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

Classificação Binária via Bayes Ingênuo: um estudo comparativo de predições

Victor Alves Dogo Martins

Trabalho de Conclusão de Curso

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

Classificação Binária via Bayes Ingênuo: um estudo comparativo de predições

Victor Alves Dogo Martins
Orientadora: Profª Drª Andressa Cerqueira

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

São Carlos
Abril de 2023

Federal University of São Carlos
Exact and Technology Sciences Center
Department of Statistics

Binary Classification via Naive Bayes: a comparative study of predictions

Victor Alves Dogo Martins
Advisor: Prof. Dr. Andressa Cerqueira

Bachelors dissertation submitted to the Department of Statistics, Federal University of São Carlos - DEs-UFSCar, in partial fulfillment of the requirements for the degree of Bachelor in Statistics.

São Carlos
March 2023

Victor Alves Dogo Martins

Classificação Binária via Bayes Ingênuo: um estudo comparativo de predições

Este exemplar corresponde à redação final do trabalho de conclusão de curso devidamente corrigido e defendido por Victor Alves Dogo Martins e aprovado pela banca examinadora.

Aprovado em 17 de Março de 2023

Banca Examinadora:

- Profª Drª Andressa Cerqueira (Orientadora)
- Profº Drº Luis Ernesto Bueno Salasar
- Profº Drº Rafael Izbicki

À Silza, Isabella, João e Manuella.

À Edevirges, que gostaria muito que pudesse ver tudo isso acontecendo.

Agradecimentos

Agradeço à minha família, em especial à minha mãe Silza e irmã Isabella, pelo suporte e amor na transição entre os anos de juventude para a vida adulta durante e pós-faculdade. Amo vocês do fundo do coração, não teria chegado onde cheguei sem vocês.

À Manuella e seus pais, que me acolhem como parte da família. Aprendi muito do rigor acadêmico que tenho com vocês, da paixão tanto de aprender quando de ensinar. Me orgulho de dizer que a Manu é minha irmã de outra mãe. Vocês sabem que têm um lugar especial no meu coração.

Aos meus irmãos da Ordem DeMolay, que me ensinaram (e ensinam) como ser um filho, amigo e companheiro melhor. Nunca esquecerei de nossa fraternidade e dos momentos de convívio.

Aos meus amigos da faculdade, com uma menção muitíssima especial para Ana Beatriz, Larissa, Raquel e Natalia. Aprendi que a máxima de "não se forma num curso sozinho e sem amigos" é verdadeira com vocês. Obrigado por tudo.

Aos meus amigos de São João: vocês acompanharam a minha trajetória discente, e tenho muito orgulho de acompanhar a de vocês. Também agradeço os amigos do Departamento de Física, onde até hoje alguns confundem qual o meu curso de tão próximo que sou. Aos amigos da organização da SEst, pelos momentos onde me descobri apaixonado por Estatística.

A todo o corpo docente do Departamento de Estatística da UFSCar. Em todos esses anos, aprendi lições valiosas que levarei para a vida. Em especial, agradeço minha orientadora, Profª Andressa, por acreditar em mim e neste projeto; nada disso teria sido possível sem você.

Por fim, agradeço a todas as pessoas que passaram por minha vida em algum momento. Seria injusto tentar lembrar o nome de todas, mas saibam que moldaram muito da minha pessoa.

Por todas as pessoas, círculos mencionados e mais, meu muito obrigado por estarem comigo. Caminho sobre o ombro de gigantes.

"Alguns diriam que todas as coisas devem acabar, para que a próxima possa acontecer."

(Paathurnax)

"Seu foco determina sua realidade."

(Qui-Gon Jinn)

Resumo

No presente Trabalho de Conclusão de Curso, propomos uma revisão do método de classificação Bayes Ingênuo aplicado à variáveis resposta binárias, com formalização mais aprofundada dos métodos Bayes Ingênuo Gaussiano e Bayes Ingênuo Flexível para covariáveis numéricas. Para comparação via software estatístico R, será utilizado um banco de dados cuja variável resposta binária representa vitória ou derrota, contendo tanto covariáveis numéricas e categóricas.

Palavras-chave: *aprendizado de máquina, bayes ingênuo, classificação, predição.*

Abstract

In this undergraduate thesis, we propose a review of the Naive Bayes classification method applied to binary response variables, with a more in-depth formalization of the Gaussian Naive Bayes and Flexible Naive Bayes methods for numerical covariates. To compare these methods using the statistical software R, we will use a database containing both numerical and categorical covariates, with the binary response variable representing victory or defeat.

Keywords: *machine learning, naive bayes, classification, prediction.*

Lista de Figuras

Lista de Tabelas

Sumário

Capítulo 1

Introdução

Estatística, de modo geral, é uma área utilizada em campos dos mais diversos. Encontramos usos dentro de campos mais "clássicos" como o da saúde, da engenharia e finanças, por exemplo. Por outro lado, áreas menos ortodoxas têm utilizado alguns recursos estatísticos para fins tão interessantes quanto: um exemplo são jogos de esporte. Mais especificamente, nos interessa neste trabalho a utilização de algoritmos de classificação binária para identificar a derrota ou vitória através da predição de resultados de partidas de vários jogos como futebol, rugby, tênis, jogos de baralho, eletrônicos, entre outros.

No artigo *"Bayesian Approach to Classification of Football Match Outcome"* (Rahman *et al.*, 2018), temos um exemplo de uma aplicação utilizando o algoritmo de classificação Bayes Ingênuo (ou *Naive Bayes*) na predição de resultados de jogos de futebol da Liga Inglesa da 1ª Divisão levando em conta variáveis **categóricas** prévias a determinado jogo. Esta abordagem, como é citada, foi iniciada nos anos 2000 devido à facilidade de algoritmos que consideram variáveis externas do tipo climáticas, por exemplo.

Por outro lado, no artigo *"Decision Trees for the Prediction of Outcome of Soccer Games - Historical Data Analysis"* (Stefano *et al.*, 2020), utiliza-se uma abordagem baseada em Árvores de Decisão para o mesmo fim. Neste caso, temos diversos "nós" na árvore que avaliam se uma condição (relacionada a uma ou mais covariáveis) é verdadeira ou não, trazendo o resultado da partida após avaliação de todos os nós da árvore. Esta abordagem também traz resultados (especificamente no que diz respeito à acurácia do modelo) satisfatórios; no entanto, dado que o banco de dados utilizado é distinto (a aplicação é feita em dados do Brasileirão Série A) daquele estudado por Rahman *et al.* (2018), não é prudente comparar os dois métodos apenas através dos artigos.

Existem também abordagens mais tradicionais para analisarmos dados de jogos em

geral: a principal delas é a Regressão Logística (Dunn e Smyth, 2018), mais restrita à suposição da distribuição dos dados e comportamento dos resíduos do modelo, mas que pode apresentar um poder preditivo tão bom quanto os métodos anteriores. Além disso, como visto no artigo *"Predicting football match results with logistic regression"* (Prasetio e Harlili, 2016), é um método que já foi utilizado para o mesmo contexto, inclusive apresentando resultados interessantes no que diz respeito à seleção de variáveis.

O foco deste trabalho é a comparação do método Bayes Ingênuo com os outros métodos citados, trazendo à luz diferenças conceituais e de que modo se diferenciam ou se assemelham no desempenho preditivo e nas inferências fornecidas por cada um. A intenção é descobrirmos se uma abordagem via Bayes Ingênuo se sai melhor caso for comparada a outras utilizadas para o mesmo contexto apresentado. Para tal, foi utilizado um banco de dados contendo partidas individuais do jogo "Pokémon" para todos os métodos.

No entanto, ao contrário do que é utilizado em aplicações mais elementares de Bayes Ingênuo, não possuímos apenas atributos categóricos no banco de dados mencionado, mas também covariáveis **numéricas**. Com isso, dois métodos específicos desse algoritmo de classificação serão utilizadas para comparação: o Bayes Ingênuo Gaussiano (ou *Gaussian Naive Bayes*), que supõe distribuição normal univariada para cada uma das covariáveis, e o Bayes Ingênuo Flexível (ou *Flexible Naive Bayes*), que estima a densidade das covariáveis por meio de kernels, sendo os dois métodos estudados por John e Langley (1995).

1.1 Objetivo

O objetivo do trabalho será, num primeiro momento, revisarmos o conceito de classificação, inclusive considerando como avaliar modelos do tipo, problemáticas relacionadas ao desbalanceamento de dados, entre outros conceitos. Também iremos estudar e revisar a literatura do método Bayes Ingênuo, apresentando sua definição mais usual e aplicações, para depois realizarmos um estudo aprofundado de suas duas versões específicas apresentadas. Enquanto que a literatura do método Bayes Ingênuo seja relativamente acessível, extensa e fácil de se encontrar, os métodos Bayes Ingênuo Gaussiano e Bayes Ingênuo Flexível não possuem o mesmo privilégio. Com isso, outro objetivo do trabalho será formalizar matematicamente estes dois algoritmos a fim de tornar mais acessível o estudo deles no futuro.

Em segundo lugar, de maneira sucinta, será revisada a literatura dos outros dois métodos mencionados (Árvores de Decisão e Regressão Logística) a fim de termos uma

base teórica para a comparação de todos os algoritmos estudados sob um foco preditivo. Após a revisão de todos os métodos, serão apresentadas abordagens para a seleção de covariáveis em cada um deles.

Quanto à aplicação prática, teremos uma descrição do contexto, pré-processamento e tratamento dos dados, com análise exploratória a fim de termos um norte quanto aos possíveis resultados na fase de ajuste dos modelos. Após isto, será feito o ajuste dos modelos e comparação via métricas de avaliação definidas anteriormente, como acurácia, sensibilidade, especificidade, etc.

Por fim, através do modelo ajustado via Bayes Ingênuo com melhor desempenho, será realizada uma interpretação contextual dos resultados obtidos no âmbito do banco de dados e se estabelecerá uma conclusão acerca de todos os resultados observados.

1.2 Organização do Trabalho

A monografia está organizada da seguinte forma:

- No Capítulo 2, teremos uma revisão do conceito de modelos de classificação, com foco na classificação binária, além de uma revisão de métodos avaliativos destes modelos e de problemáticas relacionadas ao **desbalanceamento** dos dados;

- No Capítulo 3, definiremos como funcionam os modelos de classificação obtidos via Bayes Ingênuo, bem como suas abordagens específicas à covariáveis contínuas, Bayes Ingênuo Gaussiano e Bayes Ingênuo Flexível;

- No Capítulo 4, revisaremos outros dois métodos que podem ser utilizados para a construção de modelos de classificação (Árvore de Classificação e Regressão Logística), além de definirmos de que forma podemos realizar o processo de **seleção de covariáveis** neles;

- No Capítulo 5, descreveremos como foi feita a etapa de pré-processamento, tratamento e análise exploratória dos dados, com uma contextualização em termos do problema tratado neles;

- No Capítulo 6, serão apresentados os resultados, comparando os métodos revistos no Capítulo 4 com o Bayes Ingênuo Gaussiano e o Bayes Ingênuo Flexível a fim de estudarmos qual deles apresenta melhor desempenho aplicado ao Banco de Dados visto no Capítulo 5.

Capítulo 2

Classificação

Antes de nos aprofundarmos na definição do método Bayes Ingênuo, serão apresentadas as definições mais elementares dos chamados Modelos de Classificação, do qual o método tema deste trabalho faz parte. Segundo Izbicki e dos Santos (2020), modelos de classificação são uma área do **Aprendizado de Máquina Supervisionado** onde, ao contrário dos modelos de regressão (em que o objetivo é predizer corretamente uma variável resposta numérica), o objetivo é construir um modelo capaz de, com base em um conjunto de atributos, predizer a classe à qual uma observação pertence.

2.1 Contextualização e Classificação

A necessidade da construção de um modelo preditivo se dá, em suma, pela necessidade de predizermos uma variável de interesse no contexto de novas observações com base em observações passadas.

Matematicamente falando, queremos construir uma função $g(x)$ de tal forma que ela possa predizer a variável resposta Y em novas observações da forma:

$$(X_{n+1}, Y_{n+1}), (X_{n+2}, Y_{n+2}), \dots (X_{n+m}, Y_{n+m}),$$

onde X são chamadas **atributos**, **variáveis independentes** ou **covariáveis** e Y são chamadas de **variáveis resposta**, **rótulos** ou **variáveis dependentes**. Assim, predizer Y resume-se em obter uma função de classificação (como é o foco deste trabalho) $g(x)$ de tal forma que:

$$g(x_{n+1}) \approx y_{n+1}, g(x_{n+2}) \approx y_{n+2}, \dots g(x_{n+m}) \approx y_{n+m}.$$

No contexto de classificação, Y assume uma quantidade finita de valores (também chamados de *categorias*) e o problema se resume em, dado um conjunto de d variáveis preditoras $\mathbf{X_{.d}}$, descobrirmos a qual classe elas pertencem. A quantidade de valores assu-

midos por Y define o nome que damos a esses problemas: se Y possui mais de duas classes, chamamos de **classificação**; se Y possui exatas duas classes, chamamos de **classificação binária**.

2.2 Classificação Binária

A diferença entre um problema de classificação qualquer e um problema de classificação binária, geralmente, é bastante simples e intuitivo. Vejamos a seguir dois exemplos que ilustram na prática aquilo que diferencia as duas situações.

Exemplo 2.1 *Um pesquisador interessado em jogos de futebol decide criar um modelo de classificação que, com base em variáveis prévias do tipo "gols feitos", "gols tomados", "saldo de gols", "investimento mensal" entre outras, prediz o resultado de uma partida. Com isso, seu conjunto de atributos X são as variáveis mencionadas anteriormente, enquanto que a variável resposta categórica Y, pelo contexto, pode assumir três valores (por exemplo, 0 para derrota, 1 para empate e 2 para derrota).*

O Exemplo 2.1 é comum no âmbito de pesquisas relacionadas à classificação correta do resultado de partidas dos mais variados esportes. O ponto-chave em que modelos de classificação comuns se diferem dos modelos de classificação binária são **o número de resultados possíveis** (ou, em termos gerais, o número de classes pertencentes à variável resposta Y).

Exemplo 2.2 *Um pesquisador interessado em jogos da NBA decide criar um modelo de classificação que, com base em covariáveis observadas previamente, prediz o resultado de uma partida. Neste caso, já que jogos de basquete, por regra, não terminam em empates, a variável resposta categórica Y pode assumir apenas dois valores (por exemplo, 0 para derrota e 1 para vitória).*

O Exemplo 2.2 constitui, dado que o número de resultados possíveis é 2, um **modelo de classificação binária**.

Definição 2.3 (Classificação Binária) *Dado um espaço de classes/categorias $c \in \mathcal{C}$ numa variável resposta Y de um modelo de classificação, temos que este é um modelo de* ***classificação binária*** *se, e somente se, o número de classes c no espaço $c \in \mathcal{C}$* ***for igual à 2****.*

O foco deste trabalho, portanto, será o contexto de classificação para quando Y possui apenas duas classes possíveis.

2.3 Função de Risco e Função Classificadora

É de interesse nosso, para o contexto de modelos de classificação apresentado na Seção 2.2, quantificar o quão bom do ponto de vista preditivo é determinado modelo. Para isso, convém definirmos uma **Função de Risco** que, quanto menor for, maior será a qualidade de nosso modelo construído.

No entanto, o contexto de modelos de classificação traz a necessidade de definirmos funções de risco distintas daquelas presentes em modelos de regressão. Um exemplo comumente utilizado é:

$$R(g) := \mathbb{E}[\mathbb{I}(Y \neq g(X))] = \mathbb{P}(Y \neq g(X)). \tag{2.4}$$

Em palavras, a equação descrita em (2.4) é a **probabilidade de erro do modelo**, ou seja, a probabilidade de que a classe predita $g(X)$ difira da classe real Y. Essa função de risco é a esperança da função de perda 0–1, descrita por:

$$L(g(X), Y) = \mathbb{I}(Y \neq g(X)). \tag{2.5}$$

Um raciocínio lógico, ao tentarmos medir o quão bom determinado modelo de classificação é, é construirmos uma função de classificação de tal forma que ela minimize a Equação (2.4); em outras palavras, uma função de classificação que minimize a probabilidade de erro ou que maximize a probabilidade de acerto (também chamada **acurácia do modelo**).

Denotando como $g(x)$ a função do modelo de classificação e supondo $c \in \mathcal{C}$ como o espaço de classes pertencentes à variável resposta Y, temos que a função que minimize (2.4) será dada por:

$$g(\mathbf{X}_{i.}) = \arg\max_{c \in \mathcal{C}} \mathbb{P}(Y_i = c | \mathbf{X}_{i.} = \mathbf{x}_{i.}). \tag{2.6}$$

Em outras palavras, nossa função classificadora é aquela que retorna a classe $c \in \mathcal{C}$ com maior probabilidade de ocorrer dado um conjunto de covariáveis $\mathbf{X}_{i.}$. Na literatura, esse classificador é chamado **Classificador de Bayes** (Izbicki e dos Santos, 2020).

Para casos em que utilizamos a classificação binária (ou seja. temos um espaço de classes da variável resposta no formato $c \in \{a, b\}$, por exemplo), a equação descrita em (2.6) equivale a:

$$g(\mathbf{x}) = a \Leftrightarrow \mathbb{P}(Y = a | \mathbf{X}) \geq \frac{1}{2}. \tag{2.7}$$

Existem casos em que não é vantajoso escolhermos $\frac{1}{2}$ como o **corte de probabilidade**.

Discutiremos a problemática envolvida na Subseção 2.5.1 mais a fundo.

2.4 Estimação da Função de Risco

2.4.1 Divisão de dados para validação

Antes de prosseguirmos para como é feita a estimação da função de risco de um modelo de classificação, é de bom-tom entendermos como fazer isso para evitar resultados indesejáveis ou não-confiáveis. Geralmente, problemas de regressão e classificação buscam avaliar seus modelos com base na distância ou diferença que o resultado predito (isto é, aquele gerado pelo modelo) possui com relação ao resultado real. Como em situações práticas nós não possuímos o valor real da variável resposta, essa categoria de avaliação é feita com o próprio banco de dados utilizado.

No entanto, um erro comum é, após o ajuste do modelo, validarmos ele utilizando como base os próprios dados usados para sua construção: isso leva ao **superajuste**, quando nosso modelo prediz muito bem os dados observados que temos em mão, mas que dificilmente se sairá bem com dados adicionais e/ou futuros. Para evitarmos isso, é comum dividirmos aleatoriamente os dados em dois conjuntos: **conjunto de treino** (usado para estimar os parâmetros do modelo) e **conjunto de teste** (utilizado para a avaliação do modelo).

Pelo fato das observações serem sorteadas aleatoriamente e os dados de teste não terem sido utilizados para a estimação dos parâmetros do modelo, a estimativa da função de risco é consistente seguindo a Lei dos Grandes Números. Existem formas das mais distintas para realizar este processo: neste trabalho, iremos utilizar a validação simples.

Divisão simples

Segundo Hastie *et al.* (2009), a divisão de dados simples consiste em sortear aleatoriamente um subconjunto da amostra (70%, por exemplo) para servir como treino e o restante para servir como conjunto de teste. Assim, considerando que temos uma amostra de tamanho n dividida num conjunto de treino de tamanho s e um conjunto de teste de tamanho $n - s$, teremos:

- Conjunto de Treino: $(\mathbf{X}_1, Y_1), (\mathbf{X}_2, Y_2), \ldots, (\mathbf{X}_s, Y_s)$
- Conjunto de Teste: $(\mathbf{X}_{s+1}, Y_{s+1}), (\mathbf{X}_{s+2}, Y_{s+2}), \ldots, (\mathbf{X}_n, Y_n)$.

Além disso, é comum criarmos um conjunto chamado de validação **dentro do conjunto de treino**, realizando assim uma segunda divisão. O conjunto de validação é ge-

ralmente utilizado quando queremos comparar parâmetros de um modelo (como quando queremos escolher um valor λ para penalização via lasso) ou comparar duas abordagens de um mesmo modelo de classificação: por exemplo, comparar o Bayes Ingênuo Flexível via Kernel Gaussiano contra Bayes Ingênuo Flexível via Kernel Uniforme. Mais detalhes sobre estas abordagens serão apresentados no Capítulo 3.

2.4.2 Estimativa

Se na Equação (2.4), podemos descrever o risco real como a probabilidade do erro, o estimador de máxima verossimilhança deste valor é a quantidade média de vezes que se erra a classe predita com base na classe observada no conjunto de validação ou teste.

Novamente, considerando um conjunto de treino de tamanho s e um conjunto de teste de tamanho $n - s$, temos que a estimativa do risco de um modelo de classificação é dada por:

$$\hat{R}(g) = \frac{1}{n-s} \sum_{i=s+1}^{n} \mathbb{I}(Y_i \neq g(\mathbf{X}_{i.})), \tag{2.8}$$

onde $\mathbb{I}(Y_i \neq g(\mathbf{X}_{i.}))$ é a função indicadora que assume 1 caso o i-ésimo valor observado diferir do i-ésimo valor predito e 0 caso contrário.

2.5 Outras Medidas Avaliativas

Ao contrário do contexto de regressão, onde a minimização de uma função de risco normalmente é o suficiente para a interpretação da qualidade de um modelo, no contexto de classificação dependemos de outras medidas avaliativas para interpretarmos o poder preditivo de nosso modelo.

Exemplo 2.9 *O Timinho FC, famoso por ser um dos piores times do Brasil, é convocado para um campeonato especial valendo uma vaga na Série A do Brasileirão. Seus analistas de esporte interpretam um modelo de classificação em que Y assume o valor 1 caso ganhem ou empatem um jogo, e 0 caso percam um jogo. Dado que fazem parte do pior time do Brasil, definem $g(x) \equiv 0$ como uma função de classificação para seu contexto (dado que perdem aproximadamente 95% de suas partidas, digamos).*

A função de classificação definida no Exemplo 2.9 possuirá risco baixo (aproximadamente 5%), pois $\mathbb{P}(Y = 1)$ é um valor baixo na população, mas isso não implica que o desempenho do modelo será bom, muito pelo contrário. Se por um lado acertam-se todas as partidas em que o Timinho FC perde, vitórias e empates são classificadas erroneamente todas as vezes. Situações como essa são ainda mais graves quando detectamos o

diagnóstico de uma doença, por exemplo.

Uma maneira de evitarmos esta categoria de erro grave é avaliarmos o desempenho de um modelo de classificação não apenas através de sua função de risco, mas sim por uma matriz de confusão. Para o caso binário, como o que foi citado no exemplo acima, teríamos uma matriz da seguinte forma:

Tabela 2.1: Matriz de Confusão para classificação binária.

	Valor Observado	
Valor Predito	Y = 0	Y = 1
Y = 0	VN (Verdadeiro Negativo)	FN (Falso Negativo)
Y = 1	FP (Falso Positivo)	VP (Verdadeiro Positivo)

Através dos valores definidos na Tabela 2.5, podemos utilizar as seguintes métricas:

- **Risco:**

$$\hat{R}(g) = \frac{FN + FP}{FN + FP + VN + VP} \tag{2.10}$$

  É a probabilidade de errarmos a classe verdadeira de uma observação. No exemplo dado, já que a função classificadora prevê que o time perderá todas as suas partidas e que 95% das partidas passadas foram perdidas, o risco estimado será de 5%, ou 0.05;

- **Acurácia:**

$$\hat{A}(g) = \frac{VN + VP}{FN + FP + VN + VP} = 1 - \hat{R}(g) \tag{2.11}$$

  É a probabilidade de acertarmos a classe verdadeira de uma observação. No exemplo dado, já que a função classificadora prevê que o time perderá todas as suas partidas e que 95% das partidas passadas foram perdidas, a acurácia estimada será de 95%, ou 0.95. Ademais, minimizar o risco equivale à maximização da acurácia em um problema de classificação;

- **Sensibilidade:**

$$S = \frac{VP}{VP + FN} \tag{2.12}$$

  É a probabilidade de, nas observações em que $Y = 1$, classificarmos corretamente a variável resposta. No exemplo dado, já que a função classificadora prevê que o time perderá todas as suas partidas e que 5% das partidas passadas foram vencidas, a sensibilidade estimada será igual a 0;

- **Especificidade:**

$$E = \frac{VN}{VN + FP} \tag{2.13}$$

É a probabilidade de, nas observações em que $Y = 0$, classificarmos corretamente a variável resposta. No exemplo dado, já que a função classificadora prevê que o time perderá todas as suas partidas e que 95% das partidas passadas foram perdidas, a especificidade estimada será igual a 100%, ou 1;

- **Valor Preditivo Positivo:**

$$VPP = \frac{VP}{VP + FP} \tag{2.14}$$

É a probabilidade de, nas observações classificadas como $Y = 1$, termos classificado corretamente a variável resposta. No exemplo dado, já que a função classificadora prevê que o time perderá todas as suas partidas, nenhuma das observações foi classificada como $Y = 1$. Logo, o Valor Preditivo Positivo estimado é 0;

- **Valor Preditivo Negativo:**

$$VPN = \frac{VN}{VN + FN} \tag{2.15}$$

É a probabilidade de, nas observações classificadas como $Y = 0$, termos classificado corretamente a variável resposta. No exemplo dado, já que a função classificadora prevê que o time perderá todas as suas partidas e que 95% das partidas observadas foram perdidas, o Valor Preditivo Negativo estimado será igual a 95%.

Com o conhecimento adquirido sobre as diferentes medidas avaliativas de um modelo de classificação, podemos afirmar que o modelo definido no Exemplo 2.9 não é bom; enquanto que o risco é baixo e a acurácia é alta, a sensibilidade do modelo é igual a 0. Usualmente, procura-se obter um modelo cujo risco seja baixo (e, consequentemente, a acurácia seja alta) e a soma Sensibilidade+Especificidade, apresente valores altos. No entanto, existem situações específicas que exigem uma atenção especial para outras métricas. Um problema de avaliação de modelos de classificação consiste em, basicamente, maximizar (ou minimizar) determinada métrica de interesse para encontrar uma função de classificação adequada para determinado contexto.

2.5.1 Desbalanceamento

O cálculo das métricas apresentadas é diretamente influenciado pelo corte de probabilidade utilizado na Equação (2.6), principalmente no caso binário. Isso implica que,

em muitos casos, utilizarmos 0.5 como um corte de probabilidade de modo a classificarmos um conjunto de covariáveis dentro de uma classe c qualquer não resultará na melhor função de classificação possível.

Pensemos especificamente no Exemplo 2.9: a proporção real de vitórias não é igual a 0.5 (sendo mais próxima de 0.05) e, consequentemente, um corte desse valor resultará num estimador que se aproxima muito da função de classificação $g(x) \equiv 0$. Essa é uma característica comum do classificador de bayes, quando o banco de dados está **desbalanceado**.

Brevemente, um banco de dados está desbalanceado quando **uma determinada classe ocorre muito mais vezes do que as demais**. Existem diferentes abordagens para lidar com este problema:

- **Utilizar uma função de perda diferente:** determinamos que, para o caso do Exemplo 2.9, termos um Falso Negativo (classificamos uma partida como perdida quando, na verdade, se trata de uma vitória) é mais grave do que o contrário. Assim, determina-se um peso maior para esse tipo de erro, resultando num corte diferente de probabilidade;

- **Trocamos o corte de probabilidade:** ao invés de determinarmos que uma partida será classificada como derrota caso o resultado da Equação (2.6) seja maior que 0.5, estabelecemos que esse corte será de 0.05 (ou qualquer valor distinto que traga um aumento no desempenho do classificador). Isso também pode ser feito através da análise de Curva ROC;

- **Reamostragem dos dados:** realizamos uma amostragem de cada uma das classes presentes no banco de dados para termos a mesma proporção de cada uma delas. Este é a abordagem menos recomendada; as outras duas mencionadas normalmente são mais eficazes e menos trabalhosas.

2.5.2 Curva ROC

A Curva ROC (abreviação de *Receiver Operator Characteristic*) é uma maneira visual de analisarmos o desempenho de modelos de classificação no geral. Valores em seus dois eixos (com Taxa de Falsos Positivos e Taxa de Verdadeiros Positivos, normalmente) variam de acordo com cortes de probabilidade indo de 0 até 1, como apresentado na Figura 2.1:

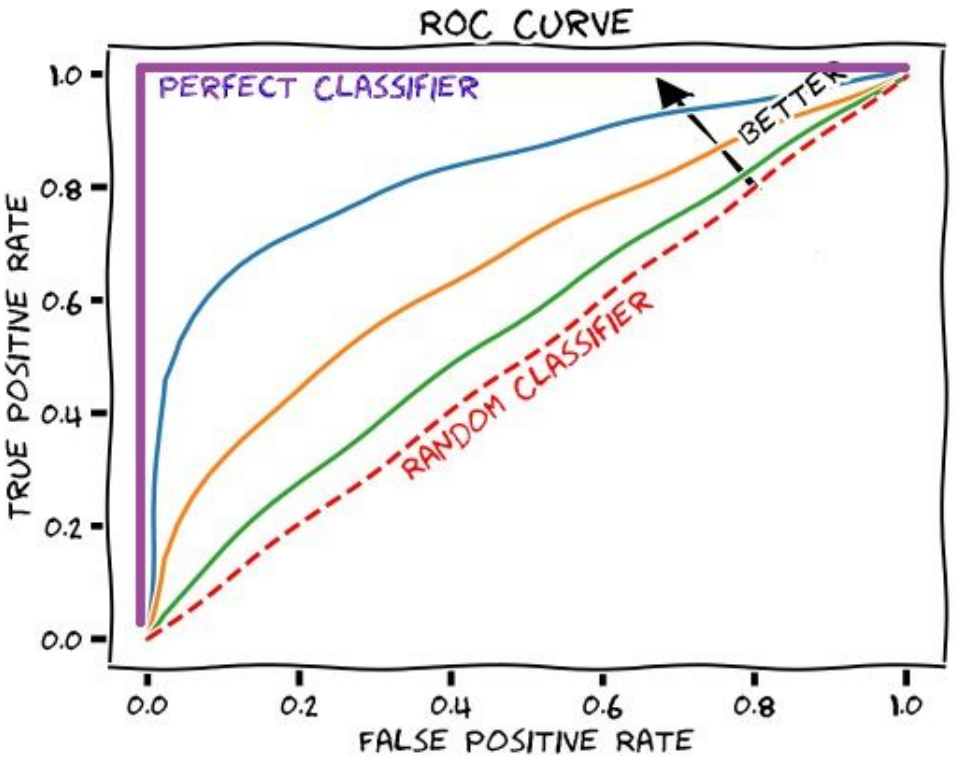

Figura 2.1: Exemplo de Curva ROC (Turing, 2022).

No Eixo X da Figura 2.1, iremos utilizar $1 - E$ (1 menos Especificidade); no Eixo Y, Sensibilidade. O melhor modelo será aquele que mais se aproximar do ponto destacado em azul, onde o suposto classificador perfeito se encontra.

A Curva ROC também é útil para encontrarmos visualmente um corte de probabilidade adequado para o classificador de bayes, principalmente em casos como os citados na Subseção 2.5.1: o valor de corte que maximizar Sensibilidade+Especificidade (denotado na Figura 2.1 por aquele mais próximo da linha lilás, onde temos escrito *"Perfect Classifier"*) pode ser utilizado, mas não é regra; casos distintos podem exigir a maximização de métricas distintas.

2.5.3 Área sob a Curva ROC

Derivada da avaliação visual vista na Subseção 2.5.2, a Área sob a Curva ROC (também chamada de *Area Under Curve*, ou *AUC*) quantifica a representação do desempenho de um modelo de classificação realizada através da Curva ROC.

Em suma, após a obtenção da Curva ROC de um modelo, estima-se sua função de densidade $f(x)$ de tal forma que:

$$AUC = \int_0^1 f(x)\ dx,\ 0 \leq AUC \leq 1 \tag{2.16}$$

Assim, um modelo melhor é aquele que apresenta a métrica AUC mais próxima de 1, e vice-versa. Ela é especialmente útil por equilibrar duas métricas: as utilizadas no eixo X e Y do gráfico da Curva ROC, como visto na Figura 2.1 (em nosso caso, 1-Especificidade e Sensibilidade respectivamente).

Capítulo 3

O Algoritmo Bayes Ingênuo

Neste capítulo, definiremos o algoritmo Bayes Ingênuo, começando pela classe de Classificadores Plug-In (a qual ele faz parte), para depois definirmos o Bayes Ingênuo para covariáveis discretas e duas abordagens para covariáveis contínuas: o Bayes Ingênuo Gaussiano e o Bayes Ingênuo Flexível.

3.1 Bayes Ingênuo Discreto

3.1.1 Classificadores Plug-In

A função classificadora de Bayes, definida em (2.6), traz por intuição um procedimento bastante simples para estimarmos o $g(x)$ de nosso modelo através de dois passos: eles descrevem os **Classificadores Plug-In**.

1. Estimamos a probabilidade condicional $\mathbb{P}(Y_i = c|\mathbf{X}_{i\cdot} = \mathbf{x}_{i\cdot})$ para cada classe $c \in \mathcal{C}$ através de algum método;

2. Considerando o Classificador de Bayes, associamos ao conjunto de covariáveis $\mathbf{X}$ a classe c que retorna a maior probabilidade dentre as probabilidades estimadas, ou seja, que maximiza a Equação (2.6).

Este nome (*plug-in*) vem do fato de que "pluga-se" (ou se insere) o estimador da probabilidade condicional na fórmula do $g(x)$ ótimo em (2.6). A seguir, constataremos a forma com que o algoritmo Bayes Ingênuo lida com a estimativa dessa probabilidade.

3.1.2 Teorema de Bayes

Bayes Ingênuo (também chamado *Naive Bayes*, em inglês) é um algoritmo de classificação bastante utilizado na literatura de modelos de classificação por possuir um poder

preditivo sólido se comparado a outros métodos mais complexos. Ele recebeu esse nome por dois motivos, o primeiro derivado da aplicação do **Teorema de Bayes** no contexto de Classificadores Plug-in:

Teorema 3.1 (Teorema de Bayes) *Suponhamos que $A_1, A_2, \ldots, A_n$ seja um conjunto de eventos de tal forma que não ocorram em simultâneo (em outras palavras, são mutualmente exclusivos e exaustivos). Além disso, suponhamos também que um determinado evento B ocorreu e desejamos definir a probabilidade de que um elemento do conjunto A tenha ocorrido **dado** que B também ocorreu. O Teorema de Bayes define que podemos obter tal probabilidade através da expressão:*

$$\mathbb{P}(A_j|B) = \frac{\mathbb{P}(B|A_j) \cdot \mathbb{P}(A_j)}{\sum_{i=1}^{n} \mathbb{P}(B|A_i) \cdot \mathbb{P}(A_i)}.$$

De maneira análoga ao que nos é apresentado no Teorema 3.1, podemos definir a função $\mathbb{P}(Y = c|X)$ do algoritmo Bayes Ingênuo através da seguinte função:

$$\mathbb{P}(Y_i = c|\mathbf{X}_{i.} = \mathbf{x}_{i.}) = \frac{\mathbb{P}(\mathbf{X}_{i.} = \mathbf{x}_{i.}|Y_i = c) \cdot \mathbb{P}(Y_i = c)}{\sum_{s \in \mathcal{C}} \mathbb{P}(\mathbf{X}_{i.} = \mathbf{x}_{i.}|Y_i = s) \cdot \mathbb{P}(Y_i = s)}. \tag{3.2}$$

3.1.3 Independência e Ingenuidade

Reescrevemos a Equação (3.2) de modo a estabelecermos a segunda propriedade do algoritmo Bayes Ingênuo que lhe dá esse nome: esta metodologia supõe **independência das covariáveis X condicional à classe Y**. É uma suposição incorreta (e *ingênua*) na grande maioria dos casos, mas torna um problema de classificação em algo convenientemente simples e fácil de lidar. Consequentemente, pela independência, temos:

$$\mathbb{P}(\mathbf{X}_{i.} = \mathbf{x}_{i.}|Y_i = c) = \prod_{j=1}^{d} \mathbb{P}(X_{i,j} = x_{i,j}|Y_i = c). \tag{3.3}$$

Além disso, no denominador da Equação (3.2), temos uma expressão equivalente à probabilidade marginal do conjunto de covariáveis $\mathbf{X}$. Portanto, na Equação (3.3), temos que:

$$\mathbb{P}(Y_i = c|\mathbf{X}_{i.} = \mathbf{x}_{i.}) = \frac{\mathbb{P}(Y_i = c) \cdot \prod_{j=1}^{d} \mathbb{P}(X_{i,j} = x_{i,j}|Y = c)}{\mathbb{P}(\mathbf{X}_{i.} = \mathbf{x}_{i.})}. \tag{3.4}$$

Visualmente, o algoritmo Bayes Ingênuo pode ter sua propriedade de independência condicional à classe da variável resposta Y representada na Figura 3.1.3:

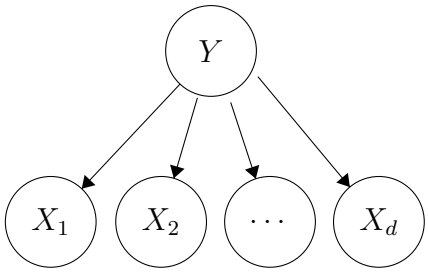

Figura 3.1: Representação visual de independência condicional à variável resposta do algoritmo Bayes Ingênuo.

Pela Figura 3.1.3, fica fácil percebermos que o algoritmo Bayes Ingênuo constitui um tipo de **Rede Bayesiana**. Temos nós (representados pelos círculos) denotando as variáveis de interesse, as arestas saindo de Y denotando a dependência condicional à variável de onde as setas saem e os nós não conectados denotando variáveis condicionalmente independentes. São modelos probabilísticos e gráficos que buscam representar as dependências existentes num conjunto de covariáveis e variáveis resposta, muito usadas para a construção de modelos de predição e/ou inferência (Hastie *et al.*, 2009).

3.1.4 Definições

Considerando a Equação (2.6) do Classificador de Bayes aplicada à Equação (3.4), temos:

$$\hat{g}(\mathbf{X}_{i.}) = \arg\max_{c \in \mathcal{C}} \left[\frac{\hat{\mathbb{P}}(Y_i = c) \cdot \prod_{j=1}^{d} \hat{\mathbb{P}}(X_{i,j} = x_{i,j} | Y = c)}{\hat{\mathbb{P}}(\mathbf{X}_{i.} = \mathbf{x}_{i.})} \right] \tag{3.5}$$

onde:

- $\mathbf{X}$ é a matriz $n \times d$ do banco de dados onde cada linha i corresponde à i-ésima observação com índice indo de 1 até n e cada coluna j corresponde à j-ésima covaríavel (ou variável explicativa) com índice no espaço $1 \times d$;

- Por consequência, $\mathbf{X}_{i.}$ corresponde ao vetor de covariávieis associado a uma observação;

- $X_{i,j}$ corresponde ao valor da j-ésima covariável na i-ésima observação;

- $Y_i = c$ é a i-ésima variável resposta assumindo o valor correspondente a uma classe c do problema.

O denominador da Equação (3.5), $\hat{\mathbb{P}}(\mathbf{X}_{i.} = \mathbf{x}_{i.})$, não faz diferença na expressão já que maximizarmos ela com relação à c com e sem ele nos trará o mesmo resultado. Logo, temos:

$$\hat{g}(\mathbf{X}_{i.}) = \arg\max_{c \in \mathcal{C}} \left[\hat{\mathbb{P}}(Y_i = c) \cdot \prod_{j=1}^{d} \hat{\mathbb{P}}(X_{i,j} = x_{i,j} | Y_i = c) \right] \tag{3.6}$$

No caso discreto, podemos estimar as probabilidades acima através da proporção observada no banco de dados (veremos isto com maiores detalhes na Subseção 3.1.5): por exemplo, $\hat{\mathbb{P}}(Y_i = c)$ seria a proporção de observações do banco de dados que pertencem à classe c.

Por outro lado, existem outras duas formas de definirmos a probabilidade populacional $\mathbb{P}(Y_i = c)$, que é uma **probabilidade à priori**: definimos ela através de conhecimento prévio acerca do contexto do problema ou afirmamos que todas as classes possuem probabilidade igual de ocorrerem (neste caso, a chamamos de **priori não informativa**). Neste último caso, é possível definirmos o estimador da função classificadora de Bayes Ingênuo da seguinte forma nestes contextos:

$$\hat{g}(\mathbf{X}_{i.}) = \arg\max_{c \in \mathcal{C}} \prod_{j=1}^{d} \hat{\mathbb{P}}(X_{i,j} = x_{i,j} | Y_i = c) \tag{3.7}$$

Alternativamente, podemos reescrever as Equações (3.6) e (3.7) aplicando a função log a elas. Por se tratar de uma função estritamente crescente, as propriedades descritas se mantém. Para probabilidades à priori diferentes ou $\hat{\mathbb{P}}(Y_i = c)$ estimada, teríamos:

$$\begin{aligned}
\hat{g}(\mathbf{X}_{i.}) &= \arg\max_{c \in \mathcal{C}} \log \left[\hat{\mathbb{P}}(Y_i = c) \cdot \prod_{j=1}^{d} \hat{\mathbb{P}}(X_{i,j} = x_{i,j} | Y_i = c) \right] \\
&= \arg\max_{c \in \mathcal{C}} \log[\hat{\mathbb{P}}(Y_i = c)] + \log \left[\prod_{j=1}^{d} \hat{\mathbb{P}}(X_{i,j} = x_{i,j} | Y_i = c) \right] \\
&= \arg\max_{c \in \mathcal{C}} \log[\hat{\mathbb{P}}(Y_i = c)] + \sum_{j=1}^{d} \log[\hat{\mathbb{P}}(X_{i,j} = x_{i,j} | Y_i = c)]
\end{aligned}$$

Já para uma probabilidade à priori não informativa, teremos:

$$\hat{g}(\mathbf{X}_{i.}) = \arg\max_{c \in \mathcal{C}} \sum_{j=1}^{d} \log[\hat{\mathbb{P}}(X_{i,j} = x_{i,j} | Y_i = c)]$$

3.1.5 Estimativas para o Caso Discreto

Como definido a partir das Equações (3.6) e (3.7) em diante, nosso interesse no algoritmo Bayes Ingênuo com covariáveis discretas é estimar as probabilidades $\mathbb{P}(X_{i,j} = x_{i,j} | Y_i = c)$ e $\mathbb{P}(Y_i = c)$. Esta última, como dito anteriormente, pode ser definida basicamente de três formas:

- Define-se a probabilidade à priori através de um conhecimento prévio acerca do problema. Por exemplo, um analista de esportes pode afirmar que, no contexto da Série A do Brasileirão, o Timinho FC possui 98% de chance de perder ou empatar uma partida, e 2% de chance de ganhar;

- Define-se a probabilidade à priori como não informativa, ou seja, o Timinho FC possui chances iguais de perder ou empatar ou ganhar uma partida;

- Estima-se a probabilidade através da proporção observada de ocorrência de cada uma das classes:
$$p_c = \frac{n_c}{n}, \tag{3.8}$$
onde n_c é o número de vezes em que a classe c, obtido através da expressão:
$$n_c = \sum_{i=1}^{n} \mathbb{I}\{Y_i = c\}. \tag{3.9}$$
e n o número de observações no banco de dados.

Para o caso de $\hat{\mathbb{P}}(X_{i,j} = x_{i,j}|Y_i = c)$, no entanto, temos que trabalhar com a distribuição condicional à Y, dada por:
$$X_{i,j}|Y_i = c \sim Multinomial(1, \boldsymbol{\theta}_{j,c} = (\theta_{j,c,1}, \theta_{j,c,2}, \ldots, \theta_{j,c,q})), \tag{3.10}$$
onde $\boldsymbol{\theta}_{j,c}$ é um vetor de probabilidades em $\mathbb{R}^q$, sendo q o número de valores que $\mathbf{X}_{.j}$ assume. O problema resume-se em estimar $\boldsymbol{\theta}_{j,c}$ e, pela definição da distribuição, seu estimador de máxima verossimilhança é dado por:
$$\hat{\theta}_{j,c,k} = \frac{n_{j,c,k}}{n_c}, \ k = 1, 2, \ldots, q, \tag{3.11}$$
onde $n_{j,c,k}$ é o número de vezes em que $\mathbf{X}_{.j}$ assume o valor k condicionado à $Y = c$, calculado através da expressão:
$$n_{j,c,k} = \sum_{i=1}^{n} \mathbb{I}\{X_{i,j} = k, Y_i = c\}, \ k = 1, 2, \ldots, q, \tag{3.12}$$
e n_c o número de vezes que Y assume a classe c no banco de dados, calculado através da Equação (3.9).

Desta forma, teremos a estimativa da probabilidade de $X_{i,j}$ condicional à Y dada por:
$$\hat{\mathbb{P}}(X_{i,j} = x_{i,j}|Y_i = c) = \frac{1}{x_{i,j}!} \cdot \prod_{k=1}^{q} \hat{\theta}_{j,c,k}^{x_{i,j}}. \tag{3.13}$$

3.2 Bayes Ingênuo Gaussiano

3.2.1 Motivação

Ao contrário do que foi apresentado na Seção 3.1, temos casos em que algumas covariáveis de nosso modelo não são discretas, e sim numéricas/contínuas. Para situações

como esta, existem procedimentos dos mais diversos para modelarmos a distribuição destas variáveis: em nosso caso, modelaremos a distribuição das covariáveis contínuas através da Distribuição Normal, quando chamamos o método de **Bayes Ingênuo Gaussiano**, e por estimação via Kernel, quando chamamos o método de **Bayes Ingênuo Flexível**. Estes dois métodos receberam destaque no artigo *"Estimating Continuous Distributions in Bayesian Classifiers"* (John e Langley, 1995).

Ambos podem fornecer estimativas sólidas no que diz respeito à classificação de suas covariáveis; no entanto, o Bayes Ingênuo Gaussiano é mais limitante por tratarmos de uma suposição (normalidade na distribuição da covariável) que nem sempre é verdade. Isso, teoricamente, pode ser traduzido em predições piores se comparado com o Bayes Ingênuo Flexível, que lida de maneira mais versátil com essa questão (e justamente por isso é chamado de flexível).

A seguir, veremos como podemos escrever as Equações (3.6) e (3.7) (bem como suas versões aplicadas ao log) e estimar a função classificadora do Bayes Ingênuo para covariáveis contínuas tendo em mente a suposição de normalidade delas.

3.2.2 Definição

Para covariáveis contínuas no geral, podemos reescrever as Equações (3.6) e (3.7) da seguinte forma:

$$\hat{g}(\mathbf{X}_{i.}) = \arg\max_{c \in \mathcal{C}} \left[\hat{\mathbb{P}}(Y_i = c) \cdot \prod_{j=1}^{d} \hat{f}\left(x_{i,j} | Y_i = c\right) \right] \tag{3.14}$$

$$\hat{g}(\mathbf{X}_{i.}) = \arg\max_{c \in \mathcal{C}} \prod_{j=1}^{d} \hat{f}\left(x_{i,j} | Y_i = c\right) \tag{3.15}$$

Estimar uma função classificadora para variáveis contínuas segundo o Bayes Ingênuo Gaussiano consiste, portanto, em estimarmos $\hat{f}\left(x_{i,j} | Y_i = c\right)$. Assim como anteriormente, quando estimamos os parâmetros da Distribuição Multinomial de $X_{i,j}$ condicional á Y_i, podemos definir os parâmetros da Distribuição Normal como dependentes da classe de Y_i e da j-ésima covariável em questão.

Para o Bayes Ingênuo Gaussiano, dizemos que $X_{i,j}$ condicional à $Y_i = c$ possui Distribuição Normal com parâmetros $\mu_{j,c}$ e $\sigma^2_{j,c}$:

$$X_{i,j} | Y_i = c \sim Normal(\mu_{j,c}, \sigma^2_{j,c}) \tag{3.16}$$

Podemos estimar os parâmetros acima por Máxima Verossimilhança:

$$\hat{\mu}_{j,c} = \frac{1}{n_c} \sum_{i=1}^{n} X_{i,j} \cdot \mathbb{I}\{Y_i = c\} \tag{3.17}$$

$$\hat{\sigma}^2_{j,c} = \frac{1}{n_c}\sum_{i=1}^{n}(X_{i,j} - \hat{\mu}_{j,c})^2 \cdot \mathbb{I}\{Y_i = c\} \tag{3.18}$$

Por definição da Distribuição Normal, temos então que:

$$\hat{f}\left(x_{i,j}|Y_i = c\right) = \frac{1}{\sqrt{2\pi\hat{\sigma}^2_{j,c}}}\exp\left\{-\frac{(x_{i,j} - \hat{\mu}_{j,c})^2}{2\hat{\sigma}^2_{j,c}}\right\}. \tag{3.19}$$

3.3 Bayes Ingênuo Flexível

3.3.1 Motivação

Se anteriormente, vimos como o ajuste de um modelo seguindo o Bayes Ingênuo Gaussiano pode nos auxiliar na tarefa de modelarmos covariáveis contínuas, é possível intuir uma certa limitação deste método: nem sempre covariáveis contínuas seguem a Distribuição Normal, de forma que uma abordagem do tipo seja considerada inadequada.

Para amostras relativamente grandes, o Teorema Central do Limite nos garante que a distribuição dos dados padronizados estará próxima o suficiente de uma distribuição Normal Padrão e, para estes fins, o Bayes Ingênuo Gaussiano talvez já seja o suficiente. No entanto, para casos em que estas condições não estão satisfeitas, convem estudarmos outras abordagens que possam ser mais adequadas e *flexíveis* em suas suposições; para isso, temos o **Bayes Ingênuo Flexível**.

Descrito por John e Langley (1995), o Bayes Ingênuo Flexível difere do Bayes Ingênuo Gaussiano na estimativa da função densidade de probabilidade da covariável contínua: se anteriormente, estimamos os parâmetros de uma distribuição normal, agora estimamos a função de densidade via **Kernel**, ou KDE (*Kernel Density Estimation*). Veremos agora qual é a definição deste método de estimação e como ele nos serve.

3.3.2 Definição

O método de estimação de densidades via Kernel foi descrito, pela primeira vez, por Parzen (1962) em seu artigo *"On Estimation of a Probability Density Function and Mode"*. Sua principal definição que utilizaremos será a seguinte:

Definição 3.20 (Estimação de Densidades via Kernel) *Sejam $X_1, \ldots, X_n$ variáveis aleatórias independentes e identicamente distribuídas como uma variável X cuja função de distribuição $F(x) = \mathbb{P}(X \leq x)$ é absolutamente contínua. Então, a estimativa de sua densidade num ponto z qualquer pode ser dada por:*

$$\hat{f}(z) = \frac{1}{nh}\cdot\sum_{i=1}^{n} K\left(\frac{z - X_i}{h}\right),$$

onde h é um número positivo definido para fornecer a melhor estimativa $\hat{f}(u)$ e a função K é chamada de ***Função Kernel****.*

A estimativa apresentada, além disso, é assintoticamente não-viesada se h for escolhido como função de n de tal forma que:

$$\lim_{n\to\infty} h(n) = 0 \tag{3.21}$$

e se $K(z)$ atender as seguintes propriedades:

$$\sup_{-\infty<x<\infty} |K(z)| < \infty \tag{3.22}$$

$$\int_{-\infty}^{\infty} |K(z)|\ dz < \infty \tag{3.23}$$

$$\lim_{x\to\infty} |x \cdot K(z)| = 0 \tag{3.24}$$

$$\int_{-\infty}^{\infty} K(z)\ dz = 1 \tag{3.25}$$

A função Kernel pode assumir diversas formas. Ela é quem decide quais "pontos" de nosso banco de dados são utilizados em maior ou menor peso para o cálculo da densidade de um novo ponto x qualquer. Para este trabalho, iremos comparar o desempenho das seguintes funções kernel aplicadas ao contexto do algoritmo Bayes Ingênuo Flexível:

- **Kernel Gaussiana**: $K(z) = \frac{1}{\sqrt{2\pi}} \exp\{-0.5z^2\}$;
- **Kernel Uniforme (ou Kernel Retangular)**: $K(z) = 0.5$;
- **Kernel Parabólica (ou Kernel de Epanechnikov)**: $K(z) = \frac{3}{4}(1 - z^2)$.

Com isso, é fácil perceber o motivo do método ser Bayes Ingênuo *Flexível*: são diversas as maneiras de se estimar a densidade de covariáveis contínuas segundo este método, algumas fornecendo melhores predições do que outras. Por fim, para que a estimativa siga este método, basta escolhermos uma função Kernel e valor h, usá-los segundo a Definição 3.20 para obtermos $\hat{f}(z)$ e inserirmos a densidade estimada nas equações (3.14) e (3.15).

Capítulo 4

Revisão Metodológica

Neste capítulo, revisaremos outros algoritmos de classificação que serão comparados ao Bayes Ingênuo de maneira mais resumida. São eles **Árvores de Classificação** e **Regressão Logística**. Após isso, veremos de que forma podemos selecionar as covariáveis relevantes o bastante para fazerem parte de nosso modelo nos casos estudados.

4.1 Árvore de Classificação

Árvores de Classificação são uma sub-divisão de Árvores de Decisão (conjunto do qual as Árvores de Regressão também fazem parte). A diferença se dá pelo próprio nome: uma delas realiza a classificação de uma variável (ou seja, a variável resposta encontra-se num espaço discreto de valores) e a outra, a regressão (quando a variável resposta encontra-se num espaço contínuo de valores).

No geral, Árvores de Classificação são um método bastante interpretável para a predição de variáveis categóricas. São compostas de nós, ramos e folhas (por isso recebem o nome de árvores):

- Nós são particionamentos recursivos das covariáveis que denotam uma condição: se a condição for satisfeita, seguimos para a direita do nó. Caso contrário, seguimos para a esquerda;
- Ramos são os caminhos feitos a partir de cada nó. No exemplo dado acima, caso a condição de um nó é satisfeita, partimos para o ramo à sua direita, ou para o ramo à sua esquerda caso contrário;
- Folhas são o resultado de uma árvore de classificação: o raciocínio é passarmos por vários nós (ou condições) até se atingir uma folha: ela denota a classe predita do

conjunto de covariáveis que passou pelas condições anteriores.

Para melhor compreensão, na Figura 4.1 apresentaremos um exemplo ilustrativo de uma árvore de decisão com os elementos mencionados.

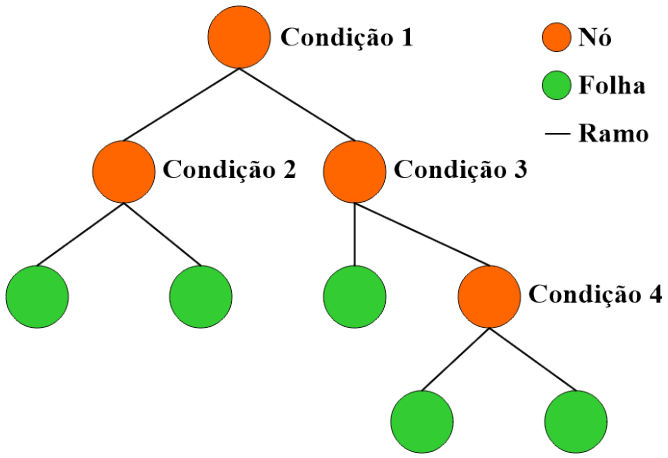

Figura 4.1: Exemplo da estrutura de uma Árvore de Classificação.

A sequência de condições atribuídas numa árvore de classificação cria um espaço de partições $R_1, R_2, \ldots, R_r$ nas covariáveis do banco de dados. Após estas divisões, a função de classificação pode ser dada por:

$$g(\mathbf{X}_{i.}) = moda\{Y_i : \mathbf{X}_{i.} \in R_r\} \tag{4.1}$$

Em outras palavras, atribui-se à i-ésima observação com covariáveis observadas $\mathbf{X}_{i.}$ a classe c que apareça mais vezes na partição R_r. No entanto, como decidirmos quais nós farão parte de nossa árvore de classificação? Nesta seção, veremos as etapas iniciais de construção de uma árvore de classificação; mais adiante, na Seção 4.3, veremos como é feito o aperfeiçoamento de uma árvore por um processo chamado de **poda**.

De início, ao criarmos o primeiro nó de uma árvore maior e mais complexa, a intenção é definirmos um corte $t_1 \in \mathbf{T}$ que divida alguma das covariáveis $\mathbf{X}_{.j}$ em regiões (R_1, R_2) de tal forma a minimizar o **índice de Gini**:

$$\sum_{R} \sum_{c \in \mathcal{C}} \hat{p}_{R,c}(1 - \hat{p}_{R,c}) \tag{4.2}$$

onde R representa o espaço de regiões delimitado pela árvore de classificação e $\hat{p}_{R,c}$ a proporção de observações que recebem a classe predita c na região R. Após a definição deste corte e das regiões, temos:

$$R_1 = \{\mathbf{X} : \mathbf{X}_{.j} < t_1\} \; e \; R_2 = \{\mathbf{X} : \mathbf{X}_{.j} \geq t_1\} \tag{4.3}$$

Após a definição do primeiro nó da árvore, escolhe-se a subdivisão de R_1 ou R_2 que, novamente, minimiza a Equação (4.2). A título de exemplo, suponhamos que a região

escolhida para ser dividida novamente foi R_2, agora denominada $R_2 = (R_{2,1}, R_{2,2})$. O mesmo processo anterior é realizado: escolhe-se um corte t_2 que seccione alguma das covariáveis $\mathbf{X_j}$ de tal forma a minimizar o índice de Gini. Após isso, teremos as seguintes regiões, por exemplo:

$$R_1 = \{\mathbf{X} : \mathbf{X}_{.j} < t_1\},\ R_{2,1} = \{\mathbf{X} : \mathbf{X}_{.j} \geq t_1, \mathbf{X}_{.k} < t_2\}\ e\ R_{2,2} = \{\mathbf{X} : \mathbf{X}_{.j} \geq t_1, \mathbf{X}_{.k} \geq t_2\} \tag{4.4}$$

O processo continua até que, segundo um parâmetro estabelecido previamente, cada uma das folhas contenha um número baixo de observações: em alguns algoritmos, define-se esse parâmetro como 5 observações para cada folha ou outro número. É importante salientarmos que toda a metodologia é realizada computacionalmente (e não seria prático que fosse o oposto); a título de curiosidade, é possível realizarmos todo o processo através da função *rpart* na linguagem computacional R.

4.2 Regressão Logística

A Regressão Logística talvez seja o algoritmo mais clássico relacionado à classificação de uma variável resposta binária. No entanto, pode ser utilizada para classificação de variáveis resposta com mais de duas classes (ainda que não seja este o foco do trabalho) com determinadas adaptações também.

Em suma, assume-se que a variável resposta Y possui Distribuição **Binomial**; o problema de classificação relacionado à regressão logística resume-se em encontrarmos uma estimativa para o parâmetro p através da seguinte equação:

$$\mathbb{P}(Y_i = 1|\mathbf{X}_{i.}) = \frac{\exp\{\beta_0 + \sum_{j=1}^{d} \beta_j x_{i,j}\}}{1 + \exp\{\beta_0 + \sum_{j=1}^{d} \beta_j x_{i,j}\}} \tag{4.5}$$

Trata-se de um **Classificador Plug-In**, ou seja: atribue-se a classe c ao conjunto de covariáveis X_i que maximize a equação (2.6). No entanto, é um método no geral mais limitante do que o Bayes Ingênuo no que diz respeito às suposições necessárias para ser bem ajustado/utilizado, segundo Dunn e Smyth (2018):

- Independência dos erros (distância entre os resultados observados e os preditos);
- Linearidade do log aplicado às covariáveis contínuas;
- Ausência de multicolinearidade (ou seja, independência das covariáveis condicional à variável resposta);

- Ausência de outliers ou pontos de influência.

A Regressão Logística assemelha-se ao Bayes Ingênuo na suposição de independência das covariáveis condicional à variável resposta. Por outro lado, assim como o Bayes Ingênuo, **pode fornecer predições sólidas ainda que as suposições não sejam totalmente atendidas.**

Para estimarmos os coeficientes da Regressão Logística, maximizamos sua função de verossimilhança, dada por:

$$L(y;(\mathbf{X},\beta)) = \prod_{i=1}^{n} \left(\frac{\exp\{\beta_0 + \sum_{j=1}^{d} \beta_j x_{i,j}\}}{1 + \exp\{\beta_0 + \sum_{j=1}^{d} \beta_j x_{i,j}\}} \right)^{y_i} \cdot \left(\frac{1}{1 + \exp\{\beta_0 + \sum_{j=1}^{d} \beta_j x_{i,j}\}} \right)^{1-y_i} \tag{4.6}$$

Assim, o vetor de estimativas ótimas dos parâmetros β será dado por:

$$\beta^* = \arg\max_{\beta} L(y;(\mathbf{X},\beta)) \tag{4.7}$$

Além de realizarmos as estimativas dos parâmetros β, também é de interesse nosso definirmos quais são as covariáveis de maior influência no contexto de estudo para serem incluídas no modelo ajustado. Para isso, estudam-se métodos de **seleção de covariáveis**: para a Regressão Logística, veremos a **penalização de covariáveis via Lasso**, além de métodos referentes às Árvores de Classificação e ao Bayes Ingênuo.

4.3 Seleção de Covariáveis

4.3.1 Árvore de Classificação

Tanto em Regressão quanto em Classificação, Árvores de Decisão possuem a sua "seleção de covariáveis" (chamada de poda da árvore) como parte essencial do processo de construção de seu modelo preditivo. Isso se dá pois, com uma árvore grande e complexa feita através do processo descrito na Seção 4.1, há um grande risco de que o modelo lide bem com predições no conjunto de treino mas não no conjunto de teste. Esse tipo de condição é consequência de uma alta variância nos estimadores e do superajuste do modelo.

Para que esta condição seja evitada, é realizado o processo de poda através dos seguintes passos:

1. Retira-se, um a um, os nós da árvore e calcula-se o risco estimado (no caso da Árvore de Classificação, descrito na Equação (2.4)) através do conjunto de **teste**;

2. Escolhe-se a árvore criada através do passo anterior que apresente menor risco;

3. Repetem-se os passos 1 e 2 até que não haja melhora no poder preditivo da árvore.

Em alguns casos, no entanto, vale mencionarmos que a árvore com o melhor poder preditivo nem sempre é a mais simples ou interpretável: para determinadas finalidades, uma árvore com pior poder preditivo, mas com menos variáveis e maior interpretabilidade pode ser escolhida, Computacionalmente falando, alguns algoritmos realizam este processo automaticamente ao ajustarem um modelo via árvore de decisão; outros requerem que a poda seja realizada manualmente.

4.3.2 Regressão Logística via Lasso

Estimativas via Lasso, descritas pela primeira vez por Tibshirani (1996) no artigo *"Regression Shrinkage and Selection via the Lasso"*, tratam de uma forma de estimarmos coeficientes de um modelo linear a fim de evitar o superajuste e fornecer predições melhores do que as fornecidas via Mínimos Quadrados. Por definição, estas estimativas podem ser obtidas na Regressão Logística.

Definição 4.8 (Seleção de Covariáveis via Lasso) *Definamos* $(\mathbf{X}_i, Y_i)$, $i = 1, 2, \ldots, n$, *onde* $\mathbf{X}_i = (X_{i,1}, X_{i,2}, \ldots, X_{i,d})$ *são as variáveis preditoras de um modelo e* Y_i, *a variável resposta. Suponhamos que as variáveis sejam independentes (ou que* Y *seja independente condicionado à* $\mathbf{X}$*) e que* $X_{i,j}$ *sejam padronizados de tal forma que* $\sum_{i=1}^{n} X_{i,j}/N = 0$ *e* $\sum_{i=1}^{n} X_{i,j}^2/N = 1$.

Definindo $\hat{\beta} = (\hat{\beta}_1, \hat{\beta}_2, \ldots, \hat{\beta}_d)$ *como as estimativas dos parâmetros relacionados às variáveis preditoras do modelo, as estimativas* $(\hat{\beta}_0, \hat{\beta})$ *via lasso são obtidas por:*

$$(\hat{\beta}_0, \hat{\beta}) = \arg\min_{(\hat{\beta}_0, \hat{\beta})} \left\{ \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{d} \beta_j x_{i,j} \right)^2 \right\} \text{ sujeito a } \sum_{j=1}^{d} |\beta_j| \leq t$$

Na equação 4.8, t é escolhido de tal forma a fornecer melhores predições via validação no conjunto de treino do banco de dados utilizado. O modelo resultante é melhor, mais interpretável (covariáveis menos importantes resultam numa estimativa próxima de 0, por exemplo) e evita o superajuste.

4.3.3 Bayes Ingênuo

Ao contrário dos dois métodos mencionados anteriormente, a seleção de covariáveis para a composição de um modelo criado via Bayes Ingênuo não é tão intuitiva ou bem definida.

Entre os métodos existentes, podemos citar a seleção de variáveis via Teste Qui-Quadrado. Em resumo, ela considera como importantes variáveis que sejam altamente correlacionadas com a variável resposta. No entanto, não considera a dependência entre variáveis preditoras ou o desempenho do modelo como fator de seleção.

Em primeiro lugar, utilizaremos um método descrito por Ratanamahatana e Gunopulos (2003) no artigo *"Feature selection for the naive bayesian classifier using decision trees"* que consiste basicamente em:

1. Ajusta-se uma Árvore de Decisão aos dados, realizando sua poda e observando as covariáveis selecionadas;

2. Ajusta-se um modelo via Bayes Ingênuo com as mesmas variáveis.

O método descrito é tido como consistente e composto de boas predições para o conjunto de teste. Por outro lado, também selecionaremos variáveis através de outro método:

1. Ajusta-se uma Regressão Logística com seleção de covariáveis via Lasso e observa-se as variáveis preditoras selecionadas;

2. Ajusta-se um modelo via Bayes Ingênuo com as mesmas variáveis.

A finalidade será comparar um modelo via Bayes Ingênuo completo (sem seleção de variáveis), um modelo via Bayes Ingênuo com seleção de covariáveis via Árvore de Decisão e outro via Regressão Logística com Lasso.

Capítulo 5

Banco de Dados

Neste capítulo, apresentaremos o banco de dados que será utilizado para aplicação prática dos conceitos definidos até aqui. Essa apresentação será dividida em Contextualização (do que se trata o banco de dados e em que contexto está inserido), pré-Processamento (como os dados originais foram organizados e tratados, e de onde foram extraídos), descrição (variáveis disponíveis no banco de dados após tratamento) e Análise Exploratória (visualização dos dados por diversos meios a fim de identificar possíveis resultados).

5.1 Contextualização

A série de jogos Pokémon, lançada em 1996 no Japão, é uma das propriedades intelectuais mais lucrativas de nosso século. Pokémons são animais dos mais diversos tipos e habitats regionais utilizados para amizade, companheirismo e até mesmo batalhas esportivas no universo ficcional da série. Alguns deles, inclusive, são tidos como divindades ou seres mitológicos, considerados bastante poderosos. O objetivo destas batalhas é desmaiar (ou reduzir seus pontos de vida até 0) os Pokémons do oponente antes que todos os seus Pokémons. A Figura 5.1 demonstra a interface destas batalhas em um dos jogos.

Figura 5.1: Exemplo de tela de batalha nos jogos Pokémon (Game Freak, 2004).

Nos jogos principais, você é uma criança/pré-adolescente que recebe um Pokémon inicial de presente e almeja receber o título de Campeão de sua região, capturando novas criaturas e batalhando até o final de sua jornada. Você encontra bichinhos de até dois tipos dentre 19 tipos elementais distintos (entre eles, água, fogo e folha, por exemplo, com vantagens e desvantagens entre si) com ataques distintos que podem ser aprendidos, tornando possíveis inúmeras estratégias.

Para o banco de dados utilizado neste trabalho, consideraremos batalhas 1v1 (um Pokémon contra o outro) ao contrário das batalhas 6v6 que normalmente ocorrem nos jogos (ou seja, um time de Pokémons contra o outro). Veremos que atributos dos jogos serão utilizados para a análise destes embates.

5.2 Pré-Processamento

O banco de dados original está disponível no repositório *"Pokémon Challenge"* (Kaggle, 2015) e contém dados referentes a 800 Pokémons distintos. Para este trabalho, utilizaremos as planilhas *combats.csv* e *pokemon.csv*, apresentadas nas Tabelas 5.1 e 5.2 respectivamente.

Tabela 5.1: Descrição das colunas do banco de dados *combats*.

Variável	Descrição
atacante	ID do Pokémon atacante (que ataca primeiro)
defesa	ID do Pokémon defensor (que ataca por último)
win	Indicador se o Pokémon atacante venceu (1 se sim e 0 caso contrário)

Tabela 5.2: Descrição das colunas do banco de dados *pokemon*.

Variável	Descrição
ID	ID do Pokémon
nome	Nome do Pokémon
tipo_1	Tipo primário do Pokémon
tipo_2	Tipo secundário do Pokémon
hp	Status base de pontos de vida do Pokémon
atk	Status base de pontos de ataque do Pokémon
def	Status base de pontos de defesa do Pokémon
satk	Status base de pontos de ataque especial do Pokémon
sdef	Status base de pontos de defesa especial do Pokémon
spd	Status base de pontos de velocidade do Pokémon
geracao	Variável categórica indicando qual a geração do Pokémon (1 até 6)
lendario	Variável indicadora de se o Pokémon é Lendário (1 se sim e 0 caso contrário)

Além disso, também foi utilizada uma tabela de vantagens e desvantagens entre todos tipos elementais para calcularmos essas variáveis entre os Pokémons atacante e defensor, como a tabela representada na Figura 5.2. Para um Pokémon com dois tipos (suponhamos, Gelo e Água), sua fraqueza/resistência a determinado tipo é calculada através da multiplicação das fraquezas/resistências de seus dois tipos primário e secundário.

A etapa de pré-processamento consistiu em:

1. A partir dos IDs presentes na tabela *combats*, determinar os nomes, tipos, status e variáveis categóricas dos Pokémon atacantes e defensores presentes na tabela *combats*, em especial a variável resposta *win*;

2. Criamos, a partir disso, as variáveis **diferença dos atributos**: caso a variável "diferença de vida" for positiva, os pontos de vida do Pokémon atacante são maiores do que os do defensor, e vice-versa;

3. Por fim, foram criadas as variáveis de resistência com base na Tabela 5.2.

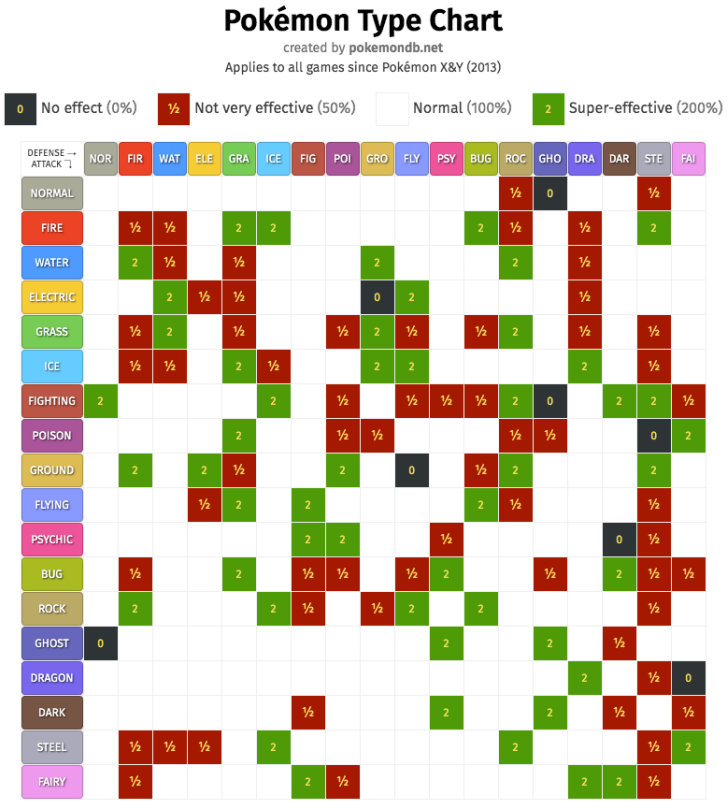

Figura 5.2: Tabela de vantagens e desvantagens entre tipos de Pokémons. (Database, 2022)

O processo de junção, tratamento e pré-processamento das tabelas foi todo feito através do software estatístico R (R Core Team, 2021), com auxílio especial de pacotes do tidyverse (Wickham *et al.*, 2019). Os códigos utilizados encontram-se em anexo apropriado neste trabalho, bem como em repositório no GitHub (Dogo, 2022).

Abaixo segue uma representação relacional do banco de dados resultante. Na Seção 5.3, veremos qual será seu formato final.

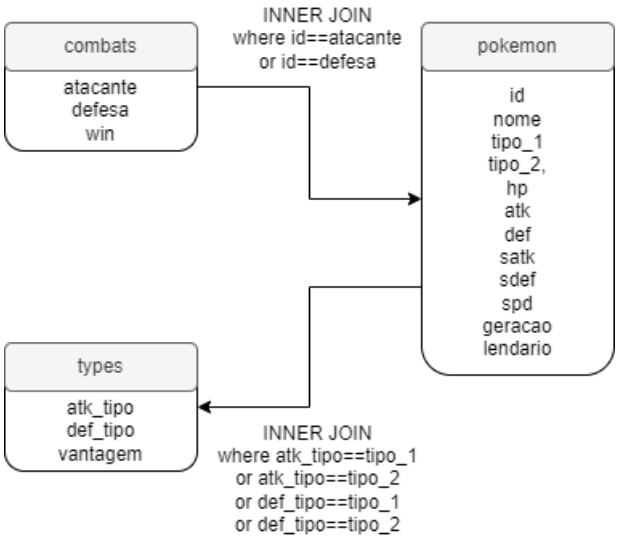

Figura 5.3: Fluxograma relacional das tabelas utilizadas para o pré-processamento do banco de dados.

5.2.1 Detecção de Valores Atípicos

Valores atípicos dentro de um banco de dados (também chamados de *Outliers*) são observações que distoam das demais por diversos motivos, entre eles: erros de medição, circunstâncias externas atípicas que influenciam nas medições, entre outros. Quando tratamos de problemas de aprendizado de máquina no geral, é costumeiro verificarmos se esses tipos de observações existem dentro de nosso banco de dados para não influenciarem negativamente nossos resultados. Lembrando do que foi dito no Exemplo 2.9, por exemplo, pode ser que uma determinada época trouxe um número incomum de derrotas sem que houvesse uma mudança justificável nas demais covariáveis. A decisão de retirar ou manter estas observações é situacional, sem que haja uma única abordagem para esse problema.

Dentre as metodologias utilizadas para a detecção destas observações com valores atípicos, colocaremos em prática a **Distância de Mahalanobis**, descrita pela primeira vez por Mahalanobis (1930).

Definição 5.1 (Distância de Mahalanobis) *Suponhamos uma distribuição Q no espaço $\mathbb{R}^d$ com parâmetros de média $\boldsymbol{\mu} = (\mu_1, \mu_2, \ldots, \mu_d)$ e matriz de variância-covariância positiva e definida S. A distância de Mahalanobis de uma observação $\mathbf{x}_{i.}$ do centro da distribuição Q é dada por:*

$$d_M(\mathbf{x}_{i.}; Q) = \sqrt{(\mathbf{x}_{i.} - \boldsymbol{\mu})^T S^{-1} (\mathbf{x}_{i.} - \boldsymbol{\mu})}$$

Com isso, calcula-se primeiro o centro da distribuição Q para determinado banco de

dados; em seguida, calcula-se a distância de todas as observações com relação ao centro encontrado. Dado um limite estabelecido (proveniente do quantil de uma distribuição, por exemplo), se a distância de uma i-ésima observação for maior que este limite, ela é classificada como um *outlier*.

Para nosso trabalho, será utilizado um limite baseado em um quantil distribuição **Qui-Quadrado** ($\chi^2_{1-\alpha;d}$) com $\alpha = 0.001$. O número e percentual relativo de valores atípicos encontrados foi:

Tabela 5.3: Descrição da Detecção de Outliers.

Quantil	**Número de outliers**	**%**
34.53	1233	0.02466

Temos que aproximadamente 2.5% das observações do banco de dados foram tratadas como valores atípicos segundo a distância de mahalanobis. Com isso, e ainda que possam não ter ocorrido devido a um erro de medição, é conveniente **retirarmos estas observações**. Assim, evitamos o impacto negativo dos *outliers* sem que retiremos uma parcela muito grande de nosso banco de dados.

5.3 Descrição e Objetivo

Com a fase de pré-processamento dos dados pronta, temos as seguintes colunas no banco de dados final, devidamente nomeado de *pokebattle*:

- **win**: Categórica, indicativa se o Pokémon atacante venceu (win=1) ou perdeu (win=0) a luta;
- **nome_atk**: Categórica, nome do Pokémon atacante;
- **nome_def**: Categórica, nome do Pokémon defensor;
- **tipo1_atk**: Categórica, tipo primário do Pokémon atacante;
- **tipo2_atk**: Categórica, tipo secundário do Pokémon atacante;
- **tipo1_def**: Categórica, tipo primário do Pokémon defensor;
- **tipo2_def**: Categórica, tipo secundário do Pokémon defensor;
- **diff_hp**: Numérica, diferença do status base de vida do Pokémon atacante menos o defensor;

- **diff_atk**: Numérica, diferença do status base de ataque do Pokémon atacante menos o defensor;
- **diff_def**: Numérica, diferença do status base de defesa do Pokémon atacante menos o defensor;
- **diff_satk**: Numérica, diferença do status base de ataque especial do Pokémon atacante menos o defensor;
- **diff_sdef**: Numérica, diferença do status base de defesa especial do Pokémon atacante menos o defensor;
- **diff_spd**: Numérica, diferença do status base de velocidade do Pokémon atacante menos o defensor;
- **res_tipo1**: Categórica, resistência do Pokémon atacante ao tipo primário do Pokémon defensor, sendo 0 muito resistente (ou imune) e 4, pouco resistente;
- **res_tipo2**: Categórica, resistência do Pokémon atacante ao tipo secundário do Pokémon defensor, sendo 0 muito resistente (ou imune) e 4, pouco resistente;
- **gen_atk**: Categórica, geração de origem do Pokémon atacante, indo de 1 até 6;
- **gen_def**: Categórica, geração de origem do Pokémon defensor, indo de 1 até 6;
- **leg_atk**: Categórica, indicador se o Pokémon atacante é lendário (leg_atk=1) ou não (leg_atk=0);
- **leg_def**: Categórica, indicador se o Pokémon defensor é lendário (leg_def=1) ou não (leg_def=0),

O objetivo, portanto, será classificarmos a variável **win** corretamente com base em um conjunto de covariáveis dentre as descritas acima. Mas antes disso, na Seção 5.4 a seguir, realizaremos uma análise exploratória com a finalidade de identificarmos possíveis resultados.

5.4 Análise Exploratória

Nesta seção, realizaremos uma análise exploratória dividida por etapas para identificarmos possíveis resultados quando o ajuste dos modelos de classificação for feito, principalmente no que diz respeito às covariáveis que serão selecionadas para a composição

destes modelos: numa análise exploratória, esperamos comportamentos que indiquem a importância destas variáveis.

No entanto, verificaremos se os dados são balanceados antes de prosseguirmos.

5.4.1 Balanceamento dos Dados

Como visto na Seção 2.5.1, o desbalanceamento de nossos dados indica problemas que acarretam diretamente no desempenho preditivo de futuros modelos de classificação binária. Ainda que a proporção de dados em cada classe não seja o problema, é o diagnóstico que nos leva a considerar abordagens como a troca da função de perda, entre outras.

Tabela 5.4: Percentual de observações com win=0 e win=1 no banco de dados.

win	**0**	**1**
% de obs.	0.5294769	0.4705231

Na Tabela 5.4 apresentada, observações em que o Pokémon atacante perdeu uma partida (ou seja, em que $win = 0$) constituem cerca de 52.94% do banco de dados, enquanto que observações em que o Pokémon atacante ganhou uma partida (ou seja, em que $win = 1$) constituem cerca de 47.06% das observações restantes.

Com isso, temos que não há uma quantidade exacerbada de observações numa classe, se comparada com outra (considerando a diferença de aproximadamente 5 pontos percentuais). Logo, não há a necessidade de mudarmos a função de perda, corte de probabilidade do classificador visto na Equação (2.6) ou de realizarmos a reamostragem do banco de dados.

5.4.2 Correlação de Pearson entre variáveis numéricas

É de interesse nosso entendermos como que nossas covariáveis se relacionam, ainda mais no caso das covariáveis contínuas. Uma abordagem adequada para tal é o cálculo da Correlação de Pearson para identificarmos correlação linear entre as variáveis. Para isso, consideremos 0.7 como uma correlação relativamente alta.

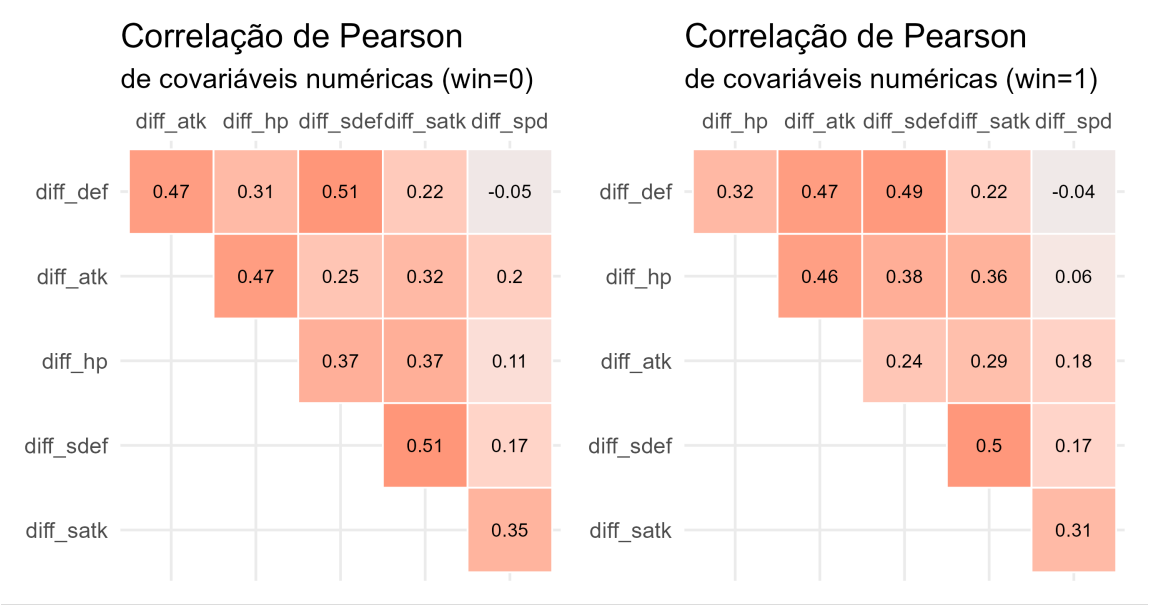

Figura 5.4: Matriz de Correlação de Pearson entre covariáveis numéricas do banco de dados *pokebattle* para $win = 0$ e para $win = 1$.

Na Figura 5.4, temos que as maiores correlações entre as variáveis numéricas ocorre entre as variáveis *diff_satk* e *diff_sdef* e entre as variáveis *diff_sdef* e *diff_atk*, com um valor igual a 0.51 quando $win = 0$. Para $win = 1$, não existem diferenças grandes dos valores observados.

Para a primeira correlação mencionada, contextualmente isso faz sentido: Pokémons com ataque especial alto normalmente são aqueles focados menos em lutas de poderes físicos, o que implicaria numa defesa especial alta, e vice-versa. Para *diff_sdef* e *diff_atk*, por outro lado, temos que Pokémons com foco em ataque físico alto podem apresentar defesa especial robusta. No entanto, nenhuma das correlações apresentadas supera o valor previamente estabelecido de 0.7.

5.4.3 Densidade estimada das variáveis numéricas

Nas Seções 3.2 e 3.3, vimos que, quando temos covariáveis contínuas ao invés de discretas e/ou categóricas, precisamos utilizar abordagens com a finalidade de estimar a densidade destas covariáveis. Para isso, existe a **aproximação normal** (como visto no método Bayes Ingênuo Gaussiano) e a **estimação via kernels** (como visto no método Bayes Ingênuo Flexível).

A seguir, compararemos a estimativa das densidades das covariáveis numéricas no banco de dados feita através da aproximação normal e dos kernels gaussiano, uniforme (ou retangular) e parabólico (ou de Epanechnikov):

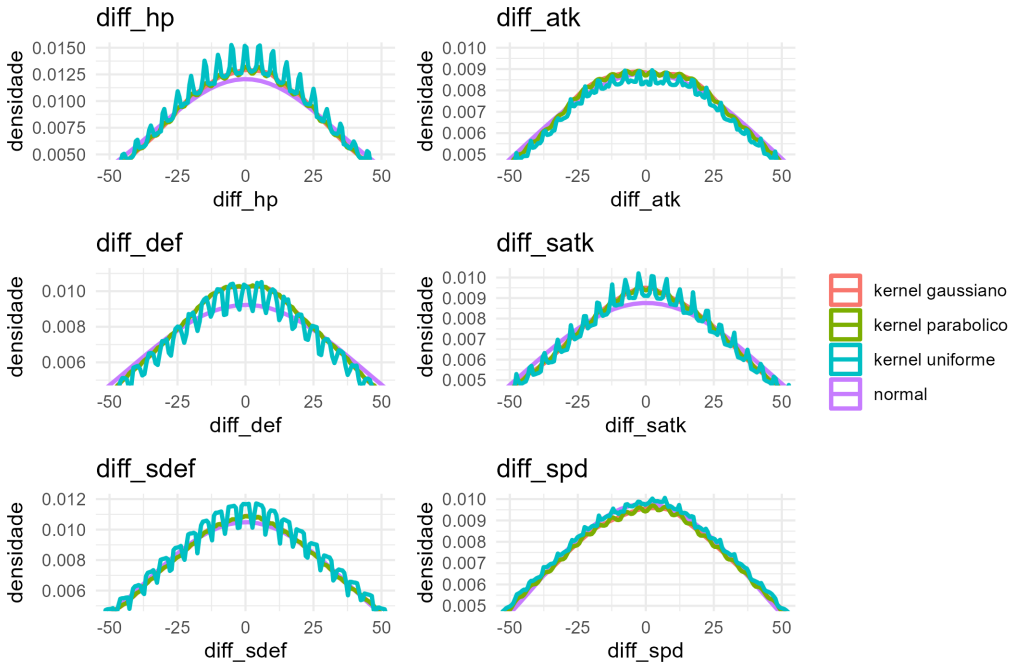

Figura 5.5: Comparação das Estimativas de Densidade das Covariáveis Contínuas.

Na Figura 5.5, temos que estimativas de densidade realizadas através do kernel uniforme, no geral, parecem possuir variância maior e serem mais ruidosas; por outro lado, variam ao redor das outras estimativas, sem estarem distantes. A aproximação normal aparentou desempenho pior, principalmente nas variáveis *diff_hp*, *diff_def* e *diff_satk*, onde podemos observar a curva roxa mais distante das demais.

Os kernels gaussiano e parabólico aparecem bastante próximos em todos os casos, possivelmente indicando que forneçam estimativas melhores do que as demais aproximações. Em especial, na variável *diff_spd*, observamos que todas as estimativas estão bem próximas.

5.4.4 Comparação de variáveis em relação à *win*

Nesta etapa, iremos comparar as covariáveis agrupadas conforme a variável resposta *win* de modo a identificar quais delas podem ser consideradas mais importantes visualmente e possivelmente serão selecionadas segundo métodos descritos na Seção 4.3.

Primeiramente, faremos isto com as covariáveis contínuas (utilizando a estimativa obtida através do kernel gaussiano). Na Figura 5.6, com a exceção da variável *diff_spd*, não há distinção clara entre densidades quando o Pokémon atacante perde ou ganha. Visualmente, isso nos indica que tais covariáveis não sejam tão importantes na predição de vitória ou derrota.

Por outro lado, quando visualizamos as densidades estimadas de vitória e derrota da

covariável *diff_spd*, há uma distinção forte e clara: situações onde o Pokémon atacante possui maior atributo base de velocidade (ou seja, quando *diff_spd*> 0) quase sempre indicam vitória, enquanto que situações onde o Pokémon atacante possui menor velocidade (ou seja, quando *diff_spd*< 0) quase sempre indicam derrota. Esta covariável, muito provavelmente, terá um papel importante na predição da variável resposta *win*.

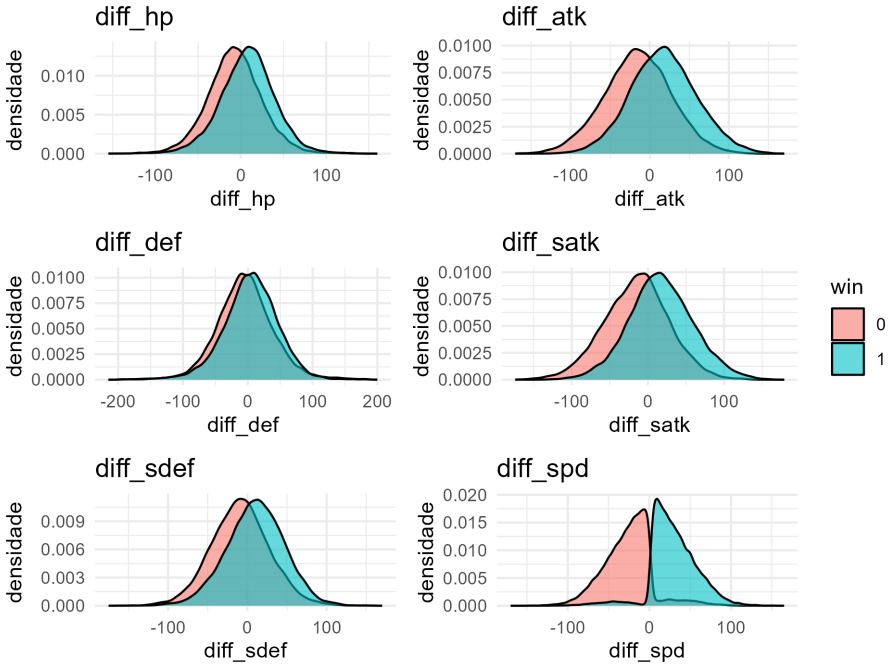

Figura 5.6: Densidade das covariáveis contínuas agrupadas conforme a variável resposta *win*.

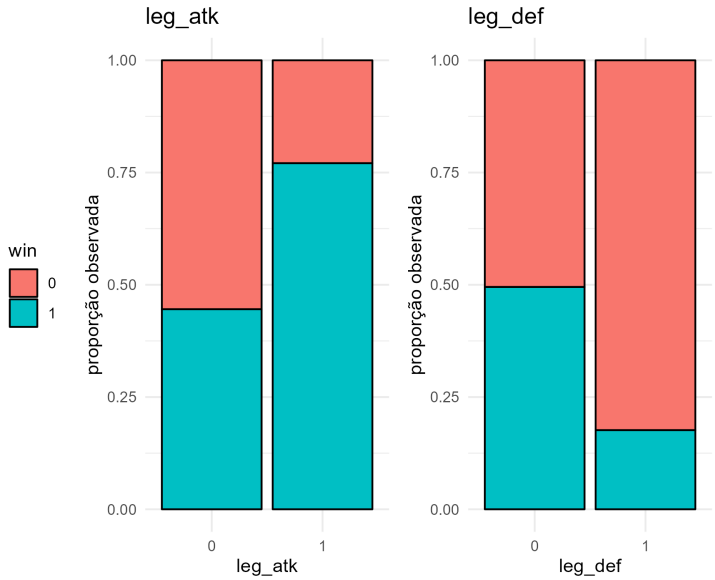

Figura 5.7: Comparação das variáveis *leg_atk* e *leg_def* agrupadas conforme a proporção de vitórias e derrotas.

Na Figura 5.7, temos que se o Pokémon atacante não for lendário (ou seja, se *leg_atk* = 0), ele mais perderá do que ganhará dado a primeira coluna; o contrário, em menor escala, também parece se aplicar, com proporção de vitórias maior quando o Pokémon atacante for lendário.

Por outro lado, quando o Pokémon defensor não é lendário (ou seja, se *leg_def* = 0), não parece haver grandes diferenças entre a proporção de vitórias e derrotas do Pokémon atacante; para quando o Pokémon defensor é lendário, temos uma proporção observada de derrotas do Pokémon atacante maior do que vitórias. No geral, ainda que as duas variáveis apresentem certa distinção na proporção de vitórias ou derrotas, é possível que ela não seja boa para compor um modelo de classificação da variável resposta *win*.

Na Figura 5.8, não temos grande distinção da proporção de vitórias ou derrotas: a maior diferença para a variável *gen_atk* aparece quando o Pokémon atacante faz parte das gerações 2 e 3, onde a proporção de derrotas é pouco maior do que 50%. O mesmo é válido, na variável *gen_def*, para as gerações 1, 4 e 5. Temos, visualmente, um forte indício de que estas duas variáveis não serão cruciais na construção de nossos modelos.

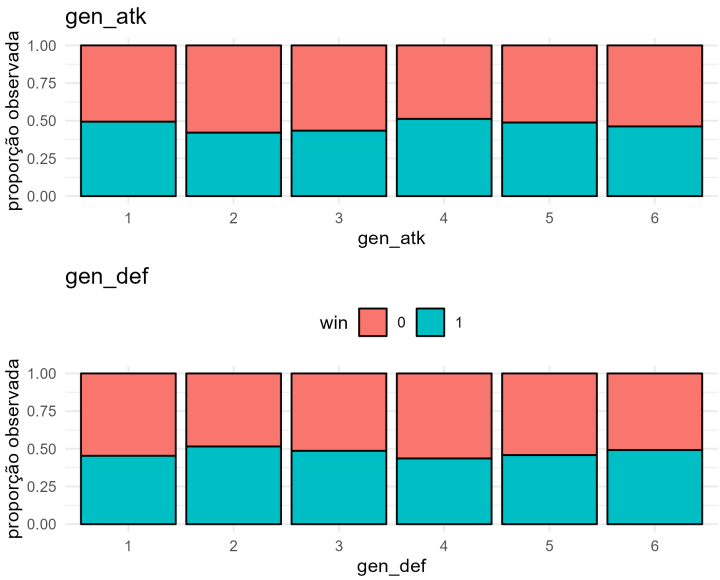

Figura 5.8: Comparação das variáveis *gen_atk* e *gen_def* agrupadas conforme a proporção de vitórias e derrotas.

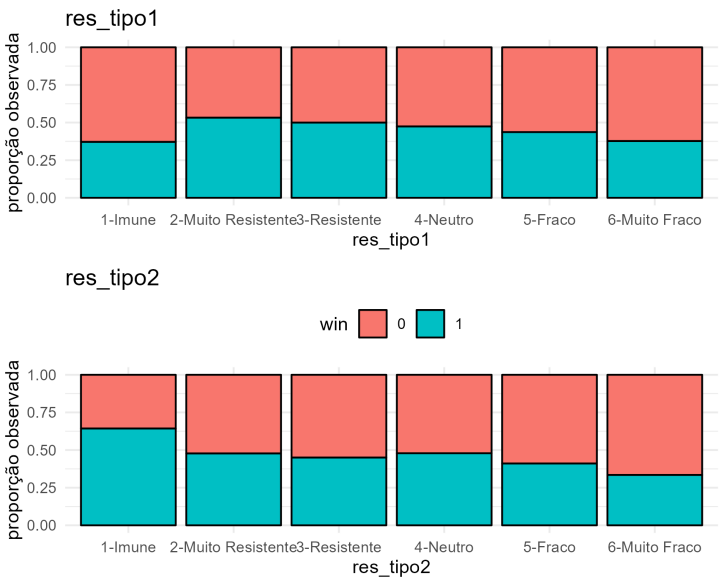

Figura 5.9: Comparação das variáveis *res_tipo1* e *res_tipo2* agrupadas conforme a proporção de vitórias e derrotas.

Por fim, na Figura 5.9, **exceto quando o Pokemon atacante é imune ao tipo primário do Pokemon defensor**, temos um padrão onde, quanto mais fraco o Pokemon é com relação à tipagem do Pokemon defensor, menor é a proporção de vitórias; com respeito ao tipo secundário, o Pokemon atacante venceu em aproximadamente 62% das vezes em que era Imune ao tipo do defensor ou 50% quando era muito resistente. Quando era muito fraco, por exemplo, o Pokemon atacante venceu apenas 40% das vezes. É possível que essa relação seja relevante para a construção futura de nossos modelos.

5.4.5 Breve conclusão

Após uma breve análise exploratória de nossas principais covariáveis, temos que:

- Nossos dados se encontram balanceados, sem que haja a necessidade de ajustes quanto à função de perda utilizada ou à probabilidade utilizada como corte no classificador de bayes;

- Não há correlação expressiva entre alguma das covariáveis numéricas de modo que a suposição de independência condicional à variável resposta das covariáveis do Bayes Ingênuo não seja cumprida;

- Entre as estimativas utilizadas para a densidade de nossas covariáveis contínuas, os melhores resultados visuais foram obtidos pelos kernel parabólico e gaussiano;

- Dentre as covariáveis contínuas, a que apresenta maior grau visual de distinção entre vitória e derrota do Pokémon atacante é a diferença de velocidade (*diff_spd*), com fortes indícios de que ela seja selecionada para nossos modelos de classificação;

- As variáveis indicadoras de Pokémons lendários atacantes ou defensores não apresentam grande distinção entre vitória e derrota, salvo caso em que o Pokémon atacante não é lendário (quando temos mais derrotas do que vitórias);

- As variáveis indicadoras de geração dos Pokémons atacantes e defensores apresentam distinções maiores entre vitória e derrota apenas em alguns casos (quando o Pokémon atacante é das gerações 2 e 3 ou quando o Pokémon defensor é das gerações 1, 4 ou 5).

- As variáveis indicadoras de resistências e fraquezas aos tipos do Pokemon defensor apresentam, ainda que pequena, distinção na proporção de vitórias ou derrotas se compararmos casos em que ele é imune ou muito resistente contra casos em que ele é muito fraco.

A seguir, no Capítulo 6, realizaremos os ajustes propriamente ditos de nossos modelos de classificação de modo a verificarmos quais deles apresentam melhor desempenho e, com base nas nossas conclusões da análise exploratória, veremos quais resultados eram e não eram esperados.

Capítulo 6

Resultados

Para a presente etapa do trabalho, dividiram-se os resultados nas seguintes etapas:

1. Inicialmente, comparam-se os modelos completos ajustados via Regressão Logística, Bayes Ingênuo Gaussiano e Bayes Ingênuo Flexível;

2. Considerando a existência da variável *diff_spd*, compararemos os melhores dos modelos ajustados (sob aspecto preditivo) via:

    - Árvore de Classificação com poda;
    - Regressão Logística Completa e via Lasso;
    - Bayes Ingênuo Gaussiano Completo, com covariáveis selecionadas nos ajustes via Regressão Logística com Lasso e via Árvore de Classificação com poda;
    - Bayes Ingênuo Flexível Completo, com covariáveis selecionadas nos ajustes via Regressão Logística com Lasso e via Árvore de Classificação com poda.

3. Desconsiderando a existência da variável *diff_spd*, repetem-se os passos em (2).

Para comparação de medidas avaliativas dos modelos apresentados, utilizaremos a divisão simples via conjuntos de treino e teste (com proporção de 70%/30%, respectivamente), como descrito na Subseção 2.4.1. Para casos em que mais de um modelo é possível, escolhe-se o melhor modelo dentro dos métodos via divisão simples **no já selecionado conjunto de treino**, via treino e validação (com proporção de 60%/40%, respectivamente) para ser comparado com as demais metodologias. Isso é feito para não compararmos muitos modelos de uma só vez e, assim, evitar o **superajuste**, nos dando mais segurança de nossas conclusões acerca de qual é o melhor modelo (Izbicki e dos Santos, 2020).

6.1 Comparação inicial dos modelos completos considerando diff_spd

Antes de realizados os ajustes, devemos verificar se as divisões dos conjuntos de treino-teste e treino-validação estão balanceadas através da Tabela 6.1:

Tabela 6.1: Verificação do balanceamento da proporção de observações nos conjuntos de treino e teste (e treino e validação) com *diff_spd*.

Conjunto	**% win=1**	**% win=0**
Treino	0.4693	0.5306
Teste	0.4732	0.5267
Treino	0.4670	0.5330
Validação	0.4729	0.5271

Em ambas as divisões, temos não haver desbalanceamento grave. Com isso, podemos prosseguir com a análise usual dos modelos ajustados, começando pela escolha do melhor modelo ajustado via Bayes Ingênuo Flexível: na Figura 6.1, **não é possível distinguirmos visualmente o desempenho dos três modelos distintos**. Com isso, nos resta verificarmos as estimativas das métricas avaliativas apresentadas na Tabela 6.2:

- O modelo ajustado via Kernel Uniforme apresenta maior acurácia dentre os três, bem como o menor risco, ainda que esta diferença se dê apenas na quarta casa decimal. Também superou o desempenho dos outros modelos no que diz respeito ao valor preditivo positivo e especificidade;

- O modelo ajustado via Kernel Gaussiano apresenta maior sensibilidade, valor preditivo negativo e AUC;

- O modelo ajustado via Kernel Parabólico não superou os outros dois em nenhuma das métricas.

Em outras palavras, o modelo ajustado via Kernel Uniforme classifica corretamente observações resultantes em derrota do Pokémon atacante além de, quando classifica observações como vitórias, as classifica corretamente mais vezes (ver especificidade e VPP, respectivamente). O oposto ocorre com o modelo ajustado via Kernel Gaussiano.

Tabela 6.2: Métricas Avaliativas estimadas para Kernels do modelo ajustado via Bayes Ingênuo Flexível, via conjunto de validação.

Ajuste	Acurácia	Risco	Sensibilidade	Especificidade	VPP	VPN	AUC
Gaussiano	0.8963	0.1037	0.9025	0.8892	0.9030	0.8886	0.9510
Uniforme	0.8964	0.1036	0.8965	0.8963	0.9081	0.8834	0.9505
Parabolico	0.8953	0.1047	0.9014	0.8883	0.9022	0.8874	0.9504

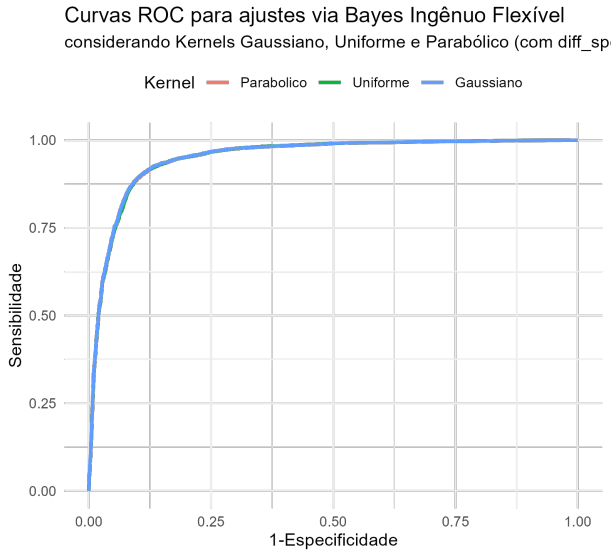

Figura 6.1: Curva ROC dos modelos ajustados via Bayes Ingênuo Flexível para diferentes Kernels, considerando o conjunto de validação do modelo completo.

Ainda que a diferença entre ajustes seja mínima, **decidimos por escolher o modelo ajustado via Kernel Gaussiano** para a próxima etapa, ou seja, a comparação com os modelos completos ajustados via Regressão Logística e Bayes Ingênuo Gaussiano via métricas estimadas no conjunto de teste:

Tabela 6.3: Métricas Avaliativas estimadas para modelos completos ajustados via Regressão Logística, BI Gaussiano e BI Flexível, estimadas via conjunto de teste.

Ajuste	Acurácia	Risco	Sensibilidade	Especificidade	VPP	VPN	AUC
Gaussiano	0.8144	0.1856	0.8203	0.8079	0.8262	0.8015	0.8931
Flexível	0.8968	0.1032	0.9058	0.8868	0.8990	0.8943	0.9534
Logística	0.8906	0.1094	0.9013	0.8787	0.8921	0.8888	0.9311

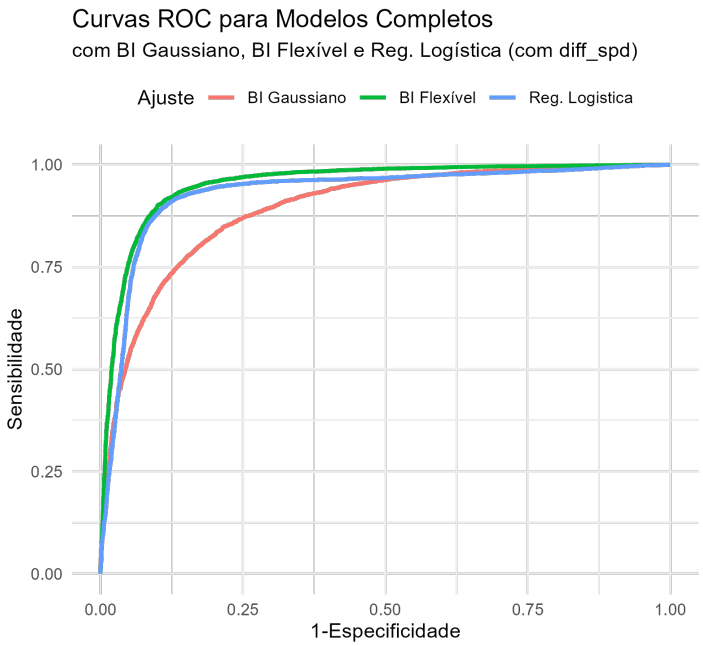

Figura 6.2: Curva ROC dos melhores modelos ajustados via BI Flexível, BI Gaussiano e Regressão Logística, considerando o conjunto de teste do modelo completo.

Como é possível vermos na Figura 6.2, o modelo ajustado via Bayes Ingênuo Flexível (via Kernel Gaussiano) apresentou melhor desempenho quando comparado com os demais, com a sua curva em verde sendo a mais próxima do canto superior esquerdo do gráfico. O segundo melhor modelo foi o ajustado via Regressão Logística, seguido pelo modelo ajustado via Bayes Ingênuo Gaussiano.

Quando olhamos para as métricas da Tabela 6.3, temos que o modelo ajustado via Bayes Ingênuo Flexível acertou aproximadamente 90% das classificações feitas no conjunto de teste, pouco mais do que o modelo via Regressão Logística, que acertou 89%.

Quando falamos das outras métricas, o Bayes Ingênuo Flexível também acerta mais classificações de vitórias e derrotas (vide sensibilidade, especificidade, VPN e VPP), além de apresentar um valor AUC maior do que os demais, por pouco mais do que 2 pontos percentuais (como visto na Figura 6.2).

Com isso, e considerando modelos com todas as variáveis, o ajustado via Bayes Ingênuo Flexível (com Kernel Gaussiano) foi o melhor de um ponto de vista preditivo.

6.2 Comparações considerando a variável *diff_spd*

6.2.1 Variáveis dos ajustes via Bayes Ingênuo

Como já dito na Subseção 4.3.3, para a seleção de covariáveis a serem utilizadas nos modelos ajustados via Bayes Ingênuo, iremos **ajustar no conjunto de treino uma**

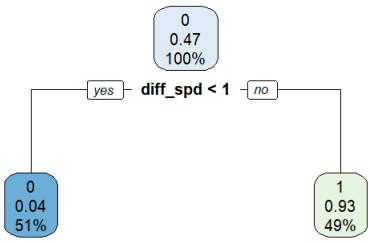

Figura 6.3: Representação visual da Árvore de Classificação obtida após poda, considerando a variável *diff_spd*.

Árvore de Classificação e uma Regressão Logística via Lasso. As covariáveis resultantes dos dois métodos, e dispostas na Tabela 6.4, serão utilizadas para comparação:

Tabela 6.4: Variáveis selecionadas para ajustes via Bayes Ingênuo considerando a existência da variável *diff_spd*.

Método de Seleção	**Variáveis Selecionadas**
Árvore de Classificação	diff_spd
Regressão Logística via Lasso	diff_hp, diff_atk, diff_sdef, diff_spd, res_tipo1, res_tipo2

No caso específico da seleção via Árvore de Classificação, **apenas a variável diff_spd foi considerada pelo algoritmo**, fato já sugerido pela análise descritiva na Figura 5.6, e a razão dos resultados serem divididos pela consideração e desconsideração dela. Na Figura 6.3, temos uma representação visual da árvore resultante:

Nas próximas seções, iremos selecionar os melhores modelos dentro de cada método, via conjunto de validação, começando pelo Bayes Ingênuo Gaussiano na Subseção 6.2.2. No final, na Figura 6.2.3, teremos uma comparação final dos melhores modelos dentro de cada método, mas desta vez no conjunto de teste.

6.2.2 Melhor Bayes Ingênuo Gaussiano

Os modelos ajustados via Bayes Ingênuo Gaussiano foram obtidos através do pacote *naivebayes* no R (Majka, 2020). As métricas avaliativas estimadas encontram-se dispostas

na Tabela 6.5, e as Curvas ROC estimadas, na Figura 6.4.

Tabela 6.5: Métricas Avaliativas estimadas para modelos ajustados via Bayes Ingênuo Gaussiano, via conjunto de validação, para diferentes seleções de variáveis, considerando a variável *diff_spd*.

Ajuste	Acurácia	Risco	Sensibilidade	Especificidade	VPP	VPN	AUC
Completo	0.8116	0.1884	0.8181	0.8041	0.8268	0.7945	0.8887
Arvore	0.9317	0.0683	0.9345	0.9284	0.9372	0.9254	0.9349
Lasso	0.8375	0.1625	0.8432	0.8310	0.8508	0.8226	0.9118

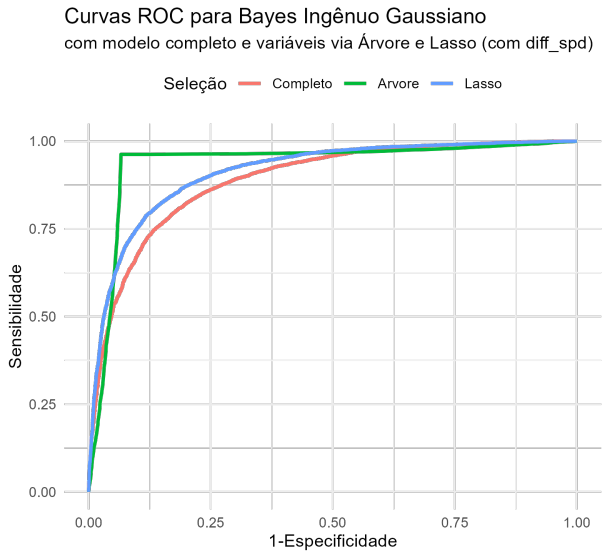

Figura 6.4: Curvas ROC dos modelos ajustados via Bayes Ingênuo Gaussiano para diferentes seleções de variáveis, estimadas via conjunto de validação, considerando a variável *diff_spd*.

Tanto nas curvas estimadas quanto nas métricas, o cenário de desempenho é claro: o modelo ajustado via Bayes Ingênuo Gaussiano, considerando variáveis selecionadas via Árvore de Classificação, teve o melhor desempenho no conjunto de validação:

- Sua curva, em verde, é a que mais se aproxima do canto superior esquerdo do gráfico, indicando a superioridade do modelo sob um ponto de vista preditivo;
- Todas as métricas são superiores aos outros modelos, com o Modelo Completo sendo o pior e o Bayes Ingênuo Gaussiano com variáveis via Lasso, o segundo pior. Assim, o Bayes Ingênuo Gaussiano com variáveis via Árvore, no geral, classificou corretamente mais vezes, bem como classificou corretamente mais vezes tanto observações em que $win = 1$ como observações em que $win = 0$.

Portanto, **selecionamos o modelo ajustado via Bayes Ingênuo Gaussiano com variáveis selecionadas via Árvore** como o melhor modelo dentro dos ajustados via Bayes Ingênuo Gaussiano.

6.2.3 Melhor Bayes Ingênuo Flexível

Para modelos ajustado via Bayes Ingênuo Flexível, para melhor organização, iremos comparar modelos ajustados considerando as variáveis selecionadas via Árvore, para cada um dos três Kernels descritos na Subseção 3.3.2; após isto, realizar o mesmo procedimento, mas considerando as variáveis selecionadas via Lasso; por fim, iremos comparar os melhores modelos dos dois ajustes citados entre si e com o modelo completo, ajustado via Bayes Ingênuo Flexível com Kernel Gaussiano, como visto na Tabela 6.2 e na Figura 6.1.

Novamente, os ajustes foram feitos através do pacote *naivebayes* no R (Majka, 2020), desta vez com o argumento $usekernel = TRUE$.

Covariáveis via Árvore de Classificação

As métricas estimadas encontram-se dispostas na Tabela 6.6 e as Curvas ROC, na Figura 6.5.

Todos os três kernels, tanto pelas métricas avaliativas quanto pelas Curvas ROC, apresentam desempenho preditivo muito próximo, de forma que é difícil compará-los (algo que será realidade para outras comparações entre Bayes Ingênuo Flexível). No entanto, diferenças que surgem em três casas decimais, o modelo ajustado via **Bayes Ingênuo Flexível com variáveis selecionadas via Árvore e kernel Gaussiano** foi o melhor dentre os três com variáveis selecionadas via árvore: sua Acurácia, Sensibilidade, Especificidade, VPP, VPN e AUC foram maiores (ou ao menos empataram) se comparadas com os outros dois.

Tabela 6.6: Métricas Avaliativas estimadas para modelos ajustados via Bayes Ingênuo Flexível, via conjunto de validação, para diferentes kernels e seleção de variáveis via Árvore, considerando a variável *diff_spd*.

Ajuste	Acurácia	Risco	Sensibilidade	Especificidade	VPP	VPN	AUC
Parabolico	0.9419	0.0581	0.9341	0.9509	0.956	0.9266	0.9516
Uniforme	0.9419	0.0581	0.9340	0.9509	0.956	0.9265	0.9521
Gaussiano	0.9424	0.0576	0.9341	0.9520	0.957	0.9267	0.9522

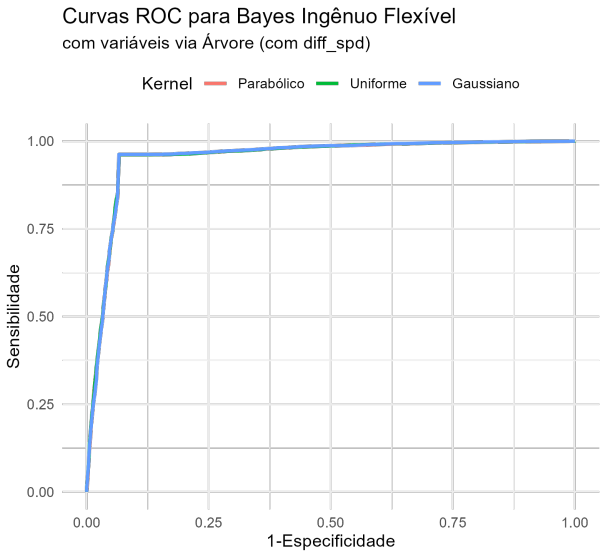

Figura 6.5: Curva ROC dos modelos ajustados via Bayes Ingênuo Flexível para diferentes kernels, considerando o conjunto de validação e seleção de variáveis via Árvore, considerando a variável *diff_spd*.

Portanto, dentre os modelos utilizando variáveis selecionadas via Árvore, **selecionamos o modelo ajustado via Bayes Ingênuo Flexível com kernel Gaussiano** como o melhor modelo.

Covariáveis via Lasso

As métricas estimadas encontram-se dispostas na Tabela 6.7 e as Curvas ROC, na Figura 6.6.

Como anteriormente, todos os três modelos apresentam desempenho muito próximo mas, neste caso, o ajustado com o kernel Uniforme fica mais abaixo dos outros se nos atentarmos à Figura 6.6. No mais, como ocorreu nos ajustes com variáveis selecionadas via Árvore, **os ajustes com variáveis selecionadas via Lasso apresentam o modelo ajustado via kernel Gaussiano** como o melhor, mas por uma margem pequena; desta vez, no entanto, não houveram "empates" até a quarta casa decimal.

Tabela 6.7: Métricas Avaliativas estimadas para modelos ajustados via Bayes Ingênuo Flexível, via conjunto de validação, para diferentes kernels e seleção de variáveis via Lasso, considerando a variável *diff_spd*.

Ajuste	Acurácia	Risco	Sensibilidade	Especificidade	VPP	VPN	AUC
Parabolico	0.9183	0.0817	0.9153	0.9217	0.9304	0.9049	0.9655
Uniforme	0.9175	0.0825	0.9131	0.9225	0.9309	0.9028	0.9653
Gaussiano	0.9217	0.0783	0.9182	0.9258	0.9339	0.9082	0.9664

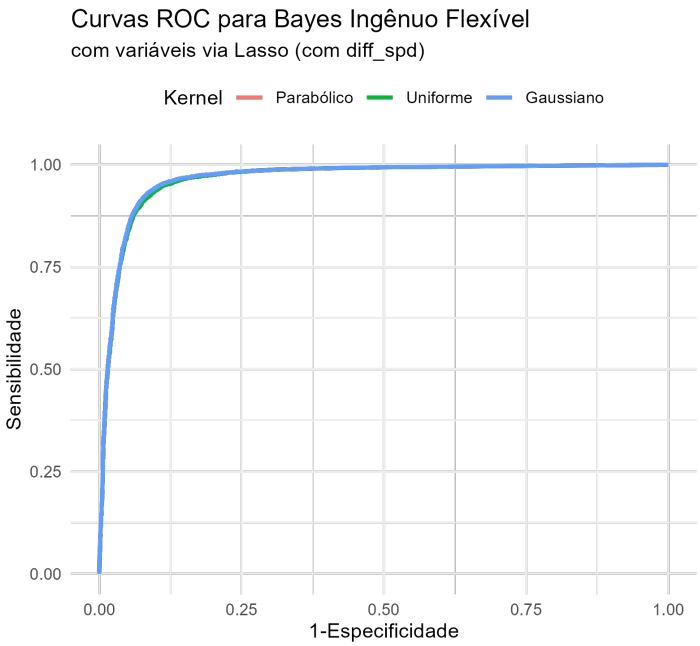

Figura 6.6: Curva ROC dos modelos ajustados via Bayes Ingênuo Flexível para diferentes kernels, considerando o conjunto de validação e seleção de variáveis via Lasso, considerando a variável *diff_spd*.

Portanto, dentre os modelos utilizando variáveis selecionadas via Lasso, **selecionamos o modelo ajustado via Bayes Ingênuo Flexível com kernel Gaussiano** como o melhor modelo.

Comparação dos Melhores Modelos

Com os melhores modelos considerando todas as covariáveis, variáveis selecionadas via Árvore e variáveis selecionadas via Lasso obtidos, compararemos seus resultados entre si para determinarmos qual ajuste via Bayes Ingênuo Flexível apresentou melhor desempenho preditivo, considerando o conjunto de validação. As métricas estimadas encontram-se dispostas na Tabela 6.8 e as Curvas ROC, na Figura 6.7.

Tabela 6.8: Métricas Avaliativas estimadas para melhores modelos ajustados via Bayes Ingênuo Flexível, via conjunto de validação, para diferentes seleções de variáveis e considerando a variável *diff_spd*.

Ajuste	Acurácia	Risco	Sensibilidade	Especificidade	VPP	VPN	AUC
Completo	0.8963	0.1037	0.9025	0.8892	0.9030	0.8886	0.9510
Arvore	0.9424	0.0576	0.9341	0.9520	0.9570	0.9267	0.9522
Lasso	0.9217	0.0783	0.9182	0.9258	0.9339	0.9082	0.9664

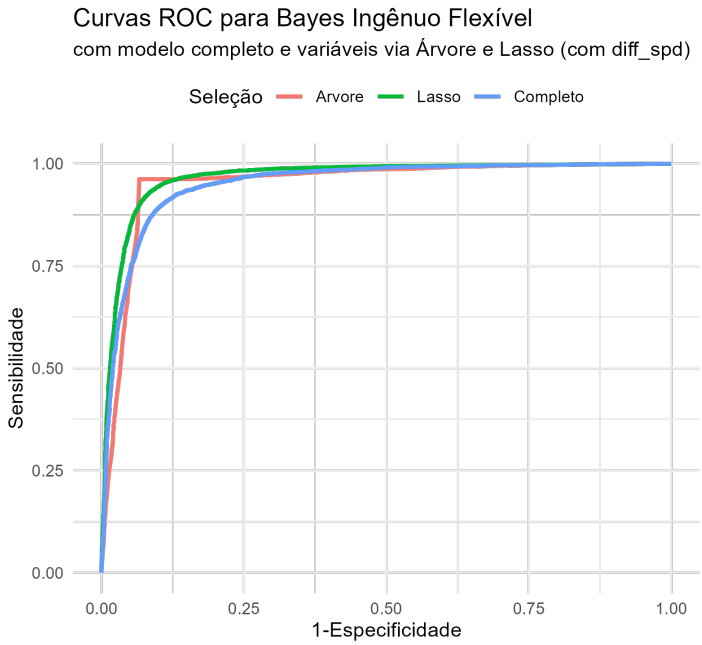

Figura 6.7: Curva ROC dos modelos ajustados via Bayes Ingênuo Flexível, estimadas pelo conjunto de validação, considerando diferentes seleções de variáveis e considerando a variável *diff_spd*.

Pela Figura 6.7, fica nítido que o modelo completo via Bayes Ingênuo Flexível apresentou o pior desempenho preditivo; no entanto, o ajuste com variáveis selecionadas via Árvore apresenta curva mais próxima do canto superior esquerdo, mas que permanece abaixo do ajuste com variáveis via Lasso no restante do gráfico.

Analisando as métricas da Tabela 6.8, o melhor modelo ajustado com variáveis via Árvore apresenta melhor poder preditivo considerando todas as métricas **exceto a Área sob a Curva (AUC)**, algo compreensível dado o comportamento exibido pelas Curvas ROC. Ainda assim, sua superioridade considerando o restante da tabela não deixa dúvidas de que, entre os três, é a melhor escolha do ponto de vista preditivo.

Portanto, dentre todos os modelos ajustados via Bayes Ingênuo Flexível, **selecionamos o modelo ajustado via Bayes Ingênuo Flexível com kernel Gaussiano e variáveis via Árvore** como o melhor modelo.

6.2.4 Melhor Regressão Logística

Para a melhor Regressão Logística, compararemos o modelo com todas as variáveis (ajustado via função *glm()* do R base) e um ajuste com penalização via Lasso (ajustado via pacote *glmnet* (Friedman *et al.*, 2010). As métricas estimadas encontram-se dispostas na Tabela 6.9 e as Curvas ROC, na Figura 6.8.

O desempenho do modelo ajustado com penalização via Lasso se destaca na Figura 6.8, porém, com uma margem menor do que a esperada (pelo menos se compararmos outras análises similares, como no artigo *"The lasso binary logistic regression method for selecting variables that affect the recovery of Covid-19 patients in China"* de Rusyana *et al.* (2021).

Ainda assim, a penalização via Lasso apresentou melhores resultados preditivos com uma boa diferença na Tabela 6.9, **exceto no caso da Área sob a Curva**, com diferença apresentada apenas na quarta casa decimal. Para as outras, o resultado foi mais próximo do desejado (ou seja, risco mínimo e outras métricas maximizadas).

Portanto, dentre todos os modelos ajustados via Regressão Logística, **selecionamos o modelo ajustado com penalização via Lasso** como o melhor modelo.

Tabela 6.9: Métricas Avaliativas estimadas para modelos ajustados via Regressão Logística, via conjunto de validação, para diferentes seleções de variáveis.

Ajuste	Acurácia	Risco	Sensibilidade	Especificidade	VPP	VPN	AUC
Completo	0.8918	0.1082	0.8999	0.8825	0.8974	0.8852	0.9355
Lasso	0.9009	0.0991	0.9088	0.8919	0.9057	0.8954	0.9357

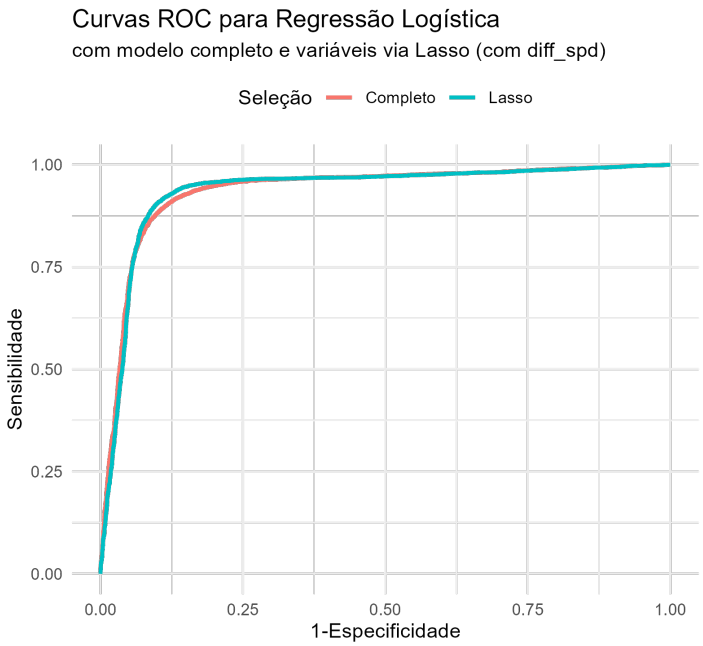

Figura 6.8: Curvas ROC dos modelos ajustados via Regressão Logística, estimadas pelo conjunto de validação, considerando diferentes seleções de variáveis, considerando a variável *diff_spd*.

6.2.5 Comparação Final

Nesta seção, iremos comparar o desempenho preditivo, **no conjunto de teste**, dos seguintes modelos, selecionados anteriormente:

- Bayes Ingênuo Gaussiano com variáveis selecionadas via Árvore;
- Bayes Ingênuo Flexível com kernel Gaussiano e variáveis selecionadas via Árvore;
- Regressão Logística com penalização via Lasso;
- Árvore de Classificação com estrutura descrita na Figura 6.3.

As métricas estimadas encontram-se dispostas na Tabela 6.10 e as Curvas ROC, na Figura 6.9.

Tabela 6.10: Métricas Avaliativas estimadas para melhores modelos ajustados, via conjunto de teste.

Ajuste	Acurácia	Risco	Sensibilidade	Especificidade	VPP	VPN	AUC
Logística	0.8988	0.1012	0.9103	0.8859	0.8988	0.8988	0.9308
Árvore	0.9421	0.0579	0.9318	0.9536	0.9572	0.9262	0.9427
BI Gaussiano	0.9264	0.0736	0.9330	0.9190	0.9276	0.9250	0.9287
BI Flexível	0.9377	0.0623	0.9321	0.9440	0.9488	0.9259	0.9499

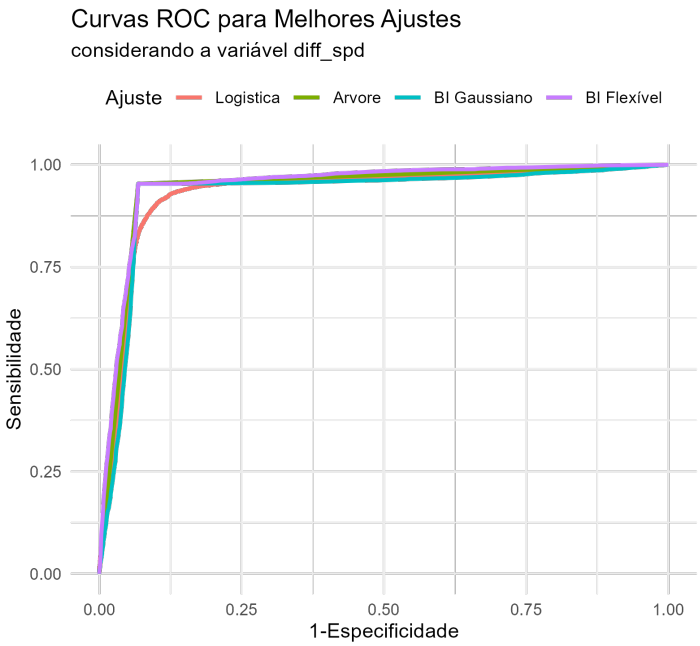

Figura 6.9: Curvas ROC dos melhores modelos ajustados, estimadas pelo conjunto de teste, considerando a variável *diff_spd*.

Dentre as curvas da Figura 6.9, a única distinção clara que é possível de ser feita é a de que o **desempenho do modelo ajustado via Regressão Logística com Lasso** está abaixo dos demais. No que diz respeito aos outros ajustes, no entanto, temos alguns destaques que não são possíveis de serem observados pelos gráficos, mas sim pelas métricas da Tabela 6.10:

- O modelo ajustado via Árvore de Classificação apresentou maior acurácia, especificidade, VPP e VPN dos que os demais;
- O melhor modelo ajustado via Bayes Ingênuo Flexível apresentou apenas maior AUC, mas também supera o melhor modelo ajustado via Bayes Ingênuo Gaussiano em acurácia, especificidade, VPP e VPN;

- O melhor modelo ajustado via Bayes Ingênuo Gaussiano apresentou apenas maior sensibilidade do que os demais.

Considerando todos os modelos, **o modelo ajustado via Árvore de Classificação é melhor do que os outros de um ponto de vista preditivo**. No entanto, se considerarmos apenas os ajustados via algum método de Bayes Ingênuo, **o modelo ajustado via Bayes Ingênuo Flexível com kernel Gaussiano e variáveis selecionadas via Árvore** foi melhor do que sua contraparte; ainda que o modelo ajustado via Bayes Ingênuo Gaussiano com variáveis via Árvore tenha obtido maior sensibilidade, a vantagem não é grande (diferença de apenas 0.0009) o bastante para que se torne uma escolha interessante em detrimento do método Flexível.

6.3 Comparações desconsiderando a variável *diff_spd*

Como pudemos ver, a variável *diff_spd* apresentou forte influência no desempenho dos modelos. Para enriquecer a análise deste trabalho, iremos realizar os mesmos procedimentos da Seção 6.2. Antes de realizarmos os ajustes propriamente ditos, verificaremos novamente o balanceamento dos dados para novos conjuntos de treino, validação e teste, desta vez desconsiderando a variável *diff_spd*.

Tabela 6.11: Verificação do balanceamento da proporção de observações nos conjuntos de treino e teste (e treino e validação) sem *diff_spd*.

Conjunto	**% win=1**	**% win=0**
Treino	0.4699	0.5301
Teste	0.4720	0.5280
Treino	0.4724	0.5276
Validação	0.4661	0.5339

Em ambas as divisões dispostas na Tabela 6.11, não temos desbalanceamento grave. Com isso, prosseguiremos com os resultados; desta vez, iremos direto para a escolha dos melhores modelos dentro de cada método, começando pela seleção de variáveis a ser utilizada em cada caso.

6.3.1 Variáveis dos ajustes via Bayes Ingênuo

As covariáveis resultantes dos dois métodos, e dispostas na Tabela 6.12, serão utilizadas para comparação:

Tabela 6.12: Variáveis selecionadas para ajustes via Bayes Ingênuo desconsiderando a existência da variável *diff_spd*.

Método de Seleção	**Variáveis Selecionadas**
Árvore de Classificação	diff_atk, diff_satk
Regressão Logística via Lasso	diff_hp, diff_atk, diff_def, diff_satk diff_sdef, res_tipo1, res_tipo2, leg_atk, leg_def

Temos que, com a retirada da variável *diff_spd*, as determinadas como mais importantes para a construção do modelo, segundo a Árvore de Classificação, foram as variáveis *diff_atk* e *diff_satk*. Para o caso das variáveis selecionadas via Lasso, todas foram consideradas importantes, **exceto as variáveis *gen_atk* e *gen_def***. Na Figura 6.10, temos uma representação visual da árvore de classificação obtida.

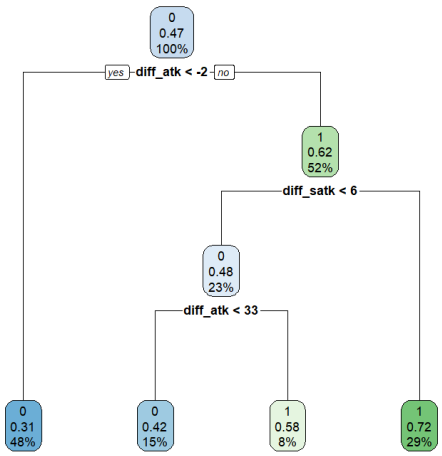

Figura 6.10: Representação visual da Árvore de Classificação obtida após poda desconsiderando a variável *diff_spd*.

6.3.2 Melhor Bayes Ingênuo Gaussiano

Os modelos ajustados, novamente, foram obtidos através do pacote *naivebayes* no R. As métricas avaliativas estimadas encontrma-se dispostas na Tabela 6.13, e as Curvas ROC estimadas, na Figura 6.11.

Tabela 6.13: Métricas Avaliativas estimadas para modelos ajustados via Bayes Ingênuo Gaussiano, via conjunto de validação, para diferentes seleções de variáveis desconsiderando a variável *diff_spd*.

Ajuste	Acurácia	Risco	Sensibilidade	Especificidade	VPP	VPN	AUC
Completo	0.6705	0.3295	0.6778	0.6620	0.6967	0.6420	0.7368
Arvore	0.6819	0.3181	0.7006	0.6606	0.7028	0.6582	0.7482
Lasso	0.6687	0.3313	0.6754	0.6611	0.6954	0.6399	0.7346

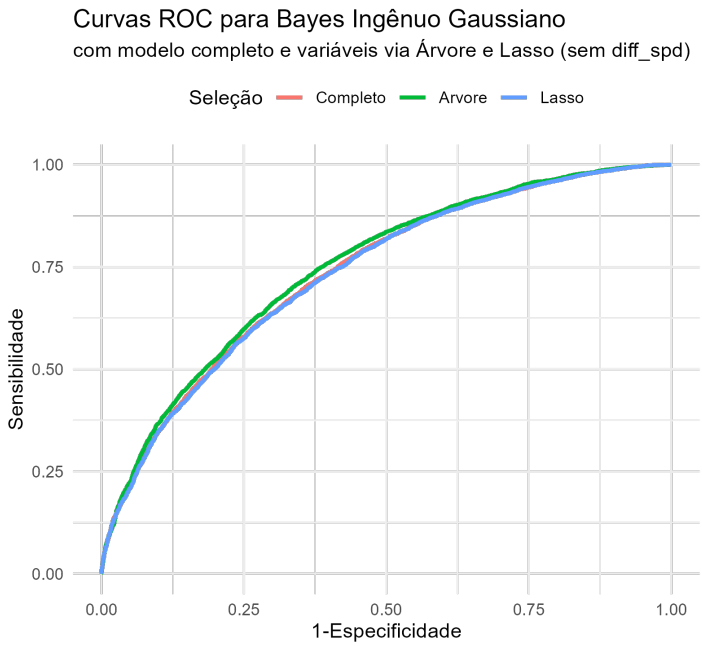

Figura 6.11: Curvas ROC dos modelos ajustados via Bayes Ingênuo Gaussiano para diferentes seleções de variáveis, estimadas via conjunto de validação, desconsiderando a variável *diff_spd*.

O modelo com o melhor desempenho preditivo é menos claro, mas já é possível que visualizemos isso através da Figura 6.11: o modelo via Bayes Ingênuo Gaussiano com variáveis selecionadas via Árvore, com a curva em verde (assim como quando consideramos *diff_spd*), está acima das demais, desta vez com uma vantagem um pouco menor.

Na Tabela 6.13, vemos que os ajustes com variáveis via Árvore e via Lasso apresentam desempenho próximo:

- O ajuste com variáveis via Árvore apresenta maior acurácia, sensibilidade, VPP, VPN e AUC;

- O ajuste com variáveis via Lasso apresenta segunda maior especificidade, perdendo para a especificidade do modelo completo.

Ainda assim, dado que a diferença de especificidades entre os modelos com variáveis via Árvore e via Lasso é de aproximadamente 0.0005 apenas, **o ajuste via Bayes Ingênuo Gaussiano com variáveis via Árvore possui maior poder preditivo do que os demais**. Logo, este será considerado o melhor modelo ajustado via Bayes Ingênuo Gaussiano.

6.3.3 Melhor Bayes Ingênuo Flexível

Como quando consideramos a variável *diff_spd*, os ajustes via Bayes Ingênuo Flexível foram feitos através do pacote *naivebayes* no R com o argumento *usekernel=TRUE*. A organização ocorrerá da mesma forma: primeiro, compararemos modelos completos, com variáveis via Árvore e variáveis via Lasso isoladamente; depois, compararemos os melhores modelos dentro de cada seleção de variáveis.

Modelos Completos

As métricas estimadas encontram-se dispostas na Tabela 6.14 e as Curvas ROC, na Figura 6.12.

O desempenho dos modelos sequer é distinguível nas curvas da Figura 6.12. Na Tabela 6.14, é possível ver que todos os desempenhos são bem próximos; no entanto, para fins de seleção, temos que:

- O ajuste com kernel Gaussiano apresentou maior acurácia, VPN e AUC;
- O ajuste com kernel Uniforme apresentou maior especificidade e VPP;
- O ajuste com kernel Parabólico apresentou maior sensibilidade apenas.

Tabela 6.14: Métricas Avaliativas estimadas para modelos ajustados via Bayes Ingênuo flexível, via conjunto de validação, para diferentes kernels e modelo completo, desconsiderando a variável *diff_spd*.

Ajuste	Acurácia	Risco	Sensibilidade	Especificidade	VPP	VPN	AUC
Parabolico	0.6665	0.3335	0.6780	0.6534	0.6914	0.6391	0.7350
Uniforme	0.6659	0.3341	0.6594	0.6733	0.6981	0.6331	0.7350
Gaussiano	0.6672	0.3328	0.6778	0.6549	0.6924	0.6396	0.7355

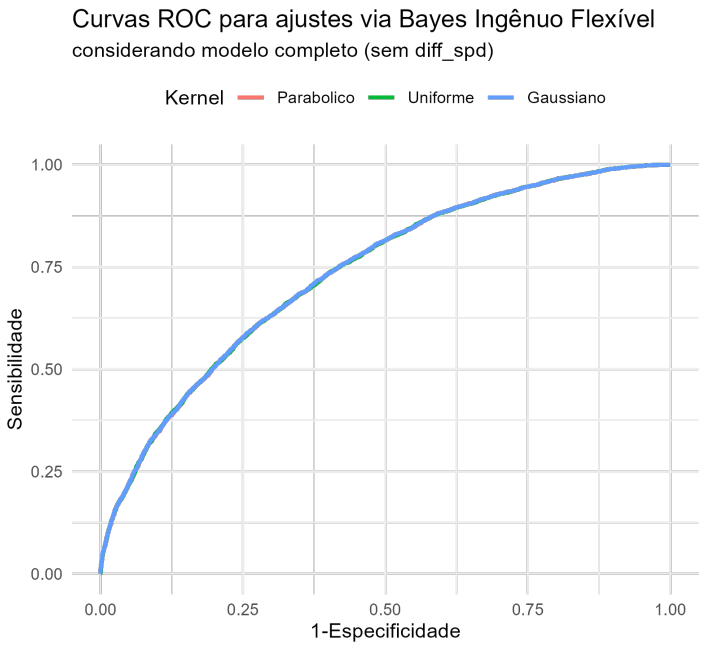

Figura 6.12: Curvas ROC dos modelos ajustados via Bayes Ingênuo Flexível para diferentes kernels e modelo completo, estimadas via conjunto de validação, desconsiderando a variável *diff_spd*.

Considerando que o ajuste com kernel Gaussiano maximizou mais métricas **e teve sensibilidade bastante próxima (diferença de 0.0002) do com kernel Parabólico**, o consideraremos como o melhor modelo completo ajustado via Bayes Ingênuo Flexível.

Covariáveis via Árvore de Classificação

As métricas estimadas encontram-se dispostas na Tabela 6.15 e as Curvas ROC, na Figura 6.13.

Da mesma forma que na Figura 6.12, o desempenho dos modelos não é distinguível para as Curvas ROC. Na Figura 6.13, temos que:

- O ajuste com kernel Gaussiano apresentou maior acurácia, sensibilidade, VPN e AUC;
- O ajuste com kernel Uniforme apresentou maior especificidade e VPP;
- O ajuste com kernel Parabólico não apresentou nenhuma métrica maior do que os demais, mas ficou "em segundo lugar" no que diz respeito à acurácia, sensibilidade e especificidade.

Tabela 6.15: Métricas Avaliativas estimadas para modelos ajustados via Bayes Ingênuo flexível, via conjunto de validação, para diferentes kernels e seleção de variáveis via Árvore, desconsiderando a variável *diff_spd*.

Ajuste	Acurácia	Risco	Sensibilidade	Especificidade	VPP	VPN	AUC
Parabolico	0.6796	0.3204	0.6944	0.6626	0.7022	0.6543	0.7473
Uniforme	0.6789	0.3211	0.6518	0.7101	0.7203	0.6403	0.7475
Gaussiano	0.6806	0.3194	0.6973	0.6614	0.7023	0.6560	0.7482

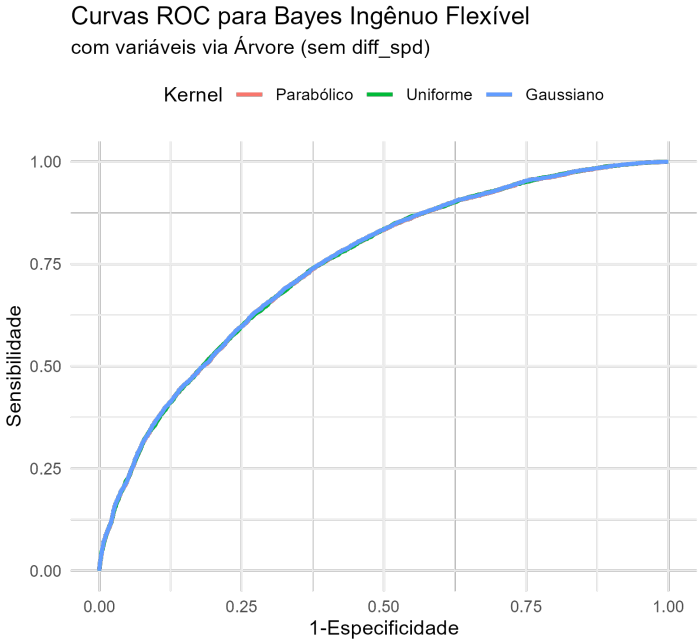

Figura 6.13: Curvas ROC dos modelos ajustados via Bayes Ingênuo Flexível para diferentes kernels e seleção de variáveis via Árvore, estimadas via conjunto de validação, desconsiderando a variável *diff_spd*.

Neste caso, ainda que as diferenças sejam maiores (a diferença no VPP dos kernels Uniforme e Gaussiano, por exemplo, é de aproximadamente 0.02), **o ajuste com kernel Gaussiano apresentou maior poder preditivo** dada a quantidade de métricas maximizadas. Portanto, este será considerado o melhor modelo ajustado via Bayes Ingênuo Flexível com variáveis selecionadas via Árvore de Classificação.

Covariáveis via Lasso

As métricas estimadas encontram-se dispostas na Tabela 6.16 e as Curvas ROC, na Figura 6.14.

Partindo para a comparação das métricas na Tabela 6.16, temos que:

- O ajuste com kernel Gaussiano apresentou maior acurácia e AUC, ficando em segundo para sensibilidade, especificidade, VPP e VPN;
- O ajuste com kernel Uniforme apresentou maior especificidade e VPP;
- O ajuste com kernel Parabólico apresentou maior sensibilidade e VPN; ficando em segundo para acurácia, VPP e AUC.

No geral, kernels Gaussiano e Parabólico apresentaram desempenho mais próximo entre si, **mas o ajuste com kernel Gaussiano, tendo maximizado ou ficado em segundo lugar para todas as métricas, pode ser considerado melhor**. Assim, este será considerado o melhor modelo ajustado via Bayes Ingênuo Flexível com seleção de variáveis via Lasso.

Tabela 6.16: Métricas Avaliativas estimadas para modelos ajustados via Bayes Ingênuo flexível, via conjunto de validação, para diferentes kernels e seleção de variáveis via Lasso, desconsiderando a variável *diff_spd*.

Ajuste	Acurácia	Risco	Sensibilidade	Especificidade	VPP	VPN	AUC
Parabolico	0.6647	0.3353	0.6760	0.6518	0.6899	0.6372	0.7331
Uniforme	0.6634	0.3366	0.6553	0.6727	0.6964	0.6301	0.7330
Gaussiano	0.6649	0.3351	0.6744	0.6540	0.6907	0.6368	0.7336

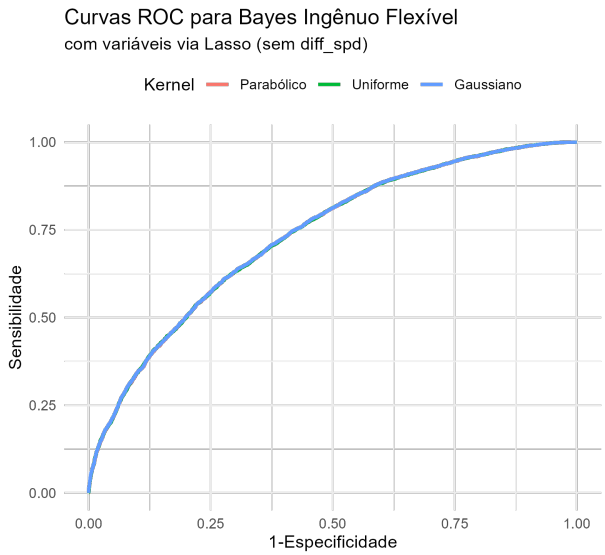

Figura 6.14: Curvas ROC dos modelos ajustados via Bayes Ingênuo Flexível para diferentes kernels e seleção de variáveis via Lasso, estimadas via conjunto de validação, desconsiderando a variável *diff_spd*.

Comparação dos Melhores Modelos

As métricas estimadas encontram-se dispostas na Tabela 6.17 e as Curvas ROC, na Figura 6.15, para os melhores modelos com todas as variáveis, as selecionadas via Árvore e as selecionadas via Lasso.

Se, desta vez, há uma diferença menor nas Curvas ROC do que as obtidas na Figura 6.7, ainda temos que o melhor modelo ajustado com variáveis selecionadas via Árvore possui maior poder preditivo do que os demais. Isto se reflete na Tabela 6.17, onde todas as métricas deste ajuste são maiores. Portanto, dentre todos os modelos ajustados via Bayes Ingênuo Flexível, **selecionamos o ajustado via Bayes Ingênuo Flexível com kernel Gaussiano e variáveis via Árvore** como melhor modelo.

Tabela 6.17: Métricas Avaliativas estimadas para modelos ajustados via Bayes Ingênuo Flexível, via conjunto de validação, para diferentes kernels e seleções de variáveis, desconsiderando a variável *diff_spd*.

Ajuste	Acurácia	Risco	Sensibilidade	Especificidade	VPP	VPN	AUC
Completo	0.6672	0.3328	0.6778	0.6549	0.6924	0.6396	0.7355
Arvore	0.6806	0.3194	0.6973	0.6614	0.7023	0.6560	0.7482
Lasso	0.6649	0.3351	0.6744	0.6540	0.6907	0.6368	0.7336

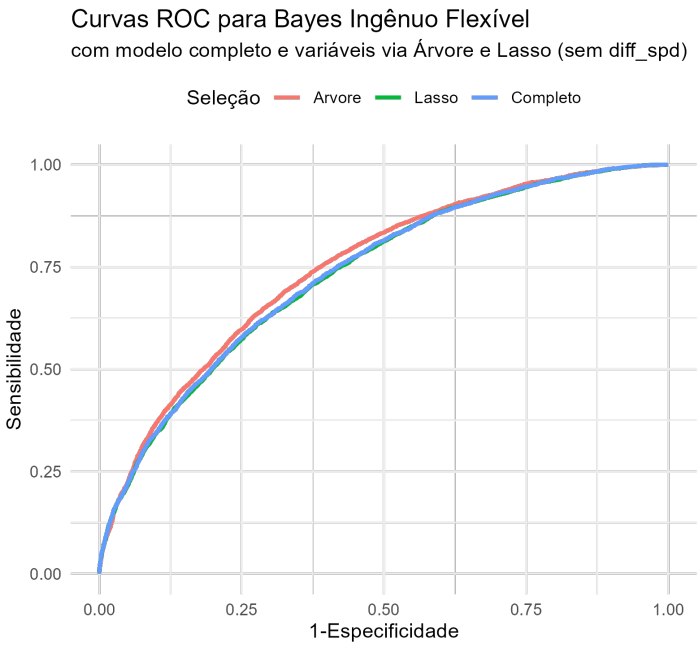

Figura 6.15: Curvas ROC dos modelos ajustados via Bayes Ingênuo Flexível para diferentes kernels e seleções de variáveis, estimadas via conjunto de validação, desconsiderando a variável *diff_spd*.

6.3.4 Melhor Regressão Logística

As métricas estimadas encontram-se dispostas na Tabela 6.16 e as Curvas ROC, na Figura 6.14. Como quando consideramos a variável *diff_spd*, o modelo completo foi ajustado via função *glm()*, enquanto que o penalizado via Lasso foi ajustado com o pacote *glmnet*.

Tabela 6.18: Métricas Avaliativas estimadas para modelos ajustados via Regressão Logística, via conjunto de validação, para modelo completo e ajuste via Lasso, desconsiderando a variável *diff_spd*.

Ajuste	Acurácia	Risco	Sensibilidade	Especificidade	VPP	VPN	AUC
Completo	0.7041	0.2959	0.7316	0.6727	0.7192	0.6863	0.7716
Lasso	0.6978	0.3022	0.7297	0.6614	0.7117	0.6811	0.7660

Ao contrário do esperado pela literatura, como é possível de ser visualizado na Figura 6.16 com a curva vermelha superando a azul, o modelo completo apresentou melhor desempenho do que o modelo penalizado via Lasso. Na Tabela 6.18, **o ajuste completo supera o ajuste via Lasso em todas as métricas avaliativas**. Assim, consideraremos ele como o melhor modelo ajustado via Regressão Logística.

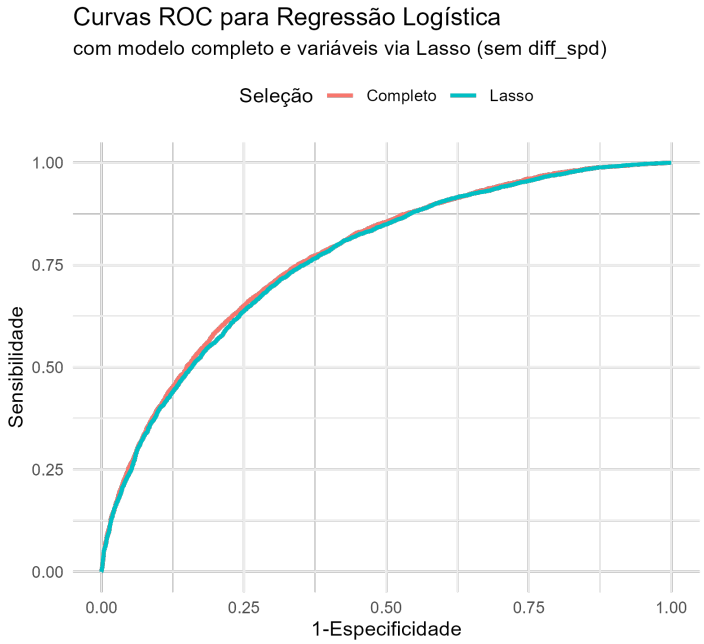

Figura 6.16: Curvas ROC dos modelos ajustados via Regressão Logística para modelo completo e ajuste via Lasso, estimadas via conjunto de validação, desconsiderando a variável *diff_spd*.

Ao contrário do esperado pela literatura, como é possível de ser visualizado na Figura 6.16 com a curva vermelha superando a azul, o modelo completo apresentou melhor desempenho do que o modelo penalizado via Lasso. Na Tabela 6.18, **o ajuste completo supera o ajuste via Lasso em todas as métricas avaliativas**. Assim, consideraremos ele como o melhor modelo ajustado via Regressão Logística.

6.3.5 Comparação Final

Nesta seção, iremos comparar o desempenho preditivo, **no conjunto de teste**, dos seguintes modelos, selecionados anteriormente;

- Bayes Ingênuo Gaussiano com variáveis selecionadas via Árvore de Classificação;
- Bayes Ingênuo Flexível com kernel Gaussiano e variáveis selecionadas via Árvore de Classificação;
- Regressão Logística com modelo completo;
- Árvore de Classificação com estrutura descrita na Figura 6.10.

As métricas estimadas encontram-se dispostas na Tabela 6.19 e as Curvas ROC, na Figura 6.17.

Na Figura 6.17, podemos ver desta vez que o modelo completo ajustado via Regressão Logística apresentou maior poder preditivo do que os demais, com o modelo ajustado via Árvore de Classificação sendo o pior. No entanto, os dois modelos ajustados via Bayes Ingênuo Flexível possuem curvas muito próximas, quase indistinguíveis.

Ao analisarmos a Tabela 6.19, considerando os ajustes via Bayes Ingênuo isoladamente, temos que:

- O melhor ajuste via Bayes Ingênuo Gaussiano apresentou maior acurácia, sensibilidade, VPN e AUC, empatando em VPP até a quarta casa decimal;
- O melhor ajuste via Bayes Ingênuo Flexível apresentou maior especificidade apenas.

Tabela 6.19: Métricas Avaliativas estimadas para melhores modelos ajustados, via conjunto de teste, desconsiderando a variável *diff_spd*.

Ajuste	Acurácia	Risco	Sensibilidade	Especificidade	VPP	VPN	AUC
Logística	0.7010	0.2990	0.7306	0.6678	0.7110	0.6891	0.7736
Árvore	0.6682	0.3318	0.7750	0.5488	0.6577	0.6856	0.6827
BI Gaussiano	0.6810	0.3190	0.7052	0.6538	0.6950	0.6648	0.7504
BI Flexível	0.6807	0.3193	0.7045	0.6542	0.6950	0.6643	0.7500

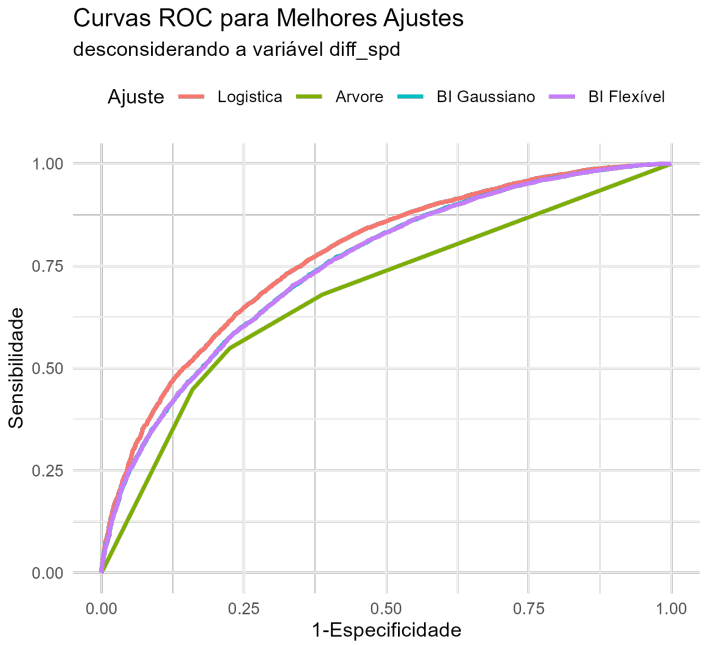

Figura 6.17: Curvas ROC dos melhores modelos ajustados, estimadas pelo conjunto de teste, desconsiderando a variável *diff_spd*.

Ainda que o desempenho de ambos os ajustes esteja muito próximo, preditivamente falando, para fins de seleção, **o melhor ajuste via Bayes Ingênuo pode ser considerado o feito via Bayes Ingênuo Flexível, com kernel Gaussiano e variáveis selecionadas via Árvore**.

Capítulo 7

Considerações Finais

No presente trabalho de graduação, o objetivo foi revisarmos a literatura do classificador Bayes Ingênuo, focado especificamente num contexto binário, e compará-lo com outros dois métodos mais tradicionais da área: a Regressão Logística e a Árvore de Classificação. Abrindo o leque de abordagens, poderíamos ajustar modelos tanto via Bayes Ingênuo Gaussiano (que supõe normalidade das covariáveis contínuas) ou via Bayes Ingênuo Flexível (que estima a densidade das variáveis contínuas via diversos kernels).

Logo na análise descritiva do banco de dados utilizado, foi possível percebermos que a variável *diff_spd* apresentou grande disparidade entre observações que o Pokemon atacante ganhava e observações que o Pokemon atacante perdia (i.e. Pokemons com velocidades maiores do que o oponente, ganhavam mais frequentemente). Assim, para enriquecer a análise, foram feitos estudos sob a ótica preditiva dos ajustes considerando e desconsiderando esta variável.

Quando consideramos a variável *diff_spd*, o melhor modelo ajustado via Bayes Ingênuo foi o **Bayes Ingênuo Flexível com kernel Gaussiano e variáveis selecionadas via Árvore de Decisão**; desconsiderando a variável *diff_spd*, o melhor modelo foi o ajustado via **Bayes Ingênuo Gaussiano com seleção de variáveis via Árvore de Decisão**. Em ambos os casos, no entanto, eles perderam em questão de poder preditivo para Árvore de Classificação e Regressão Logística, respectivamente.

Ainda que os modelos via Bayes Ingênuo não tenham sido considerados absolutamente melhores do que os outros, algumas conclusões interessantes acerca do método podem ser tiradas:

- Quando utilizamos as mesmas variáveis selecionadas na Árvore de Decisão, o de-

sempenho preditivo dos modelos via Bayes Ingênuo cresceu, tornando este método de seleção de covariáveis interessante para aplicações futuras;

- A diferença de poder preditivo entre kernels não foi tão aparente, com a diferença das métricas avaliativas se dando, em sua maioria, na terceira ou na quarta casa decimal;

- Ao termos o melhor modelo via Bayes Ingênuo "invertido" quando consideramos e desconsideramos *diff_spd*, e dado que esta variável é crucial para a distinção entre vitória e derrota, é possível que o ajuste via Bayes Ingênuo Flexível tenha sido mais adequado apenas para ela. Quando a retiramos, o mais adequado passou a ser o ajuste via Bayes Ingênuo Flexível, dentre os ajustes via Bayes Ingênuo;

- Ainda que as suposições do método Bayes Ingênuo sejam dificilmente aceitas, os modelos ajustados apresentaram desempenho que se equiparou às outras abordagens.

O penúltimo ponto, inclusive, traz à luz uma possível limitação dos ajustes: se, para cada ajuste, estes estimam a densidade das variáveis numéricas de uma mesma forma para todas elas, diferentes tipos de estimativas podem ser mais adequadas para diferentes variáveis. Por exemplo, uma covariável específica é melhor estimada via kernel Parabólico, enquanto outra é melhor estimada via Distribuição Gaussiana. **Uma abordagem do tipo não foi encontrada durante a realização deste trabalho**, indicando uma possível oportunidade de aplicação futura que melhore (e possivelmente supera outros ajustes) o método Bayes Ingênuo.

Já para as suposições mencionadas no último ponto, é possível estimarmos rapidamente uma rede bayesiana através do pacote *bnlearn* no R (Scutari, 2010). Para nosso caso, considerando as variáveis contínuas, tivemos:

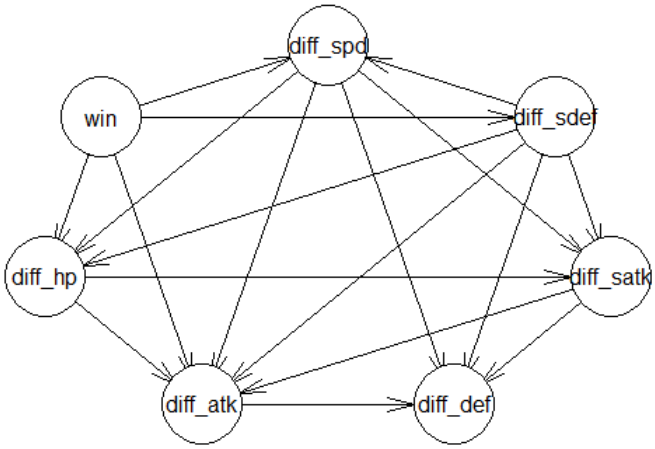

Figura 7.1: Representação da Rede Bayesiana estimada dos dados.

Na Figura 7.1, ao invés de termos setas saindo apenas da variável *win* (como visto na Equação 3.1.3), temos setas saindo de praticamente todas as variáveis, formando um emaranhado de conexões. Visualmente falando, já podemos comprovar que as **suposições de independência condicional das covariáveis com relação à variável resposta não estão atendidas**. Isto, no entanto, não impediu que o algoritmo Bayes Ingênuo pudesse demonstrar um desempenho preditivo adequado.

Referências Bibliográficas

Database, P. (2022). Pokémon type chart: strengths and weaknesses. Disponível em: https://pokemondb.net/type [Acessado em 10/09/2022].

Dogo, V. (2022). Códigos do Trabalho de Graduação em Estatística. Disponível em: https://github.com/victordogo/codigos-tg-victordogo [Acessado em 28/07/2022].

Dunn, P. K. e Smyth, G. K. (2018). *Generalized Linear Models with Examples in R*. Springer, New York, NY, 2018th edition. ISBN 978-1-4419-0117-0.

Friedman, J., Hastie, T. e Tibshirani, R. (2010). Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, **33**(1), 1–22.

Game Freak (2004). Pokemon FireRed Version & Pokemon LeafGreen Version.

Hastie, T., Tibshirani, R. e Friedman, J. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition*. Springer, New York, NY, second edition. ISBN 978-0-387-84857-0.

Izbicki, R. e dos Santos, T. M. (2020). *Aprendizado de máquina: uma abordagem estatística*. ISBN 978-65-00-02410-4.

John, G. H. e Langley, P. (1995). Estimating continuous distributions in Bayesian classifiers. Em *Proceedings of the Eleventh conference on Uncertainty in artificial intelligence*, UAI'95, páginas 338–345, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc. ISBN 978-1-55860-385-1.

Kaggle (2015). Pokemon- Weedle's Cave. Disponível em: https://www.kaggle.com/terminus7/pokemon-challenge [Acessado em 27/05/2022].

Mahalanobis, P. C. (1930). A Statistical Study of Certain Anthropometric Measurements

from Sweden. *Biometrika*, **22**(1/2), 94–108. Publisher: [Oxford University Press, Biometrika Trust].

Majka, M. (2020). naivebayes: High Performance Implementation of the Naive Bayes Algorithm.

Parzen, E. (1962). On Estimation of a Probability Density Function and Mode. *The Annals of Mathematical Statistics*, **33**(3), 1065–1076. Publisher: Institute of Mathematical Statistics.

Prasetio, D. e Harlili, D. (2016). Predicting football match results with logistic regression. Em *2016 International Conference On Advanced Informatics: Concepts, Theory And Application (ICAICTA)*, páginas 1–5.

R Core Team (2021). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

Rahman, M., Mustapha, A., Fauzi, R. e Razali, N. (2018). Bayesian Approach to Classification of Football Match Outcome. *International Journal of Integrated Engineering*, **10**.

Ratanamahatana, C. a. e Gunopulos, D. (2003). Feature selection for the naive bayesian classifier using decision trees. *Applied Artificial Intelligence*, **17**(5-6), 475–487. Publisher: Taylor & Francis _eprint: https://doi.org/10.1080/713827175.

Rusyana, A., Notodiputro, K. A. e Sartono, B. (2021). The lasso binary logistic regression method for selecting variables that affect the recovery of Covid-19 patients in China. *Journal of Physics: Conference Series*, **1882**(1), 012035.

Scutari, M. (2010). Learning bayesian networks with the bnlearn R package. *Journal of Statistical Software*, **35**(3), 1–22.

Stefano, E. d., Farroco, L. d. O., Lima, G. B. A., Parrancho, A., Gavião, L. O. e Principe, V. A. (2020). Decision Trees for the Prediction of Outcome of Soccer Games - Historical Data Analysis / Árvores de decisão para a previsão de resultados de Jogos de Futebol - Análise de Dados Históricos. *Brazilian Journal of Development*, **6**(1), 4719–4732. Number: 1.

Tibshirani, R. (1996). Regression Shrinkage and Selection via the Lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, **58**(1), 267–288. Publisher: [Royal Statistical Society, Wiley].

Turing (2022). AUC-ROC curves and their usage for classification in Python. Disponível em: https://www.turing.com/kb/auc-roc-curves-and-their-usage-for-classification-in-python [Acessado em 10/09/2022].

Wickham, H., Averick, M., Bryan, J., Chang, W., McGowan, L. D., François, R., Grolemund, G., Hayes, A., Henry, L., Hester, J., Kuhn, M., Pedersen, T. L., Miller, E., Bache, S. M., Müller, K., Ooms, J., Robinson, D., Seidel, D. P., Spinu, V., Takahashi, K., Vaughan, D., Wilke, C., Woo, K. e Yutani, H. (2019). Welcome to the Tidyverse. *Journal of Open Source Software*, **4**(43), 1686.

Apêndice A

Códigos utilizados

A.1 Pré-Processamento e Tratamento dos Dados

```
##### CODIGO PARA PREPARACAO DO DATASET 'pokebattle' #####

combats <- readr::read_csv('data-raw/combats.csv') |>
  dplyr::rename(
    atacante=First_pokemon,
    defesa=Second_pokemon
  ) |>
  dplyr::mutate(
    win=as.factor(ifelse(
      atacante==Winner,1,0
    ))
  ) |>
  dplyr::select(win,atacante,defesa)

pokemon <- readr::read_csv('data-raw/pokemon.csv') |>
  dplyr::rename(
    id=`#`, nome=Name, tipo_1=`Type 1`,
    tipo_2=`Type 2`, hp=HP, atk=Attack,
    def=Defense, satk=`Sp. Atk`, sdef=`Sp. Def`,
    spd=Speed, geracao=Generation, lendario=Legendary
  ) |>
  dplyr::mutate(
    lendario=as.factor(ifelse(lendario==TRUE, 1, 0)),
    geracao=as.factor(geracao)
  )
```

```
names <- pokemon |>
  dplyr::select(id,nome)

tipos <- readRDS("data-raw/tipos.rds")

### Preparando covariaveis

# Atacante e Defensor

combats <- combats |>
  dplyr::mutate(
    nome_atk=sapply(atacante,
                    \(x) names$nome[match(x, names$id)]),
    nome_def=sapply(defesa,
                    \(x) names$nome[match(x, names$id)])
)

# Diferenca de atk

combats <- combats |>
  dplyr::mutate(
    First_pokemon=sapply(nome_atk,
                         \(x) pokemon$atk[match(x, pokemon$nome)]),
    Second_pokemon=sapply(nome_def,
                          \(x) pokemon$atk[match(x, pokemon$nome)]),
    diff_atk=First_pokemon-Second_pokemon
  )

# Diferenca de defesa

combats <- combats |>
  dplyr::mutate(
    First_pokemon=sapply(nome_atk,
                         \(x) pokemon$def[match(x, pokemon$nome)]),
    Second_pokemon=sapply(nome_def,
                          \(x) pokemon$def[match(x, pokemon$nome)]),
    diff_def=First_pokemon-Second_pokemon
  )

# Diferenca de ataque especial
```

```
combats <- combats |>
  dplyr::mutate(
    First_pokemon=sapply(nome_atk,
                         \(x) pokemon$satk[match(x, pokemon$nome)]),
    Second_pokemon=sapply(nome_def,
                          \(x) pokemon$satk[match(x, pokemon$nome)]),
    diff_satk=First_pokemon-Second_pokemon
  )

# Diferenca de defesa especial

combats <- combats |>
  dplyr::mutate(
    First_pokemon=sapply(nome_atk,
                         \(x) pokemon$sdef[match(x, pokemon$nome)]),
    Second_pokemon=sapply(nome_def,
                          \(x) pokemon$sdef[match(x, pokemon$nome)]),
    diff_sdef=First_pokemon-Second_pokemon
  )

# Diferenca de velocidade

combats <- combats |>
  dplyr::mutate(
    First_pokemon=sapply(nome_atk,
                         \(x) pokemon$spd[match(x, pokemon$nome)]),
    Second_pokemon=sapply(nome_def,
                          \(x) pokemon$spd[match(x, pokemon$nome)]),
    diff_spd=First_pokemon-Second_pokemon
  )

# Diferenca de hp

combats <- combats |>
  dplyr::mutate(
    First_pokemon=sapply(nome_atk,
                         \(x) pokemon$hp[match(x, pokemon$nome)]),
    Second_pokemon=sapply(nome_def,
                          \(x) pokemon$hp[match(x, pokemon$nome)]),
```

```
    diff_hp=First_pokemon-Second_pokemon
  )

# Variavel de geracao

combats <- combats |>
  dplyr::mutate(
    gen_atk=sapply(nome_atk,
                   \(x) pokemon$geracao[match(x, pokemon$nome)]),
    gen_def=sapply(nome_def,
                   \(x) pokemon$geracao[match(x, pokemon$nome)])
  )

# Variavel de pokemons lendarios

combats <- combats |>
  dplyr::mutate(
    leg_atk=sapply(nome_atk,
                   \(x) pokemon$lendario[match(x, pokemon$nome)]),
    leg_def=sapply(nome_def,
                   \(x) pokemon$lendario[match(x, pokemon$nome)])
  )

## Preparando variavel de vantagem/desvantagem de tipos

combats <- combats |>
  dplyr::mutate(
    tipo1_atk=sapply(nome_atk,
                     \(x) pokemon$tipo_1[match(x, pokemon$nome)]),
    tipo2_atk=sapply(nome_atk,
                     \(x) pokemon$tipo_2[match(x, pokemon$nome)]),
    tipo1_def=sapply(nome_def,
                     \(x) pokemon$tipo_1[match(x, pokemon$nome)]),
    tipo2_def=sapply(nome_def,
                     \(x) pokemon$tipo_2[match(x, pokemon$nome)])
  )

combats[is.na(combats)] <- 'None'

# Funcao para calculo do coeficiente de vantagem/desvantagem
```

```
resist_tipo <- function(atk1,atk2,def){

  res <- tipos[tipos$Attacking==def,]

  vtg1 <- res[names(res)==atk1]

  vtg2 <- res[names(res)==atk2]

  vtg1*vtg2
}

resist_tipo <- Vectorize(resist_tipo)

combats <- combats |>
  dplyr::mutate(
    res_tipo1 = unlist(
      resist_tipo(atk1=as.vector(tipo1_atk),
                  atk2=as.vector(tipo2_atk),
                  def=as.vector(tipo1_def))
    ),
    res_tipo2 = unlist(
      resist_tipo(atk1=as.vector(tipo1_atk),
                  atk2=as.vector(tipo2_atk),
                  def=as.vector(tipo2_def))
    )
  )

## Detecção de outliers via mahalanobis

# Selecionando dataset final

df <- combats |>
  dplyr::mutate(
    across(
      .cols=c(tipo1_atk:tipo2_def,res_tipo1,res_tipo2),
      as.factor
    )
  ) |>
  dplyr::select(
```

```
    win,nome_atk,nome_def,tipo1_atk,tipo2_atk,
    tipo1_def,tipo2_def,diff_hp,diff_atk,
    diff_def,diff_satk,diff_sdef,diff_spd,
    res_tipo1,res_tipo2, gen_atk, gen_def,
    leg_atk, leg_def
  )

# Preparando dataset mahalanobis

df_ma <- df[-c(2:7)] |>
  dplyr::mutate(
    dplyr::across(.fns=as.numeric)
  )

# Realizando calculo das distancias

md <- mahalanobis(df_ma, center = colMeans(df_ma), cov = cov(df_ma))

# Estabelecendo limite

alpha <- .001

lim <- (qchisq(p = 1 - alpha, df = ncol(df_ma)))

# Encontrando e filtrando outliers

out <- which(md > lim)

df <- df[-out,]

## Exportando dataset final

df |>
  readr::write_rds('data/pokebattle.rds')
```

A.2 Análise Descritiva

```
### Lendo dados

df <- readRDS('data/pokebattle.rds')
```

```
### Carregando patchwork para junção de graficos

library(patchwork)

#### Verificando balanceamento

n <- nrow(df)

df$win |>
  table()/n

#### Grafico de Correlacao

cor_win_0 <- df[df$win==0,] |>
  corrr::correlate(method='pearson')|>
  ggplot2::autoplot(high='red')+
  ggplot2::geom_text(ggplot2::aes(label=round(r,2)))+
  ggplot2::theme_minimal(base_size = 16)+
  ggplot2::theme(legend.position = 'none')+
  ggplot2::labs(title='Correlação de Pearson',
                subtitle = 'de covariáveis numéricas (win=0)')

cor_win_1 <- df[df$win==1,] |>
  corrr::correlate(method='pearson') |>
  ggplot2::autoplot(high='red')+
  ggplot2::geom_text(ggplot2::aes(label=round(r,2)))+
  ggplot2::theme_minimal(base_size = 16)+
  ggplot2::theme(legend.position = 'none')+
  ggplot2::labs(title='Correlação de Pearson',
                subtitle = 'de covariáveis numéricas (win=1)')

cor_win_0 + cor_win_1

ggplot2::ggsave('img/1-corr-plot.png')

#### Densidade de variaveis numericas

kernel_hp <- df |>
  ggplot2::ggplot()+
  ggplot2::aes(x=diff_hp)+
```

```
  # Estimativa do Kernel Gaussiano
  ggplot2::stat_density(kernel='gaussian', size=1,
                        ggplot2::aes(color='kernel gaussiano'), alpha=0)+
  # Estimativa Normal
  ggplot2::stat_function(fun=dnorm,n=101,args=list(mean=mean(df$diff_hp),
                                                   sd=sd(df$diff_hp)),
                         ggplot2::aes(color='normal'), size=1)+
  # Kernel parabolico ou epanechnikov
  ggplot2::stat_density(kernel='epanechnikov', alpha=0, size=1,
                        ggplot2::aes(color='kernel parabolico'))+
  # Kernel uniforme
  ggplot2::stat_density(kernel='rectangular', alpha=0, size=1,
                        ggplot2::aes(color='kernel uniforme'))+
  ggplot2::theme_minimal()+
  ggplot2::theme(legend.position = 'none')+
  ggplot2::labs(y='densidade',
                title='diff_hp',
                colour='')+
  ggplot2::coord_cartesian(ylim=c(0.005, 0.015),
                           xlim=c(-50,50))

kernel_atk <- df |>
  ggplot2::ggplot()+
  ggplot2::aes(x=diff_atk)+
  # Estimativa do Kernel Gaussiano
  ggplot2::stat_density(kernel='gaussian', size=1,
                        ggplot2::aes(color='kernel gaussiano'), alpha=0)+
  # Estimativa Normal
  ggplot2::stat_function(fun=dnorm,n=101,args=list(mean=mean(df$diff_atk),
                                                   sd=sd(df$diff_atk)),
                         ggplot2::aes(color='normal'), size=1)+
  # Kernel parabolico ou epanechnikov
  ggplot2::stat_density(kernel='epanechnikov', alpha=0, size=1,
                        ggplot2::aes(color='kernel parabolico'))+
  # Kernel uniforme
  ggplot2::stat_density(kernel='rectangular', alpha=0, size=1,
                        ggplot2::aes(color='kernel uniforme'))+
  ggplot2::theme_minimal()+
  ggplot2::theme(legend.position = 'none')+
  ggplot2::labs(y='densidade',
```

```
                title='diff_atk',
                colour='')+
  ggplot2::coord_cartesian(ylim=c(0.005, 0.01),
                           xlim=c(-50,50))

kernel_def <- df |>
  ggplot2::ggplot()+
  ggplot2::aes(x=diff_def)+
  # Estimativa do Kernel Gaussiano
  ggplot2::stat_density(kernel='gaussian', size=1,
                        ggplot2::aes(color='kernel gaussiano'), alpha=0)+
  # Estimativa Normal
  ggplot2::stat_function(fun=dnorm,n=101,args=list(mean=mean(df$diff_def),
                                                   sd=sd(df$diff_def)),
                         ggplot2::aes(color='normal'), size=1)+
  # Kernel parabolico ou epanechnikov
  ggplot2::stat_density(kernel='epanechnikov', alpha=0, size=1,
                        ggplot2::aes(color='kernel parabolico'))+
  # Kernel uniforme
  ggplot2::stat_density(kernel='rectangular', alpha=0, size=1,
                        ggplot2::aes(color='kernel uniforme'))+
  ggplot2::theme_minimal()+
  ggplot2::theme(legend.position = 'none')+
  ggplot2::labs(y='densidade',
                title='diff_def',
                colour='')+
  ggplot2::coord_cartesian(ylim=c(0.005, 0.011),
                           xlim=c(-50,50))

kernel_satk <- df |>
  ggplot2::ggplot()+
  ggplot2::aes(x=diff_satk)+
  # Estimativa do Kernel Gaussiano
  ggplot2::stat_density(kernel='gaussian', size=1,
                        ggplot2::aes(color='kernel gaussiano'), alpha=0)+
  # Estimativa Normal
  ggplot2::stat_function(fun=dnorm,n=101,args=list(mean=mean(df$diff_satk),
                                                   sd=sd(df$diff_satk)),
                         ggplot2::aes(color='normal'), size=1)+
  # Kernel parabolico ou epanechnikov
```

```
  ggplot2::stat_density(kernel='epanechnikov', alpha=0, size=1,
                        ggplot2::aes(color='kernel parabolico'))+
  # Kernel uniforme
  ggplot2::stat_density(kernel='rectangular', alpha=0, size=1,
                        ggplot2::aes(color='kernel uniforme'))+
  ggplot2::theme_minimal()+
  ggplot2::theme(legend.position = 'right')+
  ggplot2::labs(y='densidade',
                title='diff_satk',
                colour='')+
  ggplot2::coord_cartesian(ylim=c(0.005, 0.0102),
                           xlim=c(-50,50))

kernel_sdef <- df |>
  ggplot2::ggplot()+
  ggplot2::aes(x=diff_sdef)+
  # Estimativa do Kernel Gaussiano
  ggplot2::stat_density(kernel='gaussian', size=1,
                        ggplot2::aes(color='kernel gaussiano'), alpha=0)+
  # Estimativa Normal
  ggplot2::stat_function(fun=dnorm,n=101,args=list(mean=mean(df$diff_sdef),
                                                   sd=sd(df$diff_sdef)),
                         ggplot2::aes(color='normal'), size=1)+
  # Kernel parabolico ou epanechnikov
  ggplot2::stat_density(kernel='epanechnikov', alpha=0, size=1,
                        ggplot2::aes(color='kernel parabolico'))+
  # Kernel uniforme
  ggplot2::stat_density(kernel='rectangular', alpha=0, size=1,
                        ggplot2::aes(color='kernel uniforme'))+
  ggplot2::theme_minimal()+
  ggplot2::theme(legend.position = 'none')+
  ggplot2::labs(y='densidade',
                title='diff_sdef',
                colour='')+
  ggplot2::coord_cartesian(ylim=c(0.005, 0.012),
                           xlim=c(-50,50))

kernel_spd <- df |>
  ggplot2::ggplot()+
  ggplot2::aes(x=diff_spd)+
```

```
  # Estimativa do Kernel Gaussiano
  ggplot2::stat_density(kernel='gaussian', size=1,
                        ggplot2::aes(color='kernel gaussiano'), alpha=0)+
  # Estimativa Normal
  ggplot2::stat_function(fun=dnorm,n=101,args=list(mean=mean(df$diff_spd),
                                                   sd=sd(df$diff_spd)),
                         ggplot2::aes(color='normal'), size=1)+
  # Kernel parabolico ou epanechnikov
  ggplot2::stat_density(kernel='epanechnikov', alpha=0, size=1,
                        ggplot2::aes(color='kernel parabolico'))+
  # Kernel uniforme
  ggplot2::stat_density(kernel='rectangular', alpha=0, size=1,
                        ggplot2::aes(color='kernel uniforme'))+
  ggplot2::theme_minimal()+
  ggplot2::theme(legend.position = 'none')+
  ggplot2::labs(y='densidade',
                title='diff_spd',
                colour='')+
  ggplot2::coord_cartesian(ylim=c(0.005, 0.01),
                           xlim=c(-50,50))

(kernel_hp+kernel_atk)/(kernel_def+kernel_satk)/(kernel_sdef+kernel_spd)

ggplot2::ggsave('img/2-kernel-compar.png')

#### Densidade de variaveis numericas para win=0 e win=1

dif_hp <- df |>
  ggplot2::ggplot()+
  ggplot2::aes(x=diff_hp, fill=win)+
  ggplot2::geom_density(alpha=0.6)+
  ggplot2::theme_minimal()+
  ggplot2::theme(legend.position = 'none')+
  ggplot2::labs(y='densidade', title='diff_hp')

dif_atk <- df |>
  ggplot2::ggplot()+
  ggplot2::aes(x=diff_atk, fill=win)+
  ggplot2::geom_density(alpha=0.6)+
  ggplot2::theme_minimal()+
```

```
  ggplot2::theme(legend.position = 'none')+
  ggplot2::labs(y='densidade', title='diff_atk')

dif_def <- df |>
  ggplot2::ggplot()+
  ggplot2::aes(x=diff_def, fill=win)+
  ggplot2::geom_density(alpha=0.6)+
  ggplot2::theme_minimal()+
  ggplot2::theme(legend.position = 'none')+
  ggplot2::labs(y='densidade', title='diff_def')

dif_satk <- df |>
  ggplot2::ggplot()+
  ggplot2::aes(x=diff_satk, fill=win)+
  ggplot2::geom_density(alpha=0.6)+
  ggplot2::theme_minimal()+
  ggplot2::labs(y='densidade', title='diff_satk')

dif_sdef <- df |>
  ggplot2::ggplot()+
  ggplot2::aes(x=diff_sdef, fill=win)+
  ggplot2::geom_density(alpha=0.6)+
  ggplot2::theme_minimal()+
  ggplot2::theme(legend.position = 'none')+
  ggplot2::labs(y='densidade', title='diff_sdef')

dif_spd <- df |>
  ggplot2::ggplot()+
  ggplot2::aes(x=diff_spd, fill=win)+
  ggplot2::geom_density(alpha=0.6)+
  ggplot2::theme_minimal()+
  ggplot2::theme(legend.position = 'none')+
  ggplot2::labs(y='densidade', title='diff_spd')

(dif_hp+dif_atk)/(dif_def+dif_satk)/(dif_sdef+dif_spd)

ggplot2::ggsave('img/3-dens-win-compar.png')

#### Comparação de pokemon lendario e nao lendario para win=0 e win=1
```

```
leg_atk <- df |>
  dplyr::group_by(leg_atk,win) |>
  dplyr::summarise(n=dplyr::n()) |>
  dplyr::ungroup() |>
  dplyr::mutate(
    dplyr::across(.cols=-3,
                  .fns=as.character)
  ) |>
  ggplot2::ggplot()+
  ggplot2::aes(y=n,x=leg_atk,fill=win)+
  ggplot2::geom_bar(stat='identity', position='fill', color='black')+
  ggplot2::theme_minimal()+
  ggplot2::theme(legend.position = 'left')+
  ggplot2::labs(y='proporção observada',
                title='leg_atk')

leg_def <- df |>
  dplyr::group_by(leg_def,win) |>
  dplyr::summarise(n=dplyr::n()) |>
  dplyr::ungroup() |>
  dplyr::mutate(
    dplyr::across(.cols=-3,
                  .fns=as.character)
  ) |>
  ggplot2::ggplot()+
  ggplot2::aes(y=n,x=leg_def,fill=win)+
  ggplot2::geom_bar(stat='identity', position='fill', color='black')+
  ggplot2::theme_minimal()+
  ggplot2::theme(legend.position = 'none')+
  ggplot2::labs(y='proporção observada',
                title='leg_def')

leg_atk+leg_def

ggplot2::ggsave('img/4-leg-win-compar.png')

#### Comparação de gerações para win=0 e win=1

gen_atk <- df |>
  dplyr::group_by(gen_atk,win) |>
```

```
  dplyr::summarise(n=dplyr::n()) |>
  dplyr::ungroup() |>
  dplyr::mutate(
    dplyr::across(.cols=-3,
                  .fns=as.character)
  ) |>
  ggplot2::ggplot()+
  ggplot2::aes(y=n,x=gen_atk,fill=win)+
  ggplot2::geom_bar(stat='identity', position='fill', color='black')+
  ggplot2::theme_minimal()+
  ggplot2::theme(legend.position = 'none')+
  ggplot2::labs(y='proporção observada',
                title='gen_atk')

gen_def <- df |>
  dplyr::group_by(gen_def,win) |>
  dplyr::summarise(n=dplyr::n()) |>
  dplyr::ungroup() |>
  dplyr::mutate(
    dplyr::across(.cols=-3,
                  .fns=as.character)
  ) |>
  ggplot2::ggplot()+
  ggplot2::aes(y=n,x=gen_def,fill=win)+
  ggplot2::geom_bar(stat='identity', position='fill', color='black')+
  ggplot2::theme_minimal()+
  ggplot2::theme(legend.position = 'top')+
  ggplot2::labs(y='proporção observada',
                title='gen_def')

gen_atk/gen_def

ggplot2::ggsave('img/5-gen-win-compar.png')

#### Comparacao de resistencias para win0 e win1

res1 <- df |>
  dplyr::group_by(res_tipo1, win) |>
  dplyr::summarise(n=dplyr::n()) |>
  dplyr::ungroup() |>
```

```
  dplyr::mutate(
    dplyr::across(.cols=-3,
                  .fns=as.character),
    res_tipo1=dplyr::case_when(
      res_tipo1=='0' ~ '1-Imune',
      res_tipo1=='0.25' ~ '2-Muito Resistente',
      res_tipo1=='0.5' ~ '3-Resistente',
      res_tipo1=='1' ~ '4-Neutro',
      res_tipo1=='2' ~ '5-Fraco',
      res_tipo1=='4' ~ '6-Muito Fraco'
    )
  ) |>
  ggplot2::ggplot()+
  ggplot2::aes(y=n,x=res_tipo1,fill=win)+
  ggplot2::geom_bar(stat='identity', position='fill', color='black')+
  ggplot2::theme_minimal()+
  ggplot2::theme(legend.position = 'none')+
  ggplot2::labs(y='proporção observada',
                title='res_tipo1')

res2 <- df |>
  dplyr::group_by(res_tipo2, win) |>
  dplyr::summarise(n=dplyr::n()) |>
  dplyr::ungroup() |>
  dplyr::mutate(
    dplyr::across(.cols=-3,
                  .fns=as.character),
    res_tipo2=dplyr::case_when(
      res_tipo2=='0' ~ '1-Imune',
      res_tipo2=='0.25' ~ '2-Muito Resistente',
      res_tipo2=='0.5' ~ '3-Resistente',
      res_tipo2=='1' ~ '4-Neutro',
      res_tipo2=='2' ~ '5-Fraco',
      res_tipo2=='4' ~ '6-Muito Fraco'
    )
  ) |>
  ggplot2::ggplot()+
  ggplot2::aes(y=n,x=res_tipo2,fill=win)+
  ggplot2::geom_bar(stat='identity', position='fill', color='black')+
  ggplot2::theme_minimal()+
```

```
  ggplot2::theme(legend.position = 'top')+
  ggplot2::labs(y='proporção observada',
                title='res_tipo2')

res1/res2

ggplot2::ggsave('img/5-res-win-compar.png')

#### GRAFICOS SECAO DE RESULTADOS

### Estimação da rede bayesiana com base nos dados

df[,c(1,8:13)] |>
  bnlearn::tabu() |>
  plot()
```

A.3 Ajustes considerando a variável *diff_spd*

```
# Lendo banco de dados

df <- readr::read_rds('data/pokebattle.rds') |>
   dplyr::select(-c(2:7))

# Definindo funcao para extracao de metricas

metricas_extract <- function(conf, roc){
  acc <- conf$overall[1] # Acuracia

  risk <- 1-acc # Risco

  other <- conf$byClass[1:4] # Sensibilidade, Especificidade, VPP e VPN

  auc <- roc$auc[1] # Area sob Curva ROC

  result <- c(acc, risk, other, auc)

  names(result) <- c('Acurácia', 'Risco', 'Sensibilidade', 'Especificidade',
                     'VPP', 'VPN', 'AUC')
  return(result)
}
```

```
### Dividindo dados em treino, teste e validação

## Treino e teste

set.seed(1985)

split_tre <- rsample::initial_split(df, prop=0.7)

tre <- rsample::training(split_tre)
tes <- rsample::testing(split_tre)

x_tre <- tre[,-1] |>
  as.matrix.data.frame() |>
  as.double() |>
  matrix(ncol=12)

y_tre <- tre[,1] |>
  as.matrix() |>
  as.double() |>
  matrix(ncol=1)

x_tes <- tes[,-1] |>
  as.matrix.data.frame() |>
  as.double() |>
  matrix(ncol=12)

y_tes <- tre[,1] |>
  as.matrix.data.frame() |>
  as.double() |>
  matrix(ncol=1)

# Verificando balanceamento no treino e no teste

tibble::tibble(
  Conjunto=c('Treino', 'Teste'),
  `win=1`= c(mean(tre$win==1), mean(tes$win==1)),
  `win=0`= c(mean(tre$win==0), mean(tes$win==0))
) |> knitr::kable('latex', digits=4)

## Validação
```

```
split_val <- rsample::initial_split(tre, prop=0.6)

tre_val <- rsample::training(split_val)
val <- rsample::testing(split_val)

x_tre_val <- tre_val[,-1] |>
  as.matrix.data.frame() |>
  as.double() |>
  matrix(ncol=12)

y_tre_val <- tre_val[,1] |>
  as.matrix() |>
  as.double() |>
  matrix(ncol=1)

x_val <- val[,-1] |>
  as.matrix.data.frame() |>
  as.double() |>
  matrix(ncol=12)

y_val <- val[,1] |>
  as.matrix.data.frame() |>
  as.double() |>
  matrix(ncol=1)

# Verificando balanceamento no treino e validacao

tibble::tibble(
  Conjunto=c('Treino', 'Validação'),
  `win=1`= c(mean(tre_val$win==1), mean(val$win==1)),
  `win=0`= c(mean(tre_val$win==0), mean(val$win==0))
) |> knitr::kable('latex', digits=4)

###### COMPARACOES ENTRE MODELOS COMPLETOS ######

### AJUSTE BI GAUSSIANO

ajs_gaus <- naivebayes::naive_bayes(win~., data=tre, usekernel=FALSE)
```

```
### AJUSTE REG. LOG.

ajs_log <- glm(win~., data=tre, family = binomial)

### AJUSTE BI FLEXIVEL

## Ajustando para cada kernel

ajs_flex_gaus <- naivebayes::naive_bayes(win~., data=tre_val,
                                         usekernel=TRUE,
                                         kernel='gaussian')

ajs_flex_uni <- naivebayes::naive_bayes(win~., data=tre_val,
                                        usekernel=TRUE,
                                        kernel='rectangular')

ajs_flex_epa <- naivebayes::naive_bayes(win~., data=tre_val,
                                        usekernel=TRUE,
                                        kernel='epanechnikov')

## Obtendo matriz de confusao para cada kernel

conf_flex_gaus <- caret::confusionMatrix(
  data=predict(ajs_flex_gaus, newdata = val[,-1]),
  reference=val$win
)

conf_flex_uni <- caret::confusionMatrix(
  data=predict(ajs_flex_uni, newdata = val[,-1]),
  reference=val$win
)

conf_flex_epa <- caret::confusionMatrix(
  data=predict(ajs_flex_epa, newdata = val[,-1]),
  reference=val$win
)

## Plotando curva ROC dos tres ajustes flexiveis

roc_flex_gaus <- pROC::roc(response = val$win,
```

```
                          predictor = predict(ajs_flex_gaus, newdata = val[,-1],
                                              type='prob')[,1])

roc_flex_uni <- pROC::roc(response = val$win,
                          predictor = predict(ajs_flex_uni, newdata = val[,-1],
                                              type='prob')[,1])

roc_flex_epa <- pROC::roc(response = val$win,
                         predictor = predict(ajs_flex_epa, newdata = val[,-1],
                                             type='prob')[,1])

pROC::ggroc(
  list(Parabolico=roc_flex_epa, Uniforme=roc_flex_uni,
       Gaussiano=roc_flex_gaus),
  legacy.axes=TRUE, size=1
)+
  ggplot2::labs(color='Kernel', x='1-Especificidade', y='Sensibilidade',
                title='Curvas ROC para ajustes via Bayes Ingênuo Flexível',
                subtitle = 'considerando Kernels Gaussiano, Uniforme e Parabólico (com diff_spd)')+
  ggplot2::theme_minimal()+
  ggplot2::theme(legend.position = 'top')

ggplot2::ggsave('img/6-roc-completo-flex.png')

## Apresentando metricas

tibble::tibble(
  Metrica=names(metricas_extract(conf_flex_epa, roc_flex_epa)),
  Gaussiano=metricas_extract(conf_flex_gaus, roc_flex_gaus),
  Uniforme=metricas_extract(conf_flex_uni, roc_flex_uni),
  Parabolico=metricas_extract(conf_flex_epa, roc_flex_epa)
) |>
  tidyr::pivot_longer(!Metrica) |>
  tidyr::pivot_wider(names_from=Metrica) |>
  dplyr::rename(Kernel=name) |>
  knitr::kable('latex', digits=4)

## Escolhido BI Flexível com kernel gaussiano

ajs_flex <- naivebayes::naive_bayes(win~., data=tre,
```

```
                                    usekernel=TRUE,
                                    kernel='gaussian')

### COMPARANDO MODELOS COMPLETOS

## Matrizes de Confusao

conf_gaus <- caret::confusionMatrix(
  data = predict(ajs_gaus, newdata = tes[,-1]),
  reference = tes$win
)

conf_flex <- caret::confusionMatrix(
  data = predict(ajs_flex, newdata = tes[,-1]),
  reference = tes$win
)

pred_log <- predict(ajs_log, newdata=tes[,-1], type='response')
pred_log <- as.factor(ifelse(pred_log>=0.5, 1, 0))
conf_log <- caret::confusionMatrix(
  data=pred_log,
  reference=tes$win
)

## Curvas ROC

roc_gaus <- pROC::roc(response = tes$win,
                      predictor = predict(ajs_gaus, newdata = tes[,-1],
                                          type='prob')[,1])

roc_flex <- pROC::roc(response = tes$win,
                      predictor = predict(ajs_flex, newdata = tes[,-1],
                                          type='prob')[,1])

roc_log <- pROC::roc(response = tes$win,
                     predictor = predict(ajs_log, newdata = tes[,-1],
                                         type='response'))

pROC::ggroc(
  list(`BI Gaussiano`=roc_gaus, `BI Flexível`=roc_flex,
```

```
       'Reg. Logistica'=roc_log),
  legacy.axes=TRUE, size=1
)+
  ggplot2::labs(color='Ajuste', x='1-Especificidade', y='Sensibilidade',
                title='Curvas ROC para Modelos Completos',
                subtitle = 'com BI Gaussiano, BI Flexível e Reg. Logística (com diff_spd)')+
  ggplot2::theme_minimal()+
  ggplot2::theme(legend.position = 'top')

ggplot2::ggsave('img/7-roc-completo-all.png')

## Apresentando metricas

tibble::tibble(
  Metrica=names(metricas_extract(conf_flex, roc_flex)),
  'BI Gaussiano' = metricas_extract(conf_gaus, roc_gaus),
  'BI Flexível'=metricas_extract(conf_flex, roc_flex),
  'Regressão Logística'=metricas_extract(conf_log, roc_log)
) |>
  tidyr::pivot_longer(!Metrica) |>
  tidyr::pivot_wider(names_from=Metrica) |>
  dplyr::rename(Ajuste=name) |>
  knitr::kable('latex', digits=4)

###### AJUSTES COM A VARIAVEL DIFF_SPD ######

#### ESCOLHENDO VARIAVEIS PARA NB

var_lasso <- glmnet::cv.glmnet(x=x_tre,y=y_tre, alpha=1)
var_lasso <- glmnet::glmnet(x=x_tre,y=y_tre, alpha=1,
                            lambda = var_lasso$lambda.1se) |>
  coef()
rownames(var_lasso) <- names(tre)

var_lasso |> round(5) ## diff_hp, diff_atk, diff_sdef, diff_spd, res_tipo1, res_tipo2

var_arv <- rpart::rpart(win~., data=tre) # diff_spd
rpart.plot::rpart.plot(var_arv)

#### VALIDACAO NB GAUSSIANO
```

```
## Ajustes

ajs_gaus_full <- naivebayes::naive_bayes(win~., data=tre_val,
                                         usekernel=FALSE)

ajs_gaus_arv <- naivebayes::naive_bayes(win~., data=tre_val[,c(1,7)],
                                        usekernel=FALSE)

ajs_gaus_lasso <- naivebayes::naive_bayes(win~., data=tre_val[,c(1,2,3,6,7,8,9)],
                                          usekernel=FALSE)

## Matrizes de Confusão

conf_full_gaus <- caret::confusionMatrix(
  data=predict(ajs_gaus_full, val),
  reference=val$win
)

conf_arv_gauss <- caret::confusionMatrix(
  data=predict(ajs_gaus_arv, val),
  reference=val$win
)

conf_lasso_gauss <- caret::confusionMatrix(
  data=predict(ajs_gaus_lasso, val),
  reference=val$win
)

## Curvas ROC

roc_full_gaus <- pROC::roc(response = val$win,
                           predictor = predict(ajs_gaus_full, newdata = val[,-1],
                                               type='prob')[,1])

roc_arv_gaus <- pROC::roc(response = val$win,
                          predictor = predict(ajs_gaus_arv, newdata = val[,-1],
                                              type='prob')[,1])

roc_lasso_gaus <- pROC::roc(response = val$win,
```

```
                          predictor = predict(ajs_gaus_lasso, newdata = val[,-1],
                                              type='prob')[,1])

pROC::ggroc(
  list(`Completo`=roc_full_gaus, `Arvore`=roc_arv_gaus,
       `Lasso`=roc_lasso_gaus),
  legacy.axes=TRUE, size=1
)+
  ggplot2::labs(color='Seleção', x='1-Especificidade', y='Sensibilidade',
                title='Curvas ROC para Bayes Ingênuo Gaussiano',
                subtitle = 'com modelo completo e variáveis via Árvore e Lasso (com diff_spd)')+
  ggplot2::theme_minimal()+
  ggplot2::theme(legend.position = 'top')

ggplot2::ggsave('img/9-roc-gaus-diff-spd.png')

## Apresentando Métricas

tibble::tibble(
  Metrica=names(metricas_extract(conf_arv_gauss, roc_arv_gaus)),
  `Completo` = metricas_extract(conf_full_gaus, roc_full_gaus),
  `Arvore`=metricas_extract(conf_arv_gauss, roc_arv_gaus),
  `Lasso`=metricas_extract(conf_lasso_gauss, roc_lasso_gaus)
) |>
  tidyr::pivot_longer(!Metrica) |>
  tidyr::pivot_wider(names_from=Metrica) |>
  dplyr::rename(Ajuste=name) |>
  knitr::kable('latex', digits=4)

#### VALIDACAO NB FLEXIVEL

## Ajustes

# Arvore
ajs_flex_epa_arv <- naivebayes::naive_bayes(win~., data=tre_val[,c(1,7)],
                                            usekernel=TRUE,
                                            kernel='epanechnikov')

ajs_flex_uni_arv <- naivebayes::naive_bayes(win~., data=tre_val[,c(1,7)],
                                            usekernel=TRUE,
```

```
                                          kernel='rectangular')

ajs_flex_gaus_arv <- naivebayes::naive_bayes(win~., data=tre_val[,c(1,7)],
                                           usekernel=TRUE,
                                           kernel='gaussian')

# Lasso
ajs_flex_epa_las <- naivebayes::naive_bayes(win~., data=tre_val[,c(1,2,3,6,7,8,9)],
                                          usekernel=TRUE,
                                          kernel='epanechnikov')

ajs_flex_uni_las <- naivebayes::naive_bayes(win~., data=tre_val[,c(1,2,3,6,7,8,9)],
                                          usekernel=TRUE,
                                          kernel='rectangular')

ajs_flex_gaus_las <- naivebayes::naive_bayes(win~., data=tre_val[,c(1,2,3,6,7,8,9)],
                                           usekernel=TRUE,
                                           kernel='gaussian')

## Matrizes de Confusão

# Arvore
conf_arv_flex_epa <- caret::confusionMatrix(
  data=predict(ajs_flex_epa_arv, val),
  reference=val$win
)

conf_arv_flex_uni <- caret::confusionMatrix(
  data=predict(ajs_flex_uni_arv, val),
  reference=val$win
)

conf_arv_flex_gaus <- caret::confusionMatrix(
  data=predict(ajs_flex_gaus_arv, val),
  reference=val$win
)

# Lasso
conf_las_flex_epa <- caret::confusionMatrix(
  data=predict(ajs_flex_epa_las, val),
```

```
  reference=val$win
)

conf_las_flex_uni <- caret::confusionMatrix(
  data=predict(ajs_flex_uni_las, val),
  reference=val$win
)

conf_las_flex_gaus <- caret::confusionMatrix(
  data=predict(ajs_flex_gaus_las, val),
  reference=val$win
)

## Curvas ROC

#Arvore
roc_arv_flex_epa <- pROC::roc(response = val$win,
                              predictor = predict(ajs_flex_epa_arv, newdata = val[,-1],
                                                  type='prob')[,1])

roc_arv_flex_uni <- pROC::roc(response = val$win,
                              predictor = predict(ajs_flex_uni_arv, newdata = val[,-1],
                                                  type='prob')[,1])

roc_arv_flex_gaus <- pROC::roc(response = val$win,
                              predictor = predict(ajs_flex_gaus_arv, newdata = val[,-1],
                                                  type='prob')[,1])

pROC::ggroc(
  list(`Parabólico`=roc_arv_flex_epa, `Uniforme`=roc_arv_flex_uni,
       `Gaussiano`=roc_arv_flex_gaus),
  legacy.axes=TRUE, size=1
)+
  ggplot2::labs(color='Kernel', x='1-Especificidade', y='Sensibilidade',
                title='Curvas ROC para Bayes Ingênuo Flexível',
                subtitle = 'com variáveis via Árvore (com diff_spd)')+
  ggplot2::theme_minimal()+
  ggplot2::theme(legend.position = 'top')

ggplot2::ggsave('img/10-roc-flex-arv-diff-spd.png')
```

```
# Lasso
roc_las_flex_epa <- pROC::roc(response = val$win,
                              predictor = predict(ajs_flex_epa_las, newdata = val[,-1],
                                                  type='prob')[,1])

roc_las_flex_uni <- pROC::roc(response = val$win,
                              predictor = predict(ajs_flex_uni_las, newdata = val[,-1],
                                                  type='prob')[,1])

roc_las_flex_gaus <- pROC::roc(response = val$win,
                               predictor = predict(ajs_flex_gaus_las, newdata = val[,-1],
                                                   type='prob')[,1])

pROC::ggroc(
  list(`Parabólico`=roc_las_flex_epa, `Uniforme`=roc_las_flex_uni,
       `Gaussiano`=roc_las_flex_gaus),
  legacy.axes=TRUE, size=1
)+
  ggplot2::labs(color='Kernel', x='1-Especificidade', y='Sensibilidade',
                title='Curvas ROC para Bayes Ingênuo Flexível',
                subtitle = 'com variáveis via Lasso (com diff_spd)')+
  ggplot2::theme_minimal()+
  ggplot2::theme(legend.position = 'top')

ggplot2::ggsave('img/11-roc-flex-lasso-diff-spd.png')

## Apresentando Metricas

# Arvore
tibble::tibble(
  Metrica=names(metricas_extract(conf_arv_flex_epa, roc_arv_flex_epa)),
  `Parabolico` = metricas_extract(conf_arv_flex_epa, roc_arv_flex_epa),
  `Uniforme`=metricas_extract(conf_arv_flex_uni, roc_arv_flex_uni),
  `Gaussiano`=metricas_extract(conf_arv_flex_gaus, roc_arv_flex_gaus)
) |>
  tidyr::pivot_longer(!Metrica) |>
  tidyr::pivot_wider(names_from=Metrica) |>
  dplyr::rename(Ajuste=name) |>
  knitr::kable('latex', digits=4)
```

```
# Lasso
tibble::tibble(
  Metrica=names(metricas_extract(conf_las_flex_epa, roc_las_flex_epa)),
  ‘Parabolico‘ = metricas_extract(conf_las_flex_epa, roc_las_flex_epa),
  ‘Uniforme‘=metricas_extract(conf_las_flex_uni, roc_las_flex_uni),
  ‘Gaussiano‘=metricas_extract(conf_las_flex_gaus, roc_las_flex_gaus)
) |>
  tidyr::pivot_longer(!Metrica) |>
  tidyr::pivot_wider(names_from=Metrica) |>
  dplyr::rename(Ajuste=name) |>
  knitr::kable('latex', digits=4)

## Curva ROC melhores árvore e lasso

pROC::ggroc(
  list(‘Arvore‘=roc_arv_flex_gaus, ‘Lasso‘=roc_las_flex_gaus,
       ‘Completo‘=roc_flex_gaus),
  legacy.axes=TRUE, size=1
)+
  ggplot2::labs(color='Seleção', x='1-Especificidade', y='Sensibilidade',
                title='Curvas ROC para Bayes Ingênuo Flexível',
                subtitle = 'com modelo completo e variáveis via Árvore e Lasso (com diff_spd)')+
  ggplot2::theme_minimal()+
  ggplot2::theme(legend.position = 'top')

ggplot2::ggsave('img/12-roc-flex-all-diff-spd.png')

## Metricas melhores arvore e lasso

tibble::tibble(
  Metrica=names(metricas_extract(conf_las_flex_gaus, roc_las_flex_gaus)),
  ‘Completo‘ = metricas_extract(conf_flex_gaus, roc_flex_gaus),
  ‘Arvore‘=metricas_extract(conf_arv_flex_gaus, roc_arv_flex_gaus),
  ‘Lasso‘=metricas_extract(conf_las_flex_gaus, roc_las_flex_gaus)
) |>
  tidyr::pivot_longer(!Metrica) |>
  tidyr::pivot_wider(names_from=Metrica) |>
  dplyr::rename(Ajuste=name) |>
  knitr::kable('latex', digits=4)
```

```
#### VALIDACAO REG. LOGISTICA

## Ajustes

# Completo
ajs_log_full <- glm(win~., data=tre_val, family = binomial)

# Lasso

ajs_log_lasso <- glmnet::cv.glmnet(x=x_tre_val,y=y_tre_val,
                                   alpha=1, family='binomial')
ajs_log_lasso <- glmnet::glmnet(x=x_tre_val,y=y_tre_val, alpha=1,
                                lambda = ajs_log_lasso$lambda.1se,
                                family='binomial')

## Matrizes de Confusao

# Completo
pred_log_full <- predict(ajs_log_full, newdata=val[,-1], type='response')
pred_log_full <- as.factor(ifelse(pred_log_full>=0.5, 1, 0))

conf_log_full <- caret::confusionMatrix(
  data=pred_log_full,
  reference=val$win
)

# Lasso

pred_log_lasso <- predict(ajs_log_lasso, x_val,type='response')
pred_log_lasso <- as.factor(ifelse(pred_log_lasso>=0.5,1,0))

conf_log_lasso <- caret::confusionMatrix(
  data=pred_log_lasso,
  reference=val$win
)

## Curvas ROC

roc_log_full <- pROC::roc(response = val$win,
```

```
                          predictor = predict(ajs_log_full, newdata = val[,-1],
                                              type='response'))

roc_log_lasso <- pROC::roc(response = val$win,
                           predictor = as.vector(predict(ajs_log_lasso, newx = x_val,
                                                 type='response')))

pROC::ggroc(
  list(`Completo`=roc_log_full, `Lasso`=roc_log_lasso),
  legacy.axes=TRUE, size=1
)+
  ggplot2::labs(color='Seleção', x='1-Especificidade', y='Sensibilidade',
                title='Curvas ROC para Regressão Logística',
                subtitle = 'com modelo completo e variáveis via Lasso (com diff_spd)')+
  ggplot2::theme_minimal()+
  ggplot2::theme(legend.position = 'top')

ggplot2::ggsave('img/13-roc-log-diff-spd.png')

## Apresentando Metricas

tibble::tibble(
  Metrica=names(metricas_extract(conf_log_full, roc_log_full)),
  `Completo` = metricas_extract(conf_log_full, roc_log_full),
  `Lasso`=metricas_extract(conf_log_lasso, roc_log_lasso)
) |>
  tidyr::pivot_longer(!Metrica) |>
  tidyr::pivot_wider(names_from=Metrica) |>
  dplyr::rename(Ajuste=name) |>
  knitr::kable('latex', digits=4)

#### COMPARANDO MELHORES MODELOS

## Ajustes

# Lasso
ajs_log <- glmnet::cv.glmnet(x=x_tre,y=y_tre,
                                  alpha=1, family='binomial')
ajs_log <- glmnet::glmnet(x=x_tre,y=y_tre, alpha=1,
                                lambda = ajs_log$lambda.1se,
```

```
                                family='binomial')

# Arvore
ajs_arv <- rpart::rpart(win~., data=tre)

# BI Gaussiano
ajs_gaus <- naivebayes::naive_bayes(win~., data=tre[,c(1,7)],
                                    usekernel=FALSE)

# BI Flexivel
ajs_flex <- naivebayes::naive_bayes(win~., data=tre[,c(1,7)],
                                    usekernel=TRUE,
                                    kernel='gaussian')

## Matrizes de Confusao

# Lasso
pred_log <- predict(ajs_log, x_tes,type='response')
pred_log <- as.factor(ifelse(pred_log>=0.5,1,0))
conf_log <- caret::confusionMatrix(
  data=pred_log,
  reference=tes$win
)

# Arvore
conf_arv <- caret::confusionMatrix(
  data=predict(ajs_arv, tes[,-1], type='class'),
  reference=tes$win
)

# BI Gaussiano
conf_gaus <- caret::confusionMatrix(
  data=predict(ajs_gaus, tes[,-1]),
  reference=tes$win
)

# BI Flexível
conf_flex <- caret::confusionMatrix(
  data=predict(ajs_flex, tes[,-1]),
  reference=tes$win
```

```
)

## Curvas ROC

# Lasso
roc_log <- pROC::roc(response = tes$win,
                     predictor = as.vector(predict(ajs_log, newx = x_tes,
                                                   type='response')))

# Arvore
roc_arv <- pROC::roc(response = tes$win,
                     predictor = predict(ajs_arv, tes[,-1])[,2])

# BI Gaussiano
roc_gaus <- pROC::roc(response = tes$win,
                      predictor = predict(ajs_gaus, tes[,-1], type='prob')[,1])

# BI Flexível
roc_flex <- pROC::roc(response = tes$win,
                      predictor = predict(ajs_flex, tes[,-1], type='prob')[,1])

pROC::ggroc(
  list(`Logistica`=roc_log, `Arvore`=roc_arv, `BI Gaussiano`=roc_gaus,
       `BI Flexível`=roc_flex),
  legacy.axes=TRUE, size=1
)+
  ggplot2::labs(color='Ajuste', x='1-Especificidade', y='Sensibilidade',
                title='Curvas ROC para Melhores Ajustes',
                subtitle = 'considerando a variável diff_spd')+
  ggplot2::theme_minimal()+
  ggplot2::theme(legend.position = 'top')

ggplot2::ggsave('img/14-roc-all-diff-spd.png')

## Apresentando Metricas

tibble::tibble(
  Metrica=names(metricas_extract(conf_log, roc_log)),
  `Logística` = metricas_extract(conf_log, roc_log),
  `Árvore`=metricas_extract(conf_arv, roc_arv),
```

```
  `BI Gaussiano`=metricas_extract(conf_gaus, roc_gaus),
  `BI Flexível`=metricas_extract(conf_flex, roc_flex)
) |>
  tidyr::pivot_longer(!Metrica) |>
  tidyr::pivot_wider(names_from=Metrica) |>
  dplyr::rename(Ajuste=name) |>
  knitr::kable('latex', digits=4)
```

A.4 Ajustes desconsiderando a variável *diff_spd*

```
# Lendo banco de dados (sem a variavel diff_spd, coluna 13)

df <- readr::read_rds('data/pokebattle.rds') |>
  dplyr::select(-c(2:7, 13))

# Definindo funcao para extracao de metricas

metricas_extract <- function(conf, roc){
  acc <- conf$overall[1] # Acuracia

  risk <- 1-acc # Risco

  other <- conf$byClass[1:4] # Sensibilidade, Especificidade, VPP e VPN

  auc <- roc$auc[1] # Area sob Curva ROC

  result <- c(acc, risk, other, auc)

  names(result) <- c('Acurácia', 'Risco', 'Sensibilidade', 'Especificidade',
                     'VPP', 'VPN', 'AUC')
  return(result)
}

### Dividindo dados em treino, teste e validação

## Treino e teste

set.seed(1732)

split_tre <- rsample::initial_split(df, prop=0.7)
```

```
tre <- rsample::training(split_tre)
tes <- rsample::testing(split_tre)

x_tre <- tre[,-1] |>
  as.matrix.data.frame() |>
  as.double() |>
  matrix(ncol=11)

y_tre <- tre[,1] |>
  as.matrix() |>
  as.double() |>
  matrix(ncol=1)

x_tes <- tes[,-1] |>
  as.matrix.data.frame() |>
  as.double() |>
  matrix(ncol=11)

y_tes <- tre[,1] |>
  as.matrix.data.frame() |>
  as.double() |>
  matrix(ncol=1)

# Verificando balanceamento no treino e no teste

tibble::tibble(
  Conjunto=c('Treino', 'Teste'),
  `win=1`= c(mean(tre$win==1), mean(tes$win==1)),
  `win=0`= c(mean(tre$win==0), mean(tes$win==0))
) |> knitr::kable('latex', digits=4)

## Validação

split_val <- rsample::initial_split(tre, prop=0.6)

tre_val <- rsample::training(split_val)
val <- rsample::testing(split_val)

x_tre_val <- tre_val[,-1] |>
```

```
  as.matrix.data.frame() |>
  as.double() |>
  matrix(ncol=11)

y_tre_val <- tre_val[,1] |>
  as.matrix() |>
  as.double() |>
  matrix(ncol=1)

x_val <- val[,-1] |>
  as.matrix.data.frame() |>
  as.double() |>
  matrix(ncol=11)

y_val <- val[,1] |>
  as.matrix.data.frame() |>
  as.double() |>
  matrix(ncol=1)

# Verificando balanceamento no treino e validacao

tibble::tibble(
  Conjunto=c('Treino', 'Validação'),
  `win=1`= c(mean(tre_val$win==1), mean(val$win==1)),
  `win=0`= c(mean(tre_val$win==0), mean(val$win==0))
) |> knitr::kable('latex', digits=4)

###### AJUSTES SEM A VARIAVEL DIFF_SPD ######

#### ESCOLHENDO VARIAVEIS PARA NB

var_lasso <- glmnet::cv.glmnet(x=x_tre,y=y_tre, alpha=1)
var_lasso <- glmnet::glmnet(x=x_tre,y=y_tre, alpha=1,
                            lambda = var_lasso$lambda.1se) |>
  coef()
rownames(var_lasso) <- names(tre)

var_lasso |> round(5) ## todas menos gen_atk e gen_def

var_arv <- rpart::rpart(win~., data=tre) # diff_atk, diff_satk
```

```
rpart.plot::rpart.plot(var_arv)

#### VALIDACAO NB GAUSSIANO

## Ajustes

ajs_gaus_full <- naivebayes::naive_bayes(win~., data=tre_val,
                                         usekernel=FALSE)

ajs_gaus_arv <- naivebayes::naive_bayes(win~., data=tre_val[,c(1,3,5)],
                                        usekernel=FALSE)

ajs_gaus_lasso <- naivebayes::naive_bayes(win~., data=tre_val[,-c(9,10)],
                                          usekernel=FALSE)

## Matrizes de Confusão

conf_full_gaus <- caret::confusionMatrix(
  data=predict(ajs_gaus_full, val),
  reference=val$win
)

conf_arv_gauss <- caret::confusionMatrix(
  data=predict(ajs_gaus_arv, val),
  reference=val$win
)

conf_lasso_gauss <- caret::confusionMatrix(
  data=predict(ajs_gaus_lasso, val),
  reference=val$win
)

## Curvas ROC

roc_full_gaus <- pROC::roc(response = val$win,
                           predictor = predict(ajs_gaus_full, newdata = val[,-1],
                                               type='prob')[,1])

roc_arv_gaus <- pROC::roc(response = val$win,
                          predictor = predict(ajs_gaus_arv, newdata = val[,-1],
```

```
                                                type='prob')[,1])

roc_lasso_gaus <- pROC::roc(response = val$win,
                            predictor = predict(ajs_gaus_lasso, newdata = val[,-1],
                                                type='prob')[,1])

pROC::ggroc(
  list(`Completo`=roc_full_gaus, `Arvore`=roc_arv_gaus,
       `Lasso`=roc_lasso_gaus),
  legacy.axes=TRUE, size=1
)+
  ggplot2::labs(color='Seleção', x='1-Especificidade', y='Sensibilidade',
                title='Curvas ROC para Bayes Ingênuo Gaussiano',
                subtitle = 'com modelo completo e variáveis via Árvore e Lasso (sem diff_spd)')+
  ggplot2::theme_minimal()+
  ggplot2::theme(legend.position = 'top')

ggplot2::ggsave('img/16-roc-gaus-sem-diff.png')

## Apresentando Métricas

tibble::tibble(
  Metrica=names(metricas_extract(conf_arv_gauss, roc_arv_gaus)),
  `Completo` = metricas_extract(conf_full_gaus, roc_full_gaus),
  `Arvore`=metricas_extract(conf_arv_gauss, roc_arv_gaus),
  `Lasso`=metricas_extract(conf_lasso_gauss, roc_lasso_gaus)
) |>
  tidyr::pivot_longer(!Metrica) |>
  tidyr::pivot_wider(names_from=Metrica) |>
  dplyr::rename(Ajuste=name) |>
  knitr::kable('latex', digits=4)

#### VALIDACAO NB FLEXIVEL

## Ajustes

# Completo
ajs_flex_gaus <- naivebayes::naive_bayes(win~., data=tre_val,
                                         usekernel=TRUE,
                                         kernel='gaussian')
```

```
ajs_flex_uni <- naivebayes::naive_bayes(win~., data=tre_val,
                                        usekernel=TRUE,
                                        kernel='rectangular')

ajs_flex_epa <- naivebayes::naive_bayes(win~., data=tre_val,
                                        usekernel=TRUE,
                                        kernel='epanechnikov')

# Arvore
ajs_flex_epa_arv <- naivebayes::naive_bayes(win~., data=tre_val[,c(1,3,5)],
                                            usekernel=TRUE,
                                            kernel='epanechnikov')

ajs_flex_uni_arv <- naivebayes::naive_bayes(win~., data=tre_val[,c(1,3,5)],
                                            usekernel=TRUE,
                                            kernel='rectangular')

ajs_flex_gaus_arv <- naivebayes::naive_bayes(win~., data=tre_val[,c(1,3,5)],
                                             usekernel=TRUE,
                                             kernel='gaussian')

# Lasso
ajs_flex_epa_las <- naivebayes::naive_bayes(win~., data=tre_val[,-c(9,10)],
                                            usekernel=TRUE,
                                            kernel='epanechnikov')

ajs_flex_uni_las <- naivebayes::naive_bayes(win~., data=tre_val[,-c(9,10)],
                                            usekernel=TRUE,
                                            kernel='rectangular')

ajs_flex_gaus_las <- naivebayes::naive_bayes(win~., data=tre_val[,-c(9,10)],
                                             usekernel=TRUE,
                                             kernel='gaussian')

## Matrizes de Confusão

# Completo
conf_flex_gaus <- caret::confusionMatrix(
  data=predict(ajs_flex_gaus, newdata = val[,-1]),
```

```
  reference=val$win
)

conf_flex_uni <- caret::confusionMatrix(
  data=predict(ajs_flex_uni, newdata = val[,-1]),
  reference=val$win
)

conf_flex_epa <- caret::confusionMatrix(
  data=predict(ajs_flex_epa, newdata = val[,-1]),
  reference=val$win
)

# Arvore
conf_arv_flex_epa <- caret::confusionMatrix(
  data=predict(ajs_flex_epa_arv, val),
  reference=val$win
)

conf_arv_flex_uni <- caret::confusionMatrix(
  data=predict(ajs_flex_uni_arv, val),
  reference=val$win
)

conf_arv_flex_gaus <- caret::confusionMatrix(
  data=predict(ajs_flex_gaus_arv, val),
  reference=val$win
)

# Lasso
conf_las_flex_epa <- caret::confusionMatrix(
  data=predict(ajs_flex_epa_las, val),
  reference=val$win
)

conf_las_flex_uni <- caret::confusionMatrix(
  data=predict(ajs_flex_uni_las, val),
  reference=val$win
)
```

```
conf_las_flex_gaus <- caret::confusionMatrix(
  data=predict(ajs_flex_gaus_las, val),
  reference=val$win
)

## Curvas ROC

#Completo

roc_flex_gaus <- pROC::roc(response = val$win,
                           predictor = predict(ajs_flex_gaus, newdata = val[,-1],
                                               type='prob')[,1])

roc_flex_uni <- pROC::roc(response = val$win,
                          predictor = predict(ajs_flex_uni, newdata = val[,-1],
                                              type='prob')[,1])

roc_flex_epa <- pROC::roc(response = val$win,
                          predictor = predict(ajs_flex_epa, newdata = val[,-1],
                                              type='prob')[,1])

pROC::ggroc(
  list(Parabolico=roc_flex_epa, Uniforme=roc_flex_uni,
       Gaussiano=roc_flex_gaus),
  legacy.axes=TRUE, size=1
)+
  ggplot2::labs(color='Kernel', x='1-Especificidade', y='Sensibilidade',
                title='Curvas ROC para ajustes via Bayes Ingênuo Flexível',
                subtitle = 'considerando modelo completo (sem diff_spd)')+
  ggplot2::theme_minimal()+
  ggplot2::theme(legend.position = 'top')

ggplot2::ggsave('img/17-roc-completo-flex-sem-diff.png')

#Arvore
roc_arv_flex_epa <- pROC::roc(response = val$win,
                              predictor = predict(ajs_flex_epa_arv, newdata = val[,-1],
                                                  type='prob')[,1])

roc_arv_flex_uni <- pROC::roc(response = val$win,
```

```
                              predictor = predict(ajs_flex_uni_arv, newdata = val[,-1],
                                                  type='prob')[,1])

roc_arv_flex_gaus <- pROC::roc(response = val$win,
                               predictor = predict(ajs_flex_gaus_arv, newdata = val[,-1],
                                                   type='prob')[,1])

pROC::ggroc(
  list(`Parabólico`=roc_arv_flex_epa, `Uniforme`=roc_arv_flex_uni,
       `Gaussiano`=roc_arv_flex_gaus),
  legacy.axes=TRUE, size=1
)+
  ggplot2::labs(color='Kernel', x='1-Especificidade', y='Sensibilidade',
                title='Curvas ROC para Bayes Ingênuo Flexível',
                subtitle = 'com variáveis via Árvore (sem diff_spd)')+
  ggplot2::theme_minimal()+
  ggplot2::theme(legend.position = 'top')

ggplot2::ggsave('img/18-roc-flex-arv-sem-diff.png')

# Lasso
roc_las_flex_epa <- pROC::roc(response = val$win,
                              predictor = predict(ajs_flex_epa_las, newdata = val[,-1],
                                                  type='prob')[,1])

roc_las_flex_uni <- pROC::roc(response = val$win,
                              predictor = predict(ajs_flex_uni_las, newdata = val[,-1],
                                                  type='prob')[,1])

roc_las_flex_gaus <- pROC::roc(response = val$win,
                               predictor = predict(ajs_flex_gaus_las, newdata = val[,-1],
                                                   type='prob')[,1])

pROC::ggroc(
  list(`Parabólico`=roc_las_flex_epa, `Uniforme`=roc_las_flex_uni,
       `Gaussiano`=roc_las_flex_gaus),
  legacy.axes=TRUE, size=1
)+
  ggplot2::labs(color='Kernel', x='1-Especificidade', y='Sensibilidade',
                title='Curvas ROC para Bayes Ingênuo Flexível',
```

```
                subtitle = 'com variáveis via Lasso (sem diff_spd)')+
  ggplot2::theme_minimal()+
  ggplot2::theme(legend.position = 'top')

ggplot2::ggsave('img/19-roc-flex-lasso-sem-diff.png')

## Apresentando Metricas

# Completo
tibble::tibble(
  Metrica=names(metricas_extract(conf_flex_epa, roc_flex_epa)),
  Parabolico=metricas_extract(conf_flex_epa, roc_flex_epa),
  Uniforme=metricas_extract(conf_flex_uni, roc_flex_uni),
  Gaussiano=metricas_extract(conf_flex_gaus, roc_flex_gaus)
) |>
  tidyr::pivot_longer(!Metrica) |>
  tidyr::pivot_wider(names_from=Metrica) |>
  dplyr::rename(Kernel=name) |>
  knitr::kable('latex', digits=4)

# Arvore
tibble::tibble(
  Metrica=names(metricas_extract(conf_arv_flex_epa, roc_arv_flex_epa)),
  `Parabolico` = metricas_extract(conf_arv_flex_epa, roc_arv_flex_epa),
  `Uniforme`=metricas_extract(conf_arv_flex_uni, roc_arv_flex_uni),
  `Gaussiano`=metricas_extract(conf_arv_flex_gaus, roc_arv_flex_gaus)
) |>
  tidyr::pivot_longer(!Metrica) |>
  tidyr::pivot_wider(names_from=Metrica) |>
  dplyr::rename(Ajuste=name) |>
  knitr::kable('latex', digits=4)

# Lasso
tibble::tibble(
  Metrica=names(metricas_extract(conf_las_flex_epa, roc_las_flex_epa)),
  `Parabolico` = metricas_extract(conf_las_flex_epa, roc_las_flex_epa),
  `Uniforme`=metricas_extract(conf_las_flex_uni, roc_las_flex_uni),
  `Gaussiano`=metricas_extract(conf_las_flex_gaus, roc_las_flex_gaus)
) |>
  tidyr::pivot_longer(!Metrica) |>
```

```
  tidyr::pivot_wider(names_from=Metrica) |>
  dplyr::rename(Ajuste=name) |>
  knitr::kable('latex', digits=4)

## Curva ROC melhores árvore, lasso e completo

pROC::ggroc(
  list(`Arvore`=roc_arv_flex_gaus, `Lasso`=roc_las_flex_gaus,
       `Completo`=roc_flex_gaus),
  legacy.axes=TRUE, size=1
)+
  ggplot2::labs(color='Seleção', x='1-Especificidade', y='Sensibilidade',
                title='Curvas ROC para Bayes Ingênuo Flexível',
                subtitle = 'com modelo completo e variáveis via Árvore e Lasso (sem diff_spd)')+
  ggplot2::theme_minimal()+
  ggplot2::theme(legend.position = 'top')

ggplot2::ggsave('img/20-roc-flex-all-sem-diff.png')

## Metricas melhores arvore e lasso

tibble::tibble(
  Metrica=names(metricas_extract(conf_las_flex_gaus, roc_las_flex_gaus)),
  `Completo` = metricas_extract(conf_flex_gaus, roc_flex_gaus),
  `Arvore`=metricas_extract(conf_arv_flex_gaus, roc_arv_flex_gaus),
  `Lasso`=metricas_extract(conf_las_flex_gaus, roc_las_flex_gaus)
) |>
  tidyr::pivot_longer(!Metrica) |>
  tidyr::pivot_wider(names_from=Metrica) |>
  dplyr::rename(Ajuste=name) |>
  knitr::kable('latex', digits=4)

#### VALIDACAO REG. LOGISTICA

## Ajustes

# Completo
ajs_log_full <- glm(win~., data=tre_val, family = binomial)

# Lasso
```

```
ajs_log_lasso <- glmnet::cv.glmnet(x=x_tre_val,y=y_tre_val,
                                   alpha=1, family='binomial')
ajs_log_lasso <- glmnet::glmnet(x=x_tre_val,y=y_tre_val, alpha=1,
                                lambda = ajs_log_lasso$lambda.1se,
                                family='binomial')

## Matrizes de Confusao

# Completo
pred_log_full <- predict(ajs_log_full, newdata=val[,-1], type='response')
pred_log_full <- as.factor(ifelse(pred_log_full>=0.5, 1, 0))

conf_log_full <- caret::confusionMatrix(
  data=pred_log_full,
  reference=val$win
)

# Lasso

pred_log_lasso <- predict(ajs_log_lasso, x_val,type='response')
pred_log_lasso <- as.factor(ifelse(pred_log_lasso>=0.5,1,0))

conf_log_lasso <- caret::confusionMatrix(
  data=pred_log_lasso,
  reference=val$win
)

## Curvas ROC

roc_log_full <- pROC::roc(response = val$win,
                          predictor = predict(ajs_log_full, newdata = val[,-1],
                                              type='response'))

roc_log_lasso <- pROC::roc(response = val$win,
                           predictor = as.vector(predict(ajs_log_lasso, newx = x_val,
                                                         type='response')))

pROC::ggroc(
  list(`Completo`=roc_log_full, `Lasso`=roc_log_lasso),
```

```
  legacy.axes=TRUE, size=1
)+
  ggplot2::labs(color='Seleção', x='1-Especificidade', y='Sensibilidade',
                title='Curvas ROC para Regressão Logística',
                subtitle = 'com modelo completo e variáveis via Lasso (sem diff_spd)')+
  ggplot2::theme_minimal()+
  ggplot2::theme(legend.position = 'top')

ggplot2::ggsave('img/21-roc-log-sem-diff.png')

## Apresentando Metricas

tibble::tibble(
  Metrica=names(metricas_extract(conf_log_full, roc_log_full)),
  `Completo` = metricas_extract(conf_log_full, roc_log_full),
  `Lasso`=metricas_extract(conf_log_lasso, roc_log_lasso)
) |>
  tidyr::pivot_longer(!Metrica) |>
  tidyr::pivot_wider(names_from=Metrica) |>
  dplyr::rename(Ajuste=name) |>
  knitr::kable('latex', digits=4)

#### COMPARANDO MELHORES MODELOS

## Ajustes

# Log
ajs_log <- glm(win~., data=tre, family = binomial)

# Arvore
ajs_arv <- rpart::rpart(win~., data=tre)

# BI Gaussiano
ajs_gaus <- naivebayes::naive_bayes(win~., data=tre[,c(1,3,5)],
                                    usekernel=FALSE)

# BI Flexivel
ajs_flex <- naivebayes::naive_bayes(win~., data=tre[,c(1,3,5)],
                                    usekernel=TRUE,
                                    kernel='gaussian')
```

```
## Matrizes de Confusao

# Log
pred_log <- predict(ajs_log, tes,type='response')
pred_log <- as.factor(ifelse(pred_log>=0.5,1,0))
conf_log <- caret::confusionMatrix(
  data=pred_log,
  reference=tes$win
)

# Arvore
conf_arv <- caret::confusionMatrix(
  data=predict(ajs_arv, tes[,-1], type='class'),
  reference=tes$win
)

# BI Gaussiano
conf_gaus <- caret::confusionMatrix(
  data=predict(ajs_gaus, tes[,-1]),
  reference=tes$win
)

# BI Flexível
conf_flex <- caret::confusionMatrix(
  data=predict(ajs_flex, tes[,-1]),
  reference=tes$win
)

## Curvas ROC

# Lasso
roc_log <- pROC::roc(response = tes$win,
                     predictor = predict(ajs_log, newdata = tes[,-1],
                                         type='response'))

# Arvore
roc_arv <- pROC::roc(response = tes$win,
                     predictor = predict(ajs_arv, tes[,-1])[,2])
```

```
# BI Gaussiano
roc_gaus <- pROC::roc(response = tes$win,
                      predictor = predict(ajs_gaus, tes[,-1], type='prob')[,1])

# BI Flexível
roc_flex <- pROC::roc(response = tes$win,
                      predictor = predict(ajs_flex, tes[,-1], type='prob')[,1])

pROC::ggroc(
  list(`Logistica`=roc_log, `Arvore`=roc_arv, `BI Gaussiano`=roc_gaus,
       `BI Flexível`=roc_flex),
  legacy.axes=TRUE, size=1
)+
  ggplot2::labs(color='Ajuste', x='1-Especificidade', y='Sensibilidade',
                title='Curvas ROC para Melhores Ajustes',
                subtitle = 'desconsiderando a variável diff_spd')+
  ggplot2::theme_minimal()+
  ggplot2::theme(legend.position = 'top')

ggplot2::ggsave('img/22-roc-all-sem-diff.png')

## Apresentando Metricas

tibble::tibble(
  Metrica=names(metricas_extract(conf_log, roc_log)),
  `Logística` = metricas_extract(conf_log, roc_log),
  `Árvore`=metricas_extract(conf_arv, roc_arv),
  `BI Gaussiano`=metricas_extract(conf_gaus, roc_gaus),
  `BI Flexível`=metricas_extract(conf_flex, roc_flex)
) |>
  tidyr::pivot_longer(!Metrica) |>
  tidyr::pivot_wider(names_from=Metrica) |>
  dplyr::rename(Ajuste=name) |>
  knitr::kable('latex', digits=4)
```