

# ENG1456 - Redes Neurais - Trabalho 1

## Classificação

Aluno: Matheus Carneiro Nogueira - 1810764

Professora: Marley Velasco

## Sumário

<b>1 Apresentação do dataset</b>	<b>2</b>
<b>2 Compreensão do problema e análise de variáveis</b>	<b>2</b>
2.1 Observe a base de dados do problema. Existem variáveis que podem ser eliminadas do dataset? Justifique. . . . .	2
2.2 Implemente técnicas de visualização de dados e seleção de variáveis para extrair características importantes sobre a base de dados. Explique a motivação destas técnicas e o que é possível inferir dos resultados obtidos. . . . .	3
<b>3 Treinamento do modelo de Rede Neural</b>	<b>6</b>
3.1 Com as configurações do modelo MLP previamente definidas no script, faça o treinamento da Rede Neural sem normalizar os atributos numéricos. Comente o resultado obtido, baseado nas métricas de avaliação disponíveis (acurácia, precision, recall, F1-Score, Matriz de confusão, etc.) . . . . .	6
3.2 Agora normalize os dados de entrada e treine novamente o modelo MLP. Avalie os resultados obtidos e comente o efeito da normalização no treinamento da Rede Neural. . . . .	9
<b>4 Mudança de configurações do modelo</b>	<b>11</b>
4.1 Insira o conjunto de validação para o treinamento do modelo. Avalie o resultado obtido. . . . .	12
4.2 Modifique o tempo de treinamento (épocas) da Rede Neural. Escolha dois valores distintos (e.g. 1 e 1000 épocas) e avalie os resultados. . . . .	14
4.2.1 5 épocas . . . . .	14
4.2.2 1000 épocas . . . . .	16
4.3 Modifique a taxa de aprendizado da Rede Neural. Escolha dois valores distintos (e.g. 0,001 e 0,1) e avalie os resultados. . . . .	17
4.3.1 Taxa de aprendizado = 0.001 . . . . .	17
4.3.2 Taxa de aprendizado = 0.4 . . . . .	19
4.4 Modifique a quantidade de neurônios na camada escondida da Rede Neural. Escolha dois valores distintos (e.g. 2 e 70 neurônios) e avalie os resultados. . . . .	21
4.4.1 5 neurônios . . . . .	22
4.4.2 50 neurônios . . . . .	24

<b>5 Teste Livre</b>	<b>26</b>
5.1 Faça novos testes para avaliar o desempenho da Rede Neural no problema designado. Use a técnica K-Fold (com $K = 10$ ) para analisar o resultado obtido. . . . .	26
5.2 Faça análises e novas implementações que você julgue importante para o seu trabalho. Não esqueça de explicar a motivação da análise realizada. . . . .	26
5.2.1 Early Stopping . . . . .	26

### Resumo

Este documento consiste no relatório do trabalho 1 do módulo de Redes Neurais da disciplina ENG1456 da PUC-Rio. Nele será explicada a implementação de modelos de Redes Neurais MLP para a classificação do dataset Ionosphere, disponibilizado pela professora da disciplina e disponível em [1]. As seções do relatório são definidas de acordo com as perguntas principais que constam no arquivo Guia de Atividades.

## 1 Apresentação do dataset

Com o intuito de criar um modelo de rede neural para classificar o dataset *Ionosphere*, é essencial que, antes de implementar a rede, estudemos o dataset em si. Como descrito no artigo [2] e pelas informações disponibilizadas em [1], este dataset consiste no sistema de radares da região de Goose Bay, Labrador, o qual possui 16 antenas de alta frequência. O alvo dessas antenas foram elétrons livres na Ionosfera e, bons retornos do radar consistem em retornos que indicam alguma estrutura na ionosfera, enquanto mals retornos são aqueles cujo sinal passa livre pela Ionosfera e não retorna. É justamente esta classificação que a Rede Neural desenvolvida neste trabalho almeja realizar, separar as entradas em "good" ou "bad". Com isso, notamos que a rede pode possuir uma saída única, que mapeia 0 em "bad" e 1 em "good".

## 2 Compreensão do problema e análise de variáveis

### 2.1 Observe a base de dados do problema. Existem variáveis que podem ser eliminadas do dataset? Justifique.

Ao analisar as colunas do dataset, percebe-se que as duas primeiras colunas, *info\_1* e *info\_2* aparentam ser colunas inteiramente de 1 e 0, respectivamente. Para ter certeza, uma vez que observar apenas o head e tail não é suficiente, as colunas inteiras foram analisadas e confirmou-se que a segunda coluna, *info\_1* de fato é inteiramente de 0. Sendo assim, é recomendável que ela seja excluída do dataset pois, além de não trazer nenhuma informação útil à classificação dos retornos de rádio, essa informação pode atrapalhar o treinamento da rede, uma vez que ambos os retornos bons e ruins apresentam valor 0 neste atributo.

**2.2 Implemente técnicas de visualização de dados e seleção de variáveis para extrair características importantes sobre a base de dados. Explique a motivação destas técnicas e o que é possível inferir dos resultados obtidos.**

Foi utilizado o *pairplot* da biblioteca seaborn para visualizar os valores das entradas de nossa rede neural com o intuito de procurar alguma informação útil, tanto para a compreensão da base de dados, quanto para a modelagem da rede neural. Serão adicionadas apenas as imagens úteis para alguma interpretação, com o intuito de não poluir desnecessariamente este relatório.

Como pode ser visto na imagem abaixo, figura 1, os retornos classificados como bons parecem possuir alguma relação com os valores maiores ou iguais a zero nas colunas info\_2, info\_4 e info\_6. Esta informação é interessante pois, caso alguma entrada de teste possua, nestas colunas, valores maiores que zero, há uma chance maior de pertencerem à classe "good".

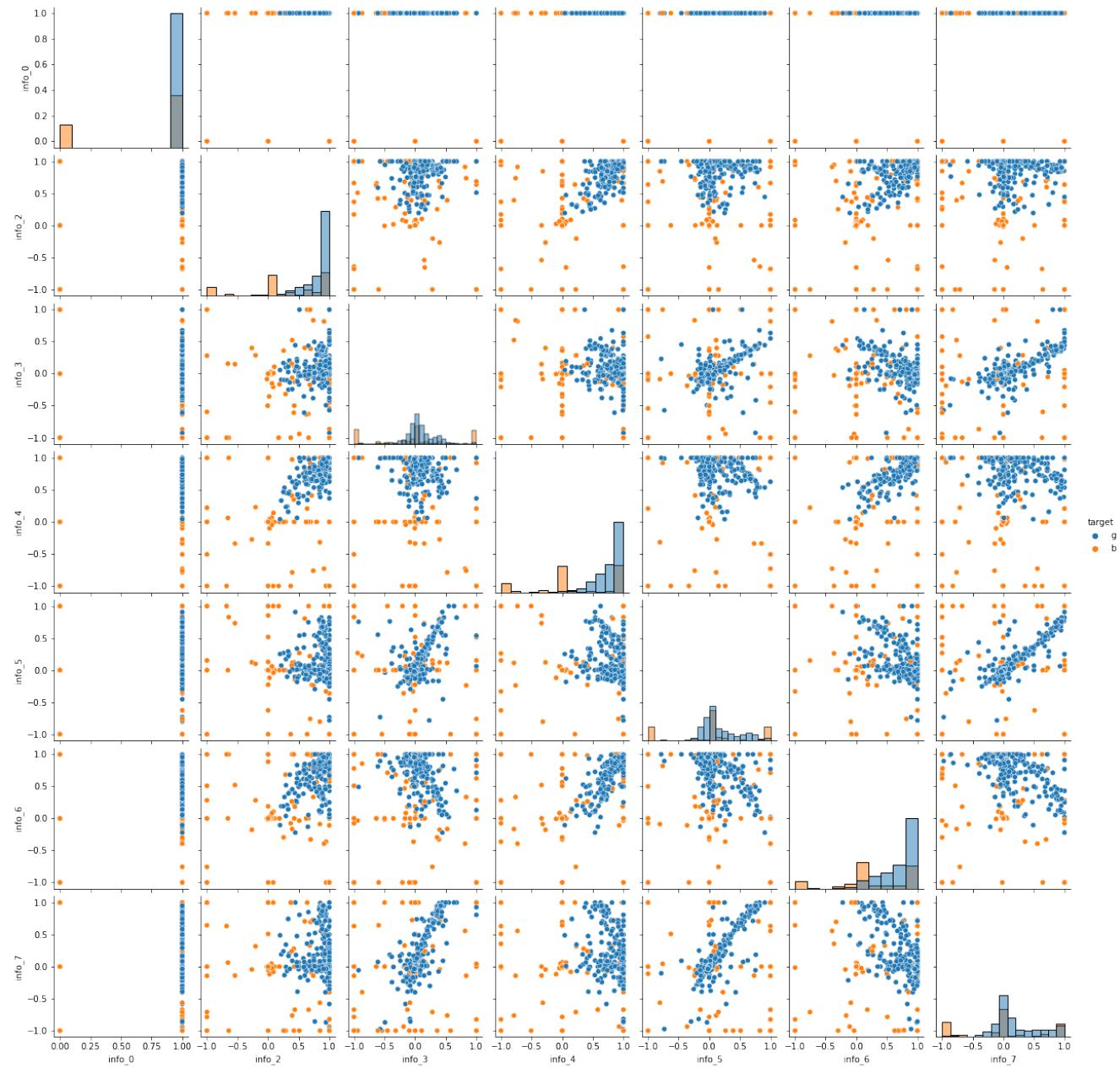
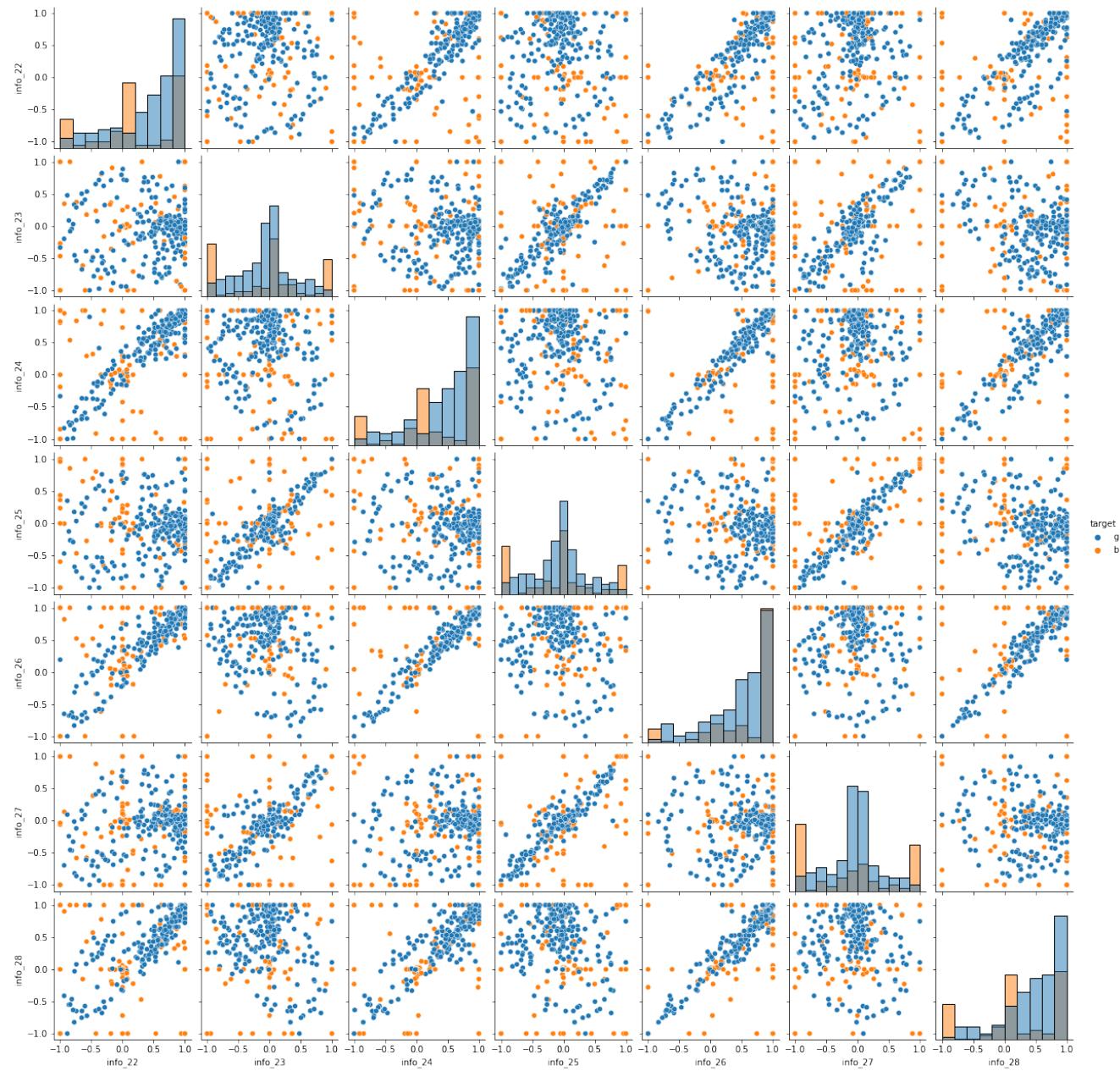


Figura 1: info\_0 a info\_7

Já a figura 2 parece revelar uma relação linear entre os retornos classificados como ”good” ao olharmos as colunas info\_23, info\_24, info\_26 e info\_28. Isso pode indicar que, entradas que possuam estes valores próximos à uma determinada reta possuem maior probabilidade de serem da classe ”good”.

Figura 2: `info_21` a `info_28`

Por fim, todas as imagens mostram que os retornos da classe "good", em azul, apresentam algum nível de concentração ou padrão, enquanto os retornos "bad" são aqueles mais dispersos ao longo de toda a escala de valores.

### 3 Treinamento do modelo de Rede Neural

- 3.1 Com as configurações do modelo MLP previamente definidas no script, faça o treinamento da Rede Neural sem normalizar os atributos numéricos. Comente o resultado obtido, baseado nas métricas de avaliação disponíveis (acurácia, precision, recall, F1-Score, Matriz de confusão, etc.)

O nosso primeiro modelo de rede MLP foi criado de acordo com as especificações já presentes no script disponibilizado. Essa Rede Neural possui as seguintes especificações:

Model: "sequential"		
Layer (type)	Output Shape	Param #
dense (Dense)	(None, 15)	510
dense_1 (Dense)	(None, 1)	16
<hr/>		
Total params: 526		
Trainable params: 526		
Non-trainable params: 0		
<hr/>		

Figura 3: Sumário do Modelo MLP Não Normalizado

Como a figura 3 revela, nossa rede possui 1 camada escondida completamente conectada (densa) com 15 neurônios e a camada de saída com apenas 1 neurônio, como explicado na seção 1. Além disso, nota-se que o número de parâmetros treináveis é 526. Essa informação será importante quando variarmos o número de épocas de treinamento, que neste caso é 50, uma vez que possuir épocas demais pode significar um overfitting de nosso modelo, gerando perda de generalização.

Para analisar a acurácia deste modelo, observemos as figuras abaixo.

```

Epoch 1/50
9/9 [=====] - 1s 4ms/step - loss: 0.5498 - accuracy: 0.7346
Epoch 2/50
9/9 [=====] - 0s 1ms/step - loss: 0.2430 - accuracy: 0.9342
Epoch 3/50
9/9 [=====] - 0s 1ms/step - loss: 0.2149 - accuracy: 0.9181
Epoch 4/50
9/9 [=====] - 0s 1ms/step - loss: 0.1356 - accuracy: 0.9420
Epoch 5/50
9/9 [=====] - 0s 1ms/step - loss: 0.1175 - accuracy: 0.9635
Epoch 6/50
9/9 [=====] - 0s 1ms/step - loss: 0.1242 - accuracy: 0.9543
Epoch 7/50
9/9 [=====] - 0s 1ms/step - loss: 0.0903 - accuracy: 0.9630
Epoch 8/50
9/9 [=====] - 0s 1ms/step - loss: 0.0681 - accuracy: 0.9764
Epoch 9/50
9/9 [=====] - 0s 1ms/step - loss: 0.0825 - accuracy: 0.9806
Epoch 10/50
9/9 [=====] - 0s 1ms/step - loss: 0.0431 - accuracy: 0.9858
Epoch 11/50
9/9 [=====] - 0s 1ms/step - loss: 0.0483 - accuracy: 0.9829
Epoch 12/50
9/9 [=====] - 0s 1ms/step - loss: 0.0428 - accuracy: 0.9937
Epoch 13/50
9/9 [=====] - 0s 1ms/step - loss: 0.0451 - accuracy: 0.9838
Epoch 14/50
9/9 [=====] - 0s 983us/step - loss: 0.0339 - accuracy: 0.9948
Epoch 15/50
9/9 [=====] - 0s 951us/step - loss: 0.0500 - accuracy: 0.9899
Epoch 16/50
9/9 [=====] - 0s 1ms/step - loss: 0.0255 - accuracy: 0.9954
Epoch 17/50
9/9 [=====] - 0s 1ms/step - loss: 0.0245 - accuracy: 0.9973
Epoch 18/50
9/9 [=====] - 0s 1ms/step - loss: 0.0283 - accuracy: 0.9938
Epoch 19/50
9/9 [=====] - 0s 1ms/step - loss: 0.0277 - accuracy: 0.9965
Epoch 20/50
9/9 [=====] - 0s 1ms/step - loss: 0.0475 - accuracy: 0.9908
Epoch 21/50
9/9 [=====] - 0s 1ms/step - loss: 0.0504 - accuracy: 0.9908
Epoch 22/50
9/9 [=====] - 0s 1ms/step - loss: 0.0439 - accuracy: 0.9863
Epoch 23/50
9/9 [=====] - 0s 922us/step - loss: 0.0208 - accuracy: 0.9950
Epoch 24/50
9/9 [=====] - 0s 1ms/step - loss: 0.0465 - accuracy: 0.9908
Epoch 25/50
9/9 [=====] - 0s 1ms/step - loss: 0.0243 - accuracy: 0.9939

```

```

Epoch 26/50
9/9 [=====] - 0s 1ms/step - loss: 0.0188 - accuracy: 0.9978
Epoch 27/50
9/9 [=====] - 0s 1ms/step - loss: 0.0176 - accuracy: 0.9973
Epoch 28/50
9/9 [=====] - 0s 1ms/step - loss: 0.0116 - accuracy: 0.9993
Epoch 29/50
9/9 [=====] - 0s 989us/step - loss: 0.0180 - accuracy: 0.9965
Epoch 30/50
9/9 [=====] - 0s 1ms/step - loss: 0.0111 - accuracy: 0.9993
Epoch 31/50
9/9 [=====] - 0s 1ms/step - loss: 0.0135 - accuracy: 0.9989
Epoch 32/50
9/9 [=====] - 0s 1ms/step - loss: 0.0088 - accuracy: 0.9989
Epoch 33/50
9/9 [=====] - 0s 987us/step - loss: 0.0182 - accuracy: 0.9955
Epoch 34/50
9/9 [=====] - 0s 1ms/step - loss: 0.0176 - accuracy: 0.9944
Epoch 35/50
9/9 [=====] - 0s 1ms/step - loss: 0.0193 - accuracy: 0.9955
Epoch 36/50
9/9 [=====] - 0s 1ms/step - loss: 0.0123 - accuracy: 0.9984
Epoch 37/50
9/9 [=====] - 0s 1ms/step - loss: 0.0187 - accuracy: 0.9939
Epoch 38/50
9/9 [=====] - 0s 1ms/step - loss: 0.0127 - accuracy: 0.9984
Epoch 39/50
9/9 [=====] - 0s 1ms/step - loss: 0.0061 - accuracy: 0.9979
Epoch 40/50
9/9 [=====] - 0s 2ms/step - loss: 0.0094 - accuracy: 0.9984
Epoch 41/50
9/9 [=====] - 0s 1ms/step - loss: 0.0146 - accuracy: 0.9908
Epoch 42/50
9/9 [=====] - 0s 1ms/step - loss: 0.0087 - accuracy: 0.9973
Epoch 43/50
9/9 [=====] - 0s 1ms/step - loss: 0.0136 - accuracy: 0.9908
Epoch 44/50
9/9 [=====] - 0s 1ms/step - loss: 0.0103 - accuracy: 0.9939
Epoch 45/50
9/9 [=====] - 0s 1ms/step - loss: 0.0044 - accuracy: 0.9979
Epoch 46/50
9/9 [=====] - 0s 988us/step - loss: 0.0063 - accuracy: 0.9993
Epoch 47/50
9/9 [=====] - 0s 1ms/step - loss: 0.0245 - accuracy: 0.9908
Epoch 48/50
9/9 [=====] - 0s 1ms/step - loss: 0.0053 - accuracy: 0.9965
Epoch 49/50
9/9 [=====] - 0s 1ms/step - loss: 0.0051 - accuracy: 0.9965
Epoch 50/50
9/9 [=====] - 0s 1ms/step - loss: 0.0036 - accuracy: 1.0000

```

Figura 4: Output do treinamento do modelo não normalizado

Confusion Matrix				
[[23 5]				
[ 1 42]]				
Classification Report				
	precision	recall	f1-score	support
0	0.96	0.82	0.88	28
1	0.89	0.98	0.93	43
accuracy			0.92	71
macro avg	0.93	0.90	0.91	71
weighted avg	0.92	0.92	0.91	71

Figura 5: Métricas do modelo não normalizado

O modelo em questão apresentou acurácia de 92%, o que, a priori, é bom, vide figura 5. Essa métrica indica que em 92% dos casos apresentados na fase de validação a rede foi capaz de classificar corretamente o retorno como bom ou ruim. Podemos observar, também, que a evolução da acurácia do modelo na figura 4. Percebe-se que, uma vez atingida a acurácia de 0.99 houve pouca flutuação nos valores dessa métrica, que na última época atingiu o valor de 1.00. Isso pode indicar que a quantidade de épocas não foi grande demais, pois, se fosse, veríamos uma queda maior na acurácia depois de

certa época ou crescimento do valor de *loss*. Novamente, apenas quando testarmos outros valores de época poderemos confirmar se este é um bom valor, mas, por hora, ele parece razoável.

A precisão de nosso modelo foi de 0.96 para os casos zero, isto é, "bad" e de 0.89 para 1, "good". Isso nos mostra que a rede possui um grande desempenho em classificar corretamente os retornos ruins e um desempenho um pouco pior em classificar os retornos bons. Isso é curioso uma vez que a nossa base de treino foi composta de 182 entradas 1 e 98 entradas 0, o que mostra que nossos dados são razoavelmente desbalanceados. Uma solução possível para melhorar essa métrica é construir a base de treino de outra forma, garantindo maior equilíbrio entre os retornos bons e ruins. Para a situação deste dataset, podemos argumentar que é melhor a rede classificar um retorno bom como ruim do que um retorno ruim como bom, uma vez que este segundo erro poderia causar a utilização de uma antena defeituosa, o que é mais prejudicial que não utilizar uma antena em bom funcionamento. Sendo assim, a rede indicar uma precisão maior para os retornos ruins é melhor do que o caso contrário, se ela apresentasse precisão menor para retornos ruins.

A métrica recall indica quanto a rede acertou a classificação de uma entrada da classe 0, por exemplo, em relação ao número de entradas da classe 0. É justamente o recall que nós podemos observar na matriz de confusão da figura 5. Nosso resultado indica que, em 82% dos casos em que o retorno era ruim, nosso modelo acertou que ele era ruim e, em 98% dos casos em que o retorno era bom, a rede classificou corretamente que era bom. Esta métrica usada em conjunto com a precisão é útil para avaliar o quanto preciso é nosso modelo em classificar corretamente as entradas em relação à classe correta da entrada. Como nossa precisão é maior para a classe "ruim" e o recall é menor para essa mesma classe, podemos ter maior confiança do que se ambas as métricas fossem maiores para uma classe específica, o que indicaria que nosso modelo não seria muito bom em classificar a entrada nesta classe.

Por fim, o F1-score é a média harmônica entre a Precisão e o Recall. Uma vez que já foram analisadas as métricas recall e precisão, o f1-score não fornece, a priori, nenhuma nova informação importante. No entanto, ao alterar o modelo, utilizaremos a comparação dessa métrica para identificar melhorias na qualidade das redes.

A figura abaixo indica a matriz de confusão relativa de nosso modelo. Como nós já comentamos sobre ela e sobre a métrica Recall, aqui apenas consta um resumo de sua interpretação.

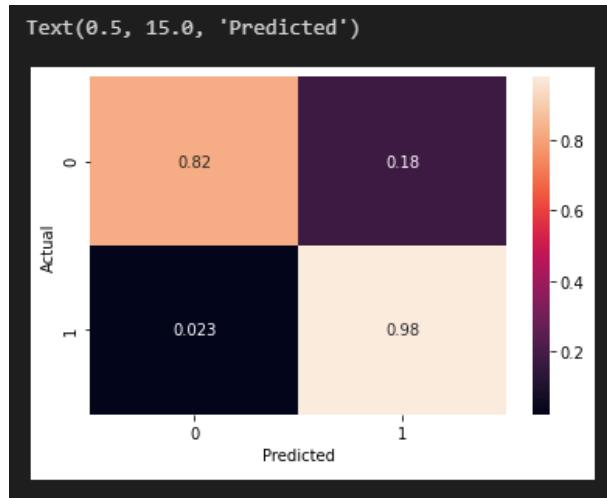


Figura 6: 82% das entradas ruins foram classificadas como ruins e 98% das boas foram classificadas como boas

### 3.2 Agora normalize os dados de entrada e treine novamente o modelo MLP. Avalie os resultados obtidos e comente o efeito da normalização no treinamento da Rede Neural.

Realizar uma normalização dos dados de entrada é uma estratégia útil para tentar aumentar tanto a eficiência computacional quanto a precisão da classificação. A normalização é importante, também, para adequar os valores aos domínios das funções de ativação que, neste caso, são a *ReLU* (Rectified Linear Activation Unit) utilizada para as camadas escondidas e a *Sigmoid* para a camada de saída.

Ao normalizar as entradas, novamente foi verificada a coluna *info\_0*, que permaneceu na base de dados na seção 2.1. Ela permanece, obviamente, inalterada, uma vez que seus valores já eram em 0 e 1. No entanto, foi modelada uma rede com e outra sem esta coluna e não foi observada grande diferença de qualidade de classificação. Sendo assim, a coluna permanece na base de dados e os resultados para a rede sem esta coluna não serão exibidos.

Além de normalizar, nenhuma outra modificação foi feita no modelo, então podemos seguir para a explanação dos resultados. As figuras a seguir exibem o output do treinamento do modelo e as métricas dos resultados de teste.

```

Epoch 1/50
9/9 [=====] - 0s 1ms/step - loss: 0.7356 - accuracy: 0.4884
Epoch 2/50
9/9 [=====] - 0s 1ms/step - loss: 0.7056 - accuracy: 0.6358
Epoch 3/50
9/9 [=====] - 0s 1ms/step - loss: 0.5348 - accuracy: 0.7287
Epoch 4/50
9/9 [=====] - 0s 970us/step - loss: 0.5210 - accuracy: 0.7662
Epoch 5/50
9/9 [=====] - 0s 1ms/step - loss: 0.5084 - accuracy: 0.7749
Epoch 6/50
9/9 [=====] - 0s 975us/step - loss: 0.4219 - accuracy: 0.8317
Epoch 7/50
9/9 [=====] - 0s 972us/step - loss: 0.3722 - accuracy: 0.8612
Epoch 8/50
9/9 [=====] - 0s 1ms/step - loss: 0.3525 - accuracy: 0.8647
Epoch 9/50
9/9 [=====] - 0s 1ms/step - loss: 0.3508 - accuracy: 0.8537
Epoch 10/50
9/9 [=====] - 0s 970us/step - loss: 0.3025 - accuracy: 0.9011
Epoch 11/50
9/9 [=====] - 0s 1ms/step - loss: 0.3187 - accuracy: 0.8750
Epoch 12/50
9/9 [=====] - 0s 1ms/step - loss: 0.2713 - accuracy: 0.9133
Epoch 13/50
9/9 [=====] - 0s 964us/step - loss: 0.2835 - accuracy: 0.8950
Epoch 14/50
9/9 [=====] - 0s 986us/step - loss: 0.2495 - accuracy: 0.9040
Epoch 15/50
9/9 [=====] - 0s 1ms/step - loss: 0.3090 - accuracy: 0.8594
Epoch 16/50
9/9 [=====] - 0s 1ms/step - loss: 0.2461 - accuracy: 0.9203
Epoch 17/50
9/9 [=====] - 0s 1ms/step - loss: 0.2091 - accuracy: 0.9373
Epoch 18/50
9/9 [=====] - 0s 2ms/step - loss: 0.2109 - accuracy: 0.9247
Epoch 19/50
9/9 [=====] - 0s 1ms/step - loss: 0.1862 - accuracy: 0.9386
Epoch 20/50
9/9 [=====] - 0s 1ms/step - loss: 0.1737 - accuracy: 0.9545
Epoch 21/50
9/9 [=====] - 0s 1ms/step - loss: 0.1542 - accuracy: 0.9391
Epoch 22/50
9/9 [=====] - 0s 1ms/step - loss: 0.1859 - accuracy: 0.9269
Epoch 23/50
9/9 [=====] - 0s 1ms/step - loss: 0.1783 - accuracy: 0.9347
Epoch 24/50
9/9 [=====] - 0s 4ms/step - loss: 0.2664 - accuracy: 0.8831
Epoch 25/50
9/9 [=====] - 0s 2ms/step - loss: 0.2407 - accuracy: 0.9104
Epoch 26/50
9/9 [=====] - 0s 1ms/step - loss: 0.1635 - accuracy: 0.9470
Epoch 27/50
9/9 [=====] - 0s 1ms/step - loss: 0.2162 - accuracy: 0.9136
Epoch 28/50
9/9 [=====] - 0s 1ms/step - loss: 0.1453 - accuracy: 0.9339
Epoch 29/50
9/9 [=====] - 0s 1ms/step - loss: 0.1772 - accuracy: 0.9435
Epoch 30/50
9/9 [=====] - 0s 1ms/step - loss: 0.2311 - accuracy: 0.9037
Epoch 31/50
9/9 [=====] - 0s 1ms/step - loss: 0.1285 - accuracy: 0.9471
Epoch 32/50
9/9 [=====] - 0s 1ms/step - loss: 0.1284 - accuracy: 0.9514
Epoch 33/50
9/9 [=====] - 0s 1ms/step - loss: 0.1558 - accuracy: 0.9508
Epoch 34/50
9/9 [=====] - 0s 2ms/step - loss: 0.1108 - accuracy: 0.9648
Epoch 35/50
9/9 [=====] - 0s 1ms/step - loss: 0.1461 - accuracy: 0.9428
Epoch 36/50
9/9 [=====] - 0s 1ms/step - loss: 0.1507 - accuracy: 0.9438
Epoch 37/50
9/9 [=====] - 0s 1ms/step - loss: 0.1401 - accuracy: 0.9533
Epoch 38/50
9/9 [=====] - 0s 1ms/step - loss: 0.1820 - accuracy: 0.9333
Epoch 39/50
9/9 [=====] - 0s 1ms/step - loss: 0.1384 - accuracy: 0.9279
Epoch 40/50
9/9 [=====] - 0s 1ms/step - loss: 0.1469 - accuracy: 0.9379
Epoch 41/50
9/9 [=====] - 0s 1ms/step - loss: 0.1300 - accuracy: 0.9694
Epoch 42/50
9/9 [=====] - 0s 1ms/step - loss: 0.1415 - accuracy: 0.9411
Epoch 43/50
9/9 [=====] - 0s 1ms/step - loss: 0.1242 - accuracy: 0.9660
Epoch 44/50
9/9 [=====] - 0s 1ms/step - loss: 0.1092 - accuracy: 0.9673
Epoch 45/50
9/9 [=====] - 0s 1ms/step - loss: 0.1104 - accuracy: 0.9682
Epoch 46/50
9/9 [=====] - 0s 1ms/step - loss: 0.0921 - accuracy: 0.9826
Epoch 47/50
9/9 [=====] - 0s 1ms/step - loss: 0.1431 - accuracy: 0.9441
Epoch 48/50
9/9 [=====] - 0s 1ms/step - loss: 0.1508 - accuracy: 0.9454
Epoch 49/50
9/9 [=====] - 0s 1ms/step - loss: 0.1170 - accuracy: 0.9711
Epoch 50/50
9/9 [=====] - 0s 1ms/step - loss: 0.1161 - accuracy: 0.9646

```

Figura 7: Output do treinamento do modelo normalizado

Confusion Matrix					
[[24 4]					
[ 0 43]]					
Classification Report					
	precision	recall	f1-score	support	
0	1.00	0.86	0.92	28	
1	0.91	1.00	0.96	43	
accuracy					0.94
macro avg					71
weighted avg					71

Figura 8: Métricas do modelo normalizado

A acurácia do modelo normalizado é maior que a do modelo não normalizado. Obtivemos um aumento de 0.02 nesta métrica, passando de 0.92 para 0.94. Como nossa acurácia já possuía um valor alto, este aumento, por si só, pode não significar grandes melhorias gerais da qualidade da rede, então é importante analisarmos, novamente, as demais métricas. Além disso, ao analisar a evolução da acurácia na figura 7, percebe-se que houve maiores flutuações do valor da acurácia e que foram necessárias mais épocas

para ela atingir pela primeira vez um valor acima de 0.9. No entanto, como todos os pesos são inicializados aleatoriamente, este fato não é de grande importância para averiguar a qualidade da rede.

Ambas as precisões das classes "ruim" e "boa" aumentaram. A precisão da classe 0 (ruim) aumentou de 0.96 para 1.0 e a classe 1 (boa) aumentou de 0.89 para 0.91. Isso quer dizer que a rede cujas entradas foram normalizadas não classificou nenhuma entrada ruim como boa e classificou apenas 0.09 entradas boas como ruins. Esse fato aumenta a confiabilidade desta rede em relação à anterior.

O recall, de modo similar à precisão, também aumentou seus valores. Para a classe "ruim" o modelo apresentou uma evolução de 0.82 para 0.86 e, para a classe "boa", de 0.98 para 1. Novamente, isso quer dizer que este modelo é mais confiável em relação ao erro de classificação positiva.

O F1-score também aumentou para ambas as classes. Para a classe "ruim" houve um aumento de 0.88 para 0.92 e, para a classe "boa", de 0.93 para 0.96. Esse resultado não é uma surpresa, pois, se ambas as métricas de precisão e recall aumentaram para ambas as classes, é claro que a média harmônica entre elas também aumentaria. Este resultado apenas confirma as conclusões anteriores: o modelo com dados normalizados de entrada mostra-se mais preciso na classificação de padrões para este dataset.

Novamente, a figura 9 a seguir exibe a Matriz de Confusão deste modelo, também explicitada na figura 8.

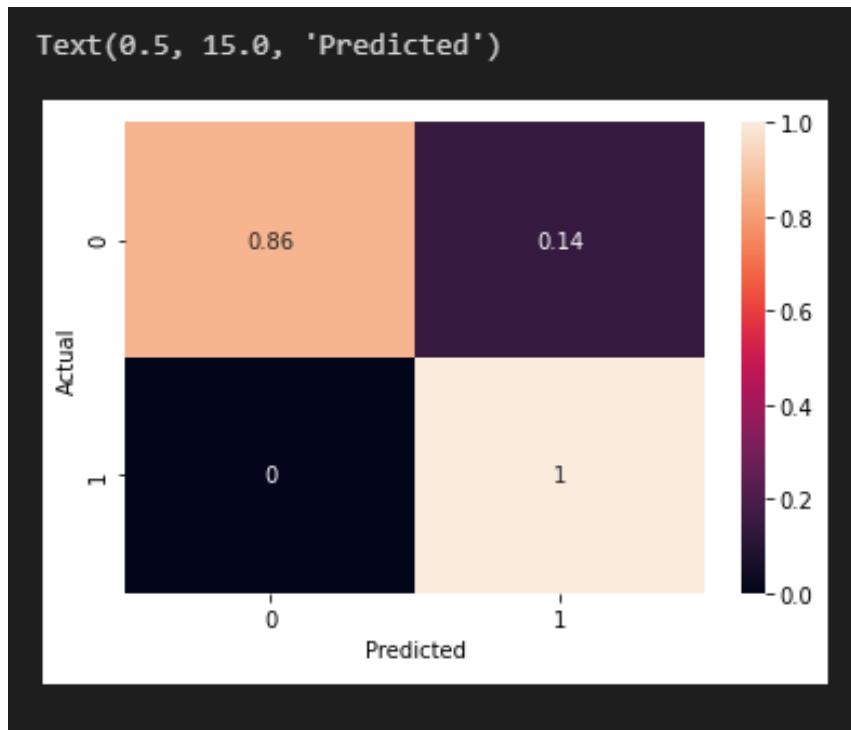


Figura 9: 86% das entradas ruins foram classificadas como ruins e 100% das boas foram classificadas como boas

## 4 Mudança de configurações do modelo

Para cada uma das seções a seguir, algum parâmetro do modelo será alterado, enquanto todos os outros serão mantidos iguais ao modelo criado na seção 3.2, modelo cujos dados

de entrada foram normalizados. O motivo de alterar apenas um parâmetro por vez é tornar mais clara a influência deste parâmetro na qualidade do modelo, uma vez que se alterássemos mais de um concomitantemente seria difícil distinguir qual mudança causou cada alteração nos resultados.

## 4.1 Insira o conjunto de validação para o treinamento do modelo. Avalie o resultado obtido.

Inserir o conjunto de validação para o treinamento do modelo significa separar parte dos dados de treino para a validação, de modo que esta parte separada não é utilizada durante o treinamento.

Ao inserir `validation_split=0.2` em nosso modelo, obtivemos os seguintes resultados exibidos nas figuras abaixo.

```
Epoch 1/50
7/7 [=====] - 0s 26ms/step - loss: 0.7512 - accuracy: 0.5216 - val_loss: 0.6458 - val_accuracy: 0.6786
Epoch 2/50
7/7 [=====] - 0s 8ms/step - loss: 0.6656 - accuracy: 0.6271 - val_loss: 0.6231 - val_accuracy: 0.6786
Epoch 3/50
7/7 [=====] - 0s 7ms/step - loss: 0.6198 - accuracy: 0.6912 - val_loss: 0.5963 - val_accuracy: 0.6786
Epoch 4/50
7/7 [=====] - 0s 7ms/step - loss: 0.6295 - accuracy: 0.6354 - val_loss: 0.6021 - val_accuracy: 0.6786
Epoch 5/50
7/7 [=====] - 0s 7ms/step - loss: 0.5999 - accuracy: 0.6718 - val_loss: 0.5620 - val_accuracy: 0.6786
Epoch 6/50
7/7 [=====] - 0s 7ms/step - loss: 0.6093 - accuracy: 0.6213 - val_loss: 0.5411 - val_accuracy: 0.7143
Epoch 7/50
7/7 [=====] - 0s 7ms/step - loss: 0.5540 - accuracy: 0.6817 - val_loss: 0.5151 - val_accuracy: 0.7679
Epoch 8/50
7/7 [=====] - 0s 7ms/step - loss: 0.5188 - accuracy: 0.7560 - val_loss: 0.5109 - val_accuracy: 0.7679
Epoch 9/50
7/7 [=====] - 0s 7ms/step - loss: 0.5233 - accuracy: 0.7593 - val_loss: 0.4727 - val_accuracy: 0.7679
Epoch 10/50
7/7 [=====] - 0s 7ms/step - loss: 0.4883 - accuracy: 0.7722 - val_loss: 0.4376 - val_accuracy: 0.8214
Epoch 11/50
7/7 [=====] - 0s 6ms/step - loss: 0.4096 - accuracy: 0.8348 - val_loss: 0.4388 - val_accuracy: 0.8758
Epoch 12/50
7/7 [=====] - 0s 7ms/step - loss: 0.3755 - accuracy: 0.8836 - val_loss: 0.4453 - val_accuracy: 0.8393
Epoch 13/50
7/7 [=====] - 0s 6ms/step - loss: 0.4146 - accuracy: 0.8527 - val_loss: 0.3867 - val_accuracy: 0.8571
Epoch 14/50
7/7 [=====] - 0s 3229 - accuracy: 0.8087 - val_loss: 0.3082 - val_accuracy: 0.8758
Epoch 15/50
7/7 [=====] - 0s 6ms/step - loss: 0.3398 - accuracy: 0.9859 - val_loss: 0.3936 - val_accuracy: 0.9107
Epoch 16/50
7/7 [=====] - 0s 7ms/step - loss: 0.3224 - accuracy: 0.9220 - val_loss: 0.3573 - val_accuracy: 0.8758
Epoch 17/50
7/7 [=====] - 0s 7ms/step - loss: 0.2696 - accuracy: 0.9444 - val_loss: 0.3605 - val_accuracy: 0.8750
Epoch 18/50
7/7 [=====] - 0s 6ms/step - loss: 0.2572 - accuracy: 0.9371 - val_loss: 0.4303 - val_accuracy: 0.8214
Epoch 19/50
7/7 [=====] - 0s 7ms/step - loss: 0.3493 - accuracy: 0.8559 - val_loss: 0.3408 - val_accuracy: 0.8758
Epoch 20/50
7/7 [=====] - 0s 11ms/step - loss: 0.2787 - accuracy: 0.8974 - val_loss: 0.3623 - val_accuracy: 0.8758
Epoch 21/50
7/7 [=====] - 0s 8ms/step - loss: 0.2870 - accuracy: 0.9168 - val_loss: 0.3428 - val_accuracy: 0.8929
Epoch 22/50
7/7 [=====] - 0s 8ms/step - loss: 0.3231 - accuracy: 0.9030 - val_loss: 0.3371 - val_accuracy: 0.8929
Epoch 23/50
7/7 [=====] - 0s 8ms/step - loss: 0.2762 - accuracy: 0.9265 - val_loss: 0.3467 - val_accuracy: 0.8929
Epoch 24/50
7/7 [=====] - 0s 8ms/step - loss: 0.2649 - accuracy: 0.9157 - val_loss: 0.3436 - val_accuracy: 0.8929
Epoch 25/50
7/7 [=====] - 0s 8ms/step - loss: 0.2783 - accuracy: 0.8849 - val_loss: 0.4408 - val_accuracy: 0.8036
Epoch 26/50
7/7 [=====] - 0s 8ms/step - loss: 0.2840 - accuracy: 0.9042 - val_loss: 0.3640 - val_accuracy: 0.8750
Epoch 27/50
7/7 [=====] - 0s 7ms/step - loss: 0.2274 - accuracy: 0.9275 - val_loss: 0.3457 - val_accuracy: 0.8750
Epoch 28/50
7/7 [=====] - 0s 8ms/step - loss: 0.2320 - accuracy: 0.9392 - val_loss: 0.3493 - val_accuracy: 0.8750
Epoch 29/50
7/7 [=====] - 0s 8ms/step - loss: 0.2719 - accuracy: 0.9119 - val_loss: 0.3298 - val_accuracy: 0.8929
Epoch 30/50
7/7 [=====] - 0s 8ms/step - loss: 0.2103 - accuracy: 0.9410 - val_loss: 0.4161 - val_accuracy: 0.8214
Epoch 31/50
7/7 [=====] - 0s 7ms/step - loss: 0.2313 - accuracy: 0.9166 - val_loss: 0.3352 - val_accuracy: 0.8750
Epoch 32/50
7/7 [=====] - 0s 7ms/step - loss: 0.2127 - accuracy: 0.9125 - val_loss: 0.3323 - val_accuracy: 0.8929
Epoch 33/50
7/7 [=====] - 0s 7ms/step - loss: 0.1964 - accuracy: 0.9475 - val_loss: 0.4029 - val_accuracy: 0.8214
Epoch 34/50
7/7 [=====] - 0s 8ms/step - loss: 0.2326 - accuracy: 0.8966 - val_loss: 0.3341 - val_accuracy: 0.8929
Epoch 35/50
7/7 [=====] - 0s 7ms/step - loss: 0.1653 - accuracy: 0.9525 - val_loss: 0.4365 - val_accuracy: 0.7857
Epoch 36/50
7/7 [=====] - 0s 8ms/step - loss: 0.2044 - accuracy: 0.9235 - val_loss: 0.3489 - val_accuracy: 0.8929
Epoch 37/50
7/7 [=====] - 0s 8ms/step - loss: 0.1796 - accuracy: 0.9424 - val_loss: 0.3407 - val_accuracy: 0.8750
Epoch 38/50
7/7 [=====] - 0s 7ms/step - loss: 0.1991 - accuracy: 0.9236 - val_loss: 0.3717 - val_accuracy: 0.8393
Epoch 39/50
7/7 [=====] - 0s 7ms/step - loss: 0.2079 - accuracy: 0.9190 - val_loss: 0.4292 - val_accuracy: 0.7857
Epoch 40/50
7/7 [=====] - 0s 8ms/step - loss: 0.2676 - accuracy: 0.8659 - val_loss: 0.3899 - val_accuracy: 0.8393
Epoch 41/50
7/7 [=====] - 0s 7ms/step - loss: 0.3041 - accuracy: 0.8753 - val_loss: 0.3689 - val_accuracy: 0.8929
Epoch 42/50
7/7 [=====] - 0s 8ms/step - loss: 0.2330 - accuracy: 0.9187 - val_loss: 0.4671 - val_accuracy: 0.8750
Epoch 43/50
7/7 [=====] - 0s 7ms/step - loss: 0.2853 - accuracy: 0.8958 - val_loss: 0.3754 - val_accuracy: 0.8393
Epoch 44/50
7/7 [=====] - 0s 7ms/step - loss: 0.2295 - accuracy: 0.9283 - val_loss: 0.4519 - val_accuracy: 0.7857
Epoch 45/50
7/7 [=====] - 0s 8ms/step - loss: 0.2894 - accuracy: 0.8729 - val_loss: 0.3355 - val_accuracy: 0.8929
Epoch 46/50
7/7 [=====] - 0s 7ms/step - loss: 0.1815 - accuracy: 0.9359 - val_loss: 0.3540 - val_accuracy: 0.8929
Epoch 47/50
7/7 [=====] - 0s 7ms/step - loss: 0.1961 - accuracy: 0.9302 - val_loss: 0.3407 - val_accuracy: 0.8750
Epoch 48/50
7/7 [=====] - 0s 7ms/step - loss: 0.2090 - accuracy: 0.9178 - val_loss: 0.3410 - val_accuracy: 0.8929
Epoch 49/50
7/7 [=====] - 0s 7ms/step - loss: 0.1593 - accuracy: 0.9516 - val_loss: 0.3600 - val_accuracy: 0.8750
Epoch 50/50
7/7 [=====] - 0s 7ms/step - loss: 0.1648 - accuracy: 0.9462 - val_loss: 0.3713 - val_accuracy: 0.8929
```

Figura 10: Retorno do treinamento com `validation_split=0.2`

Confusion Matrix				
[[ 16 12 ]]				
[ [ 0 43 ] ]				
Classification Report				
	precision	recall	f1-score	support
0	1.00	0.57	0.73	28
1	0.78	1.00	0.88	43
accuracy			0.83	71
macro avg		0.89	0.79	71
weighted avg		0.87	0.83	71

Figura 11: Métricas do teste com `validation_split=0.2`

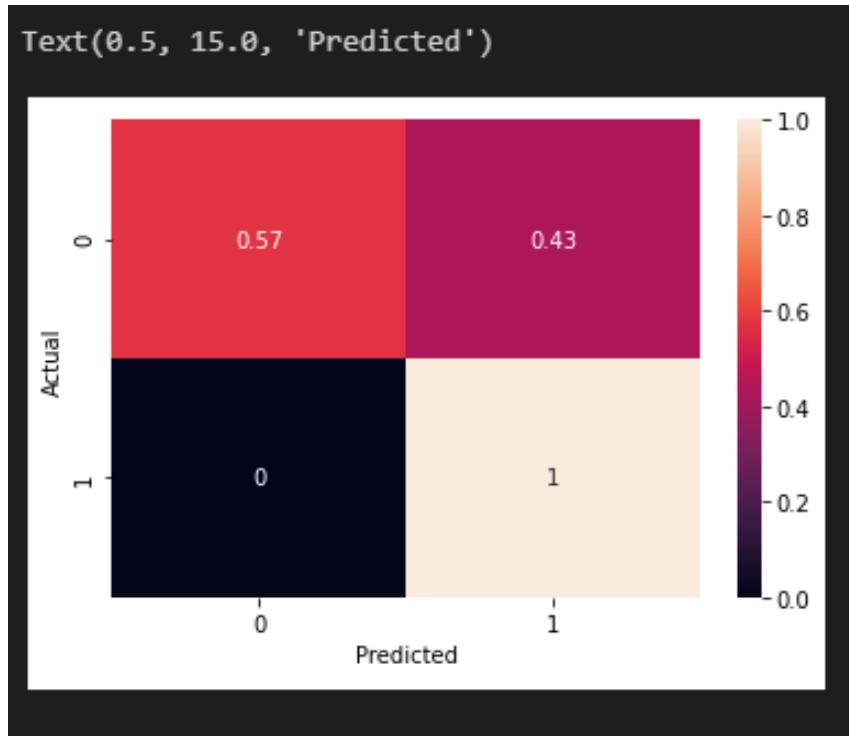


Figura 12: Matriz de Confusão

A priori, era de se esperar que separar um conjunto de dados para validação durante o treinamento melhorasse a qualidade da classificação da rede neural. No entanto, isso não parece ter acontecido.

A figura 11 nos mostra uma acurácia de teste de 84%, enquanto nossa rede original possuía acurácia de 94%. Embora este resultado não seja o esperado, tentemos buscar explicação para ele.

O conjunto de validação presente poderia ajudar a verificar a ocorrência de *overfitting*, mas a figura 10 não dá indícios que isso ocorreu. Podemos suspeitar desse fato uma vez que os valores de *loss* e *val\_loss* não apresentam perfil crescente em nenhum momento. O menor valor de *loss* durante o treinamento é, aproximadamente, 0.16 e, de *val\_loss*, 0.335. Os valores finais dessas métricas são similares aos mínimos. Logo, não parece ter ocorrido *overfitting*.

Ao analisarmos a matriz de confusão da figura 12, bem como o recall da figura 11, notamos que a rede classificou corretamente todos os padrões de teste da classe 1 (*good*), mas apenas 57% dos padrões da classe 0 (*bad*). Esta informação pode nos dar uma dica do porque desse modelo ser pior. Nossa base de dados é consideravelmente desbalanceada, com 182 padrões 1 e apenas 98 padrões 0. Sendo assim, ao retirarmos parte da base de dados do treinamento para a validação, a rede pode ter ficado com poucos padrões 0 para aprender, o que explicaria o péssimo rendimento na classificação dos padrões pertencentes a essa classe.

## 4.2 Modifique o tempo de treinamento (épocas) da Rede Neural. Escolha dois valores distintos (e.g. 1 e 1000 épocas) e avalie os resultados.

Foram escolhidos os valores de 5 e de 1000 épocas para esta comparação. Esses valores, embora quase arbitrários, possuem uma motivação de escolha. Como na seção 3.2, com 50 épocas, o resultado da rede foi bom, vamos observar o que acontece com a qualidade do modelo quando ele é pouquíssimo treinado e quando ele é super-treinado. Podemos suspeitar que veremos um *underfitting* e um *overfitting* de nosso modelo. Vejamos os resultados. Na seção 5 modelaremos mais uma rede com época próxima de 50 para confirmar essa hipótese.

### 4.2.1 5 épocas

As figuras abaixo ilustram o output do treinamento do modelo, as métricas analisadas e a matriz de confusão, respectivamente, do modelo com 5 épocas apenas.

```
model.fit(x=x_train,y=y_train,epochs=epochs)
Epoch 1/5
9/9 [=====] - 0s 1ms/step - loss: 0.7124 - accuracy: 0.4996
Epoch 2/5
9/9 [=====] - 0s 1ms/step - loss: 0.5938 - accuracy: 0.7534
Epoch 3/5
9/9 [=====] - 0s 1ms/step - loss: 0.5400 - accuracy: 0.7390
Epoch 4/5
9/9 [=====] - 0s 940us/step - loss: 0.4933 - accuracy: 0.7871
Epoch 5/5
9/9 [=====] - 0s 1ms/step - loss: 0.4648 - accuracy: 0.7795
<tensorflow.python.keras.callbacks.History at 0x7f456f72e370>
```

Figura 13: Output do treinamento do modelo

Confusion Matrix					
		0	1		
0	7	21			
	0	43			
Classification Report					
	precision	recall	f1-score	support	
0	1.00	0.25	0.40	28	
1	0.67	1.00	0.80	43	
accuracy			0.70	71	
macro avg	0.84	0.62	0.60	71	
weighted avg	0.80	0.70	0.64	71	

Figura 14: Métricas do modelo com 5 épocas

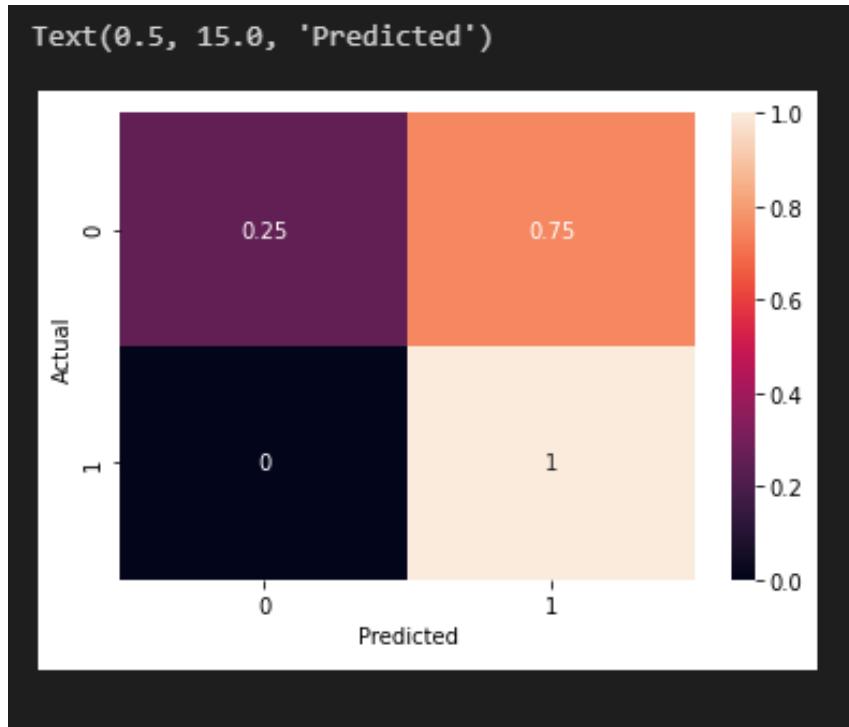


Figura 15: Matriz de Confusão do modelo com 5 épocas

Analizando a figura 13 podemos, rapidamente, perceber que a acurácia de nosso modelo ao longo do treinamento não passou dos 78%. Se analisarmos esta imagem em conjunto com a figura 7 que mostrava esta mesma informação para 50 épocas, notamos que no caso anterior precisamos de mais de 5 épocas para a acurácia atingir valores decentes.

Ao analisarmos a acurácia na figura 14 vemos que ela é muito menor do que a obtida com 50 épocas. Isso está relacionado com o que foi discutido no parágrafo anterior. Mais especificamente, podemos comentar o seguinte: nosso modelo possui 526 parâmetros treináveis e as 5 épocas não foram suficientes para treiná-los de modo a tornar esta rede satisfatoriamente especializada. Aqui percebemos um problema de *underfitting* pois nossa rede não foi capaz de aprender todos os padrões do conjunto de treinamento. As demais métricas devem corroborar com esta conclusão. Vejamos.

A precisão deste modelo é interessante. Embora nossa acurácia tenha sido ruim, a rede possui precisão de 100% para a classe "bad"(0), embora apenas 67% para a classe "good"(1).

O recall, de forma similar, apresenta 100% para a classe "good" e apenas 25% para a classe "bad". O motivo para o recall da classe 0 ser tão ruim, pode ser o fato de haver poucos padrões classificados como "bad" para o treinamento. Isso, somado com as pouquíssimas épocas, pode ter feito este modelo não aprender a classificar corretamente padrões desta classe.

O f1-score resultante foi 40% para a classe "bad" e 80% para a classe "good". Analisando apenas o comportamento da precisão e do recall deste modelo em relação ao modelo de 50 épocas, poderíamos ficar em dúvida sobre o fato dele ser pior ou melhor com base nessas métricas. No entanto, o f1-score, por ser uma análise conjunta da precisão e recall, deixa claro o pior rendimento desta rede em relação à anterior, uma vez que, para ambas as classes, o valor é menor.

Com isso, podemos argumentar fortemente que o modelo não foi treinado o suficiente e não convergiu para a função desejada.

#### 4.2.2 1000 épocas

As figuras abaixo ilustram as métricas analisadas e a matriz de confusão, respectivamente, do modelo com 1000 épocas. O output do treinamento não foi incluído por ser grande demais para este relatório, mas pode ser consultado no notebook enviado em conjunto com este documento.

Confusion Matrix					
Classification Report					
	precision	recall	f1-score	support	
0	0.95	0.71	0.82	28	
1	0.84	0.98	0.90	43	
accuracy			0.87	71	
macro avg	0.90	0.85	0.86	71	
weighted avg	0.88	0.87	0.87	71	

Figura 16: Métricas do modelo com 1000 épocas

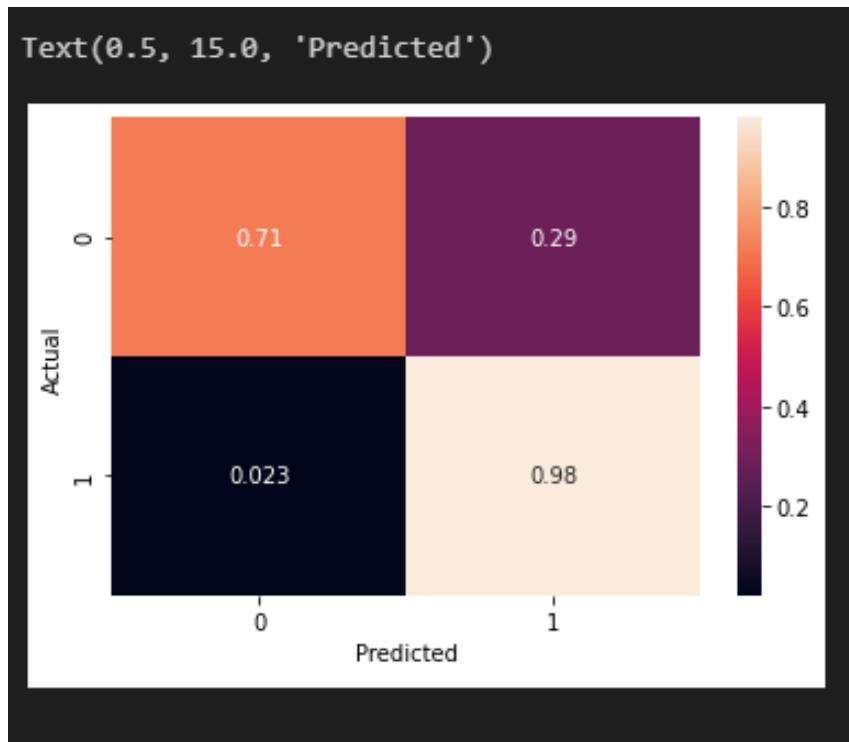


Figura 17: Matriz de Confusão do modelo com 1000 épocas

Ao analisar o output do treinamento, que não está exibido neste relatório, percebemos pouca diferença na acurácia e na função de perda do modelo, em relação à 50 épocas. Vale comentar apenas que os valores da acurácia deste output são todos muito bons, com a grande maioria acima de 95%. No entanto, veremos que isso não implica, necessariamente, em uma rede adequada.

Ao analisar a acurácia geral deste modelo notamos uma queda, uma vez que seu valor é de 87%. Essa diferença entre a acurácia geral e acurácia durante o treinamento pode ser sinal de uma superespecialização da rede, que praticamente ”decorou” os padrões de treino e criou uma função excessivamente flexível, que perde generalização. Como temos apenas 526 pesos, estabelecer 1000 épocas parece se mostrar exagerado para o treinamento. Sendo assim, considera-se inconclusiva a existência ou não de *overfitting*.

Ao analisarmos as demais métricas exibidas na figura 17 e comparar com as métricas da rede com 50 épocas percebemos uma piora em todos os valores. No entanto, a evolução da métrica *loss* não revela nenhum padrão crescente, permanecendo sempre abaixo de 0,1, o que corrobora para a argumentação que esta rede, embora treinada demais, não sofre um *overfitting*.

### **4.3 Modifique a taxa de aprendizado da Rede Neural. Escolha dois valores distintos (e.g. 0,001 e 0,1) e avalie os resultados.**

A taxa de aprendizado de uma Rede Neural está relacionada com a velocidade na qual a rede treina, com o problema dos mínimos locais e com possíveis oscilações no treinamento da rede. Caso a taxa seja muito pequena, nosso treinamento será mais lento e corremos maiores riscos de convergir para um mínimo local, ao invés de atingir o mínimo global da função de erro do modelo. Isso, obviamente, é indesejado, pois queremos o menor erro possível para a rede modelada. Caso a taxa seja grande demais, por outro lado, corremos o risco de ficar oscilando no entorno de um mínimo da função de erro, uma vez que os ”saltos” de nossa rede são grandes demais. Veremos como esses conceitos aparecem na análise da rede neural com taxas de aprendizado 0.001 e 0.4, lembrando que, originalmente, a taxa valia 0.05. Novamente, esses valores foram escolhidos por serem muito menor e muito maior que o valor original, para o qual a rede demonstrou bom desempenho.

#### **4.3.1 Taxa de aprendizado = 0.001**

As figuras abaixo ilustram o retorno do treinamento da rede, as métricas analisadas e a matriz de confusão, respectivamente, do modelo com taxa de aprendizado igual a 0.001.

```

Epoch 1/50
9/9 [=====] - 0s 1ms/step - loss: 0.8016 - accuracy: 0.3492
Epoch 2/50
9/9 [=====] - 0s 1ms/step - loss: 0.7315 - accuracy: 0.3746
Epoch 3/50
9/9 [=====] - 0s 1ms/step - loss: 0.6572 - accuracy: 0.6885
Epoch 4/50
9/9 [=====] - 0s 1ms/step - loss: 0.6300 - accuracy: 0.6485
Epoch 5/50
9/9 [=====] - 0s 1ms/step - loss: 0.5849 - accuracy: 0.7095
Epoch 6/50
9/9 [=====] - 0s 1ms/step - loss: 0.5885 - accuracy: 0.6703
Epoch 7/50
9/9 [=====] - 0s 1ms/step - loss: 0.5507 - accuracy: 0.7097
Epoch 8/50
9/9 [=====] - 0s 1ms/step - loss: 0.5704 - accuracy: 0.6951
Epoch 9/50
9/9 [=====] - 0s 1ms/step - loss: 0.5503 - accuracy: 0.7094
Epoch 10/50
9/9 [=====] - 0s 1ms/step - loss: 0.5831 - accuracy: 0.6524
Epoch 11/50
9/9 [=====] - 0s 1ms/step - loss: 0.5585 - accuracy: 0.6792
Epoch 12/50
9/9 [=====] - 0s 990us/step - loss: 0.5573 - accuracy: 0.6929
Epoch 13/50
9/9 [=====] - 0s 1ms/step - loss: 0.5381 - accuracy: 0.7222
Epoch 14/50
9/9 [=====] - 0s 1ms/step - loss: 0.5587 - accuracy: 0.6799
Epoch 15/50
9/9 [=====] - 0s 1ms/step - loss: 0.5398 - accuracy: 0.6998
Epoch 16/50
9/9 [=====] - 0s 1ms/step - loss: 0.5431 - accuracy: 0.6875
Epoch 17/50
9/9 [=====] - 0s 1ms/step - loss: 0.5278 - accuracy: 0.7100
Epoch 18/50
9/9 [=====] - 0s 1ms/step - loss: 0.5039 - accuracy: 0.7701
Epoch 19/50
9/9 [=====] - 0s 2ms/step - loss: 0.5043 - accuracy: 0.7446
Epoch 20/50
9/9 [=====] - 0s 2ms/step - loss: 0.5052 - accuracy: 0.7380
Epoch 21/50
9/9 [=====] - 0s 1ms/step - loss: 0.4941 - accuracy: 0.7792
Epoch 22/50
9/9 [=====] - 0s 1ms/step - loss: 0.4974 - accuracy: 0.7889
Epoch 23/50
9/9 [=====] - 0s 1ms/step - loss: 0.4833 - accuracy: 0.8015
Epoch 24/50
9/9 [=====] - 0s 1ms/step - loss: 0.4904 - accuracy: 0.7900
Epoch 25/50
9/9 [=====] - 0s 991us/step - loss: 0.4743 - accuracy: 0.8057

```

```

Epoch 26/50
9/9 [=====] - 0s 1ms/step - loss: 0.4676 - accuracy: 0.8121
Epoch 27/50
9/9 [=====] - 0s 1ms/step - loss: 0.4544 - accuracy: 0.8404
Epoch 28/50
9/9 [=====] - 0s 1ms/step - loss: 0.4706 - accuracy: 0.8210
Epoch 29/50
9/9 [=====] - 0s 1ms/step - loss: 0.4719 - accuracy: 0.8038
Epoch 30/50
9/9 [=====] - 0s 1ms/step - loss: 0.4785 - accuracy: 0.8022
Epoch 31/50
9/9 [=====] - 0s 1ms/step - loss: 0.4509 - accuracy: 0.8181
Epoch 32/50
9/9 [=====] - 0s 1ms/step - loss: 0.4454 - accuracy: 0.8393
Epoch 33/50
9/9 [=====] - 0s 1ms/step - loss: 0.4300 - accuracy: 0.8427
Epoch 34/50
9/9 [=====] - 0s 1ms/step - loss: 0.4488 - accuracy: 0.8172
Epoch 35/50
9/9 [=====] - 0s 1ms/step - loss: 0.4361 - accuracy: 0.8298
Epoch 36/50
9/9 [=====] - 0s 1ms/step - loss: 0.4247 - accuracy: 0.8422
Epoch 37/50
9/9 [=====] - 0s 982us/step - loss: 0.4075 - accuracy: 0.8434
Epoch 38/50
9/9 [=====] - 0s 1ms/step - loss: 0.4424 - accuracy: 0.8238
Epoch 39/50
9/9 [=====] - 0s 1ms/step - loss: 0.4116 - accuracy: 0.8302
Epoch 40/50
9/9 [=====] - 0s 1ms/step - loss: 0.4245 - accuracy: 0.8168
Epoch 41/50
9/9 [=====] - 0s 1ms/step - loss: 0.4365 - accuracy: 0.8124
Epoch 42/50
9/9 [=====] - 0s 1ms/step - loss: 0.4062 - accuracy: 0.8456
Epoch 43/50
9/9 [=====] - 0s 1ms/step - loss: 0.3967 - accuracy: 0.8480
Epoch 44/50
9/9 [=====] - 0s 1ms/step - loss: 0.4002 - accuracy: 0.8551
Epoch 45/50
9/9 [=====] - 0s 1ms/step - loss: 0.4126 - accuracy: 0.8300
Epoch 46/50
9/9 [=====] - 0s 1ms/step - loss: 0.4228 - accuracy: 0.8144
Epoch 47/50
9/9 [=====] - 0s 1ms/step - loss: 0.4292 - accuracy: 0.8048
Epoch 48/50
9/9 [=====] - 0s 1ms/step - loss: 0.3805 - accuracy: 0.8602
Epoch 49/50
9/9 [=====] - 0s 1ms/step - loss: 0.3866 - accuracy: 0.8587
Epoch 50/50
9/9 [=====] - 0s 993us/step - loss: 0.3881 - accuracy: 0.8546

```

Figura 18: Treinamento da rede neural com taxa de aprendizado 0.001

Confusion Matrix				
[[15 13]				
[ 0 43]]				
Classification Report				
	precision	recall	f1-score	support
0	1.00	0.54	0.70	28
1	0.77	1.00	0.87	43
accuracy				0.82
macro avg				0.78
weighted avg				0.80
				71

Figura 19: Métricas para taxa de aprendizado 0.001

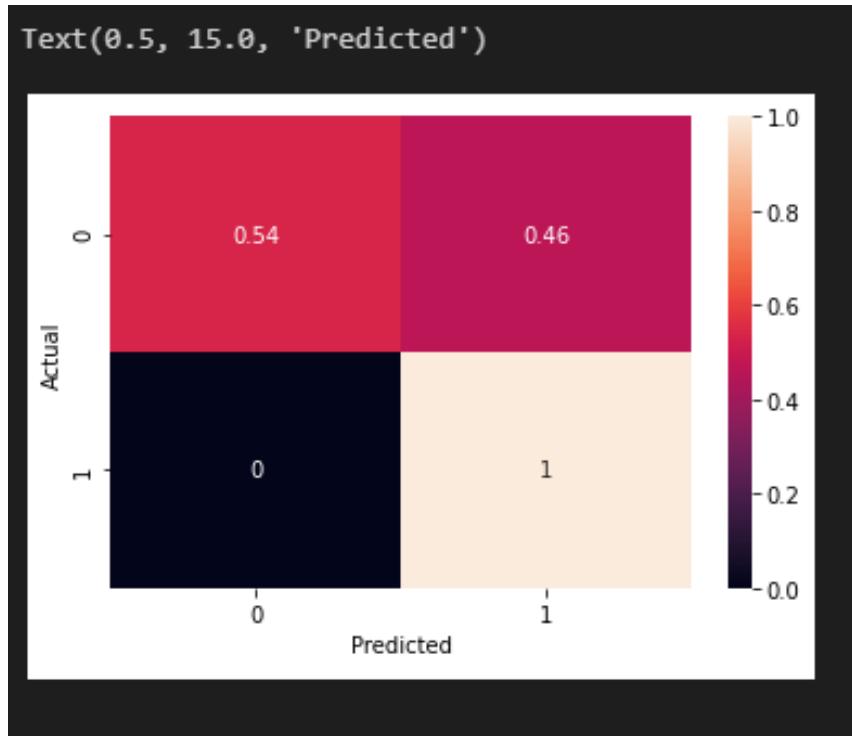


Figura 20: Matriz de Confusão para taxa de aprendizado 0.001

Ao analisar a figura 18, uma informação que se destaca é o campo *loss*. Este campo é o erro que queremos minimizar com o treinamento de rede. Com isso em mente, note que os valores de *loss* para esta taxa de aprendizado estão consideravelmente maiores em relação ao modelo com taxa 0.05. Naquele a perda era minimizada até 0.11, enquanto aqui ela pouca abaixa de 0.4. Esse fato é um grande indicador que a taxa escolhida pode ter sido pequena demais, o que revela que talvez a rede tenha convergido para um mínimo local da função de erro. A acurácia exibida nesta mesma figura também é pior que àquela apresentada com a taxa original, e a acurácia dos testes, exibida na figura 19 é de 82%, menor que a original.

A precisão para a classe "bad" e o recall para a classe "good" apresentam os mesmos valores para a taxa de aprendizado original, ambos 1.0. Entretanto, a precisão da classe "good" e o "recall" da classe "bad" são ambos piores que os originais. Mais uma vez, utilizemos o f1-score para averiguar que a média das métricas anteriores é pior, para ambas as classes, em relação ao modelo original. Essas análises, somadas com o que foi comentado sobre a perda grande do modelo, atestam para o fato de que esta rede é menos adequada que a original.

#### 4.3.2 Taxa de aprendizado = 0.4

As figuras abaixo ilustram o retorno do treinamento da rede, as métricas analisadas e a matriz de confusão, respectivamente, do modelo com taxa de aprendizado igual a 0.4.

```

Epoch 1/50
9/9 [=====] - 0s 983us/step - loss: 4.5375 - accuracy: 0.5761
Epoch 2/50
9/9 [=====] - 0s 977us/step - loss: 0.9732 - accuracy: 0.3505
Epoch 3/50
9/9 [=====] - 0s 1ms/step - loss: 0.6682 - accuracy: 0.6335
Epoch 4/50
9/9 [=====] - 0s 2ms/step - loss: 0.7030 - accuracy: 0.6458
Epoch 5/50
9/9 [=====] - 0s 4ms/step - loss: 0.6280 - accuracy: 0.6764
Epoch 6/50
9/9 [=====] - 0s 2ms/step - loss: 0.6503 - accuracy: 0.6633
Epoch 7/50
9/9 [=====] - 0s 916us/step - loss: 0.6404 - accuracy: 0.6741
Epoch 8/50
9/9 [=====] - 0s 1ms/step - loss: 0.6552 - accuracy: 0.6412
Epoch 9/50
9/9 [=====] - 0s 1ms/step - loss: 0.6740 - accuracy: 0.6121
Epoch 10/50
9/9 [=====] - 0s 3ms/step - loss: 0.6538 - accuracy: 0.6423
Epoch 11/50
9/9 [=====] - 0s 2ms/step - loss: 0.6242 - accuracy: 0.6886
Epoch 12/50
9/9 [=====] - 0s 1ms/step - loss: 0.6514 - accuracy: 0.6489
Epoch 13/50
9/9 [=====] - 0s 1ms/step - loss: 0.6475 - accuracy: 0.6597
Epoch 14/50
9/9 [=====] - 0s 1ms/step - loss: 0.6213 - accuracy: 0.6893
Epoch 15/50
9/9 [=====] - 0s 1ms/step - loss: 0.6639 - accuracy: 0.6317
Epoch 16/50
9/9 [=====] - 0s 1ms/step - loss: 0.6414 - accuracy: 0.6658
Epoch 17/50
9/9 [=====] - 0s 1ms/step - loss: 0.6584 - accuracy: 0.6401
Epoch 18/50
9/9 [=====] - 0s 1ms/step - loss: 0.6464 - accuracy: 0.6550
Epoch 19/50
9/9 [=====] - 0s 1ms/step - loss: 0.6592 - accuracy: 0.6337
Epoch 20/50
9/9 [=====] - 0s 1ms/step - loss: 0.6345 - accuracy: 0.6748
Epoch 21/50
9/9 [=====] - 0s 1ms/step - loss: 0.6588 - accuracy: 0.6339
Epoch 22/50
9/9 [=====] - 0s 1ms/step - loss: 0.6342 - accuracy: 0.6789
Epoch 23/50
9/9 [=====] - 0s 1ms/step - loss: 0.6584 - accuracy: 0.6412
Epoch 24/50
9/9 [=====] - 0s 1ms/step - loss: 0.6363 - accuracy: 0.6861
Epoch 25/50
9/9 [=====] - 0s 1ms/step - loss: 0.6525 - accuracy: 0.6436
Epoch 26/50
9/9 [=====] - 0s 1ms/step - loss: 0.6529 - accuracy: 0.6426
Epoch 27/50
9/9 [=====] - 0s 1ms/step - loss: 0.6576 - accuracy: 0.6392
Epoch 28/50
9/9 [=====] - 0s 1ms/step - loss: 0.6554 - accuracy: 0.6368
Epoch 29/50
9/9 [=====] - 0s 1ms/step - loss: 0.6450 - accuracy: 0.6574
Epoch 30/50
9/9 [=====] - 0s 1ms/step - loss: 0.6520 - accuracy: 0.6474
Epoch 31/50
9/9 [=====] - 0s 1ms/step - loss: 0.6545 - accuracy: 0.6381
Epoch 32/50
9/9 [=====] - 0s 1ms/step - loss: 0.6527 - accuracy: 0.6431
Epoch 33/50
9/9 [=====] - 0s 1ms/step - loss: 0.6523 - accuracy: 0.6493
Epoch 34/50
9/9 [=====] - 0s 1ms/step - loss: 0.6491 - accuracy: 0.6499
Epoch 35/50
9/9 [=====] - 0s 1ms/step - loss: 0.6508 - accuracy: 0.6472
Epoch 36/50
9/9 [=====] - 0s 1ms/step - loss: 0.6285 - accuracy: 0.6880
Epoch 37/50
9/9 [=====] - 0s 1ms/step - loss: 0.6378 - accuracy: 0.6697
Epoch 38/50
9/9 [=====] - 0s 1ms/step - loss: 0.6397 - accuracy: 0.6673
Epoch 39/50
9/9 [=====] - 0s 1ms/step - loss: 0.6656 - accuracy: 0.6287
Epoch 40/50
9/9 [=====] - 0s 1ms/step - loss: 0.6561 - accuracy: 0.6387
Epoch 41/50
9/9 [=====] - 0s 1ms/step - loss: 0.6362 - accuracy: 0.6723
Epoch 42/50
9/9 [=====] - 0s 1ms/step - loss: 0.6582 - accuracy: 0.6363
Epoch 43/50
9/9 [=====] - 0s 1ms/step - loss: 0.6578 - accuracy: 0.6294
Epoch 44/50
9/9 [=====] - 0s 1ms/step - loss: 0.6693 - accuracy: 0.6323
Epoch 45/50
9/9 [=====] - 0s 1ms/step - loss: 0.6708 - accuracy: 0.6121
Epoch 46/50
9/9 [=====] - 0s 1ms/step - loss: 0.6503 - accuracy: 0.6537
Epoch 47/50
9/9 [=====] - 0s 1ms/step - loss: 0.6133 - accuracy: 0.7019
Epoch 48/50
9/9 [=====] - 0s 1ms/step - loss: 0.6455 - accuracy: 0.6661
Epoch 49/50
9/9 [=====] - 0s 1ms/step - loss: 0.6553 - accuracy: 0.6400
Epoch 50/50
9/9 [=====] - 0s 1ms/step - loss: 0.6351 - accuracy: 0.6791

```

Figura 21: Treinamento da rede neural com taxa de aprendizado 0.4

Confusion Matrix				
[[ 0 28]				
[ 0 43]]				
Classification Report				
	precision	recall	f1-score	support
0	0.00	0.00	0.00	28
1	0.61	1.00	0.75	43
accuracy				0.61
macro avg				71
weighted avg				71

Figura 22: Métricas para taxa de aprendizado 0.4

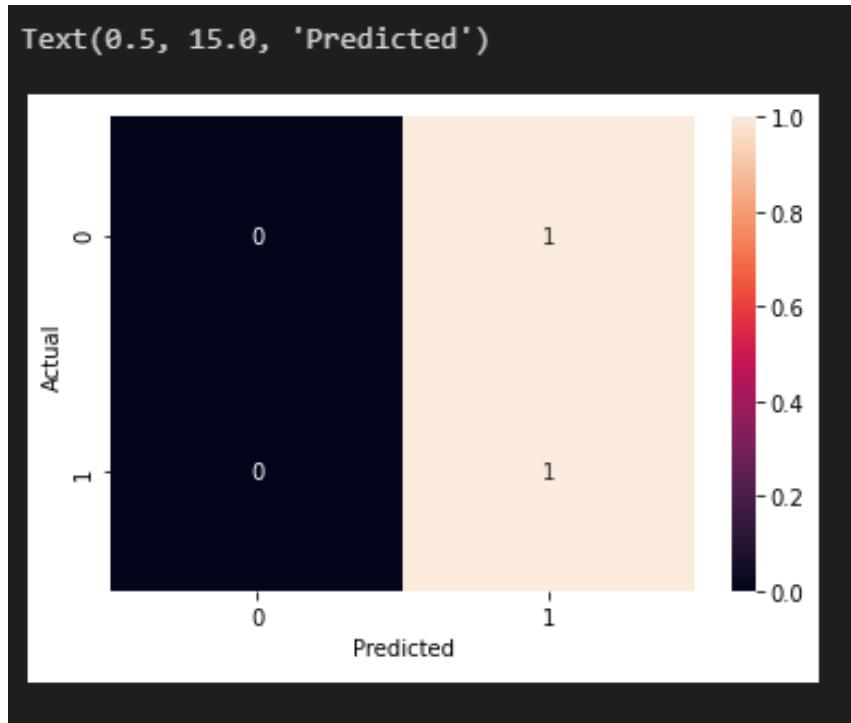


Figura 23: Matriz de Confusão para taxa de aprendizado 0.4

Vamos, novamente, começar analisando a perda do treinamento. Note que, logo a partir da quinta época, o valor de *loss* parece ficar sempre próximo a 0.64, indo de aproximadamente 0.62 até 0.67. Isso mostra, justamente, um perfil de oscilação da minimização do valor do erro da rede. Há uma diferença entre esta oscilação visível na figura 21 e a convergência para um mínimo local apresentada na figura 18. Lá os valores de erro decresceram ao longo do treinamento e convergiram para um valor ruim. Aqui não há minimização, ou, pelo menos, há pouca, o que indica a oscilação já comentada. Esse fato é indicador de que a nossa taxa de aprendizagem, 0.4, é grande demais. As demais métricas, provavelmente, darão outros indícios da má qualidade do modelo.

A acurácia do teste, exibida na figura 22 foi a pior de todas até agora, 0.61. Além disso, a figura 23 nos mostra que todos os dados de teste da classe 0 foram erroneamente classificados como 1 e todos os dados de teste da classe 1 foram corretamente classificados. Estes valores de precisão, recall e f1-score nos mostram que este rede não é confiável para fazer a classificação que buscamos.

#### 4.4 Modifique a quantidade de neurônios na camada escondida da Rede Neural. Escolha dois valores distintos (e.g. 2 e 70 neurônios) e avalie os resultados.

O número de neurônios da camada escondida está diretamente relacionado com a quantidade de pesos que nosso modelo deverá treinar. Como cada neurônio está conectado com todas as entradas e com a saída, e cada conexão possui um peso, além dos *bias* de cada neurônio, uma rede neural com muitos neurônios na camada escondida pode se tornar custosa e complexa desnecessariamente. Foram escolhidos os valores 5 e 50 por, mais uma vez, serem bem menor e maior que o original.

#### 4.4.1 5 neurônios

As figuras abaixo ilustram o sumário da rede, o retorno do treinamento da rede, as métricas analisadas e a matriz de confusão, respectivamente, do modelo com 5 neurônios na camada escondida. A figura 24 mostra que, neste caso, temos 176 parâmetros para treinar.

Model: "sequential_10"		
Layer (type)	Output Shape	Param #
dense_20 (Dense)	(None, 5)	170
dense_21 (Dense)	(None, 1)	6
<hr/>		
Total params: 176		
Trainable params: 176		
Non-trainable params: 0		
<hr/>		

Figura 24: Sumário do modelo com 5 neurônios na camada escondida

```

Epoch 1/50
9/9 [=====] - 0s 1ms/step - loss: 0.6654 - accuracy: 0.6012
Epoch 2/50
9/9 [=====] - 0s 1ms/step - loss: 0.5953 - accuracy: 0.7037
Epoch 3/50
9/9 [=====] - 0s 1ms/step - loss: 0.5558 - accuracy: 0.7405
Epoch 4/50
9/9 [=====] - 0s 1ms/step - loss: 0.4947 - accuracy: 0.7853
Epoch 5/50
9/9 [=====] - 0s 969us/step - loss: 0.4332 - accuracy: 0.8268
Epoch 6/50
9/9 [=====] - 0s 998us/step - loss: 0.3575 - accuracy: 0.8612
Epoch 7/50
9/9 [=====] - 0s 1ms/step - loss: 0.3204 - accuracy: 0.8594
Epoch 8/50
9/9 [=====] - 0s 1ms/step - loss: 0.3032 - accuracy: 0.8784
Epoch 9/50
9/9 [=====] - 0s 1ms/step - loss: 0.4096 - accuracy: 0.8100
Epoch 10/50
9/9 [=====] - 0s 1ms/step - loss: 0.3474 - accuracy: 0.8652
Epoch 11/50
9/9 [=====] - 0s 1ms/step - loss: 0.3137 - accuracy: 0.8627
Epoch 12/50
9/9 [=====] - 0s 1ms/step - loss: 0.3000 - accuracy: 0.8909
Epoch 13/50
9/9 [=====] - 0s 1ms/step - loss: 0.2689 - accuracy: 0.9273
Epoch 14/50
9/9 [=====] - 0s 1ms/step - loss: 0.2606 - accuracy: 0.9133
Epoch 15/50
9/9 [=====] - 0s 1ms/step - loss: 0.2638 - accuracy: 0.8945
Epoch 16/50
9/9 [=====] - 0s 1ms/step - loss: 0.2681 - accuracy: 0.8871
Epoch 17/50
9/9 [=====] - 0s 1ms/step - loss: 0.2233 - accuracy: 0.9305
Epoch 18/50
9/9 [=====] - 0s 1ms/step - loss: 0.2362 - accuracy: 0.9010
Epoch 19/50
9/9 [=====] - 0s 1ms/step - loss: 0.2081 - accuracy: 0.9241
Epoch 20/50
9/9 [=====] - 0s 2ms/step - loss: 0.2106 - accuracy: 0.9192
Epoch 21/50
9/9 [=====] - 0s 1ms/step - loss: 0.1966 - accuracy: 0.9247
Epoch 22/50
9/9 [=====] - 0s 1ms/step - loss: 0.2045 - accuracy: 0.9202
Epoch 23/50
9/9 [=====] - 0s 1ms/step - loss: 0.2049 - accuracy: 0.9167
Epoch 24/50
9/9 [=====] - 0s 1ms/step - loss: 0.1669 - accuracy: 0.9440
Epoch 25/50
9/9 [=====] - 0s 1ms/step - loss: 0.1845 - accuracy: 0.9353
Epoch 26/50
9/9 [=====] - 0s 1ms/step - loss: 0.2184 - accuracy: 0.9308
Epoch 27/50
9/9 [=====] - 0s 1ms/step - loss: 0.2444 - accuracy: 0.9032
Epoch 28/50
9/9 [=====] - 0s 1ms/step - loss: 0.3351 - accuracy: 0.8638
Epoch 29/50
9/9 [=====] - 0s 1ms/step - loss: 0.2184 - accuracy: 0.9211
Epoch 30/50
9/9 [=====] - 0s 1ms/step - loss: 0.2367 - accuracy: 0.8915
Epoch 31/50
9/9 [=====] - 0s 2ms/step - loss: 0.1728 - accuracy: 0.9430
Epoch 32/50
9/9 [=====] - 0s 1ms/step - loss: 0.2427 - accuracy: 0.9046
Epoch 33/50
9/9 [=====] - 0s 940us/step - loss: 0.2516 - accuracy: 0.8790
Epoch 34/50
9/9 [=====] - 0s 1ms/step - loss: 0.2346 - accuracy: 0.9136
Epoch 35/50
9/9 [=====] - 0s 1ms/step - loss: 0.2031 - accuracy: 0.9380
Epoch 36/50
9/9 [=====] - 0s 1ms/step - loss: 0.1493 - accuracy: 0.9480
Epoch 37/50
9/9 [=====] - 0s 979us/step - loss: 0.1744 - accuracy: 0.9417
Epoch 38/50
9/9 [=====] - 0s 1ms/step - loss: 0.1383 - accuracy: 0.9687
Epoch 39/50
9/9 [=====] - 0s 1ms/step - loss: 0.1631 - accuracy: 0.9465
Epoch 40/50
9/9 [=====] - 0s 1ms/step - loss: 0.1687 - accuracy: 0.9505
Epoch 41/50
9/9 [=====] - 0s 964us/step - loss: 0.1525 - accuracy: 0.9535
Epoch 42/50
9/9 [=====] - 0s 1ms/step - loss: 0.1782 - accuracy: 0.9416
Epoch 43/50
9/9 [=====] - 0s 979us/step - loss: 0.1284 - accuracy: 0.9661
Epoch 44/50
9/9 [=====] - 0s 1ms/step - loss: 0.1502 - accuracy: 0.9614
Epoch 45/50
9/9 [=====] - 0s 1ms/step - loss: 0.1612 - accuracy: 0.9452
Epoch 46/50
9/9 [=====] - 0s 1ms/step - loss: 0.1706 - accuracy: 0.9467
Epoch 47/50
9/9 [=====] - 0s 1ms/step - loss: 0.2205 - accuracy: 0.9068
Epoch 48/50
9/9 [=====] - 0s 1ms/step - loss: 0.1531 - accuracy: 0.9526
Epoch 49/50
9/9 [=====] - 0s 1ms/step - loss: 0.1554 - accuracy: 0.9486
Epoch 50/50
9/9 [=====] - 0s 1ms/step - loss: 0.1375 - accuracy: 0.9555

```

Figura 25: Treinamento do modelo com 5 neurônios na camada escondida

Confusion Matrix				
[[20  8]				
[ 0 43]]				
Classification Report				
	precision	recall	f1-score	support
0	1.00	0.71	0.83	28
1	0.84	1.00	0.91	43
accuracy			0.89	71
macro avg	0.92	0.86	0.87	71
weighted avg	0.90	0.89	0.88	71

Figura 26: Métricas do modelo com 5 neurônios na camada escondida

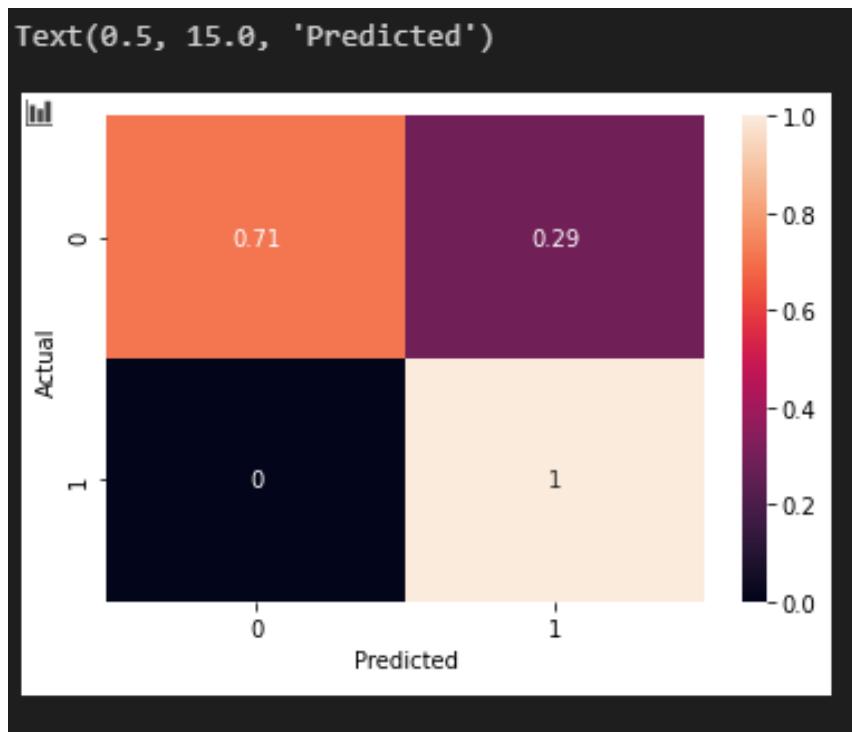


Figura 27: Matriz de Confusão do modelo com 5 neurônios na camada escondida

Ao analisarmos a evolução da acurácia e da perda na figura 25 não percebemos grandes diferenças em relação ao modelo original. Isso pode indicar que ambas as quantidades de neurônios são razoáveis para este problema de classificação. Analisemos as demais métricas para averiguar se este fato é verídico.

A acurácia dos testes, 89% foi menor que a apresentada pela rede original. A precisão para a classe 0 e o recall para a classe 1 foram iguais aos do modelo original, enquanto que a precisão da classe 1 e o recall da classe 0 foram um pouco menores. Com base nisso, não é de espantar que o f1-score é um pouco menor para ambas as classes. Todos esses valores estão apresentados na figura 26. Sendo assim, em comparação à rede com 15 neurônios, esta rede com 5 não apresentou uma perda enorme em qualidade, apesar de mostrar-se pior. Vejamos o que ocorre ao aumentar o número de neurônios da camada escondida.

#### 4.4.2 50 neurônios

As figuras abaixo ilustram o sumário da rede, o retorno do treinamento da rede, as métricas analisadas e a matriz de confusão, respectivamente, do modelo com 50 neurônios na camada escondida. A figura 28 mostra que, neste caso, temos 1751 parâmetros para treinar.

Model: "sequential_14"		
Layer (type)	Output Shape	Param #
dense_28 (Dense)	(None, 50)	1700
dense_29 (Dense)	(None, 1)	51
<b>Total params:</b>	<b>1,751</b>	
<b>Trainable params:</b>	<b>1,751</b>	
<b>Non-trainable params:</b>	<b>0</b>	

Figura 28: Sumário do modelo com 50 neurônios na camada escondida

```

Epoch 1/50
9/9 [=====] - 0s 1ms/step - loss: 0.8027 - accuracy: 0.5366
Epoch 2/50
9/9 [=====] - 0s 1ms/step - loss: 0.6984 - accuracy: 0.5875
Epoch 3/50
9/9 [=====] - 0s 1ms/step - loss: 0.5357 - accuracy: 0.7142
Epoch 4/50
9/9 [=====] - 0s 1ms/step - loss: 0.4592 - accuracy: 0.7983
Epoch 5/50
9/9 [=====] - 0s 1ms/step - loss: 0.4493 - accuracy: 0.7957
Epoch 6/50
9/9 [=====] - 0s 1ms/step - loss: 0.4462 - accuracy: 0.7880
Epoch 7/50
9/9 [=====] - 0s 1ms/step - loss: 0.4371 - accuracy: 0.7975
Epoch 8/50
9/9 [=====] - 0s 1ms/step - loss: 0.3939 - accuracy: 0.8163
Epoch 9/50
9/9 [=====] - 0s 1ms/step - loss: 0.3651 - accuracy: 0.8434
Epoch 10/50
9/9 [=====] - 0s 1ms/step - loss: 0.3164 - accuracy: 0.9011
Epoch 11/50
9/9 [=====] - 0s 1ms/step - loss: 0.3003 - accuracy: 0.9061
Epoch 12/50
9/9 [=====] - 0s 1ms/step - loss: 0.2778 - accuracy: 0.9064
Epoch 13/50
9/9 [=====] - 0s 1ms/step - loss: 0.2758 - accuracy: 0.9077
Epoch 14/50
9/9 [=====] - 0s 990us/step - loss: 0.2867 - accuracy: 0.8653
Epoch 15/50
9/9 [=====] - 0s 1ms/step - loss: 0.2591 - accuracy: 0.9249
Epoch 16/50
9/9 [=====] - 0s 1ms/step - loss: 0.2364 - accuracy: 0.9159
Epoch 17/50
9/9 [=====] - 0s 1ms/step - loss: 0.2204 - accuracy: 0.9231
Epoch 18/50
9/9 [=====] - 0s 1ms/step - loss: 0.3010 - accuracy: 0.8890
Epoch 19/50
9/9 [=====] - 0s 1ms/step - loss: 0.2409 - accuracy: 0.9234
Epoch 20/50
9/9 [=====] - 0s 1ms/step - loss: 0.2216 - accuracy: 0.9261
Epoch 21/50
9/9 [=====] - 0s 1ms/step - loss: 0.3106 - accuracy: 0.8666
Epoch 22/50
9/9 [=====] - 0s 1ms/step - loss: 0.4030 - accuracy: 0.8177
Epoch 23/50
9/9 [=====] - 0s 1ms/step - loss: 0.3621 - accuracy: 0.8348
Epoch 24/50
9/9 [=====] - 0s 1ms/step - loss: 0.2295 - accuracy: 0.9319
Epoch 25/50
9/9 [=====] - 0s 989us/step - loss: 0.2847 - accuracy: 0.9050
Epoch 26/50
9/9 [=====] - 0s 1ms/step - loss: 0.2514 - accuracy: 0.9031
Epoch 27/50
9/9 [=====] - 0s 1ms/step - loss: 0.2021 - accuracy: 0.9367
Epoch 28/50
9/9 [=====] - 0s 2ms/step - loss: 0.1781 - accuracy: 0.9379
Epoch 29/50
9/9 [=====] - 0s 1ms/step - loss: 0.2043 - accuracy: 0.9279
Epoch 30/50
9/9 [=====] - 0s 956us/step - loss: 0.1782 - accuracy: 0.9352
Epoch 31/50
9/9 [=====] - 0s 1ms/step - loss: 0.2044 - accuracy: 0.9183
Epoch 32/50
9/9 [=====] - 0s 1ms/step - loss: 0.1640 - accuracy: 0.9483
Epoch 33/50
9/9 [=====] - 0s 1ms/step - loss: 0.1632 - accuracy: 0.9486
Epoch 34/50
9/9 [=====] - 0s 1ms/step - loss: 0.1498 - accuracy: 0.9533
Epoch 35/50
9/9 [=====] - 0s 1ms/step - loss: 0.2985 - accuracy: 0.8626
Epoch 36/50
9/9 [=====] - 0s 1ms/step - loss: 0.2327 - accuracy: 0.9109
Epoch 37/50
9/9 [=====] - 0s 1ms/step - loss: 0.2374 - accuracy: 0.9011
Epoch 38/50
9/9 [=====] - 0s 975us/step - loss: 0.2094 - accuracy: 0.9263
Epoch 39/50
9/9 [=====] - 0s 1ms/step - loss: 0.2739 - accuracy: 0.8847
Epoch 40/50
9/9 [=====] - 0s 1ms/step - loss: 0.1700 - accuracy: 0.9383
Epoch 41/50
9/9 [=====] - 0s 1ms/step - loss: 0.1753 - accuracy: 0.9408
Epoch 42/50
9/9 [=====] - 0s 1ms/step - loss: 0.1719 - accuracy: 0.9484
Epoch 43/50
9/9 [=====] - 0s 1ms/step - loss: 0.1667 - accuracy: 0.9382
Epoch 44/50
9/9 [=====] - 0s 1ms/step - loss: 0.1376 - accuracy: 0.9579
Epoch 45/50
9/9 [=====] - 0s 1ms/step - loss: 0.1487 - accuracy: 0.9532
Epoch 46/50
9/9 [=====] - 0s 2ms/step - loss: 0.1367 - accuracy: 0.9557
Epoch 47/50
9/9 [=====] - 0s 2ms/step - loss: 0.1698 - accuracy: 0.9352
Epoch 48/50
9/9 [=====] - 0s 1ms/step - loss: 0.1643 - accuracy: 0.9374
Epoch 49/50
9/9 [=====] - 0s 1ms/step - loss: 0.1914 - accuracy: 0.9201
Epoch 50/50
9/9 [=====] - 0s 1ms/step - loss: 0.1401 - accuracy: 0.9546

```

Figura 29: Treinamento do modelo com 50 neurônios na camada escondida

```

Confusion Matrix
[[19  9]
 [ 0 43]]
Classification Report
precision    recall   f1-score   support
0            1.00    0.68     0.81      28
1            0.83    1.00     0.91      43

accuracy                           0.87      71
macro avg       0.91    0.84     0.86      71
weighted avg    0.90    0.87     0.87      71

```

Figura 30: Métricas do modelo com 50 neurônios na camada escondida

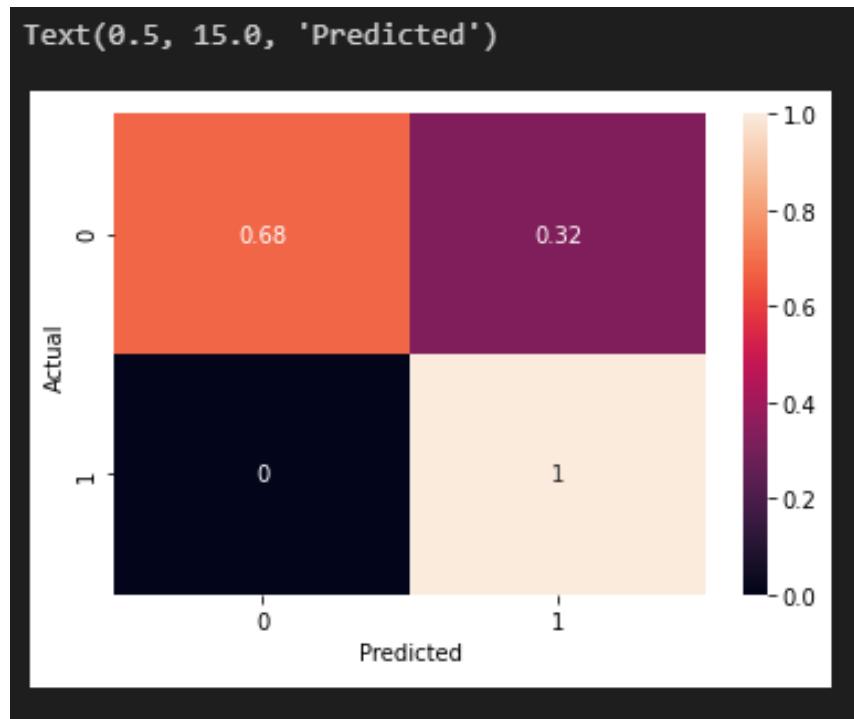


Figura 31: Matriz de Confusão do modelo com 50 neurônios na camada escondida

Curiosamente, todas as métricas são bastante similares em relação ao modelo com 5 neurônios apenas, e ainda piores que as métricas apresentadas pelo modelo original. Isso nos indica que, para este problema de classificação da base de dados Ionosphere, a quantidade de neurônios não influencia muito na qualidade da rede criada. Mais que isso, parece que a quantidade ótima de neurônios pode estar próxima da quantidade utilizada no modelo original, 15, uma vez que, tanto ao aumentar quanto ao diminuir este parâmetro, a rede piorou.

## 5 Teste Livre

- 5.1 Faça novos testes para avaliar o desempenho da Rede Neural no problema designado. Use a técnica K-Fold (com  $K = 10$ ) para analisar o resultado obtido.**
- 5.2 Faça análises e novas implementações que você julgue importante para o seu trabalho. Não esqueça de explicar a motivação da análise realizada.**

### 5.2.1 Early Stopping

O intuito de implementar o *early stopping* é, simplesmente, aprender a usá-lo na prática, analisando seus parâmetros e impactos na qualidade da rede. Alguns testes foram realizados antes do modelo exibido neste relatório com o intuito de estudar a implementação e ver como a alteração de cada parâmetro impactou a classificação. Entre esses parâmetros, foram variados a épocas (valores de 50 e 70) e o número de neurônios na camada escondida (15 e 20).

Para o *EarlyStopping* foi definido um *callback* com *monitor='accuracy'* e *patience=10*. Isso quer dizer que, se a métrica acurácia não exibir nenhum aumento em seu valor em 10 épocas, a rede para de treinar e fica com os pesos deste momento. Essa é uma estratégia para, no caso de muitas épocas (estamos com 70), evitar o *overfitting*, pois, uma vez que em 10 épocas a acurácia não aumentou, é provável que não aumente mais e nossa rede apenas perca generalização com os treinos faltantes.

A figura 32 exibe exatamente isso: a acurácia na época 26 foi de 0.9737 e após 10 épocas, nenhuma acurácia foi maior que ela, o que resultou na parada do treinamento na época 36.

```

Epoch 16/70
9/9 [=====] - 0s 1ms/step - loss: 0.2260 - accuracy: 0.9297
Epoch 17/70
9/9 [=====] - 0s 1ms/step - loss: 0.2477 - accuracy: 0.9104
Epoch 18/70
9/9 [=====] - 0s 4ms/step - loss: 0.1662 - accuracy: 0.9577
Epoch 19/70
9/9 [=====] - 0s 1ms/step - loss: 0.2903 - accuracy: 0.8836
Epoch 20/70
9/9 [=====] - 0s 5ms/step - loss: 0.4005 - accuracy: 0.8658
Epoch 21/70
9/9 [=====] - 0s 2ms/step - loss: 0.3316 - accuracy: 0.8647
Epoch 22/70
9/9 [=====] - 0s 2ms/step - loss: 0.3104 - accuracy: 0.9035
Epoch 23/70
9/9 [=====] - 0s 1ms/step - loss: 0.2087 - accuracy: 0.9385
Epoch 24/70
9/9 [=====] - 0s 1ms/step - loss: 0.1819 - accuracy: 0.9462
Epoch 25/70
9/9 [=====] - 0s 1ms/step - loss: 0.1589 - accuracy: 0.9563
Epoch 26/70
9/9 [=====] - 0s 1ms/step - loss: 0.1426 - accuracy: 0.9737
Epoch 27/70
9/9 [=====] - 0s 1ms/step - loss: 0.1963 - accuracy: 0.9221
Epoch 28/70
9/9 [=====] - 0s 1ms/step - loss: 0.2671 - accuracy: 0.8986
Epoch 29/70
9/9 [=====] - 0s 2ms/step - loss: 0.2199 - accuracy: 0.9140
Epoch 30/70
9/9 [=====] - 0s 2ms/step - loss: 0.1554 - accuracy: 0.9533
Epoch 31/70
9/9 [=====] - 0s 2ms/step - loss: 0.1817 - accuracy: 0.9372
Epoch 32/70
9/9 [=====] - 0s 2ms/step - loss: 0.1983 - accuracy: 0.9177
Epoch 33/70
9/9 [=====] - 0s 2ms/step - loss: 0.2452 - accuracy: 0.8992
Epoch 34/70
9/9 [=====] - 0s 1ms/step - loss: 0.1589 - accuracy: 0.9649
Epoch 35/70
9/9 [=====] - 0s 1ms/step - loss: 0.1469 - accuracy: 0.9657
Epoch 36/70
9/9 [=====] - 0s 1ms/step - loss: 0.1412 - accuracy: 0.9458

```

Figura 32: Treinamento do modelo com early stopping

Embora a minha ideia tenha sido evitar um *overfitting*, parece que o problema contrário ocorreu. Como pode ser visto na figura 33, a acurácia geral do modelo foi de apenas 0.85, bem menor que a original. Isso pode indicar que, ao parar na época 36 a rede ainda não havia treinado o suficiente seus pesos. O f1-score também ser menor corrobora com esta possibilidade.

[[17 11] [ 0 43]]					
Classification Report					
		precision	recall	f1-score	support
0		1.00	0.61	0.76	28
1		0.80	1.00	0.89	43
		accuracy		0.85	71
macro avg		0.90	0.80	0.82	71
weighted avg		0.88	0.85	0.83	71

Figura 33: Métricas do teste com EarlyStopping

## Referências

- [1] UCI Machine Learning Repository,  
<https://archive.ics.uci.edu/ml/datasets/Ionosphere>
- [2] Sigillito, V. G., Wing, S. P., Hutton, L. V., & Baker, K. B. (1989). *Classification of radar returns from the ionosphere using neural networks*. Johns Hopkins APL Technical Digest, 10, 262-266.