UNIVERSIDADE FEDERAL DO CEARÁ

CENTRO DE CIÊNCIAS

DEPARTAMENTO DE ESTATÍSTICA E MATEMÁTICA APLICADA

CURSO DE GRADUAÇÃO EM ESTATÍSTICA

FRANCISCO LUAN RODRIGUES DE SOUSA

MODELAGEM PREDITIVA DE PARTIDAS DE LEAGUE OF LEGENDS USANDO APRENDIZADO SUPERVISIONADO

FORTALEZA

2025

FRANCISCO LUAN RODRIGUES DE SOUSA

MODELAGEM PREDITIVA DE PARTIDAS DE LEAGUE OF LEGENDS USANDO APRENDIZADO SUPERVISIONADO

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Estatística do Centro de Ciências da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Estatística.

Orientador: Prof. Dr. Manoel Ferreira dos Santos Neto

FORTALEZA

2025

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Sistema de Bibliotecas
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

S696m Sousa, Francisco Luan Rodrigues de.
MODELAGEM PREDITIVA DE PARTIDAS DE LEAGUE OF LEGENDS USANDO APRENDIZADO SUPERVISIONADO / Francisco Luan Rodrigues de Sousa. – 2025.
69 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Centro de Ciências, Curso de Estatística, Fortaleza, 2025.
Orientação: Prof. Dr. Manoel Ferreira dos Santos Neto.

1. Aprendizado de Máquina. 2. Classificação. 3. Esportes Eletrônicos. 4. League of Legends. 5. Modelagem Preditiva. I. Título.

CDD 519.5

FRANCISCO LUAN RODRIGUES DE SOUSA

MODELAGEM PREDITIVA DE PARTIDAS DE LEAGUE OF LEGENDS USANDO APRENDIZADO SUPERVISIONADO

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Estatística do Centro de Ciências da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Estatística.

Aprovada em:

BANCA EXAMINADORA

Prof. Dr. Manoel Ferreira dos Santos Neto (Orientador)
Universidade Federal do Ceará (UFC)

Prof. Dr. Rafael Bráz Azevedo Farias
Universidade Federal do Ceará (UFC)

Prof. Dr. Ronald Targino Nojosa
Universidade Federal do Ceará (UFC)

AGRADECIMENTOS

A minha família, em especial a minha mãe (Luana), que tem sido, ao longo da minha vida, o modelo estatístico mais confiável. Sua proteção foi a constante que garantiu que os erros não me desviasse do caminho, e sua sabedoria foi o ajuste necessário para encontrar equilíbrio, garantindo que, mesmo nos momentos mais incertos, eu seguisse no caminho certo. Seu apoio foi a base que permitiu que eu chegasse até aqui.

Gostaria de expressar minha profunda gratidão ao meu orientador Prof. Dr. Manoel Santos Neto, cuja dedicação e generosidade são inspiradoras. Chegou no Departamento de Estatística e Matématica Aplicada (DEMA) conquistando não só a confiança dos discentes, mas também criando um ambiente de aprendizado estimulante. Sua disposição em ajudar e oferecer orientação vai além do esperado, sempre demonstrando interesse no nosso crescimento acadêmico e profissional. Sou imensamente grato por seu apoio, que foi fundamental para o sucesso deste trabalho.

Aos professores, minha gratidão por todo o conhecimento, apoio e dedicação, que foram fundamentais para minha trajetória acadêmica. Cada aula, orientação e conselho contribuiu não apenas para o meu crescimento acadêmico, mas também para a minha formação como profissional.

A banca examinadora composta pelo Prof. Dr. Manoel Ferreira dos Santos Neto, Prof. Dr Rafael Bráz Azevedo Farias e Prof Dr. Ronald Targino Nojosa. Quero deixar registrado meu enorme agradecimento pela participação de vocês na defesa do meu TCC. Saber que estavam ali, não apenas como avaliadores, mas como mestres que acompanharam minha jornada, trouxe confiança e significado a esse momento tão importante.

Por fim, mas tão importante quanto, às pessoas especiais, cujo apoio, risos e incentivo tornaram esta jornada mais leve e significativa, minha sincera gratidão (não preciso dizer nomes, vocês se reconhecerão).

"É possível que você nunca consiga, por exemplo, prever o que qualquer pessoa fará, mas pode dizer com precisão o que caberá, a um número médio de pessoas, fazer. Os indivíduos variam, mas as porcentagens permanecem constantes. Assim falam os estatísticos."

(SHERLOCK HOLMES - O signo dos quatro)

RESUMO

Este estudo investiga a aplicação de modelos de aprendizado supervisionado para prever o resultado de partidas de *League of Legends*, um dos esportes eletrônicos mais populares do mundo. Foram analisadas diferentes abordagens, incluindo *K-Nearest Neighbors* (KNN), Regressão Logística, *Gradient Boosting* e Árvores de Decisão, utilizando um conjunto de dados extraído do Oracle's Elixir. A metodologia envolveu a seleção de variáveis, a eliminação de metadados irrelevantes e a validação dos modelos por meio de métricas como a área sob a curva ROC (AUC-ROC) e acurácia. Os resultados indicam que a Regressão Logística e o KNN obtiveram o melhor desempenho, com AUC de até 0,930, demonstrando alta capacidade preditiva. Além da precisão dos modelos, o estudo destaca a importância de variáveis estratégicas, como diferenças de recursos no início da partida e escolhas de campeões, para a tomada de decisões em contextos competitivos.

Palavras-chave: Aprendizado de Máquina. Classificação. Esportes Eletrônicos. League of Legends. Modelagem Preditiva.

ABSTRACT

This study explores the application of supervised learning models to predict the outcomes of League of Legends matches, one of the most popular esports worldwide. Various approaches were analyzed, including K-Nearest Neighbors (KNN), Logistic Regression, Gradient Boosting, and Decision Trees, using a dataset extracted from Oracle's Elixir. The methodology involved careful feature selection, removal of irrelevant metadata, and model validation through metrics such as AUC-ROC and accuracy. Results indicate that Logistic Regression and KNN achieved the best performance, with an AUC of up to 0,930, demonstrating strong predictive capability. Beyond model accuracy, the study highlights the strategic importance of early-game resource differences and champion selection in decision-making within competitive settings.

Keywords: Machine Learning. Classification. E-Sports. League of Legends. Predictive Modeling.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE QUADROS

SUMÁRIO

1 INTRODUÇÃO

A ideia mais simples e básica de modelagem estatística, que mais tarde influenciaria o aprendizado de máquina, surgiu em 1875 com Francis Galton, primo de Charles Darwin, quando ele buscava uma maneira de expressar a correlação linear entre os tamanhos de ervilhas cultivadas por pais e filhos (STANTON, 2001). Esse estudo foi uma das primeiras tentativas de modelar matematicamente padrões em dados biológicos. Décadas depois, a evolução dessas ideias, aliada aos avanços estatísticos, levou ao desenvolvimento de métodos que se tornaram a base de muitos modelos atuais de aprendizado de máquina. Estudiosos da área de inteligência artificial, décadas depois, deram um nome especial a esses métodos: *aprendizado de máquina supervisionado*.

De acordo com Hastie *et al.* (2009), o aprendizado é chamado de "supervisionado" porque, inicialmente, um humano ou sistema automatizado fornece um conjunto de treinamento S, composto por vetores de entrada n-dimensionais de comprimento uniforme $x_i \in \mathbb{R}^n$, associados a rótulos com valores reais $y_i \in \mathbb{R}$. Em outras palavras, as variáveis de entrada, também conhecidas como preditores ou variáveis independentes, são usadas para prever as saídas, chamadas de respostas ou variáveis dependentes. Após essa fase inicial de orientação, a máquina opera de forma autônoma. O objetivo é encontrar uma função $f_s : \mathbb{R}^n \to \mathbb{R}$, escolhida dentro de um conjunto de hipóteses $\mathscr{H}$, que seja capaz de prever os rótulos com precisão. Existem várias formas de definir o que é "melhor" para essa função, e algumas delas serão discutidas posteriormente. Uma vez definida f_s, ela pode ser usada para prever os rótulos de novos dados, com $y_{\text{pred}} = f_s(x_{\text{new}})$. Além disso, as entradas podem ser qualitativas, quantitativas ou ambas, enquanto as saídas podem ser qualitativas (categóricas, discretas ou fatores) ou quantitativas. Quando se trata de prever variáveis de saída qualitativas, utiliza-se o termo *classificação*.

Os métodos de aprendizado supervisionado têm suas raízes em problemas resolvidos há mais de dois séculos, com a aplicação inicial de técnicas estatísticas para modelar relações entre variáveis. Essas ideias se consolidaram como uma base fundamental para as técnicas de aprendizado de máquina, que continuam a ser amplamente utilizadas até hoje e são catalisadoras do progresso da ciência estatística e da inteligência artificial.

Com a geração automática de grandes volumes de dados, a internet das coisas e a redução dos custos de envio e armazenamento de informações, novas abordagens metodológicas puderam ser desenvolvidas na era do *Big Data* (GALEANO, 2019). Além disso, Izbicki (2020)

cita que métodos tradicionais apresentam limitações ao lidar com bancos de dados em que o número de covariáveis (p) excede o número de observações (n), um cenário conhecido como $p \gg n$, frequentemente encontrado em aplicações modernas. Isso ocorre porque tais métodos dependem de suposições que não se sustentam em espaços de alta dimensionalidade, como a invertibilidade de matrizes ou a ausência de multicolinearidade. De maneira similar, é comum encontrar contextos em que cada observação corresponde a estruturas de alta dimensionalidade, como imagens ou documentos textuais. Esses objetos complexos requerem metodologias mais elaboradas, capazes de modelar adequadamente as relações intrínsecas nesses espaços de dados de alta dimensionalidade e complexidade.

Ter acesso a grandes volumes de dados não é suficiente; é necessário analisá-los, extrair informações relevantes e aplicá-las de forma significativa. Em diversos cenários, dependendo do tipo de dados disponíveis e do objetivo da análise, pode ser desafiador identificar e compreender com profundidade o processo que dá origem aos dados. Contudo, se for possível identificar e explorar determinados padrões nos dados, é factível construir uma aproximação eficaz de alguma parte desse processo. Dessa forma, com essa aproximação, torna-se possível alcançar uma compreensão mais ampla do mecanismo subjacente que origina os dados e realizar previsões. Fundamentalmente, a identificação desses padrões é a essência do campo de aprendizado de máquina (ALPAYDIN, 2020).

A teoria do aprendizado estatístico é uma área da estatística que teve origem no campo do aprendizado de máquina (JAMES *et al.*, 2013). Assim como no aprendizado de máquina, o conceito de aprendizado, nesse contexto, refere-se à habilidade de reconhecer e interpretar padrões e tendências em grandes conjuntos de dados. Os problemas abordados pela teoria do aprendizado estatístico geralmente se dividem em duas categorias principais: aprendizado supervisionado e aprendizado não supervisionado (HASTIE *et al.*, 2009).

Embora muitos algoritmos de aprendizado de máquina, principalmente modelos de aprendizado supervisionado, tenham alcançado um estágio de maturidade teórica, com seus pontos fortes e limitações bem compreendidos, a investigação sobre sua flexibilidade e aplicabilidade a conjuntos de dados reais é algo relativamente recente. Apesar da existência de ferramentas de software de código aberto para aprendizado de máquina, como Orange (RATRA; PREETI, 2020) e Weka (WITTEN; FRANK, 1999), a implementação e a avaliação desses algoritmos em dados do mundo real ainda demandam um trabalho significativo. Esse processo envolve codificação personalizada, ajustes de hiperparâmetros específicos para cada conjunto de

dados, escolha criteriosa de variáveis e tratamento adequado de dados, especialmente no caso de variáveis categóricas.

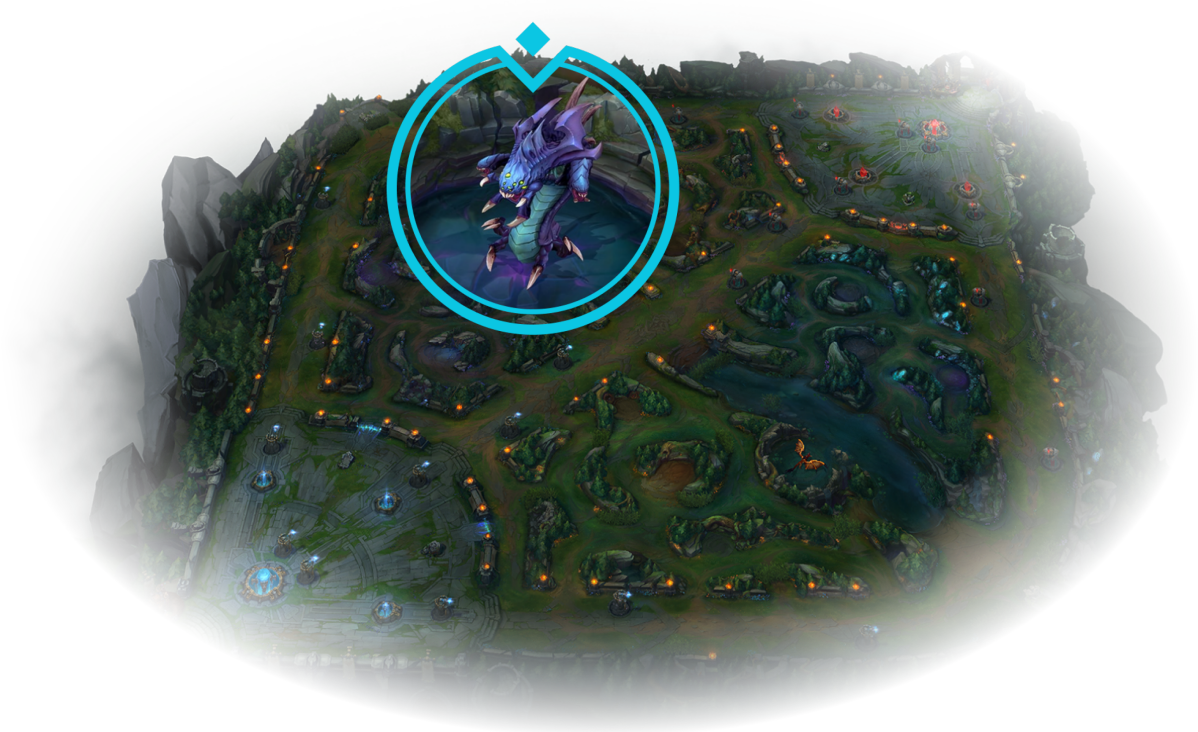
Este trabalho tem como objetivo explorar e contextualizar o aprendizado supervisionado sob uma perspectiva estatística, introduzindo as principais ideias por trás dos algoritmos que moldam a inteligência artificial. A aplicação prática será voltada para o universo dos esportes eletrônicos, especificamente em partidas de *League of Legends* (LoL), um dos jogos mais populares no cenário competitivo de eSports. Os eSports, definidos como competições organizadas de videogames multiplayer entre jogadores profissionais, têm apresentado crescimento expressivo nos últimos anos, impulsionados pela popularização das plataformas de streaming online.

Jogos como *Dota 2*, LoL, *Overwatch* e *Counter-Strike: Global Offensive* destacam-se entre as competições mais populares do setor. Em 2013, aproximadamente 71,5 milhões de pessoas assistiram a jogos competitivos de eSports, incluindo 32 milhões que acompanharam o Campeonato Mundial da Terceira Temporada de LoL (WARR, 2014). Essa audiência superou eventos tradicionais, como o jogo final da NBA no mesmo ano (SEGAL, 2014). Em 2017, a audiência global atingiu cerca de 385 milhões, dividida entre 191 milhões de espectadores regulares e o restante composto por espectadores ocasionais (HATTENSTONE, 2017). Paralelamente, a economia dos eSports apresentou um crescimento significativo, com projeções de atingir US$ 696 milhões em 2018, refletindo um aumento anual de 41,3% (WARMAN, 2017). Nos anos mais recentes, os eSports continuaram a expandir-se, consolidando-se como um setor de destaque na economia global. Em 2023, a audiência de eSports alcançou 2,76 bilhões de horas assistidas, representando um aumento de 75% em relação a 2020, com LoL liderando as transmissões, acumulando 1,6 bilhão de horas assistidas (Máquina do Esporte, 2023). Em termos econômicos, o mercado global de eSports foi avaliado em US$ 1,76 bilhão em 2023, com projeções de crescimento para US$ 2,11 bilhões em 2024 e atingindo US$ 5,27 bilhões até 2029, apresentando uma taxa de crescimento anual composta (CAGR) de 20,05% (INTELLIGENCE, 2023). No Brasil, a Pesquisa Game Brasil de 2023 revelou que 63,8% dos gamers brasileiros acompanham ou assistem a eSports regularmente, e a receita no mercado nacional foi estimada em US$ 16,03 milhões (Meio & Mensagem, 2023; NEWS, 2023). Esses dados evidenciam a crescente relevância dos eSports no panorama atual, tanto em termos de audiência quanto de impacto econômico, reforçando sua importância como objeto de estudo e análise no contexto contemporâneo.

Lançado em 2009 pela Riot Games, LoL é um jogo de estratégia do gênero Multi-

player Online Battle Arena (MOBA), no qual duas equipes de cinco jogadores competem para destruir a base adversária. Cada jogador escolhe um campeão com habilidades específicas, e a dinâmica de jogo envolve tomadas de decisão estratégicas, trabalho em equipe e domínio técnico. O mapa do jogo, que é uma parte essencial da estratégia, está ilustrado na Figura 1. Neste trabalho, os dados das partidas incluem métricas como número de abates (um abate é o evento de reduzir a saúde de um campeão inimigo a zero e fazê-lo entrar no estado de morte com sucesso. Ativar a ressurreição não conta como uma morte), assistências, ouro acumulado, controle de mapa, entre outros indicadores extraídos do desempenho das equipes durante o jogo. O objetivo deste trabalho é resolver um problema de classificação binária, onde buscamos prever qual equipe será a vencedora com base nos atributos disponíveis. Esse cenário é relevante não apenas para a análise de desempenho em competições, mas também para o desenvolvimento de sistemas de recomendação, estratégias de treinamento dentro das organizações e até mesmo para plataformas de apostas.

Figura 1 – Ilustração, mapa de Summoner's Rift.



Fonte: Riot Games.

Além desta introdução, esta monografia está dividida em capítulos que abordam os principais aspectos do estudo. No Capítulo 2, serão abordados os conceitos de aprendizado de máquina supervisionado e uma análise dos algoritmos empregados neste estudo, enfatizando os

aspectos teóricos que sustentam sua aplicação. O Capítulo 3 apresentará o contexto dos dados utilizados, detalhando os processos de coleta, pré-processamento e a implementação prática dos modelos estudados. O Capítulo 4 será dedicado à apresentação dos resultados obtidos e à avaliação de desempenho dos algoritmos, com base em métricas específicas. Por fim, o Capítulo 5 trará as considerações finais, sintetizando as contribuições deste trabalho, suas limitações e as perspectivas para estudos futuros. Assim, este estudo pretende discutir as possibilidades e limites do aprendizado supervisionado, unindo teoria e prática em um panorama abrangente e aplicável.

1.1 Ferramentas e Tecnologias Utilizadas

Para o desenvolvimento do modelo de classificação binária, foram utilizadas ferramentas e tecnologias amplamente reconhecidas na comunidade de ciência de dados e aprendizado de máquina, garantindo reprodutibilidade, eficiência e rigor metodológico. A linguagem de programação escolhida foi o **Python 3.11**, devido à sua versatilidade, ampla gama de bibliotecas especializadas e suporte da comunidade científica. O ambiente de desenvolvimento utilizado foi o Jupyter Notebook, que oferece um ambiente interativo para prototipagem rápida e documentação integrada. Para o versionamento do código e documentação, foi utilizado o GitHub (<https://encurtador.com.br/mDuWu>), garantindo rastreabilidade das alterações.

O Quadro 1 resume as principais etapas, ferramentas e técnicas utilizadas no processo de manipulação, pré-processamento, modelagem e validação dos dados. As bibliotecas **Pandas** e **NumPy** foram empregadas para a manipulação e transformação dos dados, enquanto o **Scikit-learn** foi utilizado para pré-processamento, engenharia de features e modelagem. Destacam-se o uso de `Pipeline` e `ColumnTransformer` para integração das etapas, o `RobustScaler` para escalonamento robusto de features numéricas e o `TargetEncoder` para codificação de variáveis de alta cardinalidade. Para a modelagem, foram implementados os algoritmos Gradient Boosting Machine (`GradientBoostingClassifier`), Regressão Logística (`LogisticRegression`), K-nearest neighbors(`KNeighborsClassifier`) e Árvore de Decisão (`DecisionTreeClassifier`). A validação dos modelos foi realizada com técnicas como `GroupShuffleSplit` e `GroupKFold`, garantindo a independência dos dados. Por fim, a visualização dos dados foi realizada com a biblioteca **Matplotlib** e **Plotnine**.

Quadro 1 – Resumo das etapas, ferramentas e técnicas utilizadas no processo de manipulação, pré-processamento, modelagem e validação dos dados.

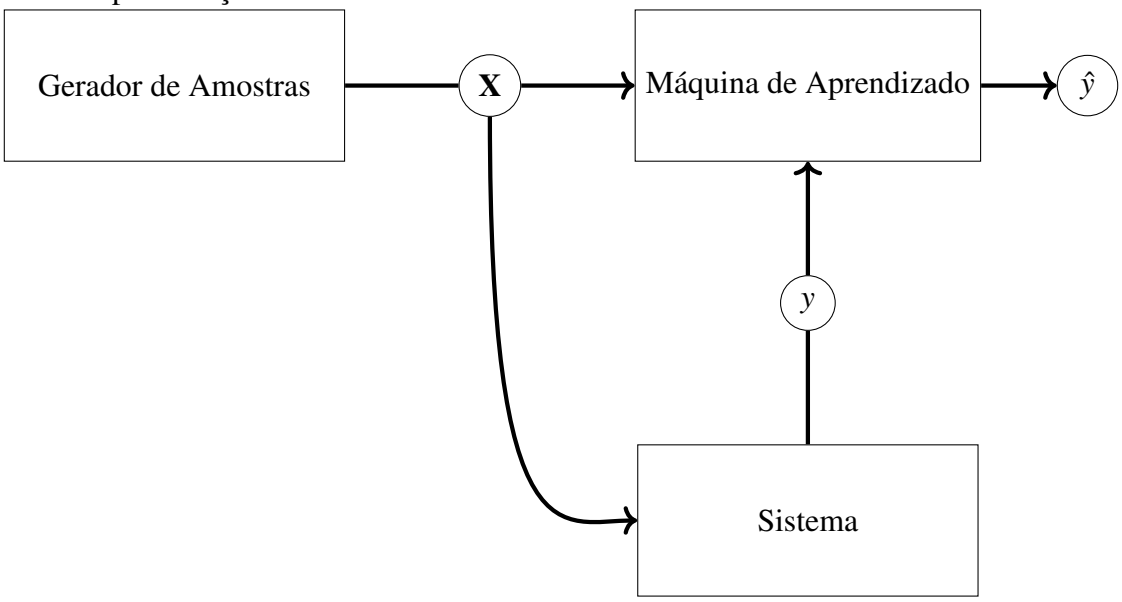
Categoria	Detalhes
Manipulação e Transformação	• Bibliotecas: **Pandas** e **NumPy**. • Ações: Carregamento, filtragem e criação de variáveis de diferença.
Pré-processamento e Engenharia de Features	• Biblioteca: **Scikit-learn** e **Category Encoders**. • Ferramentas: `ColumnTransformer` e `Pipeline`. • Técnicas: Escalonamento e codificação. • `RobustScaler`: Escalonamento robusto de features numéricas. • `OneHotEncoder`: Codificação de variáveis categóricas (ex: `side`). • `TargetEncoder`: Codificação de variáveis de alta cardinalidade (ex: `picks` e `bans`).
Modelagem	• Biblioteca: **Scikit-learn**. • Algoritmos: `GradientBoostingClassifier`, `LogisticRegression`, `KNeighborsClassifier`, `DecisionTreeClassifier`.
Validação	• Técnicas: `GroupShuffleSplit` e `GroupKFold`. • Métricas: `roc_auc_score`, `accuracy_score`, `f1_score`.
Visualização	• Biblioteca: **Matplotlib** e **Plotnine**. • Gráficos: Boxplots comparativos antes e depois do escalonamento.
Justificativas de Escolha	• **Scikit-learn**: Eficiência em pipelines reprodutíveis. • `TargetEncoder`: Evita explosão dimensional em variáveis de alta cardinalidade. • `RobustScaler`: Preserva integridade dos dados sem distorção. • `Pipeline` e `ColumnTransformer`: Evitam *data leakage*. • `GroupKFold`: Garante independência entre treino e teste (sem mistura de partidas).

Fonte: Próprio autor.

2 APRENDIZADO SUPERVISIONADO

O aprendizado de máquina supervisionado pode ser formalizado como o processo de estimar uma função $f : X \to Y$, que modela a relação entre um espaço de entrada $X \subseteq \mathbb{R}^n$ e um espaço de saída $Y \subseteq \mathbb{R}$. Para isso, utiliza-se um conjunto de treinamento composto por m pares de dados observados (x_i, y_i), onde $x_i \in X$ e $y_i \in Y$. Esse processo é ilustrado na Figura 2, baseada na técnica de "aprendizado por observações" proposta por Vapnik (1995). A abordagem de Vapnik se baseia em três componentes principais: o gerador de amostras, o sistema e a máquina de aprendizado.

Figura 2 – Uma Máquina de Aprendizagem usando observações do Sistema para formar uma aproximação de sua saída.



Fonte: Próprio autor.

O gerador de observações é responsável por produzir vetores de entrada x, que são independentes e identicamente distribuídos (IID) segundo uma distribuição de probabilidade $p(x)$. Para cada vetor x, o sistema mapeia x em uma saída y de acordo com a função densidade de probabilidade condicional $p(y \mid x)$. Dessa forma, os dados observados (x, y) são amostras da distribuição conjunta:

$$p(x, y) = p(x)\, p(y \mid x).$$

No cenário estatístico típico, assume-se que o aprendizado ocorre em um contexto observacional, onde a distribuição $p(x, y)$ é fixa e desconhecida. O objetivo do aprendizado é aproximar a função subjacente que descreve os dados. Esse cenário difere de experimentos

planejados, nos quais x é controlado de forma determinística. No contexto descrito, o gerador de observações produz vetores $x \in \mathbb{R}^n$, enquanto o sistema retorna $y \in \mathbb{R}$ de acordo com $p(y \mid x)$. Assim, pares (x, y) são formados, onde $x \sim p(x)$ e $y \sim p(y \mid x)$. O objetivo da máquina de aprendizado é utilizar n pares observados para construir uma função estimada $\hat{f}$, pertencente a um conjunto $\mathscr{F}$ de funções possíveis, tal que:

$$\hat{f}(x;\alpha) = \arg\min_{f \in \mathscr{F}} \frac{1}{m} \sum_{i=1}^{m} L(f(x_i,\alpha), y_i),$$

onde α é um vetor de parâmetros ajustáveis e $L(f(x), y)$ a função de perda. Após o aprendizado, os parâmetros α permanecem fixos, e a notação é frequentemente simplificada para $f(x)$. O sistema gera a saída y de forma determinística ou estocástica, representada pela função densidade de probabilidade condicional $p(y \mid x)$. Em muitos casos práticos, o sistema é modelado como:

$$y = t(x) + \varepsilon,$$

onde $t(x)$ é a função verdadeira (desconhecida) e ε representa uma fonte de variação, assumida como uma variável aleatória com média zero. A função alvo $f^*(x)$ é definida como a função que minimiza a expectativa do erro em relação à distribuição conjunta $p(x, y)$:

$$f^*(x) = \arg\min_{f} \mathbb{E}_{(x,y)\sim p(x,y)}[L(f(x), y)],$$

onde a função como o erro quadrático médio para regressão ou a entropia cruzada para classificação.

Portanto, o processo de aprendizado de máquina supervisionado consiste em estimar uma função a partir de um conjunto de exemplos rotulados, de modo que essa função maximize a capacidade do modelo de generalizar. Em outras palavras, o modelo deve ser capaz de produzir previsões confiáveis e consistentes quando aplicado a novos dados provenientes da mesma distribuição subjacente dos dados de treinamento. Em modelos de aprendizado de máquina supervisionado, problemas com uma variável resposta (saída) quantitativa são chamados de problemas de **regressão**, enquanto aqueles que envolvem uma variável resposta (saída) qualitativa são frequentemente chamados de problemas de **classificação**.

A regressão é uma técnica fundamental em aprendizado supervisionado, utilizada para modelar a relação entre uma variável dependente y e uma ou mais variáveis independentes x. O objetivo principal da regressão é estimar uma função $f(x, y)$ que mapeia as variáveis de entrada x para a variável de saída y, minimizando o erro entre as previsões do modelo e os valores observados.

Enquanto isso, a classificação é uma técnica amplamente utilizada em aprendizado de máquina, onde o objetivo é atribuir rótulos a novos exemplos com base em um conjunto de dados rotulados. O problema clássico de classificação pode ser descrito como um caso específico de um problema de classificação geral, que se baseia em um modelo de aprendizado restrito. Nesse contexto, para estimar as densidades condicionais de cada classe, $p(x|y=0)$ e $p(x|y=1)$, adotamos um modelo paramétrico. Nesse modelo, assumimos uma distribuição de probabilidade específica para cada classe (como a distribuição normal, por exemplo) e utilizamos o método de máxima verossimilhança (MV) para estimar os parâmetros desconhecidos dessas distribuições a partir dos dados de treinamento. Essa abordagem clássica garante que os parâmetros escolhidos maximizem a probabilidade dos dados observados em cada classe, seguindo o princípio estatístico consolidado da máxima verossimilhança.

2.1 Poder de Generalização

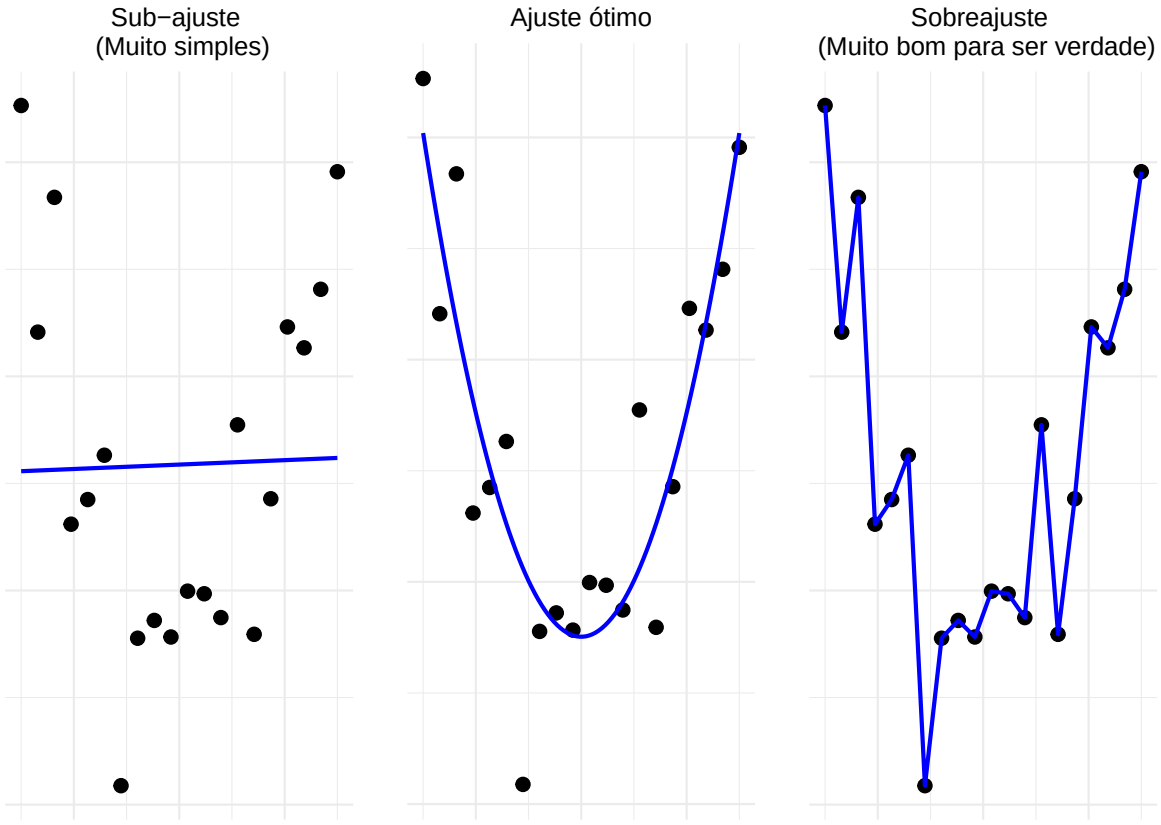
Conforme destacado por Müller *et al.* (2001), o processo de aprendizado supervisionado tem como objetivo encontrar uma função que maximize a capacidade de generalização do modelo. Isso significa que o modelo deve ser capaz de prever com precisão não apenas os dados de treinamento, mas também novos dados. No entanto, se o conjunto de treinamento contiver ruídos ou *outliers* (pontos que se distanciam significativamente dos demais), o modelo pode se ajustar excessivamente a esses pontos, resultando no fenômeno conhecido como **superajuste** (ou *sobreajuste*).

O superajuste ocorre quando o modelo se torna excessivamente específico, ajustando-se a detalhes e flutuações presentes no conjunto de treinamento que não são representativos do comportamento geral dos dados. Como consequência, o modelo pode apresentar um desempenho excelente nos dados de treinamento, mas falhar ao ser aplicado a dados novos. Por outro lado, o sub-ajuste ocorre quando o modelo é incapaz de capturar adequadamente os padrões presentes nos dados de treinamento, resultando em um desempenho ruim tanto nos dados de treinamento quanto nos dados novos. Em ambos os casos, a capacidade de generalização do modelo é prejudicada.

Como ilustrado na Figura 3, o desafio está em encontrar um modelo que seja suficientemente complexo para capturar os padrões importantes, mas não tão específico a ponto de se ajustar excessivamente aos detalhes do conjunto de treinamento.

Além disso, para controlar a complexidade do modelo e melhorar sua capacidade

Figura 3 – Ilustração, em modelos de regressão, de sub-ajuste e superajuste (sobreajuste).



Fonte: Próprio autor.

de generalização, utiliza-se o princípio de **minimização do risco estrutural (MRE)**. O MRE propõe que o modelo deve minimizar o erro em relação aos dados de treinamento e, ao mesmo tempo, considerar o risco de erro para dados não vistos. O risco estrutural é uma medida da probabilidade de o modelo cometer um erro ao ser exposto a novos exemplos. O equilíbrio entre minimizar o erro de treinamento e manter o risco de erro para dados novos baixo é fundamental para garantir uma boa generalização (VAPNIK, 1995).

Por fim, como observam Müller *et al.* (2001), a escolha do modelo e a aplicação de técnicas de regularização são fundamentais para a construção de um classificador que tenha um bom desempenho não apenas nos dados de treinamento, mas também em dados desconhecidos. Restringir a complexidade do modelo e aplicar técnicas adequadas de regularização são estratégias eficazes para maximizar a capacidade de generalização, evitando tanto o superajuste quanto o sub-ajuste.

2.2 Risco Esperado

A função de risco é um conceito central na teoria estatística e, consequentemente, em modelos supervisionados, sendo utilizada para avaliar a qualidade das aproximações feitas por um modelo. Ela se fundamenta no conceito de **função de perda**, que mede a discrepância entre a saída verdadeira y e a saída prevista pelo modelo $f(x,\theta)$, onde x representa as entradas e θ os parâmetros do modelo. A função de perda, denotada por $L(x, f(y,\theta))$, assume valores não negativos, de modo que valores maiores indicam pior desempenho do modelo. A **função de risco**, por sua vez, é o valor esperado da perda em relação à distribuição conjunta das variáveis x e y, dada por:

$$R(\theta) = \int L(y, f(x,\theta))\, p(x,y)\, dx\, dy, \tag{2.1}$$

onde $p(x,y)$ é a distribuição conjunta das variáveis de entrada x e saída y. Essa integral representa o desempenho médio esperado do modelo em relação à verdadeira distribuição dos dados. Na prática, $p(x,y)$ é desconhecida, e o aprendizado ocorre a partir de um conjunto finito de dados $\{(x_i,y_i)\}_{i=1}^{n}$. Assim, a minimização direta da função de risco verdadeira $R(\theta)$ é inviável. Em seu lugar, utiliza-se a **função de risco empírica**:

$$R_{\text{emp}}(\theta) = \frac{1}{n}\sum_{i=1}^{n} L(x_i, f(y_i,\theta)),$$

que é uma aproximação de $R(\theta)$ baseada nos dados disponíveis. No entanto, devido à natureza limitada dos dados, a solução obtida é uma estimativa $f(x,\hat{\theta})$, e não a solução ideal $f(x,\theta^*)$.

Esse cenário torna o problema de aprendizado mal posto, pois:

- Não é possível garantir que $f(x,\hat{\theta})$ seja uma boa aproximação de $f(x,\theta^*)$ com dados finitos.
- Múltiplas soluções podem atender aos dados de treinamento sem generalizar bem para dados novos.

Para resolver o problema de aprendizado, introduz-se conhecimento a priori por meio de restrições no espaço de funções $f(x,\theta)$ ou de regularização. O critério para selecionar a melhor aproximação $f(x,\hat{\theta})$ é formalizado por um princípio indutivo, como:

- **Máxima Verossimilhança**: Maximizar a probabilidade dos dados observados.
- **Minimização do erro quadrático**: Utilizar $L(y, f(x,\theta)) = (y - f(x,\theta))^2$.

Além disso, minimizar a função de risco é essencial para tarefas como:

- **Regressão**: Onde $L(y, f(x,\theta)) = (y - f(x,\theta))^2$.

- **Classificação**: Onde a perda pode ser definida como a função logarítmica ou perda de articulação (*hinge loss*).

Por fim, a função de risco destaca a complexidade de problemas como a estimativa de densidade $p(x,y)$, que é considerada o tipo mais geral e desafiador de problema de aprendizado. Técnicas como validação cruzada, regularização e seleção de modelos ajudam a mitigar os desafios associados à generalização, garantindo um equilíbrio entre ajuste aos dados de treinamento e desempenho em novos conjuntos de dados.

2.3 Algoritmos de Aprendizado Supervisionado

2.3.1 Algoritmo KNN

O método mais popular entre os algoritmos baseados em instância é o KNN, uma instância é um exemplo ou um registro individual do conjunto de dados. Por exemplo, se estamos trabalhando com um conjunto de dados de casas, cada casa (com suas características como tamanho, número de quartos e preço) é uma instância. Trata-se de um algoritmo de aprendizado supervisionado, capaz de lidar tanto com problemas de classificação quanto de regressão (COVER; HART, 1967). A ideia central do KNN é inspirada no mundo real, considerando como as pessoas tendem a formar amizades com outras que compartilham interesses ou características em comum. Esse algoritmo fornece uma maneira intuitiva de prever uma variável resposta quantitativa, utilizando as respostas de outras observações semelhantes para estimar valores de novos exemplos.

Os algoritmos de vizinhos mais próximos estão entre os algoritmos de aprendizado supervisionado mais intuitivos e têm sido amplamente estudados no campo do reconhecimento de padrões ao longo do último século, originalmente proposto por na década de 1950 (FIX; JR., 1951). Esses algoritmos mantêm relevância devido à sua flexibilidade e adaptabilidade a problemas complexos, como classificação de imagens, recomendação personalizada e análise de dados genômicos (GOU *et al.*, 2019; ZHANG, 2022). Embora modelos mais sofisticados, como redes neurais profundas, tenham ganhado destaque, o KNN persiste como uma ferramenta valiosa em cenários onde a interpretabilidade e a simplicidade são críticas (GARCIA *et al.*, 2020), além de ser frequentemente integrado a pipelines de aprendizado de máquina modernos para aprimorar resultados (BELLEM, 2023).

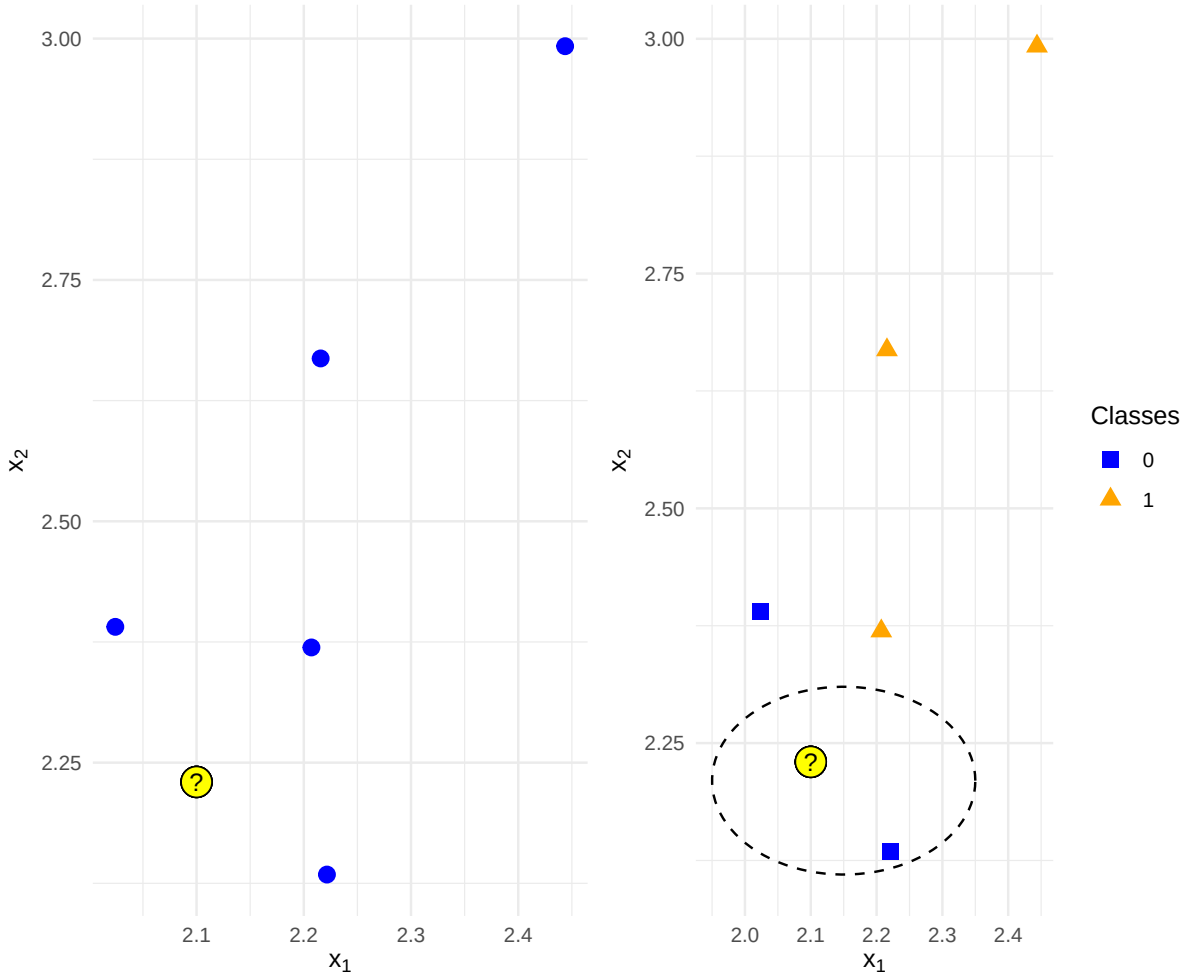
A lógica do algoritmo é simples. Em problemas de classificação, novos dados são

atribuídos à classe predominante entre seus vizinhos mais próximos. Da mesma forma, em problemas de regressão, os novos dados têm seus valores estimados com base nos valores dos vizinhos mais próximos. Em essência, o KNN compartilha o conceito de algoritmos de agrupamento (clustering), pois utiliza a distância como critério de similaridade (HASTIE *et al.*, 2009). Ele é frequentemente chamado de modelo não paramétrico e preguiçoso, uma vez que não realiza uma etapa de treinamento prévia e não faz suposições sobre os parâmetros do modelo.

Embora o KNN seja um aproximador de função universal sob certas condições, seu conceito subjacente é relativamente simples (DUDA *et al.*, 2001). Ele atua como um aproximador universal porque, ao usar a média ou votação dos vizinhos mais próximos, consegue imitar o comportamento de funções complexas localmente. Quanto mais dados disponíveis, mais vizinhos ele usa para reconstruir a função original. Ele simplesmente armazena os exemplos de treinamento rotulados, $x[i], y[i] \in D$ ($|D| = n$), sem realizar processamento significativo até a fase de previsão. Por essa razão, o KNN é classificado como um algoritmo de aprendizado preguiçoso, pois a computação real ocorre apenas no momento da inferência.

Para fazer uma previsão (rótulo de classe ou valor contínuo), o KNN encontra os k vizinhos mais próximos de um ponto de consulta e calcula o rótulo da classe (para classificação) ou o valor contínuo (para regressão) com base nesses k pontos mais próximos. Em vez de aproximar a função alvo $f(x) = y$ globalmente, durante cada previsão, o KNN aproxima a função alvo localmente. Na prática, é mais fácil aprender a aproximar uma função localmente do que globalmente (BISHOP, 2006). Na Figura 4 é apresentada uma ilustração do funcionamento do algoritmo KNN.

Figura 4 – Ilustração do algoritmo KNN em duas dimensões (atributos x_1 e x_2). No subpainel à esquerda, os exemplos de treinamento são mostrados como pontos azuis, e um ponto de consulta que queremos classificar é mostrado como um ponto de interrogação. No subpainel à direita, os rótulos das classes estão indicados, e a linha tracejada representa o vizinho mais próximo do ponto de consulta, assumindo uma métrica de distância Euclidiana. O rótulo de classe previsto é o rótulo de classe do ponto de dados mais próximo no conjunto de treinamento (aqui: classe 0).



Fonte: Próprio autor.

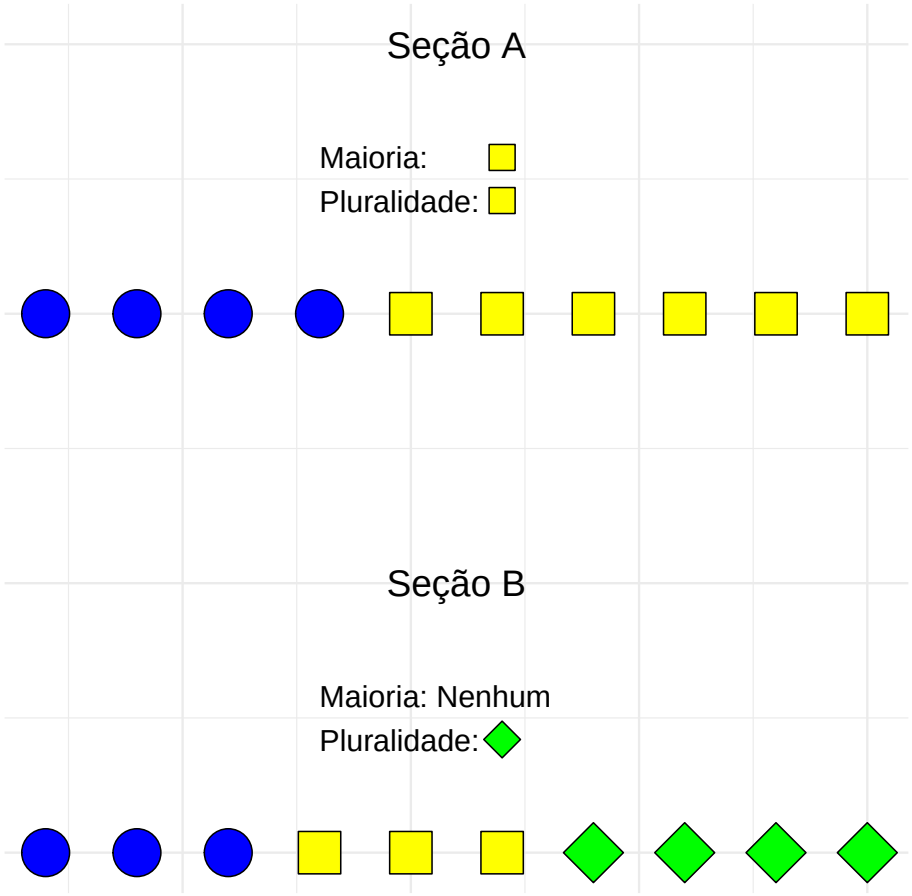
2.3.1.1 Classificação

No contexto de classificação, o algoritmo KNN realiza a previsão do rótulo da classe alvo selecionando a classe que aparece com maior frequência entre os k exemplos mais próximos de um ponto de consulta nos dados de treinamento. Isso significa que o rótulo da classe é determinado como a **moda** dos rótulos desses k vizinhos ou como o resultado de uma votação por pluralidade. É importante destacar que, na literatura, essa abordagem é muitas vezes chamada de votação por maioria. Contudo, esse termo pode ser inadequado, já que tradicionalmente se refere a um critério em que mais de 50% dos votos são necessários para

tomar uma decisão (RASCHKA, 2020).

Em problemas de classificação binária (com apenas duas classes), sempre haverá uma maioria ou um empate, de modo que a votação por maioria coincide com a votação por pluralidade. Entretanto, em problemas com múltiplas classes, não é preciso atingir uma maioria absoluta para realizar a previsão com o KNN. Como na Figura 5, por exemplo, em um cenário com três classes, uma frequência superior a $\frac{1}{3}$ (cerca de 33,3%) pode ser suficiente para definir o rótulo da classe (RASCHKA, 2020). Essa característica do KNN o torna particularmente útil em cenários onde as classes estão desbalanceadas ou quando a distribuição das classes é complexa (JAMES *et al.*, 2013).

Figura 5 – Ilustração de pluralidade e votação majoritária.



Fonte: Próprio autor.

Lembre-se de que a regra de predição de *Nearest Neighbor (NN)* (lembrando que definimos NN como o caso especial de KNN com $k = 1$) é a mesma tanto para classificação quanto para regressão. No entanto, no KNN, temos dois algoritmos de predição distintos:

- *Votação por pluralidade*: entre os k vizinhos mais próximos para classificação.

- *Média das variáveis de destino contínuas*: dos k vizinhos mais próximos para regressão.

De maneira mais formal, suponha que temos uma função de destino $f(x) = y$ que atribui um rótulo de classe $y \in \{1,\ldots,t\}$ a um exemplo de treinamento:

$$f : \mathbb{R}^m \to \{1,\ldots,t\}.$$

Suponha que tenhamos identificado os k vizinhos mais próximos ($D_k \subset D$) de um ponto de consulta $x^{[q]}$:

$$D_k = \left\{ (x^{[1]}, f(x^{[1]})), \ldots, (x^{[k]}, f(x^{[k]})) \right\}.$$

Agora, podemos definir a hipótese KNN como:

$$h(x^{[q]}) = \arg \max_{y \in \{1,\ldots,t\}} \sum_{i=1}^{k} \delta(y, f(x^{[i]})),$$

onde δ denota a função Delta de Kronecker:

$$\delta(a,b) = \begin{cases} 1, & \text{se } a = b, \\ 0, & \text{se } a \neq b. \end{cases}$$

Ou, de forma mais simples, o conceito de **moda** (BUSSAB; MORETTIN, 2017): Dado um conjunto de dados $X = \{x_1, x_2, \ldots, x_n\}$, onde cada x_i é uma observação. A moda (Mo) é o valor que maximiza a função de frequência absoluta:

$$\text{Mo} = \arg\max_{x} \left(f(x) \right),$$

onde $f(x)$ é o número de vezes que o valor x aparece no conjunto X.

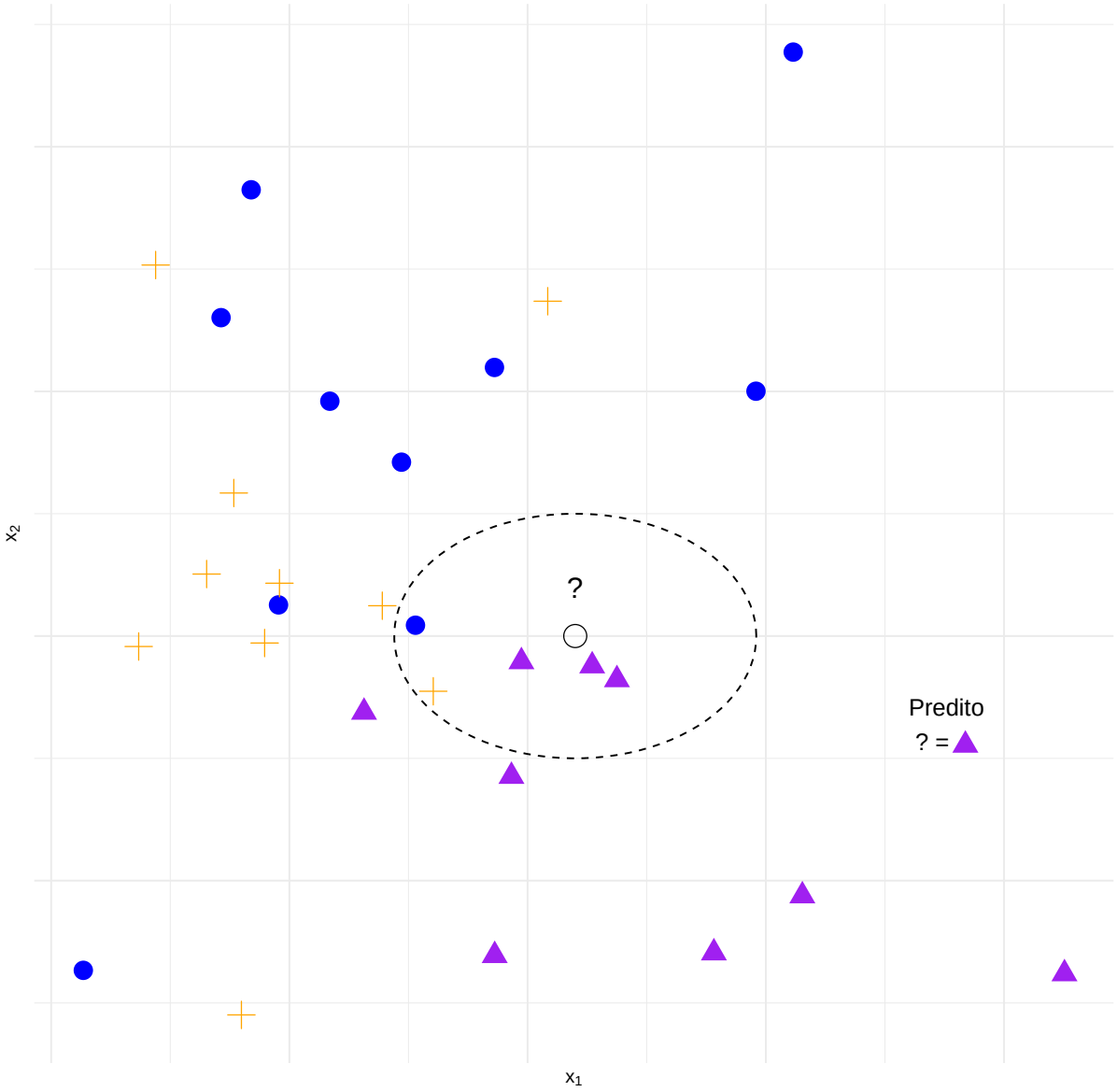
Uma métrica de distância comum para identificar os k vizinhos mais próximos D_k é a medida de distância Euclidiana:

$$d(x^{[a]}, x^{[b]}) = \sqrt{\sum_{j=1}^{m} \left(x_j^{[a]} - x_j^{[b]} \right)^2},$$

que é uma métrica de distância par a par que calcula a distância entre dois pontos de dados $x^{[a]}$ e $x^{[b]}$ sobre os m atributos de entrada. Essa métrica é amplamente utilizada devido à sua simplicidade e interpretabilidade geométrica (HASTIE *et al.*, 2009). Na Figura 6 é apresentada uma ilustração do algoritmo KNN para um problema representado por 3 classes e $k = 5$.

O Quadro 2 resume as características do algoritmo KNN que pode ser utilizado para problemas de classificação e regressão. Nela, são destacadas as vantagens, bem como

Figura 6 – Ilustração do algoritmo KNN para um problema representado por 3 classes e k = 5.



Fonte: Próprio autor.

as desvantagens. Além disso, são apresentadas as métricas de avaliação recomendadas para classificação e regressão.

Quadro 2 – Resumo das características do algoritmo KNN.

Aspecto	**Descrição**
Usado para	Problemas de regressão e classificação.
Vantagens	1. Implementação simples. 2. Robusto a dados ruidosos. 3. Muito flexível, sem nenhuma suposição prévia sobre os dados. 4. Capaz de resolver problemas de Classificação e Regressão. 5. Desempenho relativamente bom para classificação multiclasse. 6. Computação paralela permitida.
Desvantagens	1. Computacionalmente caro, com alta demanda de memória. 2. O tempo de previsão é relativamente longo. 3. Dificuldade em escolher o valor correto de k.
Métricas de Avaliação	1. Para Classificação: Acurácia, Matriz de Confusão, Precisão, Recall, Medida F e Curva AUC-ROC. 2. Para Regressão: MSE, RMSE e MAE.

Fonte: Próprio autor.

2.3.2 Gradient Boosting Machine

Proposto por Freund e Schapire (1997), o *boosting* é uma abordagem geral para construir uma predição extremamente precisa a partir de várias predições aproximadamente corretas. Abordado por Friedman (2001) e Natekin e Knoll (2013), o *Gradient Boosting Machines* (GBM) busca construir modelos preditivos por meio de ajustes sucessivos e regressões não paramétricas. Em vez de construir um único modelo, o GBM começa gerando um modelo inicial e ajusta continuamente novos modelos por meio da minimização de uma função de perda para produzir o modelo mais preciso.

Por definição, um modelo de *boosting* é uma combinação linear ponderada de

aprendizes básicos:

$$F(x;\{\beta_m,a_m\}_{m=1}^{M}) = \sum_{m=1}^{M} \beta_m h(x;a_m),$$

em que $h(x;a)$ é o aprendiz básico parametrizado por a. Esses aprendizes, chamados de fracos, geram hipóteses que têm desempenho apenas ligeiramente melhor que o acaso. Apesar disso, foi provado que o aprendizado recursivo com aprendizes fracos pode alcançar o desempenho de algoritmos fortes (SCHAPIRE, 1990).

No caso de árvores de regressão como aprendizes básicos, o parâmetro a corresponde aos nós de divisão das árvores (FRIEDMAN, 2002). Árvores de decisão dividem o espaço das variáveis em regiões específicas e ajustam a média das respostas dentro de cada região (ELITH *et al.*, 2008).

O processo de otimização é descrito como:

$$P^* = \arg\min_{P} \Phi(P), \quad \Phi(P) = \mathbb{E}_{y,x}[L(y,F(x;P))],$$

em que $F(x;P)$ é o modelo de previsão, dependente dos parâmetros P.

$L(y,F(x;P))$ a função de perda que mede o erro entre a previsão $F(x;P)$ e o valor real y.

$\mathbb{E}_{y,x}$ a expectativa (média) sobre a distribuição conjunta dos dados (x,y) representando os passos de *boosting*.

Para gerar os passos p_m, usa-se o algoritmo de descida mais íngreme, com gradiente dado por:

$$g_m = \left\{ \left. \frac{\partial \Phi(P)}{\partial P_j} \right|_{P=P_{m-1}} \right\},$$

em que $P_{m-1} = \sum_{i=0}^{m-1} p_i$. O passo p_m é então:

$$p_m = -\rho_m g_m, \quad \rho_m = \arg\min_{\rho} \Phi(P_{m-1} - \rho g_m).$$

No caso não paramétrico, a função F é resolvida minimizando:

$$\Phi(F) = \mathbb{E}_{y,x}[L(y,F(x))],$$

e o ótimo é alcançado em:

$$F^*(x) = \sum_{m=0}^{M} f_m(x), \quad f_m(x) = -\rho_m g_m(x).$$

O gradiente no caso não paramétrico é:

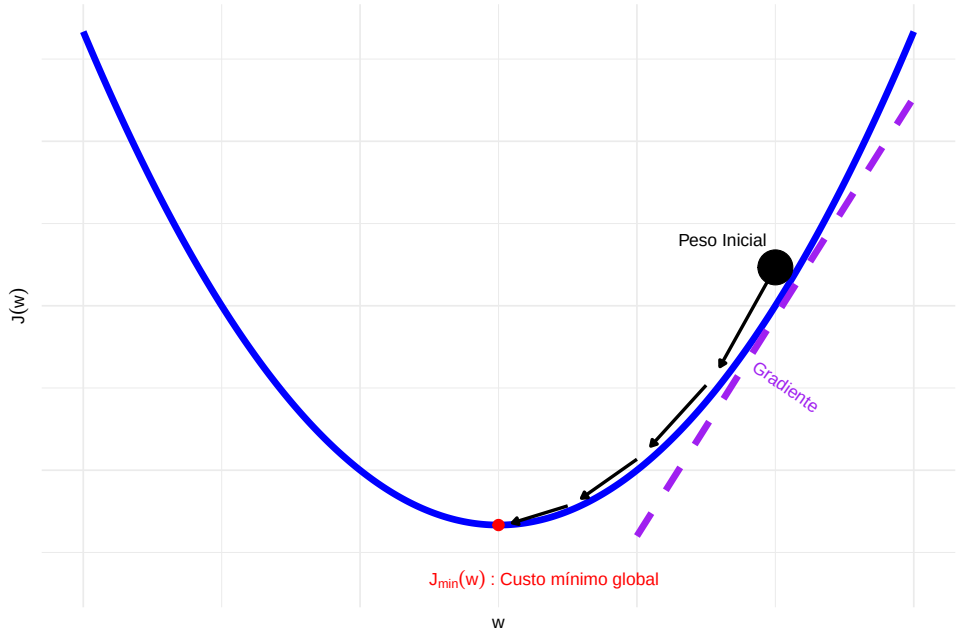
$$g_m(x) = \mathbb{E}_y \left[\frac{\partial L(y, F(x))}{\partial F(x)} \bigg|_x \right]_{F(x)=F_{m-1}(x)}.$$

Finalmente, a busca linear é resolvida como:

$$\rho_m = \arg\min_{\rho} \mathbb{E}_{y,x}[L(y, F_{m-1}(x) - \rho g_m(x))].$$

No algoritmo GBM, o método da descida do gradiente é utilizado para minimizar a função de perda. Em cada iteração m, o GBM calcula os resíduos (gradiente negativo) da função de perda L em relação às previsões atuais $F_{m-1}(x)$ (FRIEDMAN, 2002). A descida do gradiente pode ser ilustrada como na Figura 7.

Figura 7 – Ilustração da descida do gradiente com o objetivo de minimizar uma função de custo.



Fonte: Próprio autor.

No algoritmo GBM, diferentes funções de perda podem ser aplicadas dependendo da natureza do problema a ser resolvido (FRIEDMAN, 2002). Estas funções são projetadas para medir o quão bem o modelo está performando em relação aos dados.

2.3.2.1 Funções de Perda para Regressão

Para problemas de regressão, o objetivo é prever valores numéricos contínuos. Uma das funções de perda mais comuns é a perda $L1$, também conhecida como perda absoluta, que

mede o valor absoluto da diferença entre o valor real (y) e o valor previsto (F):

$$L(y,F)_{L1} = |y-F|.$$

Outra função amplamente utilizada é a perda $L2$, ou erro quadrático médio, que mede o quadrado da diferença entre os valores real e previsto:

$$L(y,F)_{L2} = \frac{1}{2}(y-F)^2.$$

Uma alternativa robusta é a perda Huber, que combina as características das perdas $L1$ e $L2$. Ela é definida como:

$$L(y,F)_{\text{Huber},\delta} = \begin{cases} \frac{1}{2}(y-F)^2, & \text{se } |y-F| \leq \delta, \\ \delta\left(|y-F| - \frac{\delta}{2}\right), & \text{se } |y-F| > \delta, \end{cases}$$

onde δ é um parâmetro que controla a transição entre as perdas $L2$ (para resíduos pequenos) e $L1$ (para resíduos grandes). Essa combinação torna a perda Huber robusta a outliers, mantendo a eficiência da perda $L2$ para erros pequenos.

Além disso, a perda quantil é utilizada para prever valores condicionais em distribuições. Ela é definida como:

$$L(y,F)_{\alpha} = \begin{cases} (1-\alpha)|y-F|, & \text{se } y-F \leq 0, \\ \alpha|y-F|, & \text{se } y-F > 0, \end{cases}$$

onde α é o quantil alvo na distribuição condicional. Quando $\alpha = 0.5$, a perda quantil se reduz à perda $L1$, que estima a mediana condicional. Essa propriedade é útil para modelar distribuições assimétricas ou focar em extremos da distribuição.

2.3.2.2 Funções de Perda para Classificação

Para problemas de classificação, o objetivo é prever categorias discretas. Uma função de perda frequentemente usada é a perda Bernoulli, que é definida como:

$$L(y,F)_{\text{Bern}} = \log(1+\exp(-2yF)),$$

onde $y \in \{-1,+1\}$ é o rótulo verdadeiro e $F(x)$ é a função de decisão do modelo. Essa perda penaliza previsões incorretas de forma suave e está relacionada à estimativa de probabilidades.

No caso do algoritmo Adaboost (Ada), a perda exponencial é amplamente empregada, sendo definida por:

$$L(y,F)_{\text{Ada}} = \exp(-yF).$$

Essa perda atribui maior peso às instâncias classificadas incorretamente, permitindo que o modelo se ajuste iterativamente para melhorar a precisão.

Essas funções de perda permitem que o GBM seja ajustado para diferentes tipos de tarefas, otimizando a previsão para valores contínuos ou categorias discretas.

O Quadro 3 apresenta as principais características do algoritmo GBM, suas vantagens, como a simplicidade de implementação e a robustez a outliers, bem como as desvantagens, como a instabilidade do modelo e o risco de sobreajuste. Além disso, são apresentadas as métricas de avaliação recomendadas para classificação e regressão, incluindo medidas como Acurácia, MSE, RMSE e MAE.

2.3.3 Regressão Logística

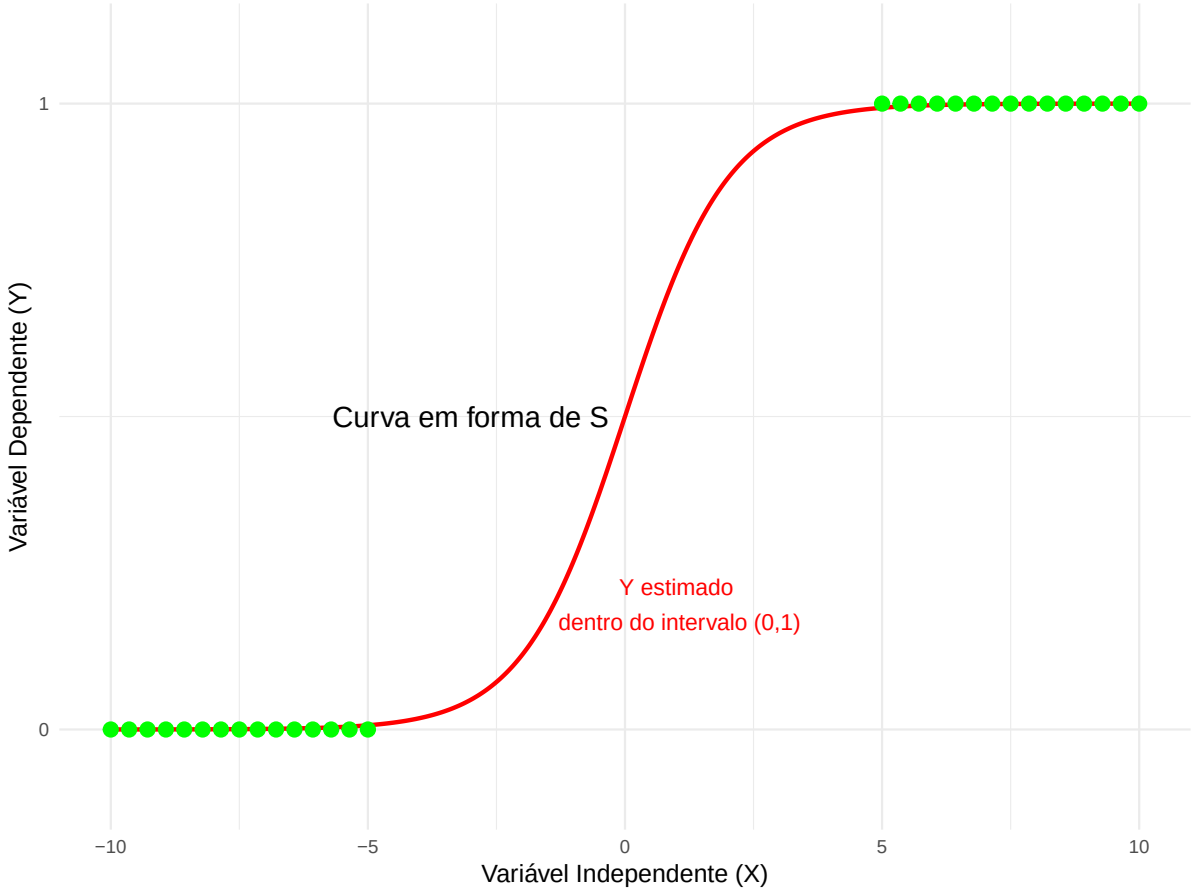
À primeira vista, a Regressão Logística pode parecer mais um problema de regressão do que uma classificação, o que pode gerar certa confusão, já que o nome do algoritmo inclui "regressão". Na realidade, a Regressão Logística é um algoritmo de classificação em Machine Learning, mas a classificação é feita por meio da estimativa de uma probabilidade, e é por isso que é chamada de “regressão” (BURKOV, 2019). Cada algoritmo tem um desempenho superior em determinadas condições, e, para cada problema, é necessário escolher o modelo mais adequado, que, consequentemente, trará os melhores resultados. Dito isso, outros modelos, como a Regressão Linear, são ferramentas úteis, mas não são adequados para problemas de classificação. Isso ocorre principalmente porque a Regressão Logística trabalha com probabilidades que estão no intervalo entre [0,1], enquanto a Regressão Linear produz resultados no intervalo [$+\infty$ e $-\infty$], o que não faz sentido em termos de probabilidade, pois uma probabilidade nunca pode ser negativa ou superior a 1. Além disso, a Regressão Linear busca encontrar a melhor linha de ajuste para a relação entre x e y, o que não é adequado para problemas de classificação. Em vez de uma linha reta, a Regressão Logística é representada por uma função sigmoide com formato de "S", variando entre os limites de 0 e 1 no eixo y, como mostrado na Figura 8.

Quadro 3 – Resumo das características do algoritmo GBM.

Aspecto	**Descrição**
Usado para	Problemas de regressão e classificação.
Vantagens	1.) Altamente preditivo. 2.) Muita flexibilidade — pode otimizar diferentes funções de perda e fornece diversas opções de ajuste de hiperparâmetros que tornam o ajuste da função muito flexível. 3.) Não é necessário pré-processamento de dados — geralmente funciona muito bem com valores categóricos e numéricos. 4.) Capaz de resolver problemas de Classificação e Regressão. 5.) Lida com dados ausentes — imputação não necessária.
Desvantagens	1.) Computacionalmente caro, com alta demanda de memória. 2.) Os GBMs continuarão melhorando para minimizar todos os erros. Isso pode enfatizar demais os outliers e causar sobreajuste. Deve-se usar validação cruzada para neutralizar. 3.) Menos interpretável, embora seja facilmente abordado com várias ferramentas (importância de variáveis, gráficos de dependência parcial, LIME, etc.).
Métricas de Avaliação	1.) Acurácia, Matriz de Confusão, Precisão, Recall, Medida F e Curva AUC-ROC. 2.) Para Regressão: MSE, RMSE e MAE.

Fonte: Próprio autor.

Figura 8 – Representação gráfica da função sigmoide.



Fonte: Próprio autor.

A função sigmoide, que mapeia qualquer valor real x para o intervalo [0, 1], pode ser expressa matematicamente como:

$$f(x) = \frac{1}{1+e^{-x}}. \tag{2.2}$$

Uma das características mais interessantes de (2.2) é a sua capacidade de transformar qualquer valor de x em um valor entre 0 e 1. No entanto, mesmo após essa conversão em probabilidade, como podemos associar essa probabilidade a uma classe de saída (suponhamos que seja uma saída binária)? A solução é simples: o primeiro passo é definir um limiar de decisão (ou limiar de classificação), onde valores acima desse limiar pertencem a uma classe, e abaixo dele pertencem à outra. Embora o limiar seja comumente estabelecido em 0,5, ele pode ser ajustado dependendo do problema.

2.3.3.1 *Características - Função Sigmoide*

A Regressão Logística está intimamente relacionada à distribuição de Bernoulli, que modela dois possíveis resultados (sucesso e fracasso), com probabilidades p e $1-p$, respectivamente.

A função de *odds* (ou razão de chances) é dada por:

$$\text{odds} = \frac{p}{1-p}, \tag{2.3}$$

que p é a probabilidade de um evento ocorrer. Para mapear as *odds* para a probabilidade, usamos o logaritmo de (2.3), conhecida como função logit:

$$\text{logit}(p) = \log\left(\frac{p}{1-p}\right). \tag{2.4}$$

A função sigmoide, que é a inversa de (2.4), é a base da Regressão Logística. A fórmula para a Regressão Logística com uma única variável independente é:

$$p(X) = \frac{1}{1+e^{-(\beta_0+\beta_1 X)}}. \tag{2.5}$$

No caso de múltiplas variáveis independentes, (2.5) é generalizada como:

$$p(X) = \frac{1}{1+e^{-(\beta_0+\beta_1 X_1+\beta_2 X_2+\cdots+\beta_n X_n)}}$$

em que β_0 é o intercepto e os demais β_i's são os coeficientes das variáveis independentes.

2.3.3.2 *Etapa de Aprendizagem*

A Regressão Logística, como outros algoritmos de aprendizado de máquina, passa por uma fase de aprendizado para ajustar o modelo. Inicialmente, os parâmetros $(\beta_0, \beta_1, \ldots, \beta_n)$ são definidos de forma aleatória ou com valores pré-definidos (como zeros). Esses parâmetros são então ajustados iterativamente por meio de métodos de otimização. O objetivo do algoritmo é ajustar esses parâmetros para que os valores estimados se aproximem o máximo possível dos valores reais.

Para isso, utiliza-se uma função de custo que mede a diferença entre os valores preditos ($\hat{y}$) e os valores observados (y). Uma função de custo amplamente utilizada é a Entropia Cruzada (ou Log-loss), que é ideal para problemas de classificação binária. O objetivo é minimizar essa função de custo, de modo que as observações sejam atribuídas à classe mais provável.

Para encontrar as estimativas do coeficientes da Regressão Logística pode-se usar o método de MV. A função de verossimilhança para este modelo é dada por:

$$L(\beta) = \prod_{i=1}^{n} p(x_i)^{y_i}(1 - p(x_i))^{1-y_i}. \tag{2.6}$$

A estimação através do método de MV busca encontrar as estimativas dos parâmetros que maximizam a probabilidade de observar os dados fornecidos. No contexto da Regressão Logística, isso envolve estimar os coeficientes β para que as probabilidades previstas se alinhem o máximo possível com as classes reais.

Para otimizar (2.6), utiliza-se o método da descida do gradiente (GERON, 2019), que ajusta iterativamente os parâmetros β para minimizar a função de custo, movendo-se na direção oposta ao gradiente da função de custo em relação aos parâmetros.

2.3.3.3 Aplicação Prática

A Regressão Logística é amplamente utilizada em problemas de classificação binária, como a detecção de spam em e-mails ou o diagnóstico de doenças com base em sintomas (HASTIE *et al.*, 2009; JAMES *et al.*, 2013). Por exemplo, em um sistema de detecção de spam, a Regressão Logística pode ser usada para prever a probabilidade de um e-mail ser spam com base em características como a frequência de palavras específicas (NG, 2012; KOTSIANTIS *et al.*, 2007). Se a probabilidade estimada for maior que 0,5, o e-mail é classificado como spam; caso contrário, é classificado como não spam (PROVOST; FAWCETT, 2013).

O Quadro 4 apresenta um resumo das caracterítica da Regressão Logística, destacando sua eficiência computacional e facilidade de implementação. No entanto, também são discutidas suas limitações, como a dependência de pré-processamento de dados e a dificuldade em lidar com problemas não linearmente separáveis. As métricas de avaliação, como Acurácia e Curva AUC-ROC, são apresentadas para auxiliar na análise de desempenho.

2.3.4 Árvore de Decisão

Os algoritmos baseados em árvores representam uma classe de modelos supervisionados, capazes de lidar com problemas tanto de classificação quanto de regressão. A Árvore de Decisão é o modelo fundamental dessa categoria, servindo como base para técnicas mais avançadas, como Random Forest e Gradient Boosting. Sua simplicidade, aliada à capacidade de

Quadro 4 – Resumo das Características da Regressão Logistica.

Aspecto	**Descrição**
Usado para	Problemas de classificação.
Vantagens	1.) Eficiência Computacional: Modelo eficiente que requer poucos recursos computacionais e tempo de execução reduzido. 2.) Facilidade de Implementação e Ajuste: Simples de implementar e ajustar, facilitando a adaptação a diferentes conjuntos de dados. 3.) Desempenho em Problemas Linearmente Separáveis: Excelente desempenho em problemas onde as classes podem ser separadas linearmente. 4.) Benchmark Padrão: Frequentemente utilizado como referência para comparar com outros algoritmos.
Desvantagens	1.) Dependência de Engenharia de Atributos: Desempenho comprometido sem um pré-processamento adequado dos dados, incluindo a identificação e exclusão de variáveis irrelevantes ou altamente correlacionadas. 2.) Limitação em Problemas Não Linearmente Separáveis: Dificuldade em resolver problemas onde as classes não podem ser separadas linearmente. 3.) Sensibilidade a Outliers e Risco de Overfitting: Vulnerável a valores discrepantes e propenso ao sobreajuste (overfitting) se não for bem regulado.
Métricas de Avaliação	1.) Para Classificação: Acurácia, Matriz de Confusão, Precisão, Recall, Medida F e Curva AUC-ROC.

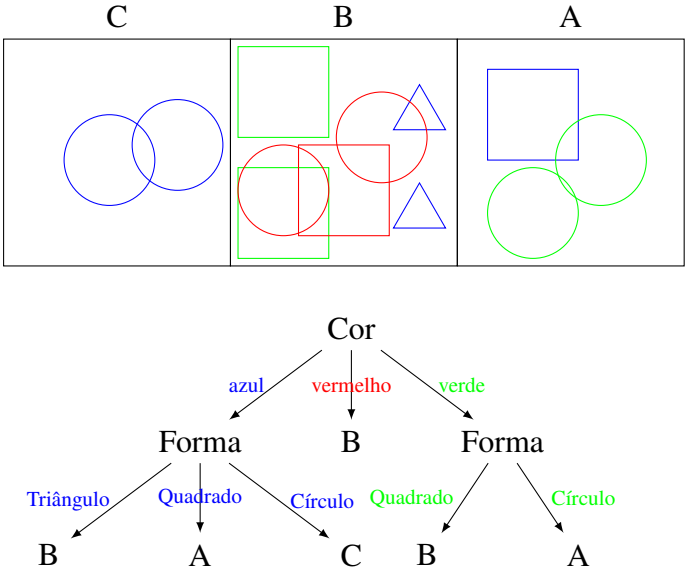
Fonte: Próprio autor.

gerar regras interpretáveis, a torna uma das técnicas mais populares e aplicadas em várias áreas, incluindo ciências sociais e científicas.

A origem dos métodos de análise de decisão remonta a 1931, quando o matemático Frank P. Ramsey introduziu uma abordagem para tomada de decisão que se baseava em probabilidade e utilidade (RAMSEY, 1931). Ao longo das décadas seguintes, outros estudiosos contribuíram significativamente para a evolução desse campo. Um exemplo notável foi o matemático Abraham Wald, que, em 1950, utilizou a teoria dos jogos para desenvolver métodos estatísticos aplicáveis à tomada de decisões em cenários incertos (WALD, 1950). No entanto, foi apenas em 1986 que o primeiro algoritmo completo para árvores de decisão foi formulado, através do trabalho de Breiman *et al.* (1986), com o famoso algoritmo *Classification and Regression Trees* (CART), que se tornou um marco na área.

A árvore de decisão é uma estrutura de dados hierárquica que representa dados por meio de uma estratégia de divisão e conquista. Nesta classe, discutimos árvores de decisão com rótulos categóricos, mas a classificação e regressão não paramétricas também podem ser realizadas com árvores de decisão. Na classificação, o objetivo é estabelecer uma árvore de decisão que represente os dados de treinamento de forma que os rótulos para novos exemplos possam ser determinados. As árvores de decisão são classificadores para instâncias representadas como vetores de atributos (por exemplo, cor=?, forma=?, rótulo=?). Os nós são testes para os valores dos atributos, as folhas definem o rótulo, e em cada nó deve haver um ramo para cada valor do atributo, como representado na Figura 9.

Figura 9 – Exemplo de dados e árvore de decisão.



Fonte: Próprio autor.

2.3.4.1 Classificação: Algoritmo ID3

O algoritmo ID3 (Interactive Dichotomiser) foi introduzido por Ross Quinlan em 1986 e é um dos métodos pioneiros para a construção de árvores de decisão (QUINLAN, 1986). O critério utilizado para dividir os nós no ID3 baseia-se nos conceitos de entropia e ganho de informação, que são fundamentais para a seleção de atributos durante a construção da árvore. A seguir, são descritos esses conceitos e suas respectivas formulações matemáticas.

2.3.4.1.1 Entropia

A entropia é uma medida de desordem ou incerteza em um conjunto de dados, indicando a heterogeneidade dos valores. Quando todos os valores são iguais (máxima homogeneidade), a entropia é igual a zero. Se os valores estiverem igualmente distribuídos, a entropia atinge seu valor máximo de 1. A fórmula para calcular a entropia é dada por:

$$E(S) = -\sum_{i=1}^{c} p_i \log p_i,$$

em que p_i representa a proporção de elementos da classe i no conjunto de dados S e c é o número de classes (QUINLAN, 1986; MITCHELL, 1997).

2.3.4.1.2 Ganho de Informação

O ganho de informação é uma métrica utilizada para selecionar o atributo que melhor divide os dados, maximizando a redução da entropia. Ele é calculado como a diferença entre a entropia do conjunto de dados antes da divisão e a entropia esperada após a divisão com base em um atributo X. A fórmula do ganho de informação (GI) é expressa por:

$$\text{GI}(Y,X) = E(Y) - E(Y|X),$$

em que $E(Y)$ representa a entropia da variável alvo e $E(Y|X)$ representa a entropia esperada após a partição de Y com base em X (QUINLAN, 1986; HAN *et al.*, 2012).

2.3.4.1.3 Vantagens e Limitações

O algoritmo ID3 possui diversas vantagens, como a capacidade de gerar uma árvore de decisão compacta, exigindo menos recursos computacionais. No entanto, ele apresenta limitações, como a incapacidade de lidar com valores contínuos ou ausentes, o que levou ao desenvolvimento de algoritmos alternativos, como o C4.5 (QUINLAN, 1993).

O ID3 foi um marco no desenvolvimento de técnicas de aprendizado de máquina, influenciando algoritmos posteriores como o CART (BREIMAN *et al.*, 1986) e o C4.5. Apesar de suas limitações, o ID3 continua sendo uma referência importante no estudo de árvores de decisão e na aplicação de conceitos como entropia e ganho de informação.

2.3.4.2 Regressão: Algoritmo C4.5

Quinlan (1993) propôs o algoritmo C4.5 como uma evolução do ID3, abordando algumas de suas limitações. O C4.5 resolve os principais problemas do ID3, sendo capaz de lidar tanto com dados contínuos quanto categóricos, além de tratar valores ausentes, removendo a seção da árvore onde esses valores estão presentes. Além disso, Quinlan (1993) percebeu que o critério de divisão baseado no ganho de informação favorecia injustamente atributos com muitos resultados possíveis. Para corrigir esse viés, o C4.5 introduziu o critério de divisão chamado razão de ganho (RG). Esse critério funciona como um normalizador para atributos com muitos resultados, e é dado por:

$$\mathrm{II}(S,A) = -\sum_i \frac{|S_i|}{|S|} \log\left(\frac{|S_i|}{|S|}\right).$$

Depois, essa informação é dividida pelo GI do atributo:

$$\mathrm{RG}(S,A) = \frac{\mathrm{GI}(S,A)}{\mathrm{II}(S,A)}.$$

em que GI e II é Ganho de Informação (GI) e a Informação Intrínseca (II) do atributo, respectivamente. O atributo com a maior RG é escolhido para a primeira divisão, e os subsequentes são selecionados em ordem decrescente de valor da RG.

A principal diferença no processo de treinamento do C4.5 em relação ao ID3 é que, no C4.5, é utilizado um subconjunto dos dados para construir a árvore de decisão. O restante dos dados é então usado para testar a árvore gerada. Se a classificação estiver correta, o processo é finalizado, mas caso contrário, o algoritmo ajusta a árvore de forma interativa (HAN *et al.*, 2012). Recentemente, o algoritmo C4.5 passou por uma atualização para a versão C5.0, que trouxe várias melhorias em relação à versão anterior. Entre as melhorias, destaca-se o fato de que o C5.0 é mais rápido e exige menos poder computacional em comparação com o C4.5, além de ser capaz de realizar classificações mais precisas utilizando árvores menores (QUINLAN, 1993).

2.3.4.3 Classificação e Regressão: Algoritmo CART

O algoritmo CART é uma técnica de árvores de decisão que se destaca por ser capaz de gerar árvores tanto para problemas de classificação quanto de regressão (BREIMAN *et al.*, 1986). A principal diferença do CART em relação aos outros algoritmos é sua flexibilidade em lidar com variáveis contínuas e categóricas.

No caso de um problema de regressão, onde a variável alvo é contínua, o critério para dividir os nós da árvore é a minimização do desvio dos quadrados mínimos (DQM). Esse desvio é uma medida que avalia a diferença entre os valores reais e os valores previstos pelo modelo (BREIMAN *et al.*, 1986). Para problemas de classificação, onde a variável alvo é categórica, o CART utiliza dois critérios principais de divisão: o Índice de Gini ou o Critério Twoing. O Índice de Gini quantifica a probabilidade de um elemento ser classificado incorretamente, sendo calculado como:

$$\text{Índice de Gini}(t) = 1 - \sum_{i} (p(i/t))^2,$$

em que $p(i/t)$ representa a fração de registros pertencentes à classe i no nó t. O valor do Índice de Gini varia entre 0 e 1: um valor de 0 indica que todos os elementos pertencem à mesma classe, enquanto 1 significa que os elementos estão igualmente distribuídos entre todas as classes (BREIMAN *et al.*, 1986). Em alguns casos, o Índice de Gini pode não ser eficiente, especialmente quando o domínio do atributo alvo é grande. Nesses casos, o *Critério Twoing* pode ser uma alternativa mais eficaz. Esse critério é calculado como:

$$\text{Critério Twoing}(t) = \sum_{L,R} |P(L) - P(R)|^2,$$

em que $P(L)$ e $P(R)$ são as probabilidades de ocorrência das classes nos dois subconjuntos gerados pela divisão no nó t (BREIMAN *et al.*, 1986). Após a escolha do critério de divisão, o algoritmo CART gera várias árvores e testa cada uma delas com dados não vistos, a fim de selecionar a árvore que melhor se adapta ao conjunto de dados e oferece a melhor performance (HAN *et al.*, 2012).

No Quadro 5 resume as características básicas dos algoritmos de árvores de decisão discutidos.

Quadro 5 – Características básicas dos algoritmos de árvores de decisão.

Algoritmos	**Critério**	**Atributo**	**Valores Null**	**Outliers**
ID3	Ganho de Informação	Categóricos	Não considerada	Suscetível
CART	Critério de Separação	Categóricos e Numéricos	Considerada	Não considerada
C4.5	Razão de Ganho	Categóricos e Numéricos	Considerada	Suscetível

Fonte: Próprio autor.

Os algoritmos de árvores de decisão, como ID3, C4.5 e CART, são ferramentas poderosas para tarefas de classificação e regressão. Cada algoritmo possui suas vantagens e limitações, sendo importante escolher o mais adequado para o problema em questão. O C4.5 e o CART, em particular, representam avanços significativos em relação ao ID3, oferecendo maior flexibilidade e robustez (BREIMAN *et al.*, 1986; QUINLAN, 1993). No Quadro 6 são apresentadas um resumo das características do metodo de árvore de decisão.

Quadro 6 – Resumo das características do método de árvore de decisão.

Aspecto	**Descrição**
Usado para	Problemas de classificação e regressão.
Vantagens	1.) Não exige muita preparação no pré-processamento em comparação com outros algoritmos. 2.) Não exige normalização e escalonamento antes de rodar o modelo. 3.) É fácil de explicar e muito intuitivo de entender, pois segue a mesma lógica que os seres humanos usam para tomar decisões. 4.) Robusto a outliers.
Desvantagens	1.) Modelo muito instável, uma pequena mudança nos dados causa grandes alterações na estrutura da árvore. 2.) O custo em termos de tempo e poder computacional aumenta exponencialmente à medida que mais rótulos de classe são adicionados ao modelo. 3.) Não é o modelo mais apropriado para problemas de regressão. 4.) Alto risco de overfitting em comparação com outros modelos. 5.) Uma única árvore raramente apresenta resultados sólidos, geralmente é necessário combinar árvores.
Métricas de Avaliação	1.) Para Classificação: Acurácia, Matriz de Confusão, Precisão, Recall, Medida F e Curva AUC-ROC. 2.) Para Regressão: MSE, RMSE e MAE. 3.) Para ambos: Estabilidade, Simplicidade e Poder Discriminatório (Osei-Bryson, 2004).

Fonte: Próprio autor.

3 METODOLOGIA

Neste capítulo, apresentamos a metodologia utilizada para desenvolver e testar modelos de aprendizado supervisionado em conjuntos de dados de esportes eletrônicos. E será realizada com base no League of Legends (LoL), escolhido como uma amostra representativa devido à sua popularidade e à disponibilidade de dados públicos de alta qualidade. A Riot Games, desenvolvedora do LoL, fornece acesso aberto a uma vasta gama de dados sobre campeões e partidas por meio de sua API (Interface de Programação de Aplicações) de Dados Estáticos. Esses dados são altamente confiáveis, uma vez que são retirados diretamente do site oficial da Riot e atualizados frequentemente para garantir a precisão das informações. A API de Dados Estáticos acessa dados através do Data Dragon, um serviço web centralizado que oferece informações detalhadas sobre o jogo, incluindo atributos dos campeões, itens, estatísticas de partidas e muito mais. Essa fonte de dados é essencial para a construção de modelos preditivos eficazes, pois possibilita uma análise robusta dos elementos do jogo.

3.1 Coleta de Dados

Para a realização desta pesquisa, foi essencial obter um conjunto de dados robusto e confiável sobre partidas profissionais de LoL. Este jogo possui uma cena competitiva consolidada, com ligas regionais e internacionais que geram uma grande quantidade de dados estatísticos detalhados.

Os dados utilizados neste estudo foram extraídos do *Oracle's Elixir*, uma plataforma reconhecida na comunidade de esportes eletrônicos por fornecer um banco de dados estruturado e abrangente sobre partidas profissionais de LoL. Esta plataforma agrega informações de diversas competições oficiais, incluindo:

1. League of Legends Championship of The Americas (LTA) - América do Norte e Sul.
2. League of Legends European Championship (LEC) - Europa.
3. League of Legends Pro League (LPL) - China.
4. League of Legends Champions Korea (LCK) - Coreia do Sul.
5. Eventos globais, como o World Championship e o Mid-Season Invitational (MSI).

A base de dados contém informações detalhadas sobre cada partida, incluindo métricas individuais e coletivas, tais como:

1. **Recursos acumulados**: Ouro (`gold`), experiência (`XP`), e abates (`kills`).

2. **Objetivos de jogo**: Torres destruídas, dragões, Barões e Arautos.
3. **Estratégias de draft**: Escolhas de campeões (`picks`) e banimentos (`bans`).
4. **Metadados da partida**: Lado do mapa (`Red` ou `Blue`), patch (versão) do jogo e duração da partida.

A coleta dos dados foi realizada em 13/02/2025 e abrange partidas das principais ligas do ano. Dessa forma, espera-se que base de dados utilizada ofereça um panorama abrangente do desempenho das equipes, permitindo a aplicação de métodos de aprendizado de máquina para prever o resultado das partidas com base em métricas estratégicas e de desempenho. Para este estudo, o foco foi direcionado à classificação binária, onde o objetivo é prever a equipe vencedora com base em atributos como diferenças de recursos em momentos-chave e estratégias de draft (escolha de campeões).

3.2 Análise Exploratória dos Dados

Originalmente, o conjunto de dados era composto com $10,800$ observações e 161 variáveis, capturando métricas detalhadas de profissionais de LoL em nível de equipe e jogador individual. Nele contém tanto variáveis categóricas como identificadores e metadados, bem como métricas de desempenho e estatísticas das partidas.

As análises descritivas iniciais revelam que, embora a maioria das colunas possua informações consistentes, algumas variáveis apresentam contagens inferiores a $10,800$, sugerindo a presença de dados faltantes. Além disso, a diversidade de tipos de dados (com 118 colunas do tipo float64, 20 do tipo int64 e 23 do tipo object) exige cuidados na estratégia e no objetivo do estudo e também na padronização e tratamento adequado.

Para este estudo, optou-se por trabalhar exclusivamente com dados agregados por equipe, evitando a complexidade de análises individuais que poderiam introduzir ruído ou *data leakage* (vazamento de dados no treinamento), uma vez que, o desempenho de um jogador específico vazando informações sobre o resultado global seria um problema. Além disso, filtrou-se apenas partidas com a varíavel `data completeness` (integridade dos dados) completas, excluindo as amostras parciais e garantindo a integridade das análises. Portanto, o conjunto final de dados passou a contar com $1,646$ observações e 161 variáveis. Espera-se que essa estratégia não só reduza possíveis fontes de ruído e redundâncias, mas também garanta uma base de dados mais consistente e alinhada com os objetivos do estudo.

No conjunto de dados em estudo a proporção de vitórias e derrotas foi exatamente

a mesma. Tal comportamento elimina a necessidade de técnicas de balanceamento de classes e sugere que o conjunto de dados é representativo de cenários competitivos equilibrados, isso mostra que a exclusão de amostras parciais foram válidas.

Como podemos visualizar, na Tabela 1, a análise de objetivos neutros, que são elementos estratégicos do mapa que não pertencem a nenhuma equipe inicialmente, mas podem ser conquistados por meio de combate ou interação. Eles fornecem vantagens significativas (como buffs, ouro, experiência ou pressão no mapa) e são disputados por ambas as equipes ao longo da partida e sua captura pode alterar drasticamente o rumo do jogo, revelou padrões críticos, ou seja, a diferença média é 0,321, logo, a taxa de vitória para times que conseguiram o primeiro objetivo é 32,1% maior do que a dos times que não conseguiram. Além disso, a análise revela uma padrão significativo entre o controle de objetivos neutros e o sucesso das equipes em partidas competitivas. Entre esses objetivos, os Dragões destacam-se como um dos fatores mais determinantes para a vitória. Equipes vencedoras tendem a acumular, em média, um número consideravelmente maior de dragões em comparação às equipes perdedoras. Isso ocorre porque cada dragão concede bônus cumulativos que impactam diretamente o desempenho da equipe ao longo da partida. Esses bônus, que variam de acordo com o tipo de dragão (como aumento de dano, velocidade ou regeneração), proporcionam vantagens estratégicas que podem ser decisivas em confrontos posteriores.

Tabela 1 – Taxas de vitória por objetivo.

Objetivo	**Derrotados**	**Vencedores**
firstblood	0,418	0,582
firstdragon	0,446	0,554
firstherald	0,339	0,663
firsttower	0,301	0,699
firstbaron	0,254	0,863

Fonte: Próprio autor.

Portanto, o controle de dragões não apenas reflete superioridade tática, mas também contribui ativamente para a construção de uma vantagem sustentável. Outro objetivo relevante, embora com impacto menos acentuado, é o Arauto do Rift, um monstro neutro localizado na parte superior do mapa. Ao ser derrotado, ele concede um buff que permite invocá-lo para auxiliar na destruição de torres inimigas, favorecendo a equipe com maior controle territorial e vantagem econômica. Equipes vencedoras tendem a capturar mais Arautos, evidenciando a importância de vantagens no inicío do jogo. Essa vantagem inicial frequentemente se traduz

em maior controle de recursos e espaço, criando uma base sólida para o desenvolvimento da partida. Já o Dragão Ancião, uma versão mais poderosa dos dragões elementais, surge nos estágios mais avançados da partida e concede um buff temporário que amplifica o dano e permite a execução instantânea de inimigos abaixo de um limite de vida. Apesar de sua taxa de captura relativamente baixa para ambos os resultados, equipes vencedoras tendem a conquistá-lo com mais frequência, ainda que a diferença em números absolutos seja pequena. Isso reforça a ideia de que, nas fases finais do jogo, o controle de objetivos poderosos como o Dragão Ancião pode ser um fator decisivo, especialmente em confrontos equilibrados.

Além dos objetivos neutros, conquistas iniciais, como First Blood (primeira eliminação), First Tower (primeira torre destruída), First Dragon (primeiro dragão abatido), First Herald (primeiro Arauto do Rift conquistado) e First Baron (primeiro Barão Nashor derrotado), também desempenham um papel crucial. Esses eventos representam momentos decisivos no início da partida, garantindo recursos adicionais como ouro, experiência e controle do mapa. Os dados mostram que equipes vencedoras tendem a conquistar esses objetivos com maior frequência, ainda que a diferença não seja tão expressiva quanto no caso dos dragões. Essas conquistas criam uma inércia positiva, permitindo que a equipe se fortaleça desde cedo e imponha pressão constante sobre o time adversário. Quando bem aproveitada, essa vantagem inicial pode dificultar a recuperação do oponente e aumentar significativamente as chances de vitória.

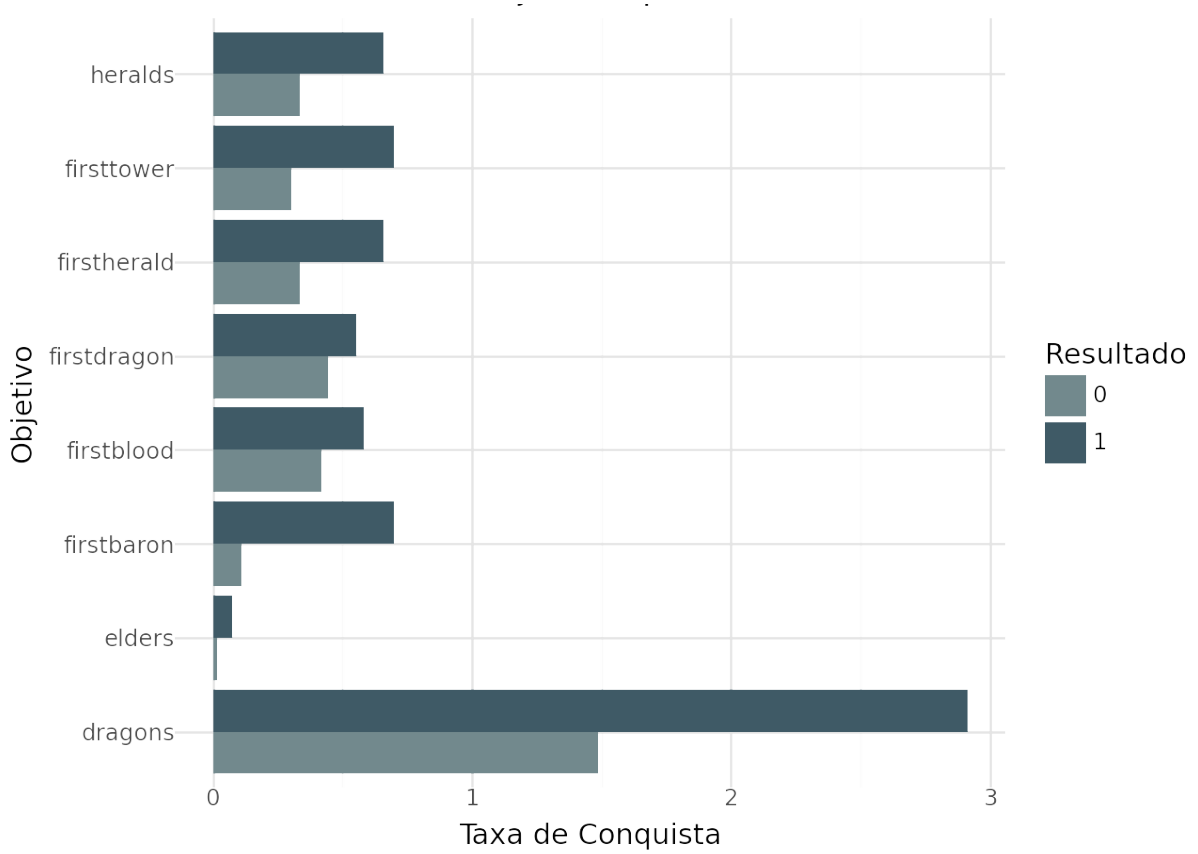
A diferença de ouro (golddiff) entre as equipes é um dos fatores mais determinantes para o resultado final. A vantagem de ouro aos 20 minutos (golddiffat20) apresentou uma correlação significativa com o resultado final ($\rho = 0.62$), indicando que equipes que acumulam mais ouro no meio do jogo tendem a manter essa vantagem até o final. Além disso, estatísticas relacionadas a eliminações em sequência – como doublekills, triplekills, quadrakills e pentakills – também mostram forte correlação com o resultado final. Esses termos se referem ao número de abates consecutivos realizados por um único jogador em um curto período de tempo, sem que um adversário consiga retaliar nesse intervalo. Um doublekill acontece quando o jogador elimina dois inimigos consecutivamente, o triplekill ocorre quando o jogador consegue eliminar três adversários de forma seguida, o quadrakill é quando são quatro eliminações consecutivas e, por fim, o pentakill é quando um único jogador elimina todos os cinco membros do time adversário em sequência, o que é considerado uma das jogadas mais impactantes do jogo.

Os dados mostram que equipes vitoriosas (result=1) apresentam valores significativamente mais elevados nessas métricas do que as derrotadas (result=0), obtendo cerca de 3,5

vezes mais doublekills, 5,3 vezes mais triplekills, 8,3 vezes mais quadrakills e 10 vezes mais pentakills. Além disso, nenhuma equipe que perdeu uma partida conseguiu um pentakill, o que reforça o impacto dessas jogadas na definição do resultado.

Esse padrão reflete o alto nível de controle em confrontos diretos, pois *multi kills* (principalmente *quadra* e *penta kills*) geralmente ocorrem quando o time acumula vantagens de ouro, experiência e coordenação, contribuindo para conquistar e manter a liderança no jogo. Dessa forma, a capacidade de executar eliminações múltiplas em sequência é um forte indicador de domínio e, consequentemente, de maior probabilidade de vitória.

Figura 10 – Média de objetivos por resultado.



Fonte: Próprio autor.

O draft que é o processo de seleção de campeões que acontece antes do início da partida, onde as equipes escolhem e banem campeões de forma alternada também desempenhou um papel estratégico significativo. A análise de picks e bans revelou padrões claros de priorização. Campeões como Xayah (62,26% de vitórias) e Poppy (61,61% de vitórias) destacaram-se como escolhas altamente eficazes, demonstrando impacto positivo consistente no resultado das partidas. Por outro lado, campeões como Skarner (702 bans) e Kalista (489 bans) foram os mais

banidos, refletindo seu impacto significativo na meta do jogo e a necessidade das equipes de neutralizar suas forças. Esses dados evidenciam a importância de uma estratégia de draft bem planejada, que considere tanto as escolhas prioritárias quanto a exclusão de ameaças-chave.

Tabela 2 – Estatísticas de jogos e taxa de vitória por campeão.

Campeão escolhido	**Quantidade de jogos**	**Taxa de Vitória**
Xayah	53	62.26%
Poppy	112	61.61%
Kai'Sa	134	60.45%
Rumble	174	59.77%
Viego	124	58.87%
Nautilus	168	58.33%
Pantheon	54	57.41%
Taliyah	111	56.76%
Skarner	71	56.34%
Yone	121	55.37%

Fonte: Próprio autor.

Tabela 3 – Campeões mais banidos.

Campeão banido	**Quantidade**
Skarner	702
Kalista	489
K'Sante	330
Varus	292
Ambessa	263
Corki	253
Vi	235
Aurora	229
Maokai	223

Fonte: Próprio autor.

Foram identificados problemas relacionados a vazamento de informações e multicolinearidade. Variáveis como towers (torres destruídas) e opptowers (torres do oponente) apresentaram uma correlação bastante forte com o resultado da partida, $\rho = 0,89$. Isso sugere que incluir tais variáveis em modelos preditivos poderia levar a superestimação do desempenho, já que elas são, em essência, consequências diretas da vitória. Além disso, métricas temporais como goldat10, goldat15 e goldat20 mostraram alta correlação entre si ($\rho > 0,85$), indicando redundância. Isso sugere que a diferença de ouro em momentos específicos (por exemplo, golddiffat10) é suficiente para capturar a vantagem acumulada ao longo do tempo. Para mitigar esses

Tabela 4 – Correlação das variáveis com o resultado do jogo.

Variável	Correlação com Resultado
towers	0.894316
opp_towers	0.894316
earned gpm	0.865648
gspd	0.795946
opp_inhibitors	0.765282
inhibitors	0.765282
gpr	0.731079
team kpm	0.707610
assists	0.692096
kills	0.685938
teamkills	0.685938
deaths	0.684676
teamdeaths	0.684676
golddiffat25	0.679585
xpdiffat25	0.663014

Fonte: Próprio autor.

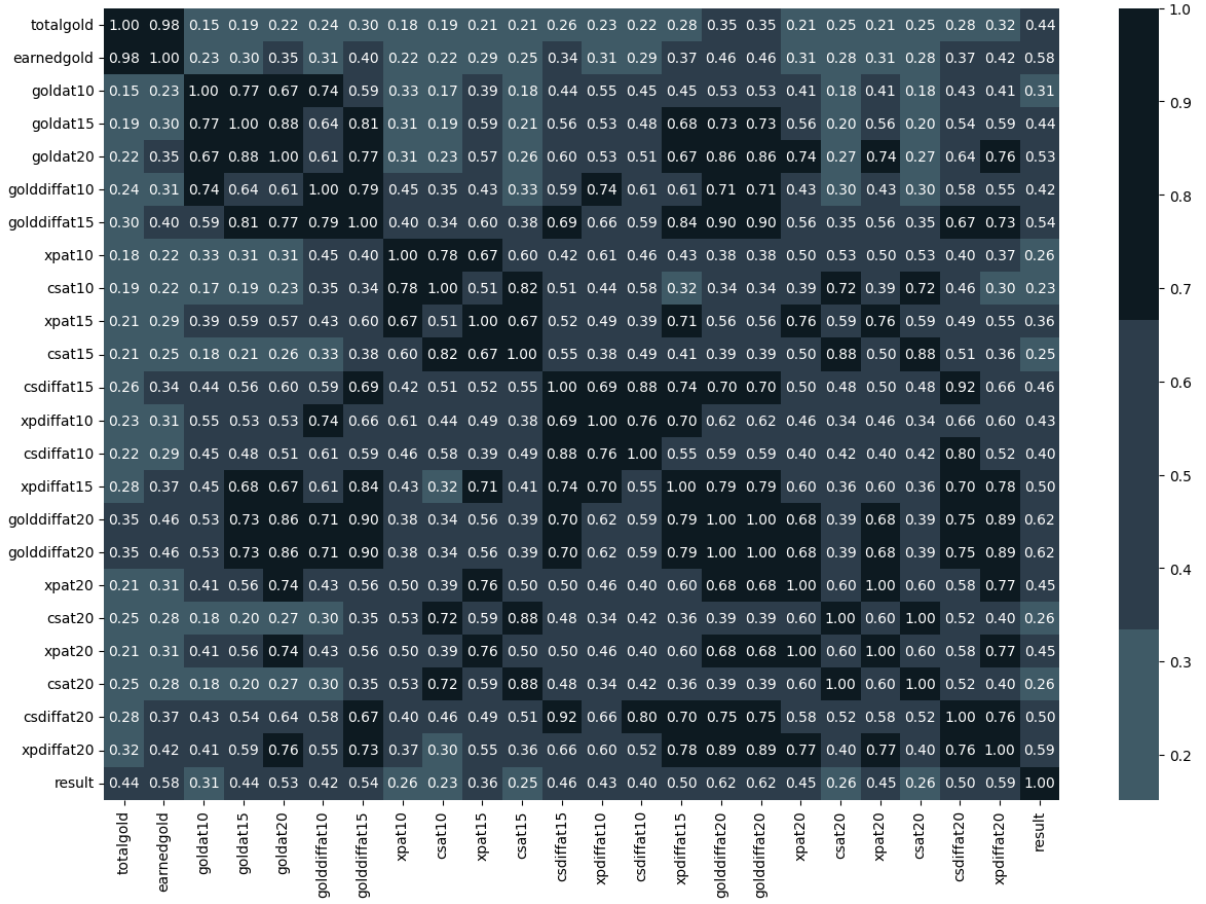
problemas, foram removidas variáveis redundantes (como towers e opptowers) e priorizadas métricas relativas, como golddiff (diferença de ouro) e xpdiff (diferença de experiência), que oferecem uma visão mais clara e concisa da dinâmica do jogo.

Em síntese, a análise demonstra que o controle de recursos, o desempenho em combate e a seleção estratégica de campeões são fatores críticos para o sucesso nas partidas. A vantagem de ouro no início do jogo, aliada a um desempenho superior em combate, cria uma base sólida para a vitória. Além disso, a escolha de campeões prioritários e a exclusão de ameaças-chave no draft são decisões estratégicas que podem influenciar significativamente o resultado.

3.3 Pré-processamento dos Dados

O pré-processamento foi uma etapa crucial, traduzindo a linguagem do jogo em variáveis estatisticamente robustas. Cada decisão, desde a filtragem inicial até a codificação de campeões, foi guiada por dois princípios: fidelidade às dinâmicas do jogo e mitigação de distorções identificadas na Análise Exploratória de Dados (AED). Partidas marcadas como "partial" no campo *datacompleteness* (8,56% do conjunto de dados original) foram excluídas por conterem métricas críticas incompletas, como *ouro* ou *dragões* não registrados. Essa perda amostral (redução para 1.646 partidas) foi uma escolha necessária para garantir análises

Figura 11 – Correlação de variaveis gold, xp e cs.



Fonte: Próprio autor.

temporais precisas, fundamentais para entender o inicio do jogo.

A criação de variáveis de diferença (`golddiffat10` = `goldat10` - `opp_goldat10`) surgiu como resposta a um achado central da AED: métricas absolutas (`goldat10`, `goldat15`) eram altamente correlacionadas entre si ($\rho > 0,70$), mas pouco informativas sobre o desequilíbrio entre equipes. Diferenças relativas, por outro lado, capturavam a assimetria estratégica. Por exemplo, `golddiffat10` teve correlação de $\rho = 0,42$ com o resultado, tornando-se um preditor mais eficiente que `goldat10` isolado ($\rho = 0,31$). Essa mudança reduziu 12 variáveis redundantes, enfatizando a importância de *gold leads* (vantagens de ouro) sobre valores absolutos.

A codificação de picks e bans representou um desafio único. Com mais de 150 campeões no jogo, o One-Hot Encoding geraria uma explosão dimensional (150+ colunas), inviabilizando modelos lineares e aumentando o risco de sobreajuste em um conjunto de dados modesto (1.646 partidas). A solução foi o Target Encoding com suavização (SCIKITLEARN, 2025) (smoothing = 10), uma técnica que substitui as variáveis categóricas por uma média

ponderada da variável-alvo (neste caso, a taxa de vitórias) para cada categoria. Isso permite que cada campeão seja representado por um valor numérico, com um peso proporcional à sua taxa de vitórias no conjunto de dados, ajustado para evitar extremos que poderiam ser causados por campeões pouco frequentes. A suavização aplicada ajuda a equilibrar as informações de campeões populares com as de campeões menos escolhidos, proporcionando uma codificação mais robusta e generalizável. Essa técnica de codificação permitiu sintetizar a complexidade do draft em duas variáveis estratégicas: `pick_strength` e `ban_strength`, calculadas como a média das taxas de vitória dos campeões escolhidos ou banidos. E assim, chegamos no resultado da primeira iteração, onde definiu-se um núcleo de 10 features (variáveis), selecionadas por seu alinhamento com as dinâmicas competitivas identificadas na AED.

A segunda iteração foi um exercício de desconfiança metodológica, projetado para desvendar se variáveis aparentemente influentes, como `side` (lado do mapa) e `patch` (versão do jogo), eram aliadas ou inimigas da generalização do modelo. Os resultados iniciais pareciam contraditórios: ao remover `side`, a acurácia caía de 92,12% para 84,55%, e sem `patch`, para 83,64%. Essas quedas sugeriam que ambas as variáveis tinham um papel decisivo na performance do modelo, embora sua correlação univariada com o resultado fosse quase irrelevante ($\rho = 0,094$ para `side` e $\rho = 0,020$ para `patch`). A investigação aprofundada revelou que outras variáveis, como `golddiffat15` ($\rho = 0,542$) e `ban_strength` ($\rho = 0,489$), tinham correlações mais fortes com o resultado, mas eram redundantes ou pouco interpretáveis.

Na terceira iteração do pré-processamento, buscou-se um refinamento que unisse a eliminação de redundâncias com a preservação do contexto do jogo. Foram removidas as variáveis `golddiffat15`, `xpdiffat15` e `ban_strength` devido à alta correlação ($\rho > 0,75$) que apresentavam com `golddiffat10`, `xpdiffat10` e `pick_strength`, respectivamente. Essa decisão visou reduzir o sobreajuste, um fenômeno que ocorre quando o modelo se ajusta excessivamente aos dados de treinamento e perde a capacidade de generalizar para novos dados, além de concentrar a informação nas variáveis mais robustas. A eliminação das redundâncias facilita a construção de um modelo mais simples e eficiente, permitindo melhor generalização. As variáveis `side` e `patch` foram mantidas e reintroduzidas com um tratamento explícito, uma abordagem que assegura que aspectos como o lado do time e a versão do patch, fatores que influenciam diretamente as dinâmicas do jogo, sejam adequadamente considerados pelo modelo. O pré-processador final foi implementado utilizando o `ColumnTransformer`, uma técnica que permite aplicar transformações específicas a diferentes tipos de dados (variáveis numéricas e

categóricas). O `ColumnTransformer` combinou a aplicação do `RobustScaler` às variáveis numéricas e o `OneHotEncoder` para as variáveis categóricas. O `RobustScaler` foi utilizado para tratar variáveis numéricas de forma que minimize a influência de outliers, essencial para garantir que o modelo não seja distorcido por valores extremos. Já o `OneHotEncoder` foi usado para transformar variáveis categóricas em uma forma que o modelo possa interpretar, criando colunas binárias para cada categoria.

3.4 Aplicação e Validação dos Modelos

Para a construção dos modelos, foi adotado o conceito de Pipeline do scikit-learn, que integra o pré-processamento com o algoritmo de aprendizado, garantindo que as mesmas transformações aplicadas aos dados de treinamento sejam replicadas no conjunto de teste. Para garantir que o modelo seja avaliado de forma justa e robusta, os dados foram divididos em dois conjuntos: 70% foram destinados ao treinamento e 30% ao teste. Foram configurados cinco modelos distintos, abrangendo diferentes abordagens: K-Nearest Neighbors (KNN), Gradient Boosting, Regressão Logística, Árvore de Decisão CART e Árvore de Decisão ID3. Para garantir a robustez na avaliação dos modelos e evitar vazamento de informações, foi empregada a técnica de validação cruzada com GroupKFold (5 divisões), preservando a independência entre grupos de dados. Isso significa que, durante o processo de validação, os dados foram divididos em 5 grupos (ou folds), e a avaliação do modelo foi realizada de forma a garantir que a mesma amostra de dados não fosse utilizada tanto para treinamento quanto para validação no mesmo ciclo. Essa abordagem ajuda a fornecer uma avaliação mais precisa e confiável do desempenho do modelo, evitando o risco de sobreajuste (overfitting). As métricas utilizadas para avaliação foram AUC (Área Sob a Curva ROC), Acurácia e F1-Score. O AUC é uma medida da capacidade do modelo em distinguir entre as classes positivas e negativas, enquanto a Acurácia mede a proporção de previsões corretas. Já o F1-Score é a média harmônica entre precisão e recall.

Na primeira iteração, a Regressão Logística destacou-se como o modelo de melhor desempenho (AUC Teste = 0.889), seguida pelo KNN (AUC Teste = 0.877), conforme mostrado na Tabela 5. Esse resultado sugere que relações lineares entre as diferenças de recursos (ouro, XP) e o resultado final são suficientes para explicar parte significativa da variabilidade dos dados. Modelos não lineares, como o KNN e o *Gradient Boosting*, também demonstraram desempenho competitivo, embora ligeiramente inferiores à Regressão Logística. Por outro lado, as Árvores de Decisão (CART e ID3) tiveram desempenho inferior, possivelmente devido às limitações

impostas pelos hiperparâmetros, como a profundidade máxima das árvores.

Tabela 5 – Desempenho dos Modelos na Primeira Iteração.

Modelo	**AUC Treino**	**AUC Teste**	**Acurácia Teste**	**F1-Score Teste**
KNN	0.915 ± 0.010	0.877	79.39%	0.793
Gradient Boosting	0.913 ± 0.015	0.876	77.88%	0.781
Regressão Logística	0.932 ± 0.007	0.889	81.21%	0.811
Árvore CART	0.865 ± 0.011	0.838	76.06%	0.764
Árvore ID3	0.866 ± 0.010	0.805	74.55%	0.759

Fonte: Próprio autor.

Na segunda iteração, a estratégia de seleção de *features* foi refinada, removendo as variáveis `patch` e `side`. Essa mudança resultou em uma melhoria significativa no desempenho dos modelos, como pode ser observado na Tabela 6. A Regressão Logística obteve uma AUC de 0.930 no conjunto de teste, com acurácia de 85,45% e F1-Score de 0.855, reforçando a hipótese de que relações lineares entre diferenças de recursos são altamente preditivas. O KNN também apresentou desempenho robusto, com AUC de 0.925, acurácia de 85,15% e F1-Score de 0.856. A exclusão de `patch` e `side` reduziu o *overfitting*, simplificando o espaço de *features* e permitindo que os modelos focassem em padrões universais.

Tabela 6 – Desempenho dos Modelos - Segunda Iteração.

Modelo	**AUC Treino**	**AUC Teste**	**Acurácia Teste**	**F1-Score Teste**
KNN	0.913 ± 0.011	0.925	85.15%	0.856
Gradient Boosting	0.899 ± 0.008	0.917	83.33%	0.837
Regressão Logística	0.924 ± 0.011	0.930	85.45%	0.855
Árvore CART	0.861 ± 0.035	0.886	80.00%	0.809
Árvore ID3	0.865 ± 0.015	0.888	80.30%	0.808

Fonte: Próprio autor.

A terceira iteração investigou duas abordagens distintas para a seleção de *features*: uma minimalista (3A), com 5 *features* (`golddiffat10`, `xpdiffat10`, `csdiffat10`, `csdiffat15` e `pickstrength`), e outra contextualizada (3B), que reintegrou os metadados `side` e `patch`. A abordagem minimalista mostrou-se superior para modelos como o KNN, onde a inclusão de metadados na versão 3B introduziu ruído, resultando em quedas de aproximadamente 2,6% no AUC e 4,5% na acurácia, conforme detalhado na Tabela 7. Para a Regressão Logística e as Árvores de Decisão, a diferença entre as abordagens foi marginal, indicando que

esses algoritmos não se beneficiam significativamente do acréscimo de `side` e `patch`.

Tabela 7 – Comparação das Métricas de Desempenho para Iterações 3A (5 features) e 3B (7 features).

Modelo	Iteração 3A (5 features)			Iteração 3B (7 features)		
	AUC	Acurácia	F1-Score	AUC	Acurácia	F1-Score
KNN	0.872	79.09%	0.794	0.849	74.55%	0.756
Gradient Boosting	0.849	76.06%	0.766	0.854	75.45%	0.761
Regressão Logística	0.868	78.18%	0.784	0.867	78.18%	0.784
Árvore CART	0.804	75.15%	0.752	0.804	75.15%	0.752
Árvore ID3	0.800	74.85%	0.752	0.800	74.85%	0.752

Fonte: Próprio autor.

A comparação entre as abordagens 3A e 3B evidencia que, para modelos como o KNN, a abordagem minimalista é preferível, uma vez que a inclusão de metadados pode introduzir ruído e comprometer o desempenho. Para modelos lineares e baseados em regras, a diferença é marginal, sugerindo que a escolha entre as abordagens deve considerar não apenas a performance, mas também a simplicidade e a interpretabilidade do modelo. A Iteração 3A é recomendada para cenários que priorizam a robustez e a clareza dos resultados, enquanto a 3B pode ser explorada em contextos onde informações contextuais adicionais sejam de interesse, desde que se tenha cuidado com o aumento de ruído.

4 RESULTADOS

Na primeira iteração, utilizou-se um conjunto de 11 *features*, incluindo diferenças de recursos nos minutos 10 e 15, além de informações estratégicas como *picks*, *bans* e metadados contextuais (`side` e `patch`). Essa configuração estabeleceu uma *baseline*(linha de base) de desempenho. A Regressão Logística obteve uma AUC de 0.889, enquanto o KNN e o *Gradient Boosting* atingiram AUC de 0.877 e 0.876, respectivamente. As Árvores de Decisão (CART e ID3) apresentaram os piores desempenhos (AUC $\leq$ 0.838).

Na segunda iteração, os metadados `side` e `patch` foram removidos, reduzindo o conjunto para 9 *features*. Essa simplificação elevou o desempenho de todos os modelos, conforme mostrado na Tabela 8. A Regressão Logística atingiu uma AUC de 0.930 (+4.6% em relação à Iteração 1), enquanto o KNN saltou para uma AUC de 0.925 (+5.5%). A remoção dos metadados reduziu o ruído estatístico, permitindo que os modelos focassem em padrões universais.

Tabela 8 – Comparação Iteração 1 vs. Iteração 2 (AUC Teste).

Modelo	**Iteração 1**	**Iteração 2**	**Δ (%)**
Regressão Logística	0.889	0.930	+4.6
KNN	0.877	0.925	+5.5
Gradient Boosting	0.876	0.917	+4.7
Árvore CART	0.838	0.886	+5.7
Árvore ID3	0.805	0.888	+10.3

Fonte: Próprio autor.

A terceira iteração testou duas abordagens: uma minimalista (3A), com 5 *features*, e outra contextualizada (3B), com 7 *features*, reintroduzindo `side` e `patch`. Ambas apresentaram desempenho inferior à Iteração 2, com quedas médias de 6.7% a 10.3% no AUC, conforme detalhado na Tabela 9. A Regressão Logística teve AUC de 0.868 em 3A e 0.867 em 3B, evidenciando que os dados do minuto 15 são insubstituíveis e que a *feature* `pick_strength` não compensa sua remoção.

A Iteração 2 emergiu como a configuração ideal, com ganhos médios de 5.3% no AUC em relação à Iteração 1 e 7.1% em relação às configurações 3A/3B. O sucesso dessa configuração deve-se à seleção inteligente de *features*, eliminação de ruído e manutenção dos dados críticos do minuto 15, que capturam pontos de virada decisivos na partida.

A análise das curvas ROC, apresentada na Figura 12, confirma a robustez dos

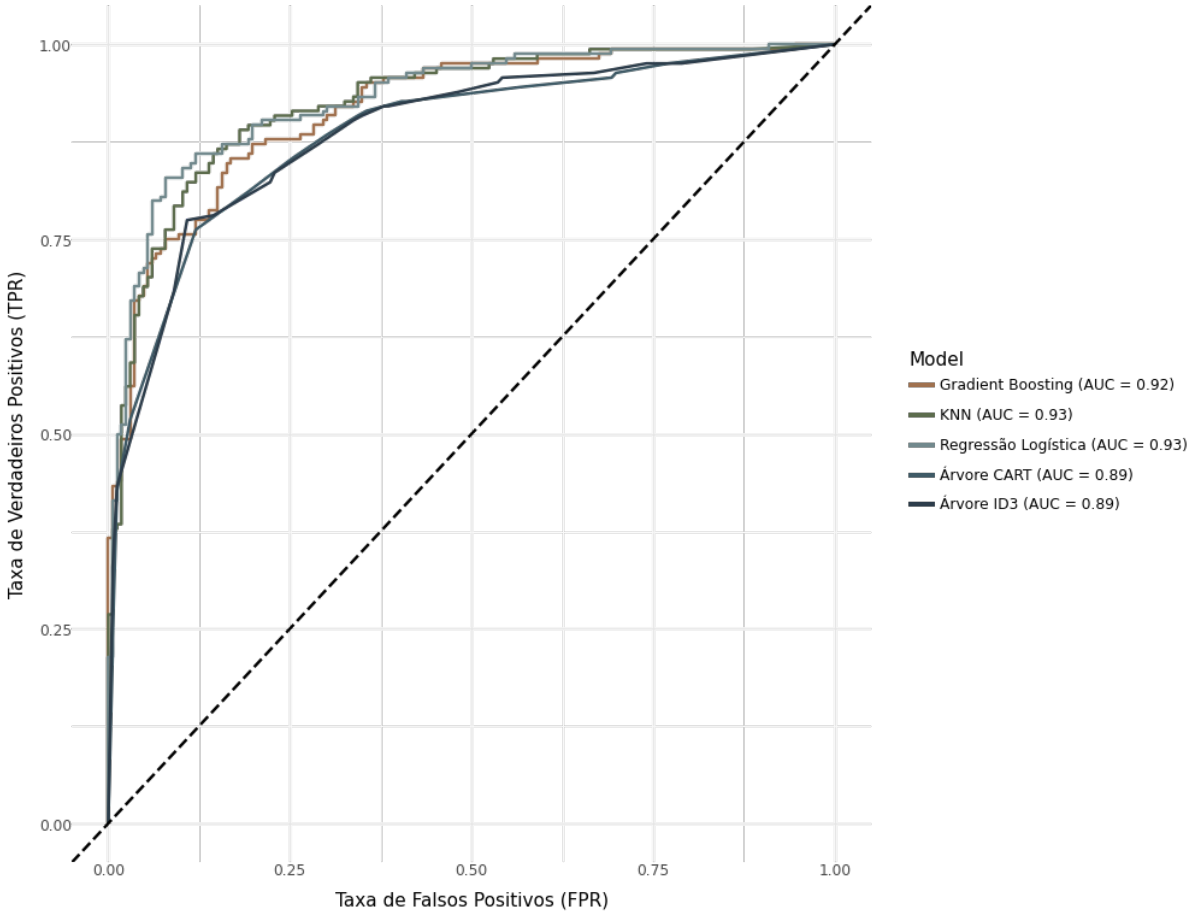
Tabela 9 – Desempenho das Iterações 3A e 3B.

Modelo	**Iteração 3A**		**Iteração 3B**	
	AUC	Acurácia	AUC	Acurácia
Regressão Logística	0.868	78.18%	0.867	78.18%
KNN	0.872	79.09%	0.849	74.55%
Gradient Boosting	0.849	76.06%	0.854	75.45%

Fonte: Próprio autor.

modelos. A Regressão Logística e o KNN atingiram as maiores áreas sob a curva (AUC = 0.93), enquanto as Árvores de Decisão (CART e ID3) ficaram ligeiramente atrás (AUC = 0.89). Esses resultados reforçam a capacidade dos modelos em distinguir corretamente entre vitórias e derrotas.

Figura 12 – Curvas ROC.

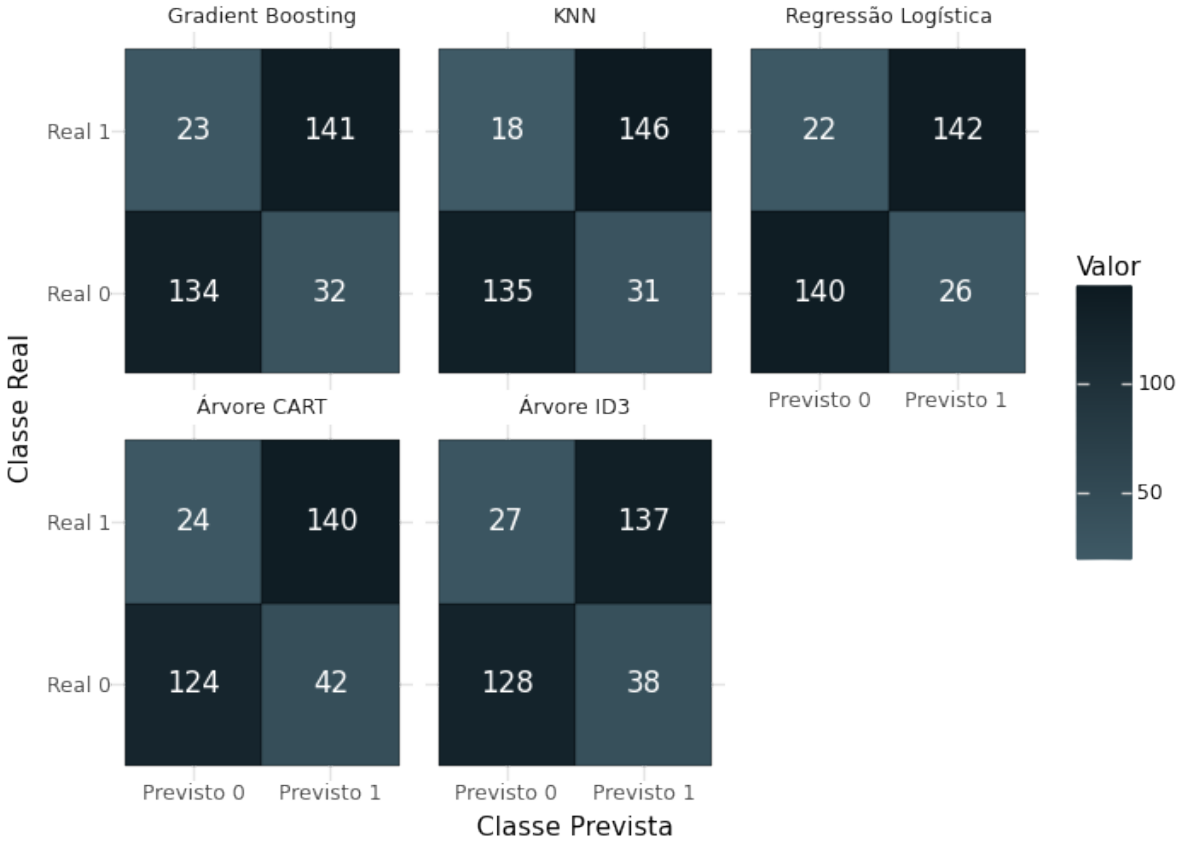


Fonte: Próprio autor.

As matrizes de confusão, ilustradas na Figura 13, mostram que os modelos com melhor desempenho (KNN e Regressão Logística) concentram mais acertos na diagonal secundária,

minimizando erros de classificação. Já as Árvores de Decisão apresentam maior quantidade de falsos negativos, indicando dificuldade em prever vitórias corretamente.

Figura 13 – Matriz de confusão dos modelos.



Fonte: Próprio autor.

Cada modelo utiliza uma abordagem distinta para avaliar a importância das variáveis. No caso do Gradient Boosting, a importância das *features* é determinada pela redução do erro ao longo das árvores do modelo, ou seja, cada variável é avaliada com base em quanto contribui para diminuir a impureza nas divisões das árvores. Variáveis como `golddiffat15` e `pick_strength` se destacam porque possuem uma forte influência na redução da impureza, tornando-as determinantes na predição. Na Regressão Logística, a importância é medida pelos coeficientes associados a cada variável. Quanto maior o valor absoluto de um coeficiente, maior é a influência dessa *feature* sobre o desfecho do modelo. A presença de `golddiffat15` como uma variável importante, com um coeficiente expressivo, sugere que a diferença de ouro aos 15 minutos impacta significativamente a probabilidade de um time vencer o jogo. Para as Árvores de Decisão, como a CART e a ID3, a importância das variáveis é calculada com base na redução da impureza nos nós das árvores. No caso da CART, utiliza-se a impureza de Gini, enquanto na

Tabela 10 – Importância das features para cada modelo.

Feature	Gradient Boosting	Regressão Logística	Árvore CART	Árvore ID3
golddiffat10	0.0276	-0.1485	0.0000	0.0084
xpdiffat10	0.0441	0.3295	0.0000	0.0000
csdiffat10	0.0352	0.1836	0.0236	0.0082
golddiffat15	0.3037	1.1363	0.4425	0.4217
xpdiffat15	0.0762	0.3308	0.0469	0.0489
csdiffat15	0.0435	0.3283	0.0534	0.0431
pick_strength	0.2320	1.2914	0.2482	0.2183
ban_strength	0.2377	1.4383	0.1854	0.2514

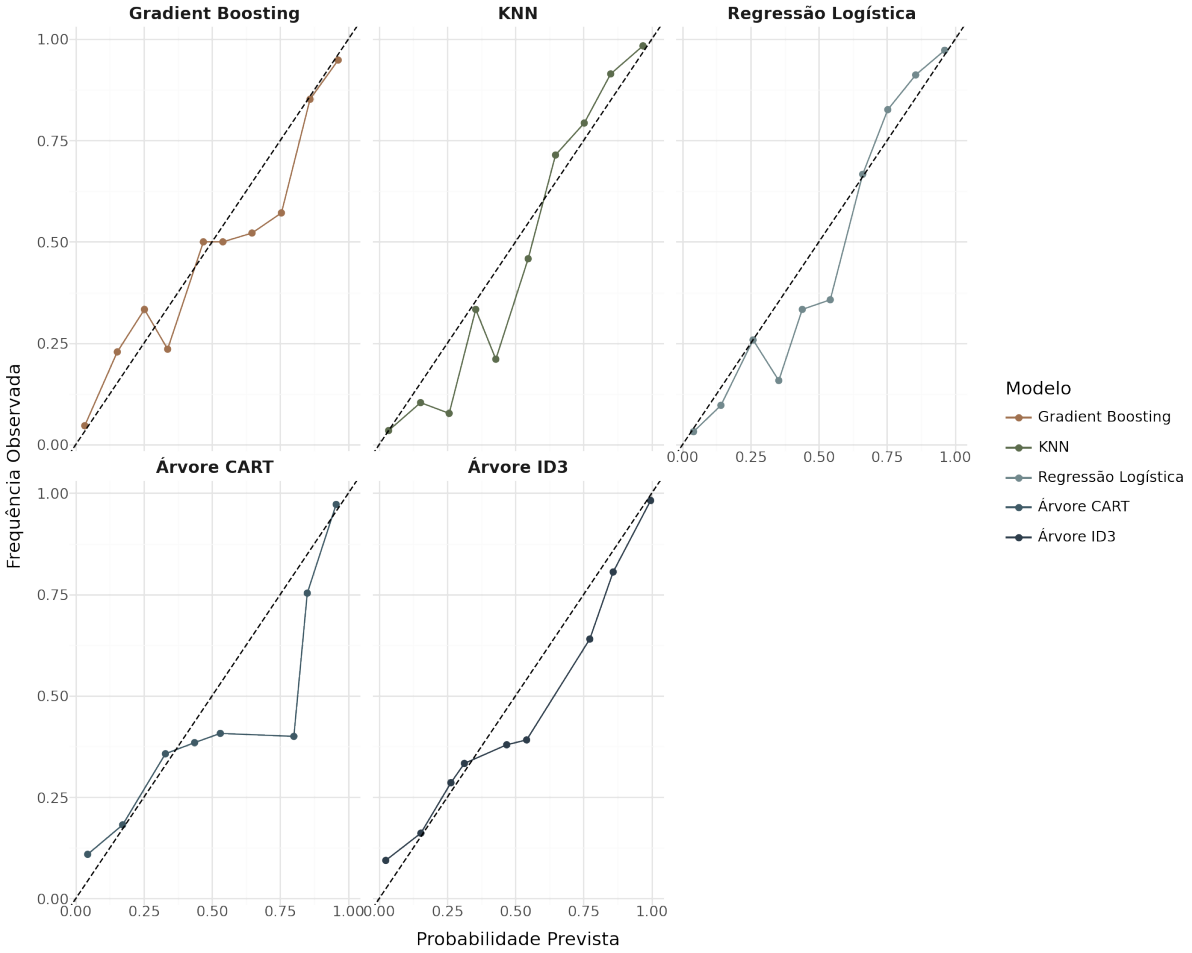
Fonte: Próprio autor.

ID3, o critério é o ganho de informação. A frequência com que uma *feature* reduz a impureza das divisões das árvores a torna mais relevante. A análise revela que variáveis como `golddiffat15` e `ban_strength` contribuem de maneira substancial para a divisão dos dados e, portanto, são as mais impactantes no modelo. Essas abordagens distintas reforçam a importância de escolher o modelo apropriado, pois cada um oferece uma perspectiva diferente sobre a influência das *features*. A presença de `golddiffat15` em todos os modelos como a variável mais impactante sugere que o estado do jogo aos 15 minutos é um fator crucial para a determinação do resultado final, enquanto a importância de variáveis como `pick_strength` e `ban_strength` destaca o papel das estratégias de draft no desempenho das equipes como resume o quadro 10.

A análise de calibração, ilustrada na Figura 14, mostrou que o *Gradient Boosting* apresentou o menor erro médio, indicando que suas probabilidades previstas estão bem alinhadas com a realidade. A Regressão Logística também demonstrou boa calibração, enquanto as Árvores de Decisão tiveram desempenho inferior, com maior variabilidade entre as faixas de probabilidade. O gráfico de calibração é uma ferramenta importante para avaliar a capacidade de um modelo de prever probabilidades bem calibradas. Ele compara as probabilidades previstas pelo modelo com a frequência observada dos eventos. Em um gráfico de calibração ideal, a linha de calibração deve ser próxima da linha reta 45°, o que indicaria que, para uma probabilidade prevista de 0,7, por exemplo, o evento ocorre cerca de 70% das vezes. No caso do *Gradient Boosting*, a linha de calibração está bem próxima da linha reta, indicando que as previsões de probabilidade estão bem ajustadas. Isso significa que, quando o modelo prevê uma probabilidade de 0,6, o evento realmente ocorre em torno de 60% das vezes. A Regressão Logística também mostra uma calibração razoavelmente boa, mas com um pequeno desvio da linha de calibração ideal. Por outro lado, as Árvores de Decisão apresentam uma calibração

inferior, com maior variabilidade entre as faixas de probabilidade. Isso sugere que as previsões de probabilidade para certos valores estão superestimadas ou subestimadas, o que pode ser interpretado como uma menor capacidade do modelo em alinhar suas previsões com a realidade. Esse gráfico é fundamental para identificar se um modelo está subestimando ou superestimando as probabilidades.

Figura 14 – Gráficos de Calibração de cada modelo.



Fonte: Próprio autor.

Em síntese, a Iteração 2 consolidou-se como a configuração ótima, atingindo a maior AUC (0.930) e evidenciando a importância de uma engenharia de *features* seletiva. A manutenção das informações críticas do minuto 15, aliada à remoção de metadados redundantes, garantiu uma melhor generalização dos modelos preditivos, redefinindo as prioridades para a modelagem em ambientes competitivos de jogos eletrônicos.

5 COMENTÁRIOS FINAIS

O presente estudo investigou a aplicação de modelos de aprendizagem supervisionada na análise e previsão de resultados em partidas de LoL. Foram revisados características de algoritmos como Regressão Logística, KNN, GBM e Árvore de Decisão, que serviram de base para a implementação prática dos modelos. A metodologia incluiu um rigoroso pré-processamento dos dados e seleção criteriosa de variáveis, mitigando problemas como *overfitting* e variabilidade, comuns em cenários reais. Essa abordagem mostrou-se eficaz para capturar padrões relevantes dos dados, validando a aplicação de modelos supervisionados em esportes eletrônicos.

A análise dos resultados revelou que, apesar das complexidades inerentes aos dados (*ruídos, outliers* e alta dimensionalidade), os modelos propostos demonstraram desempenho satisfatório. Métricas como acurácia e AUC corroboraram a eficácia dos algoritmos em prever resultados e oferecer *insights* estratégicos. Esse desempenho enfatiza o potencial dos métodos de aprendizagem supervisionada para a análise preditiva em cenários dinâmicos e complexos, como partidas competitivas de LoL.

É possível encontrar na literatura outros trabalhos e discussões em fóruns da internet sobre a aplicação de aprendizado de máquina a dados de LoL. Um desses trabalhos foi escrito por Junior (2023). Ao compararmos os resultados desse estudo com os do presente trabalho, observa-se uma convergência metodológica e conceitual. Além disso, em Junior (2023), nota-se que a precisão dos modelos aumenta conforme o tempo decorrido na partida, evidenciando que variáveis como *total de ouro*, *abates* e *controle de objetivos* tornam-se mais relevantes à medida que o jogo avança. Ao compararmos as mesmas métricas utilizadas em ambos os trabalhos, identificam-se algumas diferenças importantes. No estudo de Junior (2023), o *Random Forest* apresentou um desempenho crescente ao longo do tempo, com acurácia variando de 0.6085 (20% do tempo) até 0.8543 (80% do tempo), enquanto a Regressão Logística obteve valores semelhantes, indo de 0.6268 para 0.8502. No presente estudo, os melhores modelos já apresentaram acurácias superiores desde a segunda iteração, com a Regressão Logística atingindo 85.45% e o KNN 85.15%, sem depender da evolução temporal dos dados.

Além das discussões anteriores, é relevante contrastar este estudo com trabalhos seminalmente distintos, como (YANG *et al.*, 2023) e (CONLEY; PERRY, 2013), que exploraram predições em Dota 2. Enquanto Yang et al. alcançaram 93,73% de acurácia no 40º minuto

usando dados em tempo real e modelos sequenciais (ASM), este trabalho demonstrou que variavéis de diferença de recursos (ex.: diferença de ouro no minuto 15) no LoL permitem 85,45% de acurácia já na segunda iteração, sem exigir monitoramento contínuo. Essa diferença reflete a natureza distinta dos jogos: o LoL é decidido mais cedo, enquanto o Dota 2 depende de eventos late-game. Já Conley & Perry (2013), ao focarem em recomendação de heróis via Regressão Logística (69,8% de acurácia), destacaram sinergias estáticas entre personagens, mas negligenciaram variáveis dinâmicas (ex.: desempenho em jogo), limitando a generalização. Em contraste, a eliminação de metadados (`side`, `patch`) neste estudo reduziu ruídos estatísticos, permitindo que modelos como a Regressão Logística focassem em padrões universais (ex.: `golddiffat15`), superando abordagens baseadas apenas em combinações de heróis. Embora o estudo de Yang et al. seja superior em granularidade temporal, a simplicidade e eficiência desta metodologia a tornam mais adequada para análises em tempo real no LoL, onde decisões meio do jogo são críticas. Assim, enquanto a literatura explora complexidades temporais ou sinergias estáticas, este trabalho avança ao equilibrar interpretabilidade e eficiência, provando que variáveis econômicas e estratégias de draft são suficientes para previsões robustas em ambientes competitivos.

Em um cenário hipotético, considerando uma equipe que, durante o draft, seleciona Draven (Atirador de inicio game agressivo) e Leona (suporte com forte engajamento), garantindo domínio na rota inferior. Nos primeiros 12 minutos, a dupla acumula dois abates, assegura o primeiro dragão e estabelece prioridade no mapa. Aos 15 minutos, a equipe mantém uma diferença de ouro de 2,000 (golddiffat15) e controle de um dragão e uma torre destruída. Segundo o modelo, essa equipe teria 85% de probabilidade de vitória, desde que evite confrontos isolados e priorize objetivos estratégicos (como o Arauto do Rift para pressão em outras rotas).

REFERÊNCIAS

ALPAYDIN, E. **Introduction to machine learning**. [S.l.]: MIT press, 2020.

BELLEM, O. Hybrid models combining knn and deep learning for anomaly detection in iot networks. **Journal of Artificial Intelligence Research**, AI Access Foundation, v. 76, p. 45–67, 2023.

BISHOP, C. M. **Pattern Recognition and Machine Learning**. [S.l.]: Springer, 2006. ISBN 978-0-387-31073-3.

BREIMAN, L.; FRIEDMAN, J. H.; OLSHEN, R. A.; STONE, C. J. **Classification and Regression Trees**. [S.l.]: Wadsworth International Group, 1986.

BURKOV, A. **The Hundred-Page Machine Learning Book**. [S.l.]: Andriy Burkov, 2019.

BUSSAB, W. O.; MORETTIN, P. A. **Estatística Básica**. 9ª. ed. São Paulo: Saraiva, 2017.

CONLEY, K.; PERRY, D. How does he saw me? a recommendation engine for picking heroes in dota 2. **Stanford University Project Report**, 2013. Disponível em: <https://github.com/kconley/dota2-hero-recommendation>.

COVER, T. M.; HART, P. E. Nearest neighbor pattern classification. **IEEE Transactions on Information Theory**, v. 13, n. 1, p. 21–27, 1967.

DUDA, R. O.; HART, P. E.; STORK, D. G. **Pattern Classification**. 2nd. ed. [S.l.]: John Wiley Sons, 2001. ISBN 978-0-471-05669-0.

ELITH, J.; LEATHWICK, J. R.; HASTIE, T. A working guide to boosted regression trees. **Journal of Animal Ecology**, Wiley, v. 77, n. 4, p. 802–813, 2008.

FIX, E.; JR., J. L. H. **Discriminatory Analysis: Nonparametric Discrimination, Consistency Properties**. [S.l.], 1951.

FREUND, Y.; SCHAPIRE, R. E. A decision-theoretic generalization of on-line learning and an application to boosting. **Journal of computer and system sciences**, Elsevier, v. 55, n. 1, p. 119–139, 1997.

FRIEDMAN, J. H. Greedy function approximation: a gradient boosting machine. **Annals of Statistics**, Institute of Mathematical Statistics, v. 29, n. 5, p. 1189–1232, 2001.

FRIEDMAN, J. H. Stochastic gradient boosting. **Computational Statistics & Data Analysis**, Elsevier, v. 38, n. 4, p. 367–378, 2002.

GALEANO, P. Data science, big data and statistics. ResearchGate, 2019.

GARCIA, S.; HERRERA, F.; SHAWE-TAYLOR, J. Knn as a baseline method for machine learning: Does it still deserve its place? **Data Mining and Knowledge Discovery**, Springer, v. 34, n. 4, p. 1215–1239, 2020.

GERON, A. **Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow**. 2. ed. Sebastopol: O'Reilly Media, 2019. 107–113 p.

GOU, J.; MA, H.; OU, W.; ZENG, S.; RAO, Y.; YANG, H. A generalized mean distance-based k-nearest neighbor classifier. **Expert Systems with Applications**, Elsevier, v. 115, p. 356–372, 2019.

HAN, J.; KAMBER, M.; PEI, J. **Data Mining: Concepts and Techniques**. 3rd. ed. [S.l.]: Morgan Kaufmann, 2012.

HASTIE, T.; TIBSHIRANI, R.; FRIEDMAN, J. **The Elements of Statistical Learning: Data Mining, Inference, and Prediction**. [S.l.]: Springer, 2009.

HATTENSTONE, S. How esports became a multimillion-dollar business. **The Guardian**, 2017. Acesso em: 8 jan. 2025. Disponível em: <https://www.theguardian.com/>.

INTELLIGENCE, M. **Esports Market - Growth, Trends, COVID-19 Impact, and Forecasts (2023 - 2029)**. 2023. Acesso em: 8 jan. 2025. Disponível em: <https://www.mordorintelligence.com/>.

IZBICKI, R. **Aprendizado de Máquina: uma abordagem estatística**. livro eletrônico, 2020. Disponível em formato eletrônico. Disponível em: <https://tiagoms.com/publications/ame/AME.pdf>.

JAMES, G.; WITTEN, D.; HASTIE, T.; TIBSHIRANI, R. **An Introduction to Statistical Learning: with Applications in R**. [S.l.]: Springer, 2013.

JUNIOR, J. B. d. S. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação), **League of Legends: Predição de Resultados em Tempo Real**. Campina Grande - PB: [s.n.], 2023.

KOTSIANTIS, S. B.; ZAHARAKIS, I.; PINTELAS, P. Supervised machine learning: A review of classification techniques. **Emerging Artificial Intelligence Applications in Computer Engineering**, v. 160, p. 3–24, 2007.

Meio & Mensagem. Pesquisa game brasil: 63,8% dos brasileiros acompanham esports regularmente. 2023. Acesso em: 8 jan. 2025. Disponível em: <https://www.meioemensagem.com.br/>.

MITCHELL, T. M. **Machine Learning**. New York: McGraw-Hill, 1997.

Máquina do Esporte. 2,76 bilhões de horas de esports foram assistidos em 2023, revela relatório. 2023. Acesso em: 8 jan. 2025. Disponível em: <https://maquinadoesporte.com.br/>.

MüLLER, K.-R.; MIKA, S.; RÄTSCH, G.; TSUDA, K.; SCHÖLKOPF, B. An introduction to kernel-based learning algorithms. **IEEE Transactions on Neural Networks**, v. 12, n. 2, p. 181–191, 2001.

NATEKIN, A.; KNOLL, A. Gradient boosting machines, a tutorial. **Frontiers in Neurorobotics**, Frontiers Media SA, v. 7, p. 21, 2013.

NEWS, W. **Receita de eSports no Brasil deve atingir US$ 16 milhões em 2023, mas precisa investir em segurança**. 2023. Acesso em: 8 jan. 2025. Disponível em: <https://web3news.com.br/>.

NG, A. **Machine Learning Yearning**. 2012. <https://www.mlyearning.org/>.

PROVOST, F.; FAWCETT, T. **Data Science for Business: What You Need to Know about Data Mining and Data-Analytic Thinking**. [S.l.]: O'Reilly Media, 2013.

QUINLAN, J. R. Induction of decision trees. **Machine Learning**, Springer, v. 1, n. 1, p. 81–106, 1986.

QUINLAN, J. R. **C4.5: Programs for Machine Learning**. [S.l.]: Morgan Kaufmann, 1993.

RAMSEY, F. P. **The Foundations of Mathematics and Other Logical Essays**. [S.l.]: Routledge & Kegan Paul, 1931.

RASCHKA, S. **STAT 479: Machine Learning Lecture Notes**. 2020. Disponível em: <https://sebastianraschka.com/>. Acesso em: [inserir data].

RATRA, R.; PREETI, G. Experimental evaluation of open source data mining tools (weka and orange). International Journal of Engineering Trends and Technology., 2020.

SCHAPIRE, R. E. The strength of weak learnability. **Machine Learning**, Springer, v. 5, n. 2, p. 197–227, 1990.

SCIKITLEARN. **TargetEncoder**. 2025. Accessed: 2025-03-17. Disponível em: <https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.TargetEncoder.html>.

SEGAL, D. Behind league of legends, esports' main attraction. **The New York Times**, 2014. Acesso em: 8 jan. 2025. Disponível em: <https://www.nytimes.com/>.

STANTON, J. M. **A Brief History of Linear Regression for Statistics Instructors**. 2001. Journal of Statistics Education Volume 9, Number 3 (2001). Disponível em: <https://jse.amstat.org/v9n3/stanton.html>. Acesso em: 23 nov. 2024.

VAPNIK, V. N. **The Nature of Statistical Learning Theory**. New York: Springer, 1995. ISBN 978-0-387-94559-8.

WALD, A. **Statistical Decision Functions (Based on the Sequential Analysis)**. [S.l.]: Wiley, 1950.

WARMAN, P. **Global eSports Market Report**. 2017. Acesso em: 8 jan. 2025. Disponível em: <https://newzoo.com/>.

WARR, P. 32 million people watched the league of legends season 3 world championship. **Eurogamer**, 2014. Acesso em: 8 jan. 2025. Disponível em: <https://www.eurogamer.net/>.

WITTEN, I. H.; FRANK, E. **Data Mining: Practical Machine Learning Tools and Techniques with JavaImplementations**. [S.l.]: Morgan Kaufmann Press, 1999.

YANG, Y.; QIN, T.; LEI, Y.-H. Real-time esports match result prediction. **arXiv preprint arXiv:1701.03162**, 2023. Disponível em: <https://arxiv.org/abs/1701.03162>.

ZHANG, S. Adaptive k-nearest neighbor classification for biomedical data. **IEEE Transactions on Neural Networks and Learning Systems**, IEEE, v. 33, n. 7, p. 2891–2903, 2022.

APÊNDICE A – LISTA DE NOMENCLATURAS

Termo	Descrição
Acurácia	Proporção de previsões corretas no total de amostras.
Aprendizado Supervisionado	Método de treinamento de modelos com dados rotulados.
Arauto do Rift	Monstro que ajuda na destruição de torres.
Árvore de Decisão	Divide os dados com regras hierárquicas baseadas em atributos.
AUC-ROC	Mede a capacidade de um modelo em distinguir entre classes.
Barão Nashor	Monstro poderoso que fortalece a equipe.
Calibração	Processo de avaliar o quão bem as probabilidades previstas por um modelo probabilístico refletem as verdadeiras probabilidades observadas.
Campeão	Personagem controlado por um jogador, com habilidades únicas.
Correlação (ρ)	Mede relação linear entre duas variáveis (-1 a 1).
CS (Creep Score)	Quantidade de tropas abatidas.
Data Leakage	Uso indevido de informações dos dados de teste ou validação no treinamento do modelo, comprometendo a avaliação real.
Dragão	Monstro que concede bônus estratégicos.
Draft	Fase onde campeões são escolhidos e banidos.
Entropia	Mede a incerteza de um conjunto de dados.
F1-Score	Média harmônica entre precisão e recall.
Feature (Variável)	Atributo usado como entrada para o modelo.
First Blood	Primeira eliminação na partida.
First Tower	Primeira torre destruída no jogo.
Ganho de Informação	Redução da entropia após uma divisão.
Gold (Ouro)	Recurso usado para comprar itens e fortalecer campeões.
Golddiff	Diferença de ouro entre equipes em um momento do jogo.
Gradient Boosting Machine (GBM)	Combinação sequencial de modelos fracos para reduzir erros.
Hiperparâmetros	Configurações externas ao modelo que controlam o processo de aprendizado e precisam ser definidos antes do treinamento.

Termo	Descrição
IID (Independentes e Identicamente Distribuídas)	Conjunto de variáveis aleatórias que são estatisticamente independentes entre si e seguem a mesma distribuição de probabilidade.
K-Nearest Neighbors (KNN)	Classifica ou prevê valores com base nos "k"exemplos mais próximos.
Matriz de Confusão	Mostra acertos e erros por classe.
Meta	Estratégia dominante no patch atual do jogo.
Multi-kills	Eliminações consecutivas (doublekill, triplekill etc.).
One-Hot Encoding	Transforma variáveis categóricas em indicadores binários.
Overfitting (Superajuste)	Quando o modelo se ajusta demais aos dados de treino.
Regressão Logística	Algoritmo de classificação binária baseado em função sigmoide.
Regularização	Penaliza modelos muito complexos para evitar overfitting.
Summoner's Rift	Mapa principal do jogo competitivo.
Target Encoding	Substitui categorias pela média da variável-alvo.
Underfitting (Subajuste)	Modelo muito simples para capturar os padrões dos dados.
Validação Cruzada	Avaliação dividindo os dados em múltiplos treinos/testes.
XP (Experiência)	Pontos que permitem evoluir habilidades.
Xpdiff	Diferença de experiência entre equipes.