UNIVERSIDADE FEDERAL DO CEARÁ

TECHNICAL REPORT

Aluno: Ana Julia Da Silva Freitas

1. Introdução

O dataset apresenta classificação de remédios. Ele possui 200 linhas com 6 colunas: Idade, Sexo, Pressão Arterial (PA), Colesterol, Razão Potássio/Sódio (Razão_KA_K) e Medicamento. As variáveis Idade e Razão_KA_K são quantitativas, enquanto as demais são qualitativas. O objetivo é prever qual medicamento é mais adequado com base nas demais características.

2. Observações

A variável alvo era a coluna Medicamento, inicialmente categórica. Transformei-a em tipo int para viabilizar os modelos de classificação. Durante os testes, a coluna se tornava float após separação treino/teste, causando erro. Corrigi isso fixando explicitamente seu tipo como int.

3. Resultados e discussão

3. QUESTÃO 1

O dataset foi carregado, traduzi os nomes das colunas e corrigi o tipo da coluna Medicamento. Analisei a estrutura com .info() e .describe() e identifiquei a natureza das variáveis. Apliquei value_counts() e proporção de frequências para variáveis qualitativas (Sexo, PA, Colesterol). Realizei também alguns gráficos e salvei o dataset modificado como classificacao_ajustado.csv.

	Idade	*Sexo*	*PA*	*Colesterol*	*Razãp_KA_K*	*Medicamento*
0	*23*	*F*	*HIGH*	*HIGH*	*25.355*	*0*
1	*47*	*M*	*LOW*	*HIGH*	*13.093*	*3*
2	*47*	*M*	*LOW*	*HIGH*	*10.1145*	*3*
3	*28*	*F*	*NORMAL*	*HIGH*	*7.798*	*1*
4	*61*	*F*	*LOW*	*HIGH*	*18.043*	*0*

#	Column	Non-Null Count	Dtype
0	Idade	200 non-null	int64
1	Sexo	200 non-null	object
2	PA	200 non-null	object
3	Colesterol	200 non-null	object
4	Razão_KA_K	200 non-null	float64
5	Medicamento	200 non-null	int64
dtypes:	float64(1)	int64(2)	object(3)

#	Idade	Razão_KA_K	Medicamento
count	200	200	200
mean	44.3315000	16.084485	1.060000
std	16.544315	7.223956	1.270619
min	15	6.269000	0
25%	31	10.445500	0
50%	45	13.936500	1
75%	58	19.380000	2
max	74	38.247000	4

Sexo	
M	*104*
F	*96*
Name:count	*dtype:int64*

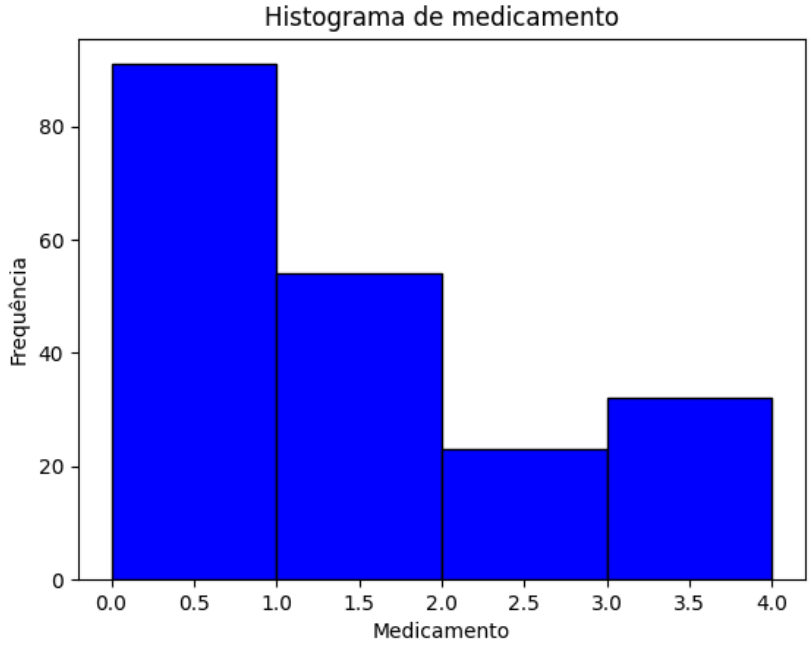
Sexo	
M	*0.52*
F	*0.48*
Name:proportion	*dtype:float64*

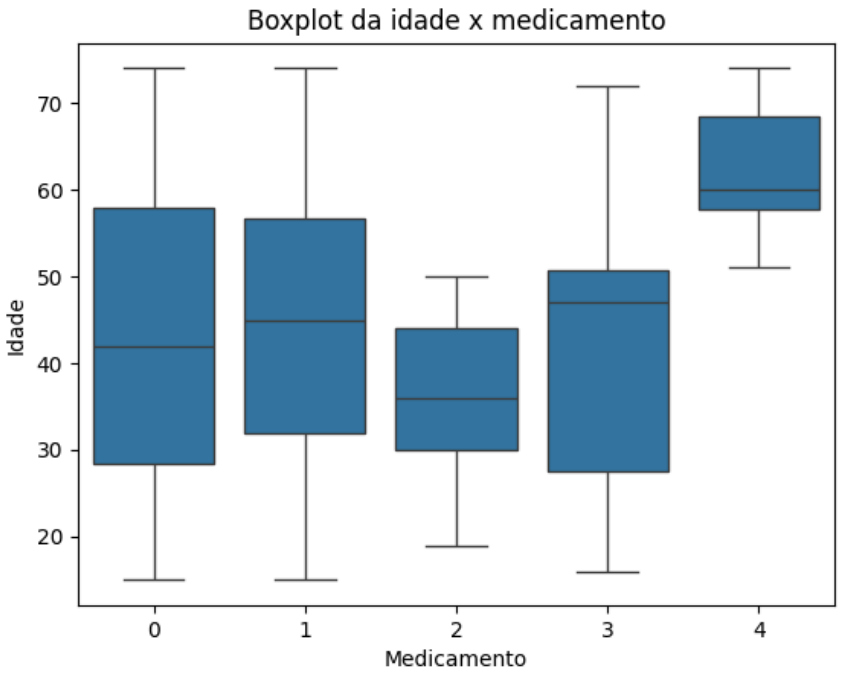
PA	
HIGH	*77*
LOW	*64*
NORMAL	*59*
Name:count	*dtype:int62*

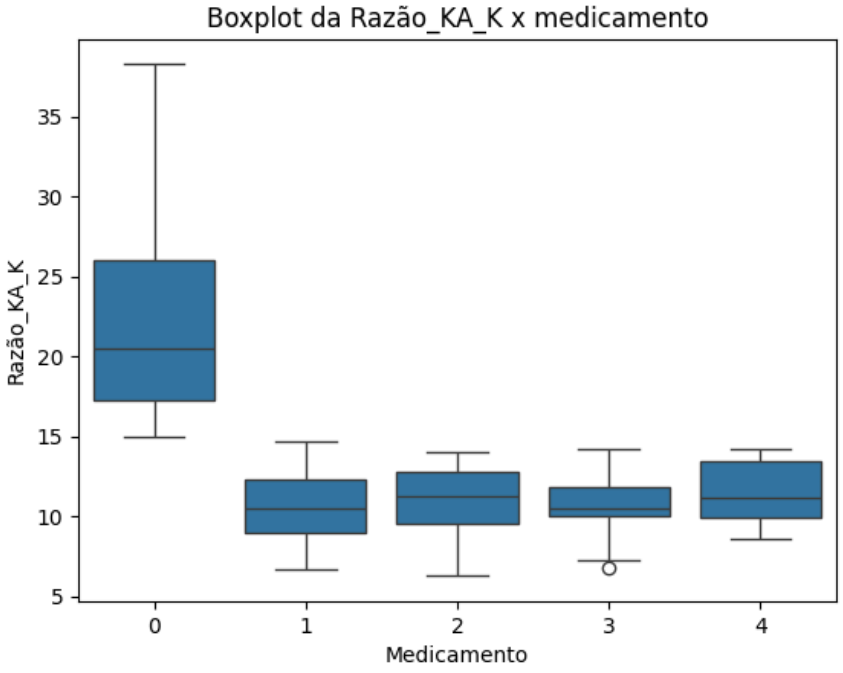
PA	
HIGH	*0.385*
LOW	*0.320*
NORMAL	*0.295*
Name:proportion	*dtype:float64*

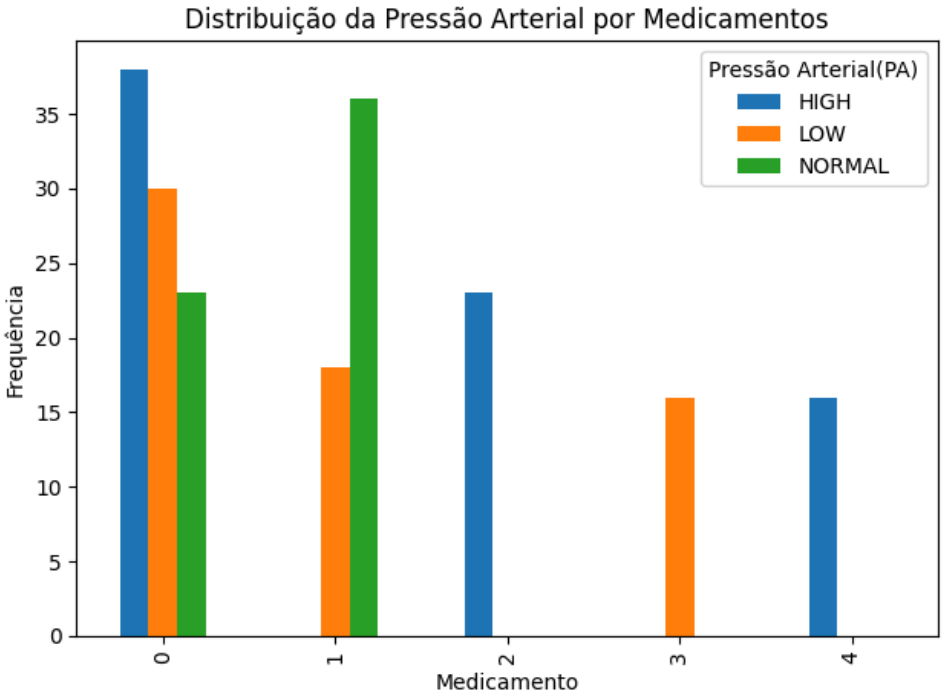
Colesterol	
HIGH	*103*
NORMAL	*97*
Name:count	*dtype:int64*

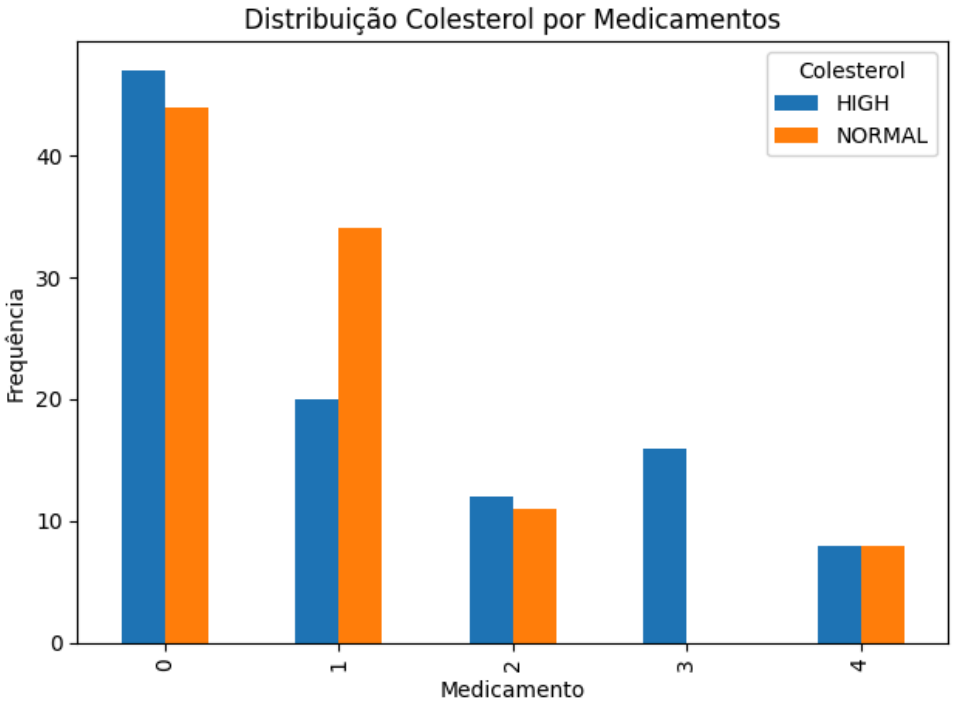
Colesterol	
HIGH	*0.515*
NORMAL	*0.485*
Name:proportion	*dtype:float64*

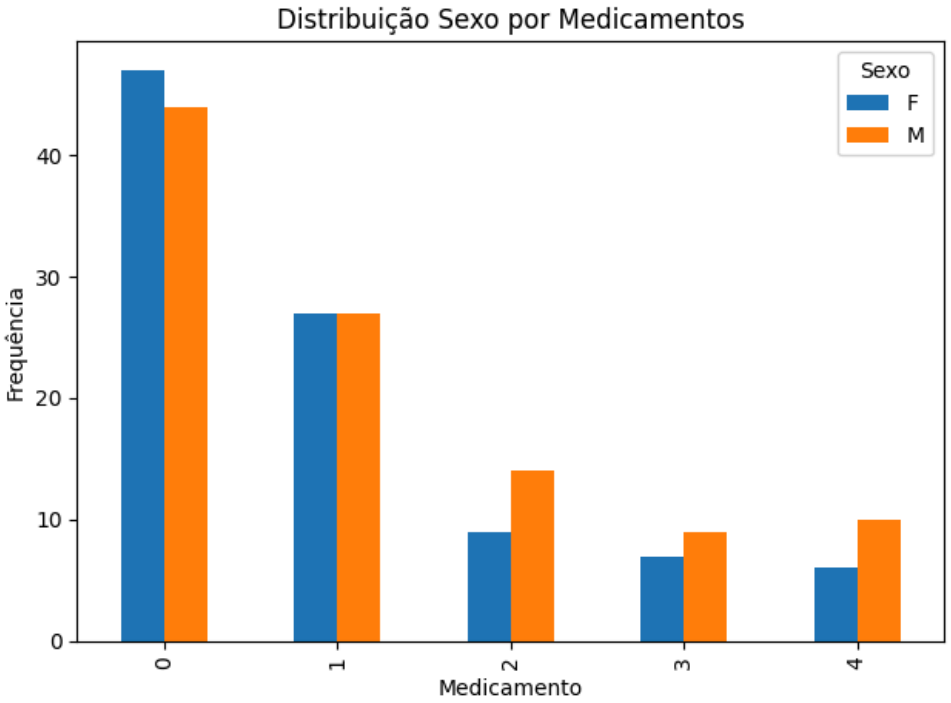



Boxplot da idade x medicamento
Idade
70
60
50
40
30
20
0
1
2
3
4
Medicamento


Boxplot da Razão_KA_K x medicamento
Razão_KA_K
35
30
25
20
15
10
5
0
1
2
3
4
Medicamento


Distribuição da Pressão Arterial por Medicamentos
Pressão Arterial(PA)
HIGH
LOW
NORMAL
Frequência
0
5
10
15
20
25
30
35
0
1
2
3
4
Medicamento


Distribuição Colesterol por Medicamentos
Colesterol
HIGH
NORMAL
Frequência
0
10
20
30
40
0
1
2
3
4
Medicamento



3. QUESTÃO 2

Implementei o algoritmo KNN manualmente testando as distâncias Euclidiana, Manhattan, Chebyshev e Mahalanobis. Os dados categóricos foram transformados em variáveis dummies, e todas as colunas convertidas para float.

Resultados de acurácia com k=5:

- *Mahalanobis: 0.725 (melhor resultado)*
- *Euclidiana: 0.650*
- *Manhattan: 0.650*
- *Chebyshev: 0.625*

3. QUESTÃO 3

Apliquei as normalizações: nenhuma, logarítmica, MinMaxScaler e StandardScaler. Testei todas com KNN manual (k=5, distância de Mahalanobis).

Melhor resultado: Normalização logarítmica com acurácia de 0.750.

Também explorei graficamente os efeitos da variação de k na acurácia para escolher o valor ideal.

3. QUESTÃO 4

Utilizei Pipeline e GridSearchCV com KNeighborsClassifier. Testei k de 1 a 20 e as normalizações: nenhuma, StandardScaler, MinMaxScaler.

Top 3 configurações (Acurácia média via CV):

- *k=1, StandardScaler, 0.89375*
- *k=17, StandardScaler, 0.87500*
- *k=14, StandardScaler, 0.86250*

3. QUESTÃO 5

Com a melhor configuração (StandardScaler + k=1), apliquei cross_val_score com 5 folds. Avaliei por confusion_matrix e classification_report.

Resultados:

- *Acurácia média: 0.905*
- *Desvio padrão: 0.01*
- *F1-score > 0.90 para as classes 1 e 2*

O modelo se mostrou robusto e generalizou bem.

4. Conclusões

Com base nos testes feitos, ficou claro que o KNN é um modelo que pode funcionar muito bem quando os dados são bem preparados. A escolha da distância certa (no caso, Mahalanobis) melhorou bastante a acurácia, mostrando que levar em conta a relação entre as variáveis faz diferença.

Além disso, percebi que normalizar os dados também teve impacto direto no desempenho. Usando a transformação logarítmica, o KNN manual teve seu melhor resultado. Já com o GridSearchCV, a normalização com StandardScaler foi a que mais contribuiu para a performance.

A escolha do valor de k também teve papel importante. Usando validação cruzada e gráficos, o valor ótimo foi encontrado, o que ajudou a evitar tanto o subajuste quanto o sobreajuste. No final, a configuração com k=1 e StandardScaler teve uma acurácia média de 90,5% e se mostrou bastante estável.

Esses resultados mostram como pequenos ajustes no pré-processamento e nos parâmetros podem melhorar bastante a performance do modelo. A classificação dos medicamentos com base nas variáveis disponíveis foi feita de forma eficiente e confiável.

5. Próximos passos

- *Continuar testando outros algoritmos além do KNN, como Árvore de Decisão, Random Forest e SVM, para comparar resultados e entender melhor as diferenças entre eles.*
- *Aprender e aplicar técnicas de redução de dimensionalidade, como PCA, para visualizar os dados e talvez melhorar a performance dos modelos.*
- *Explorar métodos para lidar com classes desbalanceadas, para que o modelo aprenda melhor em casos com poucos exemplos.*
- *Criar uma pequena aplicação onde o usuário possa inserir seus dados e receber a previsão do medicamento mais indicado com base no modelo treinado.*
- *Reforçar o entendimento sobre validação cruzada, normalização e ajuste de parâmetros, já que são etapas importantes para qualquer modelo de aprendizado de máquina.*