Redes Neurais aplicadas ao problema de classificação no dataset Iris

Renan Mendanha [1]

[1]Instituto de Computação – Universidade Federal do Rio de Janeiro (UFRJ)
Rio de Janeiro – RJ – Brasil

renanma@dcc.ufrj.br

Abstract. *This article is a report that explores the use of Neural Networks to solve multi-class classification problems [1]. The main objective is to analyze and modify the classification model, aiming to achieve more accurate and reliable predictions in relation to the real data contained in the database.*

Resumo. *Este artigo é um relatório que explora a utilização de Redes Neurais para resolver problemas de classificação com múltiplas classes[1]. O objetivo principal é analisar e modificar o modelo de classificação, buscando alcançar predições mais precisas e fidedignas em relação aos dados reais contidos na base de dados.*

1. Introdução

Este relatório explora a utilização de Redes Neurais para resolver problemas de classificação com múltiplas classes[1]. O objetivo principal é analisar experimentos de modificações nos parâmetros do modelo de classificação, de forma que seja possível construir um modelo final com o qual, combinando as mudanças positivas nos parâmetros, possamos alcançar a melhor predição possível em relação aos dados reais contidos na base de dados.

Apesar do grande poder de classificação, modelos de Redes Neurais possuem grande margem para melhorias, tendo em vista as complexidades e ruídos inerentes dos dados usados. Este relatório tem como principal motivação ajudar no entendimento do modelo e compreensão dos resultados obtidos a partir das modificações aplicadas na construção de um classificador, explorando as possíveis estratégias para otimizar o desempenho dos modelos de classificação com múltiplas classes.

2. Descrição da base

A base de dados descreve diferenças entre três espécies de flores de íris: Iris setosa, Iris virginica e Iris versicolor. A base contém 50 instâncias de cada classe, com a classe sendo indicada pelo atributo “class”, que é a nossa variável target, indicando o balanceamento total do dataset. Os outros quatro atributos caracterizam a instância de uma classe, sendo estes: “sepalLengthInCM” descrevendo o comprimento da sépala em centímetros, “sepalWidthInCM” descrevendo a largura da sépala em centímetros, “petalLengthInCM” descrevendo o comprimento da pétala em centímetros e “petalWidthInCM” descrevendo a largura da pétala em centímetros.

Todos esses quatro atributos são caracterizados numericamente, porém a variável target, que contém o nome das três classes possíveis, é categórica. Na seção seguinte, discutiremos a transformação desta variável e suas implicações.

3. Pré processamento

Inicialmente, serão utilizados todos os 4 atributos como variável de entrada, então devemos analisar os dados no dataset para garantir que não ocorra nenhuma instância com dados anormais.

Após analisar as instâncias no conjunto de dados, conclui-se que não foi encontrado nenhum dado mal formado nem faltante, tanto para as variáveis de entrada quanto para a variável target, definindo um dataset bem formado.

3.1. Variáveis de entrada

Focando nos dados de entrada, é possível observar suas distribuições e, ao utilizar o método dos quartis com o apoio visual do boxplot[2], consegue-se buscar por características discrepantes de uma instância (outliers), as quais podem vir a prejudicar o treinamento do modelo. Este processo pode ser observado abaixo na Figura 1.

Com esta análise feita, observa-se que foram encontrados três outliers superiores e um inferior no atributo sepalWidthInCM. Entretanto, sabendo que os valores desses outliers são bem próximos aos valores encontrados nos dados e assumindo a possibilidade de que esses valores possam de fato existir neste atributo, foi tomada a escolha de manter essas instâncias no dataset.

Ainda abordando as variáveis de entrada, foi aplicada uma normalização z-score sobre os dados antes do treino, isso torna mais fácil o aprendizado dos pesos do modelo, pois os dados dos atributos estarão na mesma escala e de forma centralizada, conforme uma distribuição normal (normalization e standardization)[3].

3.2. Variável target

A fim de proporcionar o treinamento adequado da Rede Neural, a variável “class”, que define a classe de íris, deve ser transformada em um vetor one-hot, de forma que cada neurônio da camada de saída da rede representa uma entrada nesse vetor, ou seja, cada neurônio da camada de saída está relacionado a uma classe.

Dessa forma, o atributo “class” não é usado no treinamento, e sim o array numérico one-hot (com valores 0 ou 1, de forma que o valor 1 estará somente em uma coluna desse array) que contém os atributos “Iris-setosa”, “Iris-versicolor” e “Iris-virginica”.

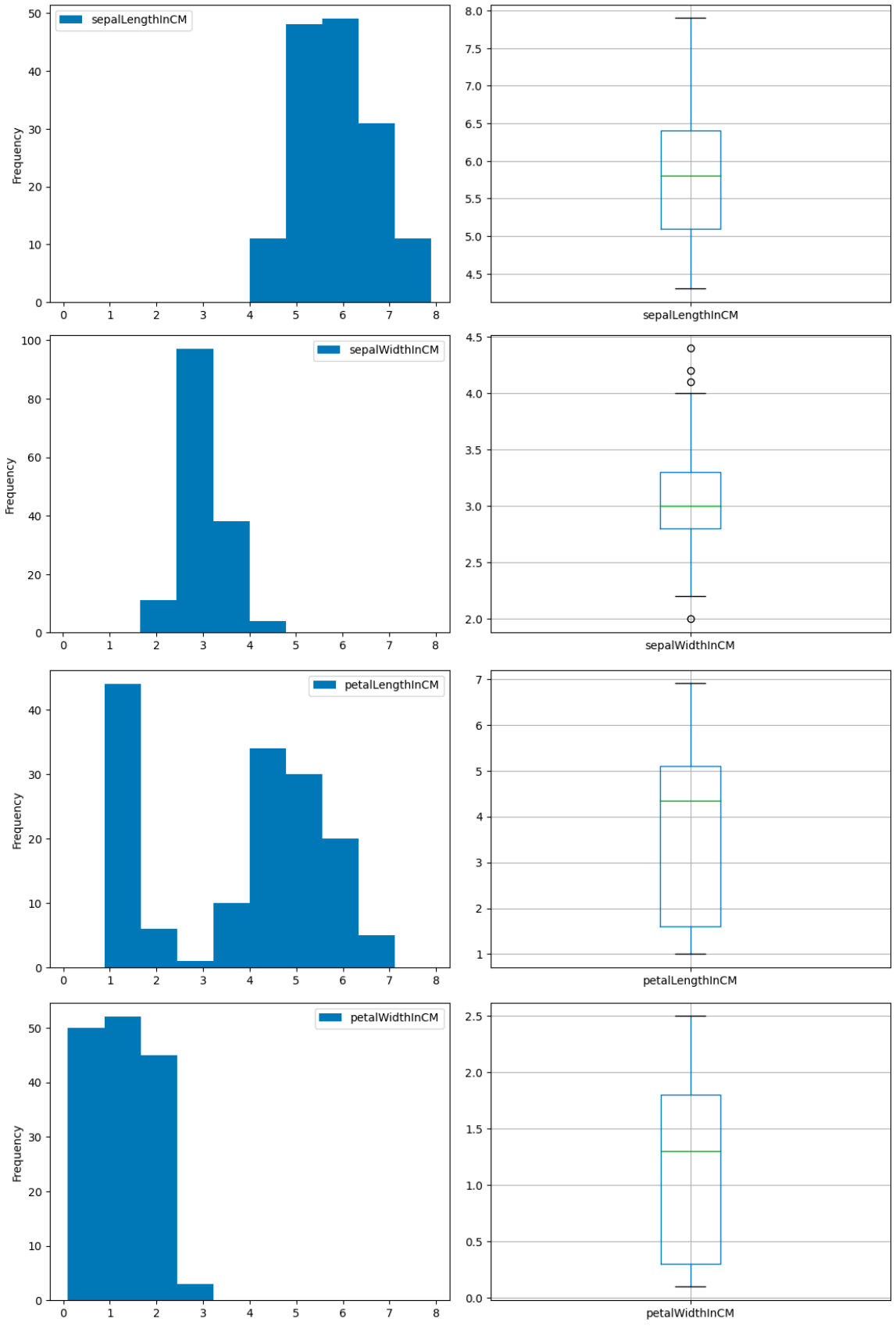


Figura 1. Histograma e boxplot das variáveis de entrada

4. Metodologia de classificação

Antes de entrar em detalhes sobre o modelo, vale ressaltar que o dataset foi dividido de forma que seja utilizado 90% dos dados para treino e validação, deixando o conjunto de testes com 5 instancias de cada classe, mantendo assim o balanceamento.

Os dados de treino e validação foram utilizados para validação cruzada k-fold, de modo que, para cada fold, um novo classificador (construído a partir dos mesmos parâmetros, mas com dados de treinamento diferentes) é gerado. Será mantido um número padrão de cinco folds durante todos os experimentos.

Foi utilizado o modelo MLPClassifier (Multi-layer Perceptron Classifier)[4] para classificar os dados, mudando os parâmetros da rede a cada experimento. Inicialmente usando-se todos os atributos de entrada como features do modelo, constrói-se uma input layer com quatro neurônios.

O output da rede é dado como uma probabilidade nos neurônios da camada de saída, com cada neurônio representando uma classe. A partir destas probabilidades, selecionando o neurônio de maior valor, é possível realizar a predição de uma instância dados os valores dos atributos de entrada para a input layer.

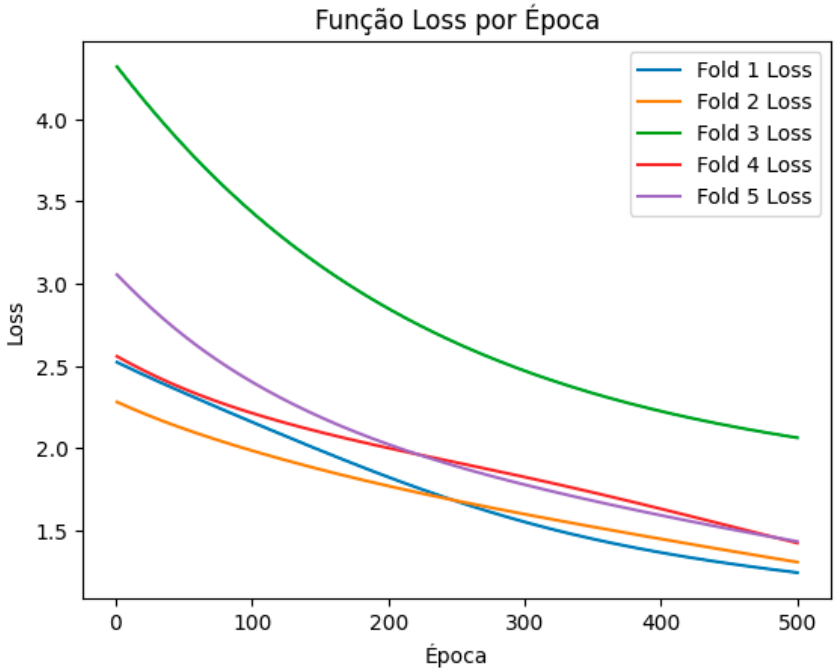
5. Experimentos realizados e resultados obtidos

O primeiro experimento será um modelo de controle, com parâmetros simples:

- 1 Camada escondida
- 2 Neurônios por camada escondida
- 500 épocas de treinamento
- Função de ativação identity
- Solver adam para otimização dos pesos da rede
- Taxa de aprendizado com valor 0.001

Os resultados obtidos podem ser observados na Figura 2. Consegue-se observar que o modelo não atinge convergência dados os parâmetros definidos. Como alternativa para a resolução desse problema, podemos aumentar o número de épocas ou até mesmo aumentar o learning rate. Vale também ressaltar que no conjunto de validação os resultados obtidos foram bem ruins, pois a acurácia e o f1-score para alguns dos folds foi bem baixo, tornando o modelo instável.

Observando a matriz de confusão, vê-se que o classificador acertou uma boa quantidade de instâncias da classe Iris-virginica e Iris-setosa porém errou consideravelmente a versicolor. Isso explica o F1-Score ruim no conjunto de validação. Apesar disso, devemos nos atentar que devido à instabilidade do modelo a matriz de confusão do último fold pode variar muito, não sendo uma boa métrica de avaliação, por este motivo, esta métrica será omitida em alguns experimentos do relatório.




```
Acurácia por fold
Conjunto de treino:      [0.68, 0.63, 0.36, 0.71, 0.62]
Conjunto de validação:  [0.59, 0.63, 0.15, 0.52, 0.7]

Acurácia média no Conjunto de treino:      0.6
Acurácia média no Conjunto de validação:  0.52

F1-Score macro avg por fold
Conjunto de treino:      [0.54, 0.52, 0.27, 0.6, 0.51]
Conjunto de validação:  [0.49, 0.6, 0.13, 0.56, 0.61]

F1-Score médio no Conjunto de treino:      0.49
F1-Score médio no Conjunto de validação:  0.48
```

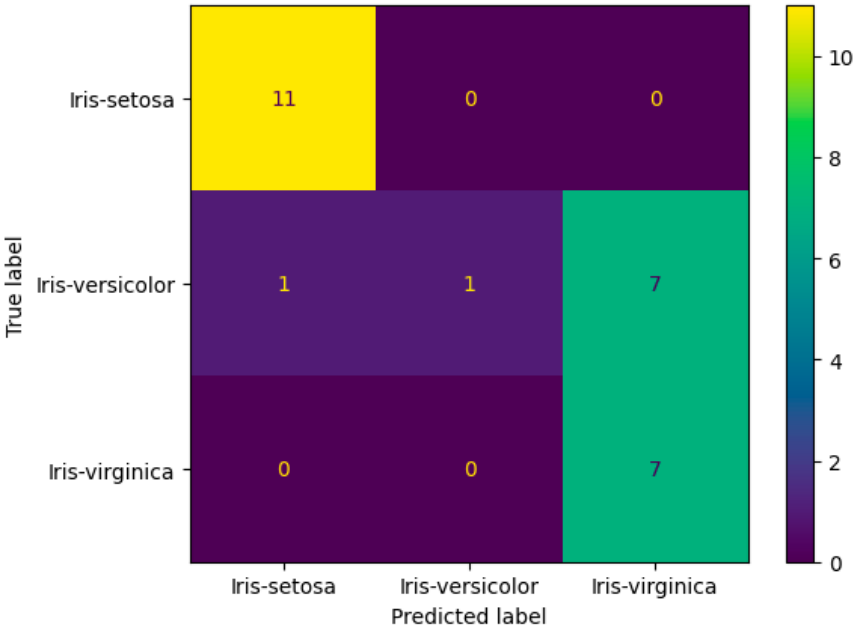


Figura 2. Resultados para modelo de controle

5.1. Experimentando mudanças nas features

Desta vez, removendo a restrição de utilizar todas as variáveis de entrada, pode-se verificar a correlação entre os atributos (a partir do método pearson), a fim de possivelmente proporcionar um melhor desempenho do modelo ao mesmo tempo que sua complexidade é diminuída, pois a input layer estará menos complexa. Pode-se observar as informações relevantes da matriz de confusão na Figura 3.

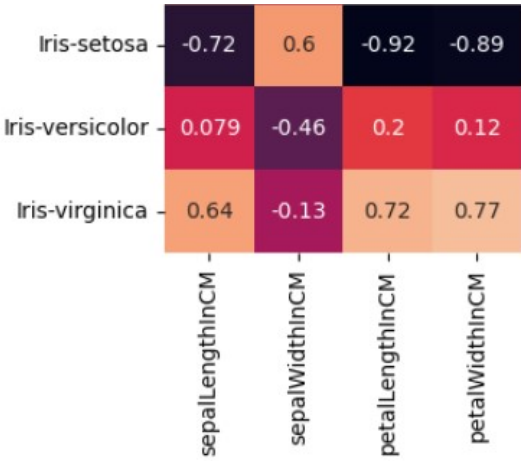


Figura 3. Correlação entre os atributos

A partir correlação acima, pode-se extrair as seguintes informações por classe:

- Atributos Com alta correlação à Iris-setosa:
  - Sepal length
  - Petal length
  - Petal width
- - Atributos Com alta correlação à Iris-versicolor:
  - Sepal width
- - Atributos Com alta correlação à Iris-virginica:
  - Sepal length
  - Petal length
  - Petal width

Portanto, é inviável deixar de usar algum atributo de entrada, já que todos eles contribuem com relativamente alta correlação na classificação de uma Íris.

5.2. Experimentando mudança na taxa de aprendizado

Como visto no modelo de controle, não atinge-se convergência, devemos então aplicar inicialmente uma mudança que possa implicar na convergência do modelo ao treiná-lo. A taxa de aprendizado será aumentada em uma escala decimal ("learning_rate_init" = 0.01) a fim de ajustar mais rapidamente os pesos da rede mantendo o mesmo número de épocas para treinamento. O resultado desta modificação pode ser observado na Figura 4.

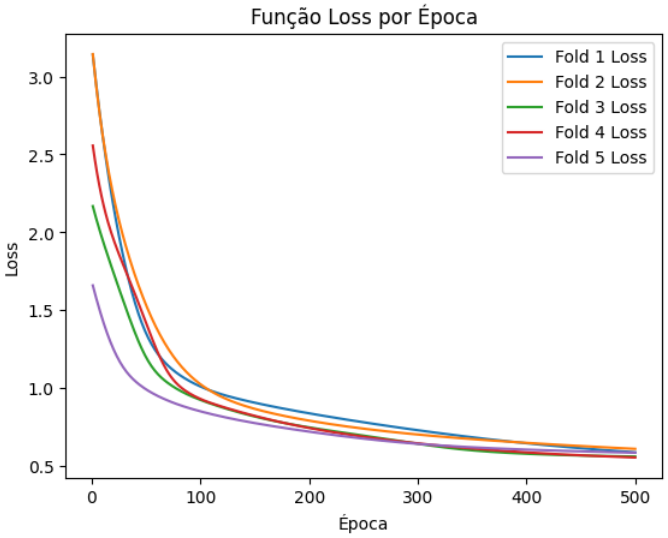


Figura 4. Loss curve com um maior learning rate

Com essa modificação, o novo classificador conseguiu valores de acurácia e F1-Score muito melhores (acima de 0.85 para todos os folds), se tornando muito mais estável e apresentando uma loss curve que se aproxima bastante da convergência, com um decaimento inicial muito mais abrupto e, nas épocas finais, sem muitas mudanças.

5.3. Experimentando mudança no número de épocas

Com o mesmo objetivo da mudança anterior, aplica-se sobre o modelo de controle um aumento no número de épocas (aumentando em uma escala decimal). A Figura 5 apresenta o novo comportamento da função Loss.

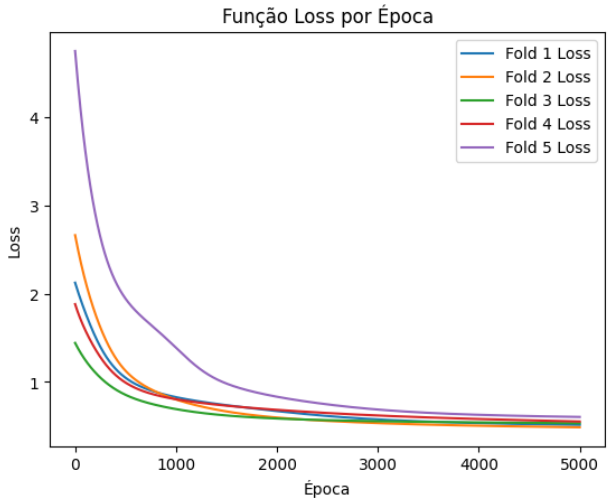


Figura 5. Loss curve com um número maior de épocas

O resultado desta modificação é extremamente parecido com a modificação anterior, pois, apesar da taxa de mudança dos pesos da rede ser menor, o modelo apresenta mais margem em relação ao número de épocas para uma maior aproximação da convergência do mesmo.

5.4. Combinando o aumento da taxa de aprendizado e número de épocas

Apesar dos bons resultados obtidos nas duas últimas mudanças, o modelo emitiu um aviso no treino, indicando que não foi possível atingir convergência completa dentro do número de épocas. Com a combinação do aumento da taxa de aprendizado e número de épocas, consegue-se um modelo que aprende mais rápido e por mais tempo, na esperança de resolver o problema de convergência completa. Resultado da Loss curve apresentado na Figura 6.

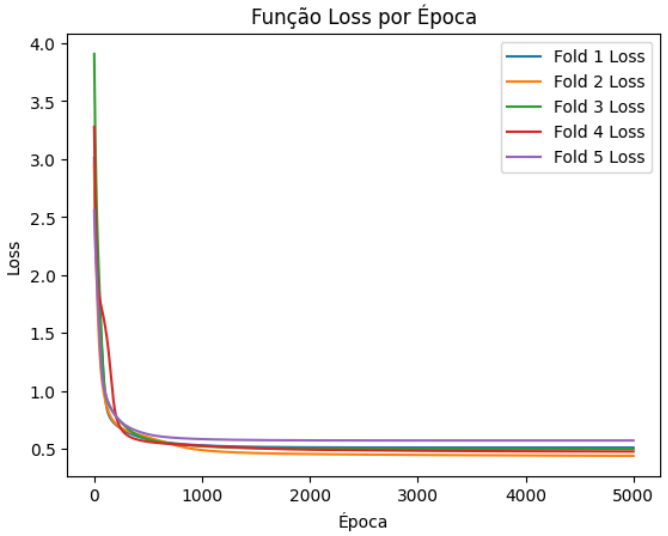


Figura 6. Loss curve maior learning rate e épocas

Novamente, o classificador conseguiu valores de acurácia e F1-Score muito bons (acima de 0.85 para todos os folds), mas alguns folds ainda apresentam o aviso de não convergência. Isto pode indicar uma insuficiência da própria estrutura da rede, continuamos as mudanças dos parâmetros com base nesta hipótese.

5.5. Mudanças nas camadas escondidas da rede

Aumentando a complexidade da rede, talvez consigamos algum mecanismo mais potente de classificação. Para isso, foi experimentado aumentar o número de neurônios (quatro em uma camada escondida, igualando o número de neurônios da input layer) e número de camadas escondidas da rede, ambos os experimentos feitos separadamente sobre o modelo de controle.

Ambos os experimentos apresentaram resultados insatisfatórios, mas não por ser uma mudança ruim, e sim por haver espaço para convergência do modelo, já que a rede se tornou mais complexa e os parâmetros iniciais não deram conta de atualizar os pesos dentro do número de épocas de forma a produzir um bom classificador. Mais detalhes sobre as métricas obtidas podem ser encontrados no código fonte.

5.6. Combinando todas as mudanças anteriores

Ao aplicar as melhorias anteriores em um só modelo, proporcionando maior margem para a convergência com uma rede mais complexa, talvez seja possível um resultado

melhor e que tenha convergência completa para todos os folds. A Figura 7 mostra algumas das métricas obtidas.

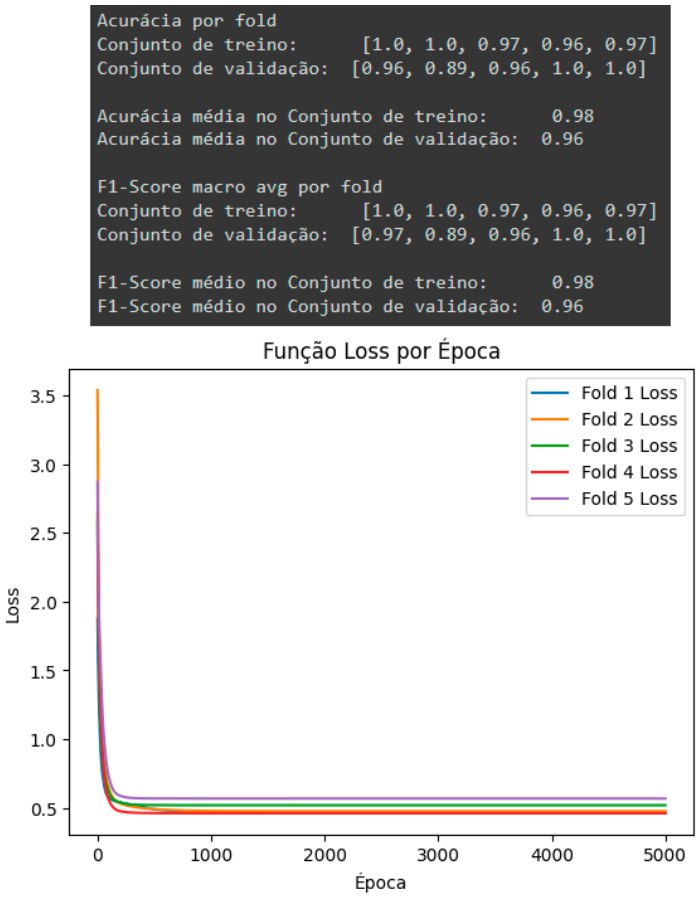


Figura 7. Métricas do modelo com parâmetros combinados

Todos os folds, além de terem métricas boas, conseguiram convergir completamente, já que não foi mostrada nenhuma flag de warning como output.

5.7. Predição com o melhor modelo

Decidir o melhor modelo é uma tarefa relativa às prioridades escolhidas, estaremos decidindo como o melhor modelo, o que tem todas as combinações de parâmetros modificados, já que este apresenta as melhores métricas avaliadas.

Desta vez, utilizando todos os 90% dos dados para treinamento e 10% para teste, consegue-se avaliar definitivamente o modelo com classes balanceadas das instâncias no teste. Segue na Figura 8 os resultados.

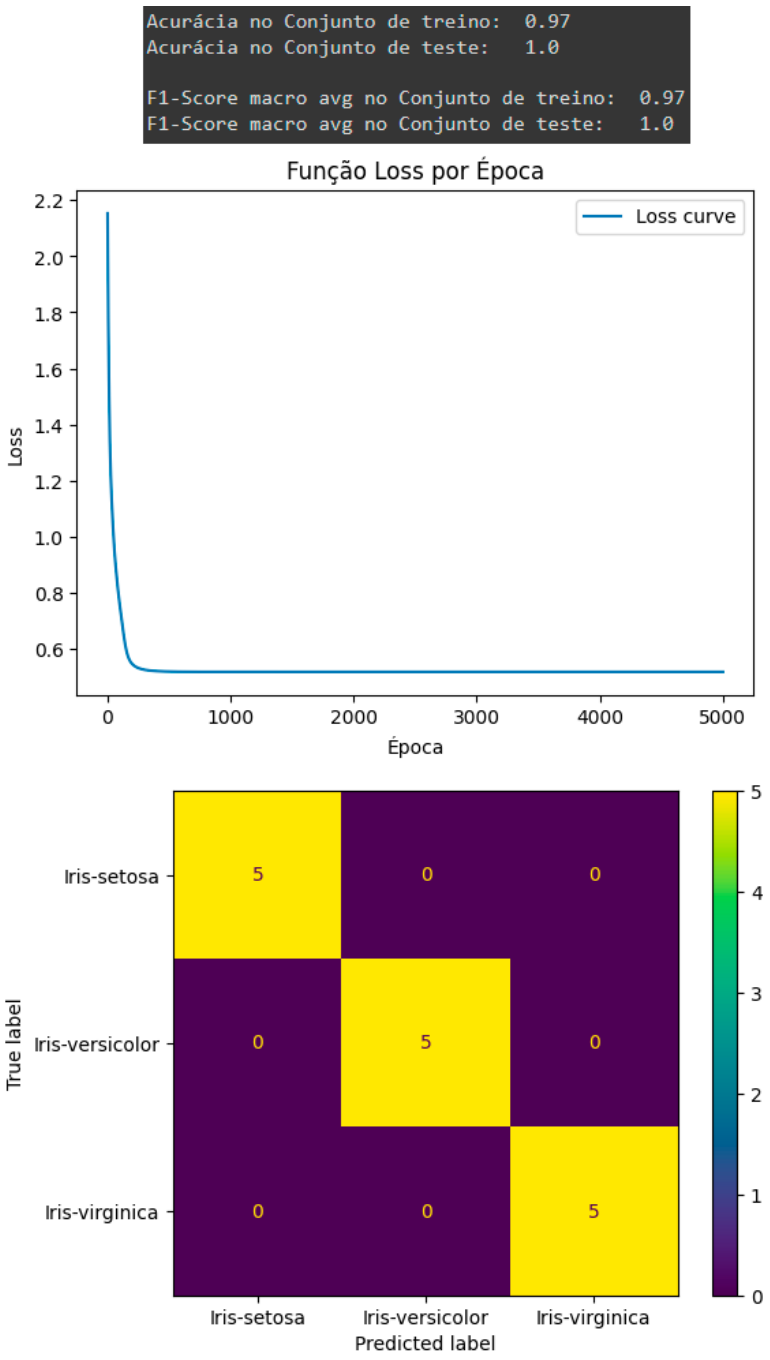


Figura 8. Métricas do melhor modelo

Além do modelo atingir a convergência, ele apresenta valores de acurácia e F1-Score muito bons, tanto para o conjunto de treino quanto para o conjunto de testes. Pode-se também observar um fato interessante ao analisar as diferenças das métricas no treino e teste, sendo os valores de teste maiores que o de treino. Isso pode indicar a resiliência do modelo ao overfitting sobre os dados de treino. Além disso, ao observar a matriz de confusão, vê-se uma boa taxa de acerto para cada classe, o que indica a capacidade de diferenciação entre classes do classificador gerado.

6. Conclusão

Neste relatório, é exposta a utilização e modificação de Redes Neurais para resolver problemas de classificação com múltiplas classes. Fica claro que usar um mecanismo de classificação baseado em um modelo Multi-layer Perceptron é mais do que suficiente para construir um classificador de três classes diferentes, para um dataset tão simples, com quatro atributos de entrada bem comportados e com relativamente alta correlação à variável target.

Além disso, como dito anteriormente, decidir o melhor modelo é uma tarefa relativa às prioridades escolhidas. É totalmente válido argumentar que um modelo não tão complexo, mas com métricas ligeiramente menores (o que foi observado nas primeiras modificações), é o melhor modelo. Devemos botar na balança o ganho na habilidade de classificação do modelo e a complexidade do mesmo, pois isso acarreta em uma maior carga computacional no momento em que é realizado o treinamento do mesmo a partir de um dataset de maior dimensionalidade (mais features e mais instâncias).

Referências

[1] Wikipedia contributors. (2023, March 21). “Multiclass classification”. https://en.wikipedia.org/w/index.php?title=Multiclass_classification&oldid=1145868961

[2] Pandas API reference. (2023). “Pandas DataFrame boxplot”. (2.0.2). https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.boxplot.html

[3] Peshawa J. Muhammad Ali, Rezhna H. Faraj. (2014). “Data Normalization and Standardization: A Technical Report”. Machine Learning Technical Reports, 2014, 1(1), pp 1-6. https://docs.google.com/document/d/1x0A1nUz1WWtMCZb5oVzF0SVMY7a_58KQulqQVT8LaVA/edit#

[4] Scikit learn. (2023). “sklearn neural_network MLPClassifier”. (1.2.2). https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html