Software R para aplicações em IA

Anderson Castro Soares de Oliveira
João Pedro Florentino de Oliveira Nascimento

2023-11-23

Table of contents

Prefácio

Bem-vindo(a) à jornada de descoberta e aprendizado sobre Inteligência Artificial com o Software R!

Este livro foi concebido com um propósito claro e animador: servir como um guia essencial para a nossa oficina intitulada **Software R para Aplicações em Inteligência Artificial**, parte da **XII Escola Regional de Informática de Mato Grosso (ERI-MT)**, promovida pelo **Instituto de Computação** (IC) da **Universidade Federal de Mato Grosso** (UFMT).

O objetivo deste guia é proporcionar um caminho estruturado e profundo para aqueles que desejam explorar o potencial do R no campo da Inteligência Artificial (IA), desde os fundamentos básicos até aplicações mais avançadas.

Neste ambiente de aprendizado, buscamos não apenas transmitir conhecimentos técnicos, mas também inspirar uma compreensão holística de como o R, uma ferramenta estatística poderosa e versátil, se encaixa no ecossistema da IA. Este livro, portanto, é mais do que apenas uma coleção de instruções e códigos; é um convite para você se aprofundar no mundo fascinante da análise de dados e da inteligência artificial.

Para Quem é Este Livro?

Este livro é destinado a um amplo espectro de leitores - desde estudantes e acadêmicos até profissionais e entusiastas da área de dados. Não importa se você é um iniciante no R ou já possui alguma experiência; aqui, você encontrará conteúdos que enriquecerão seu conhecimento e habilidades. A estrutura do livro é pensada para facilitar tanto o aprendizado incremental quanto a consulta rápida para tópicos específicos.

Como Usar Este Livro

O livro está estruturado de maneira a acompanhar a progressão da oficina, mas também serve como uma ótima fonte de consulta autônoma. Iniciamos com os conceitos básicos e fundamentos do R, avançando para aplicações práticas em IA, como machine learning e mineração de texto. Cada capítulo é composto por explicações teóricas, exemplos práticos e exercícios, visando uma compreensão integral dos tópicos abordados.

Recomendamos que use este livro como um guia prático: experimente os códigos, modifique-os e veja os resultados por si mesmo. A prática é essencial no aprendizado de qualquer linguagem de programação, e com o R, não é diferente.

Boa leitura e bom aprendizado!

1 Introdução e Contextualização

Este livro é projetado para levar o público a explorar o universo do software R e sua relação com a Inteligência Artificial (IA). O material busca iluminar as diversas facetas do R, destacando seu papel essencial e em constante evolução no mundo da IA.

Neste livro, iniciamos desvendando as sutilezas entre estatística, machine learning e IA, mostrando como o R serve como uma ponte entre a análise estatística tradicional e as inovações contemporâneas em IA.

Prosseguimos com uma imersão nas origens do R, destacando suas raízes estatísticas e sua evolução como uma ferramenta fundamental na pesquisa científica. Este passeio histórico fornece uma base sólida para entender o papel do R na IA moderna, enriquecendo sua compreensão da linguagem e de suas aplicações.

A seguir, focamos em aplicações práticas de IA utilizando o R. Os leitores terão uma visão completa do machine learning, abordando desde a regressão linear até árvores de decisão. Esta seção demonstra como o R pode ser aplicado para resolver problemas complexos de IA na prática.

Avançamos para a área da mineração de texto utilizando o R. Do pré-processamento básico a técnicas avançadas de modelagem e análise, este segmento do livro é dedicado a imergir os leitores nas práticas mais eficazes da mineração de texto, mostrando como o R pode ser usado para extrair insights valiosos de dados textuais.

Concluímos este material com uma análise das tendências e inovações recentes em análise de texto e IA. Esta seção prepara os leitores para aplicar o conhecimento adquirido e para se manterem atualizados com as futuras direções da IA e do processamento de dados.

2 Estatística, Machine Learning e IA

Embora os termos estatística, machine learning (aprendizado de máquina) e inteligência artificial (IA) sejam frequentemente usados como sinônimos, eles abrangem campos distintos com métodos, aplicações e filosofias próprias. Compreender essas diferenças é essencial para aplicar o conhecimento de forma eficaz em cada uma dessas áreas, especialmente no contexto do desenvolvimento tecnológico e da inovação (Bzdok, Altman, and Krzywinski 2018; Giorgi, Ceraolo, and Mercatelli 2022; Jalajakshi and Myna 2022; Mailund 2017; Tahsin and Hasan 2020).

2.1 Estatística: A Fundação

A estatística pode ser considerada o alicerce sobre o qual Machine Learning (ML) e Inteligência Artificial (IA) são construídos. Tradicionalmente, a estatística lida com a coleta, análise, interpretação e apresentação de dados. No contexto do ensino e pesquisa, isso se traduz em uma ampla gama de testes, modelos e métodos de análise exploratória de dados (Bzdok, Altman, and Krzywinski 2018; Jalajakshi and Myna 2022).

A **Estatística** é uma ciência que se concentra na coleta, análise, interpretação e apresentação de dados. Ela utiliza teorias probabilísticas para estimar incertezas, testar hipóteses e fazer inferências a partir de amostras de dados. A estatística é fundamental na pesquisa científica e na tomada de decisões baseada em dados, oferecendo ferramentas para entender e modelar a variação e as relações nos dados(Hothorn 2023; James et al. 2023; Zbicki and Santos 2020). Seus principais enfoques são:

- **Inferência Estatística:** A estatística foca em inferir propriedades de uma população a partir de amostras. Este processo envolve a estimativa de parâmetros, testes de hipóteses e a criação de intervalos de confiança. É fundamental na avaliação e validação de modelos de ML e IA.
- **Análise Exploratória de Dados (EDA):** Antes de aplicar técnicas avançadas de ML e IA, os estatísticos realizam a EDA para entender melhor as características dos dados. Isso inclui identificar tendências, padrões, outliers e a estrutura básica dos dados.

- **Modelagem Estatística:** Diferente de algumas técnicas de ML e IA, a modelagem estatística muitas vezes procura não apenas prever, mas também explicar as relações entre variáveis. Modelos como regressões lineares e logísticas são clássicos exemplos.

- **Tratamento da Incerteza:** A estatística fornece ferramentas para lidar com a incerteza e a variabilidade nos dados. Isso é essencial para a tomada de decisões baseadas em dados, especialmente em contextos onde os dados são limitados ou ruidosos.

2.2 Machine Learning: Construindo sobre Estatística

ML é um subcampo da IA, é primariamente focado em desenvolver algoritmos que podem 'aprender' a partir de dados e fazer previsões ou tomar decisões baseadas nesses dados. Diferentemente da estatística tradicional, que frequentemente depende de modelos especificados previamente, o machine learning se concentra mais em algoritmos que se ajustam e melhoram automaticamente através da exposição a mais dados. Enquanto a estatística pode se concentrar mais na interpretação e na inferência, o machine learning prioriza a precisão preditiva e a capacidade de generalizar para novos dados (Zbicki and Santos 2020; James et al. 2023).

Dentro do ML os algoritmos são comumente categorizados em dois tipos principais: aprendizado supervisionado e não supervisionado. O aprendizado supervisionado envolve o uso de conjuntos de dados rotulados, onde cada exemplo de treinamento tem um rótulo ou resposta correspondente. Este tipo é utilizado para tarefas como classificação e regressão, onde o modelo aprende a mapear entradas para saídas conhecidas. Já o aprendizado não supervisionado é aplicado a dados sem rótulos pré-definidos, focando na descoberta de padrões e estruturas intrínsecas aos dados. Este método é ideal para tarefas como agrupamento, redução de dimensionalidade e identificação de regras de associação. Ambos os tipos têm aplicações variadas e são escolhidos com base nas características e objetivos específicos do problema de ML em questão (Zbicki and Santos 2020; James et al. 2023).

2.2.1 Aprendizado Supervisionado

No aprendizado supervisionado, os modelos são treinados usando um conjunto de dados rotulado. Isso significa que cada exemplo no conjunto de dados é pareado com a resposta ou resultado correto. O objetivo é que o modelo aprenda a **mapear os dados de entrada para as respostas** (Zbicki and Santos 2020; James et al. 2023).

2.2.1.1 Regressão

No contexto do aprendizado supervisionado, a regressão lida com a previsão de valores quantitativos (discretos ou continuous). O objetivo é desenvolver um modelo que possa prever um valor numérico, como preço, temperatura ou vendas, a partir de um conjunto de variáveis de entrada (Zbicki and Santos 2020; James et al. 2023; Burger 2018).

Definindo o Problema de Regressão

Um problema de regressão é caracterizado da seguinte forma (Zbicki and Santos 2020; Burger 2018):

- **Dados de Entrada e Saída:** Em um problema de regressão, os dados de entrada podem ser uma ou mais variáveis preditoras (features), e a saída é uma variável contínua. Por exemplo, prever o preço de uma casa com base em seu tamanho, localização e idade.

- **Modelos Comuns:** Alguns dos modelos de regressão mais comuns incluem regressão linear simples e múltipla, regressão polinomial e regressão com regularização (como Lasso e Ridge).

Avaliando Modelos de Regressão

A avaliação de modelos de regressão foca em quão bem o modelo prevê valores contínuos. As métricas comuns incluem (Zbicki and Santos 2020; James et al. 2023; Burger 2018):

- **Erro Quadrático Médio (MSE):** Mede a média dos quadrados dos erros, ou seja, a média das diferenças quadradas entre os valores observados e os valores previstos pelo modelo.

- **Raiz do Erro Quadrático Médio (RMSE):** É a raiz quadrada do MSE, fornecendo uma medida de erro em uma escala comparável aos valores originais.

- **Erro Absoluto Médio (MAE):** Mede a média das diferenças absolutas entre previsões e valores reais, fornecendo uma ideia da magnitude do erro sem considerar sua direção.

- **Coeficiente de Determinação (R^2)**: Mede a proporção da variância total dos dados que é explicada pelo modelo. Um valor de R^2 próximo de 1 indica que o modelo explica uma grande parte da variação nos dados.

Desafios Comuns na Regressão (Zbicki and Santos 2020; James et al. 2023; Burger 2018):

- **Overfitting e Underfitting:** Overfitting ocorre quando um modelo é excessivamente complexo, adaptando-se demais aos dados, incluindo o ruído (erro), e falhando ao generalizar para novos dados. Underfitting, por outro lado, acontece quando o modelo é muito simples para capturar a complexidade dos dados, resultando em um desempenho fraco tanto nos dados.
- **Linearidade:** Muitos modelos de regressão assumem que existe uma relação linear entre as variáveis de entrada e a saída. Quando esta suposição não é válida, o modelo pode não performar bem, pois não consegue capturar as relações não lineares nos dados.
- **Multicolinearidade:** Este problema surge quando há uma alta correlação entre duas ou mais variáveis de entrada do modelo. Isso pode levar a dificuldades na estimação dos efeitos individuais das variáveis de entrado sobre a variável de saída, além de potencialmente causar instabilidade nos coeficientes estimados do modelo.

2.2.1.2 Classificação

A classificação é um tipo de problema de aprendizado supervisionado focado na previsão de variáveis categóricas, como rótulos ou classes, diferentemente da regressão, que prevê valores quantitativos. A classificação trabalha com categorias ou valores qualitativos (Zbicki and Santos 2020; James et al. 2023; Burger 2018).

Definindo o Problema de Classificação

Um problema de classificaçao é caracterizado da seguinte forma (Zbicki and Santos 2020; James et al. 2023; Burger 2018):

- **Dados de Entrada e Saída:** Em um problema de classificação, os dados de entrada podem ser uma ou mais variáveis preditoras denominadas atributos, e a saída é um variável qualititiva ou categoria. Por exemplo, identificar se um individuo tem Dengue, baseado em informações de Idade, Temperatura, Febre, Enjôo, Manchas e Dor.
- **Modelos Comuns:** Incluem regressão logística, máquinas de vetores de suporte (SVM), árvores de decisão, florestas aleatórias e redes neurais.

Avaliando Modelos de Classificação

A avaliação em classificação foca em quão precisamente o modelo pode classificar as entradas. Algumas métricas comuns incluem (Zbicki and Santos 2020; James et al. 2023; Burger 2018):

- **Acurácia:** A proporção de previsões corretas em relação ao total de casos. Apesar de ser intuitiva, não é sempre a melhor métrica, especialmente se os dados são desbalanceados.

- **Precisão e Recall:** Precisão é a proporção de previsões positivas corretas, enquanto recall (ou sensibilidade) é a proporção de casos positivos reais que foram identificados corretamente.
- **F1-Score:** Uma média harmônica entre precisão e recall. Útil quando se busca um equilíbrio entre precisão e recall.
- **Curva ROC e AUC:** A curva ROC (Receiver Operating Characteristic) é um gráfico da taxa de verdadeiros positivos contra a taxa de falsos positivos. A AUC (Area Under the Curve) é uma medida do desempenho do modelo que considera todas as taxas de classificação possíveis.

Desafios Comuns na Classificação (Zbicki and Santos 2020; James et al. 2023; Burger 2018):

- **Desequilíbrio de Classes:** Quando uma classe é muito mais frequente do que outras, o modelo pode se inclinar para a classe mais comum, reduzindo a precisão geral.
- **Overfitting e Underfitting:** Similar à regressão, a classificação também pode sofrer de overfitting e underfitting, afetando a capacidade do modelo de generalizar para novos dados.
- **Interpretabilidade:** Para alguns modelos, como redes neurais profundas, pode ser difícil interpretar como a decisão de classificação foi feita.

2.2.2 Aprendizado Não Supervisionado

No aprendizado não supervisionado, os modelos são treinados usando dados que não possuem rótulos ou categorias pré-definidas. O foco é na descoberta de padrões, estruturas ou insights intrínsecos nos dados sem a orientação de um resultado específico (Zbicki and Santos 2020; James et al. 2023; Burger 2018).

2.2.2.1 Agrupamento (Clustering)

Uma das tarefas mais comuns no aprendizado não supervisionado é o agrupamento, onde o objetivo é dividir o conjunto de dados em grupos (clusters) baseados em semelhanças (Zbicki and Santos 2020; James et al. 2023; Burger 2018).

Definindo o Problema de Agrupamento

- **Dados de Entrada:** Diferente do aprendizado supervisionado, os dados de entrada não são acompanhados por rótulos ou respostas corretas. Por exemplo, segmentar clientes com base em comportamento de compra sem uma categorização prévia.

- **Métodos Comuns:** K-means, agrupamento hierárquico e DBSCAN são alguns dos algoritmos populares usados para agrupamento.

Avaliando Modelos de Agrupamento

Avaliar o desempenho em agrupamento é desafiador devido à falta de rótulos verdadeiros. Algumas abordagens incluem (Zbicki and Santos 2020; James et al. 2023; Burger 2018):

- **Índice de Silhueta:** Mede quão bem um ponto foi agrupado, calculando a diferença entre a coesão dentro do cluster e a separação entre clusters.
- **Dunn Index:** Enfatiza a distância entre os clusters e a dispersão dentro de cada cluster.
- **Validação Cruzada Baseada em Estabilidade:** Compara a estabilidade dos clusters criados a partir de diferentes subconjuntos dos dados.

2.2.2.2 Redução de Dimensionalidade

Outra tarefa importante no aprendizado não supervisionado é a redução de dimensionalidade, que busca simplificar os dados preservando o máximo de informações relevantes (Zbicki and Santos 2020; James et al. 2023; Burger 2018).

Definindo a Redução de Dimensionalidade

- **Objetivo:** Reduzir o número de variáveis (features) nos dados, facilitando a visualização, interpretação e, em alguns casos, o processamento subsequente dos dados.
- **Métodos Comuns:** Análise de Componentes Principais (PCA), t-SNE e UMAP são técnicas amplamente utilizadas.

Avaliando a Redução de Dimensionalidade

- **Variação Preservada:** Em métodos como o PCA, uma métrica importante é a quantidade de variação dos dados originais que é preservada após a redução.
- **Qualidade da Representação:** Em técnicas como t-SNE e UMAP, avalia-se a qualidade visualizando se os dados reduzidos mantêm as relações estruturais dos dados originais.

2.2.2.3 Desafios Comuns no Aprendizado Não Supervisionado

- **Interpretação dos Resultados:** Os resultados do aprendizado não supervisionado podem ser subjetivos e sua interpretação muitas vezes requer conhecimento de domínio.
- **Seleção de Parâmetros:** A escolha de parâmetros, como o número de clusters no K-means, pode ter um grande impacto nos resultados e requer experimentação.
- **Qualidade dos Dados:** O aprendizado não supervisionado pode ser sensível à qualidade dos dados, incluindo ruídos e outliers.

2.3 Inteligência Artificial: Uma Visão Ampla

Definição e Escopo

A Inteligência Artificial (IA) é um campo abrangente que inclui o Machine Learning (ML) e outras técnicas que podem ou não ser baseadas em dados. A IA envolve o desenvolvimento de sistemas capazes de realizar tarefas que normalmente exigem inteligência humana, como percepção, raciocínio, aprendizado e tomada de decisões. Além do ML, a IA engloba áreas como processamento de linguagem natural, robótica e visão computacional (Thaichon and Quach 2022).

Tipos de IA

- **IA Fraca (ou Estreita):** Focada em tarefas específicas, como reconhecimento de voz ou processamento de linguagem natural, representando a maioria das aplicações atuais de IA.
- **Forte (ou Geral):** Visa criar um sistema com capacidade intelectual geral comparável à humana, capaz de resolver uma ampla variedade de problemas. Este tipo de IA ainda é um objetivo de longo prazo na pesquisa.

Aplicações de IA (Thaichon and Quach 2022).

- **Reconhecimento de Voz e Processamento de Linguagem Natural (PLN):** Usado em assistentes virtuais, tradução automática e análise de sentimentos.
- **Visão Computacional:** Aplicações em reconhecimento facial, diagnósticos médicos por imagem e sistemas de vigilância.
- **Robótica:** Desde robôs industriais até drones autônomos e veículos autônomos.
- **Sistemas de Recomendação:** Como os usados por plataformas de streaming e e-commerce para sugerir produtos ou conteúdos.

Desafios e Considerações Éticas (Thaichon and Quach 2022).

- **Transparência e Explicabilidade:** Entender como as decisões são feitas por sistemas de IA é crucial, especialmente em áreas sensíveis como saúde e justiça criminal.
- **Viés e Justiça:** A IA pode perpetuar ou até amplificar vieses presentes nos dados ou nos processos de desenvolvimento.
- **Privacidade de Dados:** A coleta e utilização de dados em grande escala pela IA levanta preocupações significativas de privacidade.
- **Automação e Impacto no Emprego:** A automação por IA tem o potencial de transformar o mercado de trabalho, criando novas oportunidades e desafios.

3 R: Breve Histórico e Importância

Neste capítulo, mergulhamos na evolução do R, uma linguagem e ambiente originalmente voltado para análise estatística e gráfica, que evoluiu para se tornar uma ferramenta essencial na ciência de dados e Inteligência Artificial (IA). Exploraremos como o R se desenvolveu de suas origens modestas para alcançar um papel de destaque no cenário tecnológico atual.

O Início e Desenvolvimento do R

Ross Ihaka e Robert Gentleman, da Universidade de Auckland, criaram o R em 1995, concebendo-o como uma alternativa mais acessível e flexível às ferramentas estatísticas da época, especialmente em comparação com o S-Plus. O S-Plus, uma implementação comercial da linguagem de programação S desenvolvida nos Bell Laboratories, era notável por sua capacidade analítica e gráfica, mas era restrito por limitações de licenciamento e acessibilidade (Ihaka and Gentleman 1996; Ihaka 1998)

Inspirado pela linguagem S, o R foi desenvolvido com o objetivo de oferecer uma solução de código aberto que pudesse ser livremente utilizada e modificada pela comunidade acadêmica e de pesquisa. Essa natureza de código aberto permitiu que o R rapidamente se destacasse no campo da análise estatística e gráfica, promovendo uma colaboração global intensa e levando ao desenvolvimento de uma plataforma robusta (Giorgi, Ceraolo, and Mercatelli 2022; Ihaka and Gentleman 1996; W. Venables and Ripley 2013).

A influência do S-Plus é perceptível no R, particularmente na similaridade de várias funções e na filosofia de design. No entanto, o R expandiu e evoluiu além desses conceitos iniciais, especialmente em termos de colaboração comunitária e extensibilidade. Com o passar do tempo, o R se distanciou de suas raízes no S-Plus, desenvolvendo uma identidade própria e uma comunidade dedicada. Esta comunidade continua a impulsionar seu desenvolvimento e aplicação em diversas áreas da ciência de dados e IA, reforçando a importância e o impacto do R no mundo tecnológico moderno (Giorgi, Ceraolo, and Mercatelli 2022; Ihaka and Gentleman 1996).

3.1 R no Contexto da Análise Estatística e Ciência de Dados

A importância do R no campo da análise estatística e ciência de dados é amplamente reconhecida. Com sua vasta coleção de pacotes estatísticos e gráficos, o R se adequa a uma gama

diversa de análises, abrangendo desde tarefas básicas até procedimentos altamente complexos. O software é particularmente notável por sua habilidade em gerenciar grandes conjuntos de dados, executar operações complexas de manipulação de dados, e produzir visualizações gráficas de alta qualidade (Giorgi, Ceraolo, and Mercatelli 2022; Donoho 2017)

Além de sua funcionalidade estatística, o R desempenha um papel crucial no desenvolvimento da ciência de dados como uma área de pesquisa dinâmica. Sua comunidade ativa tem contribuído imensamente para o enriquecimento do R, desenvolvendo pacotes e bibliotecas que expandem suas capacidades para além da análise estatística tradicional. Hoje, o R é uma ferramenta indispensável em campos como análise preditiva, modelagem estatística, mineração de dados e, mais recentemente, em aplicações de aprendizado de máquina e IA (Giorgi, Ceraolo, and Mercatelli 2022; W. Venables and Ripley 2013; H. Wickham and Grolemund 2016).

3.1.1 O R na Era da Inteligência Artificial

Com a ascensão da IA, o R tem se adaptado para atender às demandas deste campo em rápida evolução. Embora tradicionalmente não seja considerado a primeira escolha para aplicações de aprendizado de máquina e IA em comparação a outras linguagens, o R tem ganhado terreno significativo. Esta evolução é marcada pelo desenvolvimento de pacotes específicos para IA no R e pela sua crescente integração com plataformas e frameworks de ponta na área (James et al. 2023; Kalyan 2018; Tuffery 2023).

Além desses pacotes, uma das evoluções mais notáveis é a integração do R com outras plataformas de IA, especialmente com o Python. Essa integração permite que os usuários do R aproveitem as bibliotecas de IA do Python, combinando o melhor dos dois mundos: a análise estatística e gráfica avançada do R com as robustas capacidades de IA disponíveis no Python. Isso reflete a crescente relevância do R como uma ferramenta versátil na era da IA, capaz de se adaptar e incorporar inovações de outras linguagens e tecnologias (Kalyan 2018; Ohri 2017; Tahsin and Hasan 2020).

3.1.2 A Comunidade e o Ecossistema do R

Um dos maiores trunfos do R é sua vibrante comunidade. Com usuários e desenvolvedores ao redor do mundo, o R beneficia-se de uma ampla gama de perspectivas e experiências, contribuindo para sua contínua inovação e aprimoramento. Conferências como o "useR!", DSC, workshops, e fóruns online, como o R-help e Stack Overflow, são testemunhos da colaboração e do compartilhamento de conhecimento que impulsionam o crescimento do R. (R Project 2023a, 2023b; Stack Overflow 2023)

3.2 Popularidade do R na Pesquisa Científica

O R, desde o início, estabeleceu-se como uma ferramenta de pesquisa essencial em uma variedade de campos científicos. Seu uso se estende desde a biologia e epidemiologia até a economia e psicologia, provendo uma plataforma para análise estatística e visualização de dados. Essa versatilidade é evidenciada pelo crescente número de publicações e estudos de pesquisa que fazem uso do R, abrangendo desde análises básicas de dados até modelagens complexas e simulações (Giorgi, Ceraolo, and Mercatelli 2022; Ihaka 1998; Tippmann 2015).

3.2.1 Estudos de Caso e Aplicações em Diversos Campos

1. Bioestatística e Epidemiologia - o R é utilizado para analisar e interpretar dados relacionados à saúde. Por exemplo, é crucial no monitoramento e na modelagem da propagação de doenças infecciosas. Ferramentas de análise de sobrevivência e modelos estatísticos complexos no R ajudaram pesquisadores a entender padrões de doenças e a eficácia de intervenções médicas (Chan 2015).

2. Genômica - O R tem desempenhado um papel significativo na genômica, especialmente na análise de dados de sequenciamento de alta performance. Pacotes como Bioconductor fornecem ferramentas para a análise de expressão gênica, ajudando na identificação de genes associados a diferentes condições de saúde e na compreensão de mecanismos genéticos (Paradis 2020).

3. Análise de Experimentos - Em experimentos científicos e industriais, o R é frequentemente utilizado para desenho experimental e análise de dados. Sua capacidade de lidar com complexidades como fatores de confusão, interações e estruturas de erro heterogêneas torna-o ideal para esta área (Lawson 2014).

4. Economia e Econometria -o R é empregado para análise de dados econômicos, previsão de tendências de mercado e avaliação de políticas. Ele oferece um conjunto diversificado de pacotes para modelagem econométrica, análise de séries temporais e testes de hipóteses, contribuindo para uma compreensão mais profunda de fenômenos econômicos (Singh and Allen 2016).

5. Ciências Ambientais - O R é também uma ferramenta chave nas ciências ambientais, usada para modelar dados climáticos, avaliar biodiversidade e estudar ecologia. Ele auxilia cientistas a compreender padrões climáticos, impactos ambientais de ações humanas e a preservar ecossistemas (Al-Karkhi and Alqaraghuli 2019).

3.2.2 Contribuição da Comunidade Científica para o Desenvolvimento do R

A comunidade de usuários do R, uma coligação diversificada de cientistas e pesquisadores de várias disciplinas, tem desempenhado um papel fundamental na evolução contínua deste software. A natureza de código aberto do R tem encorajado uma participação ativa, onde os usuários não se limitam apenas a aplicar a ferramenta em suas pesquisas, mas também contribuem significativamente para o seu desenvolvimento. Esta contribuição vai além do uso convencional; eles inovam, criando pacotes e extensões que atendem às necessidades específicas de suas respectivas áreas de estudo. O resultado é um enriquecimento constante do ecossistema do R, com novas funcionalidades e ferramentas que ampliam sua aplicabilidade e eficácia (Giorgi, Ceraolo, and Mercatelli 2022; Ihaka 1998; R Project 2023b; Tippmann 2015).

Além do desenvolvimento de pacotes, a comunidade do R também desempenha um papel crucial na disseminação de conhecimento e na formação de uma base sólida de suporte. Este intercâmbio de conhecimentos e experiências não só fortalece a base de usuários do R, mas também impulsiona o avanço da ciência de dados como um todo, demonstrando o poder da colaboração e da comunidade na evolução tecnológica.

3.2.3 Adaptação às Necessidades Emergentes em Ciência de Dados e IA

À medida que os campos de ciência de dados e Inteligência Artificial (IA) continuam a se expandir e evoluir, o R tem demonstrado uma notável capacidade de adaptação e inovação. A integração do R com ferramentas avançadas de aprendizado de máquina e IA é um exemplo claro dessa evolução. Esta integração não apenas expandiu o escopo de aplicabilidade do R, mas também permitiu que cientistas e analistas de dados realizassem análises mais complexas e sofisticadas. Com pacotes específicos para algoritmos de aprendizado de máquina e redes neurais, o R agora é capaz de lidar com tarefas de IA que antes eram consideradas fora de seu alcance (James et al. 2023; Kalyan 2018; Tuffery 2023).

Além disso, a capacidade do R de se integrar com outras linguagens e plataformas, como Python e TensorFlow, destaca sua flexibilidade e relevância contínua na pesquisa científica. Esta interoperabilidade entre o R e outras tecnologias amplia as possibilidades de análise de dados, permitindo que os pesquisadores aproveitem as forças de várias ferramentas simultaneamente. Por exemplo, a integração do R com Python através de pacotes como reticulate possibilita a utilização conjunta das bibliotecas de IA do Python com as poderosas capacidades estatísticas do R, oferecendo uma abordagem mais holística e eficaz para a solução de problemas complexos de dados. Esta capacidade de adaptação contínua assegura que o R permaneça na vanguarda da tecnologia de análise de dados, atendendo às necessidades emergentes de cientistas e pesquisadores em um mundo cada vez mais orientado por dados (James et al. 2023; Kalyan 2018; Tuffery 2023)..

3.3 Vantagens e Limitações do R para IA

3.3.1 Vantagens do Uso do R em IA

Flexibilidade e Facilidade de Uso

- O R é conhecido por sua flexibilidade. Ele permite a realização de uma ampla gama de funções analíticas com relativa facilidade, desde a manipulação de dados até análises estatísticas avançadas.

- A linguagem é particularmente forte na visualização de dados, uma habilidade crucial na análise exploratória de dados, uma etapa importante na construção de modelos de IA.

Rica Biblioteca de Pacotes

- Uma das maiores vantagens do R é a sua vasta coleção de pacotes. Existem pacotes para quase todo tipo de análise estatística e modelo de machine learning, como `caret` (Kuhn and Max 2008), `randomForest` (Liaw and Wiener 2002), `e1071`, (Meyer et al. 2023) e muitos outros.

- A comunidade do R é muito ativa, o que significa que esses pacotes são regularmente atualizados e novos pacotes estão sempre sendo desenvolvidos.

Comunidade Robusta e Suporte

- A comunidade R é uma das mais colaborativas e ativas. Isso significa que é fácil encontrar suporte, seja por meio de fóruns, blogs, ou documentação detalhada.

- Conferências e workshops frequentes contribuem para o contínuo desenvolvimento profissional e para a expansão da rede de contatos na área.

Integração com Outras Linguagens e Ferramentas

- O R pode ser integrado com outras linguagens de programação, como Python, o que é uma vantagem considerável quando se trabalha em projetos de IA que podem requerer funcionalidades além das disponíveis diretamente no R.

3.3.2 Limitações e Desafios

Desempenho e Escalabilidade

- Uma das principais críticas ao R é relacionada ao seu desempenho com grandes conjuntos de dados. O R armazena dados na memória, o que pode limitar sua capacidade de lidar com grandes volumes de dados.
- Soluções envolvem a otimização do código ou a utilização de ferramentas que permitem o processamento de dados fora da memória.

Curva de Aprendizado em Programação

- Para usuários sem um forte background em programação, o R pode apresentar uma curva de aprendizado inicialmente desafiadora, especialmente quando se trata de escrever códigos mais complexos e eficientes.

3.4 Configuração e Ferramentas Essenciais

3.4.1 Instalação e Configuração Básica

- Para começar a trabalhar com IA no R, o primeiro passo é a instalação do próprio R:
  - A instalação do R pode ser feita por meio do site http:// cran.r-project.org/. Primeiro deve selecionar o sistema operacional: Linux, Mac ou Windows
  - Para o Windows é importante também instalar o Rtools https://cran.r-project.org/bin/windows/Rtools
- Também pode-se instalar o Rstudio
  - O RStudio é um ambiente de desenvolvimento integrado(IDE) para o R e traz algumas funcionalidades adicionais ao R.
  - Para instala-lo por meio do site https://posit.co/downloads/
- É importante também configurar o ambiente de trabalho, ajustando configurações para otimizar a eficiência e a facilidade de uso.

3.4.2 Gestão de Pacotes

- O R em geral é instalado apenas com as configurações mínimas para seu funcionamento básico (pacote base);
- Para realizar tarefas mais complexas pode ser necessário instalar pacotes adicionais (packages ou library);
- O gerenciamento eficiente de pacotes é crucial no R. Isso inclui saber como instalar e atualizar pacotes, bem como gerenciar dependências.
- Além disso, é útil entender como usar o CRAN (Comprehensive R Archive Network) e repositórios como o Bioconductor para encontrar e instalar pacotes relacionados a IA.

3.4.3 Pacotes de Machine Learning

- `caret` (Kuhn and Max 2008)é um dos pacotes mais populares para machine learning no R. Ele oferece uma interface consistente para treinar modelos usando uma variedade de algoritmos de aprendizado.
- Outros pacotes relevantes incluem `randomForest` (Liaw and Wiener 2002)para florestas aleatórias, `e1071` (Meyer et al. 2023)para máquinas de vetores de suporte, e `nnet` (W. N. Venables and Ripley 2002) para redes neurais .

3.4.4 Pacotes para Aprendizado Profundo

- Para quem está interessado em aprendizado profundo, pacotes como `keras` (Allaire and Chollet 2023a)e `tensorflow` (Allaire and Tang 2023) permitem a construção e treinamento de modelos de redes neurais profundas no R.
- Esses pacotes oferecem a flexibilidade necessária para construir modelos complexos, embora possam exigir um entendimento mais aprofundado da estrutura e funcionamento das redes neurais.

3.4.5 Integração e Expansão

- O R pode ser expandido e integrado com outras ferramentas e plataformas. Por exemplo, a integração com Python através do pacote `reticulate` (Ushey, Allaire, and Tang 2023) permite aos usuários acessar bibliotecas Python diretamente do R.
- Esta seção pode explorar como essas integrações podem ser configuradas e utilizadas, aumentando as capacidades do R em IA.

Ambientes Virtuais e Contêineres

- Discutir a importância de ambientes virtuais, como o Renv, para manter projetos isolados e gerenciar dependências de maneira mais eficaz.
- Também pode ser relevante abordar o uso de contêineres, como Docker, para criar ambientes de desenvolvimento replicáveis e consistentes.

4 Aplicações Básicas e Intermediárias em IA com R

Este capítulo apresenta uma visão abrangente das aplicações práticas da Inteligência Artificial (IA) utilizando o R. Cobrindo desde técnicas básicas até métodos intermediários, ele oferece aos leitores uma base sólida em teoria e prática. Os tópicos incluem Regressão Linear e Logística, Árvores de Decisão e Florestas Aleatórias, além de uma introdução a Redes Neurais. O capítulo também discute as tendências e avanços recentes em Machine Learning, proporcionando aos leitores insights sobre as futuras inovações no campo da IA.

4.1 Regressão Linear

A regressão linear é um método fundamental tanto em estatística quanto em machine learning. Ela é utilizada para modelar a relação entre uma variável de saída (dependente) contínua e uma ou mais variáveis de entrada (independentes). Esse método estabelece uma equação linear que descreve a relação entre essas variáveis, permitindo a previsão de valores da variável de saída com base em novos dados de entrada. Apesar de sua simplicidade, a regressão linear é uma ferramenta poderosa para análises preditivas e é frequentemente o ponto de partida para muitos estudos e análises em diversos campos (Zbicki and Santos 2020; James et al. 2023; Burger 2018).

Conceito: A ideia central da regressão linear é encontrar a melhor reta (ou, em casos de múltiplas variáveis independentes, um plano ou hiperplano) que se ajuste aos dados observados.

A reta (plano ou hiperplano) é obtida minimizando a diferença entre os valores reais observados nos dados e os valores previstos pelo modelo. Essa minimização geralmente é realizada através do método dos mínimos quadrados, buscando reduzir a soma dos quadrados das diferenças entre os valores observados e os previstos. Esse método fornece uma maneira eficiente de estimar os coeficientes do modelo linear, oferecendo uma previsão confiável baseada nas variáveis independentes (Singh and Allen 2016; Zbicki and Santos 2020).

A regressão linear é valiosa tanto para **visualizar tendências** quanto para **fazer previsões**. Ao ajustar uma linha a um conjunto de pontos de dados, ela facilita a **visualização** e a **compreensão** da relação entre as variáveis. Esta técnica se torna especialmente útil em grandes conjuntos de dados, onde pode ser desafiador identificar padrões. Por meio da regressão linear, torna-se mais simples discernir a relação entre variáveis, proporcionando informações que podem guiar análises mais profundas e decisões baseadas em dados (Singh and Allen 2016; Zbicki and Santos 2020):

- **Interpretação Gráfica:** A linha de regressão em um gráfico oferece uma interpretação visual imediata da relação entre as variáveis. Por exemplo, uma linha de regressão ascendente indica uma relação positiva, significando que à medida que uma variável aumenta, a outra também tende a aumentar.
- **Identificação de Anomalias:** Além de revelar tendências, a regressão linear ajuda a identificar outliers ou anomalias nos dados, que são pontos significativamente afastados da linha de regressão.

As aplicações práticas da regressão linear são vastas, abrangendo áreas como economia, meteorologia, saúde e mais, fornecendo previsões valiosas e insights para tomadas de decisão (Singh and Allen 2016; Zbicki and Santos 2020).

- **Previsões Baseadas em Dados:** Ao ajustar um modelo de regressão linear, é obtido uma equação que pode ser usada para fazer previsões. Por exemplo, em um modelo de regressão linear simples, essa equação pode ter a forma $y = mx + b$, em quende y é a variável de sáida (dependente), x é a variável de entrada (independente), m é a inclinação e b é o intercepto da linha de regressão.
- **Aplicações Práticas:** As previsões têm inúmeras aplicações práticas em diversos campos, como economia (previsão de tendências de mercado), meteorologia (previsão de temperaturas futuras), saúde (previsão de taxas de recuperação de pacientes), entre muitos outros.

Ao trabalhar com regressão linear, é crucial considerar alguns aspectos importantes (Singh and Allen 2016; Chan 2015):

- **Qualidade dos Dados:** A eficácia da regressão linear está diretamente relacionada à qualidade dos dados utilizados. Dados imprecisos, incompletos ou com erros podem resultar em previsões falhas ou enganosas.
- **Relações Lineares:** A regressão linear é ideal para situações em que a relação entre as variáveis é de fato linear. Se a relação for não-linear, modelos de regressão linear podem não ser adequados. Nestes casos podem ser aplicados modelos de regressão não linear e outras técnicas de machine learning podem ser mais apropriadas.

- **Relações Lineares**: A regressão linear é ideal para situações onde há uma relação linear entre as variáveis .Em cenários onde essa relação é não-linear a aplicação de modelos de regressão não linear ou outras técnicas de machine learning pode ser mais apropriada, permitindo uma modelagem mais precisa das complexidades inerentes aos dados.
- **Causalidade vs. Correlação:** É importante lembrar que a regressão linear por si só não implica causalidade. Ela pode identificar correlações entre variáveis, mas isso não implica uma relação de causa e efeito direta.

4.1.1 Exemplo Prático

4.1.1.1 Ajuste de modelo de regressão

Vamos considerar um conjunto de dados hipotético que representa uma cidade durante um verão particularmente quente. O objetivo é analisar a relação entre a temperatura média diária (em graus Celsius) e o consumo total diário de energia elétrica (em megawatts-hora). Espera-se que essa relação seja aproximadamente linear, com o consumo de energia aumentando à medida que as temperaturas se tornam mais altas.

Para ilustras, vamos gerar alguns dados simulados em R para representar esta situação:

```
library(tidyverse)
set.seed(123)  # Semente de numeros aleatorios para reprodutibilidade
temperatura <- 25:45  # Temperatura variando de 25 a 45 graus Celsius
consumo_energia <- 50 + 2.5 * temperatura + rnorm(21, mean = 0, sd = 5)
dados <- data.frame(temperatura, consumo_energia)
```

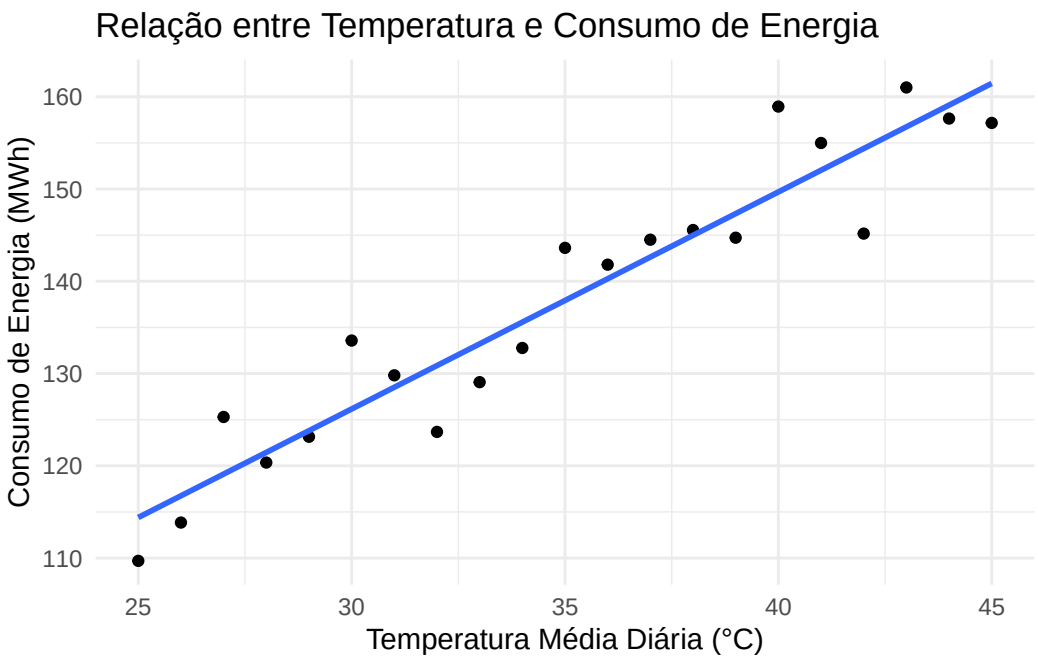
Vamos ajustar um modelo de regressão linear e visualizá-lo com pacote `ggplot2`(Hadley Wickham 2016) :

```
modelo <- lm(consumo_energia ~ temperatura, data = dados)
summary(modelo)
```

```

Call:
lm(formula = consumo_energia ~ temperatura, data = dados)

Residuals:
    Min      1Q  Median      3Q     Max
```

```
-9.2185 -2.9014 -0.6606  2.9560  9.2535

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  55.5938     6.3488   8.757 4.27e-08 ***
temperatura   2.3522     0.1787  13.160 5.37e-11 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 4.96 on 19 degrees of freedom
Multiple R-squared:  0.9011,    Adjusted R-squared:  0.8959
F-statistic: 173.2 on 1 and 19 DF,  p-value: 5.371e-11
```

```r
ggplot(dados, aes(x = temperatura, y = consumo_energia)) +
geom_point() +
geom_smooth(method = "lm", se = FALSE) +
theme_minimal() +
labs(title = "Relação entre Temperatura e Consumo de Energia",
     x = "Temperatura Média Diária (°C)",
     y = "Consumo de Energia (MWh)")
```



Neste exemplo, o modelo de regressão é representado pela equação $y = 2.35 + 55.59$ e ilustrado pela linha no gráfico, destacando a relação entre temperatura e consumo de energia. Os pontos no gráfico representam os dados observados, enquanto a linha demonstra a tendência geral. Isso sugere que existe uma correlação positiva entre o aumento da temperatura e o aumento no consumo de energia, com a linha de regressão oferecendo uma visualização clara dessa tendência.

4.1.1.2 Aplicação de Modelos de Regressão Linear para Previsões

Utilizando o mesmo exemplo da relação entre temperatura e consumo de energia, agora vamos explorar como o modelo de regressão linear pode ser usado para fazer previsões. O objetivo é estimar o consumo de energia com base na temperatura.

Primeiro, ajustamos o modelo de regressão linear, como fizemos anteriormente:

```
modelo <- lm(consumo_energia ~ temperatura, data = dados)
```

Com o modelo ajustado, podemos usar a função `predict()` predict() para fazer previsões. Por exemplo, se quisermos prever o consumo de energia para uma temperatura de 25.5, 28.2, 30, 38.5 graus Celsius, fazemos o seguinte:

```
temperatura_nova <- data.frame(temperatura = c(25.5,28.2,30,38.5))
previsao_consumo <- predict(modelo, newdata = temperatura_nova)
previsao_consumo
```

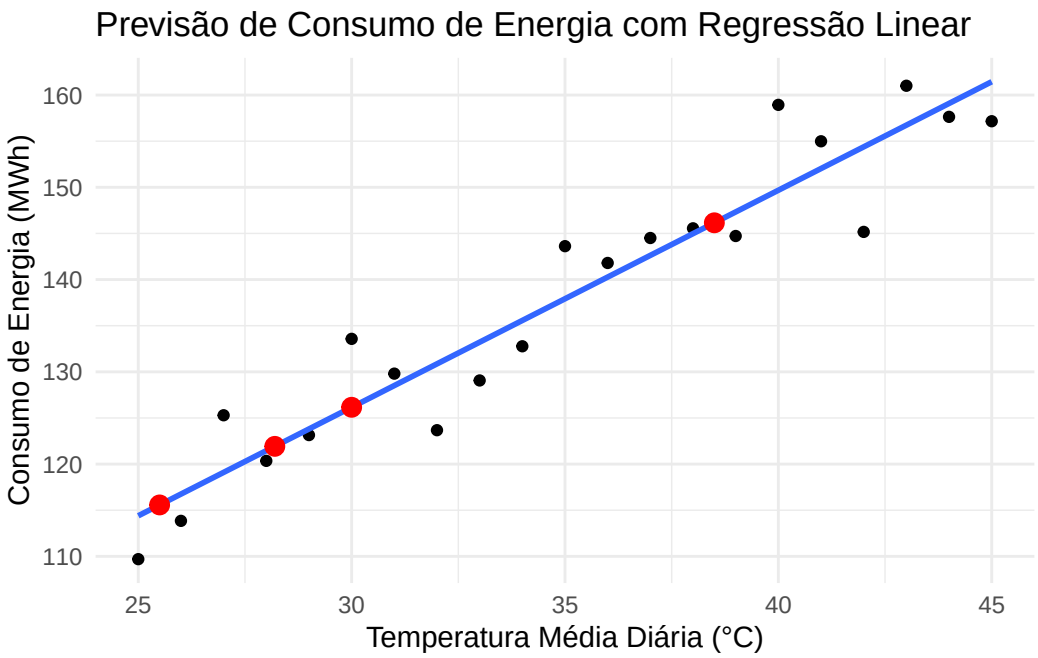
```
       1        2        3        4
115.5744 121.9253 126.1593 146.1528
```

Isso nos dará a previsão de consumo de energia para a temperatura especificada.

É útil visualizar as previsões juntamente com os dados originais e a linha de regressão. Isso pode ser feito ajustando o gráfico que criamos anteriormente:

```
dados1=cbind(temperatura_nova,previsao_consumo)

ggplot(dados, aes(x = temperatura, y = consumo_energia)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE)  +
  geom_point(data=dados1,aes(x =temperatura, y = previsao_consumo),
```

```
            colour = "red",size=3)+
 theme_minimal() +
labs(title = "Previsão de Consumo de Energia com Regressão Linear",
     x = "Temperatura Média Diária (°C)",
     y = "Consumo de Energia (MWh)")
```



No gráfico, o ponto vermelho representa a previsão de consumo de energia para a temperatura especificada.

4.2 Regressão Logística

A regressão logística é uma técnica estatística usada para modelar a probabilidade de ocorrência de um evento, categorizando o resultado em classes. Esta técnica é empregada para variáveis dependentes categóricas binárias, como "sim" ou "não", e "sucesso" ou "fracasso". Ela difere da regressão linear, que prevê valores contínuos, ao estimar a probabilidade de um evento ocorrer, baseando-se em um ou mais preditores. A regressão logística é particularmente útil em classificadores de aprendizado de máquina, sendo um componente chave dos modelos lineares generalizados(Zbicki and Santos 2020; James et al. 2023; Burger 2018).

A regressão logística, utilizada no contexto de classificadores, pode ser expressa matematicamente para uma classe binária da seguinte maneira:

$$Y_i = 1 \Rightarrow P(Y_i = 1) = \pi_i Y_i = 0 \Rightarrow P(Y_i = 0) = 1 - \pi_i$$

O modelo de regressão logistica é dado por:

$$\pi(X) = \frac{e^{\beta_0+\beta_1 X_1+\beta_2 X_2+...+\beta_m X_m}}{1 + e^{\beta_0+\beta_1 X_1+\beta_2 X_2+...+\beta_m X_m}}$$

em que X representa o conjunto de atributos ou variáveis de entrada.

Ao trabalhar com regressão logística, é crucial considerar aspectos importantes como:

- **Relação entre Variáveis**: Este método é eficaz quando há uma relação clara entre as variáveis independentes e a variável dependente binária. As variáveis independentes podem ser categóricas ou quantitativas, mas para variáveis quantitativas, é importante verificar se existe uma relação log-log.
- **Multicolinearidade**: É essencial evitar alta correlação entre as variáveis independentes, pois isso pode comprometer a interpretação dos coeficientes do modelo.
- **Avaliação do Modelo**: Para avaliar a precisão e eficácia do modelo, deve-se usar métricas apropriadas, como a área sob a curva ROC (AUC).

4.2.1 Exemplo Prático

Implementação no R

No R, a função `glm()` com a família binomial é comumente usada para realizar regressão logística.

Para ilustrar a utilização da regressão logistica, vamos utilizar o conjunto de dados **`PimaIndiansDiabetes2`** pacote mlbench (Leisch and Dimitriadou 2021). Este conjunto de dados contém informações de testes de diabetes coletadas de mulheres com pelo menos 21 anos, de herança indígena Pima e residentes próximas a Phoenix, Arizona, totalizando 768 observações em 9 variáveis

- pregnant: Número de vezes grávida.
- glucose: Concentração de glicose plasmática (teste de tolerância à glicose).
- pressure: Pressão arterial diastólica (mm Hg).

- triceps: Espessura da dobra da pele do tríceps (mm).
- insulin: Insulina sérica de 2 horas (mu U/ml).
- mass: Índice de massa corporal (peso em kg/(altura em m)^2).
- pedigree: Função de pedigree de diabetes.
- age: Idade (anos).
- diabetes: Fator indicando o resultado do teste de diabetes (neg/pos)

O conjunto de dados **PimaIndiansDiabetes2** contém informações incompletas para alguns indivíduos, ou seja, nem todas as variáveis foram observadas em todos os casos. Portanto, vamos optar por trabalhar apenas com os dados completos, o que reduz o conjunto a 392 observações. Essa abordagem nos permite realizar análises mais precisas e confiáveis, focando em dados onde todas as variáveis estão presentes.

Pacotes Necessários

```
library(tidyverse)
library(caret)
```

Prepararando os dados

```
#Ler os dados e remover os NA
data("PimaIndiansDiabetes2", package = "mlbench")
PimaIndiansDiabetes2 <- na.omit(PimaIndiansDiabetes2)
# Inspecionar os dados
sample_n(PimaIndiansDiabetes2, 3)
```

```
    pregnant glucose pressure triceps insulin mass pedigree age diabetes
225        1     100       66      15      56 23.6    0.666  26      neg
515        3      99       54      19      86 25.6    0.154  24      neg
646        2     157       74      35     440 39.4    0.134  30      neg
```

```
# Dividir o conjuntos de dados em treino e teste
set.seed(123)
training.samples <- PimaIndiansDiabetes2$diabetes %>%
  createDataPartition(p = 0.8, list = FALSE)
train.data  <- PimaIndiansDiabetes2[training.samples, ]
test.data <- PimaIndiansDiabetes2[-training.samples, ]
```

4.2.1.1 Regressão Logística Simples

A regressão logística simples é usada para prever a probabilidade de pertencer a uma classe com base em apenas uma variável preditora.

O seguinte código R constrói um modelo para prever a probabilidade de ser positivo para diabetes com base na concentração de glicose plasmática:

```
model <- glm( diabetes ~ glucose, data = train.data, family = binomial)
summary(model)
```

```
Call:
glm(formula = diabetes ~ glucose, family = binomial, data = train.data)

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -6.158820   0.700097  -8.797  < 2e-16 ***
glucose      0.043272   0.005341   8.102 5.42e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 398.8  on 313  degrees of freedom
Residual deviance: 305.7  on 312  degrees of freedom
AIC: 309.7

Number of Fisher Scoring iterations: 4
```

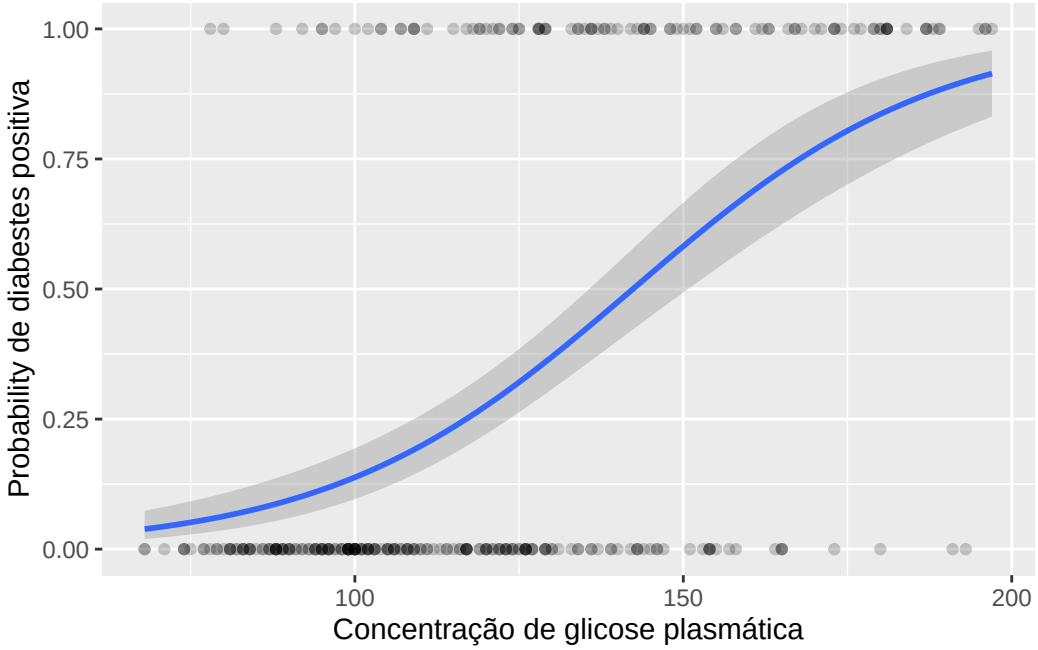
O resultado apresenta a estimativa dos coeficientes beta da regressão e seus níveis de significância. O intercepto ($\beta_0 = -6,15$) e o coeficiente da variável glicose $ _1= 0,043$.

A equação logística pode ser escrita como

$$\pi(X) = \frac{e^{-6,15+0,043Glucose}}{1 + e^{-6,15+0,043Glucose}}$$

Usando esta fórmula, para cada novo valor de concentração de glicose plasmática, é possível prever a probabilidade de os indivíduos serem positivos para diabetes.

```r
train.data %>%
  mutate(prob = ifelse(diabetes == "pos", 1, 0)) %>%
  ggplot(aes(glucose, prob)) +
  geom_point(alpha = 0.2) +
  geom_smooth(method = "glm", method.args = list(family = "binomial")) +
  labs(
    x = "Concentração de glicose plasmática",
    y = "Probability de diabestes positiva"
    )
```



Podemos avaliar a capacidade preditiva do modelo utilizando os dados que ele já conhece através do seguinte processo no R:

```r
# Predições
probabilities <- model %>%
                 predict(type = "response")
predicted.classes <- ifelse(probabilities > 0.5, "pos", "neg")
# Avaliando a acurácia
mean(predicted.classes == train.data$diabetes)
```

```
[1] 0.7611465
```

O resultado apresenta uma acurácia de 0,7611 para os dados de treinamento, , indica que o modelo é eficaz em classificar corretamente se um indivíduo tem ou não diabetes.

Podemos avaliar o modelo com os dados de teste, que são novos para o modelo ou seja desconhecidos.

```
# Predições
probabilities <- model %>%
                 predict(test.data, type = "response")
predicted.classes <- ifelse(probabilities > 0.5, "pos", "neg")
# Avaliando a acurácia
mean(predicted.classes == test.data$diabetes)
```

```
[1] 0.7692308
```

Ao aplicar o modelo aos dados de teste, que são novos para o modelo, obtemos uma acurácia de 0,7692. Isso sugere que o modelo mantém uma boa performance geral também para dados que não foram usados no treinamento, demonstrando sua eficiência e capacidade de generalização.

4.2.1.2 Regressão Logística Multipla

A regressão logística multipla é usada para prever a probabilidade de pertencer a uma classe com base em múltiplas variáveis preditoras.

O seguinte código R constrói um modelo para prever a probabilidade de ser positivo para diabetes com base na concentração de glicose plasmática, número de vezes grávida e índice de massa corporal :

```
model <- glm( diabetes ~ glucose+  pregnant+mass, data = train.data, family = binomial)
summary(model)
```

```
Call:
glm(formula = diabetes ~ glucose + pregnant + mass, family = binomial,
    data = train.data)
```

```
Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -9.323698   1.125997  -8.280  < 2e-16 ***
glucose      0.038862   0.005404   7.191 6.43e-13 ***
pregnant     0.144667   0.045126   3.206  0.00135 **
mass         0.094585   0.023530   4.020 5.83e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 398.80  on 313  degrees of freedom
Residual deviance: 279.88  on 310  degrees of freedom
AIC: 287.88

Number of Fisher Scoring iterations: 5
```

O resultado apresenta a estimativa dos coeficientes beta da regressão e seus níveis de significância. O intercepto ($\beta_0 = -9,32$), o coeficiente da variável glicose $ _1= 0,038$, o coeficiente da variável número de vezes grávida $ _2= 0,144$ e o coeficiente da variável índice de massa corporal $ _3= 0,094$

A equação logística pode ser escrita como

$$\pi(X) = \frac{e^{-9,32+0,038Glucose+0,144pregnant+0,094mass}}{1 + e^{-9,32+0,038Glucose+0,144pregnant+0,094mass}}$$

Avaliando a capacidade preditiva do com os dados de treino e teste:

```r
# Predições conjunto de treinamento
probabilities <- model %>%
       predict(type = "response")
predicted.classes <- ifelse(probabilities > 0.5,
                            "pos", "neg")
# Avaliando a acurácia
mean(predicted.classes == train.data$diabetes)
```

```
[1] 0.7866242
```

```r
#Predições
probabilities1 <- model %>%
  predict(test.data, type = "response")
predicted.classes1 <- ifelse(probabilities1 > 0.5, "pos", "neg")
# Avaliando a acurácia
mean(predicted.classes1 == test.data$diabetes)
```

```
[1] 0.7820513
```

O resultado, com uma acurácia de 0,7866 para os dados de treinamento e 0,7820 para os dados de teste, indica que o modelo de regressão logística múltipla tem uma boa capacidade de generalização. Além disso, a acurácia mais alta do modelo múltiplo em comparação com o modelo simples sugere uma melhoria na performance preditiva ao incluir múltiplas variáveis preditoras.

4.3 Árvores de Decisão

As árvores de decisão são um método gráfico e analítico que subdivide uma amostra inicial em subamostras, formando grupos onde a variável de resposta apresenta comportamento homogêneo internamente e heterogêneo entre eles. Este algoritmo de aprendizado de máquina supervisionado é aplicável tanto para classificação quanto para regressão, ou seja, pode prever tanto categorias discretas (como “sim” ou “não”) quanto valores numéricos (Singh and Allen 2016; Zbicki and Santos 2020):.

Funcionando de maneira semelhante a um fluxograma , as árvores de decisão têm nós de decisão interconectados hierarquicamente, incluindo um nó-raiz principal e nós-folha que representam os resultados finais. No machine learning, o nó-raiz corresponde a um atributo da base de dados, enquanto o nó-folha indica a classe ou valor a ser previsto (Singh and Allen 2016; Zbicki and Santos 2020).

Existem diversos algoritmos para a criação de árvores de decisão, sendo os mais comuns:

- **CHAID (Chi-square Automatic Interaction Detection)**: Este algoritmo é mais comumente usado para tarefas de classificação.Utiliza tabelas de contingência para identificar as melhores divisões.
- **CART (Classification and Regression Trees):** Um dos algoritmos mais versáteis, o CART é utilizado tanto para regressão quanto para classificação. Sua abordagem binária para dividir os nós permite uma ampla gama de aplicações.

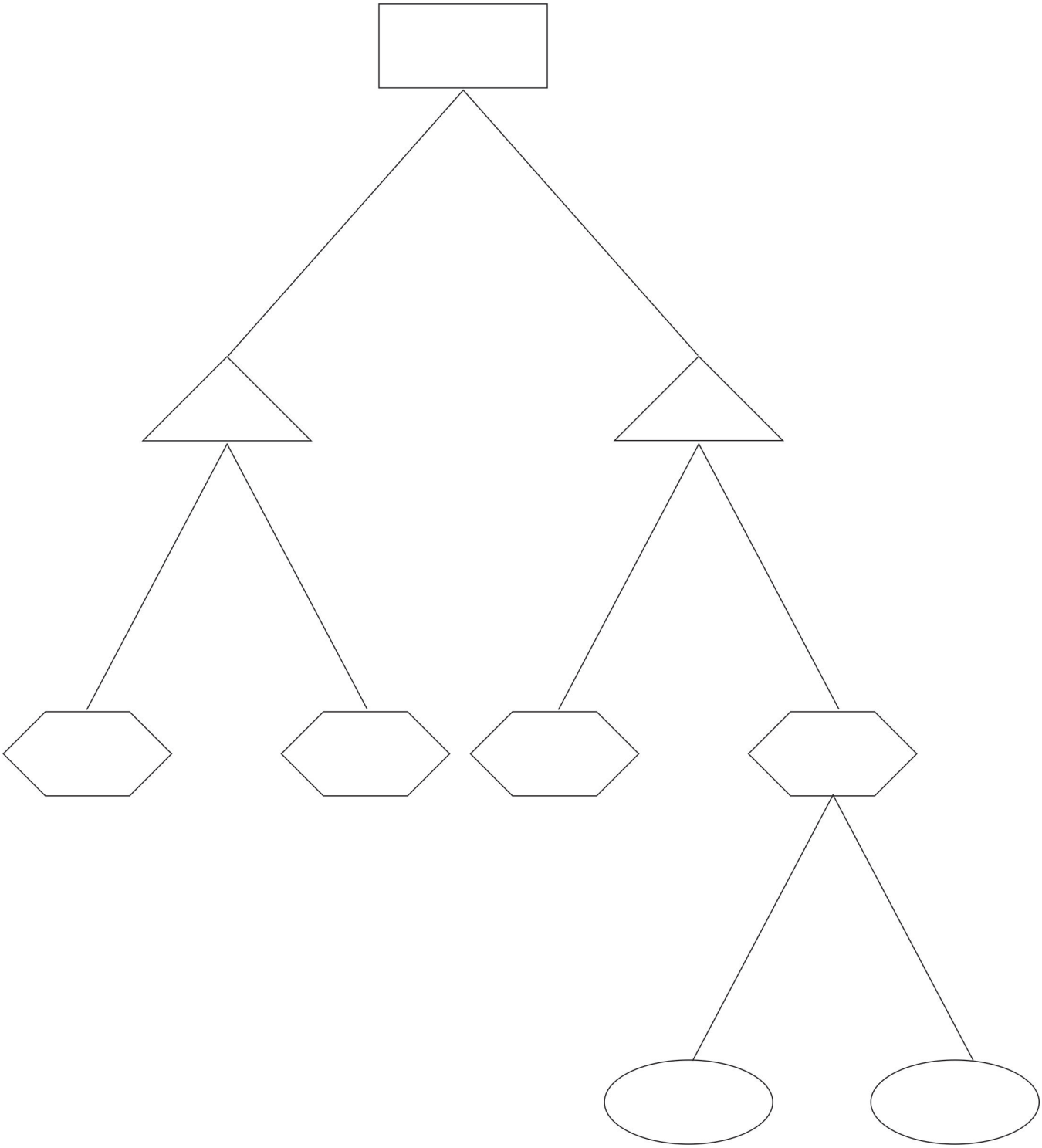


Figure 4.1: Exemplo de um esquema de arvore de decisão

- **ID3 (Iteractive Dichotomizer 3):** Geralmente aplicado em tarefas de classificação, mas existem versões para regressão, o ID3 seleciona atributos com base no Ganho de Informação, escolhendo aqueles que mais reduzem a incerteza no conjunto de dados.
- **C4.5**: Uma evolução do ID3, o C4.5 inclui melhorias como o tratamento de dados contínuos e valores ausentes, mantendo a abordagem baseada em Ganho de Informação.

Árvores de decisão são particularmente úteis quando se deseja trabalhar com dados sem a necessidade de um tratamento extensivo. Elas lidam bem com valores atípicos e dados faltantes, reduzindo a necessidade de etapas de tratamento intensivo. Além disso, não é necessário converter dados categóricos para numéricos, pois este algoritmo lida eficientemente com informações nominais. Em situações que envolvem problemas tanto de classificação quanto de regressão, as árvores de decisão oferecem flexibilidade e eficácia, tornando-se uma escolha adequada para uma variedade de cenários analíticos.

4.3.1 Exemplo Prático

Implementação no R

Para ilustrar a utilização arvore de decisão, vamos utilizar o conjunto de dados `PimaIndiansDiabetes2` pacote mlbench (Leisch and Dimitriadou 2021), apresentado na seção anterior.

Pacotes Necessários

```
library(tidyverse)
library(caret)
library(rpart)
library(rpart.plot)
```

Prepararando os dados,

```
#Ler os dados e remover os NA
data("PimaIndiansDiabetes2", package = "mlbench")
PimaIndiansDiabetes2 <- na.omit(PimaIndiansDiabetes2)
# Inspecionar os dados
sample_n(PimaIndiansDiabetes2, 3)
```

```
    pregnant glucose pressure triceps insulin mass pedigree age diabetes
221        0     177       60      29     478 34.6    1.072  21      pos
72         5     139       64      35     140 28.6    0.411  26      neg
```

```
172        6     134       70       23     130 35.4    0.542  29      pos
```

```
# Dividir o conjuntos de dados em treino e teste
set.seed(123)
training.samples <- PimaIndiansDiabetes2$diabetes %>%
  createDataPartition(p = 0.8, list = FALSE)
train.data  <- PimaIndiansDiabetes2[training.samples, ]
test.data <- PimaIndiansDiabetes2[-training.samples, ]
```

O seguinte código R constrói um modelo de árvore de decisão para prever se um indivíduo é positivo para diabetes com base em todas as variáveis preditoras disponíveis no conjunto de dados. Isso é realizado utilizando o operador ~ para incluir todas as variáveis preditoras:

```
tree_model <- rpart(
  diabetes ~.,  data = train.data,   method = "class"
)
summary(tree_model)
```

```
Call:
rpart(formula = diabetes ~ ., data = train.data, method = "class")
  n= 314

          CP nsplit rel error    xerror       xstd
1 0.30769231      0 1.0000000 1.0000000 0.08019147
2 0.05769231      1 0.6923077 0.7115385 0.07231416
3 0.04166667      3 0.5769231 0.7211538 0.07264930
4 0.02884615      6 0.4519231 0.7692308 0.07424287
5 0.01000000      8 0.3942308 0.8173077 0.07570578

Variable importance
 glucose      age  insulin pressure pregnant  triceps     mass pedigree
      37       19       17       11        7        5        2        1

Node number 1: 314 observations,    complexity param=0.3076923
  predicted class=neg  expected loss=0.3312102  P(node) =1
    class counts:   210   104
   probabilities: 0.669 0.331
  left son=2 (198 obs) right son=3 (116 obs)
  Primary splits:
```

```
      glucose  < 127.5  to the left,  improve=34.61298, (0 missing)
      insulin  < 126.5  to the left,  improve=23.56122, (0 missing)
      age      < 28.5   to the left,  improve=20.18536, (0 missing)
      mass     < 34.05  to the left,  improve=11.06798, (0 missing)
      pregnant < 6.5    to the left,  improve=10.83097, (0 missing)
  Surrogate splits:
      insulin  < 125.5  to the left,  agree=0.755, adj=0.336, (0 split)
      age      < 34.5   to the left,  agree=0.694, adj=0.172, (0 split)
      pregnant < 5.5    to the left,  agree=0.672, adj=0.112, (0 split)
      pressure < 81     to the left,  agree=0.672, adj=0.112, (0 split)
      mass     < 45.95  to the left,  agree=0.640, adj=0.026, (0 split)

Node number 2: 198 observations,    complexity param=0.02884615
  predicted class=neg  expected loss=0.1515152  P(node) =0.6305732
    class counts:   168    30
   probabilities: 0.848 0.152
  left son=4 (139 obs) right son=5 (59 obs)
  Primary splits:
      age      < 29.5   to the left,  improve=4.887386, (0 missing)
      insulin  < 143.5  to the left,  improve=4.058442, (0 missing)
      mass     < 45.4   to the left,  improve=3.738038, (0 missing)
      pedigree < 0.6775 to the left,  improve=3.112899, (0 missing)
      glucose  < 103.5  to the left,  improve=2.719190, (0 missing)
  Surrogate splits:
      pregnant < 4.5    to the left,  agree=0.854, adj=0.508, (0 split)
      mass     < 45.7   to the left,  agree=0.722, adj=0.068, (0 split)
      pedigree < 0.9215 to the left,  agree=0.722, adj=0.068, (0 split)
      glucose  < 119.5  to the left,  agree=0.717, adj=0.051, (0 split)
      insulin  < 173    to the left,  agree=0.717, adj=0.051, (0 split)

Node number 3: 116 observations,    complexity param=0.05769231
  predicted class=pos  expected loss=0.362069  P(node) =0.3694268
    class counts:    42    74
   probabilities: 0.362 0.638
  left son=6 (80 obs) right son=7 (36 obs)
  Primary splits:
      glucose  < 165.5  to the left,  improve=6.575096, (0 missing)
      mass     < 29.5   to the left,  improve=6.442471, (0 missing)
      triceps  < 32.5   to the left,  improve=5.826207, (0 missing)
      age      < 24.5   to the left,  improve=4.923350, (0 missing)
      pregnant < 6.5    to the left,  improve=2.073976, (0 missing)
```

```
  Surrogate splits:
      age      < 52     to the left,  agree=0.716, adj=0.083, (0 split)
      insulin  < 452.5  to the left,  agree=0.707, adj=0.056, (0 split)
      pressure < 104    to the left,  agree=0.698, adj=0.028, (0 split)
      pedigree < 1.764  to the left,  agree=0.698, adj=0.028, (0 split)

Node number 4: 139 observations
  predicted class=neg  expected loss=0.07913669  P(node) =0.4426752
    class counts:   128    11
   probabilities: 0.921 0.079

Node number 5: 59 observations,    complexity param=0.02884615
  predicted class=neg  expected loss=0.3220339  P(node) =0.1878981
    class counts:    40    19
   probabilities: 0.678 0.322
  left son=10 (37 obs) right son=11 (22 obs)
  Primary splits:
      insulin  < 142.5  to the left,  improve=6.932245, (0 missing)
      glucose  < 108.5  to the left,  improve=3.843730, (0 missing)
      pedigree < 0.514  to the left,  improve=2.531230, (0 missing)
      mass     < 26.5   to the left,  improve=1.919575, (0 missing)
      pregnant < 1.5    to the right, improve=1.562043, (0 missing)
  Surrogate splits:
      triceps  < 45.5   to the left,  agree=0.695, adj=0.182, (0 split)
      age      < 50.5   to the left,  agree=0.695, adj=0.182, (0 split)
      pregnant < 1.5    to the right, agree=0.678, adj=0.136, (0 split)
      pedigree < 1.153  to the left,  agree=0.661, adj=0.091, (0 split)
      glucose  < 116    to the left,  agree=0.644, adj=0.045, (0 split)

Node number 6: 80 observations,    complexity param=0.05769231
  predicted class=pos  expected loss=0.475  P(node) =0.2547771
    class counts:    38    42
   probabilities: 0.475 0.525
  left son=12 (14 obs) right son=13 (66 obs)
  Primary splits:
      age      < 23.5   to the left,  improve=6.982251, (0 missing)
      triceps  < 22.5   to the left,  improve=5.664103, (0 missing)
      mass     < 30.2   to the left,  improve=5.379624, (0 missing)
      pregnant < 7.5    to the left,  improve=2.236497, (0 missing)
      pressure < 77     to the left,  improve=1.761893, (0 missing)
  Surrogate splits:
```

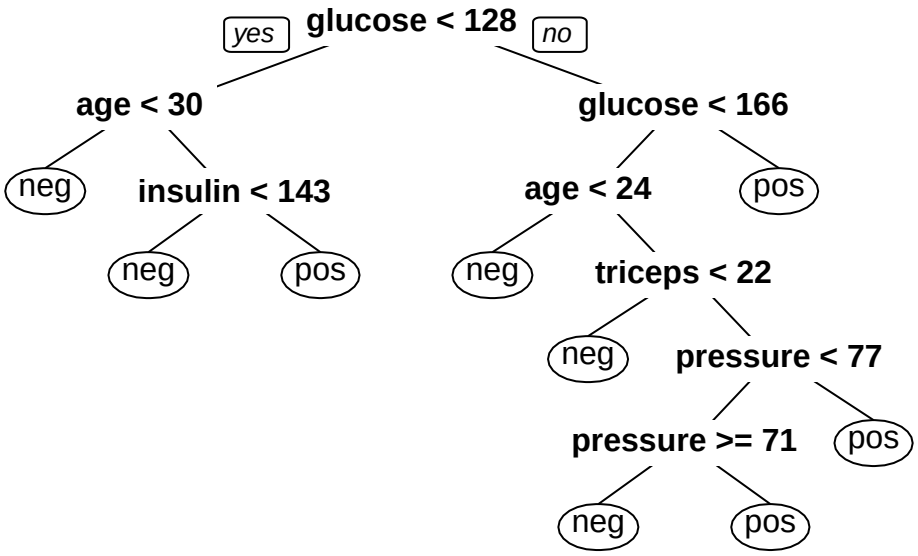
```
      mass < 25.15  to the left,  agree=0.85, adj=0.143, (0 split)

Node number 7: 36 observations
  predicted class=pos  expected loss=0.1111111  P(node) =0.1146497
    class counts:     4    32
   probabilities: 0.111 0.889

Node number 10: 37 observations
  predicted class=neg  expected loss=0.1351351  P(node) =0.1178344
    class counts:    32     5
   probabilities: 0.865 0.135

Node number 11: 22 observations
  predicted class=pos  expected loss=0.3636364  P(node) =0.07006369
    class counts:     8    14
   probabilities: 0.364 0.636

Node number 12: 14 observations
  predicted class=neg  expected loss=0.07142857  P(node) =0.04458599
    class counts:    13     1
   probabilities: 0.929 0.071

Node number 13: 66 observations,    complexity param=0.04166667
  predicted class=pos  expected loss=0.3787879  P(node) =0.2101911
    class counts:    25    41
   probabilities: 0.379 0.621
  left son=26 (8 obs) right son=27 (58 obs)
  Primary splits:
      triceps  < 22     to the left,  improve=2.508882, (0 missing)
      mass     < 33.95  to the left,  improve=2.205307, (0 missing)
      pedigree < 0.7115 to the left,  improve=1.978188, (0 missing)
      pressure < 77     to the left,  improve=1.382828, (0 missing)
      pregnant < 1.5    to the right, improve=1.240998, (0 missing)
  Surrogate splits:
      mass < 23.85  to the left,  agree=0.909, adj=0.25, (0 split)

Node number 26: 8 observations
  predicted class=neg  expected loss=0.25  P(node) =0.02547771
    class counts:     6     2
   probabilities: 0.750 0.250
```

```
Node number 27: 58 observations,    complexity param=0.04166667
  predicted class=pos  expected loss=0.3275862  P(node) =0.1847134
    class counts:    19    39
   probabilities: 0.328 0.672
  left son=54 (31 obs) right son=55 (27 obs)
  Primary splits:
      pressure < 77     to the left,  improve=2.0487370, (0 missing)
      glucose  < 145.5  to the right, improve=2.0231530, (0 missing)
      mass     < 40.75  to the left,  improve=1.2517240, (0 missing)
      pedigree < 0.7115 to the left,  improve=1.0115530, (0 missing)
      age      < 41     to the left,  improve=0.9256372, (0 missing)
  Surrogate splits:
      triceps  < 32.5   to the left,  agree=0.672, adj=0.296, (0 split)
      age      < 31     to the left,  agree=0.672, adj=0.296, (0 split)
      pregnant < 4.5    to the left,  agree=0.638, adj=0.222, (0 split)
      pedigree < 0.866  to the left,  agree=0.638, adj=0.222, (0 split)
      insulin  < 105    to the right, agree=0.603, adj=0.148, (0 split)

Node number 54: 31 observations,    complexity param=0.04166667
  predicted class=pos  expected loss=0.4516129  P(node) =0.09872611
    class counts:    14    17
   probabilities: 0.452 0.548
  left son=108 (15 obs) right son=109 (16 obs)
  Primary splits:
      pressure < 71     to the right, improve=7.0548390, (0 missing)
      mass     < 39.95  to the left,  improve=1.7238860, (0 missing)
      pregnant < 1.5    to the right, improve=0.8765778, (0 missing)
      age      < 25.5   to the right, improve=0.8765778, (0 missing)
      insulin  < 156    to the right, improve=0.8131720, (0 missing)
  Surrogate splits:
      triceps  < 31.5   to the right, agree=0.613, adj=0.200, (0 split)
      age      < 29.5   to the right, agree=0.613, adj=0.200, (0 split)
      pregnant < 3.5    to the right, agree=0.581, adj=0.133, (0 split)
      glucose  < 161.5  to the right, agree=0.581, adj=0.133, (0 split)
      insulin  < 105    to the left,  agree=0.581, adj=0.133, (0 split)

Node number 55: 27 observations
  predicted class=pos  expected loss=0.1851852  P(node) =0.08598726
    class counts:     5    22
   probabilities: 0.185 0.815
```

```
Node number 108: 15 observations
  predicted class=neg  expected loss=0.2  P(node) =0.0477707
    class counts:    12     3
   probabilities: 0.800 0.200

Node number 109: 16 observations
  predicted class=pos  expected loss=0.125  P(node) =0.05095541
    class counts:     2    14
   probabilities: 0.125 0.875
```

Este código produz um resumo detalhado do modelo de árvore de decisão, que pode ser complexo de analisar. Para facilitar a visualização, podemos representar graficamente a árvore construída:

```
prp(tree_model)
```

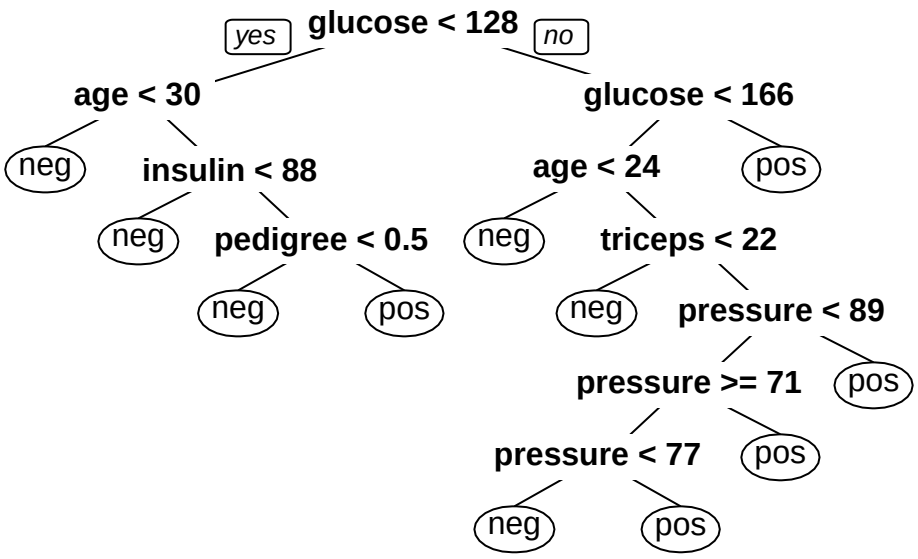


Este passo permite visualizar a estrutura da árvore de decisão de forma mais intuitiva e compreensível.

Por padrão, o **`rpart`** usa a impureza de Gini para selecionar divisões ao realizar classificação.

(Se você não está familiarizado, leia este artigo.) Você pode usar o ganho de informação em vez disso, especificando-o no parâmetro **parms**.

```
tree_model1 <- rpart(diabetes ~.,data = train.data,method = "class",
  parms = list(split = 'information')
)

prp(tree_model1)
```



Podemos avaliar a capacidade preditiva do com os dados de treino e teste, utilizando os dados que ele já conhece através do seguinte processo no R:

```
# Predições de treinamento
class <- predict(tree_model1,type = 'class')
confusionMatrix(class, train.data$diabetes,positive='pos')
```

```
Confusion Matrix and Statistics

          Reference
```

```
Prediction neg pos
       neg 194  23
       pos  16  81

               Accuracy : 0.8758
                 95% CI : (0.8341, 0.9102)
    No Information Rate : 0.6688
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.7148

 Mcnemar's Test P-Value : 0.3367

            Sensitivity : 0.7788
            Specificity : 0.9238
         Pos Pred Value : 0.8351
         Neg Pred Value : 0.8940
             Prevalence : 0.3312
         Detection Rate : 0.2580
   Detection Prevalence : 0.3089
      Balanced Accuracy : 0.8513

       'Positive' Class : pos
```

```r
# Predições de teste
class1 <- predict(tree_model1,test.data,type = 'class')
confusionMatrix(class1, test.data$diabetes,positive='pos')
```

```
Confusion Matrix and Statistics

          Reference
Prediction neg pos
       neg  41   9
       pos  11  17

               Accuracy : 0.7436
                 95% CI : (0.6321, 0.8358)
    No Information Rate : 0.6667
    P-Value [Acc > NIR] : 0.09127
```

```
                 Kappa : 0.434

 Mcnemar's Test P-Value : 0.82306

            Sensitivity : 0.6538
            Specificity : 0.7885
         Pos Pred Value : 0.6071
         Neg Pred Value : 0.8200
             Prevalence : 0.3333
         Detection Rate : 0.2179
   Detection Prevalence : 0.3590
      Balanced Accuracy : 0.7212

       'Positive' Class : pos
```

O resultado, com uma acurácia de 0,8758para os dados de treinamento e 0,7436para os dados de teste, indica que o modelo de regressão logística múltipla tem uma boa capacidade de generalização. Além disso, outras medidas de performance do modelo apresentam valores elevados.

4.4 Floresta Aleatória

Floresta Aleatória é um algoritmo de aprendizado supervisionado que cria uma "floresta" de forma aleatória. Essa floresta é na verdade um conjunto de árvores de decisão, geralmente treinadas com o método de bagging. A ideia por trás do bagging é que a combinação de vários modelos de aprendizado melhora o desempenho geral.

As Florestas Aleatórias funcionam ao criar numerosas árvores de decisão aleatoriamente, cada uma contribuindo para a decisão final. Uma grande vantagem desse algoritmo é sua aplicabilidade tanto em tarefas de classificação quanto de regressão, sendo muito relevante nos sistemas de aprendizado de máquina atuais. No contexto de classificação, as Florestas Aleatórias são consideradas um dos pilares do aprendizado de máquina. Um exemplo clássico de Floresta Aleatória pode incluir diversas árvores, cada uma contribuindo para a classificação ou previsão final.

Diferenças entre Árvore de Decisão e Florestas Aleatórias

Floresta Aleatória e Árvore de Decisão são métodos de aprendizado de máquina, mas com diferenças significativas. Enquanto a Árvore de Decisão utiliza regras e nodos baseados em

cálculos como ganho de informação e índice de Gini, a Floresta Aleatória opera de maneira aleatória e é uma coleção de várias árvores. Uma Árvore de Decisão única pode sofrer de sobreajuste, especialmente se for muito profunda. Em contraste, as Florestas Aleatórias minimizam o sobreajuste ao construir várias árvores menores a partir de subconjuntos aleatórios de características, combinando-as posteriormente. Este processo pode tornar as Florestas Aleatórias mais lentas, dependendo do número de árvores construídas.

Algoritmo para Florestas Aleatórias

O algoritmo das Florestas Aleatórias (Random Forest) funciona da seguinte maneira:

- Gerar B amostras bootstrap com reposição do conjunto de dados original.
- Para cada amostra bootstrap, criar uma árvore de decisão:
  - Em cada nó, é sorteado M atributos dentre os quais a divisão será realizada.
  - A árvore é construída sem ser podada.
- Cada árvore gera um resultado, e a classificação/regressão final é determinada pelo resultado mais frequente entre todas as árvores.

4.4.1 Exemplo Prático

Implementação no R

Para ilustrar a utilização arvore de decisão, vamos utilizar o conjunto de dados **`PimaIndiansDiabetes2`** pacote mlbench (Leisch and Dimitriadou 2021), apresentado na seção anterior.

Pacotes Necessários

```
library(tidyverse)
library(caret)
library(randomForest)
```

Prepararando os dados,

```
#Ler os dados e remover os NA
data("PimaIndiansDiabetes2", package = "mlbench")
PimaIndiansDiabetes2 <- na.omit(PimaIndiansDiabetes2)
# Inspecionar os dados
sample_n(PimaIndiansDiabetes2, 3)
```

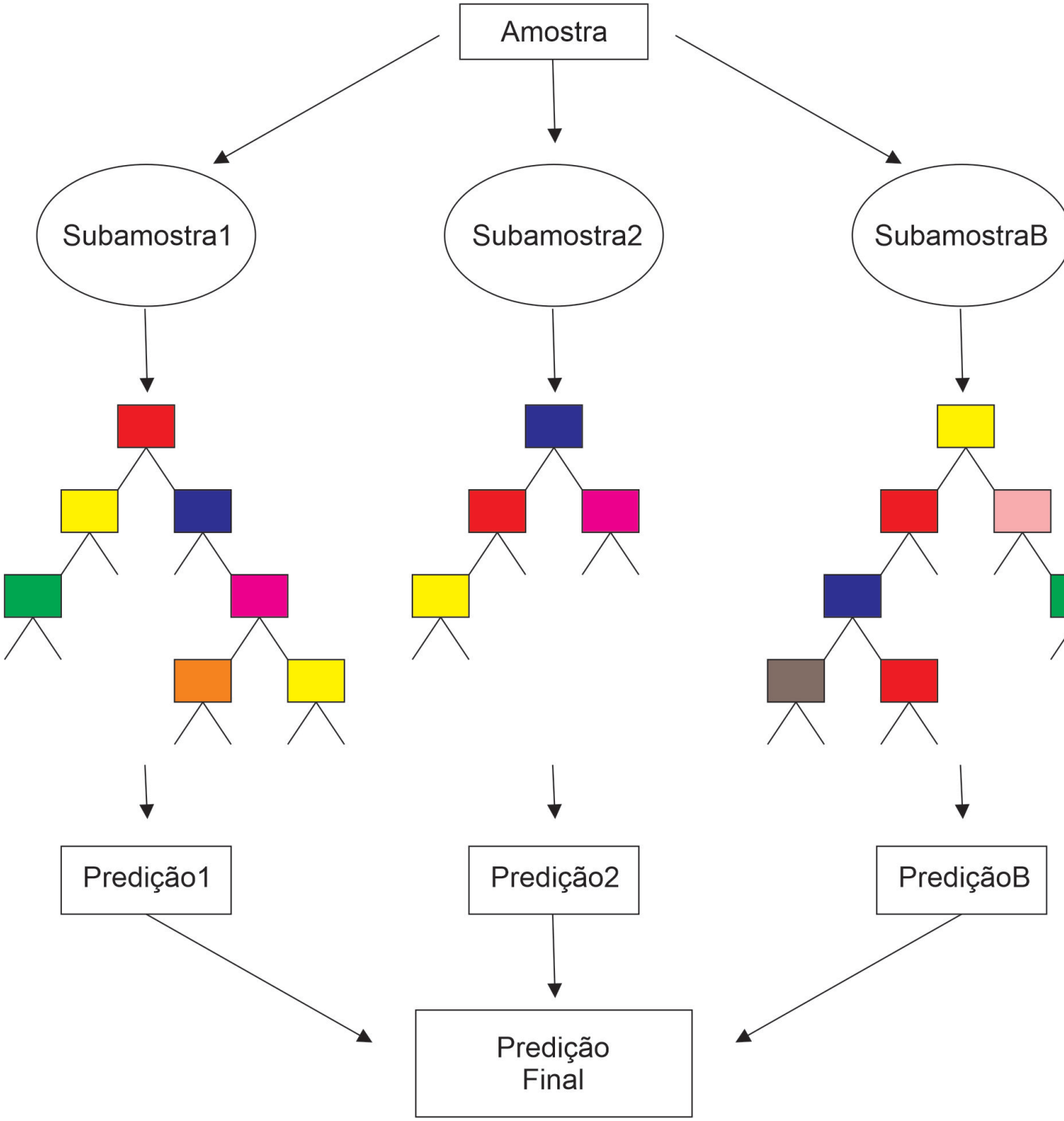


Figure 4.2: Exemplo de floresta aleatória

```
    pregnant glucose pressure triceps insulin mass pedigree age diabetes
53         5      88       66      21      23 24.4    0.342  30      neg
423        0     102       64      46      78 40.6    0.496  21      neg
541        8     100       74      40     215 39.4    0.661  43      pos
```

```
# Dividir o conjuntos de dados em treino e teste
set.seed(123)
training.samples <- PimaIndiansDiabetes2$diabetes %>%
  createDataPartition(p = 0.8, list = FALSE)
train.data  <- PimaIndiansDiabetes2[training.samples, ]
test.data <- PimaIndiansDiabetes2[-training.samples, ]
```

O seguinte código R constrói um modelo de árvore de decisão para prever se um indivíduo é positivo para diabetes com base em todas as variáveis preditoras disponíveis no conjunto de dados. Isso é realizado utilizando o operador ~ para incluir todas as variáveis preditoras:

```
rf_model <- randomForest( diabetes ~.,  data = train.data)
print(rf_model)
```

```
Call:
 randomForest(formula = diabetes ~ ., data = train.data)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 2

        OOB estimate of  error rate: 23.57%
Confusion matrix:
    neg pos class.error
neg 180  30   0.1428571
pos  44  60   0.4230769
```

Podemos avaliar a capacidade preditiva do com os dados de treino e teste, utilizando os dados que ele já conhece através do seguinte processo no R:

```
#Conjunto de treino
class <- predict(rf_model)
confusionMatrix(class, train.data$diabetes,positive='pos')
```

```
Confusion Matrix and Statistics

          Reference
Prediction neg pos
       neg 180  44
       pos  30  60

               Accuracy : 0.7643
                 95% CI : (0.7134, 0.8102)
    No Information Rate : 0.6688
    P-Value [Acc > NIR] : 0.00014

                  Kappa : 0.4493

 Mcnemar's Test P-Value : 0.13073

            Sensitivity : 0.5769
            Specificity : 0.8571
         Pos Pred Value : 0.6667
         Neg Pred Value : 0.8036
             Prevalence : 0.3312
         Detection Rate : 0.1911
   Detection Prevalence : 0.2866
      Balanced Accuracy : 0.7170

       'Positive' Class : pos
```

```r
#Conjunto de teste
class1 <- predict(rf_model,test.data)
confusionMatrix(class1,test.data$diabetes,positive='pos')
```

```
Confusion Matrix and Statistics

          Reference
Prediction neg pos
       neg  42   8
       pos  10  18

               Accuracy : 0.7692
```

```
               95% CI : (0.66, 0.8571)
    No Information Rate : 0.6667
    P-Value [Acc > NIR] : 0.03295

                  Kappa : 0.4906

 Mcnemar's Test P-Value : 0.81366

            Sensitivity : 0.6923
            Specificity : 0.8077
         Pos Pred Value : 0.6429
         Neg Pred Value : 0.8400
             Prevalence : 0.3333
         Detection Rate : 0.2308
   Detection Prevalence : 0.3590
      Balanced Accuracy : 0.7500

       'Positive' Class : pos
```

O resultado, com uma acurácia de 0,8758 para os dados de treinamento e 0,7436para os dados de teste, indica que o modelo de regressão logística múltipla tem uma boa capacidade de generalização. Além disso, outras medidas de performance do modelo apresentam valores elevados.

4.5 Introdução ao CARET

O pacote **caret** (Kuhn and Max 2008) (**C**lassification **A**nd **RE**gression **T**raining) no R é uma ferramenta valiosa para simplificar o treinamento de modelos em problemas complexos de regressão e classificação. Este pacote integra diversos outros pacotes do R, mas é projetado para carregá-los conforme a necessidade, evitando o carregamento de todos eles na inicialização. Isso reduz significativamente o tempo de inicialização do pacote e melhora a eficiência. O **caret** assume que os pacotes necessários estão instalados e, caso algum pacote de modelagem esteja faltando, ele notifica o usuário para instalá-lo. Sua popularidade se deve à sua capacidade de simplificar as etapas de treinamento e teste de modelos preditivos. Para exemplos detalhados de como utilizar o **caret**, pode-se visitar a página oficial: Caret Package.

O pacote **caret** no R oferece uma ampla gama de ferramentas para a preparação de dados, essencial para o sucesso dos modelos de machine learning. Ele facilita a normalização e padronização de dados, o que é crucial para métodos que são sensíveis à escala das variáveis.

Além disso, o **caret** pode tratar dados faltantes, realizar a binarização de variáveis categóricas e a seleção de variáveis, ajudando a melhorar a eficiência e a eficácia dos modelos. Esses recursos tornam o **caret** uma escolha excelente para o preprocessamento de dados antes da aplicação de técnicas de aprendizado de máquina.

O **caret** proporciona uma interface unificada para uma ampla gama de modelos de machine learning, permitindo aos usuários aplicar diversos métodos e técnicas com uma sintaxe consistente. Isso inclui desde modelos lineares até técnicas avançadas de ensemble.

4.5.1 Treinamento de modelos

Para ilustrar o treinamento de modelos no pacote **caret** , vamos utilizar o conjunto de dados **PimaIndiansDiabetes2** pacote **mlbench** (Leisch and Dimitriadou 2021), apresentado na seção anterior e vamos obter um modelo de arvore de decisão.

Uma das funções centrais do pacote **caret** é a **train()**, que é essencial para a construção de modelos de machine learning. Esta função automatiza o processo de treinamento, incorporando técnicas como validação cruzada e otimização de parâmetros. Ao usar a **train()**, o usuário pode aplicar diversos algoritmos aos dados, resultando em modelos bem treinados e prontos para serem testados e usados em previsões. A função **train** também facilita a comparação do desempenho de diferentes algoritmos, tornando-a uma ferramenta valiosa para a seleção de modelos adequados.

Pacotes Necessários

```
library(tidyverse)
library(caret)
library(rpart.plot)
```

Prepararando os dados,

```
#Ler os dados e remover os NA
data("PimaIndiansDiabetes2", package = "mlbench")
PimaIndiansDiabetes2 <- na.omit(PimaIndiansDiabetes2)
# Inspecionar os dados
sample_n(PimaIndiansDiabetes2, 3)
```

```
    pregnant glucose pressure triceps insulin mass pedigree age diabetes
17         0     118       84      47     230 45.8    0.551  31      pos
127        3     120       70      30     135 42.9    0.452  30      neg
349        3      99       62      19      74 21.8    0.279  26      neg
```

```
# Dividir o conjuntos de dados em treino e teste
set.seed(123)
training.samples <- PimaIndiansDiabetes2$diabetes %>%
  createDataPartition(p = 0.8, list = FALSE)
train.data  <- PimaIndiansDiabetes2[training.samples, ]
test.data <- PimaIndiansDiabetes2[-training.samples, ]
```

Treinar o modelo

```
tree_model <- train(diabetes ~.,  data = train.data,   method =  "rpart")

#Resultados do modelo
tree_model$results
```

```
          cp  Accuracy     Kappa AccuracySD    KappaSD
1 0.04166667 0.7484020 0.4120695 0.04108556 0.09521895
2 0.05769231 0.7461157 0.4013915 0.03684616 0.09119484
3 0.30769231 0.7192240 0.2975358 0.05061580 0.21417929
```

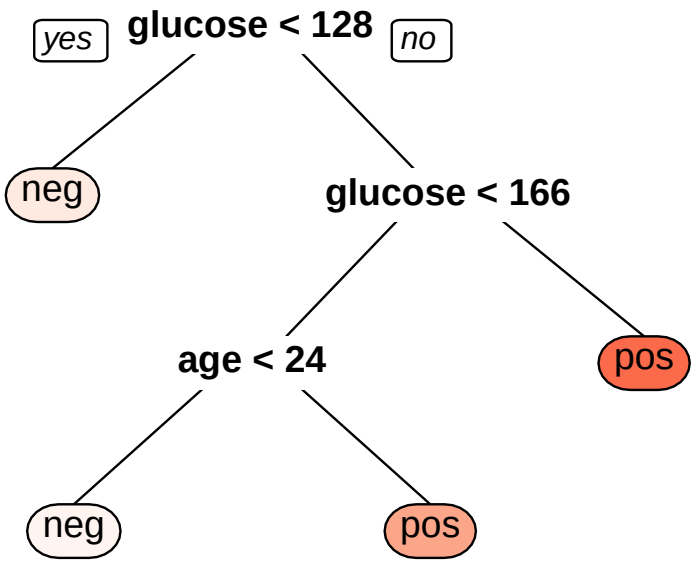
```
tree_model$bestTune
```

```
          cp
1 0.04166667
```

```
tree_model$metric
```

```
[1] "Accuracy"
```

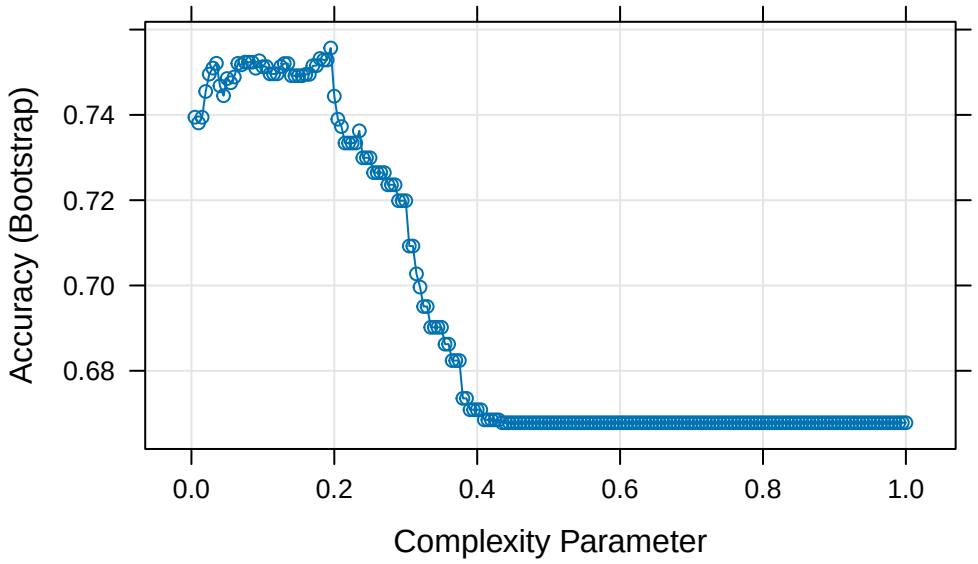
```
#Plotar a arvore obtida
m=tree_model$finalModel
prp(m, box.palette = "Reds", tweak = 1.2)
```



O **tuneGrid** é uma funcionalidade importante do pacote **caret** no R, utilizada para otimizar os hiperparâmetros de modelos de machine learning. Com o **tuneGrid**, os usuários podem especificar uma grade de hiperparâmetros que o **caret** irá explorar durante o treinamento do modelo. Isso permite a identificação da combinação de parâmetros que resulta no melhor desempenho do modelo, otimizando assim a precisão das previsões. Esta ferramenta é essencial para refinar modelos e garantir que eles estejam operando em sua capacidade máxima.

No caso de árvores de decisão, um hiperparâmetro importante é o parâmetro de complexidade (cp), que determina a poda da árvore. Este parâmetro ajuda a controlar o tamanho da árvore e a evitar overfitting. Ao utilizar o **tuneGrid** no **caret** para uma árvore de decisão, você pode especificar diferentes valores de cp para encontrar o que proporciona o melhor equilíbrio entre a complexidade da árvore e a capacidade de generalização do modelo.

```
hyper=expand.grid(
  cp = seq(0.005, 1.0, 0.005) #parametros de complexidade de 0.005 até 1.0
)
tree_model <- train(diabetes ~.,  data = train.data,   method =  "rpart",tuneGrid=hyper)

##Visuzalizar as acuráciass em função do parametro de complexidade
plot(tree_model)
```



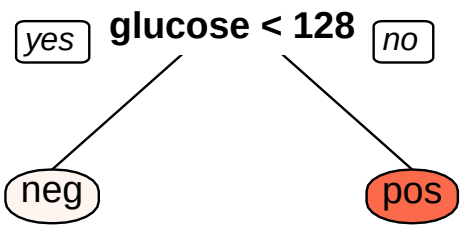

```
#Resultados do modelo
tree_model$bestTune
```

```
    cp
39 0.195
```

```
tree_model$metric
```

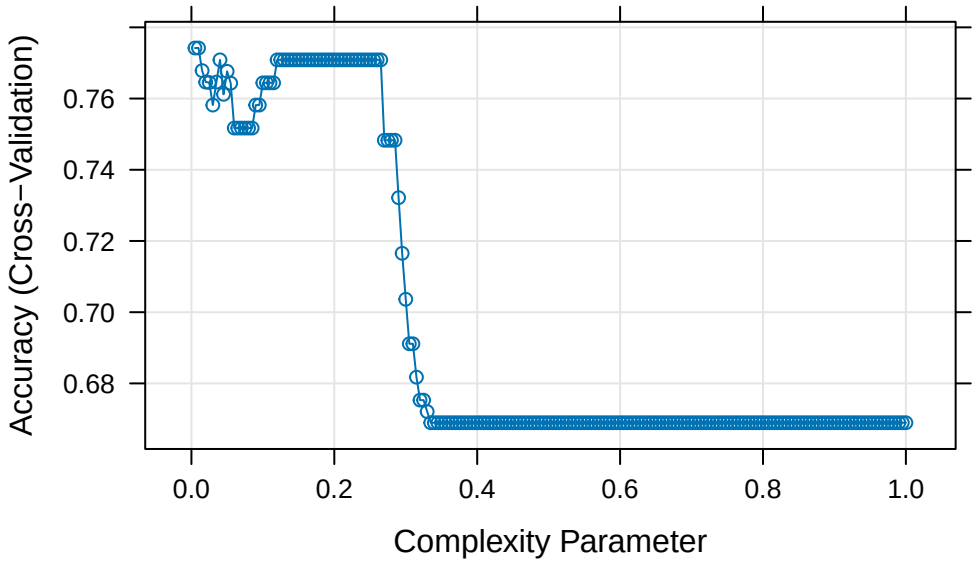
```
[1] "Accuracy"
```

```
#Plotar a arvore obtida
m=tree_model$finalModel
prp(m, box.palette = "Reds", tweak = 1.2)
```



Para a validação de modelos no **caret**, métodos como a validação cruzada k-fold e bootstrap são disponibilizados através do argumento **trControl** na função **train**. Essas técnicas são essenciais para evitar o overfitting e avaliar a capacidade de generalização do modelo. A validação cruzada k-fold divide o conjunto de dados em k partes, treinando o modelo em k-1 partes e testando-o na parte restante. O processo é repetido para cada parte. Já o método bootstrap utiliza amostragem com reposição para criar conjuntos de treino e teste, fornecendo uma avaliação robusta do modelo.

```
#Definir o hiperametros
hyper=expand.grid(
  cp = seq(0.005, 1.0, 0.005) # complexidade de 0.005 até 1.0
)
#VAlidação K-fold
ctrl=trainControl(method="cv",number=10)

tree_model <- train(diabetes ~.,  data = train.data,   method =  "rpart",tuneGrid=hyper,tr

##Visuzalizar as acuráciass em função do parametro de complexidade
plot(tree_model)
```



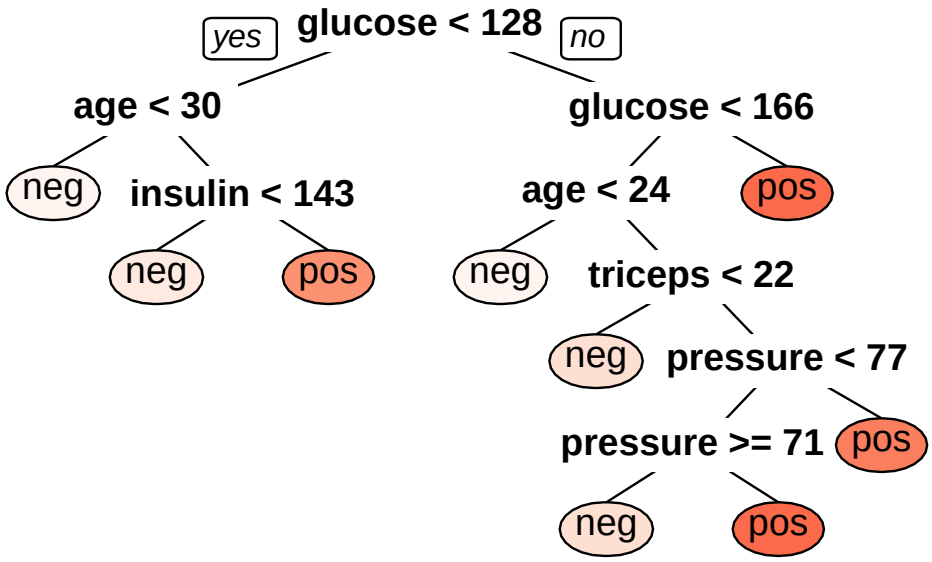

```
#Resultados do modelo
tree_model$bestTune
```

```
   cp
2 0.01
```

```
tree_model$metric
```

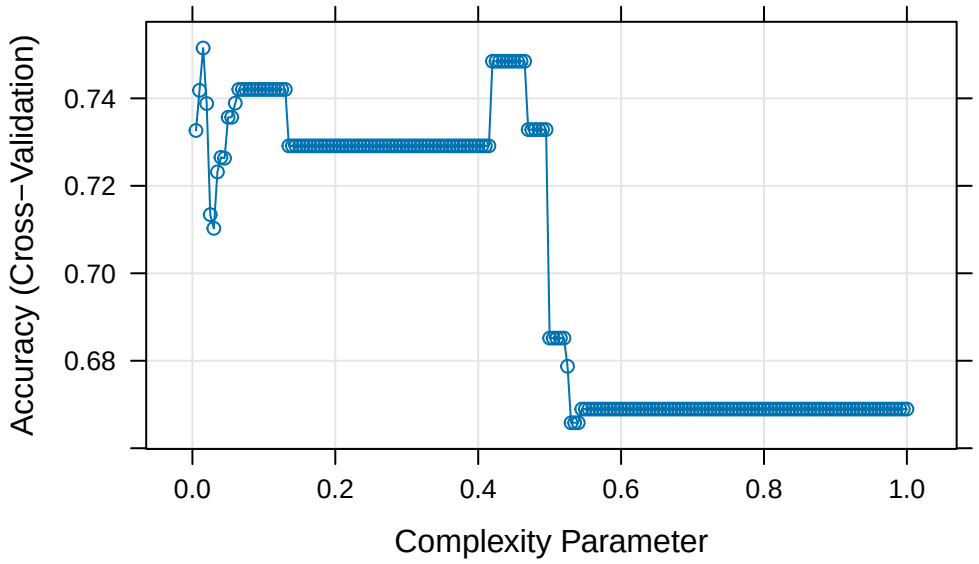
```
[1] "Accuracy"
```

```
#Plotar a arvore obtida
m=tree_model$finalModel
prp(m, box.palette = "Reds", tweak = 1.2)
```



No contexto de dados desbalanceados, o **caret** oferece técnicas de amostragem na configuração **trControl** da função **train**. Essas técnicas incluem Sobreamostragem (para aumentar a presença da classe minoritária), Subamostragem (para diminuir a presença da classe majoritária), SMOTE (Synthetic Minority Over-sampling Technique) e ROSE (Random Over-Sampling Examples). Estas são fundamentais para assegurar que o modelo de machine learning não fique enviesado em favor da classe mais representada no conjunto de dados. A utilização adequada destas técnicas ajuda a melhorar a performance do modelo em dados desbalanceados.

```
#Definir o hiperametros
hyper=expand.grid(
  cp = seq(0.005, 1.0, 0.005) # complexidade de 0.005 até 1.0
)
#VAlidação K-fold e sobreamostragem (Oversampling)
ctrl=trainControl(method="cv",number=10,
                  sampling = "up")

tree_model <- train(diabetes ~.,  data = train.data,   method =  "rpart",tuneGrid=hyper,tr

##Visuzalizar as acuráciass em função do parametro de complexidade
plot(tree_model)
```



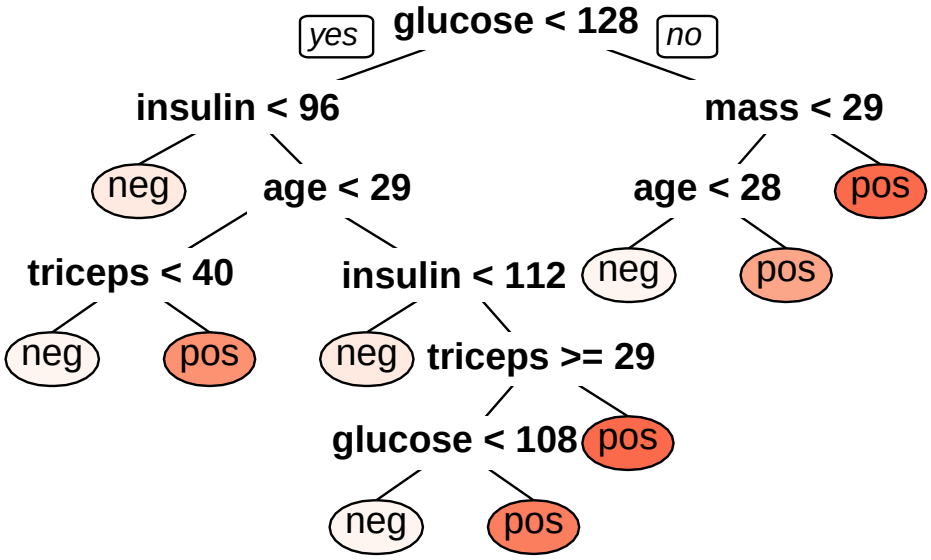

```
#Resultados do modelo
tree_model$bestTune
```

```
    cp
3 0.015
```

```
tree_model$metric
```

```
[1] "Accuracy"
```

```
#Plotar a arvore obtida
m=tree_model$finalModel
prp(m, box.palette = "Reds", tweak = 1.2)
```



4.6 Tendências e Avanços na Machine Learning no R

1. **Redes Neurais:** No R, a implementação de redes neurais é facilitada por pacotes como `nnet` (W. N. Venables and Ripley 2002) e `neuralnet` (Fritsch, Guenther, and Wright 2019), que permitem a criação de modelos de rede neural para uma variedade de tarefas de classificação e regressão. Essas redes são fundamentais para entender conceitos básicos de IA antes de avançar para técnicas mais complexas.

2. **Máquinas de Vetores de Suporte (SVM):** R oferece suporte a SVMs por meio de pacotes como `e1071` (Meyer et al. 2023), permitindo a construção de modelos eficazes para classificação e regressão, especialmente úteis em conjuntos de dados de alta dimensão.

3. **Deep Learning:** O R tem visto um crescimento significativo na integração com tecnologias de Deep Learning. Pacotes como `keras`(Allaire and Chollet 2023b) e `tensorflow`(Allaire and Tang 2023) permitem aos usuários do R acessar e implementar redes neurais profundas de forma eficiente. Esses pacotes trazem a potência do Deep Learning para a comunidade R, permitindo aplicações em visão computacional, reconhecimento de fala e outras áreas avançadas de IA.

4. **Aprendizado por Reforço:** Uma área em expansão no R é o Aprendizado por Reforço, onde o objetivo é desenvolver modelos que aprendem a tomar decisões otimizadas. Pacotes como `ReinforcementLearning` (Proellochs and Feuerriegel 2020) e `markovchain` (Spedicato 2017) estão facilitando a implementação desses complexos algoritmos de aprendizado.

5. **AutoML:** A automatização no processo de Machine Learning é uma tendência crescente. No R, ferramentas como o `automl`(Boulangé 2020) estão simplificando o processo de seleção e otimização de modelos, tornando a Machine Learning acessível até mesmo para aqueles com menos experiência técnica.

6. **Interpretabilidade e Ética em IA:** Com o aumento da complexidade dos modelos, a necessidade de interpretabilidade e considerações éticas se tornou mais premente. Pacotes como `lime` (Hvitfeldt, Pedersen, and Benesty 2022)e `DALEX`(Biecek 2018) estão na vanguarda, fornecendo ferramentas para explicar e interpretar modelos complexos, e abordar questões éticas.

7. **Integração com Big Data:** A capacidade de trabalhar com grandes volumes de dados é crucial. Pacotes como `sparklyr` (Luraschi et al. 2023)oferecem integração com Apache Spark, permitindo o processamento de grandes conjuntos de dados dentro do ambiente R.

8. **Modelagem Bayesian:** Métodos Bayesianos estão ganhando tração no R para uma ampla variedade de aplicações. Pacotes como `brms`(Bürkner 2017) e `rstan` (2023) oferecem frameworks avançados para modelagem estatística Bayesiana.

   Essas tendências demonstram como o R está evoluindo e se adaptando às necessidades de uma paisagem de dados em rápida mudança, mantendo-se como uma ferramenta valiosa e relevante no campo do Machine Learning.

5 Modelagem e Análise de Texto

A mineração de texto é uma área fascinante da ciência de dados que se concentra na extração de informações significativas de dados textuais. Na era digital de hoje, onde enormes volumes de texto são gerados diariamente, a mineração de texto torna-se essencial para analisar e compreender esses dados. Com o uso da linguagem de programação R, essa tarefa não apenas se torna acessível, mas também altamente eficiente (Kwartler 2017; J. Silge and Robinson 2017).

5.1 Conceitos Básicos de Mineração de Texto

A mineração de texto é uma área crucial da ciência de dados, dedicada à análise e extração de informações relevantes de grandes volumes de dados textuais. Com a proliferação de dados digitais, a habilidade de efetivamente analisar textos se torna cada vez mais importante. A linguagem de programação R, conhecida por sua aplicação em estatística e ciência de dados, oferece ferramentas robustas para a mineração de texto (Anandarajan, Hill, and Nolan 2019; Kwartler 2017; J. Silge and Robinson 2017; Kumar and Paul 2016).

Antes de explorarmos as ferramentas específicas do R, é essencial entender alguns conceitos fundamentais da mineração de texto (Caseli and Nunes 2023):

1. **Corpus**: No contexto da mineração de texto, um "corpus" refere-se a uma coleção de documentos de texto. Estes documentos podem variar em tamanho e forma, desde tweets e comentários online até artigos acadêmicos e livros. Um corpus é o ponto de partida para a maioria das análises de texto, servindo como o conjunto de dados primário sobre o qual os métodos de mineração de texto são aplicados.

2. **Tokenização**: Este é o processo de dividir o texto em unidades menores, chamadas "tokens". Tokens podem ser palavras, frases ou até mesmo caracteres individuais. A tokenização é um passo crucial, pois transforma grandes blocos de texto em pedaços menores e mais gerenciáveis, permitindo uma análise mais detalhada e minuciosa. Esses tokens podem ser representados de várias formas:

    - Caracteres: Onde o texto é dividido em caracteres individuais.

- Palavras: Separação do texto em palavras individuais, facilitando a análise de frequência de palavras e outras métricas baseadas em palavras.
- N-gramas: Esta forma agrupa sequências de 'n' elementos adjacentes. Por exemplo, em um bigrama (um tipo de n-grama onde n=2), palavras são agrupadas em pares, permitindo análise contextual mais detalhada.
- Sentenças: O texto é dividido em sentenças completas, útil para análises que requerem compreensão do contexto mais amplo do texto.

Na análise de texto o conceito de "termo" desempenha um papel central. Neste contexto, o termo é uma unidade flexível que pode variar de caracteres individuais a palavras ou sequências de n elementos, dependendo das necessidades específicas da análise (Anandarajan, Hill, and Nolan 2019; Caseli and Nunes 2023).

Na maioria das vezes, um "termo" é usado como sinônimo de palavra. Esta flexibilidade permite que a análise seja adaptada conforme o objetivo do estudo, seja ele focado na frequência de letras, palavras específicas ou padrões de frases (Anandarajan, Hill, and Nolan 2019; Caseli and Nunes 2023)..

A frequência de termos é uma métrica crucial na análise de textos. Ela se refere ao número de vezes que um termo específico aparece em um conjunto de documentos, como um corpus. Esta métrica é fundamental para (Anandarajan, Hill, and Nolan 2019; Kwartler 2017; J. Silge and Robinson 2017; Kumar and Paul 2016):

- Quantificar a presença ou a importância de termos individuais.
- Realizar contagem simples de ocorrências de um termo em relação ao total de documentos.
- Identificar termos-chave e padrões de uso em um corpus.

Uma das ferramentas mais úteis na análise de texto é a Matriz Documento-Termos (MDT). Esta representação tabular descreve a frequência de termos em documentos de um corpus (Anandarajan, Hill, and Nolan 2019; Kwartler 2017; J. Silge and Robinson 2017; Kumar and Paul 2016):

- Cada linha da matriz representa um documento individual.
- Cada coluna corresponde a um termo ou palavra.
- Os valores na matriz indicam a frequência de ocorrências de um termo em um documento.

A MDT é uma maneira eficaz de visualizar e analisar a relação entre documentos e termos, facilitando a identificação de padrões.

Para uma representação gráfica e intuitiva da análise de texto, as nuvens de palavras são extremamente populares. Elas destacam as palavras mais mencionadas em um texto, usando tamanhos e fontes de letras diferentes para representar a frequência das ocorrências das palavras. Essas nuvens oferecem uma visão rápida e visualmente atraente dos termos mais relevantes em um corpus (Anandarajan, Hill, and Nolan 2019; Kwartler 2017; J. Silge and Robinson 2017; Kumar and Paul 2016)..

5.1.1 A Importância da Língua na Mineração de Texto

Na mineração de texto, um aspecto crucial que muitas vezes determina a eficácia da análise é o idioma ou a língua do texto. A dependência da língua é significativa, pois diferentes línguas possuem estruturas e características únicas que influenciam a maneira como o texto é processado e analisado (Anandarajan, Hill, and Nolan 2019; Caseli and Nunes 2023)..

A mineração de textos se baseia fortemente nos níveis de organização de uma língua para criar recursos eficazes de processamento de linguagem natural (PLN). Isso inclui (Anandarajan, Hill, and Nolan 2019; Caseli and Nunes 2023).:

- **Entrada e Saída de Modelos**: O idioma do texto influencia diretamente a maneira como os dados são inseridos nos modelos de PLN e como as informações são extraídas. Diferentes idiomas podem exigir abordagens específicas para a tokenização, análise morfológica, sintática e semântica.
- **Níveis Hierárquicos de Organização da Língua**: Cada língua possui uma estrutura hierárquica única, que inclui fonemas, morfemas, palavras, frases e textos. Essa estrutura hierárquica é essencial para transformar a linguagem não estruturada em uma representação matemática adequada para modelagem. Por exemplo, a maneira como as palavras são formadas (morfologia) e organizadas em frases (sintaxe) varia significativamente de uma língua para outra.
- **Representação Matemática**: A representação matemática de textos é um passo fundamental na mineração de texto, permitindo que os modelos de PLN quantifiquem e analisem os dados de texto. A eficácia dessa representação depende de uma compreensão profunda dos níveis hierárquicos do idioma em questão.

Portanto, a consideração cuidadosa da língua é indispensável em qualquer processo de mineração de texto. Entender as nuances e características específicas de um idioma permite uma análise mais precisa e eficiente dos dados textuais.

5.2 Introdução à Mineração de Texto com R

Para ilustrar a aplicação da mineração de texto, utilizaremos dados coletados do Twitter no período de 23 a 26 de novembro de 2020, usando a palavra-chave "Coronavirus". Nesta coleta, foram obtidos mais de 100 mil tweets, mas para nossa análise, utilizaremos uma amostra de 2000. Além do texto dos tweets, estão disponíveis mais de 90 informações adicionais, como nome do usuário, data, horário, entre outras. Faça o download do arquivo aqui **`tweets.csv`**

Para começar a análise dos dados do Twitter, primeiro precisamos instalar e carregar alguns pacotes essenciais no R. Estes pacotes nos ajudarão a manipular, processar e visualizar os dados de texto:

- **`tidyverse`**: Uma coleção de pacotes para ciência de dados que torna mais fácil a manipulação e visualização de dados.
- **`tidytext`**: Especificamente focado na mineração de texto, facilita a conversão de texto em um formato estruturado para análise.
- **`tm`** (Text Mining): Um framework abrangente para mineração de texto e análise de dados textuais.
- **`wordcloud`**: Permite a criação de nuvens de palavras, que são úteis para visualizar a frequência de palavras.
- **`ggwordcloud`**: Uma extensão do **`ggplot2`** para a criação de nuvens de palavras esteticamente agradáveis e informativas.

```
library(tidyverse)
library(tidytext)
library(tm)
library(wordcloud)
library(ggwordcloud)
```

5.2.1 Preparando os Dados

Após configurar o ambiente com os pacotes necessários, o próximo passo é preparar os dados para análise. Isso envolve os seguintes passos:

1. **Download do Arquivo**: Primeiro, fazemos o download do conjunto de dados **`tweets.csv`** que contém os dados coletados do Twitter.

2. **Importar o Arquivo CSV**: Em seguida, importamos o arquivo CSV para o R. Usamos a função **`read.csv()`** para carregar os dados no R e visualizamos as primeiras linhas do conjunto de dados com **`head(rt)[6:7]`** para verificar a estrutura e o conteúdo dos dados.
3. **Selecionar os Tweets**: Finalmente, extraímos especificamente a coluna de texto dos tweets do conjunto de dados para análise. Armazenamos apenas os textos dos tweets em uma variável chamada **`tweets`**.

```r
#Importar um arquivo
rt=read.csv("tweets.csv")
head(rt)[6:7]
```

```
1
2 Pode colar tranquilo, pois estamos adotando as principais medidas para evitar a proliferaç
3
4
5
6                          Pelas imagens apresentadas pela TV, c/festas e aglomerações no R
              source
1 Jornal ADVFN Brasil
2           TweetDeck
3     Twitter Web App
4     Twitter Web App
5     Twitter Web App
6     Twitter Web App
```

```r
#Obter apenas o tweets
tweets=(rt$text)
```

Após a preparação inicial dos dados, o próximo passo na análise de texto é a criação de um corpus. Um corpus é uma coleção de documentos textuais que serve como base para a análise.

1. **Criar o Corpus**: Usamos a função **`VCorpus`** do pacote **`tm`** para criar o corpus. **`VCorpus`** é usado para criar um corpus volátil, que é armazenado na memória (em oposição a um corpus persistente, que é armazenado em disco).
2. **Fonte de Dados**: O argumento **`VectorSource`** é utilizado para indicar a fonte dos dados. No caso, usamos **`x = tweets`**, onde **`tweets`** é a variável contendo os textos extraídos dos tweets.

3. **Configuração de Idioma**: No `readerControl`, definimos o idioma do corpus como português do Brasil (`"pt-BR"`). Isso é importante para garantir que as operações subsequentes de processamento de texto levem em consideração as particularidades do idioma.
4. **Visualização do Corpus**: Ao executar `cps1`, obtemos uma visualização do corpus criado, que nos dá uma ideia da estrutura e do conteúdo do mesmo.

```
##Corpus com idioma portugues
cps1=VCorpus(VectorSource(x = tweets),
                 readerControl = list(language = "pt-BR"))
cps1
```

```
<<VCorpus>>
Metadata:  corpus specific: 0, document level (indexed): 0
Content:  documents: 2000
```

5.2.2 Pré-processamento de Termos no Corpus

O pré-processamento é uma etapa essencial na mineração de texto. Ele envolve a limpeza e a normalização dos dados para tornar a análise subsequente mais eficaz. No contexto dos tweets coletados, realizaremos várias operações de pré-processamento usando o pacote `tm` em R:

1. **Padronização de Caixa**: Converter todo o texto para letras minúsculas para uniformidade.
2. **Remoção de URLs**: Eliminar links da internet, que não são relevantes para a análise de texto.
3. **Remoção de Menções e Hashtags**: Limpar menções a usuários e hashtags para focar no conteúdo textual.
4. **Remoção de Pontuação**: Excluir pontuações que não contribuem para a análise de significado.
5. **Substituição de Palavras-Chave**: Unificar variações da palavra “coronavírus” para um termo comum (‘covid’).
6. **Remoção de Palavras de Parada**: Excluir palavras comuns em português que não agregam valor significativo à análise.
7. **Remoção de Repetições e Números**: Limpar repetições de ‘k’ e números para focar no conteúdo textual.

8. **Remoção de Espaços em Branco**: Eliminar espaços extras para manter a consistência do texto.

Este processo de pré-processamento assegura que o corpus esteja limpo e normalizado, facilitando as análises futuras, como a identificação de temas recorrentes, a análise de sentimentos ou a modelagem de tópicos.

```
# Definindo padrões de palavras-chave
padroes <- c("coronav[iíî]rus", "covid-19", "covid19")
padroes <- paste(padroes, collapse = "|")

# Pré-processamento do Corpus
cps2 = cps1 %>%
  tm_map(FUN = content_transformer(tolower)) %>% # Padronização de caixa
  tm_map(FUN = content_transformer(
    function(x) gsub("https?://\\S+", "", x))) %>% # Remoção de URLs
  tm_map(FUN = content_transformer(
    function(x) gsub("@\\S+", "", x))) %>% # Remoção de menções
  tm_map(FUN = content_transformer(
    function(x) gsub('#\\S+', '', x))) %>% # Remoção de hashtags
  tm_map(FUN = content_transformer(removePunctuation)) %>% # Remoção de pontuação
  tm_map(FUN = content_transformer(
    function(x) gsub(padroes, 'covid', x))) %>% # Substituição de variações de 'coronavíru
  tm_map(FUN = content_transformer(removeWords),
         stopwords("portuguese")) %>% # Remoção de palavras de parada
  tm_map(FUN = content_transformer(
    function(x) gsub("k{1,}", "", x))) %>% # Remoção de repetições de 'k'
  tm_map(FUN = content_transformer(removeNumbers)) %>% # Remoção de números
  tm_map(FUN = content_transformer(stripWhitespace)) # Remoção de espaços em branco
```

Além das etapas anteriores de pré-processamento, uma prática importante é a remoção de acentos. Isso pode ser particularmente útil para padronizar o texto e facilitar análises posteriores. No R, isso pode ser feito através de uma função personalizada:

```
remover_acentos=function(texto){
  texto=iconv(texto, "UTF-8", "ASCII//TRANSLIT")  # Remove os acentos
  texto=gsub("[^a-zA-Z0-9]", " ", texto)  # Substitui caracteres especiais por espaços
  texto=gsub("\\s+", " ", texto)  # Remove espaços consecutivos
  texto=trimws(texto)  # Remove espaços no início e no fim do texto
  return(texto)
```

```
}

cps3=cps2 %>%
  tm_map(FUN = content_transformer(remover_acentos))
```

5.2.3 Comparação dos Três Corpus

Para entender o impacto de cada etapa de pré-processamento, você pode comparar os conteúdos dos três corpus (**cps1**, **cps2** e **cps3**):

```
cps1[[1]]$content
```

```
[1] "O desafio de levar a vacina do coronavírus as regiões remotas e impenetráveis... https:
```

```
cps2[[1]]$content
```

```
[1] " desafio levar vacina covid regiões remotas impenetráveis "
```

```
cps3[[1]]$content
```

```
[1] "desafio levar vacina covid regioes remotas impenetraveis"
```

Essa comparação permite visualizar as mudanças no texto após cada etapa de pré-processamento no primeiro tweet, destacando a eficácia das transformações realizadas.

5.2.4 Frequência de Termos

Depois de preparar e pré-processar os dados do Twitter, o próximo passo é analisar a frequência dos termos. Isso envolve converter os corpus em um formato mais estruturado e depois contar quantas vezes cada palavra aparece em cada documento. No R, esse processo pode ser realizado usando as funções do pacote **tidytext** e a criação de Matriz de Termos Documentos. Vejamos como isso é feito:

```
#Converter para formato tidy
tt1=tidy(cps1)
tt2=tidy(cps2)
tt3=tidy(cps3)

##Contagem por palavras por documento
dtm1=tt1%>%
  unnest_tokens(texto,text,token="words")%>%
  group_by(id)%>%
  count(texto)%>%
  cast_dtm(id, texto, n,weighting=weightTf)

dtm2=tt2%>%
  unnest_tokens(texto,text,token="words")%>%
  group_by(id)%>%
  count(texto)%>%
  cast_dtm(id, texto, n,weighting=weightTf)

dtm3=tt3%>%
  unnest_tokens(texto,text,token="words")%>%
  group_by(id)%>%
  count(texto)%>%
  cast_dtm(id, texto, n,weighting=weightTf)
```

Este método de análise de frequência de termos permite uma visão quantitativa de quais palavras são mais comuns em diferentes estágios do pré-processamento, ajudando a revelar padrões e temas importantes nos dados do Twitter.

5.2.5 Criação de Nuvens de Palavras com R

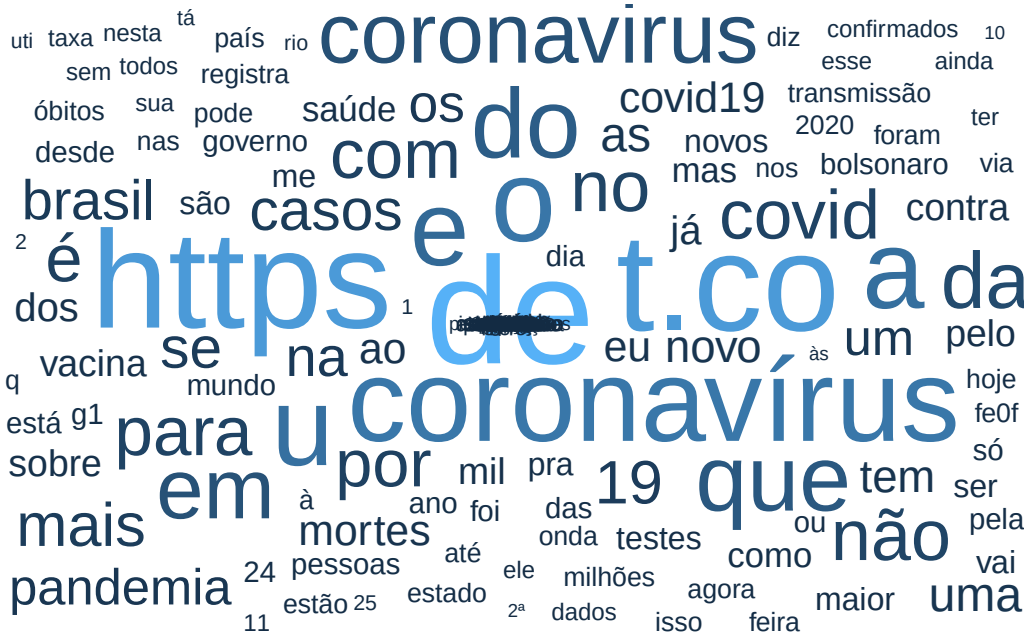
Depois de analisar a frequência dos termos nos dados do Twitter, podemos visualizar essas frequências usando nuvens de palavras. Isso nos dá uma representação gráfica intuitiva de quais termos são mais comuns nos tweets. Vamos ver como isso é feito para cada um dos três conjuntos de dados processados::

1. **Cálculo das Frequências**: Primeiro, calculamos a frequência de cada termo nos matriz de termos documentos e organizamos os termos em ordem decrescente de frequência.

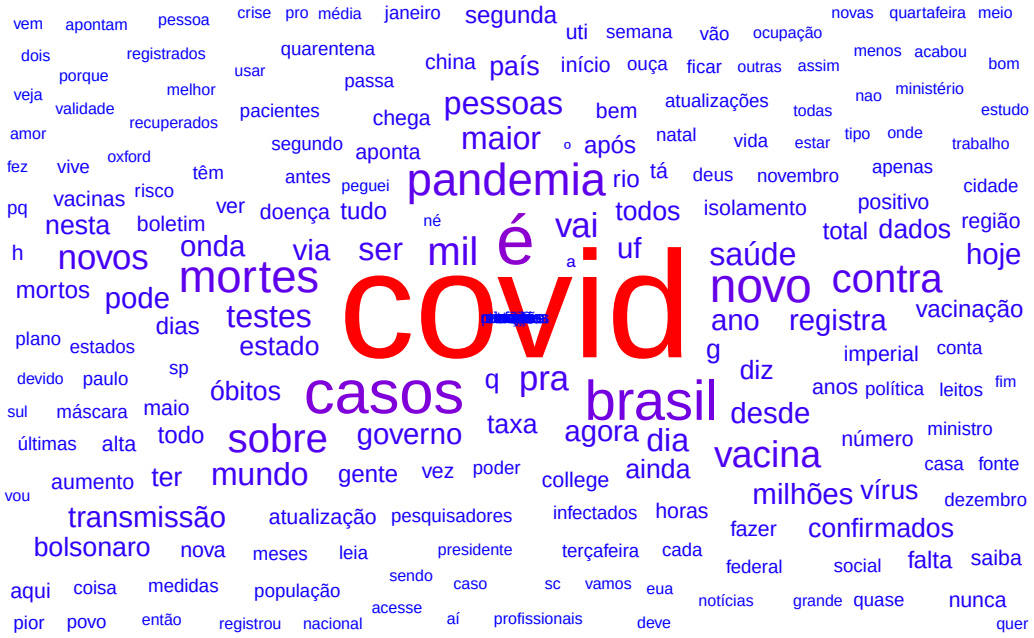
2. **Seleção dos Top Termos**: Escolhemos os termos mais frequentes para incluir na (neste caso, 200)nuvem de palavras.

3. **Criação da Nuvem de Palavras**: Usamos `ggplot` junto com `geom_text_wordcloud` para criar as nuvens de palavras.

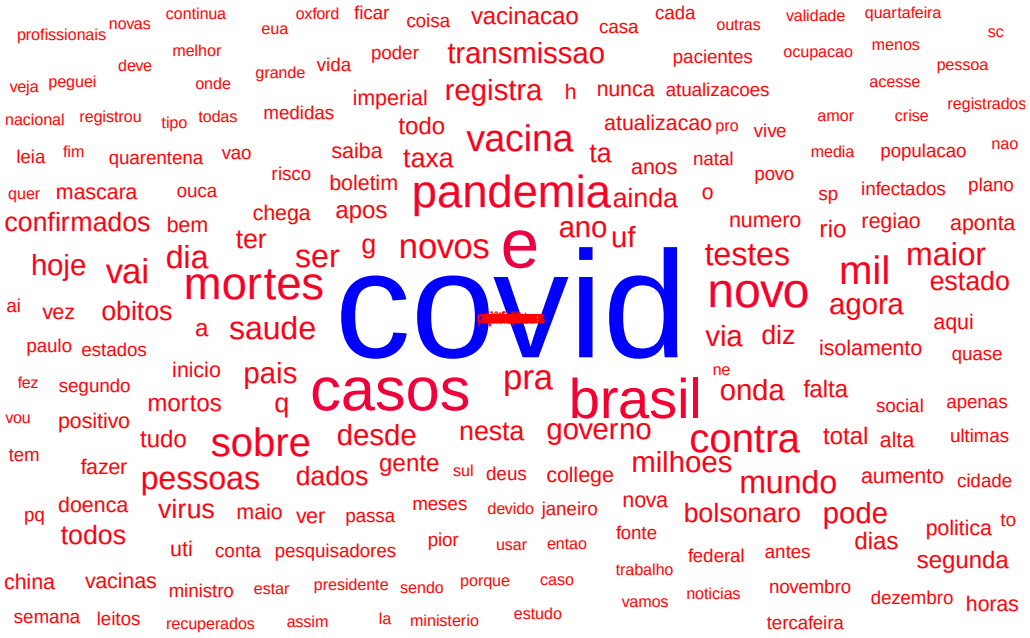
```
#Frequenciass
freq1=tidy(dtm1)%>%
  count(term,wt=count)%>%
  arrange(desc(n))

freq2=tidy(dtm2)%>%
  count(term,wt=count) %>%
  arrange(desc(n))

freq3=tidy(dtm3)%>%
  count(term,wt=count)%>%
  arrange(desc(n))

n1=200
p1=freq1 %>%
  top_n(n1)%>%
  ggplot( aes(label = reorder(term,n), size = n,color = n))+
  geom_text_wordcloud(shape = "star")+
  scale_size_area(max_size = 20)+
  theme_minimal()

p1
```




```
p2=freq2 %>%
  top_n(n1)%>%
  ggplot( aes(label = term, size = n,color=n))+
  geom_text_wordcloud(shape = "star")+
  scale_size_area(max_size = 20)+
  theme_minimal()+
  scale_color_gradient(low = "blue", high = "red")

p2
```




```
p3=freq3 %>%
  top_n(n1)%>%
  ggplot( aes(label = term, size = n,color=n))+
  geom_text_wordcloud(shape = "star")+
  scale_size_area(max_size = 20)+
  theme_minimal()+
  scale_color_gradient(low = "red", high = "blue")

p3
```



5.3 Análise de Sentimentos

A análise de sentimentos, também conhecida como mineração de opinião, é uma área interdisciplinar que estuda opiniões, sentimentos, avaliações e emoções expressas em textos. Essa técnica é fundamental para compreender a intenção emocional por trás das palavras, permitindo inferir se um texto é positivo, negativo, ou expressa emoções específicas como surpresa ou tristeza.

5.3.0.1 Aplicações da Análise de Sentimentos

Esta análise é utilizada em uma variedade de aplicações, especialmente na inteligência de negócios. Algumas aplicações comuns incluem:

- **Análise de Discussões em Redes Sociais**: Avaliar o sentimento do público sobre tópicos específicos.
- **Avaliação de Respostas de Pesquisas**: Compreender a reação das pessoas a produtos, serviços ou eventos.

- **Análise de Avaliações de Produtos**: Determinar se as opiniões expressas sobre um produto são predominantemente positivas ou negativas.

5.3.0.2 Como Funciona a Análise de Sentimentos?

A análise de sentimentos computacional busca determinar automaticamente os sentimentos expressos em um texto. Comumente, os sentimentos são classificados de forma binária (positivo x negativo), mas podem também identificar emoções específicas, como medo, alegria ou raiva.

5.3.0.3 Léxico na Análise de Sentimentos

Um método comum para realizar análise de sentimentos é baseado em um léxico, um dicionário de palavras onde cada termo recebe uma pontuação associada a um sentimento específico. As palavras podem ser classificadas como:

- Positivas, negativas ou neutras.
- Associadas a emoções específicas, como alegria, raiva, tristeza, entre outras.

5.3.0.4 NRC Emotion Lexicon

Um exemplo notável é o NRC Emotion Lexicon, que oferece uma lista extensa de palavras em vários idiomas, incluindo o português, associadas a oito emoções distintas (raiva, medo, antecipação, nojo, tristeza, surpresa, alegria e confiança) e dois sentimentos gerais (positivo e negativo). Mais informações sobre este léxico podem ser encontradas em NRC Emotion Lexicon.

5.3.1 Exemplo Prático

Para ilustrar a utilização arvore de decisão, utilizaremos dados coletados do Twitter no período de 23 a 26 de novembro de 2020, apresentado na seção anterior.

Para exemplificar a utilização da análise de sentimentos em dados do Twitter, vamos prosseguir com o exemplo, utilizando o corpus de tweets pré-processado (sem acentos) e aplicando a análise de sentimentos com o pacote `syuzhet`. Além disso, usaremos o pacote `reshape2` para realizar cálculos sumários.

```
library(syuzhet)
library(reshape2)
#Converter para formato tidy
tt2=tidy(cps2)

#Transformamos os dados para o formato adequado para análise
TF=tt2%>%
  unnest_tokens(texto,text,token="words")%>%
  mutate_at(vars(id),as.numeric)
```

Para atribuir sentimentos aos dados de texto, seguimos estes passos no R:

1. **Atribuição de Sentimento**: Usamos **`left_join`** para combinar nossos dados de texto (**`TF`**) com um dicionário de sentimentos. Com a função **`get_sentiment_dictionary('nrc', language = "portuguese")`** é obtido r um léxico de sentimentos em português. Assim, cada palavra no nosso conjunto de dados é comparada com as palavras no léxico para atribuir um sentimento correspondente.

2. **Contagem de Sentimentos por Documento**: Após atribuir os sentimentos, agrupamos os dados por documento e contamos quantas vezes cada sentimento aparece. Esse processo nos ajuda a entender a prevalência de diferentes emoções nos documentos.

```
##ATRIBUIR SENTIMENTO
TF1=TF%>%
  left_join(get_sentiment_dictionary('nrc', language = "portuguese"),
            by=c("texto"="word"))

###Contar numero de vezes que ocorre cada sentimento ocorre por documento
TF2=TF1%>%
  group_by(id)%>%
  count(sentiment)%>%
  ungroup()
```

5.3.2 Preparação dos Dados de Sentimentos para Visualização

Preparar dados de sentimentos para visualização é uma etapa crucial na análise de sentimentos. Essa tarefa envolve reorganizar e resumir os dados para que possam ser visualizados de forma eficaz. Vamos explorar como isso pode ser feito no R:

1. **Pivotamento dos Dados**: Usamos `pivot_wider` do pacote `tidyr` para transformar os dados, colocando cada tipo de sentimento em sua própria coluna. Isso facilita o cálculo de estatísticas e a visualização dos dados.
2. **Cálculo de Escores**: Calculamos um 'escore' para cada documento, que é a diferença entre a soma dos sentimentos positivos e negativos. Isso fornece uma medida geral da valência do sentimento no texto.
3. **Média de sentimentos**: Calculamos a média dos sentimentos positivos e negativos em todos os documentos
4. **Média de emoções:** Calculamos também as médias para cada emoção específica

```
##Colocar cada sentimento em uma coluna
Senti=TF2%>%
  pivot_wider(names_from = sentiment, #necessário pacote tidyr
              values_from = n,
              values_fill = 0)%>%
  group_by(id)%>%
  mutate(escore=sum(positive)-sum(negative))%>%
  ungroup()

##Visu
Senti[1:2]
```

```
# A tibble: 2,000 x 2
      id anger
   <dbl> <int>
 1     1     2
 2     2     1
 3     3     0
 4     4     0
 5     5     0
 6     6     1
 7     7     1
 8     8     1
 9     9     0
10    10     0
# i 1,990 more rows
```

```
#Obter valor médio dos sentimentos
sentiment=Senti %>%
  summarise(
    positivo = mean(positive),
    negativo = mean(negative)) %>%
  melt #reshape2

sentiment
```

```
  variable  value
1 positivo 1.0170
2 negativo 1.0435
```

```
#Obter emoções
sentiment1=Senti %>%
  summarise(
    raiva = mean(anger),
    anticipacao = mean(anticipation),
    nojo= mean(disgust),
    medo = mean(fear),
    tristeza = mean(sadness),
    alegria= mean(joy),
    surpresa = mean(surprise),
    confianca= mean(trust))%>%
  melt #pacote reshape2

sentiment1
```

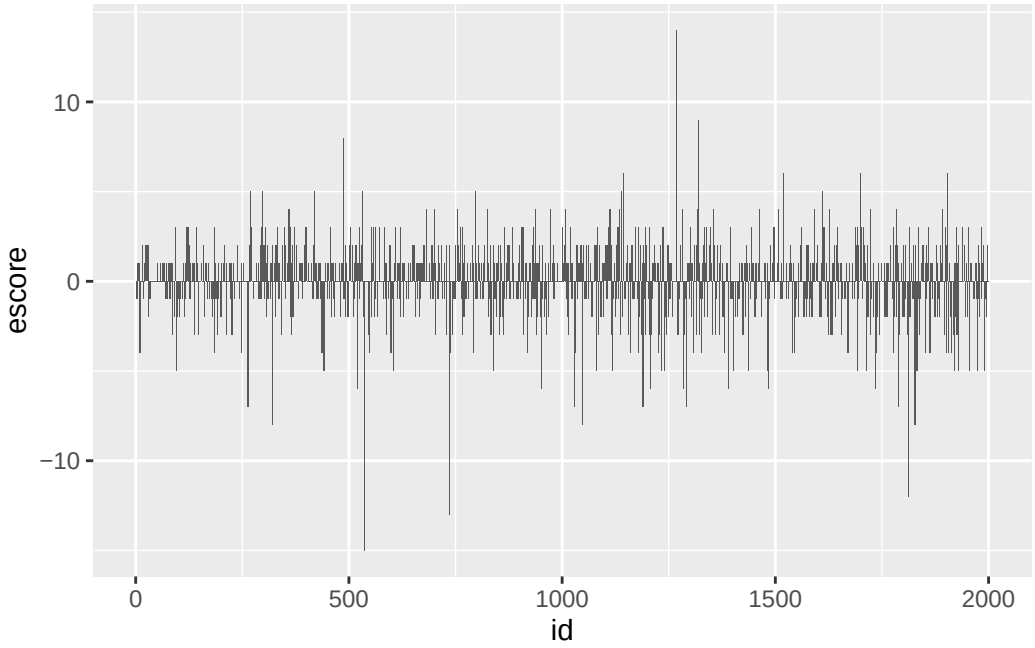
```
     variable  value
1       raiva 0.3585
2 anticipacao 0.3840
3        nojo 0.2750
4        medo 0.6175
5    tristeza 0.5125
6     alegria 0.2480
7    surpresa 0.2085
8   confianca 0.6475
```

5.3.2.1 Visualização dos Resultados

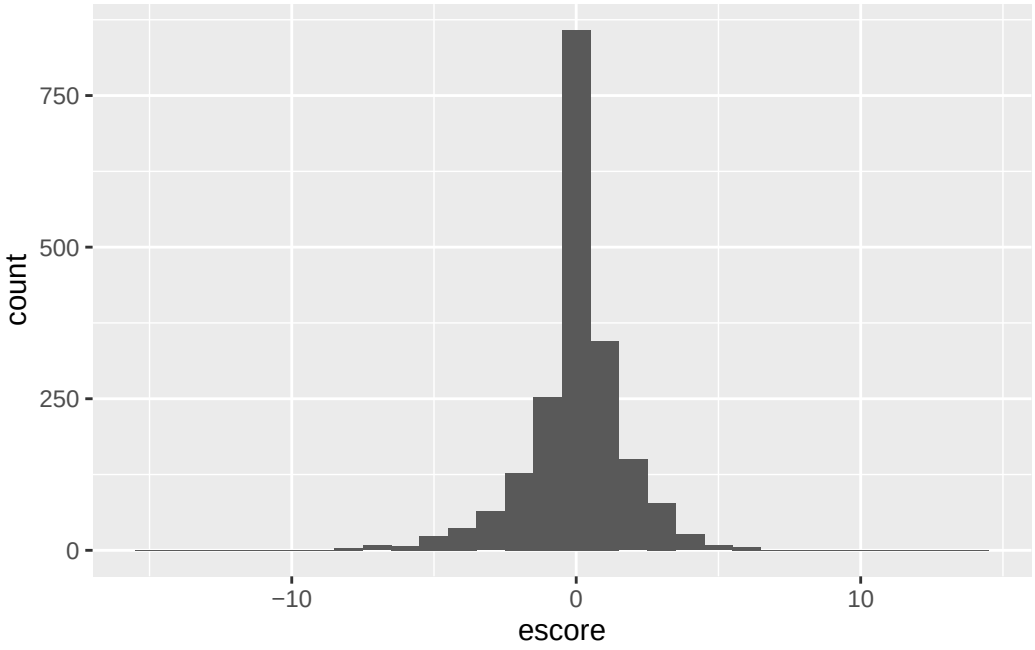
Após a transformação e cálculo das médias, os dados estão prontos para serem visualizados. Podemos criar gráficos para mostrar a distribuição dos sentimentos e emoções nos textos, oferecendo insights valiosos sobre a natureza emocional do conteúdo analisado.

Primeiro, criamos um histograma para visualizar a distribuição dos escores de sentimentos:

```
##Comparação do escore
ggplot(Senti, aes(id, escore)) +
  geom_col(show.legend = FALSE)
```




```
##Histograma
ggplot(Senti, aes(x= escore))+ geom_histogram()
```



Este gráfico mostra a frequência de diferentes escores de sentimentos nos tweets. Um escore mais alto indica uma prevalência de sentimentos positivos, enquanto um escore mais baixo indica sentimentos negativos. A distribuição desses escores pode revelar o sentimento geral dos tweets.

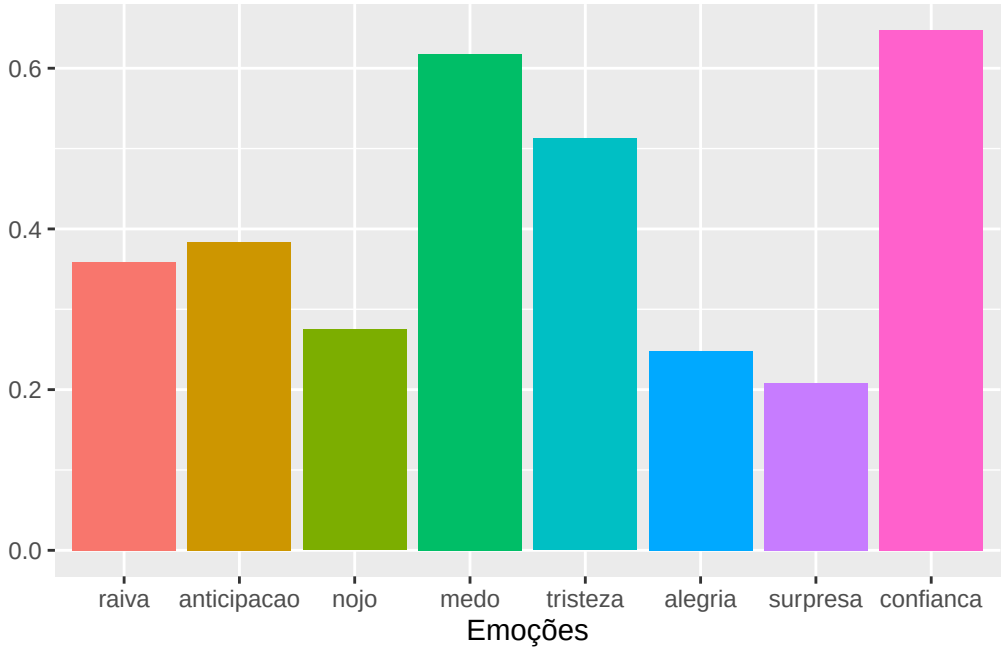
Em seguida, criamos um gráfico de barras para os sentimentos gerais (positivo e negativo):

```
##Graficos de sentimentos
ggplot(sentiment, aes(x=variable, y=value,fill=variable)) +
  geom_bar(stat="identity")+
  guides(fill="none")+
  ylab("")+
  xlab("Sentimentos")
```

Este gráfico mostra a média dos sentimentos positivos e negativos. Ele ajuda a entender qual sentimento é mais predominante no conjunto de dados.

Por fim, visualizamos as emoções específicas com outro gráfico de barras:

```
##Graficos de emoções
ggplot(data=sentiment1, aes(x=variable, y=value,fill=variable)) +
  geom_bar(stat="identity")+
  guides(fill="none")+
  ylab("")+
  xlab("Emoções")
```



Este gráfico apresenta a média de cada emoção específica (como raiva, alegria, surpresa, etc.) nos tweets. Ele fornece uma visão detalhada de quais emoções são mais expressas no conteúdo analisado.

5.3.3 Outras Análises de Texto no R

O R oferece uma variedade de pacotes e métodos para realizar análises de texto avançadas, além da mineração de sentimentos. Aqui estão algumas dessas análises (Anandarajan, Hill, and Nolan 2019; Kwartler 2017; J. Silge and Robinson 2017; Kumar and Paul 2016):

5.3.3.1 1. Análise de N-gramas e Redes Textuais

- **N-gramas**: São combinações de n itens (palavras, caracteres) usados para explorar padrões de linguagem e contextos em textos. A análise de n-gramas é útil para entender a estrutura e o uso da linguagem em um corpus. No R, pacotes como `tm` (Feinerer and Hornik 2023) e `tidytext` (Julia Silge and Robinson 2016) podem ser usados para gerar e analisar n-gramas.

- **Redes Textuais**: Essas redes representam as relações entre palavras ou frases em um texto, visualizando como os termos estão conectados. Isso pode ser feito com pacotes como `igraph` (Csardi and Nepusz 2006) e `ggraph` (Pedersen 2022), que permitem a construção e visualização de redes complexas de palavras ou frases.

5.3.3.2 2. Modelos de Classificação de Texto

- Modelos de classificação de texto são utilizados para categorizar textos em diferentes grupos ou classes. Essa análise é fundamental em aplicações como filtragem de spam e categorização de conteúdo. Pacotes como `caret` (Kuhn and Max 2008), `e1071` (Meyer et al. 2023) oferecem ferramentas para treinar e aplicar modelos de classificação, como máquinas de vetores de suporte (SVM) e modelos bayesianos.

5.3.3.3 3. Agrupamento ou Clusterização de Texto

- O agrupamento de texto envolve a organização de textos em grupos baseados em sua similaridade. É uma forma de análise não supervisionada que pode revelar padrões e temas ocultos em grandes conjuntos de dados. Pacotes como `cluster` (Maechler et al. 2022)e `factoextra` (Kassambara and Mundt 2020) fornecem métodos para realizar clusterização, como K-means e análise hierárquica.

5.3.3.4 4. Modelagem de Tópicos

- A modelagem de tópicos é uma técnica que identifica tópicos ou temas em um conjunto de documentos. É amplamente utilizada para descobrir estruturas latentes em coleções de texto. O pacote `topicmodels` (Grün and Hornik 2023)é um dos mais populares no R para esta finalidade, oferecendo implementações de algoritmos como Latent Dirichlet Allocation (LDA).

5.4 Tendências e Avanços na Análise de Texto

As tendências e avanços na análise de texto estão constantemente evoluindo, com novas técnicas e metodologias emergindo regularmente. Vamos explorar essas tendências:

- **word2vec**: Esta técnica revolucionária, desenvolvida por pesquisadores do Google, transforma palavras em vetores numéricos, capturando seu contexto e relações semânticas. O word2vec é amplamente utilizado para tarefas como agrupamento de palavras semelhantes e analogias de palavras. No R, pacotes como `text2vec` (Selivanov, Bickel,

and Wang 2023) e `word2vec` (Wijffels and Watanabe 2023) facilitam a aplicação do word2vec.

- **doc2vec**: Uma extensão do word2vec, o doc2vec é capaz de representar documentos inteiros, não apenas palavras isoladas, em espaços vetoriais. Essa técnica é útil para compreender a semântica em um nível de documento e para tarefas como a comparação de documentos. No R esta técnica pode ser aplicada pelo pacote `doc2vec` (Wijffels 2021)
- **BERT e Transformadores**: O BERT (Bidirectional Encoder Representations from Transformers) e modelos baseados em transformadores têm revolucionado o campo do processamento de linguagem natural (PLN). Esses modelos capturam contextos complexos e nuances linguísticas, sendo altamente eficientes em tarefas como compreensão de texto e tradução automática.
- **Aprendizado Profundo em PLN**: O uso de redes neurais profundas em PLN abriu caminho para avanços significativos em análise de sentimentos, geração de texto, e muito mais.
- **Análise de Texto Multilíngue**: Com a globalização, cresce a necessidade de ferramentas capazes de analisar textos em múltiplos idiomas. Isso inclui não apenas a tradução, mas também a compreensão e análise de nuances culturais e linguísticas.
- **Detecção Automática de Fake News**: A identificação de informações falsas ou enganosas é uma área emergente, com a aplicação de técnicas de PLN para detectar e sinalizar conteúdos potencialmente falsos.
- **Visualização de Dados de Texto**: Avanços em visualização, como nuvens de palavras interativas e mapeamento de tópicos, estão ajudando a tornar a análise de grandes conjuntos de dados textuais mais acessível e compreensível.

References

Al-Karkhi, A. F. M., and W. A. A. Alqaraghuli. 2019. *Applied Statistics for Environmental Science with r*. Elsevier Science.

Allaire, JJ, and François Chollet. 2023b. "Keras: R Interface to 'Keras'." https://CRAN.R-project.org/package=keras.

———. 2023a. "Keras: R Interface to 'Keras'." https://CRAN.R-project.org/package=keras.

Allaire, JJ, and Yuan Tang. 2023. "Tensorflow: R Interface to 'TensorFlow'." https://CRAN.R-project.org/package=tensorflow.

Anandarajan, M., C. Hill, and T. Nolan. 2019. *Practical Text Analytics: Maximizing the Value of Text Data*. Advances in Analytics and Data Science. Springer International Publishing.

Biecek, Przemyslaw. 2018. "DALEX: Explainers for Complex Predictive Models in r" 19: 1–5. https://jmlr.org/papers/v19/18-416.html.

Boulangé, Alex. 2020. "Automl: Deep Learning with Metaheuristic." https://CRAN.R-project.org/package=automl.

Burger, S. V. 2018. *Introduction to Machine Learning with r: Rigorous Mathematical Analysis*. O'Reilly Media.

Bürkner, Paul-Christian. 2017. "{Brms}: An {r} Package for {Bayesian} Multilevel Models Using {Stan}" 80. https://doi.org/10.18637/jss.v080.i01.

Bzdok, Danilo, Naomi Altman, and Martin Krzywinski. 2018. "Statistics Versus Machine Learning." *Nature Methods* 15: 233–34.

Caseli, H. M., and M. G. V. Nunes, eds. 2023. *Processamento de Linguagem Natural: Conceitos, Técnicas e Aplicações Em Português*. BPLN.

Chan, B. K. C. 2015. *Biostatistics for Epidemiology and Public Health Using r*. Springer Publishing Company.

Csardi, Gabor, and Tamas Nepusz. 2006. "The Igraph Software Package for Complex Network Research" Complex Systems: 1695. https://igraph.org.

Donoho, David. 2017. "50 Years of Data Science." *Journal of Computational and Graphical Statistics* 26 (4): 745–66.

Feinerer, Ingo, and Kurt Hornik. 2023. "Tm: Text Mining Package." https://CRAN.R-project.org/package=tm.

Fritsch, Stefan, Frauke Guenther, and Marvin N. Wright. 2019. "Neuralnet: Training of Neural Networks." https://CRAN.R-project.org/package=neuralnet.

Giorgi, Federico M., Carmine Ceraolo, and Daniele Mercatelli. 2022. "The r Language: An Engine for Bioinformatics and Data Science." *Life* 12 (5).

Grün, Bettina, and Kurt Hornik. 2023. "Topicmodels: Topic Models." https://CRAN.R-project.org/package=topicmodels.

Hothorn, Torsten. 2023. "CRAN Task View: Machine Learning & Statistical Learning." https://CRAN.R-project.org/view=MachineLearning.

Hvitfeldt, Emil, Thomas Lin Pedersen, and Michaël Benesty. 2022. "Lime: Local Interpretable Model-Agnostic Explanations." https://CRAN.R-project.org/package=lime.

Ihaka, Ross. 1998. "R: Past and Future History." *Computing Science and Statistics* 30: 392–96.

Ihaka, Ross, and Robert Gentleman. 1996. *R: A Language for Data Analysis and Graphics. Journal of Computational and Graphical Statistics*. Vol. 5. 3. Taylor & Francis.

Jalajakshi, V, and A N Myna. 2022. "Importance of Statistics to Data Science." *Global Transitions Proceedings* 3 (1): 326–31.

James, Gareth, Daniela Witten, Trevor Hastie, and Robert Tibshirani. 2023. *An Introduction to Statistical Learning: With Applications in r*. 2nd ed. Springer.

Kalyan, Sudhaka. 2018. "Python Vs. R Programming Language." *International Journal of Management, IT and Engineering* 8 (8): 70–79.

Kassambara, Alboukadel, and Fabian Mundt. 2020. "Factoextra: Extract and Visualize the Results of Multivariate Data Analyses." https://CRAN.R-project.org/package=factoextra.

Kuhn, and Max. 2008. "Building Predictive Models in r Using the Caret Package." *Journal of Statistical Software* 28 (5): 1–26. https://doi.org/10.18637/jss.v028.i05.

Kumar, A., and A. Paul. 2016. *Mastering Text Mining with r*. Packt Publishing.

Kwartler, T. 2017. *Text Mining in Practice with r*. Wiley.

Lawson, J. 2014. *Design and Analysis of Experiments with r*. Chapman & Hall/CRC Texts in Statistical Science. CRC Press.

Leisch, Friedrich, and Evgenia Dimitriadou. 2021. "Mlbench: Machine Learning Benchmark Problems."

Liaw, Andy, and Matthew Wiener. 2002. "Classification and Regression by randomForest" 2: 18–22. https://CRAN.R-project.org/doc/Rnews/.

Luraschi, Javier, Kevin Kuo, Kevin Ushey, JJ Allaire, Hossein Falaki, Lu Wang, Andy Zhang, Yitao Li, Edgar Ruiz, and The Apache Software Foundation. 2023. *Sparklyr: R Interface to Apache Spark*. https://CRAN.R-project.org/package=sparklyr.

Maechler, Martin, Peter Rousseeuw, Anja Struyf, Mia Hubert, and Kurt Hornik. 2022. "Cluster: Cluster Analysis Basics and Extensions." https://CRAN.R-project.org/package=cluster.

Mailund, T. 2017. *Beginning Data Science in r: Data Analysis, Visualization, and Modelling for the Data Scientist*. Apress.

Meyer, David, Evgenia Dimitriadou, Kurt Hornik, Andreas Weingessel, and Friedrich Leisch. 2023. "E1071: Misc Functions of the Department of Statistics, Probability Theory Group

(Formerly: E1071), TU Wien." https://CRAN.R-project.org/package=e1071.

Ohri, A. 2017. *Python for r Users: A Data Science Approach*. Wiley.

Paradis, E. 2020. *Population Genomics with r*. CRC Press.

Pedersen, Thomas Lin. 2022. "Ggraph: An Implementation of Grammar of Graphics for Graphs and Networks." https://CRAN.R-project.org/package=ggraph.

Proellochs, Nicolas, and Stefan Feuerriegel. 2020. "ReinforcementLearning: Model-Free Reinforcement Learning." https://CRAN.R-project.org/package=ReinforcementLearning.

R Project. 2023a. "Conferences on r." https://www.r-project.org/conferences/.

———. 2023b. "Help on r." https://www.r-project.org/help.html.

Selivanov, Dmitriy, Manuel Bickel, and Qing Wang. 2023. "Text2vec: Modern Text Mining Framework for r." https://CRAN.R-project.org/package=text2vec.

Silge, J., and D. Robinson. 2017. *Text Mining with r: A Tidy Approach*. O'Reilly Media.

Silge, Julia, and David Robinson. 2016. "Tidytext: Text Mining and Analysis Using Tidy Data Principles in r" 1. https://doi.org/10.21105/joss.00037.

Singh, A. K., and D. E. Allen. 2016. *R in Finance and Economics: A Beginner's Guide*. World Scientific Publishing Company.

Spedicato, Giorgio Alfredo. 2017. "Discrete Time Markov Chains with r" 9. https://journal.r-project.org/archive/2017/RJ-2017-036/index.html.

Stack Overflow. 2023. "R Language Collective on Stack Overflow." https://stackoverflow.com/collectives/r-language.

Stan Development Team. 2023. "{RStan}: The {r} Interface to {Stan}." https://mc-stan.org/.

Tahsin, Anika, and Md. Manzurul Hasan. 2020. "Big Data & Data Science: A Descriptive Research on Big Data Evolution and a Proposed Combined Platform by Integrating r and Python on Hadoop for Big Data Analytics and Visualization." In *Proceedings of the International Conference on Computing Advancements*. ICCA 2020. New York, NY, USA: Association for Computing Machinery.

Thaichon, P., and S. Quach, eds. 2022. *Artificial Intelligence for Marketing Management*. 1st ed. Routledge.

Tippmann, Sylvia. 2015. "Programming Tools: Adventures with r." *Nature* 517: 109–10.

Tuffery, S. 2023. "Deep Learning: From Big Data to Artificial Intelligence with r: Deep Learning for Natural Language Processing."

Ushey, Kevin, JJ Allaire, and Yuan Tang. 2023. "Reticulate: Interface to 'Python'." https://CRAN.R-project.org/package=reticulate.

Venables, W. N., and B. D. Ripley. 2002. "Modern Applied Statistics with s." https://www.stats.ox.ac.uk/pub/MASS4/.

Venables, W., and B. D. Ripley. 2013. *S Programming*. Statistics and Computing. Springer New York.

Wickham, Hadley. 2016. "Ggplot2: Elegant Graphics for Data Analysis." https://ggplot2.tidyverse.org.

Wickham, H., and G. Grolemund. 2016. *R for Data Science: Import, Tidy, Transform,*

Visualize, and Model Data. O'Reilly Media.

Wijffels, Jan. 2021. "Doc2vec: Distributed Representations of Sentences, Documents and Topics." https://CRAN.R-project.org/package=doc2vec.

Wijffels, Jan, and Kohei Watanabe. 2023. "Word2vec: Distributed Representations of Words." https://CRAN.R-project.org/package=word2vec.

Zbicki, R. E., and T. M. dos Santos. 2020. *Aprendizado de Máquina: Uma Abordagem Estatística*. 1st ed.