Métodos de classificação aplicados ao dataset de performance corporal

Jonathan Suhett Barbêdo[1], Renan Mendanha [1]

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

jonathansb@dcc.ufrj.br ,renanma@dcc.ufrj.br

Abstract. *This article is a report that explores the use of various methods to solve the problem of multi-class classification[1]. The main objective is to analyze each classification model, aiming to define the performance of each model in relation to the classification of body performance data.*

Resumo. *Este artigo é um relatório que explora a utilização de diversos métodos para resolver o problema de classificação com múltiplas classes[1]. O objetivo principal é analisar cada modelo de classificação, buscando definir o desempenho de cada modelo em relação à classificação dos dados de performance corporal.*

1. Introdução

Classificar dados em várias classes é uma tarefa desafiadora em diversos domínios, incluindo o campo da análise de desempenho corporal. Neste artigo, apresentamos um relatório que investiga a utilização de diferentes métodos para lidar com o problema de classificação com múltiplas classes.

O foco principal é analisar o desempenho de cada modelo de classificação e sua eficácia na categorização dos dados de desempenho corporal. Para isso, iremos explorar neste relatório três modelos populares: Redes neurais, utilizando um classificador Multi Layer Perceptron, regressão logística e árvore de decisão.

2. Descrição da base

A base de dados utilizada (Body performance Data[2]), gerada a partir de dados fornecidos pela Korea Sports Promotion Foundation[3], descreve a separação do desempenho corporal de uma população em quatro categorias, A, B, C e D, sendo A a melhor classificação e D a pior. A base contém 13393 instâncias, as quais possuem as classes distribuídas de maneira aproximadamente igual entre as categorias.

A categoria de cada instância é indicada pela coluna “class”, que é a nossa variável target. Temos outros onze atributos de entrada que caracterizam a instância de uma classe, sendo estes:

- age : Idade do indivíduo;
- gender : Sexo do indivíduo;
- height_cm : Altura do indivíduo em centímetros;
- weight_kg : Peso do indivíduo em quilogramas;
- body fat_% : Porcentagem de gordura corporal do indivíduo;
- diastolic : Pressão diastólica do sangue do indivíduo;

- systolic : Pressão sistólica do sangue do indivíduo;
- gripForce : Força de preensão do indivíduo;
- sit and bend forward_cm : Distância em centímetros ao sentar e inclinar para frente;
- sit-ups counts : Número de abdominais;
- broad jump_cm : Distância em centímetros do salto horizontal.

Todos os atributos de entrada, com exceção do atributo “gender” são caracterizados numericamente, e a variável target, que contém o identificador das quatro classes possíveis, é categórica. Na seção seguinte, discutiremos a transformação da variável target e suas implicações. Para o atributo “gender”, podemos aplicar uma codificação numérica (0 para masculino e 1 para feminino), possibilitando lidar numericamente com todos os dados dessa coluna.

3. Pré processamento

Inicialmente, serão utilizados todos os atributos como variável de entrada, pois não conseguimos encontrar atributos com alta correlação com as variáveis target intermediárias, como pode ser observado na figura 1, que mostra a matriz de correlação formada a partir do método pearson. Portanto, tentaremos nos aproveitar inicialmente de todos os atributos para a classificação.

Figura 1. Matriz de correlação

Após analisar as instâncias do dataset, 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.

Nos subtópicos a seguir, está descrita a busca por outliers, com a intenção de removê-los ou até mesmo recuperá-los, através da substituição dos valores anômalos pela média dos valores do atributo em questão.

3.1. Variáveis de entrada

Inicialmente, podemos olhar a distribuição e os quartis das variáveis de entrada, a fim de identificar possíveis dados discrepantes. Essas medidas podem ser observadas na figura a seguir:

	count	mean	std	min	25%	50%	75%	max
age	13393.0	36.775106	13.625639	21.0	25.0	32.0	48.0	64.0
gender	13393.0	0.632196	0.482226	0.0	0.0	1.0	1.0	1.0
height_cm	13393.0	168.559807	8.426583	125.0	162.4	169.2	174.8	193.8
weight_kg	13393.0	67.447316	11.949666	26.3	58.2	67.4	75.3	138.1
body fat_%	13393.0	23.240165	7.256844	3.0	18.0	22.8	28.0	78.4
diastolic	13393.0	78.796842	10.742033	0.0	71.0	79.0	86.0	156.2
systolic	13393.0	130.234817	14.713954	0.0	120.0	130.0	141.0	201.0
gripForce	13393.0	36.963877	10.624864	0.0	27.5	37.9	45.2	70.5
sit and bend forward_cm	13393.0	15.209268	8.456677	-25.0	10.9	16.2	20.7	213.0
sit-ups counts	13393.0	39.771224	14.276698	0.0	30.0	41.0	50.0	80.0
broad jump_cm	13393.0	190.129627	39.868000	0.0	162.0	193.0	221.0	303.0

Figura 2. Descrição e distribuição das variáveis de entrada

Olhando para a descrição das métricas das colunas do datraframe, podemos observar a possível presença de alguns dados anômalos:

- Os atributos "diastolic", "systolic", "gripForce", "sit-ups counts" e "broad jump_cm" possuem valores 0, isso pode significar que estas informações não foram coletadas. como alternativa, podemos substituí-las pela média dos dados dos mesmos atributos, a fim de evitar a inutilização de algumas instâncias do dataset.

valores alterados: 140

	count	mean	std	min	25%	50%	75%	max
diastolic	13393.0	78.802725	10.720430	6.0	71.0	79.0	86.0	156.2
systolic	13393.0	130.244541	14.670849	14.0	120.0	130.0	141.0	201.0
gripForce	13393.0	36.972158	10.610447	1.6	27.5	37.9	45.2	70.5
sit-ups counts	13393.0	40.144142	13.744872	1.0	31.0	41.0	50.0	80.0
broad jump_cm	13393.0	190.271637	39.527763	20.0	162.0	193.0	221.0	303.0

Figura 3. Substituição dos valores zerados pela média

- O atributo "height_cm" possui instâncias com valor mínimo muito baixo para altura, portanto, removeremos os outliers a partir da métrica z-score com cutoff 3.

Outliers removidos: 6

	count	mean	std	min	25%	50%	75%	max
height_cm	13387.0	168.573676	8.402106	143.4	162.4	169.2	174.8	193.8

Figura 4. Remoção dos outliers do atributo “height_cm”

- O atributo "weight_kg" possui instâncias com valor mínimo muito baixo para um peso, portanto, removeremos apenas valores muito baixos (valores menores que 40kg).

Outliers removidos: 20

	count	mean	std	min	25%	50%	75%	max
weight_kg	13367.0	67.501383	11.893978	40.3	58.2	67.5	75.4	138.1

Figura 5. Remoção dos outliers do atributo “weight_kg”

- O atributo "body fat_%" possui instâncias com valor máximo muito alto, portanto, removeremos os outliers a partir da métrica z-score com cutoff 3.

Outliers removidos: 44

	count	mean	std	min	25%	50%	75%	max
body fat_%	13323.0	23.153286	7.111078	3.0	18.0	22.8	28.0	44.8

Figura 6. Remoção dos outliers do atributo "body fat_%"

- Para o atributo "sit and bend forward_cm", valores de mínimo negativos são permitidos neste exercício, significa que você não conseguiu alcançar a ponta dos dedos das mãos na ponta dos pés. Entretanto, o valor máximo está muito alto, logo, removeremos os outliers a partir da métrica z-score com cutoff 3.

Outliers removidos: 133

	count	mean	std	min	25%	50%	75%	max
sit and bend forward_cm	13190.0	15.491867	7.628574	-10.0	11.1	16.3	20.8	40.0

Figura 7. Remoção dos outliers do atributo "sit and bend forward_cm"

Finalmente, após a remoção dos outliers, restaram 13190 instâncias do conjunto de dados, a partir dessas instâncias, foi aplicada uma normalização z-score sobre os dados antes do treino, isso torna mais preciso o aprendizado dos modelos, pois os dados dos atributos estarão na mesma escala e de forma centralizada, conforme uma distribuição normal (normalization e standardization)[4].

3.2. Variável target

A fim de proporcionar o treinamento adequado da rede neural, e possibilitar a heurística one versus rest do modelo de regressão logística, a variável "class" deve ser transformada em um vetor one-hot. Assim, cada neurônio da camada de saída da rede neural representa uma entrada nesse vetor, ou seja, cada neurônio da camada de saída está relacionado a uma classe. Além disso, a regressão pode ser realizada separadamente por classe, como dita o algoritmo one versus rest.

4. Metodologia

Antes de entrar em detalhes sobre os modelos, vale ressaltar que o dataset foi dividido de forma que seja utilizado 75% dos dados para treino e validação, deixando o conjunto de testes com os outros 25%. Vale ressaltar que todas as divisões dos dados foram feitas de forma estratificada, mantendo assim o balanceamento das instâncias por classe.

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 quatro folds durante todos os experimentos.

Durante o treinamento, são geradas métricas por fold para avaliar a estabilidade e desempenho do modelo. Após o treinamento, é realizada a predição dos dados de teste e a geração das métricas finais do modelo.

5. Experimentos realizados

Foram realizados experimentos com modelos de rede neural MLP, regressão logística e árvore de decisão, com e sem poda. A seguir, são apresentadas as informações extraídas durante o treinamento dos mesmos.

5.1. Multi Layer Perceptron

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

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

- 1 Camada escondida
- 4 Neurônios por camada escondida
- 100 épocas de treinamento
- Função de ativação identity
- Solver adam para otimização dos pesos da rede
- Taxa de aprendizado com valor 0.01

Após o treinamento do modelo, foram observadas as seguintes métricas:

```
Acurácia por fold
Conjunto de treino:      [0.58, 0.58, 0.59, 0.58]
Conjunto de validação:   [0.6, 0.58, 0.57, 0.59]

Acurácia média no Conjunto de treino:      0.58
Acurácia média no Conjunto de validação:   0.58

F1-Score macro avg por fold
Conjunto de treino:      [0.57, 0.56, 0.56, 0.55]
Conjunto de validação:   [0.58, 0.55, 0.55, 0.56]

F1-Score médio no Conjunto de treino:      0.56
F1-Score médio no Conjunto de validação:   0.56
```

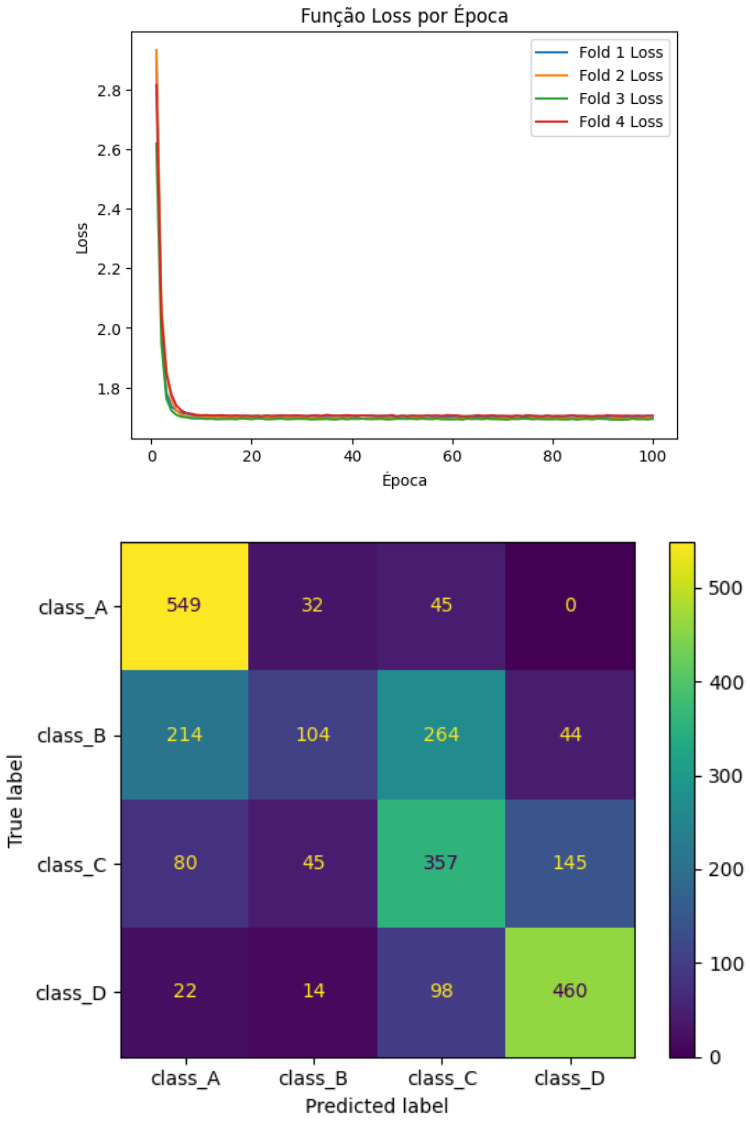

Figura 8. Treinamento inicial MLP

A partir dessas informações, conseguimos identificar estabilidade do modelo, pois os valores das métricas de cada fold não divergem tanto, além disso, podemos perceber uma certa simplicidade da rede, pois não foi necessário um grande número de épocas para atingir a convergência do mesmo. No último gráfico, a matriz de confusão do último fold, é possível perceber a dificuldade em classificar as classes intermediárias do modelo (B e C), este resultado já era esperado, dada a observação da correlação das variáveis de entrada feita anteriormente.

Ao tentarmos mudar a estrutura das camadas internas para 2 camadas com 16 neurônios cada, esperávamos obter métricas de treinamento muito melhores, porém fomos surpreendidos com a mudança mínima dos resultados que obtivemos anteriormente. Houve mudança mínima nas classes B e C da matriz de confusão e para as médias de acurácia e F1-score os valores foram extremamente parecidos.

5.2. Regressão Logística

Desta vez, utilizaremos o modelo de regressão logística, baseando-se na heurística one versus rest para resolver o problema de classificação multiclasse. Abaixo estão apresentados os dados obtidos a partir da validação cruzada.

```
Acurácia por fold
Conjunto de treino:      [0.59, 0.59, 0.58, 0.58]
Conjunto de validação:  [0.58, 0.58, 0.59, 0.57]

Acurácia média no Conjunto de treino:      0.58
Acurácia média no Conjunto de validação:  0.58

F1-Score macro avg por fold
Conjunto de treino:      [0.57, 0.57, 0.56, 0.56]
Conjunto de validação:  [0.56, 0.56, 0.57, 0.56]

F1-Score médio no Conjunto de treino:      0.56
F1-Score médio no Conjunto de validação:  0.56
```

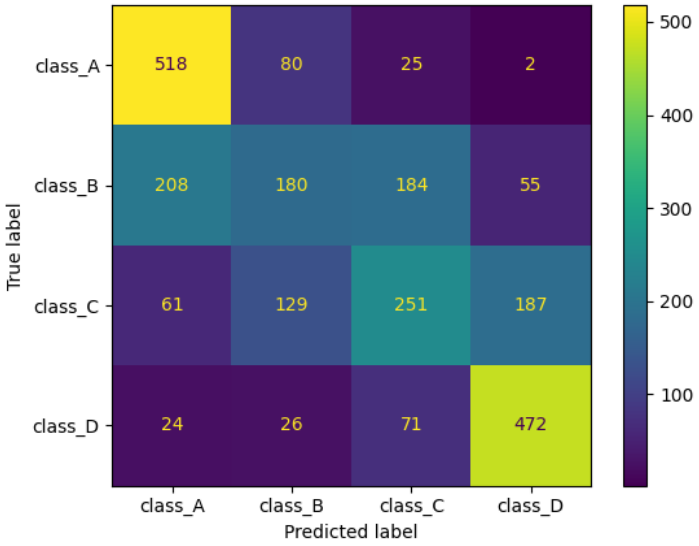

Figura 9. Treinamento da Regressão Logística

A partir dessas informações, conseguimos observar um comportamento parecido ao encontrado no treinamento do modelo MLP, onde identificamos estabilidade do modelo e no último gráfico, da matriz de confusão do último fold, é possível perceber novamente a dificuldade em classificar as classes intermediárias do modelo (B e C).

5.3. Árvore de Decisão

Foi utilizado o Decision Tree Classifier, da biblioteca sklearn, com critério de decisão sendo a entropia.

Utilizando todos os atributos de entrada e sem limitação para o crescimento da árvore, esses foram os resultados obtidos através da validação cruzada:

```
Acurácia por fold
Conjunto de treino:      [1.0, 1.0, 1.0, 1.0]
Conjunto de validação:  [0.62, 0.62, 0.61, 0.63]

Acurácia média no Conjunto de treino:      1.0
Acurácia média no Conjunto de validação:  0.62

F1-Score macro avg por fold
Conjunto de treino:      [1.0, 1.0, 1.0, 1.0]
Conjunto de validação:  [0.63, 0.62, 0.61, 0.63]

F1-Score médio no Conjunto de treino:      1.0
F1-Score médio no Conjunto de validação:  0.62
```

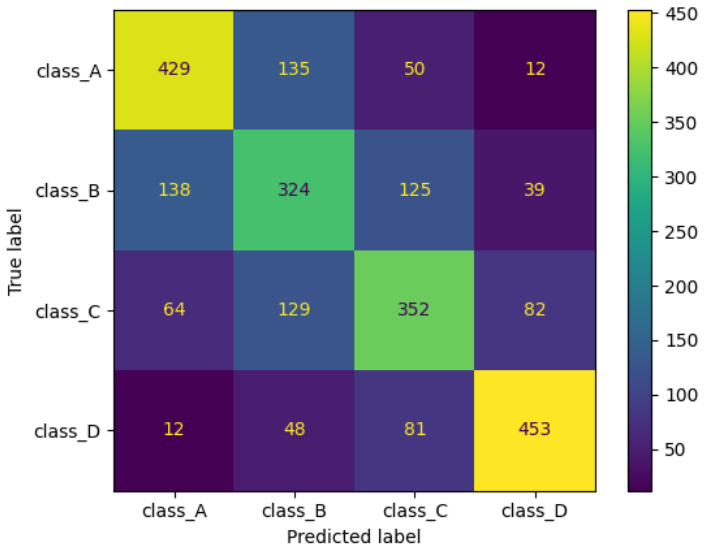

Figura 10. Treinamento da árvore de decisão

A partir dessas informações, podemos observar que o classificador acerta todas as instâncias do conjunto de treinamento, porém isso não se repete para o conjunto de validação, o que caracteriza claramente um overfitting dos dados. Apesar disso, esse modelo é muito mais eficiente em classificar as classes intermediárias do modelo.

A árvore resultante ficou muito grande para ser visualizada, e para evitar overfit, foi realizada uma poda. Por tentativa e erro, um parâmetro de cada vez, foram sendo alterados os

valores de profundidade máxima e de nós máximos da árvore. Ao definir profundidades baixas (entre 5 e 7), o aprendizado ficava limitado e o resultado estava ruim, porém para profundidades um pouco maiores (entre 10 e 12), a árvore se expandia muito horizontalmente. Portanto, a solução encontrada foi deixar a profundidade máxima livre e limitar o número de nós máximos. O número final foi 250 nós. Provavelmente este não é o melhor resultado, pois não foge muito da situação inicial.

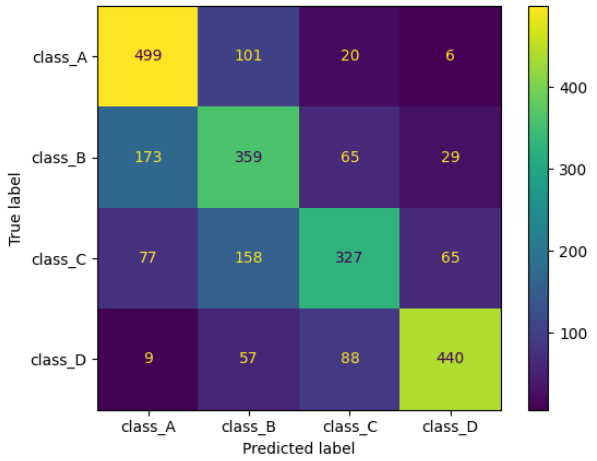

Figura 11. Matriz de confusão da árvore de decisão podada

6. Resultados obtidos

Com todos os modelos já treinados, podemos obter os resultados finais e observar as métricas das predições do conjunto de teste

6.1. Multi Layer Perceptron (melhor modelo)

Analogamente ao que foi apresentado no treinamento, as métricas para as classes intermediárias (B e C) foram bem ruins, como pode ser observado na figura abaixo.

	precision	recall	f1-score	support
0	0.73	0.52	0.61	834
1	0.50	0.00	0.00	836
2	0.18	0.01	0.02	837
3	0.77	0.72	0.75	791
micro avg	0.74	0.31	0.43	3298
macro avg	0.55	0.31	0.34	3298
weighted avg	0.54	0.31	0.34	3298
samples avg	0.30	0.31	0.30	3298

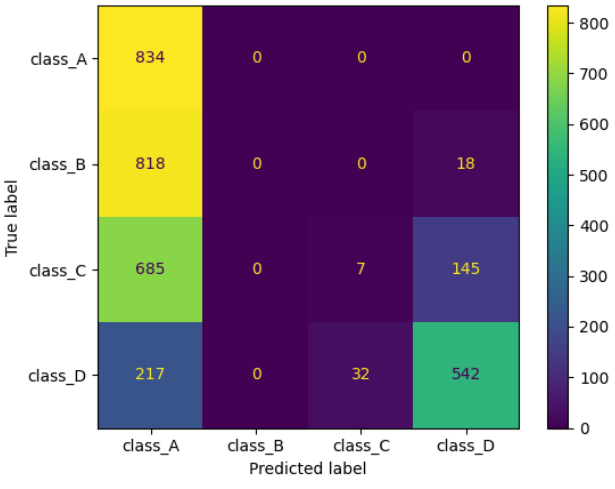

Figura 12. Resultados de teste MLP

6.2. Regressão Logística

Os resultados foram bons para as classes A e D, porém as classes B e C não ficaram muito bem definidas. Muitos elementos de B foram classificados em A ou C, e muitos elementos de C foram classificados em B ou D. Apesar disso, as métricas desse modelo foram melhores do que as obtidas no modelo de rede neural.

	precision	recall	f1-score	support
0	0.64	0.82	0.72	834
1	0.48	0.32	0.38	836
2	0.47	0.39	0.43	837
3	0.64	0.80	0.71	791
micro avg	0.58	0.58	0.58	3298
macro avg	0.56	0.58	0.56	3298
weighted avg	0.56	0.58	0.56	3298
samples avg	0.58	0.58	0.58	3298

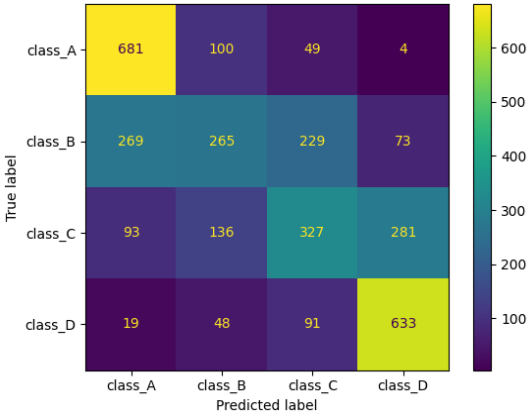

Figura 13. Resultados de teste Regressão Logística

6.3. Árvore de Decisão

Os resultados de uma forma geral foram bons. Assim como no experimento anterior, as classes A e D tiveram um melhor desempenho, porém não chegaram a ser ruins.

	precision	recall	f1-score	support
0	0.68	0.63	0.66	834
1	0.47	0.51	0.49	836
2	0.56	0.57	0.57	837
3	0.77	0.75	0.76	791
micro avg	0.61	0.61	0.61	3298
macro avg	0.62	0.62	0.62	3298
weighted avg	0.62	0.61	0.62	3298
samples avg	0.61	0.61	0.61	3298

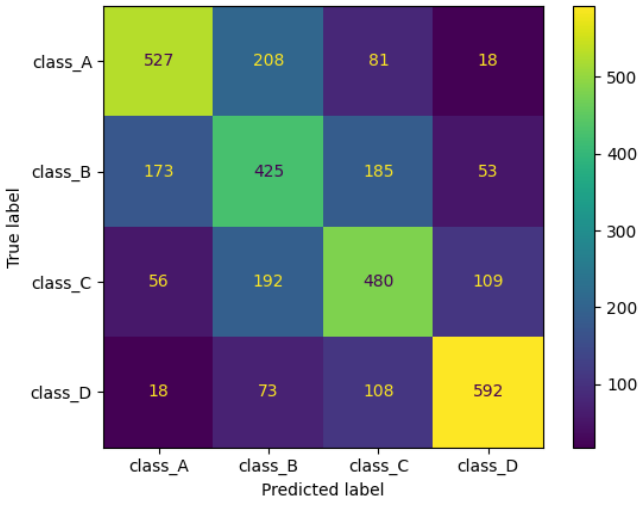

Figura 14. Resultados de teste Árvore de decisão

Para a árvore podada, os resultados não foram muito diferentes, mas pôde-se observar uma tendência de melhorar um pouco a classificação A e B e piorar um pouco C e D.

7. Conclusão

Neste relatório, é exposta a utilização de diversos modelos para resolver problemas de classificação com múltiplas classes. Ao observar os resultados, é claro que o modelo de melhor desempenho foi a árvore de decisão, porém, vale ressaltar que pode existir uma configuração do modelo de redes neurais que performe melhor. Dito isso, o grande desafio está em encontrar a configuração do classificador MLP que atinja esse resultado.

Além disso, ao observar outros modelos de classificação disponibilizados pelos usuários do Kaggle, conseguimos constatar que modelos mais robustos, como random forest e modelos de Boost, conseguem valores consideravelmente mais altos do que os apresentados (por volta de 70% de acurácia, precisão e recall), apesar disso, vale botar na balança o esforço computacional necessário para treinar estes métodos, como é o caso do XGBoost.

Referências

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

[2] Kaggle datasets. (2023). “Body performance Data”. https://www.kaggle.com/datasets/kukuroo3/body-performance-data

[3] Korea Sports Promotion Foundation. (2023). https://www.kspo.or.kr

[4] 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#