Universidade Federal do Rio Grande do Sul - UFRGS
Instituto de Informática
INF01017 - Aprendizado de Máquina
Professora: Mariana Recamonde Mendoza
Fábio de Oliveira Petkowicz 118627
Rafael Júnior Ribeiro 265830

Relatório do Trabalho 1: Treinamento e validação de modelos preditivos com AM

1. Introdução

Este trabalho tem como objetivo desenvolver um modelo baseado em AM para prever a qualidade do vinho branco português, "vinho verde" - bom ou ruim - através de suas características físico-químicas. Para tanto, utilizamos o conjunto de dados **wine-quality-white** disponível em: https://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/winequality-white.csv.

Além disso, conforme a definição, foi construída uma implementação própria da estratégia de avaliação de modelos *Nested Cross Validation* e foram explorados três algoritmos de aprendizado supervisionado: KNN, Floresta Aleatória e Regressão Logística.

2. Dataset

A base original apresenta as seguintes características:

- número de instâncias: 4898
- número de atributos: 11 físico-químicos + 1 atributo de classificação (sensorial)
- variável dependente: qualidade (nota de 0 a 10).
  Quantidade por classe:
  - 6: 2198
  - 5: 1457
  - 7: 880
  - 8: 175
  - 4: 163
  - 3: 20
  - 9: 5

- variáveis independentes: 11 todos numéricos e representando características físico-químicas do vinho:
  a. acidez fixa,
  b. acidez volátil
  c. ácido cítrico,
  d. açúcar residual
  e. cloretos
  f. dióxido de enxofre livre

g. dióxido de enxofre total
h. densidade
i. pH
j. sulfatos
k. álcool

- Não existem valores faltantes.

3. Linguagem utilizada

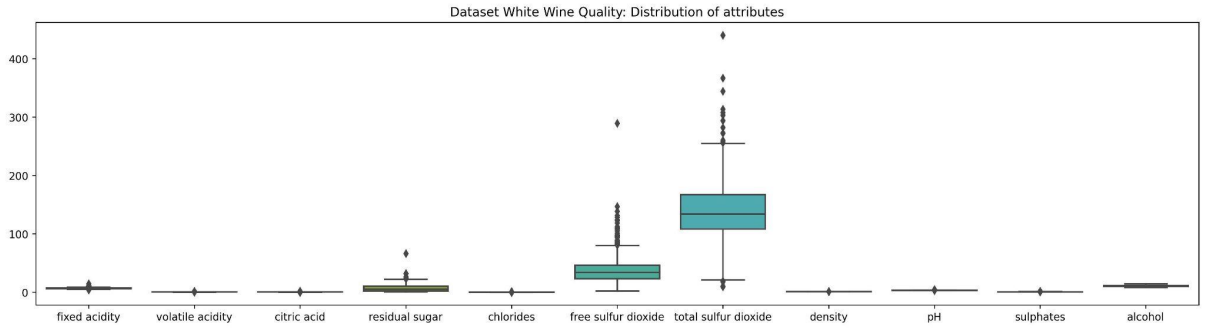
A linguagem escolhida foi Python 3 em função da variedade de bibliotecas disponíveis para manipulação e análise de dados. Além de recursos *built-in* da linguagem, foram usadas as seguintes bibliotecas: pandas, scikit-learn, numpy, seaborn e matplotlib. Jupyter Notebook foi utilizada para análise exploratória, criação dos plots e análise do resultado final. Ao lado do arquivo deste relatório há um arquivo chamado **README.md** com instruções de como configurar o ambiente.

4. Desenvolvimento

a. Pré-processamento

A classificação inicial do atributo *quality* era multivalorada, classificando os vinhos com valores de 0 a 10. Com a finalidade de simplificar a análise a classificação foi convertida em dois *labels* “ruim” para valores até 6 e “bom” para valores acima de 6.

Os atributos independentes possuíam escalas muito diferentes conforme mostrado no gráfico abaixo (clique no link da imagem para visualizar melhor). Por isso eles foram normalizados para a escala entre 0 e 1, usando o método MinMax.



b. Criação dos folds

Foram realizados 2 experimentos que se diferenciam apenas no seu início, na criação dos folds:

- Experiência com folds balanceados. O script realiza um loop inserindo em cada fold sempre uma tupla contendo 1 exemplo da classe “ruim” e 1 exemplo da classe “bom”. O script termina quando não há mais exemplares de alguma classe para inserir nos folds. O que resta da outra classe é descartado. Pode acontecer de algum fold conter 1 tupla (2 exemplos) a menos que os demais.
- Experiência com folds mantendo o balanceamento original do dataset que era aproximadamente 3 exemplos de vinho ruim para cada 1 exemplo de vinho bom. O script realiza um loop inserindo em cada

fold sempre uma quádrupla contendo 3 exemplos da classe “ruim” e 1 exemplo da classe “bom”. O script termina quando não há mais exemplares de alguma classe para inserir nos folds. O que resta da outra classe é descartado. Pode acontecer de algum fold conter 1 quádrupla (4 exemplos) a menos que os demais.

A partir daqui ambos experimentos continuam iguais, se diferenciando apenas nos resultados finais. Para os experimentos foram criados 30 folds para validação cruzada e 29 para a otimização de hiperparâmetros. Cada fold contém uma combinação distinta para treino e teste.

c. Validação cruzada

Passando por todas as combinações de 29 folds para treino e 1 fold para teste, com os dados contidos nos folds de treinamento é realizada a otimização de hiperparâmetros (explicada abaixo). O melhor conjunto de hiperparâmetros é usado para instanciar a classe classificadora, que então é treinada com os dados contidos no fold de treino e avaliada com os folds de teste e também o de treino. As métricas acurácia, precisão, recall e F1 score das 2 avaliações forão guardadas num csv na pasta data/results. Os classificadores que possuírem o desempenho melhor ou mediano são também salvos na pasta pickle.

d. Otimização de hiperparâmetros

Etapa aninhada a cada iteração da validação cruzada. Com um dicionário contendo os possíveis valores de hiperparâmetros para a classe classificadora são escolhidos 15 combinações aleatórias de hiperparâmetros. Para cada combinação serão criados 29 folds, 28 para treino e 1 para teste. O classificador é instanciado com a combinação de hiperparâmetros, treinado com os dados dos folds de treino, e avaliado com o fold de teste, com seu F1 score sendo guardado.

Ao final é computado a média de F1 scores das 29 iterações de folds. A combinação de hiperparâmetros que possuir a melhor média é escolhida como melhor combinação e retornada para a validação cruzada.

Hiperparâmetros possíveis para cada opção de classificador:

i. KNN:
    1. número de vizinhos: 5, 7, 11, 15, 21
    2. métrica de distância: euclidean, manhattan, minkowski
ii. Regressão Logística:
    1. penalidade (regularização): L1, L2
    2. c (inverso da regularização): 100, 10, 1.0, 0.1, 0.01
iii. Floresta Aleatória:
    1. quantidade de estimadores: 50, 100, 200, 300
    2. critério para medir qualidade do split: gini, entropy, log_loss

5. Pseudocódigo

Devido a quantidade de código próprio escrito para o trabalho, a visualização do relatório seria prejudicada se o pseudocódigo fosse colocado aqui. Devido isso ele foi colocado num arquivo próprio chamado **PSEUDOCODIGOS.pdf** ao lado do arquivo deste relatório.

6. Resultados obtidos - Síntese geral

a. O que mais impactou no desempenho geral, tuning e validação em todos os classificadores, foi o balanceamento dos folds. Quanto mais balanceado, melhor.

b. Nos folds desbalanceados, entre precisão e recall, é o recall que tem mais significância na piora do F1 score. É o recall que aumenta a escala do eixo Y nos plots, trazendo para valores até próximos de 0.2. Como há menos classes positivas (vinhos bons) nos folds, os modelos enviesados aumentam a quantidade de falsos negativos.

c. Na etapa de tuning, poucos hiperparâmetros conseguiram se destacar em relação a uma escolha aleatória. Em Floresta Aleatória, uma quantidade maior de estimadores parece impactar positivamente no resultado final, mas não o suficiente para justificar seu uso, já que a quantidade de estimadores é proporcional ao tempo necessário para treinar e testar. No KNN, o fato dos dados estarem balanceados ou não parece estar relacionado de forma oposta com o número de vizinhos.

d. Em relação ao melhor classificador, Floresta Aleatória possui todos os seus quartis com valores maiores em relação aos respectivos quartis dos demais classificadores, tendo na validação uma mediana de F1 score em 0.84. KNN possui 0.73 e Regressão Logística 0.71.
   Esse melhor desempenho de Floresta Aleatória vem com um fato negativo já que, conforme mostrado no seu plot de tuning, para aumentar o desempenho é necessário aumentar a quantidade de estimadores, aumentando assim também o tempo de treino e teste. Há um tradeoff de o quão benéfico é aumentar uns poucos décimos no desempenho a custo de tempo.

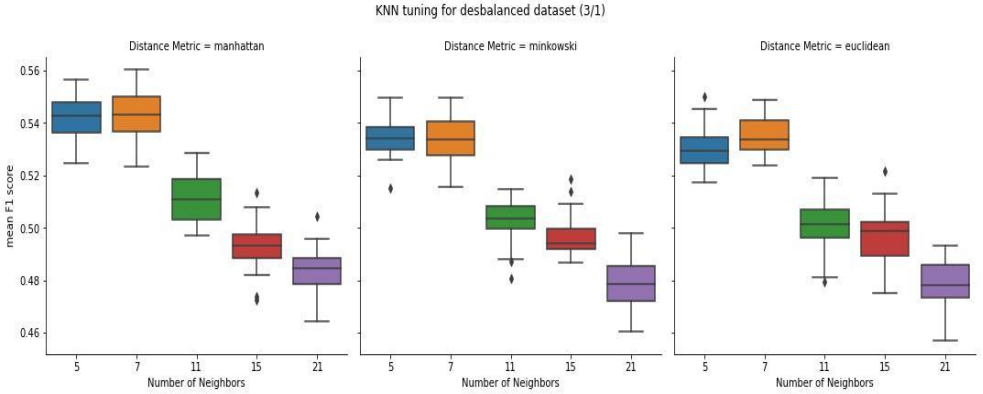
e. Em relação ao classificador com desempenho mais estável, Regressão Logística se destaca, já que na validação com folds balanceados praticamente não houve overfitting. Treino e validação estão com métricas quase iguais, demonstrando assim que o modelo conseguiu generalizar bem.

7. Resultados obtidos - Plots

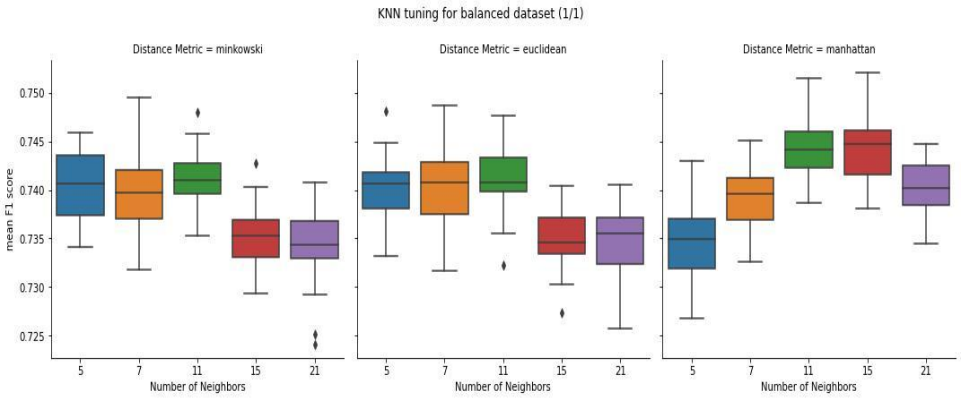
a. KNN

i. Tuning

1. Folds desbalanceados (3x1)

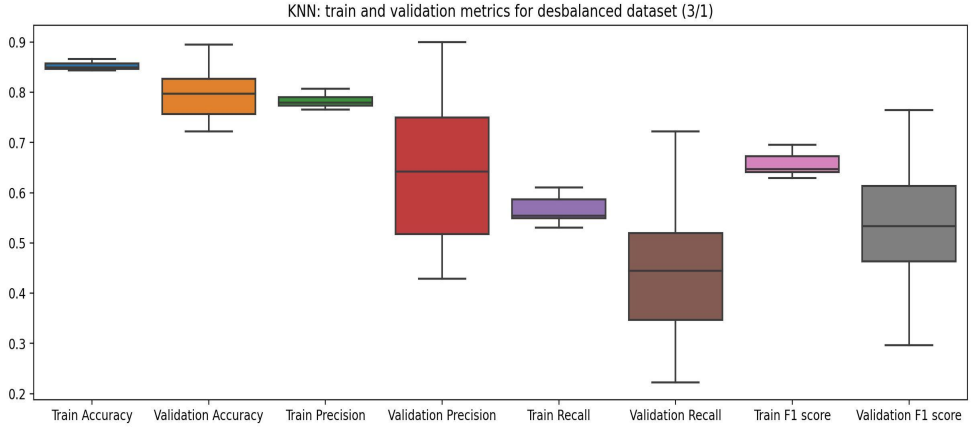


2. Folds balanceados

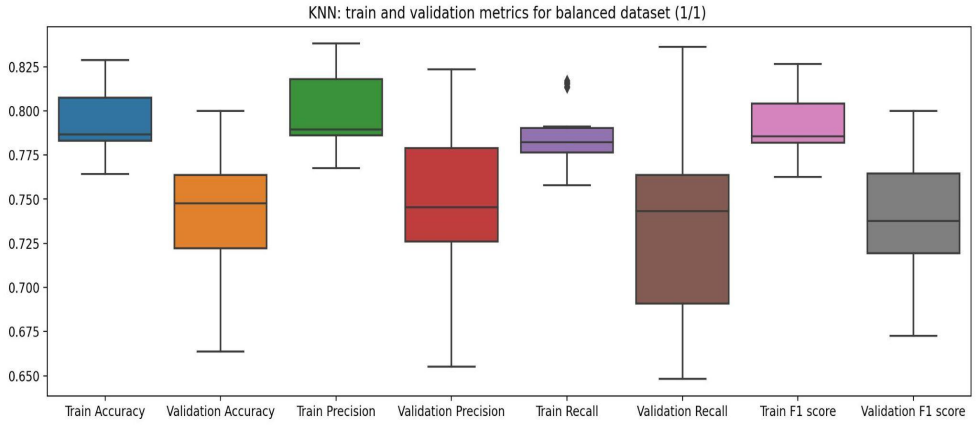


ii. Validação

1. Folds desbalanceados (3x1)



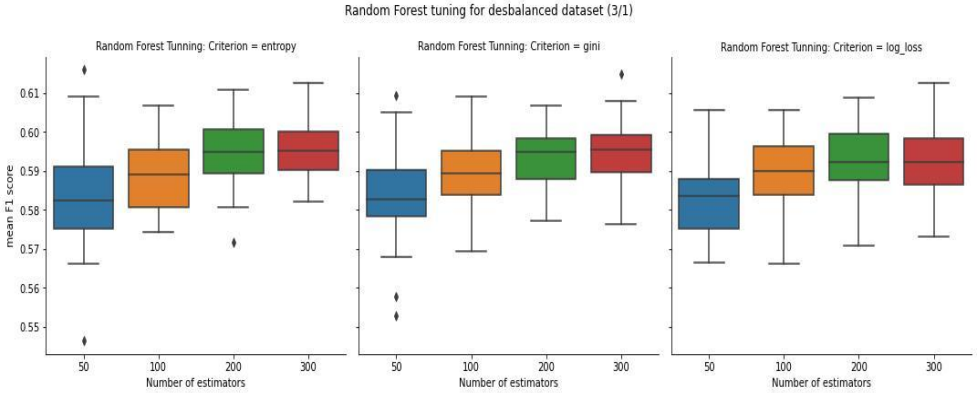
2. Folds balanceados



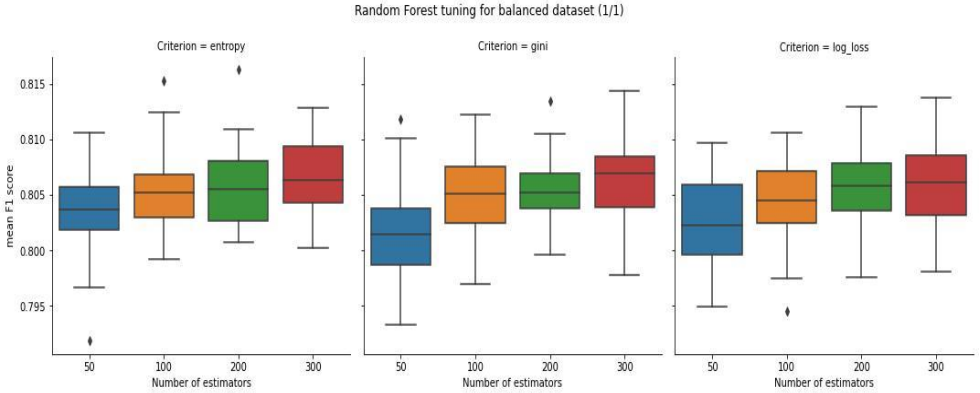
b. Floresta Aleatória

i. Tuning

1. Folds desbalanceados (3x1)

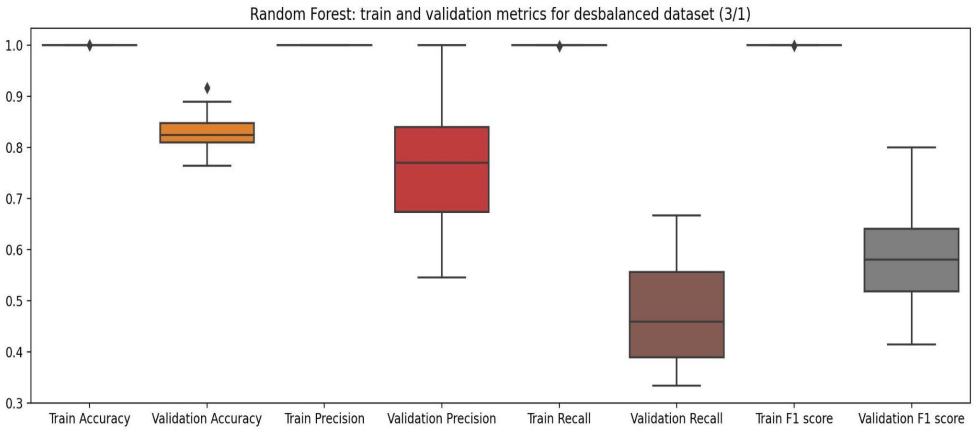


2. Folds balanceados

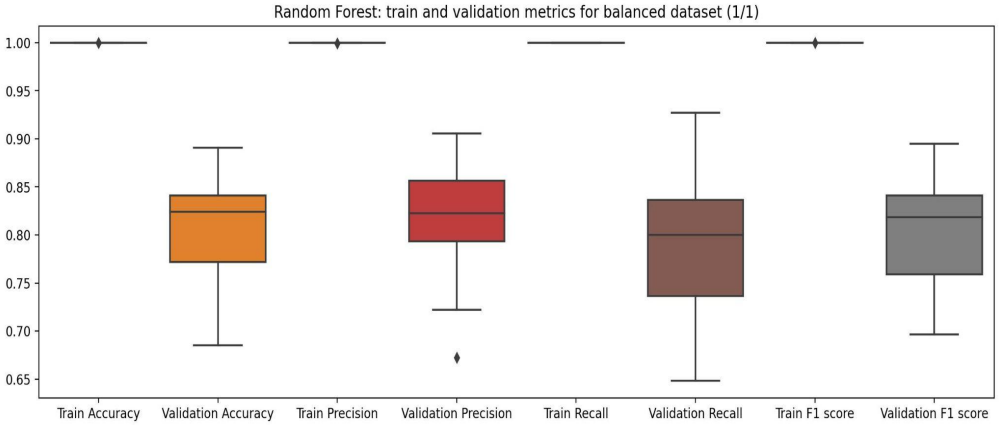


ii. Validação

1. Folds desbalanceados (3x1)



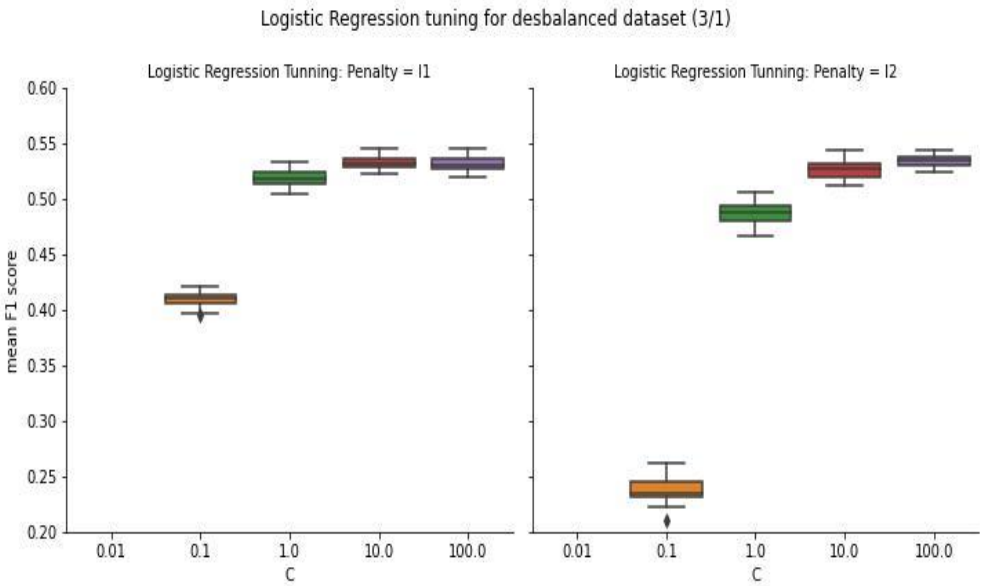
2. Folds balanceados



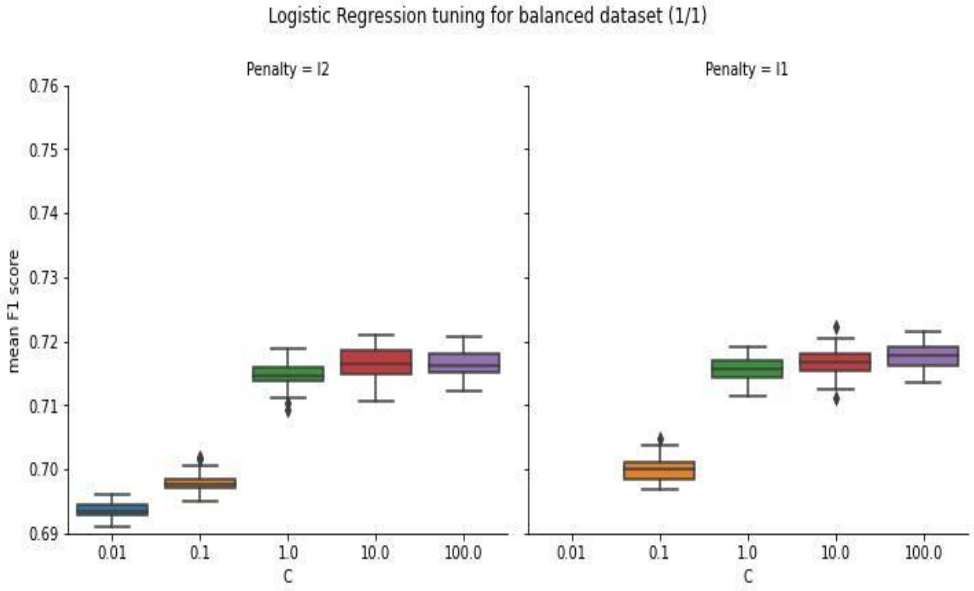
c. Regressão Logística

i. Tuning

1. Folds desbalanceados (3x1)

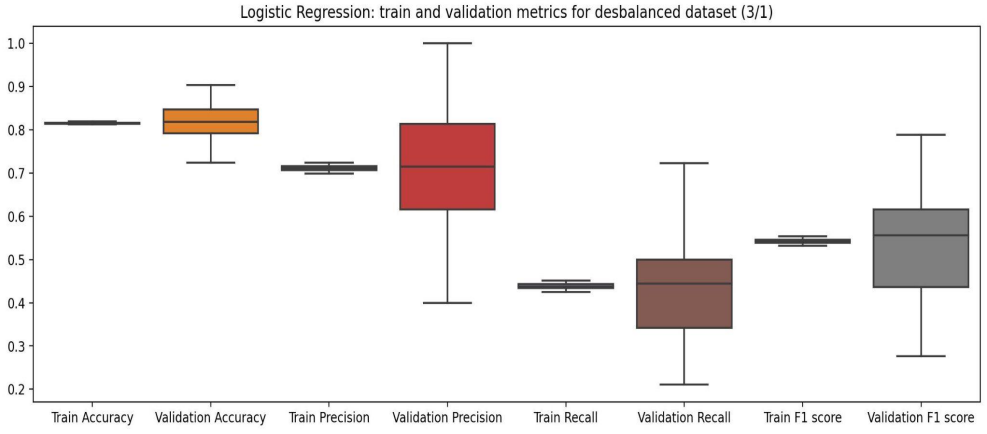


2. Folds balanceados

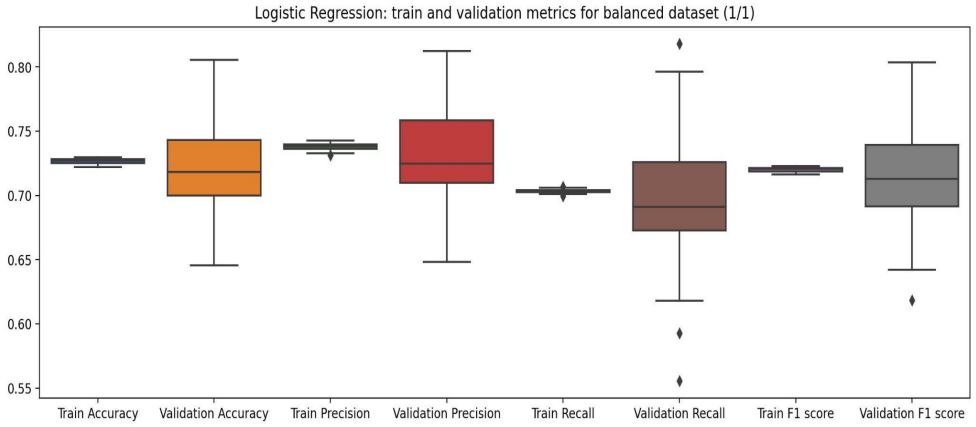


ii. Validação

1. Folds desbalanceados (3x1)



2. Folds balanceados



8. **Conclusão**

Dependendo do dataset o custo computacional de se aplicar K-Fold Cross Validation pode ser muito grande. Entre possuir mais dados - porém desbalanceados - e menos dados - só que balanceados -, a segunda opção provou ter um impacto positivo maior nos métricas de desempenho. Floresta Aleatória é poderoso, porém demanda mais tempo e recursos. Otimização de hiperparâmetros não influenciou muito as métricas de validação, porém isso pode ser porque o espaço de busca não foi grande o suficiente.