AVALIAÇÃO – APRENDIZADO DE MÁQUINA

Rodrigo Barroso Rodrigues \_ 219031139

**Índice**

1. [Introdução 4](#_TOC_250040)
   1. [Descrição da Avaliação 4](#_TOC_250039)
      1. [Dataset 4](#_TOC_250038)
      2. [Sobre os dados de treino e teste 4](#_TOC_250037)
      3. [Sobre o tratamento dos dados](#_TOC_250036) 4
      4. [Sobre as features do dataset](#_TOC_250035) 5
      5. [Introdução dos modelos treinados 7](#_TOC_250034)
2. [Avaliações dos Modelos 9](#_TOC_250032)
   1. [Como foi feita a coleta? 9](#_TOC_250031)
   2. [Como foi feita a comparação de cada algoritmo ? 9](#_TOC_250030)
3. [Resultados 11](#_TOC_250029)
   1. [Caminho percorrido pelo HillClimb: 11](#_TOC_250028)
   2. [Caminho percorrido pelo Best First: 12](#_TOC_250027)
   3. [Caminho percorrido pelo Branch and Bound: 13](#_TOC_250026)
   4. [Caminho percorrido pelo A\*: 15](#_TOC_250025)
4. [Dificuldades: 17](#_TOC_250024)
   1. [Limitações: 17](#_TOC_250023)
      1. [Branch and Bound: 17](#_TOC_250022)
   2. [Dificuldades e como superamos: 17](#_TOC_250021)
      1. [Swish.SWI-Prolog: 17](#_TOC_250020)
      2. [Problema na implementação: 17](#_TOC_250019)
5. [Comparação dos algoritmos: 18](#_TOC_250018)
   1. [Gráfico de Posições Percorridas: 18](#_TOC_250017)
      1. [Custo dos movimentos: 18](#_TOC_250016)
   2. [Gráfico do Tempo de Execução: 19](#_TOC_250015)
      1. [Tempo de execução: 19](#_TOC_250014)
   3. [Conclusão: 20](#_TOC_250013)
6. [Desenvolvimento das Atividades: 21](#_TOC_250012)
   1. [Divisão de trabalho: 21](#_TOC_250011)
      1. [Programação: 21](#_TOC_250010)
      2. [Relatório: 22](#_TOC_250009)
      3. [Apresentação: 24](#_TOC_250008)
   2. [Ferramentas: 25](#_TOC_250007)
      1. [Comunicação: 25](#_TOC_250006)
      2. [Programação: 25](#_TOC_250005)
      3. [Relatório e Apresentação: 26](#_TOC_250004)
   3. [Contribuição individual: 26](#_TOC_250003)
7. [Links e Referências Bibliográficas: 28](#_TOC_250002)
   1. [Links: 28](#_TOC_250001)
      1. [Links para auxílio: 28](#_TOC_250000)

# Introdução

## Descrição da Avaliação

### Dataset

O dataset utilizado para a seguinte avaliação de aprendizado de máquina foi obtido na plataforma Kaggle através do seguinte [link](https://www.kaggle.com/datasets/iabhishekofficial/mobile-price-classification). Ele consiste em uma série de dados referentes a especificações de aparelhos de telefonia móvel (smartphones), e a partir desses dados, foi inferido uma coluna chamada de “price\_range”, que guarda o valor de 4 classes, com valores inteiros de 0 a 3, onde cada valor inteiro representa um intervalo de preços para cada aparelho celular baseado em suas específicações, sendo 0 o mais barato até 3 sendo os aparelhos mais caros.

### Sobre os dados de treino e teste

O dataset “Mobile Price Classification” contém 21 features, sendo 1 delas, “price\_range”, que será utilizado como o target para o aprendizado de máquina. Apesar desse conjunto apresentar um arquivo CSV tanto para treino quanto para teste, o arquivo de teste não conta com a coluna “price\_range”, o que faz com que não seja possível utilizá-lo em um aprendizado supervisionado.

A solução que optei por utilizar foi utilizar apenas o arquivo CSV de treino, porém dividindo-o através do código Python para que 70% dele fosse utilizado para treino e os 30% restantes como teste.

### Sobre o tratamento dos dados

Todas as features presentes no dataset utilizado estão representadas como valores do tipo int64 ou float64 sem precisar fazer nenhum tratamento prévio como a aplicação de Label Encoding ou One-Hot Encoding para termos apenas valores númericos.

Além disso, todas as linhas do conjunto estão preenchidas, ou seja, nenhuma coluna apresentou valores em branco ou nulo, esses que também precisariam ser tratados antes de serem aplicados aos modelos de aprendizado de máquina.

### Sobre as features do dataset:

O dataset em questão apresenta 21 features:

- “battery\_power”, do tipo int64, contendo a quantidade em miliampere-hora da bateria do aparelho celular.

- “blue”, do tipo int64, contendo apenas valores binários (0 ou 1) para a existência (1) ou não (0) de adaptador bluetooth no aparelho.

- “clock\_speed”, do tipo float64, contendo a velocidade na qual o microprocessador do celular em questão executa instruções.

- “dual\_sim”, do tipo int64, contendo apenas valores binários (0 ou 1) para a existência (1) ou não (0) de 2 slots para chips de operadoras de telefonia móvel no aparelho.

- “fc”, do tipo int64, contendo a quantidade de megapixels da câmera frontal do aparelho.

- “four\_g”, do tipo int64, contendo apenas valores binários (0 ou 1) para a existência (1) ou não (0) da tecnologia 4G no aparelho.

- “int\_memory”, do tipo int64, contendo a quantidade de memória interna em gigabytes do aparelho.

- “m\_dep”, do tipo float64, contendo a quantidade em centímetros da espessura lateral do aparelho celular.

- “mobile\_wt”, do tipo int64, contendo o peso em gramas do aparelho celular.

- “n\_cores”, do tipo int64, contendo a quantidade de núcleos do processador do aparelho.

- “pc”, do tipo int64, contendo a quantidade de megapixels da câmera primária do aparelho.

- “px\_height”, do tipo int64, contendo a resolução em pixels da altura do aparelho.

- “px\_width”, do tipo int64, contendo a resolução em pixels da largura do aparelho.

- “ram”, do tipo int64, contendo a quantidade de memória RAM em megabytes do celular.

- “sc\_h”, do tipo int64, contendo a altura da tela do aparelho em centímetros.

- “sc\_w”, do tipo int64, contendo a largura da tela do aparelho em centímetros.

- “talk\_time”, do tipo int64, contendo a quantidade em horas que uma carga completa na bateria consegue manter o telefone ligado durante uma chamada telefônica.

- “touch\_screen”, do tipo int64, contendo apenas valores binários (0 ou 1) para a existência (1) ou não (0) da tecnologia touch screen no aparelho.

- “wifi”, do tipo int64, contendo apenas valores binários (0 ou 1) para a existência (1) ou não (0) da tecnologia WiFi no aparelho.

- “price\_range”, do tipo int64, sendo a target variable, contém os valores de 0 a 4, com 0 representando um aparelho celular de baixo custo, 1 sendo de médio custo, 2 de alto custo e 3 de altíssimo custo.

### Introdução dos modelos treinados:

Os seguintes modelos de aprendizado de máquina foram utilizados para o dataset utilizado nessa avaliação: Árvores de Decisão, Random Forest e MLP (Multilayer Perceptron).

- Árvores de Decisão: É um modelo baseado em árvores binárias para classificação ou regressão. Cada nó da árvore representa uma decisão baseada em uma condição nos dados. É um modelo de fácil interpretação, e não requer normalização dos dados, entretanto, é mais propício a overfitting, principalmente em árvores mais profundas.

- Random Forest: São um ensemble de árvores de decisão, onde múltiplas árvores são treinadas em subconjuntos dos dados com substituição (bootstrapping) e combina os resultados por votação (classificação) ou média (regressão). É um modelo que reduz o overfitting, melhorando a precisão, porém que pode ser bastante lento em cenários com muitas árvores.

- Multilayer Perceptron (MLP): É um modelo de rede neural feedforward que utiliza camadas densamente conectadas (fully connected). É adequado para resolver problemas não lineares. É vantajoso pois permite uma alta flexibilidade na captura de relações complexas nos dados , entretanto, requer mais dados e poder computacional, é mais sensível em relação aos parâmetros setados como a a taxa de aprendizado e a quantidade de neurônios utilizada e pode levar tempos de treino maiores.

# Avaliações dos modelos

### Introdução

Para treinar os modelos em questão citados acima, optei por utilizar a ferramenta ‘GridSearchCV’ do Scikit-learn, para que pudesse ser feita busca matricial (grid search), com o objetivo de fazer uma validação cruzada de valores para os parâmetros dos modelos. Ao encontrar a melhor combinação de hiperparâmetros para cada modelo, utilizei-os para o fit e o predict de cada modelo.

Além da ferramenta GridSearchCV do pacote Scikit-learn, também foram utilizadas as bibliotecas “time”, para poder calcular o tempo de treino e teste dos modelos, a biblioteca “numpy” para calcular a distribuição de classes da predição de cada modelo, a biblioteca “pandas”, utilizada para transformar o CSV do dataset em um DataFrame para ser utilizado para o treino e teste dos modelos.

Em relação as ferramentas do Scikit-learn, foram utilizadas:

- “sklearn.model\_selection.train\_test\_split”, para dividir o dataset na proporção 70% para treino, 30% para teste, com o random\_state = 42.

- “sklearn.metrics.accuracy\_score”, para calcular a acurácia da predição do modelo nos testes após o treino.

- “sklearn.metrics.f1\_score”, para calcular o F1 score da predição do modelo nos testes após o treino.

- “sklearn.metrics.precision\_score”, para calcular a pontuação da precisão da predição do modelo nos testes após o treino.

- “sklearn.metrics.recall\_score”, para calcular a pontuação de recall da predição do modelo nos testes após o treino.

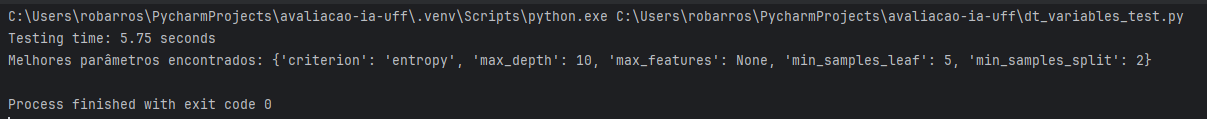
- “sklearn.metrics.confusion\_matrix”, para gerar a matriz de confusão da predição do modelo nos testes após o treino.

As pontuações de acurácia, F1, precisão, recall e matriz de confusão foram utilizados para avaliar a performance de cada modelo utilizado nessa avaliação.

### Árvores de Decisão (Decision Tree)

No modelo Decision Tree, utilizei a biblioteca DecisionTreeClassifier do pacote sklearn.tree do Scikit-learn.

Após testar uma série de combinações possíveis de parametrização através de validações cruzadas com a biblioteca GridSearchCV, o resultado foi que a melhor combinação para o treino do modelo de árvore de decisão seria:



- Para a lógica de divisão, com o objetivo de medir a qualidade da divisão, foi utilizado o parâmetro “criterion” como “entropy”. Foi escolhido ao invés dos valores “gini” ou “log\_loss”. O impacto dessa decisão influencia como a árvore vai dividir os nós na hora do treino.

- Para a lógica de limite do tamanho da árvore, com o intuito de ajudar a prevenir o overfitting , foram usados os parâmetros:

- “max\_depth”: 10, que mantém fixo a profundidade máxima da árvore em 10 níveis

- “min\_samples\_split”: 2, sendo o número mínimo de amostras do conjunto de dados necessária para dividir um nó interno. Por padrão, esse valor já é 2.

- “min\_samples\_leaf”: 5, sendo o número mínimo de amostras necessárias nos nó folhas.

- “max\_features”: None, sendo o número de features usadas na melhor divisão dos nós. Nesse caso, sendo None, todas as features do dataset foram consideradas.

- E para a randomização, foi utilizado o “Random\_s= 42, pois é um valor que garante a reprodutividade dos resultados,

# Resultados

### Caminho percorrido pelo HillClimb:

* + 1. **Matriz A:**

[[0,0],[1,1],[0,2],[1,2],[2,3],[1,2],[1,1],[1,0],[2,0],[2,1],[2,2],[2,3],[2,4]]

* + 1. **Matriz B:**

[[0,0],[0,1],[0,2],[0,3],[1,2],[1,1],[2,0],[3,1],[2,2],[3,2],[4,2],[5,1],[4,0],[5,1],[5,2],[4,3],[5,4],[5,5]]

* + 1. **Matriz C:**

[[0,0],[0,1],[0,2],[0,3],[1,2],[1,1],[2,1],[2,2],[1,3],[2,4],[1,5],[1,6],[0,7],[1,7],[1,8],[2,7],[2,8],[3,7],[

3,6],[4,6],[4,7],[4,8],[5,8],[6,8],[6,7],[5,6],[5,5],[4,4],[3,4],[4,3],[5,2],[4,1],[3,1],[4,2],[5,3],[5,4],[6,

3],[6,2],[7,2],[8,2],[7,1],[7,0],[7,1],[7,2],[7,3],[7,4],[6,5],[7,5],[7,6],[8,6],[8,7],[8,8]]

* + 1. **Matriz D:**

[[0,0],[0,1],[0,2],[0,3],[0,4],[1,3],[1,2],[1,1],[1,0],[2,1],[3,0],[3,1],[3,2],[4,2],[5,3],[5,4],[5,5],[5,6],[

6,6],[7,6],[7,7],[6,8],[5,8],[4,9],[3,10],[2,9],[1,9],[0,9],[0,10],[0,9],[0,8],[0,7],[1,6],[1,5],[1,4],[2,4],

[2,5],[2,6],[2,7],[3,6],[3,5],[3,4],[4,3],[5,3],[6,3],[6,4],[7,3],[8,2],[7,1],[6,0],[7,0],[6,0],[5,0],[4,0],[5

,1],[6,2],[7,3],[8,4],[8,5],[8,6],[7,7],[6,8],[5,9],[6,10],[7,9],[8,9],[8,10]]

* + 1. **Matriz E:**

[[0,0],[0,1],[1,0],[2,1],[2,2],[3,1],[3,0],[4,1],[4,2],[3,3],[2,4],[1,4],[0,3],[1,4],[1,5],[1,6],[1,7],[0,8],[

1,8],[2,8],[3,7],[3,6],[3,5],[4,5],[5,4],[6,3],[6,4],[6,5],[5,6],[4,7],[4,8],[5,7],[6,6],[7,5],[7,4],[7,3],[7,

2],[6,1],[6,0],[7,0],[8,0],[9,0],[9,1],[8,2],[8,3],[9,2],[10,1],[10,0],[11,0],[12,0],[13,0],[14,0],[15,0],[

16,0],[17,1],[16,2],[15,2],[15,3],[16,4],[17,3],[17,4],[18,3],[18,4],[18,5],[17,6],[18,6],[19,6],[19,7],

[18,8],[17,8],[16,8],[15,8],[14,7],[13,6],[13,7],[14,8],[15,7],[15,6],[14,5],[14,4],[13,3],[12,3],[11,2

],[12,2],[13,2],[12,3],[11,4],[10,5],[9,6],[8,6],[8,7],[8,8],[9,7],[9,8],[10,7],[11,7],[11,6],[11,5],[12,4

],[13,3],[13,2],[14,1],[15,0],[16,0],[17,0],[18,0],[19,0],[19,1],[19,2],[19,3],[19,4],[19,5],[19,6],[19,

7],[19,8]]

### Caminho percorrido pelo Best First:

* + 1. **Matriz A:**

[[0,0],[1,1],[0,2],[1,2],[2,3],[1,2],[1,1],[1,0],[2,0],[2,1],[2,2],[2,3],[2,4]]

* + 1. **Matriz B :**

[[0,0],[0,1],[0,2],[0,3],[1,2],[1,1],[2,0],[3,1],[2,2],[3,2],[4,2],[5,1],[4,0],[5,1],[5,2],[4,3],[5,4],[5,5]]

* + 1. **Matriz C :**

[[0,0],[0,1],[0,2],[0,3],[1,2],[1,1],[2,1],[2,2],[1,3],[2,4],[1,5],[1,6],[0,7],[1,7],[1,8],[2,7],[2,8],[3,7],[

3,6],[4,6],[4,7],[4,8],[5,8],[6,8],[6,7],[5,6],[5,5],[4,4],[3,4],[4,3],[5,2],[4,1],[3,1],[4,2],[5,3],[5,4],[6,

3],[6,2],[7,2],[8,2],[7,1],[7,0],[7,1],[7,2],[7,3],[7,4],[6,5],[7,5],[7,6],[8,6],[8,7],[8,8]]

* + 1. **Matriz D:**

[[0,0],[0,1],[0,2],[0,3],[0,4],[1,3],[1,2],[1,1],[1,0],[2,1],[3,0],[3,1],[3,2],[4,2],[5,3],[5,4],[5,5],[5,6],[

6,6],[7,6],[7,7],[6,8],[5,8],[4,9],[3,10],[2,9],[1,9],[0,9],[0,10],[0,9],[0,8],[0,7],[1,6],[1,5],[1,4],[2,4],

[2,5],[2,6],[2,7],[3,6],[3,5],[3,4],[4,3],[5,3],[6,3],[6,4],[7,3],[8,2],[7,1],[6,0],[7,0],[6,0],[5,0],[4,0],[5

,1],[6,2],[7,3],[8,4],[8,5],[8,6],[7,7],[6,8],[5,9],[6,10],[7,9],[8,9],[8,10]]

* + 1. **Matriz E:**

[[0,0],[0,1],[1,0],[2,1],[2,2],[3,1],[3,0],[4,1],[4,2],[3,3],[2,4],[1,4],[0,3],[1,4],[1,5],[1,6],[1,7],[0,8],[

1,8],[2,8],[3,7],[3,6],[3,5],[4,5],[5,4],[6,3],[6,4],[6,5],[5,6],[4,7],[4,8],[5,7],[6,6],[7,5],[7,4],[7,3],[7,

2],[6,1],[6,0],[7,0],[8,0],[9,0],[9,1],[8,2],[8,3],[9,2],[10,1],[10,0],[11,0],[12,0],[13,0],[14,0],[15,0],[

16,0],[17,1],[16,2],[15,2],[15,3],[16,4],[17,3],[17,4],[18,3],[18,4],[18,5],[17,6],[18,6],[19,6],[19,7],

[18,8],[17,8],[16,8],[15,8],[14,7],[13,6],[13,7],[14,8],[15,7],[15,6],[14,5],[14,4],[13,3],[12,3],[11,2

],[12,2],[13,2],[12,3],[11,4],[10,5],[9,6],[8,6],[8,7],[8,8],[9,7],[9,8],[10,7],[11,7],[11,6],[11,5],[12,4

],[13,3],[13,2],[14,1],[15,0],[16,0],[17,0],[18,0],[19,0],[19,1],[19,2],[19,3],[19,4],[19,5],[19,6],[19,

7],[19,8]]

### Caminho percorrido pelo Branch and Bound:

* + 1. **Matriz A:**

[[0,0],[1,1],[0,2],[1,2],[2,3],[2,2],[2,1],[1,0],[2,0],[1,1],[1,2],[1,3],[2,4]]

* + 1. **Matriz B:**

[[0,0],[0,1],[1,2],[0,3],[0,2],[1,1],[2,0],[3,1],[2,2],[3,2],[4,2],[5,1],[4,0],[4,1],[5,2],[4,3],[5,4],[5,5]]

* + 1. **Matriz C:**

[[0,0],[0,1],[1,2],[0,3],[1,3],[2,2],[1,1],[2,1],[3,1],[4,1],[5,2],[4,3],[3,4],[2,4],[1,5],[1,6],[0,7],[1,7],[

1,8],[2,7],[2,8],[3,7],[3,6],[4,6],[4,7],[4,8],[5,8],[6,8],[6,7],[5,6],[5,5],[4,4],[5,4],[6,3],[6,2],[7,2],[8,

2],[7,1],[7,0],[6,1],[5,1],[4,2],[5,3],[6,4],[7,4],[6,5],[7,5],[7,6],[8,6],[7,7],[8,8]]

* + 1. **Matriz D:**

[[0,0],[1,1],[0,2],[0,3],[0,4],[1,3],[1,2],[2,1],[1,0],[0,1],[1,2],[2,2],[3,1],[3,0],[4,0],[4,1],[3,2],[4,2],[5,3],

[5,4],[5,5],[5,6],[6,6],[7,6],[7,7],[6,8],[5,8],[4,9],[3,10],[2,9],[1,9],[0,9],[0,10],[1,10],[2,10],[3,9],[2,8],[

1,7],[0,7],[1,6],[1,5],[1,4],[2,4],[2,5],[2,6],[2,7],[3,6],[3,5],[3,4],[4,3],[5,2],[6,3],[6,4],[7,3],[8,2],[7,1],[

6,0],[7,0],[8,1],[7,2],[8,3],[8,4],[8,5],[8,6],[8,7],[7,8],[6,9],[5,9],[6,10],[7,9],[8,9],[8,10]]

**3.3.4 Matriz E:**

[[0,0],[0,1],[1,0],[2,1],[2,2],[3,1],[3,0],[4,1],[4,2],[3,3],[2,4],[1,4],[0,3],[0,4],[1,5],[1,6],[1,7],[0,8],[

1,8],[2,8],[3,7],[3,6],[3,5],[4,5],[5,4],[6,3],[6,4],[6,5],[5,6],[4,7],[4,8],[5,7],[6,6],[7,5],[7,4],[7,3],[7,

2],[6,1],[6,0],[7,0],[8,0],[9,0],[9,1],[8,2],[8,3],[9,2],[10,1],[10,0],[11,0],[12,0],[13,0],[14,0],[15,0],[

16,0],[17,1],[16,2],[15,2],[15,3],[16,4],[17,3],[17,4],[18,3],[18,4],[18,5],[17,6],[18,6],[19,6],[19,7],

[18,8],[17,8],[16,8],[15,8],[14,7],[13,6],[13,7],[14,8],[15,7],[15,6],[14,5],[14,4],[13,3],[12,3],[11,2

],[12,2],[13,2],[12,1],[11,1],[10,2],[11,3],[11,4],[10,5],[9,6],[8,6],[8,7],[8,8],[9,7],[9,8],[10,7],[11,7

],[10,6],[11,5],[12,4],[13,5],[14,6],[15,5],[15,4],[16,3],[17,2],[18,1],[18,0],[19,0],[19,1],[19,2],[19,

3],[19,4],[19,5],[19,6],[18,7],[19,8]]

### Caminho percorrido pelo A\*:

* + 1. **Matriz A:**

[0,0],[1,1],[0,2],[1,2],[2,3],[1,2],[1,1],[1,0],[2,0],[2,1],[2,2],[2,3],[2,4]]

* + 1. **Matriz B:**

[[0,0],[0,1],[0,2],[0,3],[1,2],[1,1],[2,0],[3,1],[2,2],[3,2],[4,2],[5,1],[4,0],[5,1],[5,2],[4,3],[5,4],[5,5]]

* + 1. **Matriz C :**

[[0,0],[0,1],[0,2],[0,3],[1,2],[1,1],[2,1],[2,2],[1,3],[2,4],[1,5],[1,6],[0,7],[1,7],[1,8],[2,7],[2,8],[3,7],[

3,6],[4,6],[4,7],[4,8],[5,8],[6,8],[6,7],[5,6],[5,5],[4,4],[3,4],[4,3],[5,2],[4,1],[3,1],[4,2],[5,3],[5,4],[6,

3],[6,2],[7,2],[8,2],[7,1],[7,0],[7,1],[7,2],[7,3],[7,4],[6,5],[7,5],[7,6],[8,6],[8,7],[8,8]]

* + 1. **Matriz D :**

[[0,0],[0,1],[0,2],[0,3],[0,4],[1,3],[1,2],[1,1],[1,0],[2,1],[3,0],[3,1],[3,2],[4,2],[5,3],[5,4],[5,5],[5,6],[

6,6],[7,6],[7,7],[6,8],[5,8],[4,9],[3,10],[2,9],[1,9],[0,9],[0,10],[0,9],[0,8],[0,7],[1,6],[1,5],[1,4],[2,4],

[2,5],[2,6],[2,7],[3,6],[3,5],[3,4],[4,3],[5,3],[6,3],[6,4],[7,3],[8,2],[7,1],[6,0],[7,0],[6,0],[5,0],[4,0],[5

,1],[6,2],[7,3],[8,4],[8,5],[8,6],[7,7],[6,8],[5,9],[6,10],[7,9],[8,9],[8,10]]

* + 1. **Matriz E :**

[[0,0],[0,1],[1,0],[2,1],[2,2],[3,1],[3,0],[4,1],[4,2],[3,3],[2,4],[1,4],[0,3],[1,4],[1,5],[1,6],[1,7],[0,8],[

1,8],[2,8],[3,7],[3,6],[3,5],[4,5],[5,4],[6,3],[6,4],[6,5],[5,6],[4,7],[4,8],[5,7],[6,6],[7,5],[7,4],[7,3],[7,

2],[6,1],[6,0],[7,0],[8,0],[9,0],[9,1],[8,2],[8,3],[9,2],[10,1],[10,0],[11,0],[12,0],[13,0],[14,0],[15,0],[

16,0],[17,1],[16,2],[15,2],[15,3],[16,4],[17,3],[17,4],[18,3],[18,4],[18,5],[17,6],[18,6],[19,6],[19,7],

[18,8],[17,8],[16,8],[15,8],[14,7],[13,6],[13,7],[14,8],[15,7],[15,6],[14,5],[14,4],[13,3],[12,3],[11,2

],[12,2],[13,2],[12,3],[11,4],[10,5],[9,6],[8,6],[8,7],[8,8],[9,7],[9,8],[10,7],[11,7],[11,6],[11,5],[12,4

],[13,3],[13,2],[14,1],[15,0],[16,0],[17,0],[18,0],[19,0],[19,1],[19,2],[19,3],[19,4],[19,5],[19,6],[19,

7],[19,8]]

# Dificuldades:

### Limitações:

### Branch and Bound:

* + - * Embora o algoritmo "Branch and Bound" seja poderoso e eficiente para muitos problemas, ele enfrenta desafios quando aplicado a matrizes maiores, levanto um tempo exponencial e podendo estourar a pilha. A principal razão para essa limitação está relacionada ao crescimento exponencial do espaço de busca à medida que as dimensões da matriz aumentam.

### Dificuldades e como superamos:

### Swish.SWI-Prolog:

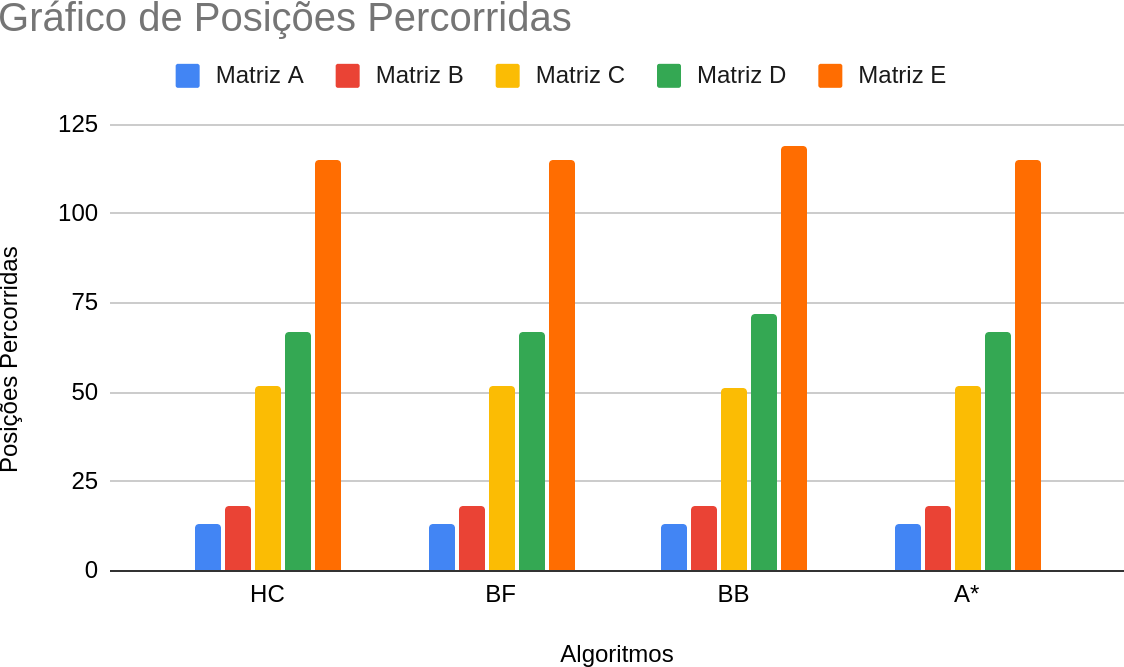
* + - * Na implementação dos algoritmos de busca, estávamos utilizando o Swish via Web, a pilha de execução estourava e não conseguimos testar. Esse fato ocorreu e acarretou em diversos testes falhos e verificações na nossa implementação. Em contrapartida, para a resolução do problema, decidimos rodar o código por meio do software “Visual Studio Code”, a fim de aumentar o limite da pilha, identificar o problema e ,dessa forma, otimizá-lo.

### Problema na implementação:

* + - * Na criação da solução, os algoritmos não conseguiam percorrer e encontrar todas as sujeiras e acabavam por não solucionar o problema. Outro ponto é a implementação dos algoritmos que não conseguiam percorrer em matrizes grandes, como 10x10, limitando demais nosso escopo.
      * Para resolver, tivemos que mudar a lógica das implementações algumas vezes de forma que atendessem não só o escopo, mas que resolvesse em tempos razoáveis. A partir desse problema, surgiram como solução uma ordem de prioridade para as sujeiras e a função “*auxiliar*” desses algoritmos.

# Comparação dos algoritmos:

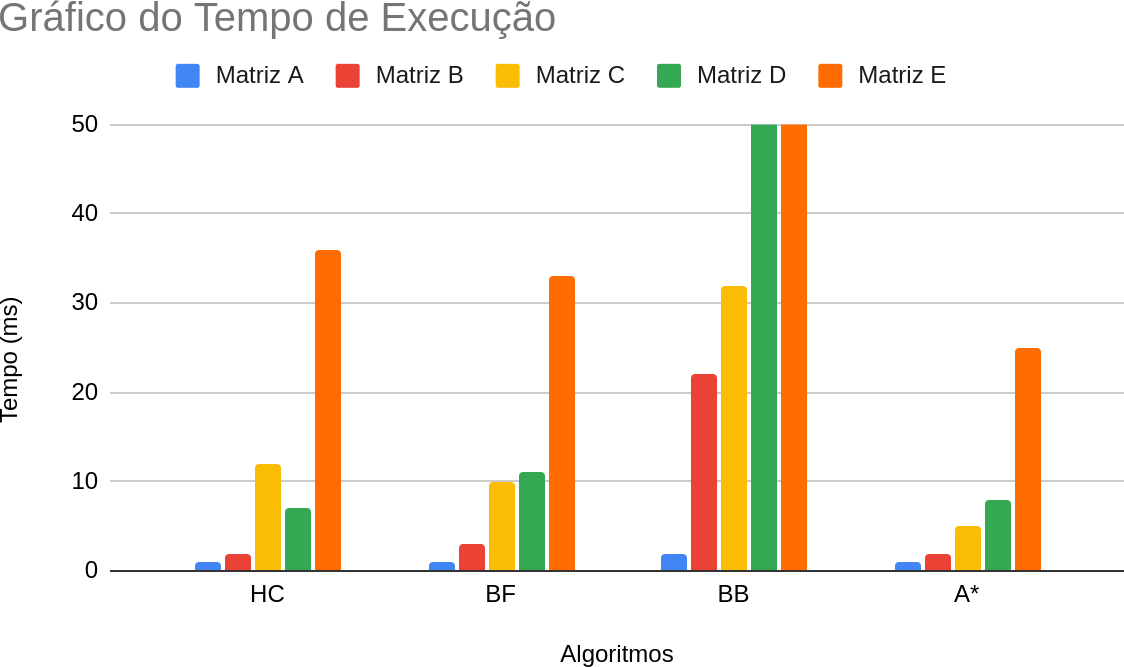
### Gráfico de Posições Percorridas:

****

### Custo dos movimentos:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Custo** | **Matriz A** | **Matriz B** | **Matriz C** | **Matriz D** | **Matriz E** |
| **HC** | 13 | 18 | 52 | 67 | 115 |
| **BF** | 13 | 18 | 52 | 67 | 115 |
| **BB** | 13 | 18 | 51 | 72 | 119 |
| **A\*** | 13 | 18 | 52 | 67 | 115 |

### Gráfico do Tempo de Execução:

****

### Tempo de execução:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Tempo de solução (ms)** | **Matriz A** | **Matriz B** | **Matriz C** | **Matriz D** | **Matriz E** |
| **HC** | 1 | 2 | 12 | 7 | 36 |
| **BF** | 1 | 3 | 10 | 11 | 33 |
| **BB** | 2 | 22 | 32 | 112 | 335 |
| **A\*** | 1 | 2 | 5 | 8 | 25 |

### Conclusão:

O algoritmo Hill Climb é caracterizado por um tempo de execução rápido, o que o torna adequado para problemas menores. No entanto, ele tem uma abordagem unidirecional, pois se baseia em busca e profundidade. Isso significa que o algoritmo pode ficar preso em ótimos locais e não explorar todas as opções disponíveis, levando a uma solução subótima.

Já o Best First possui uma visão mais ampla ao selecionar o melhor caminho primeiro. Ele avalia as opções disponíveis com base em uma heurística específica e escolhe o caminho que parece mais promissor. Isso resultou em um tempo de execução razoável e boas soluções, mas o algoritmo não garante a melhor solução global, pois está focado apenas em escolher a melhor opção imediata.

O Branch and Bound utiliza uma estratégia de divisão e conquista, explorando todas as possíveis soluções em uma busca em largura. No entanto, à medida que o tamanho do problema aumenta, o tempo de execução também aumenta significativamente, tornando-o menos eficiente para problemas maiores.

O algoritmo mais eficiente estudado é o A\*. Ele possui uma visão abrangente de todos os caminhos possíveis, utilizando uma heurística baseada em custo e avaliação para determinar o seguimento do caminho. O A\* combina as vantagens da busca em largura e em profundidade, garantindo a melhor solução e tendo o melhor tempo de execução no nosso estudo.

# Desenvolvimento das Atividades:

### Divisão de trabalho:

### Programação:

* + - * No projeto de programação em Prolog, nosso grupo adotou uma abordagem colaborativa e organizada para maximizar nossa eficiência e garantir um resultado final de alta qualidade. Para atingir esse objetivo, dividimos o trabalho com diferentes práticas como Pair Programming, além de separar tarefas para corrigir bugs e implementar novas funcionalidades.
      * Durante a fase de Pair Programming, todos os membros da equipe trabalharam juntos em uma única estação de trabalho. Essa abordagem nos permitiu compartilhar ideias e conhecimentos em tempo real, além de promover uma maior comunicação e colaboração entre os membros da equipe. Enquanto um dos membros escrevia o código, os outros acompanhavam e revisavam o trabalho, identificando possíveis erros ou melhorias. Essa troca constante de informações resultou em um código mais limpo, menos propenso a erros e melhor documentado. Todo o core do trabalho foi feito dessa forma até para poder utilizar melhor as capacidades de cada integrante. Camila Ferreira foi a responsável por codificar na maior parte dessa fase e que tinha maior conhecimento de Prolog em si, Fernando Moreira cuidou principalmente de corrigir bugs e melhorar certos detalhes da implementação e Thiago Mozart ficou responsável nessa fase por direcionar nas soluções mais lógicas e também corrigir bugs e verificar como estava sendo implementada as soluções do projeto.
      * Após a fase de Pair Programming, passamos para a etapa de correção de bugs e implementação de novas funcionalidades. Nessa fase, cada membro da equipe foi designado para trabalhar individualmente em tarefas específicas. Essa abordagem nos permitiu aproveitar as habilidades individuais de cada membro e acelerar o progresso do projeto. Cada pessoa era responsável por resolver os bugs atribuídos a ela, além de desenvolver novas funcionalidades de acordo com os requisitos estabelecidos.
      * Para garantir a sincronização e a integração do trabalho realizado em diferentes fases, utilizamos um sistema de controle de versão, o Git(*Github*). Isso nos permitiu gerenciar o código-fonte de forma eficiente, rastrear as alterações realizadas por cada membro e mesclar as diferentes contribuições em um código final coeso.
      * Além disso, durante todo o processo, realizamos reuniões regulares para discutir o progresso do projeto, compartilhar feedback e resolver dúvidas. Essas reuniões foram fundamentais para manter a equipe alinhada e garantir que todos estivessem cientes do andamento do projeto e dos desafios enfrentados.
      * Ao final do projeto, graças à nossa organização e colaboração, acrescentamos a nossa experiência de trabalhar em equipe, combinando técnicas como Pair Programming e tarefas separadas, que nos proporcionou um aprendizado valioso e uma base sólida para futuros projetos de programação. Vale ressaltar que trabalhamos em equipe na maior parte das matérias e nos conhecemos muito, então já estávamos preparados para qualquer problema enfrentado no projeto.

### Relatório:

* + - * Após a conclusão do projeto de programação em Prolog, nossa equipe se dedicou à elaboração de um relatório abrangente em que fosse documentado nosso trabalho e resultados. Para garantir que o relatório atendesse aos padrões de qualidade e fosse satisfatório, adotamos, mais uma vez, uma abordagem colaborativa, na qual cada membro contribuiu ativamente e trouxe ideias adicionais.
      * Inicialmente, realizamos uma reunião para discutir a estrutura e o conteúdo do relatório. Cada membro teve a oportunidade de compartilhar suas perspectivas sobre as partes mais relevantes do projeto, bem como as principais descobertas e desafios enfrentados. Essa troca de ideias enriqueceu o relatório, pois cada membro pôde trazer sua expertise e insights únicos, até por conta da separação do grupo no momento de programar.
      * A partir dessa reunião, dividimos as seções do relatório em tarefas individuais, atribuindo responsabilidades específicas a cada membro da equipe. Cada um foi responsável por redigir sua parte do relatório, detalhando as atividades realizadas, as metodologias utilizadas e os resultados obtidos. No entanto, além de simplesmente cumprir suas tarefas, incentivamos todos a irem além e trazerem ideias adicionais para enriquecer o conteúdo.
      * Durante o processo de redação, mantivemos uma comunicação constante por meio de reuniões regulares e feedback em um grupo de Whatsapp. Isso nos permitiu compartilhar o progresso, esclarecer dúvidas e fornecer feedback uns aos outros. Também utilizamos ferramentas de colaboração online, o Google Docs, para facilitar a edição e o compartilhamento de documentos em tempo real.
      * Além de contribuir com as partes designadas, cada membro da equipe foi incentivado a revisar e fornecer feedback sobre as seções escritas pelos outros. Essa abordagem colaborativa de revisão cruzada garantiu uma visão mais abrangente do relatório e ajudou a identificar possíveis melhorias ou lacunas.
      * À medida que avançávamos na elaboração do relatório, mantivemos o foco em uma linguagem clara e concisa, evitando jargões técnicos excessivos e explicando conceitos complexos de forma acessível. Também nos certificamos de incluir gráficos e exemplos relevantes para tornar o relatório mais visualmente atrativo e compreensível.

### Apresentação:

* + - * Para complementar nosso projeto de programação em Prolog, decidimos criar uma apresentação de slides no Google Slides para compartilhar nosso trabalho e resultados de forma visual e interativa. Durante o processo, o integrante Fernando Moreira desempenhou um papel fundamental, sendo o membro mais ativo na realização das tarefas relacionadas à criação da apresentação.
      * Com um esforço conjunto, conseguimos criar uma apresentação envolvente e divertida, ao mesmo tempo em que mantínhamos o conteúdo condizente com o projeto desenvolvido. A apresentação foi estruturada de forma a mostrar partes da implementação do nosso robô de limpeza.
      * Na apresentação, fornecemos explicações detalhadas sobre cada parte da implementação, seja como funcionava, e também construímos uma timeline de como cada parte do código se unifica e consegue construir a solução para o projeto. Utilizamos gráficos para analisar o tempo de execução de cada algoritmo, permitindo uma comparação direta entre eles. Também criamos um gráfico para demonstrar quantas casas cada algoritmo teve que percorrer na matriz, o que nos ajudou a avaliar sua eficiência em diferentes cenários.
      * Ao longo da apresentação, buscamos manter um equilíbrio entre a parte técnica e a apresentação visualmente atraente. Utilizamos ilustrações e esquemas claros para tornar os conceitos mais acessíveis e interessantes. Também inserimos exemplos práticos e resultados reais do nosso projeto para demonstrar a aplicação dos algoritmos de busca na solução do problema do robô de limpeza.
      * A criação dessa apresentação não apenas permitiu que compartilhássemos nosso trabalho de forma clara e eficaz, mas também nos desafiou a consolidar nosso conhecimento e apresentá-lo de uma maneira envolvente. A participação ativa do integrante Fernando Moreira e a colaboração de todo o grupo garantiram que o resultado final fosse de alta qualidade e transmitisse de forma eficiente o valor do projeto.

### Ferramentas:

### Comunicação:

* + - * WhatsApp: Utilizamos o WhatsApp como uma ferramenta de comunicação instantânea para manter contato rápido e eficiente entre os membros da equipe. Foi útil para troca de mensagens, compartilhamento de informações e agendamento de reuniões.
      * Discord: Além do WhatsApp, também utilizamos o Discord, uma plataforma de comunicação por voz e texto, que nos permitiu realizar chamadas de grupo, compartilhar arquivos e manter conversas em tempo real.

### Programação:

* + - * Git com GitHub: Optamos pelo Git como sistema de controle de versão para gerenciar o código-fonte do projeto. Utilizamos o GitHub como plataforma para hospedar nosso repositório, facilitando o compartilhamento de código e o trabalho colaborativo.
      * IDE - Visual Studio Code (VSCode) com Swish: Escolhemos o Visual Studio Code como nossa IDE principal, aproveitando suas funcionalidades e extensão disponível para o Prolog. Também obtivemos vantagens como controle do aumento de pilha. Contudo, o Swish, um ambiente de programação interativo e online para Prolog, nos auxiliou principalmente na depuração dos programas.
      * ChatGPT: Para fornecer assistência no código na parte de criação dos nós e nos métodos utilizados pelo robô para se movimentar, usamos o ChatGPT, um modelo de linguagem desenvolvido pela OpenAI.

### Relatório e Apresentação:

* + - * Google Docs: Para a criação do relatório, utilizamos o Google Docs, uma ferramenta de processamento de texto colaborativo baseada em nuvem. Essa plataforma nos permitiu trabalhar simultaneamente em diferentes partes do relatório, facilitando a revisão e a edição conjunta.
      * Google Slides: Para a criação da apresentação, escolhemos o Google Slides, uma ferramenta de criação de slides online. Com ela, conseguimos colaborar na elaboração da apresentação, adicionar elementos visuais e tornar a apresentação mais atraente e interativa.
      * Bing Image Creator: Para a criação da imagem de um robô, utilizamos o Bing Image Creator, uma ferramenta online que nos permitiu pesquisar e criar imagens de forma rápida e fácil. Com ela, encontramos a ilustração do Rob, que complementou a apresentação e ajudou a entreter a apresentação de maneira visualmente atraente.

### Contribuição individual:

* Cada membro do nosso grupo desempenhou um papel importante em diferentes aspectos do projeto, trazendo suas habilidades e contribuições únicas para alcançarmos o sucesso coletivo. Embora todos tenham participado de todas as etapas do trabalho, alguns membros se destacaram em áreas específicas.
* Camila assumiu a liderança na parte de programação, demonstrando habilidades sólidas e uma compreensão profunda do Prolog. Sua experiência e conhecimento técnico permitiram que liderasse o grupo nessa área, orientando e apoiando os demais membros na implementação das funcionalidades. Sua dedicação em explorar soluções eficientes e garantir a qualidade do código foi fundamental para o sucesso do projeto como um todo.
* Thiago se destacou na parte dos algoritmos, mostrando uma grande habilidade em análise e pesquisa. Sua expertise nessa área permitiu que ele explorasse os algoritmos e suas implementações fornecendo insights valiosos para a implementação do robô de limpeza. Além disso, ele contribuiu significativamente na elaboração do relatório, fornecendo explicações claras e comparando o desempenho dos algoritmos de forma precisa.
* Fernando foi fundamental na parte de correção de bugs e na criação da apresentação. Sua atenção aos detalhes e habilidades de resolução de problemas foram essenciais para identificar e corrigir eventuais erros no código. Além disso, sua criatividade e habilidades de comunicação se destacaram na criação de uma apresentação envolvente e interativa. Ele trouxe elementos visuais, tornando a apresentação mais atraente e auxiliando na compreensão dos conceitos abordados.
* É importante ressaltar que, apesar dos papéis destacados, todos os membros do grupo estiveram envolvidos em todas as etapas do projeto. Houve uma colaboração constante e uma troca de conhecimento entre todos. Cada membro trouxe contribuições valiosas e se empenhou em realizar tarefas além de suas áreas específicas, fortalecendo o trabalho em equipe e garantindo um resultado final de qualidade.

# Links e Referências Bibliográficas:

### Links:

### Links para auxílio:

Distância de Chebyshev: [link](http://www.augustobaffa.pro.br/wiki/Dist%C3%A2ncia_de_Chebyshev) Algoritmo de busca Hill Climbing : [link](https://youtu.be/OD012QkZpCA)

Algoritmo de busca Best First e A\*: [link](https://www.youtube.com/watch?v=GV1jOZBD1CQ&t=2469s) Algoritmo de busca Branch and Bound: [link](https://www.baeldung.com/cs/branch-and-bound) Problemas similares: [link](https://docs.ufpr.br/~volmir/PO_II_12_TSP.pdf)

Uso de IA para alguns conceitos: [link](https://chat.openai.com/) Uso do material dado em aula: [link](https://classroom.google.com/u/1/c/NjAyNjM1MTU5ODg2) Bing Image Creator: [link](https://www.bing.com/create)