Universidade Federal de São Carlos
Centro de Ciências Exatas e de Tecnologia
Departamento de Estatística

Influência da normalização no desempenho de classificadores monolíticos e combinados

Emily Sato
Orientador: Prof. Dr. José Carlos Fogo

Trabalho de Conclusão de Curso

Universidade Federal de São Carlos
Centro de Ciências Exatas e de Tecnologia
Departamento de Estatística

Influência da normalização no desempenho de classificadores monolíticos e combinados

Emily Sato
Orientador: Prof. Dr. José Carlos Fogo

Trabalho de Conclusão de Curso apresentado como parte dos requisitos para obtenção do título de Bacharel em Estatística.

São Carlos
Fevereiro de 2025

Emily Sato

Influência da normalização no desempenho de classificadores monolíticos e combinados

Este exemplar corresponde à redação final do trabalho de conclusão de curso devidamente corrigido e defendido por Emily Sato e aprovado pela banca examinadora.

Aprovado em 19 de fevereiro de 2025

Banca Examinadora:

- Prof. Dr. José Carlos Fogo
- Profª. Drª. Teresa Cristina Martins Dias
- Prof. Dr. Thiago Rodrigo Ramos

Resumo

A classificação é um processo que utilizamos dados e algoritmos para categorizar objetos baseados em características específicas. Na estatística existem diferentes abordagens para classificação, como por exemplo: redes neurais, métodos ensemble, modelos estatísticos, máquinas de vetores de suporte (SVM), entre outros. Sendo assim, é importante escolher a abordagem com base nas características específicas do problema e dos dados.

O presente Trabalho de Graduação tem como objetivo analisar se a aplicação das técnicas de normalização nas variáveis influencia na melhoria do desempenho de classificadores monolíticos e combinados. Para isso, são empregadas técnicas de escalonamento (normalização) e avaliadas métricas de desempenho dos classificadores, tais como F1, precisão, especificidade e sensibilidade. O estudo inclui, ainda, uma definição estatística dos conceitos utilizados e uma revisão metodológica, além da fase de pré-processamento dos dados.

Palavras-chave: *classificação, normalização, transformação, medidas de performance.*

Sumário

Capítulo 1

Introdução

O problema de classificação e o problema de predição em regressão compartilham semelhanças, mas divergem em um aspecto fundamental: a natureza da variável resposta. Em um problema de classificação, essa variável não é quantitativa, mas sim qualitativa. Isso significa que estamos lidando com categorias (Izbicki e Santos, 2017). Por exemplo, a classificação determina se uma transação é fraudulenta.

A classificação é uma tarefa de aprendizado de máquina que consiste em categorizar um conjunto de dados em classes distintas. O objetivo é determinar a qual categoria (ou subpopulação) uma nova observação pertence, com base em um conjunto de variáveis previamente rotulado. Esse processo envolve a construção de um modelo que aprende padrões e relações entre as variáveis para realizar previsões precisas.

Assim como em toda a análise estatística, a etapa de pré-processamento dos dados é essencial para se obter um bom desempenho na aplicação de um classificador. Essa etapa é fundamental para a preparação, organização e estruturação dos dados antes da realização de análises e predições permitindo obter resultados mais confiáveis. Um aspecto importante desse processo é a normalização dos dados, o qual envolve a transformação de variáveis em um intervalo comum, em que o principal objetivo é adequar os dados de forma que cada variável varie dentro do mesmo intervalo (Amorim *et al.*, 2022).

A normalização de dados tem sido amplamente reconhecida como uma prática importante em diversos campos, desde a medicina até a análise de dados agrícolas. Em estudos prévios, foi comprovado que a normalização exerce um papel importante na melhoria do desempenho da classificação em uma variedade de cenários, abrangendo desde sistemas biométricos até previsões do mercado financeiro e detecção de falhas em motores. Sendo assim, essa prática tem se destacado como uma etapa da preparação dos dados

para a construção de modelos classificadores, contribuindo para resultados mais robustos e confiáveis (Dalwinder e Birmohan, 2020).

O objetivo neste trabalho é analisar o impacto da normalização das variáveis no desempenho dos classificadores. Para isso, aplicamos diferentes técnicas de normalização a diversos classificadores e a três conjuntos de dados distintos, a fim de identificar se há um padrão nos resultados ou se a influência da normalização varia conforme o classificador e o contexto dos dados.

Este trabalho é organizado como segue. No Capítulo 1 é apresentada a introdução do trabalho. No Capítulo 2 são detalhados exemplos de tipos de classificadores, incluindo os monolíticos (sendo estes: máquina de vetores de suporte, análise discriminante linear, análise discriminante quadrática e regressão logística) e os combinados (como bagging e florestas aleatórias), fornecendo definições detalhadas e explicações sobre o funcionamento de cada um desses classificadores. Em seguida temos a apresentação das técnicas de normalização e das medidas de desempenho. No Capítulo 3, descrevemos sobre detalhes do banco de dados e é apresentado a aplicação dos dados. Inicialmente, realizamos uma análise descritiva e exploratória, utilizando alguns diferentes gráficos para visualizar o comportamento das variáveis. Posteriormente, os dados são testados tanto sem normalização quanto com diferentes técnicas de normalização para identificar aquela que proporciona o melhor desempenho na classificação. Além disso, são apresentadas as matrizes de confudimento e as métricas de desempenho correspondentes, permitindo uma comparação detalhada entre as abordagens. Por fim, no Capítulo 4, apresentamos as considerações finais do trabalho.

Capítulo 2

Metodologia

2.1 Classificadores monolíticos

Os classificadores monolíticos se referem a um único classificador que faz todas as previsões com base em um único modelo. Por exemplo, um algoritmo de classificação como a máquina de vetores de suporte (SVM) ou uma árvore de decisão pode ser considerado um classificador monolítico, pois é responsável por toda a tarefa de classificação sem o auxílio de outros modelos.

2.1.1 Máquina de Vetores de Suporte (SVM)

O SVM é um algoritmo baseado na ideia de encontrar um hiperplano que melhor separe os dados em diferentes classes e é especialmente eficaz em problemas de classificação com dados não lineares.

No contexto de um problema de classificação, o SVM procura identificar um hiperplano que maximize a margem de separação entre as diferentes classes. Essa margem representa a maior distância possível entre os pontos mais próximos de cada classe e o hiperplano. Esses pontos, que estão localizados na margem, são conhecidos como vetores de suporte.

Ao determinar um hiperplano com a maior margem possível, o SVM busca aumentar a robustez do modelo, garantindo que as classes sejam separadas de forma mais confiável, o que reduz a chance de erros na classificação.

De acordo com Smola *et al.* (2000), o SVM se destaca por sua elevada capacidade de generalização, alcançando desempenhos consistentes em tarefas de classificação e regressão, mesmo no caso em que são aplicadas a dados que não foram utilizados durante o

treinamento. Na Figura 2.1 temos uma representação do funcionamento da técnica SVM em problemas de classificação:

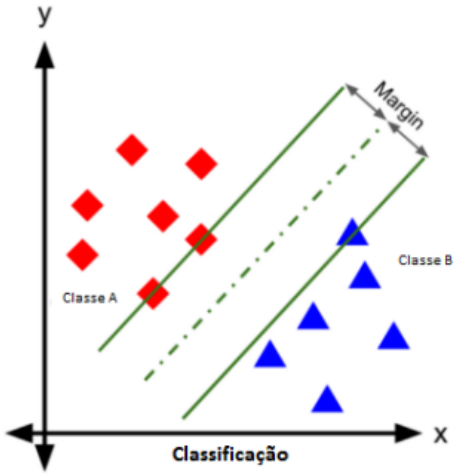

Figura 2.1: Representação da técnica de Máquinas de Vetores de Suporte em tarefas de classificação. Extraído de Achsan (2020).

2.1.2 Análise Discriminante Linear (LDA) e Quadrática (QDA)

No método de Bayes ingênuo, um algoritmo de classificação baseado no Teorema de Bayes (Izbicki e Santos, 2017), assumindo que todas as variáveis preditoras são independentemente relevantes para a classificação, a distribuição condicional das covariáveis é tratada como uma multiplicação de distribuições individuais, ou seja,

$$f(x \mid Y = c) = f(x_1, \ldots, x_d \mid Y = c) = \prod_{j=1}^{d} f(x_j \mid Y = c),$$

mas outras abordagens também podem ser utilizadas para estimar essa distribuição. Já na análise discriminante, assume-se que, dado $Y = c$, o vetor X segue uma distribuição normal multivariada. Existem duas abordagens principais dentro da análise discriminante, que são exploradas a seguir (Izbicki e Santos, 2017).

Ao depararmos com conjuntos de dados de alta dimensão, precisamos aplicar técnicas de redução de dimensionalidade para explorá-los e utilizá-los de forma eficaz na modelagem. A análise discriminante linear ou, mais conhecida como análise discriminante normal ou análise de função discriminante é uma técnica específica que nos permite mapear dados complexos para uma dimensão mais baixa sem comprometer significativamente a informação contida. Além disso, funciona como um algoritmo de aprendizado supervisio-

nado desenvolvido para tarefas de classificação. O propósito é encontrar uma combinação linear de características que melhor separe as classes dentro de um conjunto de dados, otimizando assim a discriminação entre elas.

O LDA funciona projetando os dados em um espaço de menor dimensão que maximiza a separação entre as classes. Nesta técnica são encontradas um conjunto de discriminantes lineares que maximizam a razão entre a variância entre classes e a variância dentro da classe. Em outras palavras, encontra as direções no espaço de características que melhor separam as diferentes classes de dados (Johnson e Wichern, 2002).

No LDA assumimos que os dados possuem distribuição gaussiana e que as matrizes de covariância das diferentes classes são iguais (Izbicki e Santos, 2017). Também assume que os dados são linearmente separáveis, o que significa que um limite de decisão linear pode classificar com precisão as diferentes classes. Assim, assume-se que:

$$\mathbf{X} = (X_1, \ldots, X_d) \mid Y = c \sim \text{Normal}(\mu_c, \Sigma),$$

isto é,

$$f(\mathbf{x} \mid Y = c) = \frac{1}{\sqrt{(2\pi)^d |\Sigma|}} \exp^{-(\mathbf{x}-\mu_c)^T \Sigma^{-1} (\mathbf{x}-\mu_c)}$$

O QDA é similar ao LDA pois permite uma modelagem mais flexível, adaptando-se melhor a conjuntos de dados nos quais as classes têm características distintas. Além disso, o QDA é menos restrito e permite diferentes matrizes de covariâncias de variáveis para diferentes classes, o que leva a um limite de decisão quadrático. O QDA é particularmente útil se houver conhecimento prévio de que classes individuais apresentam covariâncias distintas. Assim, assume-se que

$$\mathbf{X} = (X_1, \ldots, X_d) \mid Y = c \sim \text{Normal}(\mu_c, \Sigma_c),$$

isto é,

$$f(\mathbf{x} \mid Y = c) = \frac{1}{\sqrt{(2\pi)^d |\Sigma_c|}} \exp^{-(\mathbf{x}-\mu_c)^T \Sigma_c^{-1} (\mathbf{x}-\mu_c)}$$

Enquanto o LDA é eficaz em cenários com limites de decisão lineares, o QDA se destaca quando a relação entre as classes não é linear. Isso se deve à sua capacidade de modelar fronteiras de decisão mais complexas, permitindo uma melhor adaptação aos dados em situações mais desafiadoras. Em contrapartida, o QDA não é adequado para reduzir a

dimensionalidade dos dados, ou seja, não é uma ferramenta apropriada para reduzir a quantidade de variáveis em um conjunto de dados (Sharma, 2022).

2.1.3 Regressão Logística

A regressão logística é uma técnica estatística paramétrica amplamente utilizada para modelar a probabilidade de ocorrência de um evento binário, ou seja, quando a variável dependente assume apenas dois valores possíveis. A regressão logística tem como característica estimar uma função de regressão que pode ser parametrizada por um número finito de parâmetros, associados às covariáveis x_1, x_2, ..., x_d com o objetivo de encontrar os valores que melhor se ajustam aos dados.

A técnica de regressão logística modela a probabilidade de Y pertencer a uma das categorias, utilizando uma distribuição de bernoulli. Isso garante que as probabilidades calculadas variem entre 0 e 1. De acordo com Fernandes *et al.* (2020), esse processo envolve o ajuste de um modelo para prever valores de uma variável dependente categórica binária, com base em um conjunto de dados observados.

Sendo assim, ao ajustarmos uma linha reta a uma variável dicotômica podemos obter valores menores que 0 ou maiores que 1. Para evitar esse problema, modelamos $P(Y = 1|x)$ utilizando uma função que restringe os resultados ao intervalo entre 0 e 1. Na regressão logística, essa função é a função logística, que garante que a probabilidade de Y ser igual a 1 permaneça no intervalo entre [0,1].

$$P(Y = 1|x) = \frac{e^{\beta_0 + \sum_{i=1}^{d} \beta_i x_i}}{1 + e^{\beta_0 + \sum_{i=1}^{d} \beta_i x_i}}$$

em que $x = (1, x_1, \ldots, x_d)$ denota o vetor composto pela constante 1 e pelos valores observados das covariáveis, β_0 é uma constante e β_i são os d parâmetros de regressão associados às covariáveis (James *et al.*, 2013).

Para estimar os coeficientes de uma regressão logística, utilizamos o método de máxima verossimilhança (Izbicki e Santos, 2017), em que dada uma amostra independente e identicamente distribuída, $(X_1, Y_1), \ldots, (X_n, Y_n)$, a função de verossimilhança condicional é dada por:

$$L(\beta \mid x, y)) = \prod_{k=1}^{n} \Big(\mathbb{P}(Y_k = 1 \mid x_k, \beta)\Big)^{y_k} \Big(1 - \mathbb{P}(Y_k = 1 \mid x_k, \beta)\Big)^{1-y_k}$$

$$= \prod_{k=1}^{n} \left(\frac{e^{\beta_0 + \sum_{i=1}^{d} \beta_i x_{k,i}}}{1 + e^{\beta_0 + \sum_{i=1}^{d} \beta_i x_{k,i}}} \right)^{y_k} \left(\frac{1}{1 + e^{\beta_0 + \sum_{i=1}^{d} \beta_i x_{k,i}}} \right)^{1-y_k}.$$

Utilizando métodos numéricos para maximizar $\text{Ł}(\beta \mid x, y)$, obtemos as estimativas $\hat{\beta}_0, \hat{\beta}_1, \ldots, \hat{\beta}_d$, que correspondem aos parâmetros $\beta_0, \beta_1, \ldots, \beta_d$.

2.2 Classificadores combinados

Classificadores combinados integram as decisões de um conjunto de classificadores com o objetivo de gerar uma decisão final mais precisa do que as decisões de cada classificador individualmente. Estudos teóricos e empíricos demonstram que, em geral, um conjunto de classificadores é mais preciso do que um classificador monolítico (Amorim *et al.*, 2022).

2.2.1 *Bagging* e Florestas Aleatórias

Tanto o método de *bagging* quanto as florestas aleatórias utilizam ideia similar para melhorar as previsões feitas por árvores de decisão. Ambas as abordagens têm o objetivo de criar B árvores distintas e combinar seus resultados para aumentar o poder preditivo em comparação com uma única árvore. No caso do bagging, essas B árvores são criadas a partir de B amostras bootstrap extraídas da amostra original (Breiman, 2001).

A função de predição gerada pelo bagging é dada por:

$$g(\mathbf{x}) = \frac{1}{B} \sum_{b=1}^{B} g_b(\mathbf{x}),$$

no qual a função $g_b(\mathbf{x})$ é a função de predição obtida segundo a b-ésima árvore. Cada árvore é treinada de forma independente e, ao final, as predições de todas as árvores são combinadas para gerar a predição final.

Assim como o *bagging*, o método de florestas aleatórias consiste na criação de B árvores distintas, utilizando B amostras bootstrap extraídas da amostra original, mas as árvores construídas tendem a dar predições parecidas, em que cada nó só é permitido que seja escolhida uma dentre as $m < d$ covariáveis. Estas m covariáveis são escolhidas aleatoriamente dentre as covariáveis originais e, a cada nó criado, um novo subconjunto de covariáveis é sorteado (Izbicki e Santos, 2017).

A combinação das diferentes árvores de classificação é realizada por meio da aplicação da função:

$$g(\mathbf{x}) = moda\left\{g^b(\mathbf{x}), b = 1, ..., B\right\}.$$

É importante destacar que, o valor de m pode ser determinado por meio de validação cruzada. Resultados empíricos apontam que quando o número de variáveis selecionadas m corresponde a cerca de um terço do número total de variáveis d, o desempenho do modelo tende a ter boa perfomance. Esse ajuste proporciona bom equilíbrio entre a diversidade das árvores e a capacidade de modelar adequadamente os dados (Izbicki e Santos, 2017).

2.3 Normalização

Normalização e padronização, também identificadas como scaling techniques, são conceitos importantes em estatística e aprendizado de máquina, especialmente quando se trata de pré-processamento de dados. Embora ambos os métodos sejam usados para preparar dados para análise ou modelagem, eles têm abordagens diferentes.

De acordo com Dancker (2022), a normalização ajusta os valores de uma variável para um intervalo específico, comumente entre 0 e 1 ou -1 e 1. Isso garante que diferentes variáveis estejam na mesma escala, facilitando a comparação, evitando assim que uma variável com valores muito grandes ou muito pequenos domine o processo de modelagem.

Ao padronizar dados, as variáveis estarão na mesma escala, com média igual a 0 e uma variação igual a 1. Isso os torna mais comparáveis e fáceis de analisar, já que foram colocados todos na mesma escala, facilitando a compreensão de suas relações e padrões.

A escolha sobre qual técnica normalizadora utilizar depende do contexto específico da aplicação. A normalização é comumente empregada quando os valores dos dados têm limites bem definidos, como no processamento de imagens. Já a padronização é uma técnica amplamente aplicada em aprendizado de máquina, especialmente quando é importante que as características dos dados se assemelhem a uma distribuição normal padrão.

Embora o termo *scaling* possa ser interpretado de várias maneiras, neste trabalho, vamos usar o termo para se referir especificamente à técnica de normalização.

Diante disso, a escolha de se utilizar ou não as técnicas de *scaling* deve ser feita criteriosamente, visto que estudos anteriores destacaram que optar pela técnica inadequada pode ter um impacto mais negativo no desempenho da classificação do que não escalonar os dados (Amorim *et al.*, 2022).

Sendo assim, abordamos algumas técnicas de normalização, começando pela normalização **min-max**, que é definida pela fórmula:

$$x' = \frac{x - x_{min}}{x_{max} - x_{min}},$$

em que x é o valor original, x' é o valor normalizado, x_{min} é o valor mínimo e x_{max} é o valor máximo da variável.

A escala min-max ajusta cada variável individualmente para estar dentro de um intervalo entre 0 e 1. A ideia é transformar os valores para que todos estejam na mesma escala, independentemente de seus valores originais. É útil para algoritmos de otimização, como o gradiente descendente, amplamente empregado em técnicas de aprendizado de máquina que atribuem pesos às entradas, como em modelos de regressão e redes neurais.

Porém, essa abordagem é sensível a outliers, o que pode ser um problema se seus dados tiverem valores extremos. Além disso, como a escala depende dos valores mínimo e máximo da variável, pequenas variações nos dados podem resultar em grandes mudanças na escala.

Outra técnica para tornar os dados comparáveis é a do **desvio padrão**, ou conhecida também como Z-Score, que transforma os dados em uma distribuição com média 0 e desvio padrão igual a 1. É dada pela fórmula:

$$x' = \frac{(x - \bar{x})}{s},$$

em que x é o valor original, x' é o valor normalizado, $\bar{x}$ é a média dos valores e s é o desvio padrão das observações.

Outra fórmula para normalização é o **escalonamento robusto**, que ajusta cada variável independentemente com base nos seus quartis amostrais. Geralmente, usa-se o intervalo interquartil para essa tarefa. Isso é útil porque torna a abordagem mais robusta contra valores extremos, garantindo que outliers não influenciem muito no resultado final (Dancker, 2022).

Esse método procura mitigar os efeitos de outliers, centralizando os dados em torno da mediana $Q_2(\mathbf{x})$ e escalonando-o de acordo com o intervalo interquartil, que é a magnitude da diferença entre o primeiro quartil $Q_1(\mathbf{x})$ e o terceiro quartil $Q_3(\mathbf{x})$ de $\mathbf{x}$, conforme

mostrado na equação abaixo:

$$x' = \frac{x_i - Q_2(\mathbf{x})}{Q_3(\mathbf{x}) - Q_1(\mathbf{x})} \ .$$

Uma possibilidade muito poderosa é a **transformação quantílica**, ao invés de, apenas ajustar a escala dos dados, realizar uma transformação não linear. Isso implica em alterar a forma da distribuição original dos atributos, o que pode ser vantajoso, uma vez que evidências sugerem que ajustar os atributos para seguir uma distribuição semelhante à gaussiana pode melhorar o desempenho de classificação (Amorim *et al.*, 2022).

Essa técnica funciona transformando cada variável individualmente. Primeiro, determina a distribuição cumulativa empírica da variável e a usa para mapear os valores originais para uma distribuição uniforme. Em seguida, mapeia esses valores para a distribuição desejada, como, por exemplo, uma distribuição normal. Os valores que estão fora do intervalo da distribuição desejada são ajustados para os limites dessa distribuição.

Uma vantagem importante dessa técnica é sua robustez, pois não é afetada por valores extremos nos dados. Além disso, os resultados são consistentes, independentemente dos atributos originais fornecidos.

Essa transformação é especialmente valiosa para tornar as variáveis comparáveis, especialmente quando têm escalas e formas de distribuição diferentes. No entanto, como é uma transformação não linear, pode distorcer as correlações entre variáveis medidas na mesma escala.

2.4 Medidas de perfomance

As medidas de desempenho de um modelo são utilizadas para avaliar o quão eficaz ele é em prever ou classificar dados, comparando suas previsões com os valores reais. Diante disso, a precisão da classificação, embora seja uma métrica comum, pode ser enganosa em conjuntos de dados altamente desbalanceados. Por exemplo, se a maioria das variáveis pertence a uma única classe, simplesmente classificar todas as variáveis como pertencentes a essa classe dominante pode resultar em uma precisão artificialmente alta (Amorim *et al.*, 2022).

Por isso, é importante escolher métricas que considerem o desbalanceamento das classes. Optamos por usar a métrica escore F1 (Santos, 2017). Nessa métrica são levados

em conta tanto os verdadeiros positivos (VP) e verdadeiros negativos (VN), quanto os falsos positivos (FP) e falsos negativos (FN) em uma classificação. O resultado de uma classificação pode ser resumido na matriz de confusão, ou confundimento (ver Tabela 2.1), que é usada para descrever o desempenho de classificação, ou seja, é uma representação tabular das previsões feitas pelo modelo em comparação com as classes reais dos dados.

Tabela 2.1: Resultado da classificação ou matriz de confundimento.

		Situação Real		
		1 (P)	0 (N)	Totais
Classificado como	1 (P)	VP	FP	(VP + FP)
	0 (N)	FN	VN	(VN + FN)
	Totais	(VP + FN)	(VN + FP)	

Os verdadeiros positivos são observações originalmente positivos (classe 1), corretamente classificadas como positivos, enquanto os verdadeiros negativos, ou VN, são as observações originalmente negativos (classe 0) corretamente classificadas como negativos. Os falsos positivos (FP) são erroneamente classificados como positivos, e os falsos negativos (FN) são as observações erroneamente classificados como negativos.

A métrica F1 é a média harmônica entre a precisão (proporção de verdadeiros positivos entre todos as observações classificados como positivos) e o *recall* (ou sensibilidade), que é definida pela equação:

$$F1 = 2 \times \frac{Precisão \times Sensibilidade}{Precisão + Sensibilidade}\,, \tag{2.1}$$

sendo a precisão e sensibilidade dadas por

$$Precisão = \frac{VP}{VP + FP} \quad \text{e,}$$

$$Sensibilidade = \frac{VP}{VP + FN}\,.$$

Essa métrica fornece uma avaliação mais robusta do desempenho do modelo em conjuntos de dados desbalanceados, pois levam em conta tanto os acertos quanto os erros de classificação em ambas as classes. A interpretação das medidas de desempenho citadas anteriormente é descrita a seguir:

- **Estatística F1**: fornece uma medida única de desempenho que leva em consi-

deração tanto os verdadeiros positivos quanto os falsos positivos e falsos negativos.

- **Sensibilidade**: essa métrica avalia a proporção de observações positivas identificadas corretamente pelo modelo em relação ao número total de observações que são verdadeiramente positivas. Em suma, ela quantifica a capacidade do modelo em capturar de forma precisa os casos positivos.

- **Especificidade**: essa métrica expressa a proporção de observações negativas identificadas corretamente pelo modelo em relação ao total de observações verdadeiramente negativas. Em resumo, ela avalia quão precisamente o modelo é capaz de identificar casos negativos.

- **Precisão**: é uma medida que representa a fração de resultados positivos corretos em relação a todos os resultados positivos previstos pelo modelo, em outras palavras, ela indica a proporção de previsões positivas que o modelo fez corretamente.

- **Taxa de Erro**: também conhecida como Taxa de Erro Aparente (APER), essa medida refere-se à proporção de observações no conjunto de dados que foram classificadas incorretamente pelo modelo preditivo. Ou seja, basicamente é o total de casos classificados incorretamente dividido pelo total de observações.

No desenvolvimento do trabalho, portanto, são aplicadas diferentes normalizações aos dados e avaliados os efeitos dessas normalizações no desempenho dos classificadores, monolíticos ou combinados.

2.5 Dados Desbalanceados e Ponto de Corte

Dados desbalanceados ocorrem quando as classes dentro de um conjunto de dados possuem quantidades desiguais de exemplos. Esse desequilíbrio pode representar um desafio para os modelos de aprendizado de máquina, pois eles podem acabar favorecendo a classe majoritária, comprometendo a capacidade de prever corretamente a classe minoritária.

Ao lidar com dados desbalanceados, uma estratégia frequentemente utilizada é o ajuste do limiar de decisão na classificação. Em problemas binários, o valor padrão desse limiar costuma ser 0,5, o que significa que, se a probabilidade prevista de um exemplo pertencer à classe 1 for superior a esse valor, o modelo atribui essa classe.

Para contornar esse problema, uma abordagem comum é explorar diferentes valores em vez de fixá-lo em 0,5. Isso permite ajustar a sensibilidade do modelo, buscando um equilíbrio mais adequado entre as classes e melhorando a capacidade de prever corretamente a classe minoritária. Dessa forma, buscamos definir a função de decisão da seguinte maneira:

$$g(x) = I(P(Y = 1 \mid x) \geq K)$$

Capítulo 3

Aplicação

Nesta seção é feita a aplicação de diversas técnicas de normalização em três conjuntos de dados distintos, analisando o comportamento das variáveis de cada base de dados. Será utilizado diferentes classificadores e comparado seus desempenhos por meio das matrizes de confusão, a fim de avaliar a capacidade de classificação correta dos dados, tanto com quanto sem normalização. Além disso, será empregada várias métricas de desempenho para medir a performance de cada classificador em cada conjunto de dados.

3.1 Aplicação no conjunto de dados de diabetes

O conjunto de dados, originado do *National Institute of Diabetes and Digestive and Kidney Diseases* de *Maryland*, EUA, é uma amostra de um banco de dados maior e contém informações de 768 mulheres. Esses dados fornecem informações para avaliar a incidência de diabetes em mulheres indígenas. A amostra inclui mulheres com pelo menos 21 anos e com herança genética indígena *Pima*. O banco de dados é composto por 9 covariáveis, sendo:

Tabela 3.1: Definição das variáveis do conjunto de dados.

Variáveis	**Definição**
Gravidezes	Número de gravidezes
Glicose	Concentração de glicose plasmática a 2 horas num teste oral de tolerância à glicose
Pressão	Pressão arterial diastólica (mmHg)
Insulina	Insulina sérica de 2 horas (micro UI/ml)
Dobra	Espessura da dobra da pele do tríceps (mm)
IMC	Índice de massa corpórea (kg/m^2)
DPF	Função Pedigree de Diabetes - mede a predisposição genética de desenvolver diabetes
Idade	Idade (anos)
TesteDiabete	positivo e negativo

Nessa primeira aplicação, devido à natureza das normalizações, é considerada apenas as variáveis quantitativas. O objetivo é classificar (para identificar) quem tem ou não diabetes.

É importante salientar que não consideramos os dados que são iguais a zero em medidas que não há razão de ser zero, assim, essas informações zeradas foram consideradas como *missing* e excluídas do banco de dados.

3.1.1 Resultados

Para compreender o comportamento das variáveis no banco de dados em estudo, realizamos análises descritivas. Foram feitos histogramas para cada covariável, comparando os gráficos para as variáveis sem normalização e normalizadas pelas técnicas escalonamento robusto, desvio-padrão e min-max. Dessa forma, é possível compará-los e visualizar o comportamento das covariáveis em cada umas das técnicas aplicadas.

Além disso, aplicamos um LDA nos dados normalizados e sem normalizar, com o objetivo de determinar qual técnica de normalização proporciona os melhores resultados na classificação da presença ou ausência de diabetes.

3.1.2 Análise Descritiva e Exploratória dos Dados

A seguir são apresentados alguns resultados da análise descritiva realizada com o software *R*.

Nas Figuras 3.1 a 3.8 são apresentados os histogramas das variáveis em estudo, sendo que, na linha superior à esquerda temos os histogramas sem a normalização e à direita, com a normalização do escalonamento robusto. Na linha inferior, por sua vez, são apresentados os histogramas com as normalizações pelo desvio padrão e min-max, na esquerda e direita, respectivamente.

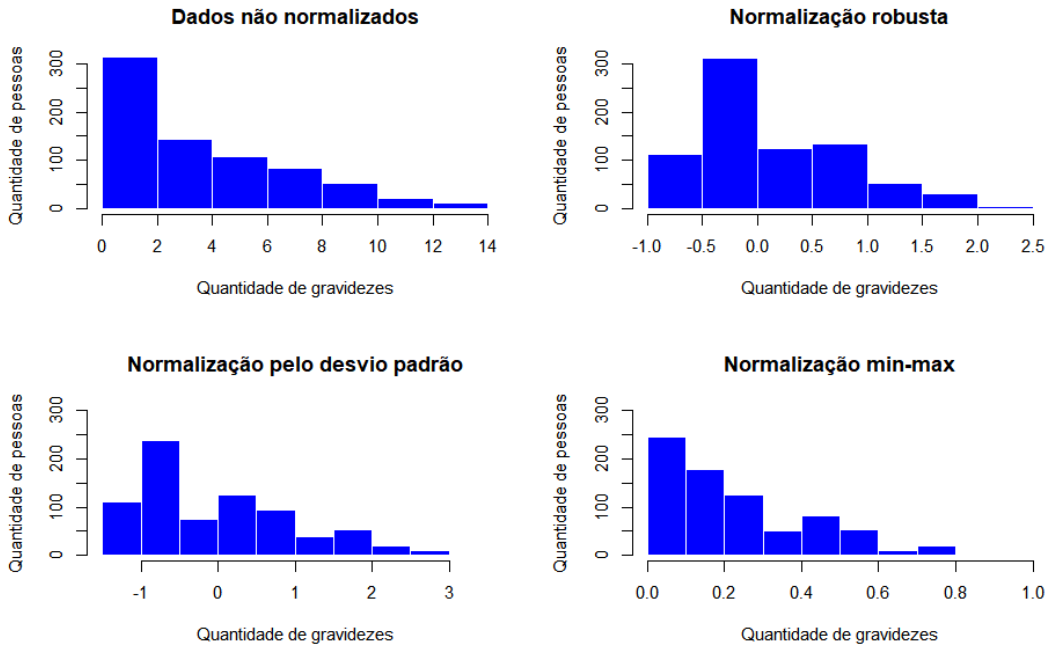

Figura 3.1: Histogramas do número de gravidezes.

Na Figura 3.1 vemos o comportamento da variável gravidezes. É possível observar que os histogramas via técnicas de normalizações por escalonamento robusto e desvio padrão seguem uma mesma ideia.

Já a normalização min-max notamos que não centraliza os dados, reescalonando-os de 0 a 1 e possui o valor mínimo na origem e o valor máximo no 1, capturando com mais detalhes a variabilidade. Vemos o mesmo comportamento também para os histogramas das variáveis idade e DPF (Figura 3.8 e Figura 3.7, respectivamente).

Para a covariável glicose (Figura 3.2), podemos observar uma similaridade entre os histogramas, no qual os dados se concentram, em sua maioria, em um ponto central. Além disso, nota-se que para a concentração de glicose uma forte assimetria positiva, que

pode indicar a presença de outliers. Além disso, temos duas observações fora do padrão na cauda inferior do histograma, abaixo da concentração 50.

Na variável pressão (Figura 3.3) observa-se que os histogramas possuem um comportamento similar, em que todos apresentam boa simetria, com a maior parte dos dados centralizados em torno de sua média.

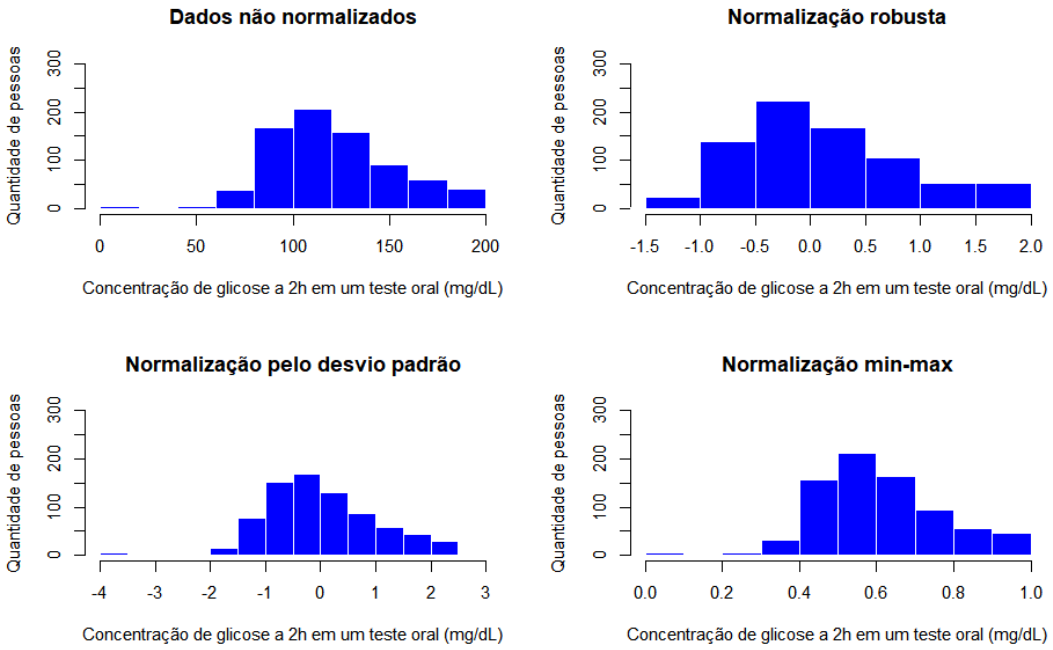

Figura 3.2: Histogramas da concentração de glicose.

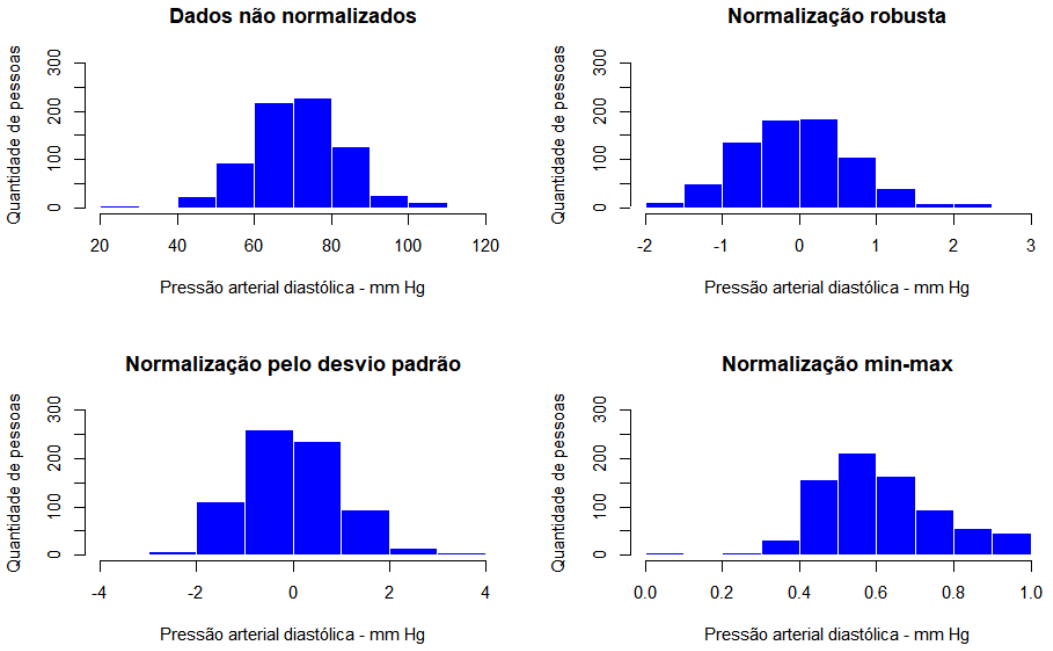

Figura 3.3: Histogramas da Pressão Arterial Diastólica

Com relação à variável insulina (Figura 3.4) nota-se que os dados apresentam forte assimetria à direita, possibilitando a presença de valores discrepantes.

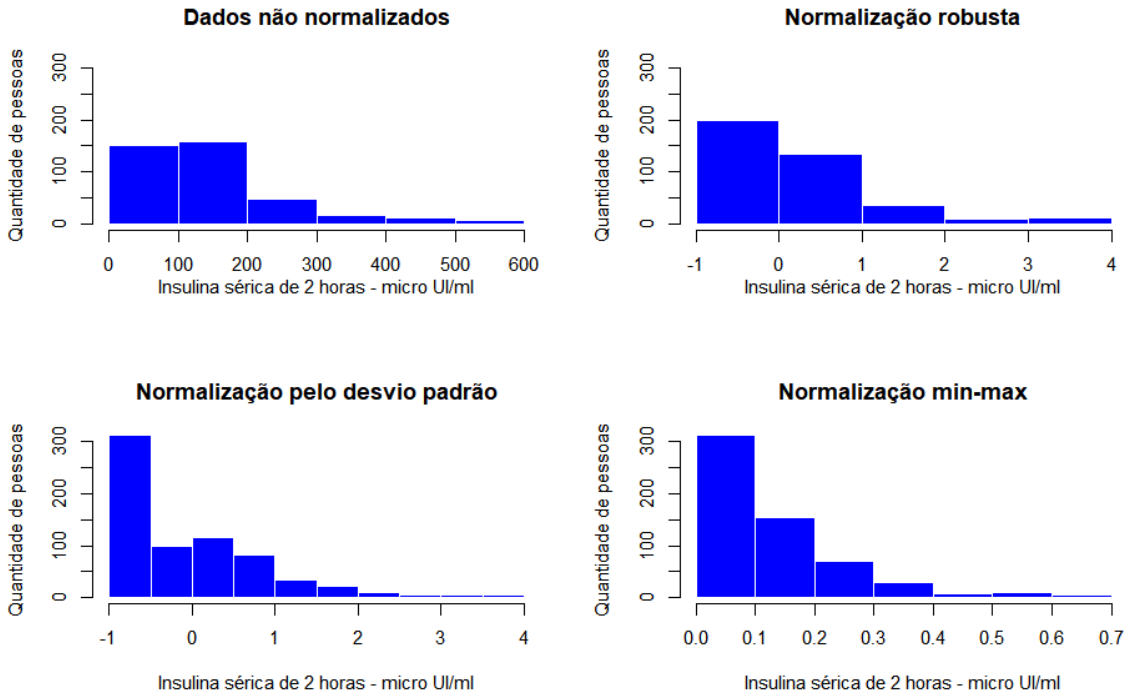

Figura 3.4: Histogramas da insulina sérica de 2 horas.

Nos histogramas da variável dobra (Figura 3.5) os dados não normalizados e escalonados pela normalização robusta são parecidos, com a concentração dos dados em torno de sua média. Notamos que os dados normalizados pelo desvio padrão e pela técnica min-max também possuem comportamentos similares e um pico na frequência baixa de espessura da dobra, variando bastante conforme a espessura vai aumentando.

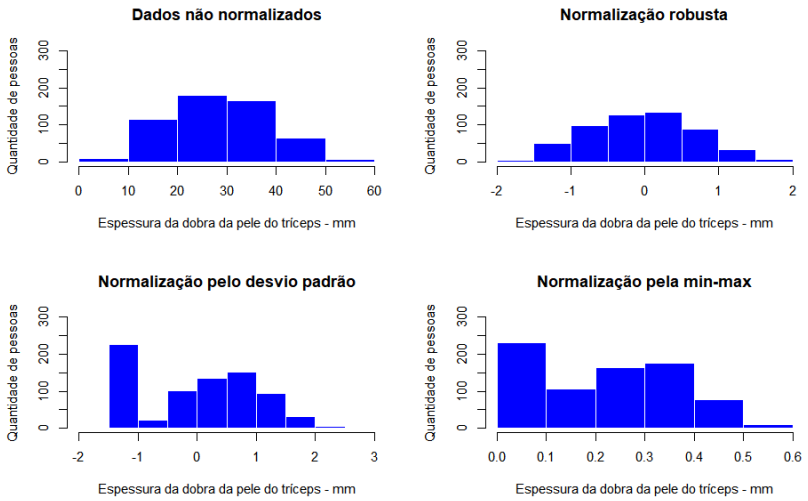

Figura 3.5: Histogramas da espessura da dobra da pele do tríceps.

Os histogramas dos dados da variável IMC (Figura 3.6) mostram ligeira assimetria à direita, tanto antes como após as normalizações.

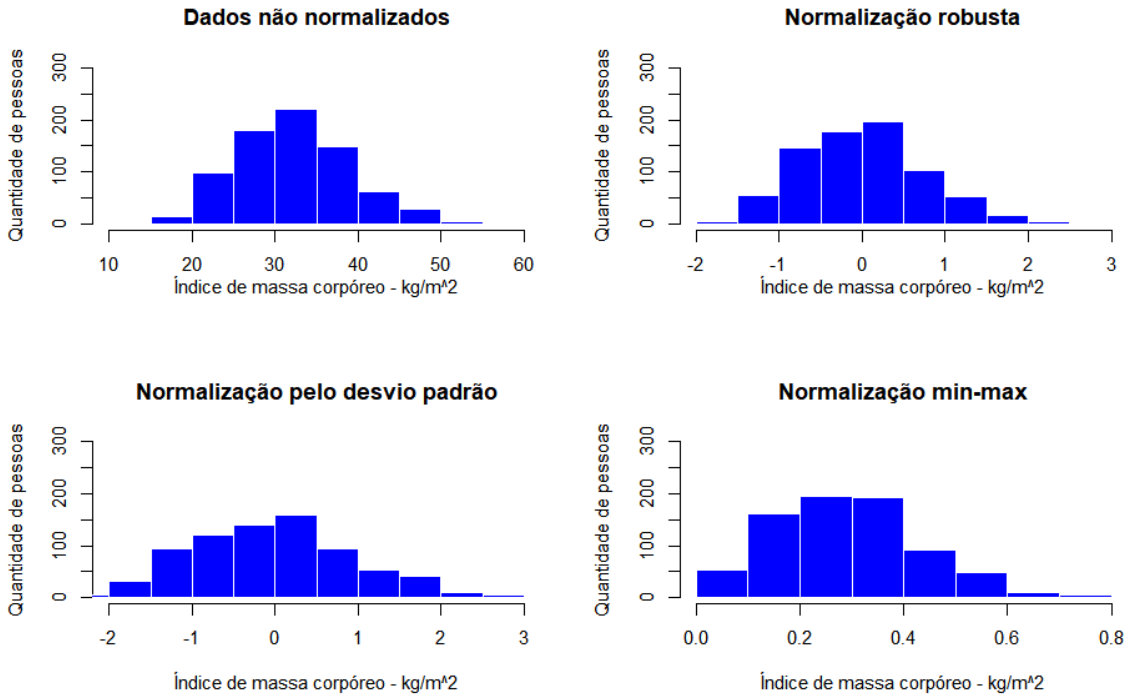

Figura 3.6: Histogramas do índice de massa corpóreo.

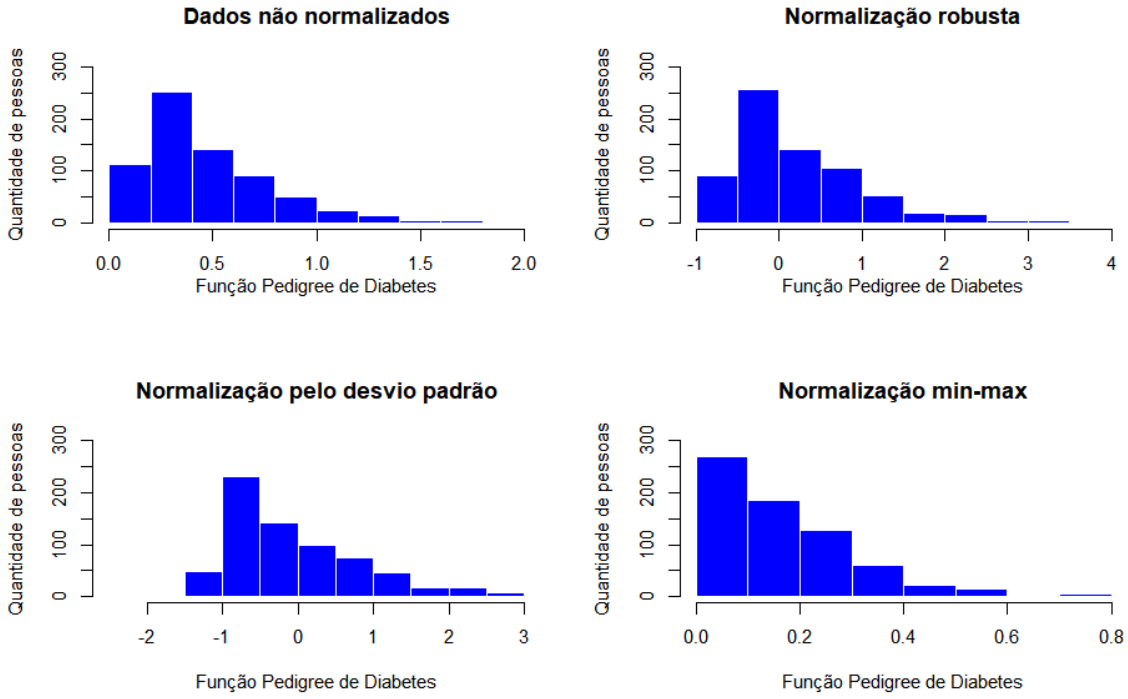

Figura 3.7: Histogramas da Função de Diabetes.

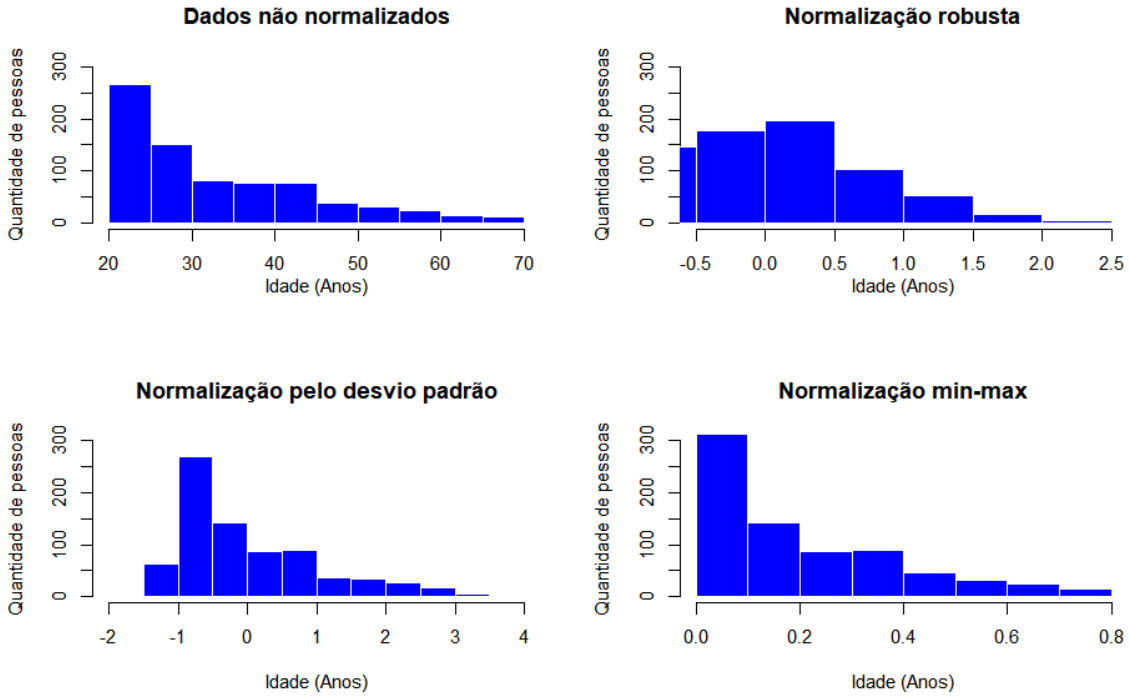

Figura 3.8: Histogramas das idades em anos.

3.1.3 Análise Discriminante Linear (LDA) e Quadrática (QDA)

Nesta seção, apresentamos os resultados das análises discriminantes linear (LDA) e quadrática (QDA) realizadas com o software R.

Para isso, ajustamos cada conjunto de dados em quatro condições distintas: sem normalização, normalização pela técnica de escalonamento robusto, desvio padrão e min-max. Após esses ajustes, aplicamos os classificadores a cada conjunto de dados e geramos as matrizes de confusão correspondentes. Em seguida, utilizamos métricas de desempenho para avaliar se melhoraram a eficácia da classificação. O objetivo nesta análise é comparar o desempenho dos diferentes métodos de normalização para determinar qual método proporciona o melhor desempenho na classificação da presença ou ausência de diabetes.

Na Tabela 3.2, observamos que o modelo, ao ser aplicado aos dados originais, classificou corretamente 59+156=215 casos. Além disso, foram identificados 28 casos como falsos positivos, ou seja, casos que o modelo previu como positivos, mas que, na realidade, são negativos.

Tabela 3.2: Matriz de confundimento da classificação sem a normalização dos dados.

		Situação Real		
		pos	neg	Totais
Classificado como	pos	59	28	87
	neg	32	156	188
	Totais	91	184	

Na Tabela 3.3, observamos que o modelo, aplicado aos dados normalizados pela técnica de escalonamento robusto, classificou corretamente 60+141=201 casos. Além disso, foram identificados 31 casos como falsos positivos e 43 casos como falsos negativos, que são casos que foram erroneamente classificados como negativos, mas são positivos.

Tabela 3.3: Matriz de confundimento da classificação com a normalização do escalonamento robusto.

		Situação Real		
		pos	neg	Totais
Classificado como	pos	60	43	103
	neg	31	141	172
	Totais	91	184	

Já na Tabela 3.4 tivemos 54+156=210 casos classificados de forma correta, destacando-se como a técnica que apresentou o maior número de classificações corretas. Também, tivemos 37 casos falsos positivos e 28 falsos negativos.

Tabela 3.4: Matriz de confundimento da classificação com a normalização pelo desvio padrão.

		Situação Real		
		pos	neg	Totais
Classificado como	pos	54	30	84
	neg	37	154	191
	Totais	91	184	

Na Tabela 3.5, observamos que o modelo classificou corretamente 63+143=206 casos. No entanto, essa técnica apresentou 28 casos de falsos positivos e destaca-se por ser a técnica com mais casos de falsos negativos.

Tabela 3.5: Matriz de confundimento da classificação com a normalização min-max.

		Situação Real		
		pos	neg	Totais
Classificado como	pos	63	41	104
	neg	28	143	171
	Totais	91	184	

Tabela 3.6: Resultado das medidas de desempenho do LDA.

Medidas	Sem normalizar	Escalonamento Robusto	Desvio Padrão	Min-max
Sensibilidade	0,648	0,659	0,593	0,648
Especificidade	0,848	0,766	0,848	0,793
Precisão	0,678	0,583	0,659	0,624
F1	0,663	0,619	0,624	0,656
Taxa de Erro	0,218	0,269	0,236	0,240

As medidas de desempenho calculadas incluem especificidade, sensibilidade, escore F1, precisão e taxa de erro para cada condição do conjunto de dados. Observando os resultados na Tabela 3.6, nota-se que os casos sem normalização apresentam um desempenho superior em termos de especificidade e precisão, pois para essas métricas, quanto maiores os valores, melhor o desempenho.

No caso da especificidade, os melhores desempenhos foram obtidos tanto pelos casos sem normalização quanto pelas normalizadas utilizando a técnica de desvio padrão. Além disso, todas as demais apresentaram valores de precisão bastante próximos entre si.

A técnica de escalonamento robusto foi a que registrou a maior taxa de erro aparente, sendo, a menos recomendada.

Os resultados das medidas de desempenho são apresentados de forma mais clara na Figura 3.9. Nesta figura são exibidos quatro gráficos de barras distintos, cada um correspondente a uma medida de desempenho específica. Os gráficos de barras comparam os valores obtidos dispostos lado a lado para facilitar a comparação.

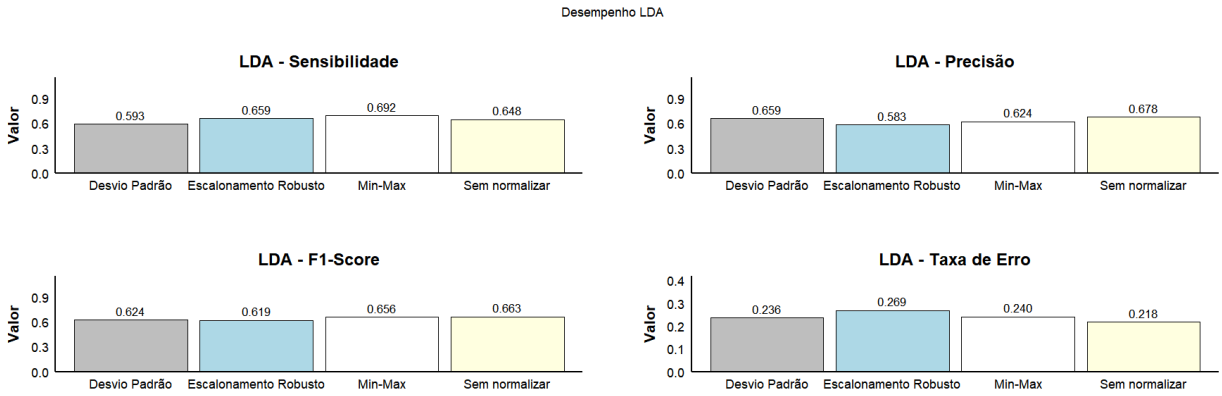

Figura 3.9: Gráfico de barras das medidas de desempenho.

Na Tabela 3.7 são apresentadas uma visão geral das medidas de desempenho obtidas ao aplicar a análise discriminante quadrática. Observa-se que os dados sem normalização exibem melhores valores de sensibilidade e taxa de erro, enquanto as técnicas de normalização por desvio padrão e min-max apresentam valores de especificidade e precisão bastante semelhantes. De modo geral, as métricas de desempenho são bastante próximas entre si, assim como ocorreu com o modelo LDA, não havendo uma diferença significativa entre os dados originais e os dados normalizados.

Tabela 3.7: Resultado das medidas de desempenho do QDA.

Medidas	Sem normalizar	Escalonamento Robusto	Desvio Padrão	Min-max
Sensibilidade	0,908	0,859	0,864	0,788
Especificidade	0,462	0,505	0,593	0,615
Precisão	0,773	0,778	0,811	0,806
F1	0,835	0,817	0,837	0,797
Taxa de Erro	0,240	0,258	0,225	0,269

3.2 Aplicação no conjunto de dados sobre câncer de mama

O conjunto de dados em análise contém 569 observações e 12 covariáveis. Entre essas covariáveis, destacam-se o diagnóstico do câncer (benigno ou maligno) e 10 covariáveis numéricas que levam a medidas laboratoriais extraídas por aspiração com agulha fina de células mamárias.

A Tabela 3.10 mostra a descrição das variáveis presentes no banco de dados.

Tabela 3.8: Variáveis do conjunto de dados e suas descrições.

Variável	**Descrição**
diagnosis	Diagnóstico do câncer: benigno (B) ou maligno (M).
radius	Distância do centro aos pontos do perímetro do núcleo celular.
texture	Variações dos níveis de cinza na imagem do núcleo.
perimeter	Perímetro do núcleo celular.
area	Área do núcleo celular.
smoothness	Variação local nos comprimentos dos raios.
compactness	Complexidade do contorno.
concavity	Severidade das porções côncavas do contorno.
concave.points	Número de pontos côncavos no contorno.
symmetry	Simetria do núcleo celular.
fractal dimension	Complexidade do perímetro.

Calculando a proporção de cada categoria da variável resposta, observamos o seu comportamento, no qual notamos que a maioria dos indivíduos, em torno de 62,74% são classificados como benigno e 37,26% como maligno.

3.2.1 Análise descritiva e exploratória dos dados

A análise descritiva e exploratória dos dados é uma etapa importante para podermos compreender o comportamento dos dados, como a presença de padrões, tendências e anomalias. De início, analisando o histograma da Figura 3.10, observamos uma assimetria positiva nas variáveis radius, perimeter e area. Já a variável texture é a que apresenta melhor simetria.

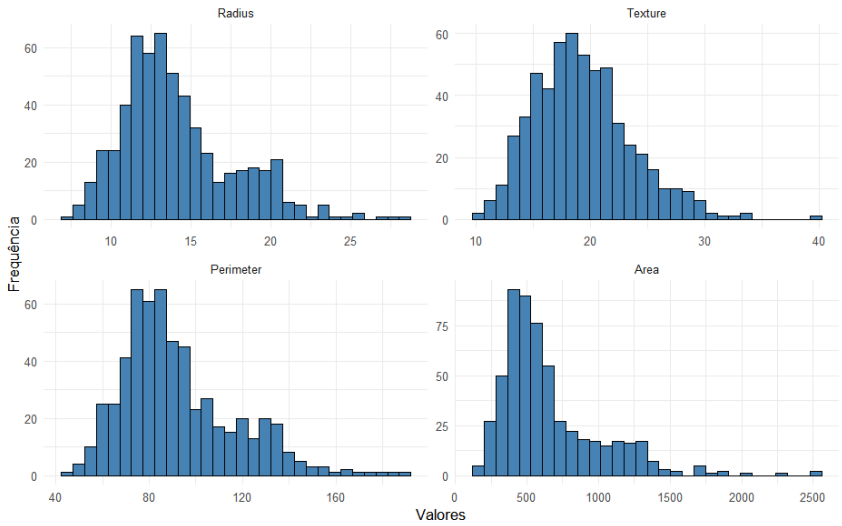

Figura 3.10: Histograma das covariáveis radius, texture, perimeter e area.

Para facilitar a visualização da distribuição das diferentes classes e compreender melhor o comportamento neste conjunto de dados, incluindo possíveis padrões, realizando uma análise de componentes principais (PCA) (Johnson e Wichern, 2007).

O gráfico da Figura 3.11 apresenta o PCA com dois grupos distintos correspondendo a dois diagnósticos: maligno e benigno. A separação clara entre os grupos ao longo dos componentes principais (PC1 e PC2) indica que há uma diferença significativa nos padrões dos dados que os distinguem.

Uma boa separação sugere que os grupos possuem características suficientemente distintas para que uma técnica de classificação ou diagnóstico consiga diferenciar com eficácia. Isso é um indicativo positivo de que o modelo baseado nos dados originais pode ter alta assertividade na discriminação entre os dois grupos.

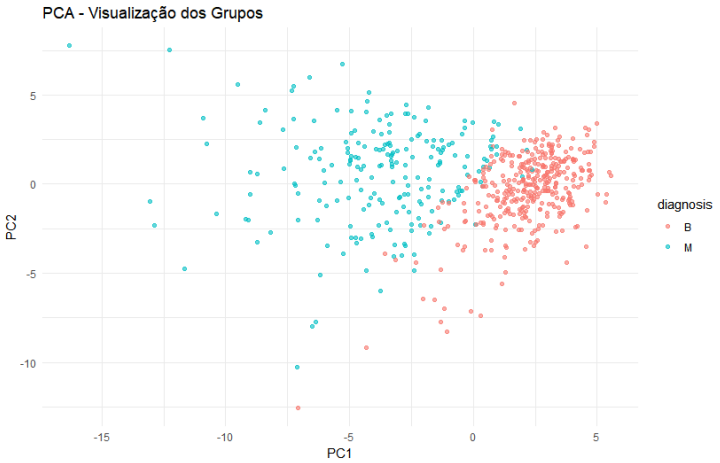

Figura 3.11: Análise dos Componentes Principais.

3.2.2 Classificação

A análise descritiva e exploratória dos dados revela, por meio do gráfico de componentes principais, que os dois grupos da variável resposta (benigno e maligno) apresentam uma separação significativa, evidenciando que já se distinguem de forma clara. Isso indica que os dados, em sua forma original, possuem uma classificação eficaz. Por esse motivo, a aplicação de classificadores ou técnicas de normalização não deve resultar em melhorias substanciais no desempenho, dado que a separação natural entre os grupos já é expressiva. Ainda assim, para garantir uma avaliação completa, realizamos uma análise de forma geral para verificar o desempenho desses classificadores mesmo após a aplicação das normalizações, buscando confirmar essa observação inicial.

Para uma melhor compreensão de quais modelos se destacam em cada medida de

desempenho, foi elaborado um gráfico de radar, que facilita a análise comparativa ao evidenciar os pontos fortes e fracos de cada classificador. Sendo assim, o gráfico (3.12) apresenta uma comparação entre diferentes modelos de classificação: LDA, QDA, RF e SVM em relação a várias medidas de desempenho: sensibilidade, acurácia, escore F1 e especificidade. Cada modelo é representado por uma linha colorida, e o tamanho da área coberta reflete o desempenho geral.

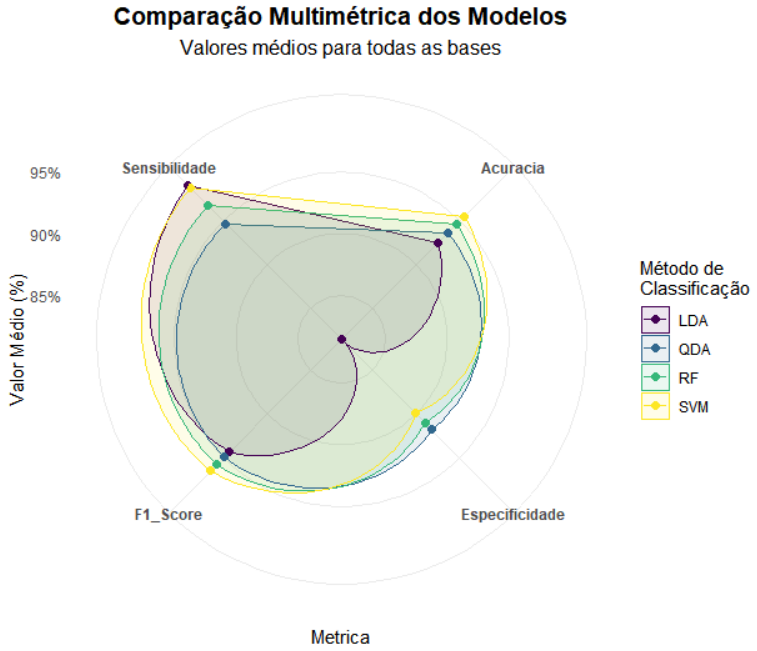

Figura 3.12: Comparação Multimétrica dos Modelos.

Sendo assim, observa-se que todos os modelos apresentam alta acurácia, indicando um bom desempenho geral nessa métrica. No entanto, as diferenças começam a surgir na sensibilidade, no qual o SVM e o LDA se destacam, demonstrando maior capacidade de identificar verdadeiros positivos.

Já em relação a especificidade, apenas o classificador LDA apresenta valor baixo, o que sugere menor capacidade de identificar verdadeiros negativos. Em contrapartida, o escore F1, que equilibra precisão e sensibilidade, também evidencia discrepâncias entre os métodos, refletindo a habilidade de cada um em evitar falsos positivos enquanto identifica corretamente os positivos. No geral, modelos como RF e SVM parecem oferecer um desempenho mais equilibrado, cobrindo áreas maiores no gráfico, enquanto LDA e QDA demonstram desempenhos mais específicos, com maior variação entre as métricas.

Com base na análise do gráfico (3.13), observa-se que as porcentagens de acurácia são bastante semelhantes entre as diferentes técnicas de normalização quando analisadas individualmente para cada classificador. Isso ocorre porque, por meio da análise dos

componentes principais, foi possível identificar uma separação linear clara entre as duas classes. Em outras palavras, a escolha da técnica de normalização não parece influenciar significativamente a acurácia dos classificadores. Portanto, concluímos que os classificadores mantêm uma performance praticamente constante, independentemente do método de normalização nestas aplicações.

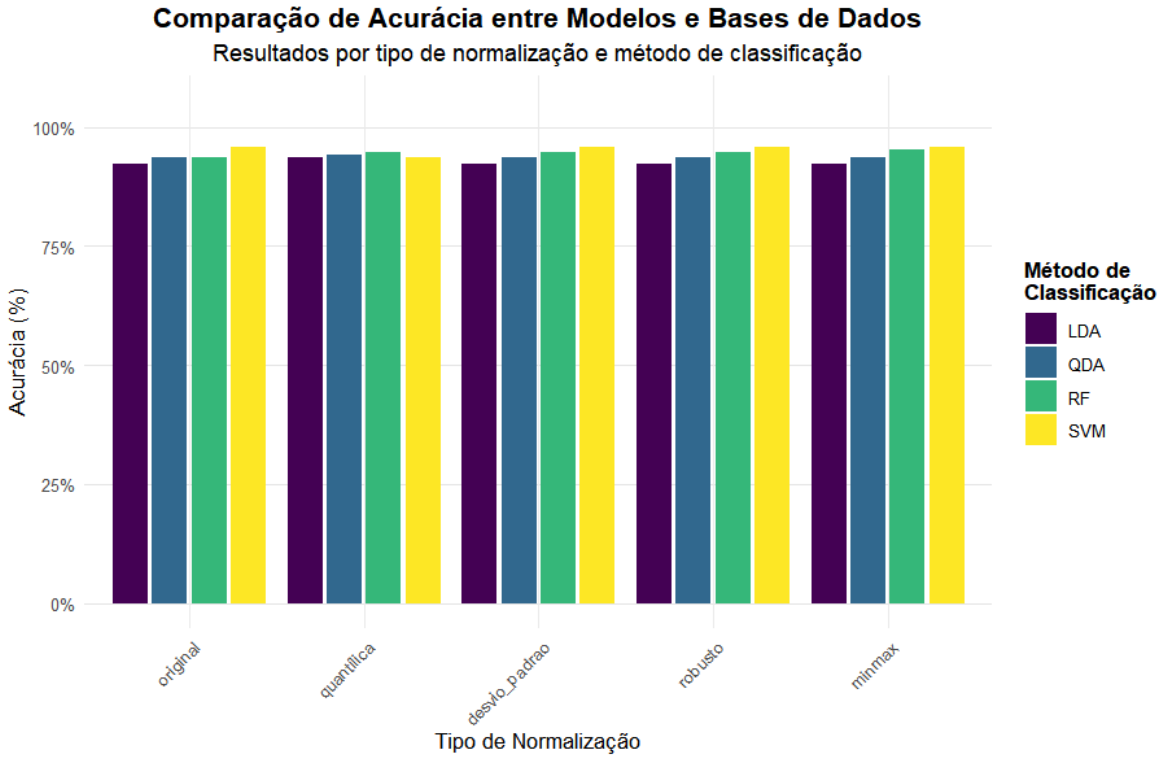

Figura 3.13: Gráfico de barras entre modelos e técnicas de normalização.

Na Tabela 3.9 nota-se que, entre todos os classificadores e as diferentes técnicas de normalização, as medidas de desempenho apresentam valores semelhantes, independentemente de as variáveis serem normalizadas ou não. Isso indica que a normalização das variáveis não resulta em uma diferença significativa no desempenho. Além disso, os valores muito próximos entre si sugerem que, para este conjunto de dados, a normalização das variáveis não é relevante.

Tabela 3.9: Desempenho dos classificadores com diferentes técnicas de normalização

Classificador	Normalização	Acurácia	Sensibilidade	Especificidade	Escore F1
LDA	Sem normalizar	0,9235	0,9907	0,8095	0,9422
	Desvio padrão	0,9235	0,9907	0,8095	0,9422
	Minmax	0,9235	0,9907	0,8095	0,9422
	Quantílica	0,9353	0,9907	**0,8413**	0,9507
	Robusto	0,9235	0,9907	0,8095	0,9422
QDA	Sem normalizar	0,9353	0,9439	0,9206	0,9484
	Desvio padrão	0,9353	0,9439	0,9206	0,9484
	Minmax	0,9353	0,9439	0,9206	0,9484
	Quantílica	0,9412	**0,9626**	0,9048	**0,9537**
	Robusto	0,9353	0,9439	0,9206	0,9484
RF	Sem normalizar	0,9353	0,9533	0,9048	0,9488
	Desvio padrão	0,9471	0,9720	0,9048	0,9585
	Minmax	0,9529	0,9720	0,9206	**0,9630**
	Quantílica	0,9471	0,9720	0,9048	0,9585
	Robusto	0,9471	0,9626	0,9206	0,9581
SVM	Sem normalizar	0,9588	0,9907	0,9048	0,9680
	Desvio padrão	0,9588	0,9907	0,9048	0,9680
	Minmax	0,9588	0,9907	0,9048	0,9680
	Quantílica	0,9353	0,9720	0,8730	0,9498
	Robusto	0,9588	0,9907	0,9048	0,9680

3.3 Aplicação no conjunto de dados sobre acupuntura

A base de dados "Acupuncture" é derivada de um estudo científico rigoroso intitulado "Acupuncture for chronic headache in primary care: large, pragmatic, randomized trial", publicado em 2004 (Vickers *et al.*, 2004). Este conjunto contém a informação de 401 participantes que sofriam de dores de cabeça crônicas e que concordaram em participar da pesquisa. Os participantes foram recrutados em 36 diferentes centros de atenção primária.

Este estudo investiga a eficácia da acupuntura como tratamento para dores de cabeça crônicas, abrangendo tanto enxaquecas quanto cefaleias tensionais. A pesquisa compara um grupo de pacientes que recebeu tratamento com acupuntura a um grupo controle que não recebeu tal tratamento. Na Tabela 3.10, temos a descrição de cada covariável:

Tabela 3.10: Variáveis do conjunto de dados e suas descrições.

Variável	**Descrição**
age	Idade do participante no início do estudo (em anos).
sex	Sexo do participante: 1 para feminino, 0 para masculino.
migraine	Diagnóstico do tipo de dor de cabeça: 1 para enxaqueca, 0 para cefaleia tensional.
chronicity	Número de anos que o participante sofre de dores de cabeça.
acupuncturist	Identificação do acupunturista que tratou o paciente (aplicável apenas ao grupo de acupuntura).
practice_id	Identificação da clínica geral onde o participante foi recrutado.
group	Grupo de tratamento ao qual o participante foi alocado: 1 para acupuntura, 0 para controle.
pk1	Pontuação da gravidade da dor de cabeça no início do estudo (*baseline*). Escala aprox. 0-100.
pk2	Pontuação da gravidade da dor de cabeça 3 meses após o início do estudo. Escala aprox. 0-100.
pk5	Pontuação da gravidade da dor de cabeça 1 ano após o início do estudo. Escala aprox. 0-100.
f1	Frequência da dor de cabeça no início do estudo (*baseline*).
f2	Frequência da dor de cabeça 3 meses após o início do estudo.
f5	Frequência da dor de cabeça 1 ano após o início do estudo.

É importante ressaltar a presença de valores ausentes (NA's) nas variáveis pk2 e pk5, referentes à pontuação da gravidade da dor nos acompanhamentos de 3 meses e 1 ano, respectivamente. Este aspecto deverá ser considerado nas análises futuras. Adicionalmente, a análise inicial sugere que a escala de pontuação da gravidade da dor (pk1, pk2, pk5) varia aproximadamente de 0 a 100.

O principal objetivo do estudo foi determinar o efeito da terapia com acupuntura em comparação com a ausência de tratamento com acupuntura em diversos aspectos relacionados à dor de cabeça crônica. A gestão dos dados faltantes é uma consideração importante nas análises subsequentes, em que é também aplicar classificadores.

3.3.1 Métodos de imputação

Os métodos de imputação são técnicas utilizadas para lidar com valores ausentes (NA's) em conjuntos de dados. Esses métodos consistem em substituir os valores faltantes por estimativas baseadas nas características dos dados, evitando a perda de informações e permitindo que análises estatísticas ou modelagens sejam realizadas de forma consistente.

A literatura nos oferece diversos métodos de imputação, cada um com suas características, conforme descrito a seguir:

Imputação pela Média do Grupo

A imputação pela média do grupo consiste em substituir os valores ausentes pela média do respectivo grupo. Essa abordagem preserva as diferenças médias entre os grupos de tratamento e é de fácil implementação e interpretação. No entanto, uma desvantagem é que pode reduzir a variabilidade dos dados, subestimando a incerteza, além de não considerar as relações entre as variáveis.

Imputação por Regressão

A imputação por regressão linear prevê os valores ausentes com base em outras variáveis do conjunto de dados. Para imputar pk2, são consideradas idade, sexo, diagnóstico, cronicidade, grupo e pk1. Já para pk5, incluem-se todas essas variáveis mais pk2. Essa abordagem permite estimativas mais precisas ao manter as relações entre as variáveis, porém é sensível a outliers e pressupõe relações lineares, o que pode não ser realista. Esse foi o método utilizado para substituir os valores faltantes, pois é o método que torna o modelo menos enviesado.

3.3.2 Criação da variável resposta

No presente estudo, foram realizadas transformações, tratamentos e a criação de novas variáveis para enriquecer as análises. A seguir, descrevemos as principais variáveis derivadas.

Foi criada a variável *resposta_tratamento*, que indica se o paciente obteve ou não uma redução de pelo menos 70% na severidade da dor de cabeça em 3 meses. A lógica utilizada foi:

$$resposta\_tratamento = \begin{cases} 1, & \text{se } \dfrac{\texttt{pk1} - \texttt{pk2}}{\texttt{pk1}} \times 100 > 70\% \\ 0, & \text{caso contrário,} \end{cases}$$

em que:

- pk1: severidade da dor no início do estudo (*baseline*).
- pk2: severidade da dor após 3 meses.

Se a redução percentual na severidade foi superior a 70%, o paciente foi considerado como tendo respondido ao tratamento (*resposta_tratamento* = 1), ou seja, teve uma melhora significativa. Caso contrário, foi classificado como não-respondente (*resposta_tratamento* = 0), então a dor de cabeça se manteve estável.

3.3.3 Tratamento de Dados

O tratamento de dados foi realizado em diversas etapas, visando melhorar a qualidade dos dados e a eficácia dos modelos analíticos subsequentes.

- Algumas covariáveis foram convertidas para o tipo fator (categórico), como group, sex, migraine e resposta_tratamento, pois são utilizadas como preditores categóricos ou como variável resposta nos modelos analíticos;
- As variáveis numéricas, tanto derivadas quanto originais, foram verificadas e confirmadas como do tipo numérica, garantindo a correta interpretação por parte dos algoritmos e análises subsequentes.

3.3.4 Classificação

Nesta subseção apresentamos os resultados obtidos por meio da implementação dos diferentes classificadores utilizando o *R*. Sendo assim, antes de treinar os modelos de classificação para prever se paciente teve uma redução significativa na dor de cabeça, vamos normalizar as variáveis numéricas utilizando 4 diferentes técnicas de normalização: min-max, robusto, desvio padrão e transformação quantílica.

Neste conjunto de dados, é explorada também a técnica de *boosting* utilizando o pacote *XGBoost* para avaliar seu desempenho e comportamento, que otimiza a combinação de múltiplas árvores de decisão para melhorar os resultados. Em seguida, avaliamos métricas como acurácia, sensibilidade, especificidade e escore F1. Ao aplicar regressão logística, florestas aleatórias, SVM e *boosting*, temos:

Tabela 3.11: Desempenho dos classificadores com diferentes técnicas de normalização

Classificador	Normalização	Acurácia	Sensibilidade	Especificidade	F1
Regressão Logística	Nenhum	0,725	0,533	0,789	0,492
	MinMax	0,725	0,533	0,789	0,492
	Quantil	0,700	0,467	0,778	0,438
	Robust	0,708	0,500	0,778	0,462
	Standard	0,708	0,500	0,778	0,462
Florestas Aleatórias	Nenhum	0,758	0,300	0,911	0,383
	MinMax	0,750	0,333	0,889	0,400
	Quantil	**0,775**	**0,333**	**0,922**	**0,426**
	Robust	0,742	0,300	0,889	0,367
	Standard	0,733	0,267	0,889	0,333
SVM	Nenhum	0,733	0,400	0,844	0,429
	MinMax	0,725	**0,433**	0,822	0,441
	Quantil	0,700	0,333	0,822	0,357
	Robust	0,717	0,367	0,833	0,393
	Standard	0,700	0,300	0,833	0,333
Boosting	Nenhum	**0,783**	**0,500**	**0,878**	**0,536**
	MinMax	0,750	0,400	0,867	0,444
	Quantil	0,783	0,400	0,911	0,480
	Robust	0,775	0,467	0,878	0,509
	Standard	0,758	0,400	0,878	0,453

Analisando as medidas de desempenho, podemos ver que o classificador de regressão logística teve o menor desempenho ao comparar os valores de especificidade com os demais classificadores mas teve melhores resultados de sensibilidade em todas as normalizações. O desempenho do SVM também foi bom, mas ao avaliar todas as métricas de desempenho, observamos que os modelos de florestas aleatórias com a técnica transformação quantílica e boosting sem normalizar apresentam as melhores medidas de desempenho.

A seguir, apresentamos as matrizes de confusão de todos os classificadores, permitindo uma comparação detalhada entre o desempenho sem normalização dos dados e com a normalização utilizando a técnica que, com base nas tabelas acima (3.11), obteve os melhores resultados para cada classificador.

Avaliando o desempenho da regressão logística na Tabela 3.12, observa-se que a aplicação da técnica de escalonamento não trouxe impacto significativo. O modelo apresentou resultados idênticos tanto com os dados normalizados quanto com os dados sem normalização.

Tabela 3.12: Resultados da classificação via regressão logística sem a normalização dos dados.

		Situação Real		
		Estabilidade	Melhoria	Totais
Classificado como	Estabilidade	71	14	85
	Melhoria	19	16	35
	Totais	90	30	

Tabela 3.13: Resultados da classificação via regressão logística com a normalização min-max dos dados.

		Situação Real		
		Estabilidade	Melhoria	Totais
Classificado como	Estabilidade	71	14	85
	Melhoria	19	16	35
	Totais	90	30	

Conforme observado nas Tabelas 3.14 e 3.15, o modelo de florestas aleatórias apresentou um desempenho ligeiramente superior ao utilizar a técnica de transformação quantílica para normalizar os dados. Com a normalização, o classificador previu corretamente 83+10=93 casos, em comparação com 82+9=91 casos corretos sem a normalização. Embora a diferença seja pequena, os resultados indicam que, para este conjunto de dados, a aplicação da técnica de normalização contribui para uma melhoria no desempenho do modelo.

Tabela 3.14: Resultados da classificação via Florestas Aleatórias sem a normalização dos dados.

		Situação Real		
		Estabilidade	Melhoria	Totais
Classificado como	Estabilidade	82	21	103
	Melhoria	8	9	17
	Totais	90	30	

Tabela 3.15: Resultados da classificação via Florestas Aleatórias com a normalização quantílica dos dados.

		Situação Real		
		Estabilidade	Melhoria	Totais
Classificado como	Estabilidade	83	20	103
	Melhoria	7	10	17
	Totais	90	30	

Conforme apresentado nas Tabelas 3.16 e 3.17, o modelo classificou corretamente o mesmo número de casos em ambos os cenários, totalizando 94 indivíduos. No entanto,

observa-se uma diferença na distribuição das classificações. Nos dados normalizados com a transformação quantílica, aproximadamente 80% dos indivíduos foram classificados como não apresentando uma melhora significativa na dor de cabeça após a acupuntura. Por outro lado, nos dados sem normalização, a maioria dos indivíduos (26 casos) foi classificada como tendo uma melhora significativa.

Tabela 3.16: Resultados da classificação via XGBoost sem a normalização dos dados.

		Situação Real		
		Estabilidade	Melhoria	Totais
Classificado como	Estabilidade	79	15	94
	Melhoria	11	15	26
	Totais	90	30	

Tabela 3.17: Resultados da classificação via XGBoost com a normalização quantílica dos dados.

		Situação Real		
		Estabilidade	Melhoria	Totais
Classificado como	Estabilidade	82	18	100
	Melhoria	8	12	20
	Totais	90	30	

Com base nos resultados das matrizes de confusão das Tabelas 3.18 e 3.19, a aplicação da normalização Min-Max no modelo SVM não trouxe uma melhoria significativa no desempenho geral, mas gerou uma leve redistribuição das classificações, priorizando uma maior sensibilidade para identificar "Melhoria".

Tabela 3.18: Matriz de confundimento do modelo SVM sem a normalização dos dados.

		Situação Real		
		Estabilidade	Melhoria	Totais
Classificado como	Estabilidade	76	18	94
	Melhoria	14	12	26
	Totais	90	30	

Tabela 3.19: Matriz de confundimento do modelo SVM com a normalização min-max dos dados.

		Situação Real		
		Estabilidade	Melhoria	Totais
Classificado como	Estabilidade	74	17	91
	Melhoria	16	13	29
	Totais	90	30	

Com base na Tabela 3.11, com o resumo das medidas, a normalização quantílica apresenta uma leve melhoria no desempenho das florestas aleatórias, refletida em melhorias nos valores de acurácia, especificidade, sensibilidade e F1. Por outro lado, as demais técnicas de normalização não exibem melhorias consistentes nos resultados, sugerindo que seus impactos sobre o modelo são pouco significativos.

Resumidamente, este conjunto de dados em particular e com os modelos empregados, a normalização não resultou em um aprimoramento significativo no desempenho geral dos classificadores.

- Para florestas aleatórias, a normalização quantílica parece melhorar levemente o desempenho do classificador;
- Para XGBoost, a opção de não aplicar a normalização parece ser a mais adequada, ou, no mínimo, equiparável às opções com normalização, além de apresentar maior simplicidade;
- Para regressão logística e SVM, o impacto da normalização é menos evidente e consistente, com algumas técnicas demonstrando pequenas melhorias em métricas específicas, mas não de forma generalizada.

Capítulo 4

Considerações Finais

Neste trabalho foi investigado se a aplicação de técnicas de normalização influencia significativamente o desempenho de modelos de classificação e se esse impacto varia conforme o tipo de modelo utilizado, incluindo modelos monolíticos e combinados.

Nossos resultados mostraram que a normalização pode, de fato, melhora o desempenho dos classificadores, mas essa melhoria não é garantida. Em alguns casos, os modelos tiveram um desempenho superior sem a normalização, enquanto, em outros, a normalização tornou o modelo mais robusto, permitindo uma previsão mais precisa. Isso sugere que a efetividade da normalização depende fortemente das características dos dados analisados.

No conjunto de dados sobre diabetes, por exemplo, observamos que o uso do LDA proporcionou uma leve melhora no desempenho do classificador. No entanto, ao aplicar o QDA, não houve um ganho significativo na capacidade de prever se as mulheres indígenas tinham ou não diabetes. Já no conjunto de dados sobre câncer de mama, a análise exploratória revelou que a variável resposta – que indica se as células são benignas ou malignas – já apresentava uma separação clara, o que permitiu que os modelos atingissem bons resultados mesmo sem a normalização. Por outro lado, no conjunto de dados sobre acupuntura, o desempenho dos classificadores variou consideravelmente dependendo da técnica de normalização empregada, evidenciando que a escolha da técnica pode impactar diretamente os resultados.

Este estudo seguiu a mesma linha do artigo do Amorim *et al.* (2022), utilizado de referência cujos outros analisaram o impacto da normalização em 82 conjuntos de dados. Assim como no artigo, observamos que a normalização pode trazer melhorias, embora nem sempre de forma significativa. Além disso, constatamos que os conjuntos de dados apresentam características bastante distintas, o que reforça a necessidade de avaliar

essa influência em diferentes contextos. Também, nos dados de acupuntura mantivemos as variáveis categóricas, enquanto nos demais utilizamos apenas variáveis quantitativas, permitindo uma análise mais ampla dos efeitos da normalização.

Outro ponto importante foi o alto custo computacional envolvido. A necessidade de testar diversos classificadores e técnicas de normalização demandou um tempo significativo de processamento, o que deve ser considerado em estudos futuros.

Assim, concluímos que não existe uma solução universal para a escolha de modelos e técnicas de normalização. Um classificador que apresenta ótimo desempenho em um conjunto de dados com uma técnica específica pode não obter os mesmos resultados em outro contexto. Isso destaca a importância de avaliar cuidadosamente as características dos dados ao definir a melhor estratégia para cada caso.

Referências Bibliográficas

Achsan, B. M. (2020). *Support Vector Machine: Regression*.

Amorim, L. B., Cavalcanti, G. D. e Cruz, R. M. (2022). The choice of scaling technique matters for classification performance. *Amsterdam: Elsevier*.

Breiman, L. (2001). Random forests. *Machine Learning*, **45**, 5–32.

Dalwinder, S. e Birmohan, S. (2020). Investigating the impact of data normalization on classification performance. *Amsterdam: Elsevier*.

Dancker, J. (2022). A brief introduction to feature scaling. *Hamburg: Medium*.

Fernandes, A. A. T., Figueiredo Filho, D. B., Rocha, E. C. d. e Nascimento, W. d. S. (2020). Leia este artigo se você quiser aprender regressão logística. *Revista de Sociologia e Política*, **28**(74), 1–20.

Izbicki, R. e Santos, T. M. (2017). *Aprendizado de máquina: Uma abordagem estatística*. UICLAP.

James, G., Witten, D., Hastie, T. e Tibshirani, R. (2013). *An Introduction to Statistical Learning*, volume 112. Springer.

Johnson, R. A. e Wichern, D. W. (2002). *Applied multivariate statistical analysis*. Prentice Hall Upper Saddle River, New Jersey.

Johnson, R. A. e Wichern, D. W. (2007). *Applied Multivariate Statistical Analysis*. Pearson Prentice Hall, Upper Saddle River, NJ, 6th edition. ISBN 978-0131877153.

Santos, C. d. J. (2017). *Avaliação do uso de classificadores para verificação de atendimento a critérios de seleção em programas sociais*. Tese de doutorado, Universidade Federal de Juiz de Fora.

Sharma, S. (2022). What is quadratic discriminant analysis? assumptions of quadratic discriminant analysis, advantages, and disadvantages. *Medium*. Available online: https://medium.com.

Smola, A. J., Bartlett, P. L., Scholkopf, B. e Schuurmans, D. (2000). *Advances in Large Margin Classifiers*. The MIT Press, London.

Vickers, A. J., Rees, R. W., Zollman, C. E., McCarney, R., Smith, C. M., Ellis, N., Fisher, P. e Van Haselen, R. (2004). Acupuncture for chronic headache in primary care: large, pragmatic, randomized trial. *BMJ*, **328**(7442), 744.

Apêndice

A - Análise Descritiva

```
###Ler base
Diabetes <- read_excel("C:/Users/emily/Downloads/Diabetes.xlsx")

###Lendo as variáveis da base

gra <- Diabetes$Gravidezes
gli <- Diabetes$Glicose2h
pre <- Diabetes$PresDiast
dobra <- Diabetes$DobraTriceps
insu <- Diabetes$Insulina
imc <- Diabetes$IMC
DPF <- Diabetes$DPF
idade <- Diabetes$Idade



###Cálculo da técnica de escalonamento robusto

Q <- quantile(gra,c(0.25,0.50,0.75))
var.n <- (gra-Q[2])/(Q[3]-Q[1])

plot(density(var.n))
hist(var.n)

###Histograma da variável Gravidezes

par(mfrow=c(2,2))

hist_gravidezes <- hist(Diabetes$Gravidezes,
     main = "Dados não normalizados",
```

```
     xlab = "Quantidade de gravidezes", ylab = "Quantidade de pessoas",
     col = c("blue"),
     border = FALSE,
     xlim = c(0,14), ylim = c(0,300)
     )

###Histograma da variável Gravidezes normalizada pelo escalonamento robusto

hist_gravidezes_robusto <- hist(var.n,
     main = "Normalização robusta",
     xlab = "Quantidade de gravidezes", ylab = "Quantidade de pessoas",
     col = c("blue"),
     border = FALSE,
     xlim = c(-1,2.5), ylim = c(0,300)
     )

###Histograma da variável Gravidezes normalizada pela min-max

f_minmax <- function(gra){
  return((gra - min(gra))/(max(gra)-min(gra)))
}

minmax <- f_minmax(gra)
hist(minmax,
    main = "Normalização min-max",
    xlab = "Quantidade de gravidezes", ylab = "Quantidade de pessoas",
    col = c("blue"),
    border = FALSE,
    xlim = c(0,1), ylim = c(0,300)
)

####Variável Glicose
Q <- quantile( gli, c(0.25,0.50,0.75))
Q
var.n_2 <- ( gli - Q[2])/(Q[3]-Q[1])
hist(var.n_2)

###Histograma da variável Glicose

par(mfrow=c(2,2))
```

```
hist_glicose <- hist(gli,
    main = "Dados não normalizados",
    xlab = "Concentração de glicose a 2h em um teste oral (mg/dL)",
    ylab = "Quantidade de pessoas",
    col = c("blue"),
    border = FALSE,
    xlim = c(0,200), ylim = c(0,300)
)


###Histograma da variável Glicose normalizada pelo escalonamento robusto

hist_glicose_robusto <- hist(var.n_2,
    main = "Normalização robusta",
    xlab = "Concentração de glicose a 2h em um teste oral {mg/dL)",
    ylab = "Quantidade de pessoas",
    col = c("blue"),
    border = FALSE,
    xlim = c(-1.5,2), ylim = c(0,300)
)

###Histograma da variável Glicose normalizada pelo desvio padrão

glicose.n <- (gli-mean(gli))/sd(gli)
hist(glicose.n,
     main = "Normalização pelo desvio padrão",
     xlab = "Concentração de glicose a 2h em um teste oral (mg/dL)",
     ylab = "Quantidade de pessoas",
     col = c("blue"),
     border = FALSE,
     xlim = c(-4,3), ylim = c(0,300)
)

###Histograma da variável Glicose normalizada pela min-max

f_minmax <- function(gli){
  return((gli - min(gli))/(max(gli)-min(gli)))
}

minmax <- f_minmax(gli)
```

```
hist(minmax,
     main = "Normalização min-max",
     xlab = "Concentração de glicose a 2h em um teste oral (mg/dL)",
     ylab = "Quantidade de pessoas",
     col = c("blue"),
     border = FALSE,
     xlim = c(0,1), ylim = c(0,300)
)

###Variável Pressão

pres <- pre[which(pre>0)]
Q <- quantile( pre, c(0.25,0.50,0.75))
var.n_3 <- ( pre - Q[2])/(Q[3]-Q[1])
hist(var.n_3)

###Histograma da variável Pressão

par(mfrow=c(2,2))
hist_pressao <- hist(pre,
    main = "Dados não normalizados",
    xlab = "Pressão arterial diastólica - mm Hg",
    ylab = "Quantidade de pessoas",
    col = c("blue"),
    border = FALSE,
    xlim = c(20,120), ylim = c(0,300)
)

###Histograma da variável Pressão normalizada pelo escalonamento robusto
hist_pressao_robusto <- hist(var.n_3,
    main = "Normalização robusta",
    xlab = "Pressão arterial diastólica - mm Hg",
    ylab = "Quantidade de pessoas",
    col = c("blue"),
    border = FALSE,
    xlim = c(-2,3), ylim = c(0,300)
)

###Histograma da variável Pressão normalizada pelo desvio padrão
```

```
glicose.n <- (pre-mean(pre))/sd(pre)
hist(glicose.n,
     main = "Normalização pelo desvio padrão",
     xlab = "Pressão arterial diastólica - mm Hg",
     ylab = "Quantidade de pessoas",
     col = c("blue"),
     border = FALSE,
     xlim = c(-4,4), ylim = c(0,300)
)

###Histograma da variável Pressão normalizada pela min-maax

f_minmax <- function(gli){
  return((gli - min(gli))/(max(gli)-min(gli)))
}

minmax <- f_minmax(gli)
hist(minmax,
     main = "Normalização min-max",
     xlab = "Pressão arterial diastólica - mm Hg",
     ylab = "Quantidade de pessoas",
     col = c("blue"),
     border = FALSE,
     xlim = c(0,1), ylim = c(0,300)
)

###Variável Dobra

dobra_2 <- dobra[which(dobra>0)]
Q <- quantile( dobra_2, c(0.25,0.50,0.75))
var.n_4 <- ( dobra_2 - Q[2])/(Q[3]-Q[1])

hist(var.n_4)

###Histograma da variável Pressão

par(mfrow=c(2,2))
hist_pressao <- hist(dobra_2,
    main = "Dados não normalizados",
    xlab = "Espessura da dobra da pele do tríceps - mm",
```

```
    ylab = "Quantidade de pessoas",
    col = c("blue"),
    border = FALSE,
    xlim = c(0,60), ylim = c(0,300)
)

###Histograma da variável Pressão normalizada pelo escalonamento robusto

hist_pressao_robusto <- hist(var.n_4,
    main = "Normalização robusta",
    xlab = "Espessura da dobra da pele do tríceps - mm",
    ylab = "Quantidade de pessoas",
    col = c("blue"),
    border = FALSE,
    xlim = c(-2,2), ylim = c(0,300)
)

###Histograma da variável Pressão normalizada pelo desvio padrão

dobra.n <- (dobra-mean(dobra))/sd(dobra)
hist(dobra.n,
     main = "Normalização pelo desvio padrão",
     xlab = "Espessura da dobra da pele do tríceps - mm",
     ylab = "Quantidade de pessoas",
     col = c("blue"),
     border = FALSE,
     xlim = c(-2,3), ylim = c(0,300)
)

###Histograma da variável Pressão normalizada pela min-max

f_minmax <- function(dobra){
  return((dobra - min(dobra))/(max(dobra)-min(dobra)))
}

minmax <- f_minmax(dobra)
hist(minmax,
     main = "Normalização pela min-max",
     xlab = "Espessura da dobra da pele do tríceps - mm",
     ylab = "Quantidade de pessoas",
```

```
     col = c("blue"),
     border = FALSE,
     xlim = c(0,0.6), ylim = c(0,300)
)

###variável Insulina

insulina_2 <- insu[which(insu>0)]
Q <- quantile( insulina_2, c(0.25,0.50,0.75))
var.n_5 <- ( insulina_2 - Q[2])/(Q[3]-Q[1])
plot(density(insulina_2))
plot(density(insulina_2, kernel = "gaussian",adjust = 1.2))

hist(var.n_5)

###Histograma da variável Insulina

par(mfrow=c(2,2))
hist_pressao <- hist(insulina_2,
    main = "Dados não normalizados",
    xlab = "Insulina sérica de 2 horas - micro UI/ml",
    ylab = "Quantidade de pessoas",
    col = c("blue"),
    border = FALSE,
    xlim = c(0,600), ylim = c(0,300)
)

###Histograma da variável Insulina normalizada pelo escalonamento robusto

hist_pressao_robusto <- hist(var.n_5,
    main ="Normalização robusta",
    xlab = "Insulina sérica de 2 horas - micro UI/ml",
    ylab = "Quantidade de pessoas",
    col = c("blue"),
    border = FALSE,
    xlim = c(-1,4), ylim = c(0,300)
)

###Histograma da variável Insulina normalizada pelo desvio padrão
```

```
insu.n <- (insu-mean(insu))/sd(insu)
hist(insu.n,
     main = "Normalização pelo desvio padrão",
     xlab = "Insulina sérica de 2 horas - micro UI/ml",
     ylab = "Quantidade de pessoas",
     col = c("blue"),
     border = FALSE,
     xlim = c(-1,4), ylim = c(0,300)
)

###Histograma da variável Insulina normalizada pela min-max

f_minmax <- function(insu){
  return((insu - min(insu))/(max(insu)-min(insu)))
}

minmax <- f_minmax(insu)
hist(minmax,
     main = "Normalização min-max",
     xlab = "Insulina sérica de 2 horas - micro UI/ml",
     ylab = "Quantidade de pessoas",
     col = c("blue"),
     border = FALSE,
     xlim = c(0,0.7), ylim = c(0,300)
)

###Variável Idade

Q <- quantile( idade, c(0.25,0.50,0.75))
var.n_6 <- ( idade - Q[2])/(Q[3]-Q[1])
plot(density(idade))
hist(var.n_6)

###Histograma da variável Idade

par(mfrow=c(2,2))
hist_idade <- hist(idade,
    main = "Dados não normalizados",
    xlab = "Idade (Anos)",
    ylab = "Quantidade de pessoas",
```

```
    col = c("blue"),
    border = FALSE,
    xlim = c(20,70), ylim = c(0,300)

)

###Histograma da variável Idade normalizada pelo escalonamento robusto

hist_idade_robusto <- hist(var.n_6,
    main ="Normalização robusta",
    xlab = "Idade (Anos)",
    ylab = "Quantidade de pessoas",
    col = c("blue"),
    border = FALSE,
    xlim = c(-0.5,2.5), ylim = c(0,300)
)

###Histograma da variável Idade normalizada pelo desvio padrão

idade.n <- (idade-mean(idade))/sd(idade)
hist(idade.n,
     main = "Normalização pelo desvio padrão",
     xlab = "Idade (Anos)",
     ylab = "Quantidade de pessoas",
     col = c("blue"),
     border = FALSE,
     xlim = c(-2,4), ylim = c(0,300)
)

###Histograma da variável Idade normalizada pela min-max

f_minmax <- function(idade){
  return((idade - min(idade))/(max(idade)-min(idade)))
}

minmax <- f_minmax(idade)
hist(minmax,
     main = "Normalização min-max",
     xlab = "Idade (Anos)",
     ylab = "Quantidade de pessoas",
```

```
     col = c("blue"),
     border = FALSE,
     xlim = c(0,0.8), ylim = c(0,300)
)

###Variável IMC

imc_2 <- imc[which(imc>0)]
imc <- as.numeric(imc_2)
Q <- quantile( imc, c(0.25,0.50,0.75))
Q
max(imc)
var.n_6 <- ( imc - Q[2])/(Q[3]-Q[1])

hist(var.n_6)
hist(imc)

###Histograma da variável IMC

par(mfrow=c(2,2))
hist_pressao <- hist(imc,
    main = "Dados não normalizados",
    xlab = "Índice de massa corpóreo - kg/m^2",
    ylab = "Quantidade de pessoas",
    col = c("blue"),
    border = FALSE,
    xlim = c(10,60), ylim = c(0,300)
)

###Histograma da variável IMC normalizada pelo escalonamento robusto

hist_pressao_robusto <- hist(var.n_6,
    main ="Normalização robusta",
    xlab = "Índice de massa corpóreo - kg/m^2",
    ylab = "Quantidade de pessoas",
    col = c("blue"),
    border = FALSE,
    xlim = c(-2,3), ylim = c(0,300)
)
```

```
###Histograma da variável IMC normalizada pelo desvio padrão

imc.n <- (imc-mean(imc))/sd(imc)
hist(imc.n,
     main = "Normalização pelo desvio padrão",
     xlab = "Índice de massa corpóreo - kg/m^2",
     ylab = "Quantidade de pessoas",
     col = c("blue"),
     border = FALSE,
     xlim = c(-2,3), ylim = c(0,300)
)

###Histograma da variável IMC normalizada pela min-max

f_minmax <- function(imc){
  return((imc - min(imc))/(max(imc)-min(imc)))
}
minmax <- f_minmax(imc)
hist(minmax,
     main = "Normalização min-max",
     xlab = "Índice de massa corpóreo - kg/m^2",
     ylab = "Quantidade de pessoas",
     col = c("blue"),
     border = FALSE,
     freq = TRUE,
     xlim = c(0,0.8), ylim = c(0,300)
)

###Variável DPF

dpf <- Diabetes$DPF
dpf_2 <- dpf[which(dpf>0)]
dpf_numeric <- as.numeric(dpf_2)
Q <- quantile( dpf_numeric, c(0.25,0.50,0.75))
max(dpf_numeric)
var.n_7 <- ( dpf_numeric - Q[2])/(Q[3]-Q[1])

hist(var.n_7)

###Histograma da variável DPF
```

```
par(mfrow=c(2,2))
hist_pressao <- hist(dpf_numeric,
    main = "Dados não normalizados",
    xlab = "Função Pedigree de Diabete",
    ylab = "Quantidade de pessoas",
    col = c("blue"),
    border = FALSE,
    xlim = c(0,2), ylim = c(0,300)
)

###Histograma da variável DPF normalizada pelo escalonamento robusto

hist_pressao_robusto <- hist(var.n_7,
    main ="Normalização robusta",
    xlab = "Função Pedigree de Diabetes",
    ylab = "Quantidade de pessoas",
    col = c("blue"),
    border = FALSE,
    xlim = c(-1,4), ylim = c(0,300)
)

###Histograma da variável DPF normalizada pelo desvio padrão

dpf_numeric.n <- (dpf_numeric-mean(dpf_numeric))/sd(dpf_numeric)
hist(dpf_numeric.n,
     main = "Normalização pelo desvio padrão",
     xlab = "Função Pedigree de Diabetes",
     ylab = "Quantidade de pessoas",
     col = c("blue"),
     border = FALSE,
     xlim = c(-2.5,3), ylim = c(0,300)
)

###Histograma da variável DPF normalizada pela min-max

f_minmax <- function(dpf_numeric){
  return((dpf_numeric - min(dpf_numeric))/(max(dpf_numeric)-min(dpf_numeric)))
}
minmax <- f_minmax(dpf_numeric)
```

```
hist(minmax,
     main = "Normalização min-max",
     xlab = "Função Pedigree de Diabetes", ylab = "Quantidade de pessoas",
     col = c("blue"),
     border = FALSE,
     xlim = c(0,0.8), ylim = c(0,300)
)

###Dados Cancêr de Mama
# Padronização dos dados (exceto a variável diagnosis)
dados_scaled <- dados %>%
  mutate(across(-diagnosis, scale))

# Verificar normalidade das variáveis principais através de histogramas
#ggplot(melt(dados[,c("radius_mean", "texture_mean", "perimeter_mean", "area_mean")]),
#       aes(x = value)) +
#  geom_histogram(bins = 30, fill = "steelblue", color = "black") +
#  facet_wrap(~variable, scales = "free") +
#  theme_minimal() +
#  labs(x = "Valores", y = "Frequência")

ggplot(melt(dados[,c("radius_mean", "texture_mean", "perimeter_mean", "area_mean")]),
       aes(x = value)) +
  geom_histogram(bins = 30, fill = "steelblue", color = "black") +
  facet_wrap(~variable, scales = "free", labeller = as_labeller(c(
    "radius_mean" = "Radius",
    "texture_mean" = "Texture",
    "perimeter_mean" = "Perimeter",
    "area_mean" = "Area"
  ))) +
  theme_minimal() +
  labs(x = "Valores", y = "Frequência")


# Proporção de casos benignos e malignos
prop.table(table(dados$diagnosis)) * 100

# Estatísticas descritivas por grupo (Benigno e Maligno)
describeBy(dados[,-1], group = dados$diagnosis)
```

```
# Correlação entre variáveis
cor_matrix <- cor(dados[,-1])
corrplot(cor_matrix, method = "color", type = "upper",
         tl.col = "black", tl.srt = 45, tl.cex = 0.7)

# PCA para visualização
pca_result <- prcomp(dados[,-1], scale. = TRUE)
pca_data <- data.frame(
  PC1 = pca_result$x[,1],
  PC2 = pca_result$x[,2],
  diagnosis = dados$diagnosis
)

# Plot do PCA
ggplot(pca_data, aes(x = PC1, y = PC2, color = diagnosis)) +
  geom_point(alpha = 0.6) +
  theme_minimal() +
  labs(title = "PCA - Visualização dos Grupos", x = "PC1", y = "PC2")

###Dados Acupuntura

matriz_cor <- cor(base_final_v3 %>% select_if(is.numeric))

cor_plot <- corrplot(matriz_cor, method = "color", type = "upper",
                     tl.col = "black", tl.srt = 45,
                     addCoef.col = "black")
```

Apêndice A

B - Modelagem

```
# Procesamento ------------------------------------------------------------

library(dplyr)
library(readxl)

mode_func <- function(x) {
  ux <- unique(x)
  ux[which.max(tabulate(match(x, ux)))]
}

# Função para normalização
normalize <- function(x, method = "robust") {
  if (method == "robust") {
    Q <- quantile(x, c(0.25, 0.50, 0.75), na.rm = TRUE)
    return((x - Q[2]) / (Q[3] - Q[1]))
  } else if (method == "standard") {
    return((x - mean(x, na.rm = TRUE)) / sd(x, na.rm = TRUE))
  } else if (method == "minmax") {
    return((x - min(x, na.rm = TRUE)) /
    (max(x, na.rm = TRUE) - min(x, na.rm = TRUE)))
  } else {
    stop("Método de normalização não reconhecido.")
  }
}

Diabetes <- read_excel("C:/Users/emily/Downloads/Diabetes.xlsx")

df <- Diabetes
```

```
df

# Tratamento de valores ausentes/nulos

df <- df %>%
  mutate(
    IMC = as.numeric(IMC),  # Transformando "IMC" para numérica
    PressaoArterial = if_else(is.na(PressaoArterial),
    mode_func(PressaoArterial), PressaoArterial),
    DPF = if_else(is.na(DPF), mean(DPF, na.rm = TRUE), DPF)
  ) %>%
  filter(
    PresDiast != 0,
    DobraTriceps != 0,
    Insulina != 0,
    IMC != 0
  )

df <- dplyr::select(df, -ordem)

# Normalizar os dados e criar diferentes bases
df_base_1 <- df
df_base_2 <- df %>% mutate(across(where(is.numeric), normalize,
method = "robust"))
df_base_3 <- df %>% mutate(across(where(is.numeric), normalize,
method = "standard"))
df_base_4 <- df %>% mutate(across(where(is.numeric), normalize,
method = "minmax"))

# Modelar ----------------------------------------------------------------

library(MASS)
library(caret)
library(tibble)

# Função para aplicar LDA e retornar a acurácia e a matriz de confusão
apply_lda <- function(df, target_column, positive_class = "pos") {
  X <- df %>% dplyr::select(where(is.numeric))
  X <- X[, !(names(X) %in% target_column)]
  y <- df[[target_column]]
```

```
  # set.seed(42)
  train_index <- createDataPartition(y, p = .3, list = FALSE)
  X_train <- X[train_index, ]
  X_test <- X[-train_index, ]
  y_train <- y[train_index]
  y_test <- y[-train_index]

  lda_model <- lda(X_train, grouping = y_train)
  y_pred <- predict(lda_model, X_test)$class

  accuracy <- mean(y_pred == y_test)

  y_test <- factor(y_test, levels = c("pos", "neg"))
  y_pred <- factor(y_pred, levels = c("pos", "neg"))

  confusion_mat <- confusionMatrix(as.factor(y_pred), as.factor(y_test),
  positive = positive_class)

  list(accuracy = accuracy, confusion_mat = confusion_mat)
}


# Função para aplicar QDA e retornar a acurácia e a matriz de confusão
apply_qda <- function(df, target_column, positive_class = "pos") {
  X <- df %>% dplyr::select(where(is.numeric))
  X <- X[, !(names(X) %in% target_column)]
  y <- df[[target_column]]

  # set.seed(42)
  train_index <- createDataPartition(y, p = .3, list = FALSE)
  X_train <- X[train_index, ]
  X_test <- X[-train_index, ]
  y_train <- y[train_index]
  y_test <- y[-train_index]

  qda_model <- qda(X_train, grouping = y_train)
  y_pred <- predict(qda_model, X_test)$class

  accuracy <- mean(y_pred == y_test)
```

```
  confusion_mat <- confusionMatrix(as.factor(y_pred), as.factor(y_test),
  positive = positive_class)

  return(list(accuracy = accuracy, confusion_mat = confusion_mat))
}

# Aplicar LDA e QDA nas bases de dados
lda_results <- tibble(
  Base = c("Sem normalizar", "Escalonamento Robusto", "Desvio Padrão",
  "Min-Max"),
  Acurácia = c(
    apply_lda(df_base_1, 'TesteDiabete')$accuracy,
    apply_lda(df_base_2, 'TesteDiabete')$accuracy,
    apply_lda(df_base_3, 'TesteDiabete')$accuracy,
    apply_lda(df_base_4, 'TesteDiabete')$accuracy
  )
)

qda_results <- tibble(
  Base = c("Sem normalizar", "Escalonamento Robusto", "Desvio Padrão",
  "Min-Max"),
  Acurácia = c(
    apply_qda(df_base_1, 'TesteDiabete')$accuracy,
    apply_qda(df_base_2, 'TesteDiabete')$accuracy,
    apply_qda(df_base_3, 'TesteDiabete')$accuracy,
    apply_qda(df_base_4, 'TesteDiabete')$accuracy
  )
)

# Comparar resultados LDA e QDA
comparison_results <- tibble(
  Base = c("Sem normalizar", "Escalonamento Robusto", "Desvio Padrão",
  "Min-Max"),
  LDA_Acuracia = lda_results$Acurácia,
  QDA_Acuracia = qda_results$Acurácia
)

# visualização ------------------------------------------------------------

library(ggplot2)
```

```
library(reshape2)
library(gridExtra)

# Função para plotar distribuições escalonadas
plot_scaled_distributions <- function(df_list, titles) {
  num_vars <- names(dplyr::select(df_list[[1]], where(is.numeric)))
  plots <- list()

  for (var in num_vars) {
    for (i in seq_along(df_list)) {
      p <- ggplot(df_list[[i]], aes(x = log1p(!!sym(var)))) +
        geom_histogram(aes(y = ..density..), bins = 25, fill = i, alpha = 0.5) +
        geom_density(color = i) +
        ggtitle(paste(titles[i], ":", var)) +
        theme_minimal(base_size = 12) +
        theme(
          plot.title = element_text(size = 8, face = "bold", hjust = 0.5),
          axis.title = element_text(size = 6, face = "bold"),
          axis.text = element_text(size = 6),
          plot.background = element_rect(fill = "white", color = NA),
          panel.background = element_rect(fill = "white", color = NA),
          panel.grid = element_blank(),
          axis.line = element_line(color = "black")
        )

      plots <- append(plots, list(p))
    }
  }

  grid.arrange(grobs = plots, ncol = 4)
}

# Plotar matrizes de confusão para LDA
# Definir os títulos das diferentes bases
titles <- c('Sem normalizar', 'Escalonamento Robusto', 'Desvio Padrão',
'Min-Max')

# Calcular as matrizes de confusão utilizando a função modificada apply_lda
lda_matrices <- list(
  apply_lda(df_base_1, 'TesteDiabete', positive_class = "pos")$confusion_mat,
```

```
  apply_lda(df_base_2, 'TesteDiabete', positive_class = "pos")$confusion_mat,
  apply_lda(df_base_3, 'TesteDiabete', positive_class = "pos")$confusion_mat,
  apply_lda(df_base_4, 'TesteDiabete', positive_class = "pos")$confusion_mat
)

# Criar uma lista de gráficos das matrizes de confusão
plot_list <- list()
for (i in seq_along(lda_matrices)) {
  plot_list[[i]] <- plot_confusion_matrix(lda_matrices[[i]], paste("LDA -",
  titles[i]))
}

# Organizar os gráficos em um grid de 2 colunas
grid.arrange(grobs = plot_list, ncol = 2)

# Plotar matrizes de confusão para QDA
qda_matrices <- list(
  apply_qda(df_base_1, 'TesteDiabete', positive_class = "pos")$confusion_mat,
  apply_qda(df_base_2, 'TesteDiabete', positive_class = "pos")$confusion_mat,
  apply_qda(df_base_3, 'TesteDiabete', positive_class = "pos")$confusion_mat,
  apply_qda(df_base_4, 'TesteDiabete', positive_class = "pos")$confusion_mat
)

qda_plot_list <- list()
for (i in seq_along(qda_matrices)) {
  qda_plot_list[[i]] <- plot_confusion_matrix(qda_matrices[[i]],
  paste("QDA -", titles[i]))
}
grid.arrange(grobs = qda_plot_list, ncol = 2)

# Comparar resultados LDA e QDA
ggplot(melt(comparison_results, id.vars = "Base"), aes(x = Base, y = value,
fill = variable)) + geom_bar(stat = "identity", position = position_dodge(),
color = "black") + geom_text(aes(label = sprintf("%.3f", value)),
position = position_dodge(width = 0.9), vjust = -0.5, size = 4) +
  labs(title = "Comparação de Acurácia: LDA vs QDA",
       x = "Base de Dados", y = "Acurácia", fill = "Modelo") +
  theme_minimal(base_size = 16, base_family = "Gadugi") +
  theme(
    plot.background = element_rect(fill = "white", color = NA),
```

```
    panel.background = element_rect(fill = "white", color = NA),
    panel.grid = element_blank(),
    axis.line = element_line(color = "gray20"),
    plot.title = element_text(hjust = 0.5, face = "bold", color = "gray20"),
    plot.subtitle = element_text(hjust = 0.5, face = "italic", color = "gray40"),
    plot.caption = element_text(hjust = 0.5, face = "italic", color = "gray50"),
    axis.text = element_text(color = "gray30"),
    axis.title = element_text(face = "bold", color = "gray20"),
    panel.border = element_blank()
  )

calculate_performance_metrics <- function(confusion_matrix) {
  TP <- confusion_matrix$table[1, 1]  # Verdadeiro Positivo (pos/pos)
  FP <- confusion_matrix$table[1, 2]  # Falso Positivo (pos/neg)
  FN <- confusion_matrix$table[2, 1]  # Falso Negativo (neg/pos)
  TN <- confusion_matrix$table[2, 2]  # Verdadeiro Negativo (neg/neg)

  # Especificidade (Specificity)
  specificity <- TN / (TN + FP)

  # Sensibilidade (Sensitivity) ou Recall
  sensitivity <- TP / (TP + FN)

  # Precisão (Precision)
  precision <- TP / (TP + FP)

  # F1-Score
  f1_score <- 2 * ((precision * sensitivity) / (precision + sensitivity))

  # Taxa de Erro Aparente (Apparent Error Rate)
  error_rate <- (FP + FN) / (TP + TN + FP + FN)

  list(
    Specificity = specificity,
    Sensitivity = sensitivity,
    Precision = precision,
    F1_Score = f1_score,
    Error_Rate = error_rate
  )
}
```

```
# Calcular as métricas de desempenho para cada técnica de normalização com LDA
lda_performance_metrics <- lapply(lda_matrices, calculate_performance_metrics)

# Calcular as métricas de desempenho para cada técnica de normalização com QDA
qda_performance_metrics <- lapply(qda_matrices, calculate_performance_metrics)
library(tibble)

# Consolidar resultados para LDA
lda_metrics_table <- tibble(
  Base = titles,
  Specificity = sapply(lda_performance_metrics, function(x) x$Specificity),
  Sensitivity = sapply(lda_performance_metrics, function(x) x$Sensitivity),
  Precision = sapply(lda_performance_metrics, function(x) x$Precision),
  F1_Score = sapply(lda_performance_metrics, function(x) x$F1_Score),
  Error_Rate = sapply(lda_performance_metrics, function(x) x$Error_Rate)
)

# Consolidar resultados para QDA
qda_metrics_table <- tibble(
  Base = titles,
  Specificity = sapply(qda_performance_metrics, function(x) x$Specificity),
  Sensitivity = sapply(qda_performance_metrics, function(x) x$Sensitivity),
  Precision = sapply(qda_performance_metrics, function(x) x$Precision),
  F1_Score = sapply(qda_performance_metrics, function(x) x$F1_Score),
  Error_Rate = sapply(qda_performance_metrics, function(x) x$Error_Rate)
)
ggplot(melt(lda_metrics_table, id.vars = "Base"), aes(x = Base, y = value,
fill = variable)) +
  geom_bar(stat = "identity", position = position_dodge(), color = "black") +
  geom_text(aes(label = sprintf("%.3f", value)), position = position_dodge
  (width = 0.9), vjust = -0.5, size = 4) +
  labs(title = "Comparação de Métricas de Desempenho - LDA",
       x = "Base de Dados", y = "Valor", fill = "Métrica") +
  theme_minimal(base_size = 16, base_family = "Gadugi") +
  theme(
    plot.background = element_rect(fill = "white", color = NA),
    panel.background = element_rect(fill = "white", color = NA),
    panel.grid = element_blank(),
    axis.line = element_line(color = "gray20"),
```

```
    plot.title = element_text(hjust = 0.5, face = "bold", color = "gray20"),
    axis.text = element_text(color = "gray30"),
    axis.title = element_text(face = "bold", color = "gray20")
  )
ggplot(melt(qda_metrics_table, id.vars = "Base"), aes(x = Base, y = value,
fill = variable)) +
  geom_bar(stat = "identity", position = position_dodge(), color = "black") +
  geom_text(aes(label = sprintf("%.3f", value)), position = position_dodge
  (width = 0.9), vjust = -0.5, size = 4) +
  labs(title = "Comparação de Métricas de Desempenho - LDA",
       x = "Base de Dados", y = "Valor", fill = "Métrica") +
  theme_minimal(base_size = 16, base_family = "Gadugi") +
  theme(
    plot.background = element_rect(fill = "white", color = NA),
    panel.background = element_rect(fill = "white", color = NA),
    panel.grid = element_blank(),
    axis.line = element_line(color = "gray20"),
    plot.title = element_text(hjust = 0.5, face = "bold", color = "gray20"),
    axis.text = element_text(color = "gray30"),
    axis.title = element_text(face = "bold", color = "gray20")
  )

create_metric_plot <- function(data, metric, metric_name, model_name,
y_limit = 1.1) {
  ggplot(data, aes(x = Base, y = data[[metric]], fill = Base)) +
    geom_bar(stat = "identity", color = "black", show.legend = FALSE) +
    geom_text(aes(label = sprintf("%.3f", data[[metric]])), vjust = -0.5,
    size = 4, color = "black") +
    labs(title = paste(model_name, "-", metric_name), x = NULL, y = "Valor") +
    theme_minimal(base_size = 14) +
    theme(
      plot.title = element_text(hjust = 0.5, face = "bold", size = 16,
      color = "black"),
      axis.title.y = element_text(size = 14, color = "black", face = "bold",
      margin = margin(r = 10)),
      axis.text.x = element_text(size = 12, color = "black"),
      axis.text.y = element_text(size = 12, color = "black"),
      panel.grid.major = element_blank(),
      panel.grid.minor = element_blank(),
      axis.line = element_line(color = "black", size = 0.8),
```

```
      plot.margin = unit(c(1, 1, 1, 1), "cm")
    ) +
    scale_y_continuous(expand = expansion(mult = c(0, 0.05)),
    limits = c(0, y_limit)) + scale_fill_manual(values = c("gray",
    "lightblue", "white", "lightyellow"))
}

lda_sensitivity_plot <- create_metric_plot(lda_metrics_table, "Sensitivity",
"Sensibilidade", "LDA")
lda_precision_plot <- create_metric_plot(lda_metrics_table, "Precision",
"Precisão", "LDA")
lda_f1_plot <- create_metric_plot(lda_metrics_table, "F1_Score",
"F1-Score", "LDA")
lda_error_plot <- create_metric_plot(lda_metrics_table, "Error_Rate",
"Taxa de Erro", "LDA", y_limit = 0.4)

qda_sensitivity_plot <- create_metric_plot(qda_metrics_table, "Sensitivity",
"Sensibilidade", "QDA")
qda_precision_plot <- create_metric_plot(qda_metrics_table, "Precision",
"Precisão", "QDA")
qda_f1_plot <- create_metric_plot(qda_metrics_table, "F1_Score",
"F1-Score", "QDA")
qda_error_plot <- create_metric_plot(qda_metrics_table, "Error_Rate",
"Taxa de Erro", "QDA", y_limit = 0.4)

lda_grid <- grid.arrange(lda_sensitivity_plot, lda_precision_plot, lda_f1_plot,
lda_error_plot,
            nrow = 2, ncol = 2,
            top = "Desempenho LDA",
            padding = unit(1, "lines"))

qda_grid <- grid.arrange(qda_sensitivity_plot, qda_precision_plot, qda_f1_plot,
qda_error_plot,
            nrow = 2, ncol = 2,
            top = "Desempenho QDA",
            padding = unit(1, "lines"))

###Dados Cancêr de mama
# Padronizar as variáveis
dados_lda_scaled <- dados_lda_clean %>%
```

```
  mutate(across(-diagnosis, scale))

# Normalização Min-Max
dados_minmax <- dados_lda_clean %>%
  mutate(across(where(is.numeric), ~ scales::rescale(., to = c(0,1))))

# Normalização Quantile
numeric_cols <- dados_lda_clean %>% dplyr::select(where(is.numeric)) %>% colnames()
numeric_matrix <- as.matrix(dados_lda_clean %>% dplyr::select(all_of(numeric_cols)))
quantile_normalized_matrix <- normalize.quantiles(numeric_matrix)
dados_quantile <- dados_lda_clean
dados_quantile[ , numeric_cols] <- quantile_normalized_matrix
dados_standardized <- dados_lda_clean %>% mutate(across(where(is.numeric),
~ as.vector(scale(.))))

# Normalização Robusta
normalizar_robusto <- function(x) {
  mediana <- median(x, na.rm = TRUE)
  iqr <- IQR(x, na.rm = TRUE)
  (x - mediana) / iqr
}
dados_robust <- dados_lda_clean %>%
  mutate(across(where(is.numeric), ~ normalizar_robusto(.)))

  # Aplicar testes em todas as bases
resultados <- list(
  original = testar_pressupostos(dados_lda, "Original"),
  minmax = testar_pressupostos(dados_minmax, "Min-Max"),
  standardized = testar_pressupostos(dados_lda_scaled, "Padronizada"),
  quantile = testar_pressupostos(dados_quantile, "Quantile"),
  robust = testar_pressupostos(dados_robust, "Robusta")
)
# Criar lista com todas as bases de dados
bases <- list(
  original = dados_lda_clean,
  desvio_padrao = dados_lda_scaled,
  minmax = dados_minmax,
  quantílica = dados_quantile,
  robusto = dados_robust
)
```

```
# Função para calcular métricas de performance
calcular_metricas <- function(real, previsto) {
  cm <- confusionMatrix(factor(previsto), factor(real))
  return(c(
    Acuracia = cm$overall["Accuracy"],
    Sensibilidade = cm$byClass["Sensitivity"],
    Especificidade = cm$byClass["Specificity"],
    F1_Score = cm$byClass["F1"]
  ))
}

# Função para aplicar modelos e avaliar
avaliar_base <- function(base, train_index) {
  # Separar features e target
  X <- base %>% dplyr::select(-diagnosis)
  y <- base$diagnosis

  # Dividir em treino e teste usando o índice fornecido
  X_train <- X[train_index, ]
  X_test <- X[-train_index, ]
  y_train <- y[train_index]
  y_test <- y[-train_index]

  # Lista para armazenar resultados
  resultados <- list()

  # 1. LDA
  tryCatch({
    lda_model <- lda(diagnosis ~ ., data = base[train_index,])
    lda_pred <- predict(lda_model, X_test)$class
    resultados$lda <- calcular_metricas(y_test, lda_pred)
  }, error = function(e) {
    resultados$lda <- rep(NA, 4)
    print(paste("Erro no LDA:", e$message))
  })

  # 2. QDA
  tryCatch({
    qda_model <- qda(diagnosis ~ ., data = base[train_index,])
```

```
    qda_pred <- predict(qda_model, X_test)$class
    resultados$qda <- calcular_metricas(y_test, qda_pred)
  }, error = function(e) {
    resultados$qda <- rep(NA, 4)
    print(paste("Erro no QDA:", e$message))
  })

  # 3. Random Forest
  tryCatch({
    rf_model <- randomForest(diagnosis ~ ., data = base[train_index,],
                             ntree = 500, importance = TRUE)
    rf_pred <- predict(rf_model, X_test)
    resultados$rf <- calcular_metricas(y_test, rf_pred)
  }, error = function(e) {
    resultados$rf <- rep(NA, 4)
    print(paste("Erro no RF:", e$message))
  })

  # 4. SVM
  tryCatch({
    svm_model <- svm(diagnosis ~ ., data = base[train_index,],
                     kernel = "radial", probability = TRUE)
    svm_pred <- predict(svm_model, X_test)
    resultados$svm <- calcular_metricas(y_test, svm_pred)
  }, error = function(e) {
    resultados$svm <- rep(NA, 4)
    print(paste("Erro no SVM:", e$message))
  })

  return(resultados)
}
# Aplicar modelos em todas as bases
resultados_completos <- lapply(bases, function(base) {
  avaliar_base(base, train_index)
})

# 1. Gráfico de barras para acurácia
plot_acuracia <- ggplot(subset(resultados_finais, Metrica == "Acuracia"),
                        aes(x = reorder(Base, Valor), y = Valor, fill = Modelo)) +
  geom_bar(stat = "identity", position = position_dodge(0.9), width = 0.8) +
```

```
  scale_y_continuous(labels = percent,
                     limits = c(0, max(subset(resultados_finais,
                                              Metrica == "Acuracia")$Valor) * 1.1)) +
  scale_fill_viridis(discrete = TRUE, option = "D") +
  theme_minimal() +
  labs(title = "Comparação de Acurácia entre Modelos e Bases de Dados",
       subtitle = "Resultados por tipo de normalização e método de classificação",
       x = "Tipo de Normalização",
       y = "Acurácia (%)",
       fill = "Método de\nClassificação") +
  theme(
    plot.title = element_text(hjust = 0.5, size = 14, face = "bold"),
    plot.subtitle = element_text(hjust = 0.5, size = 12),
    axis.text.x = element_text(angle = 45, hjust = 1),
    legend.position = "right",
    legend.title = element_text(face = "bold"),
    panel.grid.major.y = element_line(color = "gray90"),
    panel.grid.minor = element_blank()
  )

  # 3. Gráfico de radar para comparação multimétrica
# Preparar dados para o gráfico de radar
radar_data <- resultados_finais %>%
  group_by(Modelo, Metrica) %>%
  summarise(Valor_Medio = mean(Valor, na.rm = TRUE), .groups = 'drop')

# Plot do radar
plot_radar <- ggplot(radar_data,
                     aes(x = Metrica, y = Valor_Medio,
                         color = Modelo, group = Modelo)) +
  geom_polygon(aes(fill = Modelo), alpha = 0.1) +
  geom_point(size = 2) +
  geom_line() +
  coord_polar() +
  scale_y_continuous(labels = percent) +
  scale_color_viridis(discrete = TRUE) +
  scale_fill_viridis(discrete = TRUE) +
  theme_minimal() +
  labs(title = "Comparação Multimétrica dos Modelos",
       subtitle = "Valores médios para todas as bases",
```

```
       y = "Valor Médio (%)",
       color = "Método de\nClassificação",
       fill = "Método de\nClassificação") +
  theme(
    plot.title = element_text(hjust = 0.5, size = 14, face = "bold"),
    plot.subtitle = element_text(hjust = 0.5, size = 12),
    axis.text.x = element_text(face = "bold"),
    legend.position = "right"
  )

  # Criar tabela resumo formatada
tabela_resumo <- resultados_finais %>%
  group_by(Modelo, Metrica) %>%
  summarise(
    Media = mean(Valor, na.rm = TRUE),
    DP = sd(Valor, na.rm = TRUE),
    Minimo = min(Valor, na.rm = TRUE),
    Maximo = max(Valor, na.rm = TRUE),
    .groups = 'drop'
  ) %>%
  mutate(across(where(is.numeric),
                ~sprintf("%.1f%%", . * 100)))

# Criar índices de treino e teste (70-30)
set.seed(123)  # Para reprodutibilidade
train_index <- createDataPartition(dados_lda_clean$diagnosis, p = 0.7, list = FALSE)

# Aplicar modelos em todas as bases
resultados_completos <- lapply(bases, function(base) {
  avaliar_base(base, train_index)
})

# Formatar os resultados para facilitar a visualização
resultados_finais <- formatar_resultados(resultados_completos)

###Dados Acupuntura
# Método: Imputação baseada em regressão
regression_imputation <- function(data) {
  df <- data
```

```
  # Cria modelo para pk2 usando variáveis baseline
  pk2_model <- lm(pk2 ~ pk1 + age + sex + migraine + chronicity + group,
                  data=df[!is.na(df$pk2),])

  # Prediz valores faltantes
  df$pk2[is.na(df$pk2)] <- predict(pk2_model, newdata=df[is.na(df$pk2),])
  df$pk5[is.na(df$pk5)] <- predict(pk5_model, newdata=df[is.na(df$pk5),])

  return(df)
}

library(ggplot2)
library(dplyr)
library(corrplot)
library(tidyr)
library(grid)
library(ggpubr)

tema_cientifico <- theme_minimal() +
  theme(
    # Elementos de texto
    plot.title = element_text(
      hjust = 0.5,
      size = 16,
      face = "bold",
      color = "#2F4F4F"
    ),
    plot.subtitle = element_text(
      hjust = 0.5,
      size = 12,
      margin = unit(c(0, 0, 20, 0), "pt"),  # Added unit parameter
      color = "#696969"
    ),
    axis.title = element_text(
      size = 12,
      color = "#404040"
    ),
    axis.text = element_text(
      size = 10,
      color = "#606060"
```

```
),
legend.title = element_text(
  size = 11,
  face = "bold",
  color = "#404040"
),
legend.text = element_text(
  size = 10,
  color = "#606060"
),

# Eixos e grades
panel.grid.major = element_line(
  color = "#F0F0F0",
  linewidth = 0.4
),
panel.grid.minor = element_blank(),
axis.line = element_line(
  color = "#D3D3D3",
  linewidth = 0.4
),
axis.ticks = element_line(
  color = "#D3D3D3",
  linewidth = 0.4
),

# Layout geral
plot.margin = unit(c(25, 25, 25, 25), "pt"),  # Corrected margin
plot.background = element_rect(fill = "white", color = NA),
panel.background = element_rect(fill = "white", color = NA),
legend.position = "bottom",
legend.margin = unit(c(10, 0, 0, 0), "pt"),  # Added unit parameter
legend.key.height = unit(0.4, "cm"),
legend.key.width = unit(1.5, "cm"),

# Facetas (se usar)
strip.text = element_text(
  color = "#404040",
  face = "bold",
  size = 11
```

```
    ),
    strip.background = element_rect(
      fill = "#F8F8F8",
      color = NA
    )
  )

#Distribuição das variáveis principais
p1 <- ggplot(data, aes(x = pk1)) +
  geom_histogram(fill = "#69b3a2", alpha = 0.7, bins = 30) +
  labs(title = "Distribuição da Severidade Inicial",
       x = "Severidade (pk1)", y = "Frequência") +
  tema_cientifico

# Correlação
print(corrplot(matriz_cor, method = "color", type = "upper",
               tl.col = "black", tl.srt = 45, addCoef.col = "black"))

# Base final sem normalização
base_final <- data_features

base_final_v3 <- base_final %>%
  mutate(
    # Binários (0/1)
    resposta_tratamento = as.factor(resposta_tratamento),
    group = as.factor(group),
    sex = as.factor(sex),
    migraine = as.factor(migraine),

    # Confirmando numéricos
    across(c(pk1, f1, f2, f5, age, chronicity), as.numeric)
  ) %>% dplyr::select(-f5, -pk2, -f2, -practice_id)

# Dividir em treino (70%) e teste (30%)
indice_treino <- createDataPartition(base_final_v3$resposta_tratamento, p = 0.7, list = FALSE)
train_data <- base_final_v3[indice_treino, ]
test_data  <- base_final_v3[-indice_treino, ]

variaveis_numericas <- c(
  "pk1", "f1",
```

```
  "age", "chronicity"
)

contents(train_data)

#Sem normalizar
rec_none <- recipe(resposta_tratamento ~ ., data = train_data) %>%
  step_rm(id,acupuncturist,practice_id) %>%
  # Se tiver colunas irrelevantes (ex.: id), remova:
  # step_rm(id, ...)
  # Poderia incluir step_rm(acupuncturist, practice_id) caso existam
  update_role(everything(), new_role = "predictor") %>%
  update_role(resposta_tratamento, new_role = "outcome")

prep_none <- prep(rec_none, training = train_data)
train_none <- bake(prep_none, new_data = NULL)
test_none  <- bake(prep_none, new_data = test_data)

#Desvio padrão
rec_std <- recipe(resposta_tratamento ~ ., data = train_data) %>%
  step_rm(id,acupuncturist,practice_id) %>%
  update_role(everything(), new_role = "predictor") %>%
  update_role(resposta_tratamento, new_role = "outcome") %>%
  step_center(all_of(variaveis_numericas)) %>%
  step_scale(all_of(variaveis_numericas))

prep_std <- prep(rec_std, training = train_data)
train_std <- bake(prep_std, new_data = NULL)
test_std  <- bake(prep_std, new_data = test_data)

#Min-Max
rec_minmax <- recipe(resposta_tratamento ~ ., data = train_data) %>%
  step_rm(id,acupuncturist,practice_id) %>%
  update_role(everything(), new_role = "predictor") %>%
  update_role(resposta_tratamento, new_role = "outcome") %>%
  step_range(all_of(variaveis_numericas), min = 0, max = 1)

prep_minmax <- prep(rec_minmax, training = train_data)
train_minmax <- bake(prep_minmax, new_data = NULL)
test_minmax  <- bake(prep_minmax, new_data = test_data)
```

```
# Escalonamento Robusto
rec_robust <- recipe(resposta_tratamento ~ ., data = train_data) %>%
  step_rm(id,acupuncturist,practice_id) %>%
  update_role(everything(), new_role = "predictor") %>%
  update_role(resposta_tratamento, new_role = "outcome") %>%
  # step_YeoJohnson tenta corrigir assimetria;
  # em conjunto com step_center e step_scale, vira algo mais robusto.
  step_YeoJohnson(all_of(variaveis_numericas)) %>%
  step_center(all_of(variaveis_numericas)) %>%
  step_scale(all_of(variaveis_numericas))

prep_robust <- prep(rec_robust, training = train_data)
train_robust <- bake(prep_robust, new_data = NULL)
test_robust  <- bake(prep_robust, new_data = test_data)

library(bestNormalize)  # para step_orderNorm()

rec_quantile <- recipe(resposta_tratamento ~ ., data = train_data) %>%
  step_rm(id,acupuncturist,practice_id) %>%
  update_role(everything(), new_role = "predictor") %>%
  update_role(resposta_tratamento, new_role = "outcome") %>%
  # Aplica a transformação rank-based (approx normal) nas variáveis numéricas
  step_orderNorm(all_of(variaveis_numericas))

prep_quantile <- prep(rec_quantile, training = train_data)
train_quantile <- bake(prep_quantile, new_data = NULL)
test_quantile  <- bake(prep_quantile, new_data = test_data)

treinar_modelos <- function(train_df, test_df,
                            target = "resposta_tratamento") {

  # Definir a mesma semente para cada treinamento (comparável)
  set.seed(1234)

 modelo_glmnet <- train(
    form, data = train_df,
    method = "glmnet",
    family = "binomial",
    trControl = ctrl,
```

```
  metric = "ROC",
  tuneGrid = expand.grid(alpha = seq(0, 1, 0.1), lambda = 10^seq(-3, 0, length = 10))
)

modelo_rf <- train(
  form, data = train_df,
  method = "rf",
  trControl = ctrl,
  metric = "ROC",
  importance = TRUE
)

modelo_xgb <- train(
  form, data = train_df,
  method = "xgbTree",
  trControl = ctrl,
  metric = "ROC",
  verbosity = 0
)

modelo_svm <- train(
  form, data = train_df,
  method = "svmRadial",
  trControl = ctrl,
  metric = "ROC"
)

modelos <- list(
  GLMNet = modelo_glmnet,
  RandomForest = modelo_rf,
  XGBoost = modelo_xgb,
  SVM = modelo_svm
)

calc_metricas <- function(mod) {
  pred <- predict(mod, newdata = test_df)
  cm <- confusionMatrix(pred, test_df[[target]], positive = "Melhoria")

  # Cálculo de probabilidades para AUC
  probs <- predict(mod, newdata = test_df, type = "prob")[, "Melhoria"]
```

```
    roc_obj <- pROC::roc(response = test_df[[target]], predictor = probs)
    auc_val <- pROC::auc(roc_obj)

    # Cálculo de F1 e MCC
    precision <- cm$byClass["Pos Pred Value"]
    recall <- cm$byClass["Sensitivity"]
    f1 <- 2 * (precision * recall) / (precision + recall)

#SEM Escalonamento
resultados_none <- treinar_modelos(train_none, test_none)

#Standard
resultados_std <- treinar_modelos(train_std, test_std)

#MinMax
resultados_minmax <- treinar_modelos(train_minmax, test_minmax)

#Robusto
resultados_robust <- treinar_modelos(train_robust, test_robust)

#Quantilica
resultados_quantile <- treinar_modelos(train_quantile, test_quantile)

comparacao_df <- bind_rows(
  resultados_none$Resultados  %>% mutate(Escalonamento = "Nenhum"),
  resultados_std$Resultados   %>% mutate(Escalonamento = "Standard"),
  resultados_minmax$Resultados %>% mutate(Escalonamento = "MinMax"),
  resultados_robust$Resultados  %>% mutate(Escalonamento = "Robust"),
  resultados_quantile$Resultados  %>% mutate(Escalonamento = "Quantil")
)

base_final_v3 <- base_final_v3 %>%
  mutate(acupuncturist = as.factor(acupuncturist),
         practice_id = as.factor(practice_id))

contents(base_final_v3)

levels(base_final_v3$resposta_tratamento) <- c("Estabilidade", "Melhoria")
table(base_final_v3$resposta_tratamento)
# Para GLMNet
```

```
conf_glmnet_none <- confusionMatrix(predict(resultados_none$Modelos$GLMNet,
newdata=test_none),
test_none$resposta_tratamento)

conf_glmnet_minmax <- confusionMatrix(predict(resultados_minmax$Modelos$GLMNet,
newdata=test_minmax),
test_minmax$resposta_tratamento)

# Para Random Forest
conf_rf_none <- confusionMatrix(predict(resultados_none$Modelos$RandomForest,
newdata=test_none),
test_none$resposta_tratamento)


conf_rf_quantile <- confusionMatrix(predict(resultados_quantile$Modelos$RandomForest,
newdata=test_quantile),
test_quantile$resposta_tratamento)

# Para XGBoost
conf_xgb_none <- confusionMatrix(predict(resultados_none$Modelos$XGBoost,
newdata=test_none),
test_none$resposta_tratamento)

conf_xgb_quantile <- confusionMatrix(predict(resultados_quantile$Modelos$XGBoost,
newdata=test_quantile),
test_quantile$resposta_tratamento)

# Para SVM
conf_svm_none <- confusionMatrix(predict(resultados_none$Modelos$SVM,
newdata=test_none),
test_none$resposta_tratamento)

conf_svm_minmax <- confusionMatrix(predict(resultados_minmax$Modelos$SVM,
newdata=test_minmax),
test_minmax$resposta_tratamento)
```