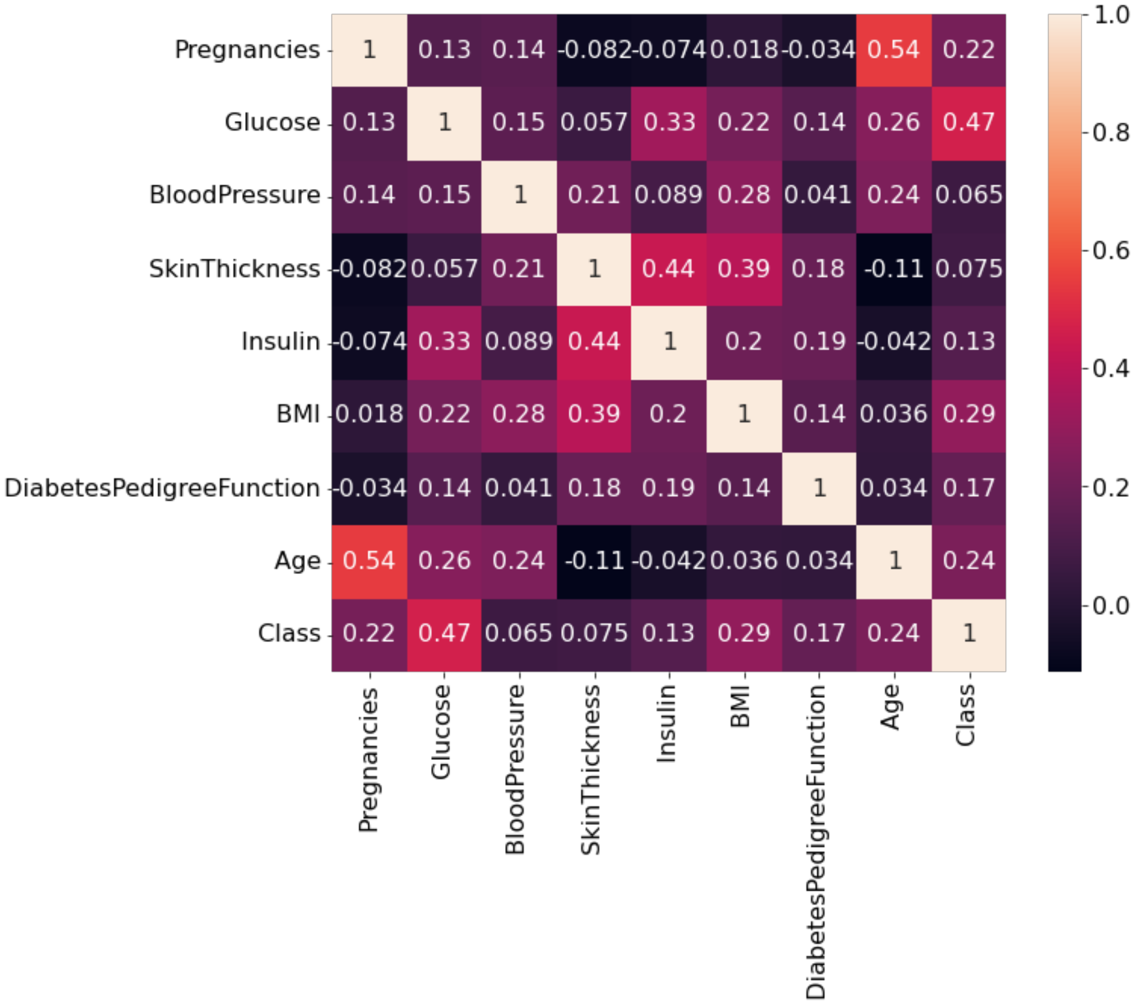
Glocuse Level and BMI.

P2. what two attributes have top correlation between them?

Age and pregnancias are related because people tend to have children at a specific age interval Insulin and skin thickness

In [96]:
```
fig = plt.figure(figsize=(10, 8))
plt.rcParams['font.size'] = '16'
sns.heatmap(dataset.corr(), annot=True)
```

Out[96]: <AxesSubplot:>



P3. What does the next code do?

Normalizar os dados

In [97]:

```
#standardizing the input feature
from sklearn.preprocessing import StandardScaler

X = dataset.iloc[:,0:8]
y = dataset.iloc[:,8]
sc = StandardScaler()

X = sc.fit_transform(X)
X
```

Out[97]:

```
array([[ 0.63994726,  0.84832379,  0.14964075, ...,  0.20401277,
         0.46849198,  1.4259954 ],
       [-0.84488505, -1.12339636, -0.16054575, ..., -0.68442195,
        -0.36506078, -0.19067191],
       [ 1.23388019,  1.94372388, -0.26394125, ..., -1.10325546,
         0.60439732, -0.10558415],
       ...,
       [ 0.3429808 ,  0.00330087,  0.14964075, ..., -0.73518964,
        -0.68519336, -0.27575966],
       [-0.84488505,  0.1597866 , -0.47073225, ..., -0.24020459,
        -0.37110101,  1.17073215],
       [-0.84488505, -0.8730192 ,  0.04624525, ..., -0.20212881,
        -0.47378505, -0.87137393]])
```

P4. WHat does the next code do? how big are train and test data?

We split the dataset in training and testing data, we do this in the default sizes of 70/30 in order the test the model with new data

In [98]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
```

O que faremos de seguida

Pré-processámos os dados e agora estamos prontos para construir a rede neuronal.

Estamos a usar keras para construir a rede neuronal. Importamos a biblioteca keras para criar as camadas de rede neuronal.

Existem dois tipos principais de modelos disponíveis no keras - "Sequential" e "Model". usaremos o modelo "sequential" para construir nossa rede neuronal.

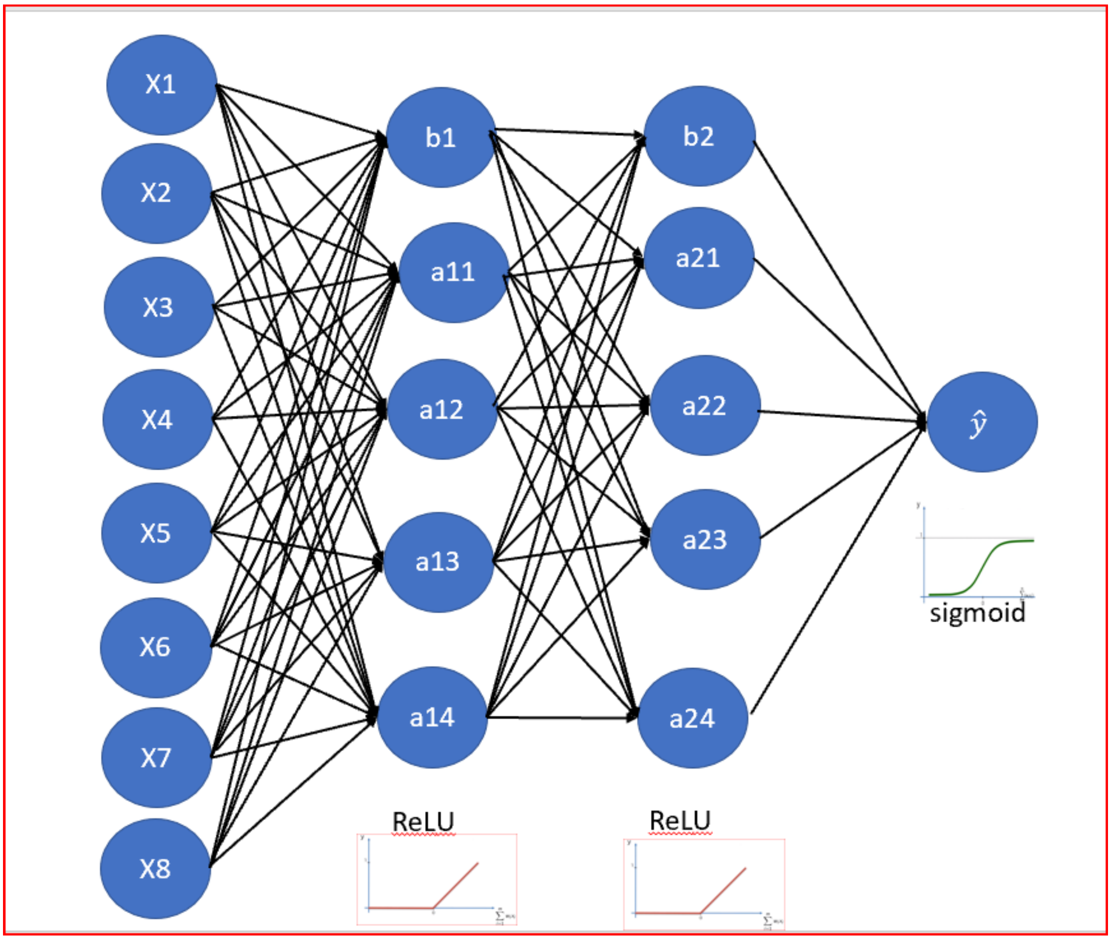
Utilizamos a biblioteca "Dense" para criar camadas de entrada, oculta e saída de uma rede neuronal.

In [99]:
```
from keras import Sequential
from keras.layers import Dense
```

P5. Olhando para a figura da rede neuronal construida, preencha o que falta nos pontos seguintes

(a) Temos 8 variáveis e uma variável objectivo (diabetes ou não). Teremos 2 camadas escondidas (hidden). Cada camada escondida terá 5 nós. Temos uma funcão de activacão dos neuronios das camadas escondidas ReLu, a qual usará a funcão de activacão sigmoide. Nota: uma camada "Dense" implementa a funcão: output = activation(dot(input, kernel) + bias)", sendo que dot é o produto e o kernel são os pesos mais o bias

Keras oferece multiplos initializadores possiveis para o kernel (pesos)



In [100…

```
classifier = Sequential()

#First Hidden Layer
classifier.add(Dense(4, activation='relu', input_dim=8))
#Second  Hidden Layer
classifier.add(Dense(4, activation='relu'))
#Output Layer
classifier.add(Dense(1, activation='sigmoid'))
```

Uma vez criadas as diferentes camadas, compilamos agora a rede neuronal.

Como este é um problema de classificacão binária, usamos binary_crossentropy para calcular a funcão de perda entre a saída real e a saída prevista.

Para otimizar nossa rede neuronal, usamos Adam. Adam significa estimativa adaptativa do momento. Adam é uma combinacão de RMSProp + Momentum.

O momento leva em consideracão os gradientes passados para suavizar a descida do gradiente.

usamos métricas para medir o desempenho do modelo

In [101…

```
#Compiling the neural network
import tensorflow as tf
classifier.compile(optimizer ='adam',loss='binary_crossentropy', metrics =['accu
```

agora ajustamos os dados de treino ao modelo que criámos. usamos um batch_size de 10. Isso implica que usamos 10 amostras por atualizacão de gradiente.

Iteramos mais de 100 épocas para treinar o modelo. Uma época é uma iteracão em todo o conjunto de dados.

P6. what's the "loss"? what do we want loss to be? what about "accuracy", "precision" and "recall"? is this a good result?

Loss is a measurment of error, as the nn is trained the loss lowers, ideally it would be 0. Accurancy is the percentage of casses that were well classified. This is a reasonably a good model

In [102…

```
#Fitting the data to the training dataset
classifier.fit(X_train,y_train, batch_size=10, epochs=100)
```

```
Epoch 1/100
54/54 [==============================] - 0s 510us/step - loss: 0.7044 - accurac
y: 0.6536 - precision_5: 0.3529 - recall_5: 0.0331
Epoch 2/100
54/54 [==============================] - 0s 491us/step - loss: 0.6749 - accurac
y: 0.6629 - precision_5: 0.5000 - recall_5: 0.0331
Epoch 3/100
54/54 [==============================] - 0s 529us/step - loss: 0.6511 - accurac
y: 0.6648 - precision_5: 0.5556 - recall_5: 0.0276
Epoch 4/100
54/54 [==============================] - 0s 548us/step - loss: 0.6301 - accurac
y: 0.6629 - precision_5: 0.5000 - recall_5: 0.0276
Epoch 5/100
54/54 [==============================] - 0s 548us/step - loss: 0.6112 - accurac
y: 0.6648 - precision_5: 0.5714 - recall_5: 0.0221
Epoch 6/100
54/54 [==============================] - 0s 548us/step - loss: 0.5938 - accurac
y: 0.6648 - precision_5: 0.5714 - recall_5: 0.0221
Epoch 7/100
54/54 [==============================] - 0s 548us/step - loss: 0.5781 - accurac
y: 0.6648 - precision_5: 0.5455 - recall_5: 0.0331
Epoch 8/100
54/54 [==============================] - 0s 548us/step - loss: 0.5637 - accurac
y: 0.6648 - precision_5: 0.5294 - recall_5: 0.0497
Epoch 9/100
54/54 [==============================] - 0s 548us/step - loss: 0.5504 - accurac
y: 0.6816 - precision_5: 0.6667 - recall_5: 0.1105
Epoch 10/100
54/54 [==============================] - 0s 548us/step - loss: 0.5389 - accurac
y: 0.6834 - precision_5: 0.6170 - recall_5: 0.1602
Epoch 11/100
54/54 [==============================] - 0s 529us/step - loss: 0.5289 - accurac
y: 0.6983 - precision_5: 0.6610 - recall_5: 0.2155
Epoch 12/100
54/54 [==============================] - 0s 529us/step - loss: 0.5200 - accurac
y: 0.7318 - precision_5: 0.7079 - recall_5: 0.3481
Epoch 13/100
54/54 [==============================] - 0s 548us/step - loss: 0.5110 - accurac
y: 0.7486 - precision_5: 0.7170 - recall_5: 0.4199
Epoch 14/100
54/54 [==============================] - 0s 548us/step - loss: 0.5038 - accurac
y: 0.7523 - precision_5: 0.7069 - recall_5: 0.4530
Epoch 15/100
54/54 [==============================] - 0s 548us/step - loss: 0.4968 - accurac
y: 0.7635 - precision_5: 0.7077 - recall_5: 0.5083
Epoch 16/100
54/54 [==============================] - 0s 548us/step - loss: 0.4912 - accurac
y: 0.7635 - precision_5: 0.7015 - recall_5: 0.5193
Epoch 17/100
54/54 [==============================] - 0s 567us/step - loss: 0.4853 - accurac
y: 0.7728 - precision_5: 0.7007 - recall_5: 0.5691
Epoch 18/100
54/54 [==============================] - 0s 567us/step - loss: 0.4805 - accurac
y: 0.7728 - precision_5: 0.6980 - recall_5: 0.5746
Epoch 19/100
54/54 [==============================] - 0s 585us/step - loss: 0.4771 - accurac
y: 0.7803 - precision_5: 0.7032 - recall_5: 0.6022
Epoch 20/100
54/54 [==============================] - 0s 567us/step - loss: 0.4732 - accurac
y: 0.7858 - precision_5: 0.7115 - recall_5: 0.6133
Epoch 21/100
54/54 [==============================] - 0s 585us/step - loss: 0.4710 - accurac
y: 0.7840 - precision_5: 0.7070 - recall_5: 0.6133
```

```
Epoch 22/100
54/54 [==============================] - 0s 567us/step - loss: 0.4689 - accurac
y: 0.7840 - precision_5: 0.7019 - recall_5: 0.6243
Epoch 23/100
54/54 [==============================] - 0s 548us/step - loss: 0.4664 - accurac
y: 0.7784 - precision_5: 0.6914 - recall_5: 0.6188
Epoch 24/100
54/54 [==============================] - 0s 567us/step - loss: 0.4648 - accurac
y: 0.7765 - precision_5: 0.6871 - recall_5: 0.6188
Epoch 25/100
54/54 [==============================] - 0s 548us/step - loss: 0.4632 - accurac
y: 0.7803 - precision_5: 0.6909 - recall_5: 0.6298
Epoch 26/100
54/54 [==============================] - 0s 548us/step - loss: 0.4616 - accurac
y: 0.7840 - precision_5: 0.6970 - recall_5: 0.6354
Epoch 27/100
54/54 [==============================] - 0s 567us/step - loss: 0.4598 - accurac
y: 0.7840 - precision_5: 0.6970 - recall_5: 0.6354
Epoch 28/100
54/54 [==============================] - 0s 567us/step - loss: 0.4584 - accurac
y: 0.7821 - precision_5: 0.6951 - recall_5: 0.6298
Epoch 29/100
54/54 [==============================] - 0s 567us/step - loss: 0.4577 - accurac
y: 0.7821 - precision_5: 0.6928 - recall_5: 0.6354
Epoch 30/100
54/54 [==============================] - 0s 548us/step - loss: 0.4557 - accurac
y: 0.7858 - precision_5: 0.7012 - recall_5: 0.6354
Epoch 31/100
54/54 [==============================] - 0s 548us/step - loss: 0.4545 - accurac
y: 0.7877 - precision_5: 0.7055 - recall_5: 0.6354
Epoch 32/100
54/54 [==============================] - 0s 548us/step - loss: 0.4537 - accurac
y: 0.7896 - precision_5: 0.7073 - recall_5: 0.6409
Epoch 33/100
54/54 [==============================] - 0s 548us/step - loss: 0.4526 - accurac
y: 0.7914 - precision_5: 0.7143 - recall_5: 0.6354
Epoch 34/100
54/54 [==============================] - 0s 548us/step - loss: 0.4513 - accurac
y: 0.7914 - precision_5: 0.7117 - recall_5: 0.6409
Epoch 35/100
54/54 [==============================] - 0s 567us/step - loss: 0.4506 - accurac
y: 0.7914 - precision_5: 0.7117 - recall_5: 0.6409
Epoch 36/100
54/54 [==============================] - 0s 548us/step - loss: 0.4498 - accurac
y: 0.7914 - precision_5: 0.7117 - recall_5: 0.6409
Epoch 37/100
54/54 [==============================] - 0s 548us/step - loss: 0.4490 - accurac
y: 0.7952 - precision_5: 0.7261 - recall_5: 0.6298
Epoch 38/100
54/54 [==============================] - 0s 567us/step - loss: 0.4489 - accurac
y: 0.7952 - precision_5: 0.7261 - recall_5: 0.6298
Epoch 39/100
54/54 [==============================] - 0s 585us/step - loss: 0.4478 - accurac
y: 0.7970 - precision_5: 0.7278 - recall_5: 0.6354
Epoch 40/100
54/54 [==============================] - 0s 567us/step - loss: 0.4472 - accurac
y: 0.7952 - precision_5: 0.7205 - recall_5: 0.6409
Epoch 41/100
54/54 [==============================] - 0s 548us/step - loss: 0.4472 - accurac
y: 0.7914 - precision_5: 0.7143 - recall_5: 0.6354
Epoch 42/100
54/54 [==============================] - 0s 529us/step - loss: 0.4468 - accurac
y: 0.7952 - precision_5: 0.7233 - recall_5: 0.6354
```

```
Epoch 43/100
54/54 [==============================] - 0s 529us/step - loss: 0.4462 - accurac
y: 0.7970 - precision_5: 0.7308 - recall_5: 0.6298
Epoch 44/100
54/54 [==============================] - 0s 529us/step - loss: 0.4455 - accurac
y: 0.8026 - precision_5: 0.7358 - recall_5: 0.6464
Epoch 45/100
54/54 [==============================] - 0s 548us/step - loss: 0.4447 - accurac
y: 0.7989 - precision_5: 0.7296 - recall_5: 0.6409
Epoch 46/100
54/54 [==============================] - 0s 548us/step - loss: 0.4444 - accurac
y: 0.8007 - precision_5: 0.7342 - recall_5: 0.6409
Epoch 47/100
54/54 [==============================] - 0s 567us/step - loss: 0.4443 - accurac
y: 0.8007 - precision_5: 0.7342 - recall_5: 0.6409
Epoch 48/100
54/54 [==============================] - 0s 529us/step - loss: 0.4436 - accurac
y: 0.8007 - precision_5: 0.7342 - recall_5: 0.6409
Epoch 49/100
54/54 [==============================] - 0s 548us/step - loss: 0.4431 - accurac
y: 0.8007 - precision_5: 0.7342 - recall_5: 0.6409
Epoch 50/100
54/54 [==============================] - 0s 548us/step - loss: 0.4430 - accurac
y: 0.7989 - precision_5: 0.7325 - recall_5: 0.6354
Epoch 51/100
54/54 [==============================] - 0s 548us/step - loss: 0.4426 - accurac
y: 0.7989 - precision_5: 0.7296 - recall_5: 0.6409
Epoch 52/100
54/54 [==============================] - 0s 548us/step - loss: 0.4425 - accurac
y: 0.8007 - precision_5: 0.7312 - recall_5: 0.6464
Epoch 53/100
54/54 [==============================] - 0s 567us/step - loss: 0.4415 - accurac
y: 0.8007 - precision_5: 0.7342 - recall_5: 0.6409
Epoch 54/100
54/54 [==============================] - 0s 567us/step - loss: 0.4412 - accurac
y: 0.7970 - precision_5: 0.7278 - recall_5: 0.6354
Epoch 55/100
54/54 [==============================] - 0s 567us/step - loss: 0.4409 - accurac
y: 0.7952 - precision_5: 0.7261 - recall_5: 0.6298
Epoch 56/100
54/54 [==============================] - 0s 567us/step - loss: 0.4409 - accurac
y: 0.7989 - precision_5: 0.7267 - recall_5: 0.6464
Epoch 57/100
54/54 [==============================] - 0s 567us/step - loss: 0.4408 - accurac
y: 0.7970 - precision_5: 0.7222 - recall_5: 0.6464
Epoch 58/100
54/54 [==============================] - 0s 567us/step - loss: 0.4400 - accurac
y: 0.7896 - precision_5: 0.7179 - recall_5: 0.6188
Epoch 59/100
54/54 [==============================] - 0s 548us/step - loss: 0.4402 - accurac
y: 0.7914 - precision_5: 0.7170 - recall_5: 0.6298
Epoch 60/100
54/54 [==============================] - 0s 567us/step - loss: 0.4400 - accurac
y: 0.7914 - precision_5: 0.7170 - recall_5: 0.6298
Epoch 61/100
54/54 [==============================] - 0s 567us/step - loss: 0.4393 - accurac
y: 0.7933 - precision_5: 0.7215 - recall_5: 0.6298
Epoch 62/100
54/54 [==============================] - 0s 548us/step - loss: 0.4390 - accurac
y: 0.7989 - precision_5: 0.7296 - recall_5: 0.6409
Epoch 63/100
54/54 [==============================] - 0s 567us/step - loss: 0.4384 - accurac
y: 0.7989 - precision_5: 0.7267 - recall_5: 0.6464
```

```
Epoch 64/100
54/54 [==============================] - 0s 567us/step - loss: 0.4383 - accurac
y: 0.8007 - precision_5: 0.7256 - recall_5: 0.6575
Epoch 65/100
54/54 [==============================] - 0s 548us/step - loss: 0.4386 - accurac
y: 0.7933 - precision_5: 0.7188 - recall_5: 0.6354
Epoch 66/100
54/54 [==============================] - 0s 548us/step - loss: 0.4375 - accurac
y: 0.7989 - precision_5: 0.7267 - recall_5: 0.6464
Epoch 67/100
54/54 [==============================] - 0s 548us/step - loss: 0.4375 - accurac
y: 0.7970 - precision_5: 0.7250 - recall_5: 0.6409
Epoch 68/100
54/54 [==============================] - 0s 548us/step - loss: 0.4371 - accurac
y: 0.7952 - precision_5: 0.7178 - recall_5: 0.6464
Epoch 69/100
54/54 [==============================] - 0s 548us/step - loss: 0.4365 - accurac
y: 0.7970 - precision_5: 0.7195 - recall_5: 0.6519
Epoch 70/100
54/54 [==============================] - 0s 548us/step - loss: 0.4364 - accurac
y: 0.7970 - precision_5: 0.7195 - recall_5: 0.6519
Epoch 71/100
54/54 [==============================] - 0s 548us/step - loss: 0.4360 - accurac
y: 0.7970 - precision_5: 0.7195 - recall_5: 0.6519
Epoch 72/100
54/54 [==============================] - 0s 567us/step - loss: 0.4358 - accurac
y: 0.8007 - precision_5: 0.7312 - recall_5: 0.6464
Epoch 73/100
54/54 [==============================] - 0s 548us/step - loss: 0.4354 - accurac
y: 0.7970 - precision_5: 0.7195 - recall_5: 0.6519
Epoch 74/100
54/54 [==============================] - 0s 548us/step - loss: 0.4356 - accurac
y: 0.7970 - precision_5: 0.7143 - recall_5: 0.6630
Epoch 75/100
54/54 [==============================] - 0s 548us/step - loss: 0.4351 - accurac
y: 0.7952 - precision_5: 0.7152 - recall_5: 0.6519
Epoch 76/100
54/54 [==============================] - 0s 529us/step - loss: 0.4346 - accurac
y: 0.7970 - precision_5: 0.7143 - recall_5: 0.6630
Epoch 77/100
54/54 [==============================] - 0s 548us/step - loss: 0.4345 - accurac
y: 0.7970 - precision_5: 0.7143 - recall_5: 0.6630
Epoch 78/100
54/54 [==============================] - 0s 548us/step - loss: 0.4341 - accurac
y: 0.8007 - precision_5: 0.7229 - recall_5: 0.6630
Epoch 79/100
54/54 [==============================] - 0s 548us/step - loss: 0.4343 - accurac
y: 0.8026 - precision_5: 0.7301 - recall_5: 0.6575
Epoch 80/100
54/54 [==============================] - 0s 548us/step - loss: 0.4333 - accurac
y: 0.8026 - precision_5: 0.7273 - recall_5: 0.6630
Epoch 81/100
54/54 [==============================] - 0s 548us/step - loss: 0.4339 - accurac
y: 0.7952 - precision_5: 0.7101 - recall_5: 0.6630
Epoch 82/100
54/54 [==============================] - 0s 548us/step - loss: 0.4332 - accurac
y: 0.8007 - precision_5: 0.7229 - recall_5: 0.6630
Epoch 83/100
54/54 [==============================] - 0s 567us/step - loss: 0.4328 - accurac
y: 0.8007 - precision_5: 0.7229 - recall_5: 0.6630
Epoch 84/100
54/54 [==============================] - 0s 567us/step - loss: 0.4324 - accurac
y: 0.8026 - precision_5: 0.7273 - recall_5: 0.6630
```

```
Epoch 85/100
54/54 [==============================] - 0s 548us/step - loss: 0.4324 - accurac
y: 0.8045 - precision_5: 0.7317 - recall_5: 0.6630
Epoch 86/100
54/54 [==============================] - 0s 567us/step - loss: 0.4321 - accurac
y: 0.8007 - precision_5: 0.7229 - recall_5: 0.6630
Epoch 87/100
54/54 [==============================] - 0s 567us/step - loss: 0.4316 - accurac
y: 0.7970 - precision_5: 0.7143 - recall_5: 0.6630
Epoch 88/100
54/54 [==============================] - 0s 567us/step - loss: 0.4315 - accurac
y: 0.7989 - precision_5: 0.7186 - recall_5: 0.6630
Epoch 89/100
54/54 [==============================] - 0s 585us/step - loss: 0.4315 - accurac
y: 0.8045 - precision_5: 0.7317 - recall_5: 0.6630
Epoch 90/100
54/54 [==============================] - 0s 585us/step - loss: 0.4312 - accurac
y: 0.8045 - precision_5: 0.7317 - recall_5: 0.6630
Epoch 91/100
54/54 [==============================] - 0s 548us/step - loss: 0.4305 - accurac
y: 0.7989 - precision_5: 0.7186 - recall_5: 0.6630
Epoch 92/100
54/54 [==============================] - 0s 548us/step - loss: 0.4305 - accurac
y: 0.8007 - precision_5: 0.7229 - recall_5: 0.6630
Epoch 93/100
54/54 [==============================] - 0s 567us/step - loss: 0.4304 - accurac
y: 0.8063 - precision_5: 0.7362 - recall_5: 0.6630
Epoch 94/100
54/54 [==============================] - 0s 548us/step - loss: 0.4304 - accurac
y: 0.7989 - precision_5: 0.7186 - recall_5: 0.6630
Epoch 95/100
54/54 [==============================] - 0s 548us/step - loss: 0.4310 - accurac
y: 0.8063 - precision_5: 0.7362 - recall_5: 0.6630
Epoch 96/100
54/54 [==============================] - 0s 548us/step - loss: 0.4303 - accurac
y: 0.7989 - precision_5: 0.7186 - recall_5: 0.6630
Epoch 97/100
54/54 [==============================] - 0s 548us/step - loss: 0.4298 - accurac
y: 0.8063 - precision_5: 0.7362 - recall_5: 0.6630
Epoch 98/100
54/54 [==============================] - 0s 548us/step - loss: 0.4293 - accurac
y: 0.8007 - precision_5: 0.7229 - recall_5: 0.6630
Epoch 99/100
54/54 [==============================] - 0s 529us/step - loss: 0.4299 - accurac
y: 0.8026 - precision_5: 0.7273 - recall_5: 0.6630
Epoch 100/100
54/54 [==============================] - 0s 548us/step - loss: 0.4292 - accurac
y: 0.8045 - precision_5: 0.7317 - recall_5: 0.6630
```

Out[102]:

```
<keras.callbacks.History at 0x180f9550850>
```

P8. Is the comparison of results train/test the one you would expect? why?

There's a little bit of a loss in the test data because the model is trained for the train data. however the loss isnt big so the model isn't over-fitted as it works well for the test data

In [103…

```
eval_model=classifier.evaluate(X_train, y_train)
eval_model
```

```
17/17 [==============================] - 0s 563us/step - loss: 0.4280 - accurac
y: 0.8063 - precision_5: 0.7362 - recall_5: 0.6630
```

Out[103]:

```
[0.4279765784740448, 0.80633145570755, 0.7361963391304016, 0.6629834175109863]
```

In [104…

```
eval_model=classifier.evaluate(X_test, y_test)
eval_model
```

```
8/8 [==============================] - 0s 572us/step - loss: 0.5142 - accuracy:
0.7489 - precision_5: 0.7042 - recall_5: 0.5747
```

Out[104]:

```
[0.5141667723655701,
 0.7489177584648132,
 0.7042253613471985,
 0.5747126340866089]
```

Tell us if the first 5 estimations are correct? Why?

For the first one, the real is 0 and the estimation was <0.00005 The second one matches as well Third one matches 4 doesnt and 5fth does

In [105…

```
y1_pred=classifier.predict(X_test)
y_pred =(y1_pred>0.5)
```

In [106…

```
import numpy as np
y1_test = pd.DataFrame(y_test.values)
np.stack((y1_test.values,y1_pred),1)
```

Out[106]:
```
array([[[0.00000000e+00],
        [4.19051647e-02]],

       [[1.00000000e+00],
        [1.70485735e-01]],

       [[0.00000000e+00],
        [8.63781273e-02]],

       [[0.00000000e+00],
        [9.74681973e-03]],

       [[0.00000000e+00],
        [4.15869057e-02]],

       [[1.00000000e+00],
        [9.10970569e-01]],

       [[1.00000000e+00],
        [4.42241371e-01]],

       [[0.00000000e+00],
        [7.42866099e-02]],

       [[0.00000000e+00],
        [1.70955956e-02]],

       [[0.00000000e+00],
        [2.30990529e-01]],

       [[0.00000000e+00],
        [2.11915880e-01]],

       [[1.00000000e+00],
        [1.95084959e-01]],

       [[0.00000000e+00],
        [3.18321586e-01]],

       [[0.00000000e+00],
        [2.73628414e-01]],

       [[0.00000000e+00],
        [7.69550204e-01]],

       [[0.00000000e+00],
        [4.36064124e-01]],

       [[0.00000000e+00],
        [4.83161986e-01]],

       [[1.00000000e+00],
        [6.89603090e-02]],

       [[0.00000000e+00],
        [1.86583400e-01]],

       [[0.00000000e+00],
        [9.09952223e-02]],

       [[0.00000000e+00],
        [5.35147786e-02]],
```

```
       [[0.00000000e+00],
        [8.41509402e-02]],

       [[1.00000000e+00],
        [7.49523818e-01]],

       [[0.00000000e+00],
        [7.93950260e-02]],

       [[0.00000000e+00],
        [1.57448322e-01]],

       [[0.00000000e+00],
        [1.58143997e-01]],

       [[0.00000000e+00],
        [4.12296355e-01]],

       [[1.00000000e+00],
        [8.34241629e-01]],

       [[0.00000000e+00],
        [1.29341483e-02]],

       [[0.00000000e+00],
        [2.17387974e-02]],

       [[0.00000000e+00],
        [1.24863684e-02]],

       [[0.00000000e+00],
        [5.11075556e-02]],

       [[0.00000000e+00],
        [4.46627736e-02]],

       [[1.00000000e+00],
        [8.85143995e-01]],

       [[0.00000000e+00],
        [3.50569934e-01]],

       [[1.00000000e+00],
        [7.14002728e-01]],

       [[0.00000000e+00],
        [1.01713449e-01]],

       [[1.00000000e+00],
        [5.73615134e-02]],

       [[0.00000000e+00],
        [4.83161986e-01]],

       [[1.00000000e+00],
        [3.31524819e-01]],

       [[0.00000000e+00],
        [1.74459964e-01]],

       [[0.00000000e+00],
        [9.63623524e-01]],
```

```
[[1.00000000e+00],
 [7.18134940e-01]],

[[0.00000000e+00],
 [5.55219352e-02]],

[[0.00000000e+00],
 [7.22114623e-01]],

[[0.00000000e+00],
 [2.90758610e-02]],

[[0.00000000e+00],
 [3.92775297e-01]],

[[0.00000000e+00],
 [8.28418136e-02]],

[[1.00000000e+00],
 [4.97886449e-01]],

[[0.00000000e+00],
 [8.05512428e-01]],

[[0.00000000e+00],
 [7.06805587e-01]],

[[1.00000000e+00],
 [1.02960229e-01]],

[[0.00000000e+00],
 [6.79021478e-02]],

[[0.00000000e+00],
 [1.90976858e-02]],

[[0.00000000e+00],
 [2.08216012e-02]],

[[1.00000000e+00],
 [7.96986997e-01]],

[[1.00000000e+00],
 [4.94535208e-01]],

[[0.00000000e+00],
 [1.08886242e-01]],

[[0.00000000e+00],
 [2.06055343e-02]],

[[0.00000000e+00],
 [2.86592901e-01]],

[[1.00000000e+00],
 [6.16617084e-01]],

[[1.00000000e+00],
 [1.71490103e-01]],

[[0.00000000e+00],
 [3.81058156e-01]],
```

```
[[1.00000000e+00],
 [3.86805832e-01]],

[[1.00000000e+00],
 [7.45732069e-01]],

[[0.00000000e+00],
 [3.05386782e-02]],

[[0.00000000e+00],
 [5.23006082e-01]],

[[0.00000000e+00],
 [1.62362754e-02]],

[[1.00000000e+00],
 [7.69595504e-02]],

[[1.00000000e+00],
 [8.87106895e-01]],

[[0.00000000e+00],
 [2.34959602e-01]],

[[1.00000000e+00],
 [8.17133069e-01]],

[[0.00000000e+00],
 [1.57654792e-01]],

[[0.00000000e+00],
 [6.16246760e-02]],

[[0.00000000e+00],
 [4.04267311e-02]],

[[0.00000000e+00],
 [6.30097151e-01]],

[[0.00000000e+00],
 [2.29455411e-01]],

[[1.00000000e+00],
 [6.29160225e-01]],

[[1.00000000e+00],
 [1.15978897e-01]],

[[0.00000000e+00],
 [3.58091593e-02]],

[[0.00000000e+00],
 [3.34790647e-02]],

[[1.00000000e+00],
 [2.81481326e-01]],

[[0.00000000e+00],
 [1.62232041e-01]],

[[1.00000000e+00],
 [3.07229280e-01]],
```

```
[[0.00000000e+00],
 [2.04647779e-01]],

[[1.00000000e+00],
 [5.37780821e-01]],

[[0.00000000e+00],
 [7.79473305e-01]],

[[0.00000000e+00],
 [4.66384590e-01]],

[[1.00000000e+00],
 [4.73361850e-01]],

[[0.00000000e+00],
 [1.26946718e-01]],

[[1.00000000e+00],
 [6.86594307e-01]],

[[0.00000000e+00],
 [3.98972303e-01]],

[[0.00000000e+00],
 [4.88289297e-02]],

[[0.00000000e+00],
 [1.34828269e-01]],

[[0.00000000e+00],
 [3.35113406e-02]],

[[0.00000000e+00],
 [2.23888636e-01]],

[[1.00000000e+00],
 [3.30302835e-01]],

[[0.00000000e+00],
 [5.18195033e-02]],

[[1.00000000e+00],
 [2.33246297e-01]],

[[1.00000000e+00],
 [5.99363089e-01]],

[[0.00000000e+00],
 [1.36648357e-01]],

[[1.00000000e+00],
 [6.21530414e-03]],

[[1.00000000e+00],
 [1.00766987e-01]],

[[1.00000000e+00],
 [6.74648046e-01]],

[[0.00000000e+00],
 [1.22657597e-01]],
```

```
[[0.00000000e+00],
 [6.28742039e-01]],

[[1.00000000e+00],
 [6.42318606e-01]],

[[0.00000000e+00],
 [2.73707867e-01]],

[[0.00000000e+00],
 [8.11967254e-03]],

[[0.00000000e+00],
 [2.11015880e-01]],

[[1.00000000e+00],
 [7.89864659e-01]],

[[1.00000000e+00],
 [5.43404698e-01]],

[[0.00000000e+00],
 [3.89183760e-02]],

[[1.00000000e+00],
 [7.94957161e-01]],

[[0.00000000e+00],
 [1.60840154e-03]],

[[0.00000000e+00],
 [8.05041790e-02]],

[[0.00000000e+00],
 [4.80265021e-02]],

[[1.00000000e+00],
 [6.51893556e-01]],

[[1.00000000e+00],
 [7.24870682e-01]],

[[1.00000000e+00],
 [8.61164272e-01]],

[[1.00000000e+00],
 [3.00170958e-01]],

[[1.00000000e+00],
 [3.03847849e-01]],

[[0.00000000e+00],
 [7.33246803e-02]],

[[0.00000000e+00],
 [2.32945085e-02]],

[[0.00000000e+00],
 [7.00594306e-01]],

[[1.00000000e+00],
 [8.29412460e-01]],
```

```
[[1.00000000e+00],
 [7.20144808e-01]],

[[0.00000000e+00],
 [4.32786345e-01]],

[[1.00000000e+00],
 [8.19831491e-01]],

[[0.00000000e+00],
 [3.16154659e-02]],

[[0.00000000e+00],
 [4.16725874e-03]],

[[0.00000000e+00],
 [2.83527374e-02]],

[[0.00000000e+00],
 [1.33217573e-02]],

[[0.00000000e+00],
 [3.82827580e-01]],

[[0.00000000e+00],
 [4.09663618e-01]],

[[0.00000000e+00],
 [5.09913027e-01]],

[[1.00000000e+00],
 [4.83161986e-01]],

[[0.00000000e+00],
 [6.51515841e-01]],

[[1.00000000e+00],
 [3.10893893e-01]],

[[0.00000000e+00],
 [4.65627670e-01]],

[[1.00000000e+00],
 [6.48349643e-01]],

[[1.00000000e+00],
 [8.83718848e-01]],

[[0.00000000e+00],
 [1.47591054e-01]],

[[0.00000000e+00],
 [1.86750054e-01]],

[[0.00000000e+00],
 [3.73926163e-02]],

[[0.00000000e+00],
 [2.64197588e-04]],

[[0.00000000e+00],
 [2.15172619e-01]],
```

```
[[0.00000000e+00],
 [1.87537700e-01]],

[[1.00000000e+00],
 [8.22409809e-01]],

[[0.00000000e+00],
 [1.33641660e-02]],

[[0.00000000e+00],
 [5.38992584e-02]],

[[0.00000000e+00],
 [5.52861989e-01]],

[[0.00000000e+00],
 [4.86302346e-01]],

[[0.00000000e+00],
 [6.11797512e-01]],

[[1.00000000e+00],
 [8.35637450e-01]],

[[0.00000000e+00],
 [5.63997269e-01]],

[[0.00000000e+00],
 [5.89982867e-02]],

[[1.00000000e+00],
 [9.06287909e-01]],

[[1.00000000e+00],
 [5.56770265e-01]],

[[0.00000000e+00],
 [1.59991980e-02]],

[[0.00000000e+00],
 [6.13298833e-01]],

[[1.00000000e+00],
 [2.66835809e-01]],

[[1.00000000e+00],
 [4.83161986e-01]],

[[0.00000000e+00],
 [1.25336558e-01]],

[[1.00000000e+00],
 [2.16938704e-01]],

[[1.00000000e+00],
 [6.46078825e-01]],

[[0.00000000e+00],
 [6.59598172e-01]],

[[0.00000000e+00],
 [7.76376128e-02]],
```

```
[[0.00000000e+00],
 [2.52971292e-01]],

[[0.00000000e+00],
 [1.27730429e-01]],

[[0.00000000e+00],
 [1.02685034e-01]],

[[0.00000000e+00],
 [1.66474015e-01]],

[[1.00000000e+00],
 [9.05044675e-01]],

[[0.00000000e+00],
 [2.27006972e-02]],

[[1.00000000e+00],
 [2.72450507e-01]],

[[0.00000000e+00],
 [8.23473930e-01]],

[[1.00000000e+00],
 [3.51333261e-01]],

[[0.00000000e+00],
 [5.64578056e-01]],

[[0.00000000e+00],
 [3.13637912e-01]],

[[0.00000000e+00],
 [7.53473938e-02]],

[[1.00000000e+00],
 [2.86802351e-01]],

[[0.00000000e+00],
 [2.67079026e-01]],

[[1.00000000e+00],
 [6.13296866e-01]],

[[0.00000000e+00],
 [4.56070304e-02]],

[[1.00000000e+00],
 [6.97800994e-01]],

[[1.00000000e+00],
 [5.85947931e-01]],

[[1.00000000e+00],
 [8.33560705e-01]],

[[1.00000000e+00],
 [2.19991922e-01]],

[[1.00000000e+00],
 [7.84728765e-01]],
```

```
[[0.00000000e+00],
 [1.95622742e-02]],

[[1.00000000e+00],
 [7.90143013e-01]],

[[0.00000000e+00],
 [1.82824135e-02]],

[[1.00000000e+00],
 [4.85337704e-01]],

[[0.00000000e+00],
 [3.29596996e-02]],

[[0.00000000e+00],
 [5.57469010e-01]],

[[0.00000000e+00],
 [2.72430778e-01]],

[[0.00000000e+00],
 [3.19072843e-01]],

[[0.00000000e+00],
 [1.38021708e-02]],

[[1.00000000e+00],
 [8.46935809e-02]],

[[0.00000000e+00],
 [4.44105774e-01]],

[[0.00000000e+00],
 [3.05427611e-02]],

[[1.00000000e+00],
 [7.01158524e-01]],

[[1.00000000e+00],
 [7.58924723e-01]],

[[1.00000000e+00],
 [2.79104859e-01]],

[[1.00000000e+00],
 [2.66898870e-01]],

[[0.00000000e+00],
 [4.35499549e-02]],

[[0.00000000e+00],
 [1.07292831e-02]],

[[0.00000000e+00],
 [4.16746318e-01]],

[[0.00000000e+00],
 [2.61716366e-01]],

[[1.00000000e+00],
 [5.15282094e-01]],
```

```
[[0.00000000e+00],
 [1.61011159e-01]],

[[1.00000000e+00],
 [7.64420271e-01]],

[[0.00000000e+00],
 [1.77265406e-01]],

[[0.00000000e+00],
 [1.28653497e-01]],

[[1.00000000e+00],
 [8.76136959e-01]],

[[1.00000000e+00],
 [4.22255665e-01]],

[[0.00000000e+00],
 [2.74811685e-02]],

[[1.00000000e+00],
 [5.89290023e-01]],

[[0.00000000e+00],
 [9.82698798e-03]],

[[1.00000000e+00],
 [7.46450305e-01]],

[[0.00000000e+00],
 [1.76856071e-01]],

[[0.00000000e+00],
 [1.83990300e-02]],

[[0.00000000e+00],
 [1.30047798e-02]],

[[0.00000000e+00],
 [5.63855052e-01]],

[[1.00000000e+00],
 [7.87477136e-01]],

[[1.00000000e+00],
 [7.95959234e-01]],

[[0.00000000e+00],
 [3.03110957e-01]],

[[0.00000000e+00],
 [5.23776710e-02]],

[[1.00000000e+00],
 [5.13402879e-01]],

[[0.00000000e+00],
 [1.54330820e-01]],

[[1.00000000e+00],
 [1.28895074e-01]]]])
```

P11. What does the next matrix tell us?

40 false positives 39 false negatives

In [107…

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
print(cm)
```

```
[[123  21]
 [ 37  50]]
```

C:\Users\guibs\Documents\GitHub\SGD\Labs\lab8_class# THE END

Algumas respostas

Temos 8 variáveis e uma variável objectivo (diabetes ou não).

Teremos duas camadas escondidas (hidden).

Cada camada escondida terá 4 nós.

Temos uma funcão de activacão dos neuronios das camadas escondidas ReLu, a qual usará a funcão de activacão sigmoide

"Dense layer implements output = activation(dot(input, kernel) + bias)"

Keras oferece multiplos initializadores possiveis

Nota: uma camada "Dense" implementa a funcão: output = activation(dot(input, kernel) + bias)", sendo que dot é o produto e o kernel são pesos

Keras oferece multiplos initializadores possiveis para o kernel (pesos)