UNIVERSIDADE FEDERAL DO CEARÁ

# TECHNICAL REPORT

Aluno: Suziane Brandão Andrade

## 1. Introdução

- **Classificação: Hotel Reservations Dataset**

  Este Dataset contém informações sobre reservas em hotéis, com um foco específico nos cancelamentos frequentes e nos casos de não comparação dos clientes. Esses cancelamentos e faltas representam um grande desafio para as redes hoteleiras, pois impactam diretamente a capacidade de alocação e a otimização das operações.

  Quando uma reserva é cancelada ou quando um hóspede não aparece, o hotel perde a oportunidade de ocupar o quarto com outro cliente, o que resulta em uma perda de receita.

- **Regressão: Preços de Relógios Inteligentes**

  Este conjunto de dados fornece uma visão detalhada de diversos modelos de smartwatches, permitindo uma análise comparativa entre marcas, características como tamanho da tela, duração da bateria, sistemas operacionais e outras funcionalidades importantes. Através dessa análise, é possível identificar tendências do mercado, comparar o desempenho de diferentes dispositivos e explorar os fatores que influenciam a escolha do consumidor, como preços e funcionalidades.

## 2. Observações

*Algum problema que aconteceu? Alguma observação? Algum imprevisto? Se não houve problemas, deixe em branco.*

## 3. Resultados e discussão

### QUESTÃO 1:

**Realizar manipulação em um dataset com a biblioteca pandas e realizar o pré-processamento.**

Nesta questão foi utilizado o dataset de **Classificação: Hotel Reservations.**

**Passo 1:**

Inicialmente foi verificado as informações gerais do dataset, como a dimensão e os tipos de dados com qual seria trabalhado, usando os argumentos: **shape, dtypes e Info.**

**Informações extraídas:**

| Quantidade de Colunas | Quantidade de Linhas | Tipos de Dados | Valores Nulos |
|---|---|---|---|
| 19 | 36274 | Int64, Float64, object | Inexistentes |

**Passo 2**:

Após essa análise inicial, foi concluído que não havia células vazias ou **Nan**. Também foi verificado quais colunas eram desnecessárias e retiradas.

**Colunas Deletadas:**

| | | |
|---|---|---|
| Booking_ID | Type_of_meal_plan | required_car_parking_space |
| arrival_yea | arrival_date | market_segment_typ |

**Passo 3:**

Foi feita uma conversão de duas colunas Usando o Pandas para **One-Hot Encoding**, a fim de normalizar os dados, que consistia em mapear cada categoria para um valor numérico único.

| | |
|---|---|
| room_type_reserved | booking_status |

Com as modificações foi gerado um novo data frame que será usado posteriormente, chamado: **Hotel_Normalizado,** que resultou em um total de **18 colunas**.

**QUESTÃO 2** :

**Realizar uma classificação utilizando KNN implementado de forma manual**

Para implementar o KNN manualmente, segui os seguintes passos:

**Passo1**: **Conferir o estado do Dataset**

O objetivo principal, era classificar quais perfis em potencial poderiam cancelar ou mesmo faltar nas reservas de hotéis.

Inicialmente carreguei o Dataset atualizado: **Hotel_Atualizado**

Usei as funções - **shape e info()** , para ter certeza de que que o dataset estava conforme salvo na questão anterior.

**Passo 2: Amostragem Aleatória**

Inicialmente, foi feita a implementação com a quantidade de dados total do dataset, o que resultou em um processamento extremamente lento. Uma forma de contornar a situação, foi fazendo a retirada de uma amostragem, reduzindo o tamanho do dataset para 10% dos dados originais para testes e desenvolvimento.

**Passo 3: Separar Features (X) e Classe/Target (y)**

Nesse terceiro passo, foi separado os dados em Features e Target.

Para X foi escolhida a coluna **booking_status_Not_Canceled,** referente aos Status das reservas que não foram canceladas, removendo-a.

**Passo 4: Divisão Manual em Treino e Teste**

O Dataset foi dividido em conjuntos de treino e teste.

Na proporção de treino foi usado um total de 70% (0.7), como pedido na questão.

Os índices foram embaralhados usando a função **random**, retornando um conjunto de dados para treino e outro para teste.

**Passo 5: Funções de Distância**

Nesse passo foi Implementado as diferentes métricas de distância para comparação das amostras.

1. **Euclidiana:** Raiz quadrada da soma dos quadrados das diferenças.
2. **Manhattan:** Soma das diferenças absolutas.
3. **Chebyshev:** Máxima diferença absoluta.

4. **Mahalanobis:** Mede a distância considerando a correlação entre variáveis (usando a matriz inversa da covariância).

**Passo 6: Cálculo da Matriz de Covariância e Sua Inversa**

Seguindo o fluxo, foi feita uma preparação da matriz de covariância para calcular a distância de **Mahalanobis.**

- **`np.cov`**: Calcula a matriz de covariância entre as variáveis.
- **`np.linalg.pinv`:** Calcula a pseudo-inversa da matriz de covariância.

**Passo 7: Implementação do KNN Manual**

1. A implementação do KNN, calcula a distância entre a nova amostras e todas as amostras de treino, logo após, Seleciona os `k` vizinhos mais próximos e retorna a classe mais votada entre os vizinhos.

2. O código segue para avaliar o desempenho do KNN no conjunto de testes, no qual houve:

- Compara a predição do KNN com os valores reais (`y_test`).
- Calcula a acurácia como o percentual de acertos.

3. Comparar as Métricas de Distância

- Avalia a acurácia do KNN usando as quatro funções de distância implementadas.
- Armazenar os resultados de cada métrica para análise comparativa posteriormente.

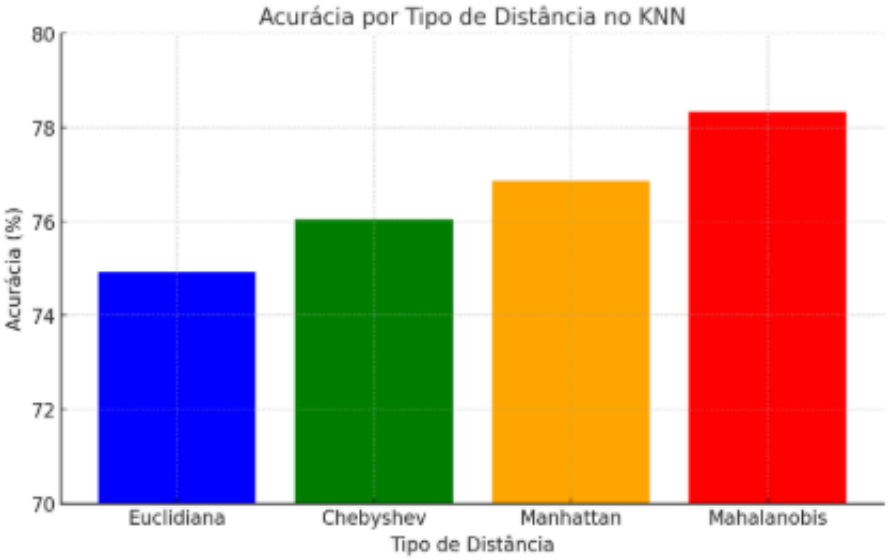
**Passo 8: Resultados Finais**

Foi exibido no terminal as acurácias obtidas para cada métrica de distância.

```
Tamanho do conjunto de treino: 2539
Tamanho do conjunto de teste: 1089
Acurácia final: 74.93%
Precisão usando Euclidiana: 74.93%
Acurácia final: 76.03%
Precisão usando Manhattan: 76.03%
Acurácia final: 75.67%
Precisão usando Chebyshev: 75.67%
Acurácia final: 78.33%
Precisão usando Mahalanobis: 78.33%

Resultados Finais:
Euclidiana: 74.93%
Manhattan: 76.03%
Chebyshev: 75.67%
Mahalanobis: 78.33%
```

**Gráfico conforme a saída:**




Podemos observar as diferenças de acurácia no algoritmo KNN ao utilizar as diferentes métricas de distância.

## Análise do Gráfico

1. **Melhor Desempenho com Distância Mahalanobis:**
   - A métrica Mahalanobis apresentou a maior acurácia (cerca de **78%**), indicando que leva em consideração a correlação entre as variáveis do dataset de forma eficiente.

2. **Pior Desempenho com Distância Euclidiana:**
   - A distância Euclidiana teve a menor acurácia, o que sugere que os dados não foram normalizados corretamente, exatamente na questão da escala, que estava diferente em algumas colunas.

**Variação em escalas das variáveis:**

Após essa primeira análise ficou constatado que havia uma variação nas escalas de algumas colunas no dataset:

- A variável **lead_time** (tempo de antecedência da reserva) variava de **0 a 443**, enquanto **no_of_adults** variava de **0 a 4**.
- A variável **avg_price_per_room** variava de **0 a 540**, o que é significativamente maior em escala, comparando com as outras variáveis.

**Impacto em métricas de distância:**

- Já que distâncias como a **Manhattan** e **Euclidiana** são sensíveis à escala dos dados. Variáveis com valores maiores (como lead_time ou avg_price_per_room) tiveram um peso desproporcional no cálculo da distância. Isso pode explicar o desempenho inferior dessas métricas.
- Já a **Mahalanobis**, que usa a matriz de covariância, ajustava automaticamente as contribuições de cada variável, resultando em um desempenho melhor neste caso.

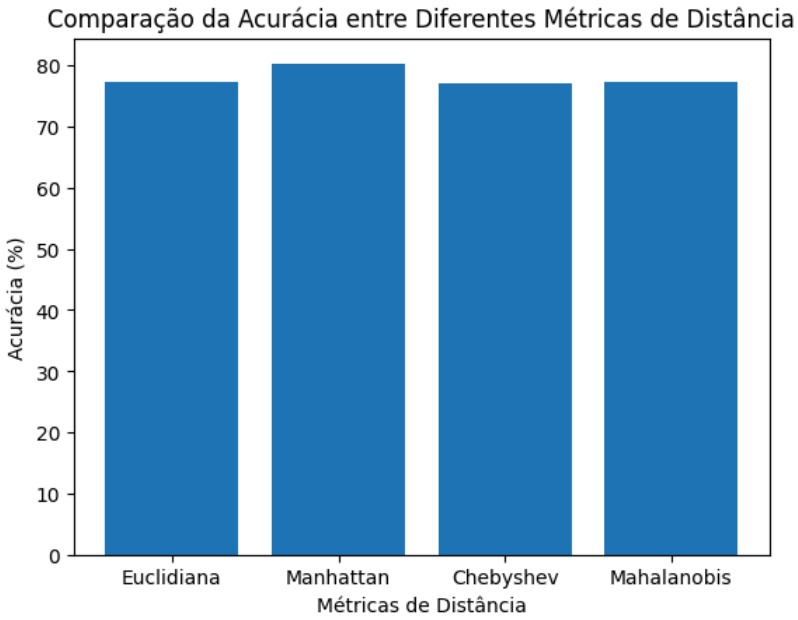
**Necessidade de normalização:**

- Levando esse cenário em consideração, voltei ao código da primeira questão e usei **Standard Scaling** para garantir que todas as variáveis tenham a mesma influência;

```
memory usage: 3.3 MB
Informações do dataset: None
Tamanho do conjunto de treino: 2539
Tamanho do conjunto de teste: 1089
Acurácia final: 79.71%
Precisão usando Euclidiana: 79.71%
Acurácia final: 78.33%
Precisão usando Manhattan: 78.33%
Acurácia final: 78.60%
Precisão usando Chebyshev: 78.60%
Acurácia final: 80.07%
Precisão usando Mahalanobis: 80.07%

Resultados Finais:
Euclidiana: 79.71%
Manhattan: 78.33%
Chebyshev: 78.60%
Mahalanobis: 80.07%
```

**Gráfico conforme a saída:**




**Manhattan** foi a métrica de distância que obteve a **melhor acurácia** (80.35%). Isso indica que a métrica de Manhattan pode ser mais eficaz para classificar as instâncias corretamente, considerando a distribuição dos dados.

**Euclidiana** e **Mahalanobis** apresentaram a mesma acurácia (77.23%), o que sugere que essas duas métricas de distância têm um desempenho semelhante.

**Chebyshev** obteve o pior desempenho entre essas métricas, com uma acurácia de 77.13%, ligeiramente inferior às demais.

QUESTÃO 3 :

**Considerando a melhor distância observada no exercício anterior, neste você deve verificar se a normalização interfere nos resultados de sua classificação.**

**Passo 1:**

Foi feita duas normalizações a Logarítmica e Médio Zero Variância Unitária. Eu já havia usado a Normalização de Média Zero e Variância Unitária StandardScaler(), pois havia dado um erro no processamento do modelo sem essa normalização. segue abaixo o gráfico comparativo entre os métodos com normalização e sem normalização.

**Gráfico Comparativo**

| Método | Acurácia (%) |
|---|---|
| Sem Normalização | |
| - Euclidiana | 77.23 |
| - Manhattan | 80.35 |
| - Chebyshev | 77.13 |
| - Mahalanobis | 77.23 |
| Com Normalização | |
| - Logarítmica | 82.42 |
| - Média 0, Variância 1 | 83.45 |

Com a normalização dos dados a acurácia aumentou , uma vez que variava entre 77% a 80%, porém, não houve um aumento significativo, com apenas 3%.

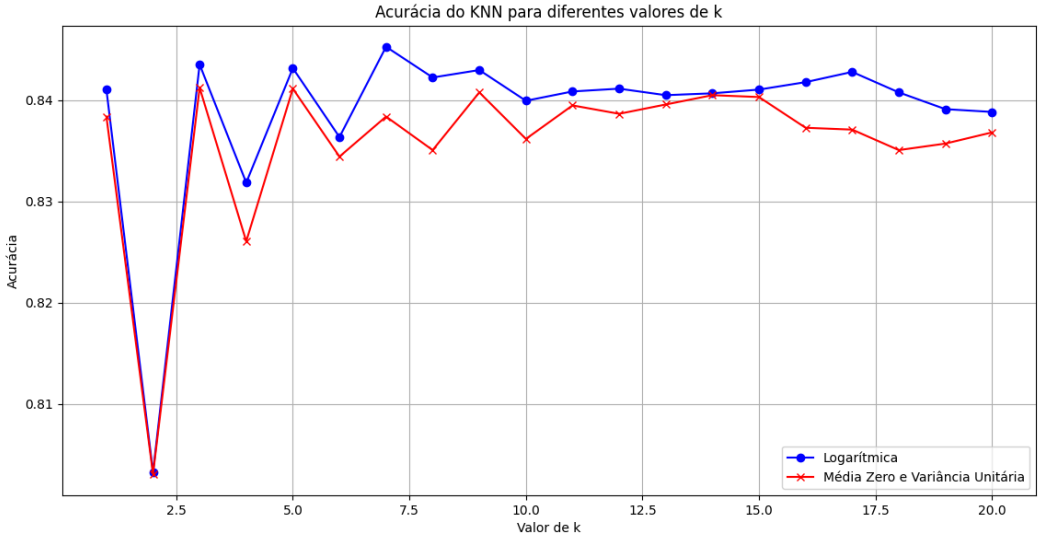
QUESTÃO 4 :

**Buscar saber a melhor parametrização do knn implementado.**

Normalize com a melhor normalização o conjunto de dados se houver melhoria. Plote o gráfico com a indicação do melhor k.

Realizei a normalização anteriormente, mas estava aplicando uma transformação logarítmica no conjunto de dados que já foi normalizado, e isso causou valores **NaN** aparecendo após a transformação. Voltei a questão 1, onde havia usado StandardScaler(), que estava gerando esses valores e deletei, fazendo apenas as normalização exigidas nas questões.

Após esse ajuste o gráfico foi plotado e mostra essa comparação da Acurácia do KNN, comparando as duas normalizações. É possível perceber que por mais próximas, a Logarítmica, quando o valor está próximo de K =7, está com a acurácia bem alta.




**QUESTÃO 5:**

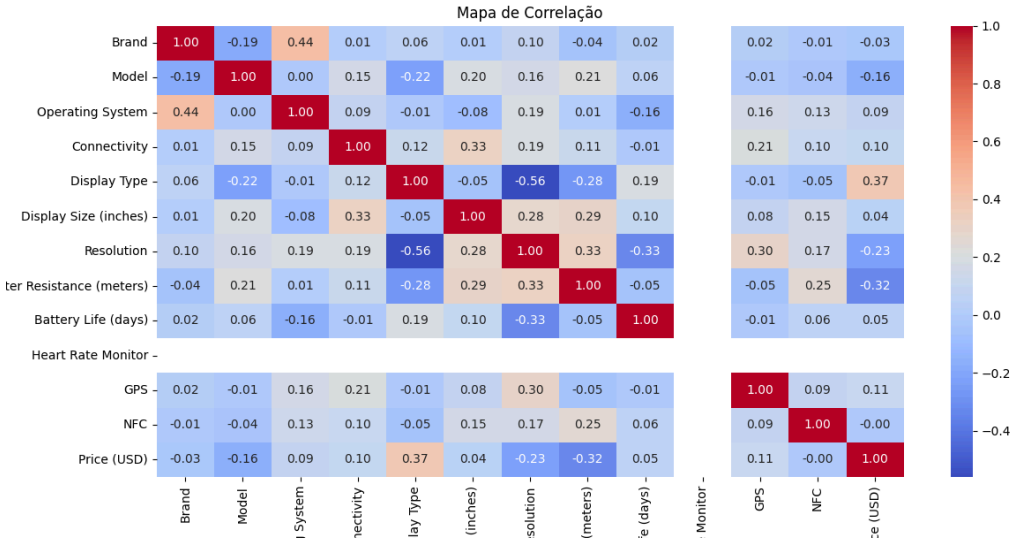
**Observe o dataset de regressão e realize o pré-processamento. Verifique qual atributo será o alvo para regressão no seu dataset e faça uma análise de qual atributo é mais relevante para realizar a regressão do alvo escolhido. Lembre de comprovar via gráfico. Caso necessário remova colunas que são insignificantes, valore NaN também devem ser removidos.**

**Passo 1:**

No dataset de **Regressão: Preços de Relógios Inteligentes,** também foi feita uma análise dos dados disponíveis. Ficou constatado alguns valores Nan, que logo foram removidos com comando `dropna`. Também foi necessário fazer a normalização de alguns dados do tipo `object`.

Inicialmente eu gostaria de fazer uma regressão visando saber quanto duraria a bateria com base em atributos do aparelho como modelo e fabricante. Mas ao plotar o mapa de correlação, percebi que a duração de bateria tinha pouquíssima relação com os outros dados dos Smarth Watch. Então mudei para uma regressão linear usando **Brand**(marca) ou **Operating System**. Segue abaixo, duas visualizações dos dados referentes a correlação entre as features com a Marca.

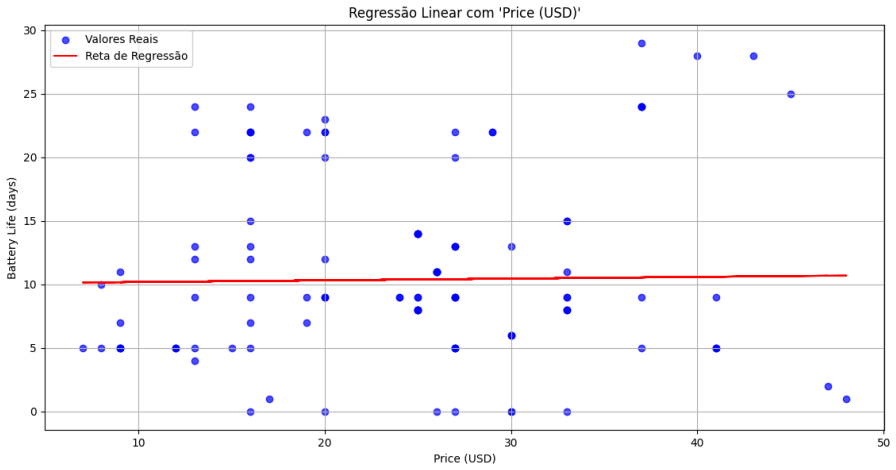
| Feature | Correlation with Brand |
|---|---|
| Brand | 1.000.000 |
| Operating Syste | 439.745 |
| Resolution | 95.606 |
| Display Type | 57.369 |
| Battery Life (day | 23.324 |
| GPS | 22.942 |
| Display Size (inc | 10.727 |
| Connectivity | 6.953 |
| NFC | -12.348 |
| Price (USD) | -31.471 |
| Water Resistanc | -44.495 |
| Model | -189.113 |



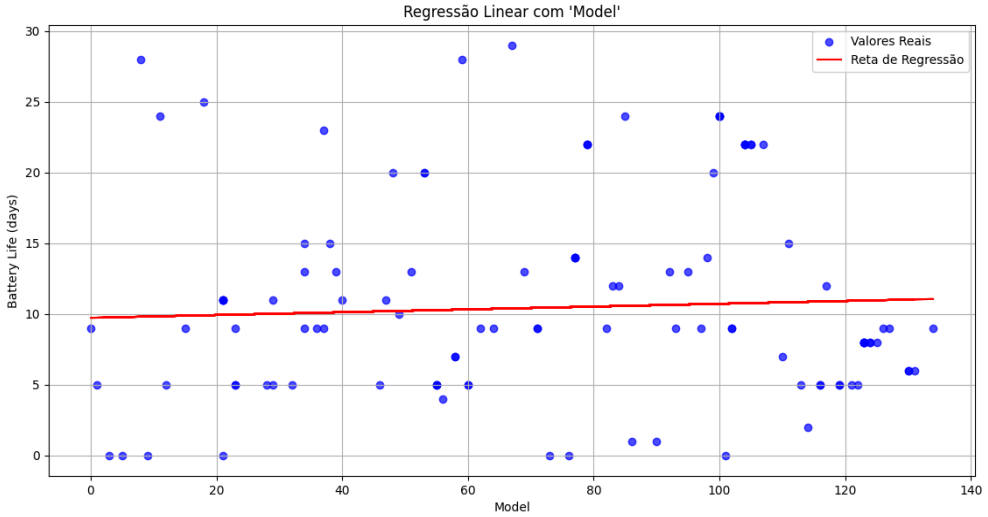

## QUESTÃO 6:

**Implemente uma regressão linear utilizando somente este atributo mais relevante, para predição do atributo alvo determinado na questão 5 também**

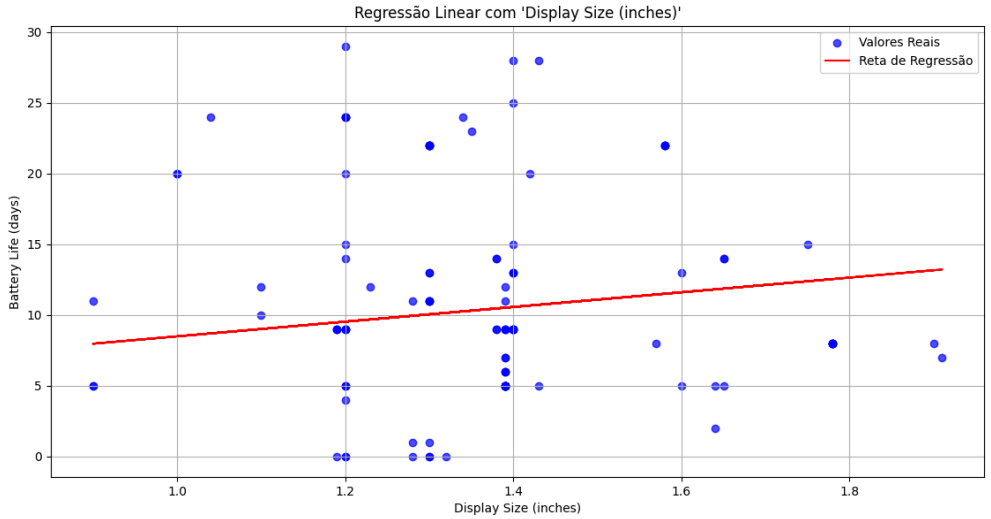
Como dito anteriormente, tentei fazer a regressão tendo como alvo a duração da bateria, o gráfico segue logo abaixo:



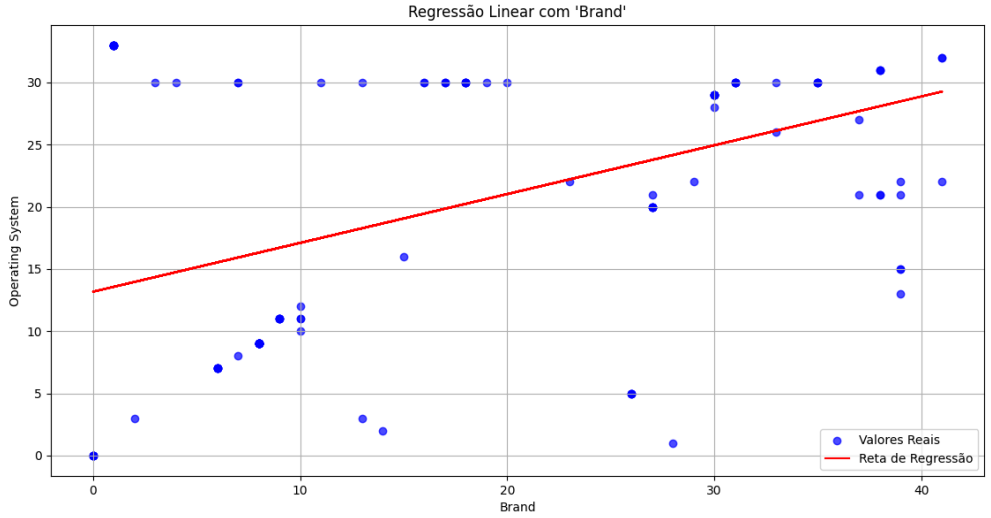

Nesse gráfico utilizei o preço como um atributo relevante para a duração da bateria, pensando que dispositivos comumente mais caros tivessem uma duração melhor em comparação com os mais baratos. Logo em seguida, pensei em usar o modelo e o SO do smartwatch, pois poderia consumir mais energia, mas como pode-se visualizar o R-squared está muito baixo. Insisti um pouco com algumas features que estavam medianamente relacionadas. Teve apenas um pequeno aumento com o Display Size(inches).




Display Size(inches)




Logo após, seguindo o que foi mostrado no mapa de correlação, apliquei a regressão usando a marca(brand), como atributo relevante, tendo Sistema operacional como alvo, o que resultou r-squared significativamente melhor.

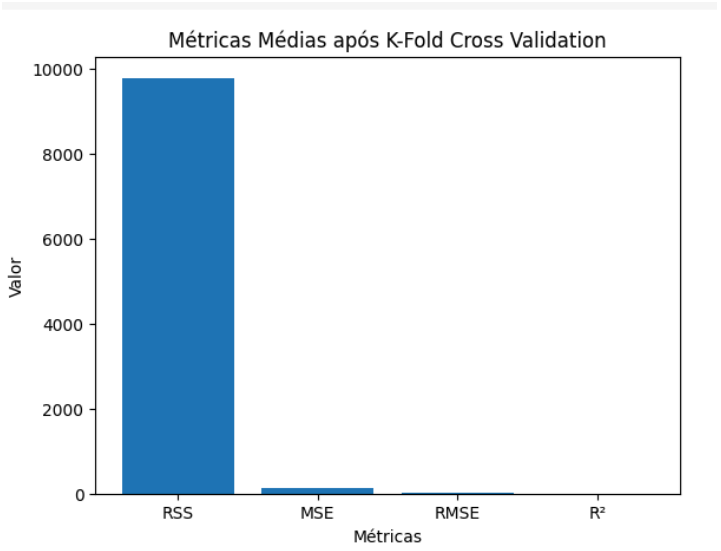



## QUESTÃO 7:

**Utilizando kfold e cross-validation faça uma regressão linear. Utilize uma implementação manual do kfold e cross-validation. calule as métricas RSS, MSE, RMSE e R_squared que também devem ser implementadas manualmente.**

Para a implementação manual, ainda usando Brand(marca) para a regressão, foi feita a separação dos dados de treino e teste. Criei uma função chamada **k_fold_cross_validation** para dividir os dados em várias partes e testar a precisão da previsão.

Após aplicado o modelo fit de regressão linear. Cada uma das métricas foram implementadas manualmente de acordo com suas funções. Para cada "fold", guardei as métricas e, ao final, calculamos a média de todas as métricas. Nessa questão tive dificuldades principalmente quando se tratou do Kfold manual, não sei se está correta, mas este foi o resultado obtido:




**Análise:**

1. **RSS Médio (Residual Sum of Squares): 9778.43**

   O meu RSS está bem alto, e significa que meu modelo não está capturando bem a variabilidade nos dados.

2. **MSE Médio (Mean Squared Error)**: **130.60**

   Meu MSE de 130.60 pode ser considerado de certa forma alto, indicando que as previsões não estão próximas dos valores reais.

3. **RMSE Médio (Root Mean Squared Error): 11.39**

   O meu RMSE é de 11.39 e sugere que o modelo comete erros médios nas previsões.

4. **$R^2$ Médio (Coeficiente de Determinação): 0.218**

   Um **$R^2$ de 0.218** é baixo, indicando que o meu modelo explica apenas cerca de 21,8% da variabilidade nos dados.

Baseado nisso, creio que o meu modelo não está bom, pois não tratei o dataset adequadamente, possivelmente na normalização dos dados e escolha das features.

## 4. Conclusões

Apesar de enfrentar muitas dificuldades, principalmente na implementação manual, tanto do KNN quando do Kfold, acredito que em parte, os resultados foram satisfatórios. Mas ainda preciso de prática, inclusive no aprofundamento na parte de regressão, já que meu modelo não foi satisfatório.

## 5. Próximos passos

*Não sei o que sugerir no momento.*