

# Trabalho Prático 1 - Computação Natural

Mateus Cursino Gomes Costa - 2022043191

18 de novembro de 2024

## 1 Introdução

Esta é a documentação da implementação de um algoritmo de Programação Genética(GP) baseado nas aulas e nos Slides de Aula[1]. O objetivo desse GP é encontrar uma equação de distância entre pontos de uma base de dados para uma aglomeração por clustering eficiente dos mesmos. O código foi implementado em um notebook python e será descrito ao decorrer do documento.

Foram realizados uma série de testes de parâmetros e diferentes formas de implementação que têm seus prós e contras, o intuito desse documento é explicar e analisar a eficiência das decisões tomadas.

## 2 Implementação

### 2.1 Representação de Indivíduos

Por se tratar de um GP, indivíduos foram representados como strings binárias de até 64 bits. Cada um deles equivale a uma derivação da gramática da Figura 1. Desta forma, usando o mapeamento apresentado em sala na aula, cada string binária é derivada em uma expressão usando de uma árvore derivação com profundidade máxima igual a 7.

### 2.2 Métricas e Fitness

Dado um indivíduo, ou seja, uma expressão aritmética, é calculada uma matriz de distâncias entre todos os pontos da base analisada. Essa matriz é então usada como parâmetro na função AgglomerativeClustering[3] da biblioteca Sklearn para ser feito um clustering dos pontos. A qualidade da aglomeração é usada como Fitness do indivíduo. O cálculo da Fitness é feito com a função 'v measure score'[2], também do Sklearn, usando os labels esperados para cada ponto fornecidos na base e os labels que aquele indivíduo definiu.

### 2.3 Seleção

O processo de seleção usado neste GP é o torneio. Para cada geração de indivíduos são feitos diversos torneios de  $k$  participantes aleatórios, dos quais o vencedor recebe a chance de sofrer mutação e/ou participar de um cruzamento. Devido ao caráter probabilístico desse método, o GP conta com a presença de elitismo. A influência desses métodos no GP será discutida na parte da análise experimental do trabalho.

```
<start> ::= <exp>
           | <exp> <op> <exp>
           | (<exp> <op> <exp>)
<exp>   ::= <exp> <op> <exp>
           | (<exp> <op> <exp>)
           | <protop> ( <term> , <term> )
           | <term>
<op>    ::= + | - | *
<protop>:= div
<term>  ::= x0|x1|...|xn
```

Figura 1: Gramática usada para gerar indivíduos. O número n de termos é igual ao número de variáveis de cada base.

## 2.4 População, Gerações e Parâmetros

A primeira população de cada execução do GP consta com  $p$  indivíduos aleatoriamente escolhidos entre os binários de 0 até  $2^{64}$ . Após o cálculo da Fitness deles ocorre a seleção e a próxima geração é formada pelos  $e$  melhores indivíduos da anterior, além de todos os selecionados que sofreram mutação e pelos seus filhos gerados.

O processo de gerar uma nova população é feito por  $g$  gerações, um torneio tem  $k$  participantes, um vencedor de torneio sofre mutação com probabilidade  $P_m$  e um par de vencedores realiza cruzamento com probabilidade  $P_c$ .

A mutação e cruzamento implementados foram mutação e cruzamento de um ponto. Isso quer dizer que um individuo a ser mutado tem um bit alterado em uma posição aleatória e os filhos de um cruzamento são a mistura das duas strings binarias de seus pais a partir de um ponto aleatório. Esse método tem como consequência a baixa localidade das alterações dos indivíduos e isso será analisado na fase experimental.

## 3 Experimentação e Analise

Para entender e definir um bom conjunto de parâmetros para o GP foram feitos uma série de testes. Tais testes serão listados e analisados nesse documento, eles foram feitos na seguinte ordem:

- Tamanhos da População
- Número de Gerações
- Diferentes Probabilidade de Reprodução
- Influênciia do Elitismo
- Base de teste

As variáveis coletadas em cada geração de cada teste foram:

- Pior, Melhor e Fitness Média
- Filhos Piores e Melhores que os Pais
- Número de Indivíduos Repetidos
- Desvio Padrão da Fitness Média

### 3.1 Tamanhos da População

Foram definidos como variáveis iniciais  $p = x$ ,  $g = 30$ ,  $e = 10$ ,  $P_c = 0,9$ ,  $P_m = 0,05$  e  $k = 2$ , onde  $x$  foi variado com os valores 30, 50, 100 e foram coletados os seguintes resultados:

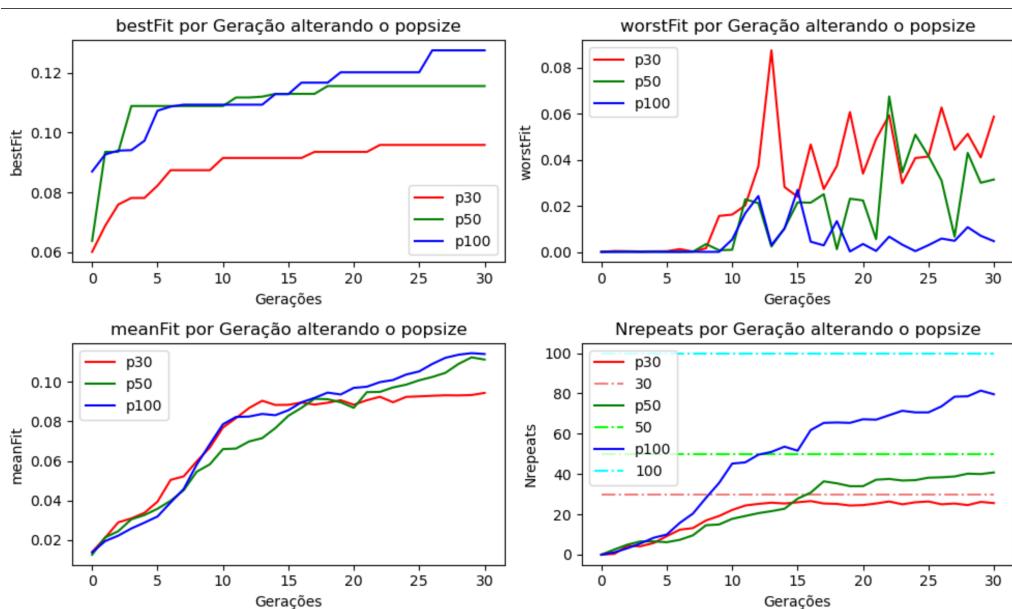


Figura 2: Gráficos dos dados coletados durante os testes da GP

Com esses primeiros gráficos da Figura 2, foi possível perceber que um tamanho de população de 30 indivíduos trazia resultados piores que tamanhos iguais a 50 ou 100. Testes com 100 indivíduos possuíam um número muito grande de repetições, o que torna o processamento mais ineficiente. Portanto, tamanho da população de 50 indivíduos foi o melhor resultado considerando o custo computacional, tempo de execução e desempenho de cada teste.

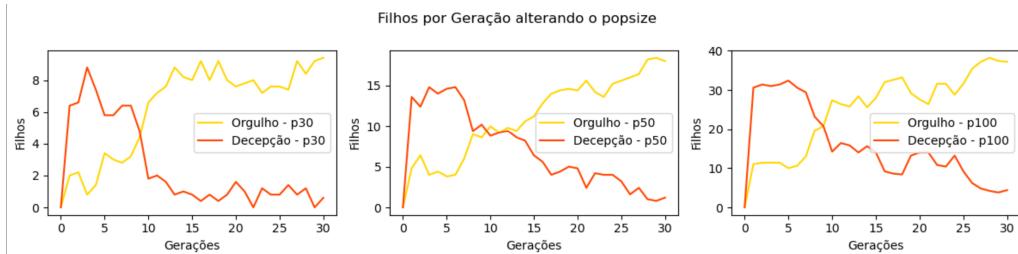


Figura 3: Filhos piores(Decepção) e melhores(Orgulho) que seus pais por geração

Quanto à eficiência dos cruzamentos, analisando a Figura 3, é sempre similar o comportamento do número de filhos piores que os pais (Decepção) e filhos melhores ou iguais aos pais (Orgulho). Inicialmente filhos são majoritariamente piores e após a decima geração eles tendem a ser melhores.

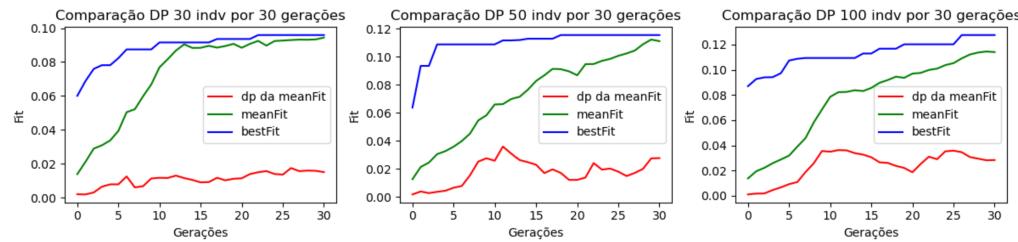


Figura 4: bestFit, meanFit e DP da meanFit por gerações

Por fim, com a Figura 4, podemos analisar o desvio padrão da Fitness média de cada geração dos testes. Foram feitos 5 execuções de cada teste nesse trabalho. O número de execuções idealmente seria maior, porém teve de ser reduzido devido a limitações de poder computacional disponível para esse trabalho. O desvio padrão é relativamente alto quando comparado ao valor da meanFit dos testes, porém isso não quer dizer que os resultados estejam muito distantes do real desempenho do algoritmo implementado.

### 3.2 Número de Gerações

Com  $p = 50$  foi variado o valor de  $g$ , foi novamente usado o conjunto 30,50,100 para os testes com 5 execuções em cada teste.

Ao analisar os resultados presentes nas Figuras 6, 5 e 7, é possível perceber que há uma diferença muito pequena nos resultados encontrados em todos os testes, enquanto o custo computacional e o tempo de execução de cada teste quase dobrou a cada aumento do número de gerações, portanto  $g$  receberá o menor valor testado de 30 gerações.

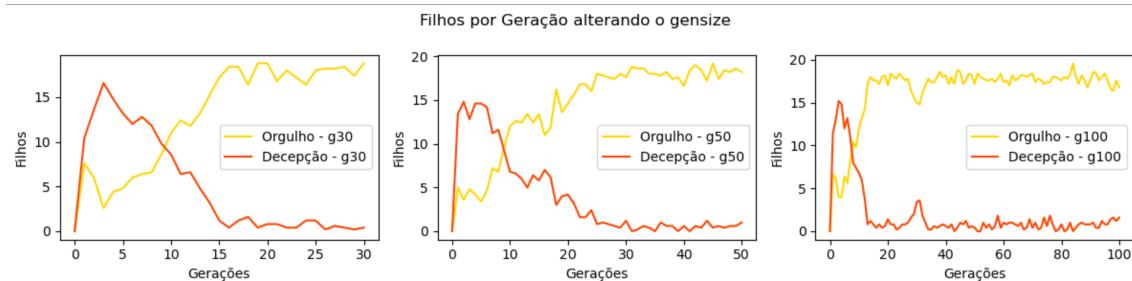


Figura 5: Filhos piores(Decepção) e melhores(Orgulho) que seus pais por geração

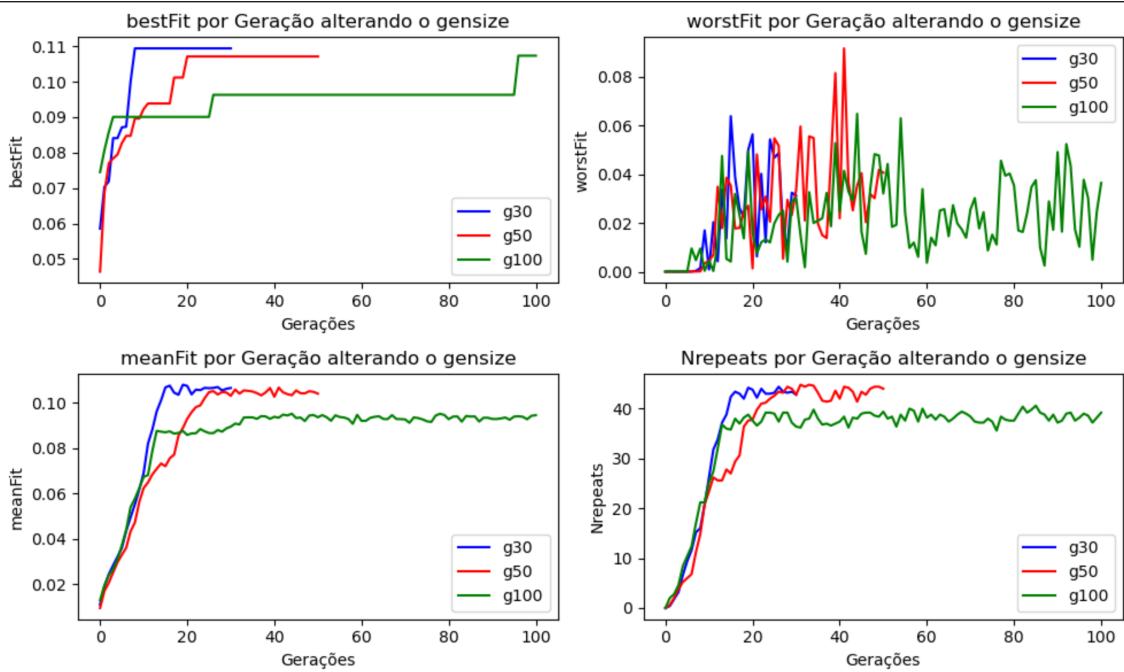


Figura 6: Gráficos dos dados coletados durante os testes da GP

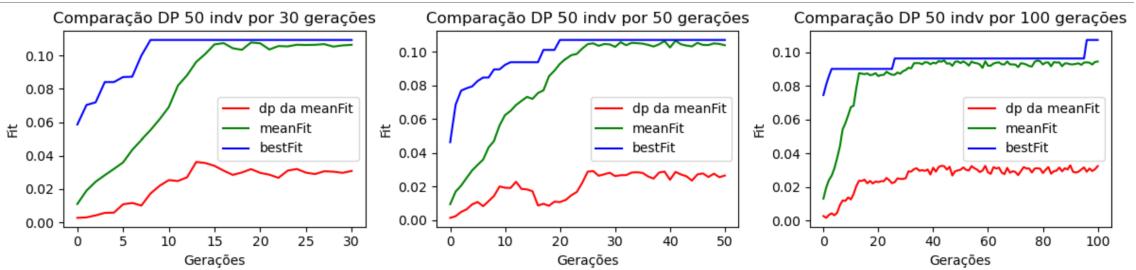


Figura 7: bestFit, meanFit e DP da meanFit por gerações

### 3.3 Diferentes Probabilidade de Reprodução

Como foi mencionado na sessão de Implementação, o cruzamento e mutação em um ponto são operações de baixa localidade, portanto podem ter efeito destrutivo na Fitness dos indivíduos gerados. Logo um teste com esses valores alterados poderia ser extremamente melhor, como foi observado, visto que esse foi o teste que trouxe a maior melhora nos resultados encontrados.

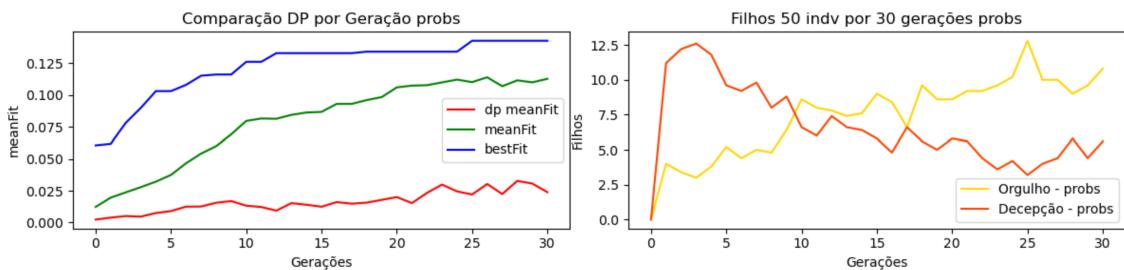


Figura 8: Resultados com,  $p = 50$ ,  $g = 30$ ,  $e = 10$ ,  $P_c = 0.6$ ,  $P_m = 0.3$  e  $k = 2$

Com as chances de cruzamento menores e mutação maiores, houve um crescimento do valor de bestFit de 0,11 para quase 0,14, alcançando a melhor Fitness de todos os testes feitos nessa base.

### 3.4 Influência do Elitismo

Ao longo de todos os testes feitos, o elitismo sempre esteve presente, o que pode ser percebido pelo comportamento da variável bestFit ao longo das gerações que, como sempre havia cópia dos melhores indivíduos da geração passada para a seguinte, sempre se permanecia igual ou melhorava.

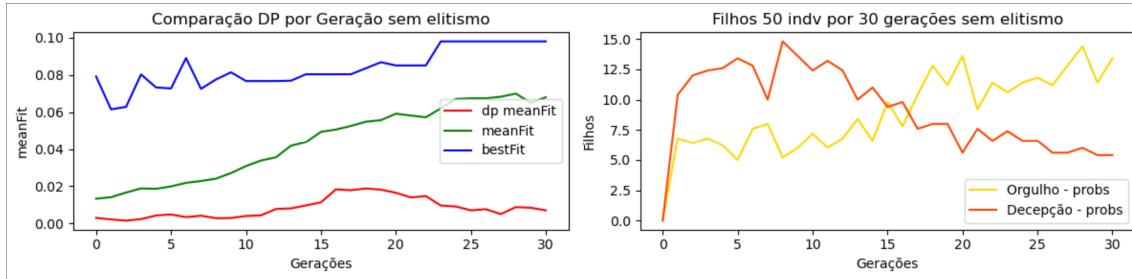


Figura 9: Resultados com,  $p = 50$ ,  $g = 30$ ,  $e = 0$ ,  $P_c = 0,6$ ,  $P_m = 0,3$  e  $k = 2$

Sem elitismo bestFit da geração seguinte pode ser pior do que da geração anterior, e grandes melhorias podem se perder como observado por volta da sexta e sétima gerações. Essas perdas levam a uma convergência mais tardia, aparente na demora maior para que os filhos gerados superassem os pais e na diferença menor entre filhos menores e piores, e possivelmente a uma piora do resultado encontrado, visto que essa foi a pior Fitness encontrada em todos os testes.

### 3.5 Tamanho do Torneio e Pressão Evolutiva

O último parâmetro a ser testado é o tamanho  $k$  de cada torneio realizado.

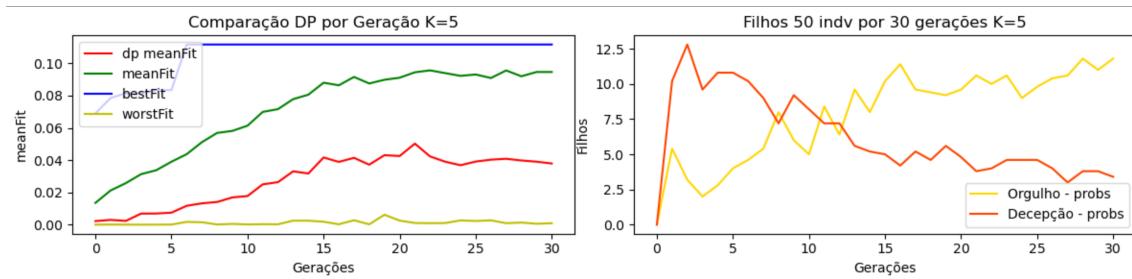


Figura 10: Resultados com,  $p = 50$ ,  $g = 30$ ,  $e = 0$ ,  $P_c = 0,6$ ,  $P_m = 0,3$  e  $k = 5$

A pressão evolutiva é a métrica que avalia o quanto rápido uma evolução converge para um valor de Fitness. Com um  $k$  maior, mais indivíduos participam do torneio de cada vez, fazendo com que o vencedor tenda para um novo ótimo mais rápido. Com  $k = 5$ , obtivemos uma bestFit próxima da melhor encontrada e uma meanFitness maior do que nos testes anteriores. Por isso, manteremos  $K = 5$  para o experimento definitivo com a base de teste.

### 3.6 Base de teste

A base de testes foi executada com o melhor conjunto de parâmetros encontrado ao longo dos testes, exceto pelo elitismo que, devido a um erro de input, não foi utilizado. Apesar dessa falha, os resultados ainda foram surpreendentemente melhores do que o esperado por mim. Uma Fitness de mais de 3 é maior do que o alcançado em todos os testes que realizei, o que demonstra que o algoritmo funciona bem.

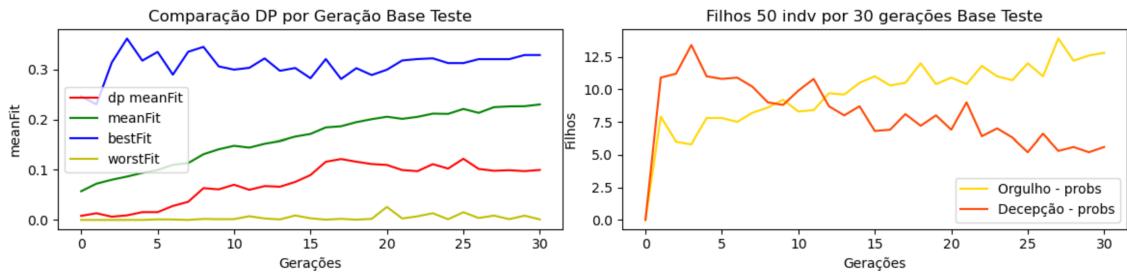


Figura 11: Resultados com,  $p = 50$ ,  $g = 30$ ,  $e = 0$ ,  $P_c = 0,6$ ,  $P_m = 0,3$  e  $k = 5$  Na base de testes

## 4 Segunda Base

### 4.1 Base de Treino

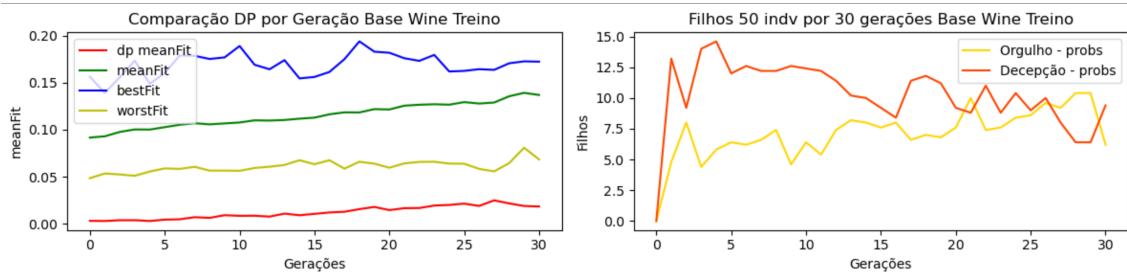


Figura 12: Resultados com,  $p = 50$ ,  $g = 30$ ,  $e = 0$ ,  $P_c = 0,6$ ,  $P_m = 0,3$  e  $k = 5$  na base Wine Train

A segunda base de dados foi rodada inicialmente com os mesmos parâmetros da primeira e foram obtidos bom resultados. Alterações foram feitas nos parâmetros em testes menos detalhados e ainda assim essa configuração se provou a mais consistente. Portanto, essa mesma configuração será usada na base de teste. O erro de input do elitismo também foi cometido nesse e no próximo teste.

### 4.2 Base de Teste

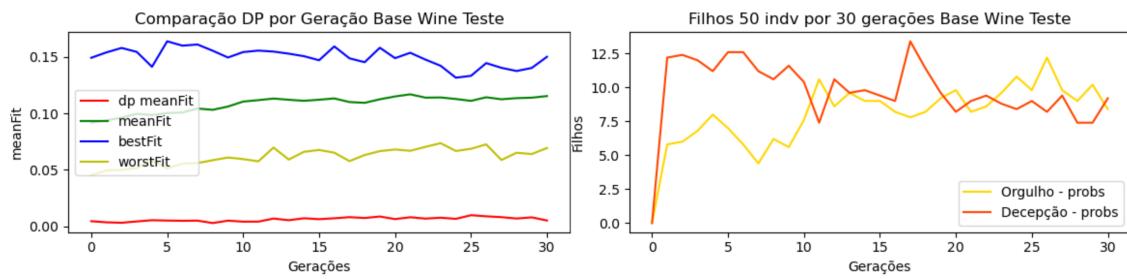


Figura 13: Resultados com,  $p = 50$ ,  $g = 30$ ,  $e = 0$ ,  $P_c = 0,6$ ,  $P_m = 0,3$  e  $k = 5$  na base Wine Test

Por fim, a base de teste foi utilizada e os resultados, apesar de um pouco piores, também não foram ruins, especialmente comparados à primeira base de testes. Foi possível perceber que entre conjunto de treino e teste os resultados foram consistentes ao usar os mesmos parâmetros nas duas bases.

## Referências

- [1] Gisele L. Pappa. Slides de aula de computação natural. 2024.
- [2] SkLearn. Agglomerativeclustering. [https://scikit-learn.org/dev/modules/generated/sklearn.metrics.v\\_measure\\_score.html](https://scikit-learn.org/dev/modules/generated/sklearn.metrics.v_measure_score.html).

- [3] SkLearn. Agglomerativeclustering. <https://scikit-learn.org/dev/modules/generated/sklearn.cluster.AgglomerativeClustering.html>.