Universidade Federal do Estado do Rio de Janeiro
Centro de Ciências Exatas e Tecnologia
Escola de Informática Aplicada

Aprendizagem de Máquina para Predição de Partidas de *League of Legends*

Gabriel da Rocha Lage

Orientador
Pedro Nuno de Souza Moura

Rio de Janeiro, RJ – Brasil
Fevereiro, 2025

Ficha Catalográfica

Catalogação informatizada pelo autor

Lage, Gabriel da Rocha
L174 Aprendizagem de Máquina para Predição de Partidas de League of Legends / Gabriel da Rocha Lage. -- Rio de Janeiro : UNIRIO, 2025.
50 p

Orientador: Pedro Nuno de Souza Moura.
Trabalho de Conclusão de Curso (Graduação) - Universidade Federal do Estado do Rio de Janeiro, Graduação em Sistemas de Informação, 2025.

1. Aprendizagem de Máquina. 2. League of Legends. 3. Valores Shapley. I. Moura, Pedro Nuno de Souza, orient. II. Título.

Aprendizagem de Máquina para Predição de Partidas de *League of Legends*

Gabriel da Rocha Lage

Projeto de Graduação apresentado à Escola de Informática Aplicada da Universidade Federal do Estado do Rio de Janeiro (UNIRIO) para obtenção do título de Bacharel em Sistemas de Informação.

Aprovado por:

Prof. Pedro Nuno de Souza Moura, D.Sc. (UNIRIO)

Prof. Jobson Luiz Massollar da Silva, D.Sc. (UNIRIO)

Prof. Jefferson Elbert Simões, D.Sc. (UNIRIO)

Rio de Janeiro, RJ – Brasil
Fevereiro, 2023

Resumo

A década de 2020 presenciou uma nova revolução no campo da Inteligência Artificial, com produtos como o ChatGPT, da empresa OpenAI, dominando as notícias e se inserindo no senso comum. Com isso, a força das IAs, sobretudo no aspecto de Aprendizagem de Máquina, foi demonstrada em um nível nunca antes visto, possibilitando a imaginação e a exploração de suas técnicas nos mais variados âmbitos da sociedade. À luz desta possibilidade, este estudo busca inserir a prática e as soluções de IA no campo do esporte eletrônico, situado entre a competitividade e o entretenimento, em busca de encontrar vantagens em padrões desconhecidos através de dados, inspirado na revolução gerada pela análise de dados ocorrida na principal liga de *baseball* do mundo (Lewis, 2004). O intuito do projeto é utilizar técnicas de Aprendizagem de Máquina para ler *drafts* profissionais de *League of Legends* e, a partir destes, prever qual das equipes será vencedora da partida. Apesar do resultado da partida não ser determinado diretamente pelo *draft*, a rede poderá trazer novas ideias e encontrar novos paradigmas para estratégias profissionais do jogo.

Palavras-chave: League of Legends, Aprendizagem de Máquina, Interpretabilidade, Valor de Shapley.

Abstract

The 2020 decade has witnessed a new revolution in the field of Artificial Intelligence, as products such as ChatGPT, from the OpenAI company, dominate the news and are inserted into common sense. As a result, the power of AI, especially in the aspect of Machine Learning, was demonstrated at an unprecedented level, enabling the imagination and exploration of its techniques in various aspects of society. In light of this possibility, this study seeks to integrate the practice and solutions of AI into the realm of esports, situated between competitiveness and entertainment, in an effort to uncover advantages in unknown patterns through data, inspired by the revolution sparked by data analysis in the world's premier baseball league (Lewis, 2004). The project aims to use Machine Learning techniques to analyze professional League of Legends drafts and, through them, predict which team will emerge victorious in the matches. Although the outcome of the match is not directly determined by the draft, the network can bring new ideas and discover new paradigms for professional game strategies.

Key-Words: League of Legends, Machine Learning, Shapley Values

Conteúdo

Lista de Figuras

Lista de Tabelas

1 Introdução

1.1 Motivação

Os jogos eletrônicos têm se consolidado como uma das principais formas de entretenimento na sociedade, influenciando diversos aspectos culturais e econômicos. Desde as grandes máquinas de fliperamas, os jogos eletrônicos, muitas vezes citados como videogames, tornaram-se cada vez mais acessíveis, estando atualmente presentes na maioria dos *smartphones* e possivelmente atingindo mais de três bilhões de pessoas em 2024 (Buijsman, 2024).

Dentro desse contexto, o jogo para computadores "*League of Legends*" (LoL)[1], lançado em 2009 pela empresa *Riot Games*, estabeleceu-se como um fenômeno global. Com uma estimativa de 130 milhões de jogadores ativos em janeiro de 2025 (Gill, 2025), o jogo é uma das principais forças de um mercado que arrecadou mais de 220 bilhões de dólares em 2023 e está previsto para alcançar 300 bilhões em 2028 (PwC, 2024), mesmo 15 anos após seu lançamento.

Além de sua enorme comunidade de jogadores casuais, LoL desempenha um papel central no mundo dos esportes eletrônicos, com grandes competições, como o Campeonato Mundial, alcançando picos de 99 milhões de espectadores únicos (Lol Esports, 2018). Tais esportes eletrônicos ganham força em um cenário em que esportes tradicionais não conseguem manter sua base de fãs, crescendo até 10% seus números de audiência, enquanto os de grandes instituições esportivas como a *NBA* (liga profissional de basquete dos Estados Unidos) e a *UEFA* (federação de futebol europeia) diminuem em até 11% (Schudey et al., 2023).

Paralelamente, a área de Inteligência Artificial (IA) vive uma verdadeira revolução desde os resultados surpreendentes obtidos em 2012 pela rede AlexNet na competição *ImageNet Large Scale Visual Recognition Challenge (ILSVRC)* (Krizhevsky, Sutskever e Hinton, 2012). Tais resultados demonstraram o potencial da Aprendizagem Profunda (AP) para abordar problemas complexos. Tal potencial foi explorado em diversas frentes, culminando na proposição da arquitetura *Transformer* por pes-

1. https://www.leagueoflegends.com/pt-br/

quisadores do Google (Vaswani et al., 2017), que resultou em produtos como o ChatGPT [2], DALL-E [3] e diversos outros.

Muitos destes produtos furaram a bolha acadêmica, moldaram o pensamento do público em geral sobre o assunto e impactaram diversas áreas além da pesquisa. Em novembro de 2022, a empresa *OpenAI* lançou o ChatGPT ao público em geral de maneira gratuita (OpenAI, 2022) e, em janeiro de 2023, já havia alcançado 100 milhões de usuários, tornando-se a aplicação de crescimento mais rápido da história (Vaghasiya, 2024). Tal fenômeno impacta a indústria de jogos eletrônicos, sobretudo no desenvolvimento dos mesmos, com IA generativa assumindo um papel de auxílio na criação de roteiros, personagens, modelos 3D e diversos outros componentes.

No contexto competitivo dos esportes eletrônicos (*esports*), a aplicação de técnicas de Aprendizagem de Máquina (AM) pode ser promissora para auxiliar a análise e a predição de resultados de partidas, com base no vasto volume de dados gerados diariamente. Torna-se possível, então, o fornecimento de informações valiosas a jogadores profissionais, treinadores e organizações, além da potencialização da experiência do público, com a geração de análises e enriquecimento da transmissão de eventos.

1.2 Objetivos

O objetivo geral deste estudo é a aplicação de métodos de AM para gerar diferentes modelos para predição de vencedores de partidas de *League of Legends* e a realização de uma avaliação comparativa de seus resultados obtidos. Os modelos serão treinados utilizando apenas informações da etapa de escolha de personagens, chamada de *draft*, em que cada equipe seleciona e exclui personagens para utilizar na partida seguinte. Para tal, será construído um conjunto de dados (*dataset*) que permita a realização de experimentos computacionais utilizando métodos de aprendizado supervisionado, como árvores de decisão, regressão logística e redes neurais.

2. https://chat.openai.com/
3. https://openai.com/index/dall-e/

1.3 Metodologia

A primeira etapa do projeto consiste em entender a viabilidade do estudo, sobretudo em relação a dados: por se tratar de uma propriedade intelectual privada, não há um banco de dados público com informações de partidas profissionais de *League of Legends* realizadas. Portanto, é necessário encontrar uma forma de obter dados suficientes para possibilitar o treinamento de modelos de AM.

Por meio de buscas na internet, é possível encontrar sites especializados na divulgação de estatísticas do jogo que, apesar de não serem ligados à desenvolvedora *Riot Games*, são endossados pela mesma e possuem acesso direto a algumas das bases de dados oficiais. Um desses sites, chamado *Leaguepedia*[4], disponibiliza uma API (*Application Programming Interface*) em que é possível receber as estatísticas de cada partida profissional realizada, incluindo tanto as informações do *draft*, como os personagens escolhidos e jogadores participantes, quanto o que ocorreu durante a partida, como a quantidade de eliminações de cada equipe, objetivos concluídos e, por fim, a equipe que saiu vitoriosa do confronto.

Com essa possibilidade em mente, foi criado um *script* na linguagem *Python* para realizar requisições em série para essa API, de forma a obter os dados de todas as partidas realizadas oficialmente até o fim de 2024, totalizando mais de dez anos de campeonatos profissionais, além de armazená-los localmente em uma única base. Foi escolhido o formato *csv* para tal armazenamento, devido à facilidade de utilização em códigos desenvolvidos.

De posse do *dataset*, foi realizado um estudo exploratório sobre a literatura existente no assunto de *machine learning*, de forma a encontrar quais métodos existentes poderiam ser utilizados para uma tentativa de abordagem ao problema. Para entender que tipos de abordagens seriam utilizadas, foi levada em conta a natureza dos dados: organizados de forma tabular, com números e textos representando seus atributos; a classificação de cada exemplo será feita de forma binária, dizendo qual das duas equipes participantes foi a vencedora. Dessa forma, foi possível levantar os métodos que lidam melhor com essa problemática, listados e discutidos no Capítulo 2.

4. https://lol.fandom.com/wiki/League_of_Legends_Esports_Wiki

Os modelos, portanto, foram treinados sempre com os mesmos dados a cada bateria de experimentos, através da plataforma *Google Colab*[5] e três baterias de experimentos foram realizadas. A primeira buscava validar a capacidade de cada método de entender o que é uma vitória e quais estatísticas a expressam. Para tal, foram usados no treinamento todos os dados das partidas, ou seja, tudo o que aconteceu até o fim da mesma, exceto a classificação de qual time foi o vencedor. Tais informações, porém, eliminam o desafio da predição, uma vez que só existem após o fim da partida, quando já sabemos seu resultado.

Portanto, em seguida, buscou-se avaliar o desempenho de cada modelo quando estes eram treinados somente com os dados do *draft*, ou seja, somente aqueles existentes antes do início da partida. Tais dados englobam somente os campeões, jogadores e times envolvidos no confronto. Para encerrar os experimentos, foi realizado um pré-processamento nos dados, de forma a gerar informações relevantes para o desempenho dos modelos através de engenharia de características (*feature engineering*) e executados os testes finais.

Após a execução dos experimentos, foi utilizada uma abordagem baseada em *Shapley Values* para gerar visualizações da importância de cada atributo para a definição da saída. Isso permite maior entendimento de como cada modelo funciona internamente, bem como quais caminhos são mais relevantes, possibilitando futuros estudos.

Finalmente, foram elaboradas conclusões acerca da viabilidade de cada abordagem para a tarefa descrita, bem como iniciadas discussões sobre futuras possibilidades para se lidar com essa problemática.

1.4 Organização do texto

O presente trabalho está estruturado em capítulos e, além desta introdução, será desenvolvido da seguinte forma:

- Capítulo 2: apresenta os conceitos necessários para o entendimento deste trabalho;

5. https://colab.research.google.com/

- Capítulo 3: aborda os trabalhos relacionados a esta pesquisa;
- Capítulo 4: descreve a configurações dos métodos de aprendizagem de máquina adotados;
- Capítulo 5: detalha os experimentos computacionais e as análises conduzidas; e
- Capítulo 6: apresenta as conclusões, reunindo as considerações finais e sugerindo possibilidades futuras de continuidade desta pesquisa.

2 Conceitos Fundamentais

Este capítulo tem como objetivo apresentar os principais conceitos necessários para o entendimento deste trabalho, notadamente os que dizem respeito a *League of Legends*, à Aprendizagem de Máquina e à Aprendizagem Profunda. Aqueles que já se sentirem confortáveis com tais conceitos podem pular este capítulo sem prejuízo.

2.1 *League of Legends*

O *League of Legends* (LoL) é um jogo eletrônico para computadores composto por dois times de cinco jogadores que se embatem com o objetivo de destruir a base adversária. Tal estilo de jogo é chamado de MOBA (sigla para *Multiplayer Online Battle Arena*) e, de forma a alcançar o objetivo final, as equipes devem eliminar adversários, destruir estruturas inimigas e conquistar objetivos secundários. Cada equipe é associada a um dos lados do mapa, lado azul ou lado vermelho, e, caso um time destrua a estrutura central da base adversária, chamada de *Nexus*, a partida acaba, com o time destruidor como vencedor. A Figura 1 apresenta a tela de um jogador dentro de uma partida de LoL.

No mapa do jogo, mostrado na Figura 2, há três grandes caminhos, chamados de rotas, e os jogadores usualmente se dividem entre eles, com um se dirigindo para a rota do topo, um para a rota do meio e dois para a rota inferior. O jogador restante irá para a selva, que consiste em regiões entre as três rotas em que surgirão pequenos monstros, que concedem pequenas recompensas a quem os abater. Similarmente, os jogadores das rotas podem abater pequenos soldados que surgem simetricamente para ambas as equipes e também dão pequenas recompensas, chamados de *minions*.

Figura 1: Uma partida de LoL. Em azul, os aliados e, em vermelho, os inimigos. Na barra inferior há as habilidades e itens do jogador, enquanto no canto inferior direito há o mini-mapa do jogo.

Ao longo do jogo, são inseridos monstros mais fortes, mais difíceis de serem derrotados e que dão bônus maiores aos membros da equipe que os derrotarem. Tais objetivos requerem a coordenação conjunta do time para que seja possível derrotar o monstro ao mesmo tempo em que a equipe inimiga tenta impedir que isso aconteça ou tenta até mesmo roubar o bônus para si. Essas regras, somadas ao fato de que abater o personagem de um jogador inimigo garante grandes recompensas, implicam que os jogadores estejam a todo o tempo interagindo entre si, seja para lutar, atrapalhar o adversário ou auxiliar seus aliados.

Figura 2: Simplificação do mapa de LoL, chamado de *Summoner's Rift*, com as três rotas e os monstros mais fortes destacados (extraído de www.riotgames.com).

Cada jogador utiliza um personagem, chamado de campeão, que usualmente possui cinco habilidades únicas e que possui atributos com valores únicos, como “Dano de Ataque”, “Poder de Habilidade”, “Velocidade de Movimento”, entre outros, e, atualmente, existem 170 campeões disponíveis, completamente diferentes uns dos outros. A Figura 3 mostra a tela do jogador no momento em que ele deve realizar a escolha de seu personagem.

Tal variedade se alia às regras projetadas para a interação entre os jogadores e gera uma complexidade tão grande que permite que nenhuma partida seja igual a alguma outra, assim como cada combinação de campeões trará características específicas e diferentes de qualquer outra combinação.

Figura 3: Tela de seleção de campeão.

2.1.1 *Draft*

League of Legends, tal qual esportes como futebol e basquete, foi criado com o objetivo de diversão e ainda pode ser praticado para tal, mas houve um crescimento orgânico da competitividade de seus jogadores, a ponto destes se organizarem em instituições, campeonatos serem organizados ao redor do mundo e, hoje, haver um ambiente profissional em que jogadores vivem em função de competir como atletas de LoL.

Uma vez que há atletas nesse meio, tornou-se também necessária a adoção de uma equipe maior de profissionais e, atualmente, a grande maioria dos times pos-

sui uma comissão técnica que pode incluir, entre outros cargos, treinadores, *scouts* e dirigentes, e acompanhamento médico diário de fisioterapeutas, nutricionistas e psicólogos.

No cenário profissional, imediatamente antes de cada partida, há um preparativo em que jogadores e treinadores devem decidir quais campeões serão utilizados ou banidos da partida, o que é chamado de *draft*. O *draft* é realizado da seguinte forma:

- Alternadamente, cada equipe bane um personagem, até que se atinja um limite de três banimentos para cada equipe. Os campeões banidos não poderão ser escolhidos por nenhuma das equipes;
- Novamente de forma alternada, as equipes escolhem os campeões que utilizarão, até que sejam feitas três escolhas para cada uma. Um personagem escolhido por uma equipe não poderá ser selecionado pela equipe adversária;
- Uma segunda rodada de banimentos alternados ocorre, desta vez com dois banimentos para cada equipe;
- A última rodada de escolhas alternadas é feita, com as duas últimas escolhas de cada equipe.

Esse processo possui extrema importância, pois os campeões, itens e até o mapa do jogo sofrem alterações constantes pela empresa desenvolvedora, implicando que um grupo de campeões se encontre mais forte do que outros, sendo então necessário criar estratégias para obtê-los, ao mesmo tempo que evita seus adversários de fazerem o mesmo. Além disso, as fraquezas e forças de cada campeão podem ser complementares ou antagônicas a outros escolhidos, sendo importante, portanto, levar em consideração a composição das escolhas de ambas as equipes.

É importante frisar que não há uma vantagem direta como no jogo *Pokémon*, em que "água vence fogo", entre os personagens. Apesar disso, os campeões possuem atributos, como pontos de vida (conhecidos como *Health Points* - HP) e armadura, além de habilidades que concedem mobilidade, ou um alcance de ataque à distância, e diversas outras possibilidades que diferem um campeão do outro. Por

consequência, certos campeões podem ser melhores em determinados momentos, mas piores em outros. Portanto, a vantagem que cada time pode garantir no *draft* é extremamente situacional, a depender das escolhas de cada lado e da forma como cada equipe está acostumada e melhor treinada para jogar.

Na Figura 4, jogadores e técnico discutem qual será a última escolha de seu time, simbolizada pelo retângulo cinza no canto inferior direito. Os outros retângulos, com imagens de personagens, simbolizam as escolhas já confirmadas, discriminando o nome do respectivo jogador embaixo, enquanto os personagens com retratos nos quadrados menores são aqueles banidos e, portanto, não utilizáveis para aquela partida. Também estão informadas na imagem a rodada atual do torneio e as equipes que se enfrentam, na parte central.

Figura 4: Transmissão de um *draft* do campeonato mundial de *League of Legends* (extraído de www.lolesports.com).

2.2 Inteligência Artificial e Aprendizagem de Máquina

Desde a Primeira Revolução Industrial, o conceito de máquinas substituindo seres humanos gera medo em grande parte da população mundial, tendo seu mais recente capítulo com a popularização do ChatGPT da OpenAI. Apesar disso, a IA nada mais é do que o conjunto das pesquisas, métodos, tecnologias e aplicações para simulação, extensão e expansão da inteligência humana (Jiang et al., 2022), servindo como um termo guarda-chuva para diversas vertentes.

Uma dessas vertentes é a Aprendizagem de Máquina - AM que estuda programas que aprendem segundo uma experiência E, com respeito a alguma classe de

tarefas T e métrica de avaliação P, se a sua performance em tarefas em T, conforme medido por P, melhora com experiência E (Mitchell, 1997). Esse campo se baseia no uso de modelos matemáticos que aprendem a aproximar uma determinada função desconhecida a partir dos dados fornecidos. Esses modelos possuem parâmetros internos cujos valores são modificados ao longo de um processo de treinamento, de forma a otimizar um determinado objetivo, que pode corresponder, por exemplo, a fazer as saídas de tais modelos se aproximarem ao máximo das saídas esperadas conhecidas.

Dentro do escopo de AM, há também uma diversidade de estratégias de aprendizado, a depender da natureza da tarefa e dos dados disponíveis. Uma dessas estratégias lida com um conjunto de dados devidamente rotulados, isto é, com o atributo de saída esperada conhecido (*ground-truth*), enquanto outra, por exemplo, lida com dados que não possuem rótulos (ou classes) claramente definidos.

Mais especificamente, é possível definir pelo menos três estratégias de aprendizado distintas no âmbito de AM. A primeira é o Aprendizado por Reforço, em que se dispõe de um ambiente no qual um agente, sem conhecimento prévio, realiza ações e é recompensado positiva ou negativamente por isso, para fins de atingir um determinado objetivo. Ao longo do processo de treinamento, o agente aprende a tomar as melhores decisões de acordo com cada situação vivida, visando à maximização das recompensas obtidas. Um exemplo em que tal estratégia foi adotada é o trabalho realizado pela empresa OpenAI, em que dois agentes aprendem a jogar rodadas de "esconde-esconde" autonomamente a partir das interações com o ambiente e as respectivas recompensas obtidas (Baker et al., 2020).

Por sua vez, na estratégia de Aprendizado Não-Supervisionado, deseja-se encontrar relações implícitas existentes nos dados, tais como classificações previamente não existentes. Alguns exemplos são a análise de avaliações de usuário sobre hotéis para predição e segmentação de mercado (Ahani et al., 2019).

Finalmente, há a estratégia de Aprendizado Supervisionado, que será adotada neste trabalho. Nessa, os exemplos precisam estar rotulados com a saída desejada, de forma que o modelo possa comparar sua saída predita com tal rótulo, entender se houve acerto ou erro e ajustar seus parâmetros de acordo com uma função ma-

temática predefinida, chamada de função de custo, e a possível distância entre os valores. Uma vez que serão utilizadas partidas profissionais de LoL previamente ocorridas para treinar os modelos desenvolvidos, com a indicação da equipe vencedora servindo como rótulo para a classificação, tal estratégia se mostra como a mais recomendada a ser utilizada.

No contexto deste trabalho, o desafio será tratado como um problema de classificação, de modo que cada exemplo está associado a uma classe e os modelos construídos precisam aprender a separar corretamente os exemplos conforme sua classe. Nas próximas subseções, serão descritas as técnicas de aprendizado supervisionado utilizadas neste trabalho.

2.3 Árvore de Decisão

Um dos primeiros métodos de AM, as Árvores de Decisão são um tipo de modelo já maduro, utilizado há mais de 50 anos em diferentes disciplinas, tais como negócios (Magee, 1964), biologia (Kingsford e Salzberg, 2008) e computação (Quinlan, 1990), que têm como estratégia transformar em decisões os valores que os atributos do problema podem assumir, gerando caminhos para cada possibilidade ou agrupamentos destas, até que se chegue a um resultado final, como exemplificado na Figura 5 que especifica se é aconselhável ou não jogar futebol de acordo com atributos de condições climáticas do dia.

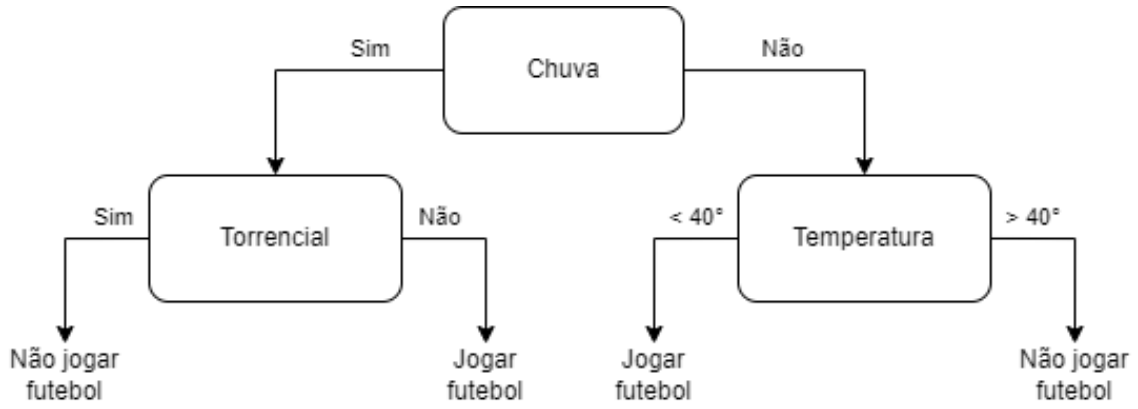

Figura 5: Exemplo de uma árvore de decisão.

Apesar de os modelos mais populares atualmente se basearem em redes neurais, outro tipo de técnica de AM, as árvores de decisão costumam gerar melhores

resultados para conjuntos de dados tabulares (Grinsztajn, Oyallon e Varoquaux, 2022), como os utilizados neste trabalho e, portanto, correspondem a um dos métodos adotados nos experimentos computacionais realizados.

2.3.1 Entropia

Entropia é um termo cunhado no campo da Física que representa a desordem e incerteza de modelos ou ambientes. Apesar de diferir ligeiramente de acordo com o domínio em que está inserida, a entropia costuma se tratar de uma quantidade definida, que quantifica o grau de desordem de determinado estado (Wehrl, 1978), em que 0 é o valor representativo de um estado puro e ordenado. No contexto da teoria da informação, a entropia mede a incerteza em relação a uma fonte, permitindo a quantificação de quanta informação está presente em um determinado sistema (Shannon, 1951).

As árvores de decisão seguirão esta última definição, sendo a entropia (E) utilizada como um critério de seleção de atributos que servirão como nós da árvore, sendo calculada a partir da Equação 2.1 , em que p_i consiste na probabilidade de um exemplo da classe i ser selecionado aleatoriamente.

$$E = -\sum_{i=1}^{n} p_i \log_2(p_i) \tag{2.1}$$

2.3.2 Ganho de Informação

No processo de construção da árvore de decisão, o nó raiz conterá todos os exemplos, portanto terá probabilidade de 100% de ter exemplos de todas as classes e, por consequência, terá valor de entropia 1. Em sequência, para o próximo nó filho a ser criado, é selecionado um atributo do *dataset*, como, por exemplo, nível de escolaridade, são separados os exemplos para cada valor possível desse atributo e é calculada a entropia resultante de tal divisão. Após o cálculo da entropia para cada atributo ainda não explorado do *dataset*, é selecionado como nó filho aquele que fornece o maior ganho de informação, isto é, aquele que leva à maior separação entre os exemplos, calculado por meio da Equação 2.2.

$$Ganho = Entropia_{pai} - Entropia_{filho} \tag{2.2}$$

Tal processo é repetido recursivamente até que se encontre um estado puro de $E = 0$, ou sejam utilizados todos os atributos, ou se tenha atingido um tamanho máximo de árvore, ou alguma outra condição predeterminada, sendo, portanto, decidida a classe de saída através da distribuição dos exemplos do nó-folha construído.

2.4 Florestas Aleatórias

Criadas por Tim Kan Ho (1995), aperfeiçoadas por Leo Breiman (2001) e implementadas de forma prática pela primeira vez por Adele Cutler (2007), as florestas aleatórias (*Random Forests*) são um tipo de algoritmo que constrói diversas árvores de decisão de forma controlada, porém com graus de aleatoriedade, de modo a avaliar a saída de cada uma dessas árvores e convergir em um único resultado, com maior precisão e grau de variabilidade.

O processo se inicia com a seleção de amostras de *bootstrap*, uma técnica que consiste em substituir partes do *dataset* original, de forma a aumentar a quantidade de exemplos disponíveis, sem afetar negativamente qualquer inferência a ser realizada sobre esses (Efron e Tibshirani, 1994). Uma árvore de decisão é criada e treinada para cada amostra de *bootstrap* gerada, considerando que apenas um subconjunto dos atributos, selecionado ao acaso, está disponível para cada nó da árvore, a fim de gerar mais um grau de aleatoriedade.

Em uma amostra de *bootstrap*, espera-se que seja encontrada a maior parte dos exemplos originais, mas não todos eles. Com isso em mente, para prever a classe de um exemplo na etapa de inferência, são utilizadas todas as árvores que não tiveram tal exemplo em sua amostra de treinamento. Por fim, a saída predita pelo algoritmo será a que foi escolhida na maioria das vezes por essas árvores. Caso haja um empate, o resultado será escolhido aleatoriamente.

2.5 *Naive Bayes*

Diferentemente dos métodos anteriores, cujas saídas são uma das classes determinadas pelo problema, o modelo *Naive Bayes* (Bayes Ingênuo) leva em conta o Teorema de Bayes proposto na área de Probabilidade, que descreve a probabilidade condicional de um evento (Berrar, 2018). No contexto de AM, isso significa que é calculada a probabilidade de o exemplo ser da classe previamente mencionada, baseada na probabilidade de outros eventos independentes entre si (Joyce, 2021). Tal técnica é chamada de ingênua, pois presume que os atributos de um exemplo serão sempre independentes entre si, o que nem sempre ocorre, mas, ainda assim, é capaz de obter resultados relevantes para um grande número de problemas (Webb, Keogh e Miikkulainen, 2010).

2.6 *Support Vector Machine*

A técnica de *Support Vector Machines* (SVM) foi criada por pesquisadores da empresa *AT&T* (Boser, Guyon e Vapnik, 1992) e constitui uma abordagem para problemas de classificação binária através da separação linear de classes. Para *datasets* de apenas um atributo e a classe a ser predita, isto é, representados em um espaço bidimensional, é definida uma linha reta separada dos seus exemplos. Em *datasets* com mais atributos, logo representados em espaços mais dimensionais, essa separação pode ser atingida através de um plano ou de um hiperplano. Tal hiperplano é construído a partir da resolução de um problema de otimização, em que se deseja obter um hiperplano separador que maximiza a distância aos exemplos das classes separadas.

Contudo, a maior parte dos problemas de interesse não são linearmente separáveis, o que impõe uma grande limitação à forma tradicional dessa abordagem. Para transpor essa dificuldade, foi proposta a adoção do chamado *Kernel Trick*, que corresponde à aplicação de uma operação de transformação de representação aos dados. Mais especificamente, essas operações, chamadas de funções de *kernel*, são aplicadas sobre os exemplos do problema de forma a representá-los em um espaço mais dimensional em que seja possível encontrar um separador linear, como repre-

sentado na Figura 6. Dessa forma, essa estratégia de *Kernel Trick* foi crucial para o sucesso da técnica de SVM ao longo das últimas décadas, sobretudo nos anos 90 e 2000.

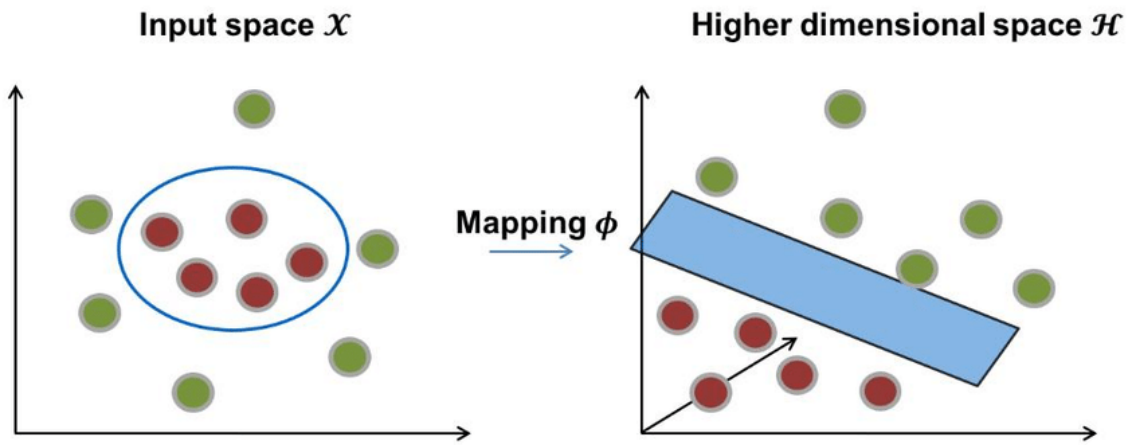

Figura 6: Representação do *Kernel Trick*, com a representação original dos dados à esquerda e a nova representação, em um espaço de maior dimensão, à direita (extraído de Rodríguez-Pérez e Bajorath, 2022).

É possível provar que, para todos os conjuntos de dados sem exemplos idênticos em ambas as classes, há uma função de *kernel* que permitirá uma separação linear (Noble, 2006). Entretanto, nem sempre a melhor solução corresponde a representar os dados em um espaço de alta dimensionalidade, uma vez que o treinamento pode resultar em um modelo muito especializado, tornando a classificação de exemplos não vistos na etapa de treinamento pouco efetiva, correspondendo a um sobreajuste (*overfitting*) ao conjunto de treinamento.

2.7 Regressão Logística

No contexto estatístico, uma regressão de uma variável y sobre uma outra variável x é a observação de qualquer característica da distribuição de probabilidade de y ocorrer condicionalmente a x (Manski, 1991). Nos dias atuais, existem diversos métodos de regressão, sendo um dos mais utilizados a regressão linear, que busca correlacionar uma saída, y, a uma ou mais variáveis independentes, x_i, de forma linear, como diz seu nome.

$$\hat{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_n x_n, \tag{2.3}$$

Seu algoritmo segue a Equação 2.3, em que $\hat{y}$ é o resultado esperado, β_0 é o ponto em que a reta da regressão linear encontra o eixo y e $\beta_i x_i$ corresponde ao valor da variável x_i ponderada por seu coeficiente β_i, para $i = 1, \ldots, n$. Tais coeficientes β_i constituem os parâmetros cujos valores devem ser encontrados pela abordagem e descrevem a contribuição de cada variável para o resultado final.

Apesar de útil para muitos casos, a regressão linear possui uma limitação clara, uma vez que nem todos os problemas são linearmente solucionáveis. Dessa forma, a regressão logística utiliza-se das premissas da regressão linear, porém expressa seu resultado através de duas possíveis categorias de saída, representando a probabilidade estimada de o exemplo ser de uma categoria c_1, em relação à outra categoria c_2 (Stoltzfus, 2011).

Com isso, há a necessidade de transformar a equação de regressão para que sua saída esteja sempre no intervalo $[0, 1]$, de forma a expressar uma probabilidade. Tal transformação é realizada ao representar a equação na escala *logit*, utilizando o logaritmo natural das chances de um exemplo ser de uma categoria, em relação a outra, que corresponde ao *log odds ratio*, expresso na Equação 2.4.

$$\ln(\hat{y}/1 - \hat{y}) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_n x_n. \tag{2.4}$$

Assim, dois problemas são resolvidos: a saída passa a ser representada como uma probabilidade e a equação de definição do modelo deixa de ser linear, expandindo suas possibilidades de uso para casos mais complexos.

É importante ressaltar que a regressão logística, como o próprio nome diz, normalmente não lida com problemas de classificação, pois sua saída é contínua, porém as categorias serão sempre discretas; contudo, é possível utilizar a saída do modelo como a probabilidade p de o exemplo ser de uma classe e o seu complementar, $1 - p$, a probabilidade de o exemplo ser da outra classe.

2.8 Redes Neurais

Inspiradas no cérebro humano, as redes neurais possuem uma arquitetura baseada em nós, chamados de neurônios, que geralmente se dividem em uma camada de entrada, uma ou mais camadas ocultas e uma camada de saída, como representado na Figura 7, respectivamente em vermelho, azul e verde, embora outras formas de estruturação existam.

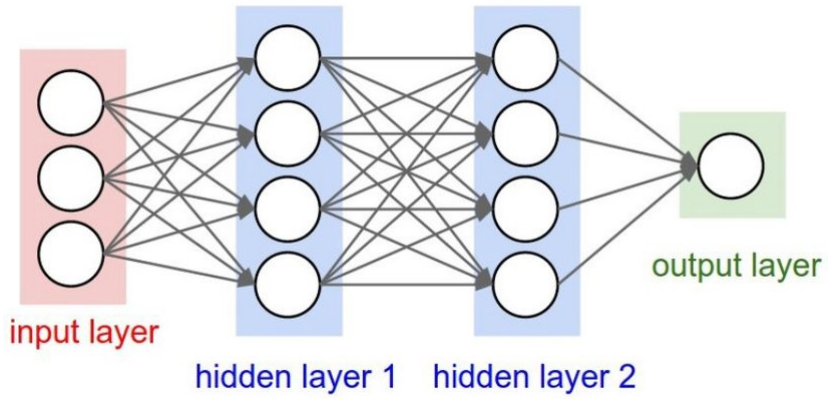

Figura 7: Representação de uma rede neural (extraído de Amherd e Rodriguez, 2021).

A camada de entrada é constituída de espaços reservados para os dados a serem fornecidos à rede, de forma que não há nenhum neurônio e os dados recebidos são repassados para a próxima camada. Já as camadas ocultas possuem os neurônios que recebem as suas entradas, multiplicam-nas pelos respectivos pesos e calculam a sua ativação através da aplicação da função de ativação. As ativações dos neurônios de uma camada c_i são repassadas aos neurônios da camada c_{i+1}. A adoção de múltiplas camadas com funções de ativação não lineares torna possível a abordagem a problemas não linearmente separáveis (Aggarwal, 2023). Por fim, os neurônios da camada de saída são responsáveis por gerar a saída da rede, que pode ser composta de um ou mais valores a depender do problema abordado.

Não há uma função de ativação que seja ideal para todo tipo de problema, portanto algumas funções conhecidas da literatura normalmente são testadas empiricamente na construção das redes, como a ReLU (Schmidt-Hieber, 2020), ilustrada na Figura 8, que transforma valores negativos em 0, ou a Sigmoide (Chen et al., 2024), ilustrada na Figura 9, que transforma todas as entradas no intervalo $[0, 1]$.

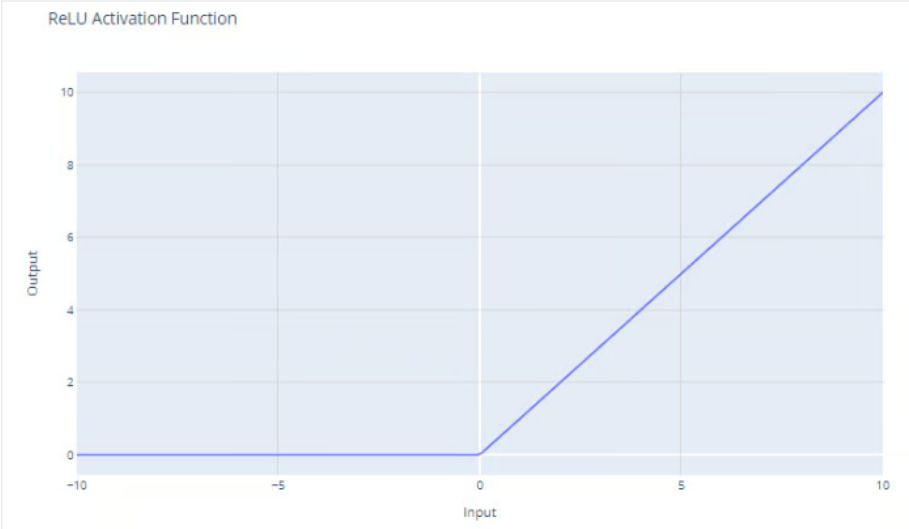

Figura 8: Ilustração da função ReLU (extraído de www.datacamp.com).

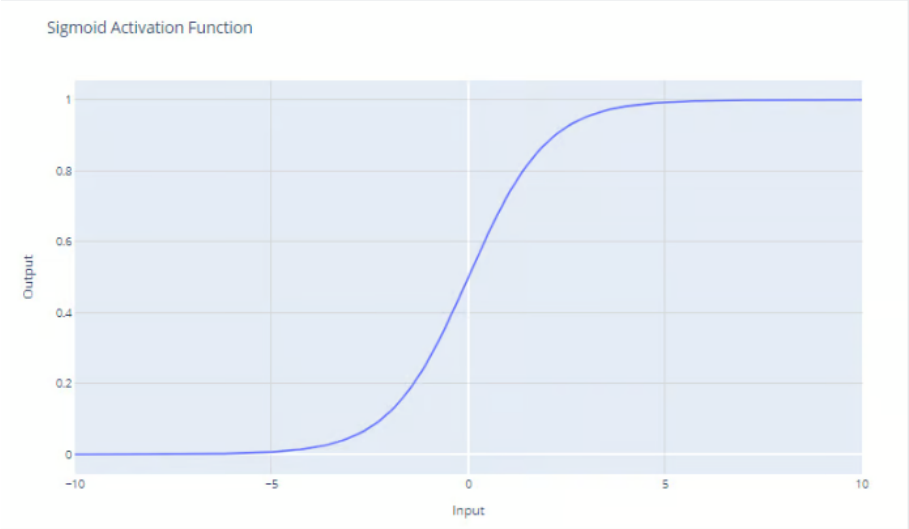

Figura 9: Ilustração da função Sigmoide (extraído de www.datacamp.com).

Além disso, normalmente os pesos da rede são inicializados segundo alguma distribuição de interesse, tais como a *Xavier-Glorot Normal* e a *Glorot Uniform* (Aggarwal, 2023).

Durante o processo de treinamento, os exemplos do conjunto de treino são fornecidos à rede, um a um, a fim de se obter a sua estimativa. Por meio da aplicação da função de custo, as estimativas da rede são avaliadas a partir da comparação com as saídas esperadas. Tal função de custo é, então, otimizada por meio do algoritmo de Gradiente Descendente que, associado ao algoritmo de *Backpropagation*, determinará a modificação a ser feita em cada um dos parâmetros (pesos) da rede

(Aggarwal, 2023). Quando todos os exemplos de treino são fornecidos à rede, tem-se então que se completou uma época. Esse processo é repetido até que uma determinada condição de parada tenha sido atingida, a saber: convergência a um mínimo local, número máximo de épocas atingido, entre outros.

Dentre outras nomenclaturas, as redes neurais podem ser separadas entre profundas ou rasas e densas ou esparsas. Uma rede é considerada profunda se possuir três ou mais camadas ocultas e, caso contrário, é dita ser rasa. Por sua vez, uma rede neural é considerada densa caso todos os neurônios de uma camada possuam conexões com todos os neurônios da camada posterior. Essa combinação de muitas camadas com alta densidade gera uma complexidade computacional bastante onerosa devido ao efeito combinatório refletido na quantidade de parâmetros da rede, que precisam ser calculados a cada iteração. Tal quantidade de parâmetros, porém, pode ser necessária para a resolução de problemas mais complexos, portanto faz-se necessária uma avaliação minuciosa da arquitetura a ser criada para cada abordagem.

2.9 Engenharia de Características

Normalmente, em um projeto de AM, utilizar apenas os atributos (características) existentes originalmente no *dataset* não costuma levar a um nível de desempenho relevante para uma ampla classe de problemas. Dessa forma, é muito comum utilizar os atributos existentes no *dataset*, junto com conhecimento específico do domínio do problema, a fim de se criarem novos atributos com os quais o modelo poderá abordar melhor o problema. Essa estratégia é chamada de Engenharia de Características (*Feature Engineering*) e foi utilizada de mais de uma forma neste projeto, como será explicado no Capítulo 5.

3 Trabalhos Relacionados

Neste capítulo serão apresentados estudos semelhantes ou relevantes ao atual, de forma a comparar descobertas, entender o que já foi feito e encontrar novos caminhos previamente não contemplados.

3.1 Aprendizagem de Máquina em Jogos Eletrônicos

Outros autores já realizaram estudos acerca do uso de AM para a predição de resultados de jogos eletrônicos. Shen (2022) buscou prever os resultados de partidas amadoras de *League of Legends* utilizando os dados dos dez minutos iniciais de cada uma, consistindo em uma abordagem diferente daquela adotada neste trabalho. O autor utilizou alguns métodos em comum a este trabalho, como SVM e florestas aleatórias, além de outros diferentes, como *Gradient Boosting* e KNN (*K-Nearest Neighbors*), alcançando uma acurácia de $72,68\%$.

Akhmedov e Phan (2021), por sua vez, buscaram prever o resultado de partidas de *Dota 2*[6], um *MOBA* semelhante a *League of Legends*, porém utilizando dados em tempo real, isto é, enquanto as partidas aconteciam. Os autores adotaram as técnicas de regressão linear e redes neurais nos experimentos conduzidos, obtendo resultados de até 93% de acurácia.

Finalmente, Kinkade e Lim (2015) também utilizaram técnicas de AM para analisar *Dota 2*, buscando prever o resultado de partidas amadoras utilizando apenas as escolhas de personagens. Os autores usaram apenas modelos de regressão logística e florestas aleatórias, alcançando uma acurácia de 73%, mas discorreram sobre métodos de engenharia de características que foram utilizados neste trabalho e serão detalhados a seguir.

6. https://www.dota2.com/home

3.2 Métodos de Engenharia de Características

3.2.1 Sinergia

Uma vez que uma partida possui 10 campeões escolhidos e apenas uma equipe vencedora, é possível inferir a porcentagem de vitórias de um determinado campeão em relação a quantas partidas ele esteve presente. Este dado, por si só, tende a não agregar ao desempenho do modelo, mas pode ser utilizado para gerar características derivadas mais complexas. Kinkade e Lim (2015) sugerem a criação de uma nova característica chamada sinergia, que é calculada para cada equipe, baseada em pares de campeões i e j presentes na mesma, sendo V_{ij} a porcentagem de vitórias dado o total de partidas em que ambos foram selecionados conjuntamente pela equipe vencedora. Assim, se e_1 e e_2 são as equipes de uma partida, a sinergia para a equipe e_1 é calculada como na Equação 3.1, levando em consideração as partidas anteriores para o cálculo da porcentagem de vitórias.

$$S_{e_1} = \sum_{i \in e_1} \sum_{j \in e_1, i \neq j} V_{ij} \tag{3.1}$$

em que i e j são dois campeões disponíveis no jogo.

A mesma Equação 3.1 é também aplicada à equipe e_2, de forma a adicionar ao *dataset* a característica $S = S_{e_2} - S_{e_1}$ para cada partida. Com isso, é possível que o modelo, ao realizar as predições, considere quais campeões desempenham melhor em conjunto, gerando um cenário em que o todo é maior que a soma de suas partes e quais são desencorajados de estarem na mesma equipe, como demonstrado pelo estudo de Kinkade e Lim (2015).

Utilizando esse raciocínio como premissa, foi calculado neste estudo, além da característica de sinergia dos campeões, também um atributo de sinergia para os jogadores, utilizando a mesma metodologia. Além disso, espera-se um comportamento de análise semelhante com os jogadores, visto que o fator humano é uma incógnita em muitos ambientes, e duas pessoas individualmente boas podem não conviver harmonicamente, tanto dentro quanto fora do jogo, gerando resultados variados para pares diferentes.

3.2.2 *Countering*

Countering é um termo comum no meio dos jogos eletrônicos e representa quando um personagem é particularmente bom contra outro, seja por suas habilidades, atributos ou pela forma com que ambos interagem dentro do jogo. Para incorporar tal conceito, o estudo de Kinkade e Lim (2015) também propõe cálculos com base na taxa de vitórias, mas desta vez quando os campeões i e j estiverem em lados opostos da partida. Com as mesmas equipes e_1 e e_2, o valor de *countering* é calculado através da Equação 3.2.

$$C_{e_1} = \sum_{i \in e_1} \sum_{j \in e_2} C_{ij} \tag{3.2}$$

em que C_{ij} é a taxa de vitórias do campeão i sobre o campeão j, ou seja, a porcentagem de vezes em que i ganhou de j, estando em equipes opostas.

Analogamente à característica anterior, essa abordagem também será expandida para os jogadores. Assim, serão adicionados a todas as partidas presentes no *dataset* os dois novos atributos de *countering* calculados, sendo um para campeões, futuramente mencionado apenas como *Countering* e outro para jogadores, que será mencionado como *PlayerCountering*.

4 Métodos e *Dataset*

Este capítulo aborda as escolhas e os detalhes de implementação dos métodos de AM adotados neste trabalho, assim como as especificidades do *dataset* utilizado.

4.1 Métodos Escolhidos

A literatura acerca do tema de AM possui uma gama de métodos e abordagens que diferem na forma de encarar um problema e, por consequência, encontram diferentes resultados a depender do escopo em que são utilizados. Devido à abordagem exploratória do estudo, foram selecionados cinco métodos muito utilizados para lidar com dados tabulares. Foram utilizados modelos de árvores de decisão, florestas aleatórias, *Naive Bayes*, *Support Vector Machines* e regressão logística, implementados através da biblioteca em Python *scikit-learn*[7], que disponibiliza funções para a criação, treinamento e uso de modelos conhecidos na literatura.

Há configurações necessárias para alguns dos modelos: para as árvores e florestas, não há um máximo de nós, o que implica que novos nós serão formados até que todas as folhas do modelo sejam puras, ou seja, não exista mais nenhuma decisão a ser tomada; o modelo de *Naive Bayes* assumirá que os valores dos atributos sigam uma distribuição normal e, por fim, a regressão logística terá um máximo de 1000 iterações. Tais configurações foram encontradas por meio de um processo empírico e consideradas ideais, de forma a balancear o desempenho dos modelos e a viabilidade de execução no ambiente computacional mencionado.

Além disso, devido à sua grande relevância na atualidade, foram também adotados modelos baseados em redes neurais. A biblioteca *Keras* [8] foi utilizada para a criação das camadas de cada rede, enquanto a escolha da quantidade de camadas, de neurônios e épocas, bem como a escolha das funções de ativação e custo, ficaram a cargo do autor deste texto. Foram implementadas três redes, com, respectivamente, uma, três e seis camadas ocultas. Nas camadas ocultas, a primeira rede tem oito neurônios em sua única camada, a segunda rede tem 64 em todas as camadas e a terceira inicia com 64 neurônios, duplicando a quantidade por camada até chegar

7. https://scikit-learn.org/stable/
8. https://keras.io/

a 256 e depois dividindo pela metade até voltar a 64. Devido à natureza binária do problema, a função de ativação Sigmoide é utilizada em todas as camadas das redes menores, mas, uma vez que seu desempenho piora com redes maiores (Oostwal, Straat e Biehl, 2021), é usada a função ReLU, mantendo a função Sigmoide apenas para a camada de saída. A Figura 10 ilustra a arquitetura da rede neural rasa, com as demais arquiteturas seguindo a mesma forma, porém variando no número de neurônios e camadas.

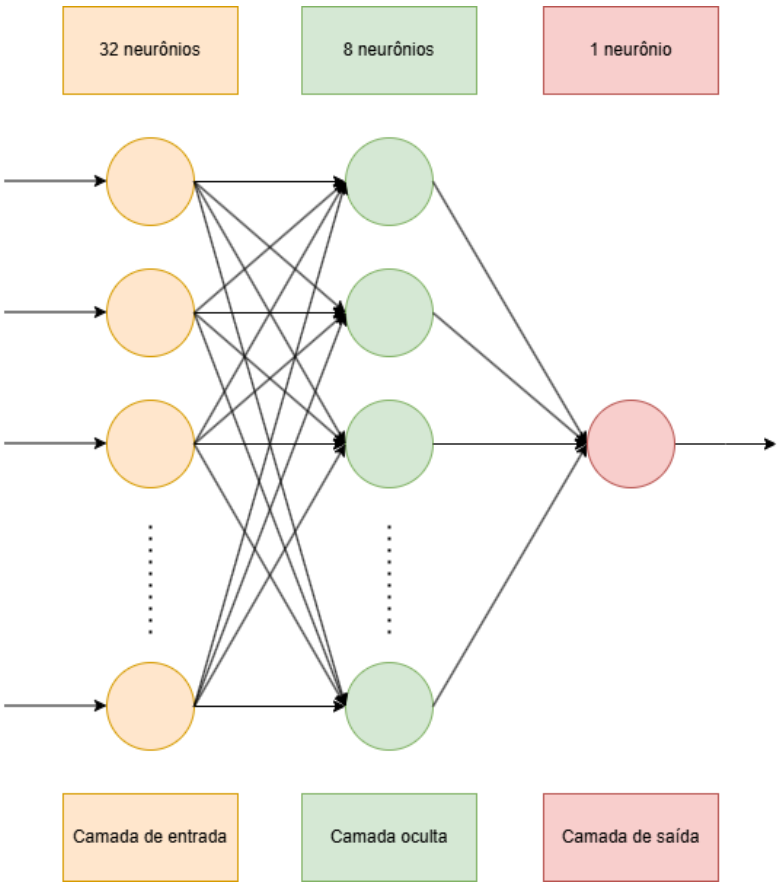

Figura 10: Ilustração da rede neural rasa (NN1)

Através de testes, verificou-se que a função de custo *Binary Cross Entropy* (Ruby e Yendapalli, 2020) gerou melhores resultados e, portanto, foi adotada em todos os modelos, bem como o otimizador Adam, responsável pela otimização da função de custo, que representa o estado-da-arte devido à sua eficiência e eficácia (Kingma e Ba, 2017). Nas tabelas de resultados a serem apresentadas no Capítulo 5, as redes neurais são representadas por NN1 (*Neural Network* 1), NN2 e NN3 de acordo com seu respectivo tamanho.

4.2 *Dataset* adotado

Os dados utilizados no projeto foram coletados por meio de uma API (*Application Programming Interface*) disponibilizada pelo site *Leaguepedia*[9], que age como um centralizador de diversas fontes sobre LoL, sendo muitas vezes um *proxy* do banco de dados oficial da Riot Games. Foi construído um script em *Python* realizar requisições em sequência à API, extraindo um conjunto de dados que contém informações de 96 mil partidas realizadas até o fim de 2024, entre elas dados do *draft*, da própria partida e o rótulo principal ao nosso escopo: o time vencedor.

Esses dados foram agrupados em um arquivo no formato *csv*, com os atributos de *draft*, da partida e do vencedor como colunas (atributos) e cada partida como uma linha (exemplo). As classes definidas para o problema correspondem às equipes vencedoras de cada partida, simbolizadas como 0, caso o lado azul seja vencedor, e 1, caso o lado vermelho seja vencedor. É importante que cada classe simbolize um dos lados, além de apenas ditar o vencedor, uma vez que o mapa de *LoL* não é exatamente simétrico e, portanto, pode implicar em alguma vantagem para um dos lados. Apesar disso, é possível afirmar que o *dataset* é balanceado, uma vez que contém 53% dos exemplos com o lado azul sendo vencedor, com o restante tendo o lado vermelho como vitorioso.

Os atributos foram divididos em dois blocos, sendo o primeiro utilizado apenas para fins de validação e composto por todos os dados referentes à partida, como a quantidade de mortes de cada jogador, quantos monstros cada equipe derrotou e a duração do confronto. O segundo bloco é um subconjunto do primeiro e representa o problema real, compreendendo apenas os atributos referentes ao *draft*.

O segundo bloco é composto por 32 atributos, dos quais 20 são os campeões - os cinco banidos e os cinco escolhidos por cada equipe -, o nome dos dez jogadores participantes do jogo e o nome de cada equipe. Muitos dos atributos foram obtidos de forma textual, como, por exemplo, o nome dos jogadores e campeões. Portanto, foi realizada uma transformação desses dados, substituindo-os por uma lista numérica, ordenada de acordo com a primeira aparição de cada exemplo no *dataset*.

9. https://lol.fandom.com/wiki/League_of_Legends_Esports_Wiki

5 Experimentos Computacionais

Neste capítulo, são reunidas e expostas as informações referentes aos experimentos computacionais realizados, apresentando os dados do ambiente computacional, métricas, resultados obtidos e suas análises.

5.1 Ambiente computacional

Para o desenvolvimento dos experimentos, foi utilizada a linguagem de programação Python, na versão 3.9.10[10]. Os dados foram obtidos da API do site *Leaguepedia* através da biblioteca *mwrogue*, disponibilizada pelo próprio site, na versão 0.1.5[11], tratados com a biblioteca *pandas*, na versão 2.1.0[12] e armazenados com o sistema MySQL, na versão *Community* 8.0.22[13]. Foram utilizadas duas bibliotecas de Python para a criação e treinamento dos modelos de AM: *scikit-learn*, na versão 1.3.1[14] e *Keras*, na versão 2.9.0[15]. Por fim, os experimentos foram executados na CPU de uma instância com 12 gigabytes de memória RAM, através do ambiente em nuvem disponibilizado na plataforma *Google Colab*[16].

Os códigos desenvolvidos nesta pesquisa, assim como o resultado dos experimentos, estão disponíveis publicamente em um repositório criado no GitHub[17].

5.2 *Baseline* e separação do *dataset*

Em todos os experimentos, foi definida uma métrica-base que representa a porcentagem mínima de acurácia aceitável para que um experimento seja considerado bem-sucedido. Analisando-se o *dataset*, encontra-se que, em 53% dos exemplos, a equipe que jogou do lado azul, representada pelo valor 0, foi vitoriosa e, nos demais casos, a equipe que jogou do lado vermelho, denotada pelo valor 1, logrou vitória.

10. https://docs.python.org/release/3.9.10/
11. https://mwrogue.readthedocs.io/en/latest/index.html
12. https://pandas.pydata.org/
13. https://www.mysql.com/products/community/
14. https://scikit-learn.org/stable/
15. https://keras.io/api/
16. https://research.google.com/colaboratory/faq.html
17. https://github.com/lagega/ml-draft-prediction

Portanto, espera-se que qualquer modelo treinado deva possuir uma acurácia maior do que um modelo não-treinado que indica sempre que a equipe vermelha foi vitoriosa, isto é, que fornece uma saída 0 para todos os exemplos apresentados. Assim, define-se que o *baseline* dos experimentos é uma acurácia de 53%.

Os experimentos foram conduzidos separando-se o *dataset* em partes menores para diferentes objetivos. Assim, foi adotada a estratégia *hold-out*, reservando-se 70% dos exemplos para a etapa de treinamento e 30% para a etapa de teste. Todos os modelos receberão os mesmos conjuntos de exemplos, de forma a realizar uma comparação válida entre seus resultados.

5.3 Métricas utilizadas

Para o problema abordado, a saída de um modelo está correta caso a classe predita para um exemplo seja igual à sua classe real. É possível pensar em cada classe como uma resposta para "o time vermelho será vencedor da partida?", sendo 1 o caso positivo e 0 o caso negativo, com o time azul sendo o vencedor predito. Assim, há quatro possibilidades para o resultado de uma predição:

- Verdadeiro Positivo (*True Positive* - TP) - O modelo prevê o time vermelho como vencedor e o time vermelho é de fato o vencedor;
- Falso Positivo (*False Positive* - FP) - O modelo prevê o time vermelho como vencedor e o time azul é o vencedor;
- Falso Negativo (*False Negative* - FN) - O modelo prevê o time azul como vencedor e o time vermelho é o vencedor;
- Verdadeiro Negativo (*True Negative* - TN) - O modelo prevê o time azul como vencedor e o time azul é de fato o vencedor;

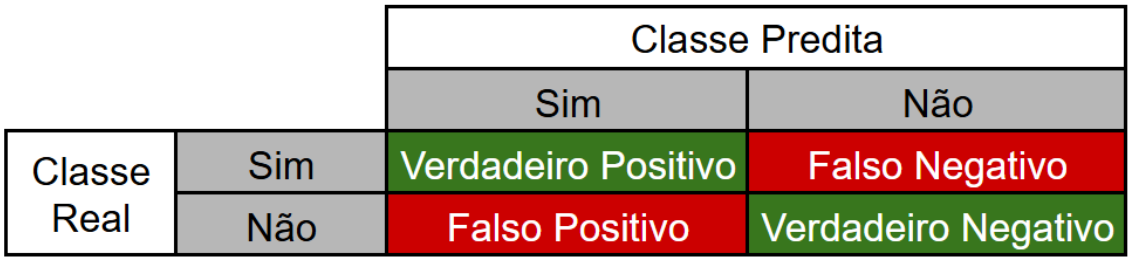

Figura 11: Exemplo de uma matriz de confusão

Com esses casos em mente, é possível criar a chamada matriz de confusão para cada modelo, como exemplificado na Figura 11. A partir da mesma, é possível alcançar as duas métricas que serão utilizadas para entender os desempenhos das abordagens utilizadas ao longo dos experimentos: a Acurácia A de um modelo representa a porcentagem das vezes em que a saída do mesmo esteve correta, conforme a Equação 5.1, enquanto a Precisão P indica a porcentagem das vezes em que o modelo esteve correto quando previu um caso positivo, como mostra a Equação 5.2.

$$A = \frac{TP + TN}{TP + TN + FP + FN} \tag{5.1}$$

$$P = \frac{TP}{TP + FP} \tag{5.2}$$

5.4 Resultados

5.4.1 Experimentos Preliminares

Como um primeiro experimento, foram utilizados todos os dados da partida, incluindo aqueles que designam informações que aconteceram durante a própria realização da partida, tais como a quantidade de abates de cada jogador, o ouro obtido por cada equipe, a quantidade de monstros abatidos, etc. Tal experimentação teve como objetivo entender a viabilidade dos métodos selecionados para trabalhar com o problema: caso os modelos obtidos tivessem um mau desempenho mesmo sendo treinados com todos os dados, incluindo aqueles do decorrer da partida, teriam um desempenho ainda pior caso fossem usados apenas os dados de *draft*, que é o real desafio em questão.

Os resultados dos experimentos preliminares estão expostos na Tabela 1, em que é possível observar que todos os modelos obtiveram valores de acurácia e precisão maiores ou iguais a 97%. Isso faz sentido, na medida em que foram utilizados atributos cujos valores possuem uma forte correlação com a indicação da equipe vencedora. Esses resultados mostraram que os métodos possuem potencial para serem adotados na pesquisa, uma vez que o pior resultado foi de 97% de acurácia, obtido pela rede neural rasa (NN1). No geral, as redes neurais tiveram resultados piores que a grande maioria dos modelos, o que era esperado, visto que, na literatura, é bem conhecido o fato de que tais métodos obtêm resultados piores para conjuntos de dados tabulares em relação a, por exemplo, métodos baseados em árvores (Grinsztajn, Oyallon e Varoquaux, 2022).

	DT	RF	NB	SVM	LogR	NN1	NN2	NN3
Acurácia	99,4%	99,7%	98,4%	98%	99,3%	97%	98%	98,4%
Precisão	99,4%	99,7%	97,9%	97,7%	99,1%	98,4%	98,7%	98,2%

Tabela 1: Resultados dos experimentos preliminares com todos os dados das partidas,

Em seguida, foi realizada uma segunda bateria de experimentos, utilizando-se somente os dados iniciais de *draft*, a saber: campeões banidos e escolhidos, jogadores e equipes participantes da partida. Cada modelo recebeu os mesmos conjuntos de treinamento e de validação e gerou resultados de acordo com a Tabela 2. Tal modelagem, considerando somente os dados de *draft* para prever os resultados das partidas, representa a real problemática a ser abordada nesta pesquisa.

	DT	RF	NB	SVM	LogR	NN1	NN2	NN3
Acurácia	52,3%	57,9%	54,3%	55,3%	54,9%	53,7%	54,7%	54,9%
Precisão	49,1%	57,1%	51,6%	57,8%	52,9%	52,3%	53,2%	53,1%

Tabela 2: Resultados dos experimentos preliminares somente considerando dados de *draft*.

Como exposto na Tabela 2, nenhum dos modelos foi capaz de se distanciar do *baseline* adotado apenas considerando esses dados de *draft*, o que atesta a complexidade do problema abordado e aponta ao fato de que abordagens triviais tendem a levar a maus resultados. Tornam-se necessárias, portanto, a adoção de técnicas que pré-processam esses dados, de forma a gerar atributos novos que sejam relevantes

para a classificação feita pelos modelos, que os permitam ter maior discernimento entre o que é importante ou não para a vitória em uma partida.

5.4.2 Experimentos com Engenharia de Características

Dessa forma, foi adotada a estratégia de engenharia de características (*feature engineering*), a fim de gerar os novos atributos de sinergia e *countering*, descritos nas Seções 3.2.1 e 3.2.2. Esses novos atributos foram então adicionados a todos os exemplos do *dataset*. Uma nova leva de experimentos foi realizada, utilizando a mesma metodologia anterior, consistida dos métodos previamente citados e com conjuntos de dados idênticos para todos.

Os resultados obtidos estão expostos na Tabela 3. A maioria dos modelos alcança desempenhos significativamente melhores do que aqueles da Tabela 2, com acurácias até 25 pontos percentuais maiores. Dessa forma, os modelos não somente superaram o *baseline* estabelecido, como também obtiveram uma taxa de acerto suficientemente relevante para que se obtenham consistentemente resultados corretos.

	DT	RF	NB	SVM	LogR	NN1	NN2	NN3
Acurácia	$76,8\%$	$82,9\%$	$82,6\%$	$78,2\%$	$83,4\%$	55%	$81,6\%$	$81,8\%$
Precisão	$75,3\%$	$82,6\%$	$81,5\%$	$73,8\%$	$82,9\%$	$53,7\%$	$76,5\%$	$77,3\%$

Tabela 3: Resultados dos experimentos utilizando engenharia de características.

5.5 Análise dos Resultados

A primeira observação a ser feita em relação aos resultados vem do desempenho das redes neurais: apesar de seu amplo uso nos dias atuais, a sequência de experimentos indica que estas redes não são tão competitivas em relação a métodos como florestas aleatórias e regressão logística, desde os experimentos preliminares com todos os dados das partidas até o cenário final contemplando engenharia de características. Apesar de produtos que utilizam redes neurais possuírem uma quantidade de neurônios e camadas em escalas muito maiores, tal resultado permite entender que elas não serão a melhor escolha em todos os casos possíveis, sobretudo aqueles que envolverem dados estruturados em tabelas ou em pouca quantidade.

Outro ponto a ser discutido é o baixo desempenho dos modelos baseados em SVM e redes neurais rasas. Para os modelos baseados em SVM, é possível pensar na falta de linearidade dos dados, dificultando, assim, a regulagem dos hiperparâmetros utilizados pelos modelos e afetando a classificação dos dados de validação. Além disso, a função *kernel* utilizada pode não ser recomendada para os dados trabalhados, transformando-os de forma não otimizada. Para este trabalho, não foi possível realizar explorações acerca de outras funções devido ao alto custo computacional deste processo e da baixa customização disponibilizada nativamente pela biblioteca utilizada, mas é possivelmente um caminho para o melhor desempenho deste tipo de modelo.

Por sua vez, as redes neurais rasas também sofreram com a alta dimensionalidade dos atributos, já que possuem poucos neurônios e, portanto, poucos parâmetros para interpretá-los. Dessa forma, cada neurônio precisa levar em consideração mais de um atributo, o que causa uma falta de especialização dos mesmos, levando a rede a não conseguir identificar as diferenças entre exemplos de classes distintas.

Por fim, é possível discutir sobre os modelos de melhor desempenho: a acurácia relativamente alta do método de *Naive Bayes* sugere que há uma real independência entre os atributos até certo ponto, porém, devido à grande quantidade de suposições que a abordagem leva em conta, é esperado que ela não produza os melhores resultados. Em contrapartida, algo que não seria esperado antes dos experimentos é o melhor desempenho da regressão logística, uma vez que tal método é normalmente utilizado para outro tipo de problema. É possível que o modelo de regressão tenha sido capaz de determinar as variáveis mais importantes para determinar a saída correta, dando maior peso às mesmas, enquanto o modelo de floresta aleatória focou, em determinado momento, em atributos menos relevantes, criando nós à sua volta que não impactaram positivamente na acurácia.

Em relação aos resultados de Kinkade e Lim (2015), os resultados obtidos neste trabalho foram até 10% melhores. Apesar da comparação ser apenas indireta, uma vez que os estudos tratam de jogos diferentes, a adição dos dados dos jogadores de cada partida é uma possível responsável pela melhor acurácia dos modelos.

Além do entendimento dos resultados, é natural que haja o questionamento

em relação ao que está acontecendo dentro dos modelos, isto é, à sua interpretabilidade. Deseja-se, assim, saber quais atributos são mais relevantes para a decisão de cada modelo. Para chegar a tal conclusão, é possível utilizar uma abordagem baseada em *Shapley Values*, um conceito proveniente da Teoria dos Jogos, descrito pela primeira vez em 1953, que busca distribuir recompensas aos jogadores de acordo com sua contribuição ao montante total (Shapley, 1953). Tal abordagem será utilizada para explicar os modelos treinados usando engenharia de características.

No contexto deste trabalho, o montante total é a saída do modelo, os jogadores são os atributos de cada exemplo, e sua contribuição é o peso de cada atributo para a predição do modelo. Para tal, foi utilizada a biblioteca SHAP[18], de forma a buscar entender quais aspectos do problema possuem maior peso na tomada de decisão e, portanto, devem ser tratados com maior importância em soluções futuras.

A Figura 12 representa a distribuição dos valores SHAP do modelo de regressão logística após seu treinamento. Cada exemplo testado representa um pixel no gráfico: quanto mais longe do centro, maior a importância daquele atributo para a escolha de uma das classes e, consequentemente, maior seu valor Shapley; quanto mais grossa a linha, maior o número de exemplos com esse valor e quanto mais próximo de vermelho, maior o valor de entrada observado para aquele atributo. É possível perceber que, se um exemplo do *dataset* possui um *PlayerCountering* grande, ou seja, próximo do vermelho, esse atributo possuirá grande relevância para a escolha da classe 1 e, portanto, terá um valor Shapley também elevado.

18. https://shap.readthedocs.io/en/latest/

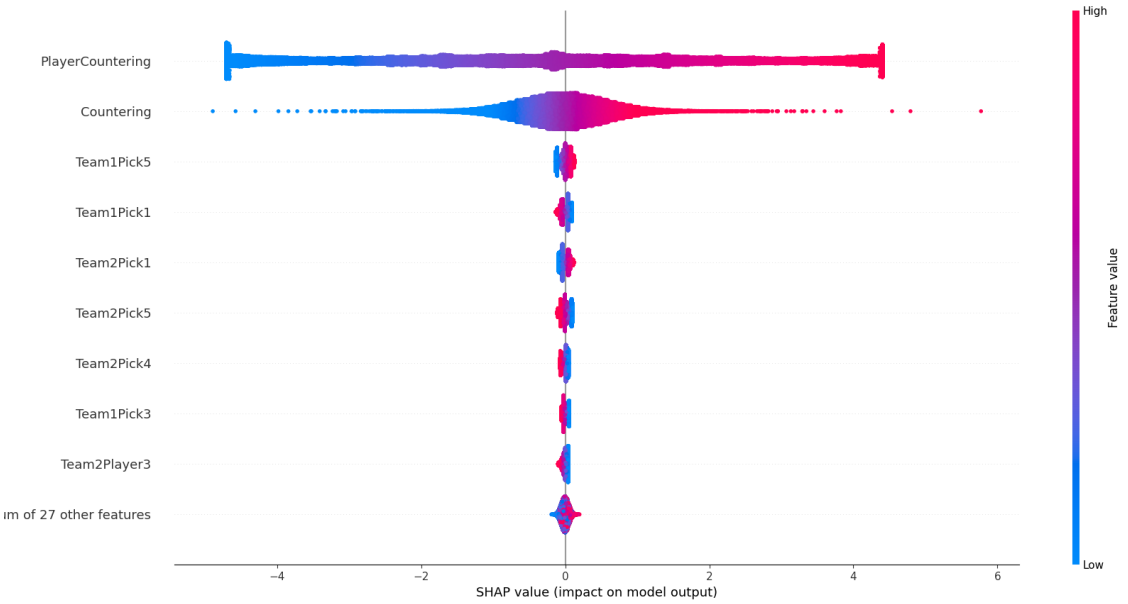

Figura 12: Gráfico do tipo “enxame” dos valores Shapley do modelo de regressão logística. Cada atributo “Pick” é um campeão escolhido

Apesar de se referir a um modelo em específico, a distribuição encontrada nos outros modelos bem-sucedidos é extremamente semelhante a esta, como mostrado na Figura 13, desta vez em forma de barras apenas para possibilitar uma visualização diferente. Quanto maior a barra, maior a importância daquele atributo para a escolha de uma das classes e, portanto, maior o valor Shapley.

Através desta abordagem, é possível perceber que os atributos mais relevantes para a decisão dos modelos são aqueles provenientes de engenharia de características, o que está de acordo com o que era esperado, já que esses atributos são criados a partir de diversos outros a fim de auxiliar a classificação do modelo. Outra interpretação importante é que os dados de jogadores são mais pertinentes do que os dados de campeões, visto que o tamanho da barra para o atributo *PlayerCountering* é maior do que os demais. Mais uma vez, tal propriedade pode ser dita como esperada, uma vez que jogadores melhores tendem a ganhar mais partidas, independentemente do campeão que estiverem utilizando.

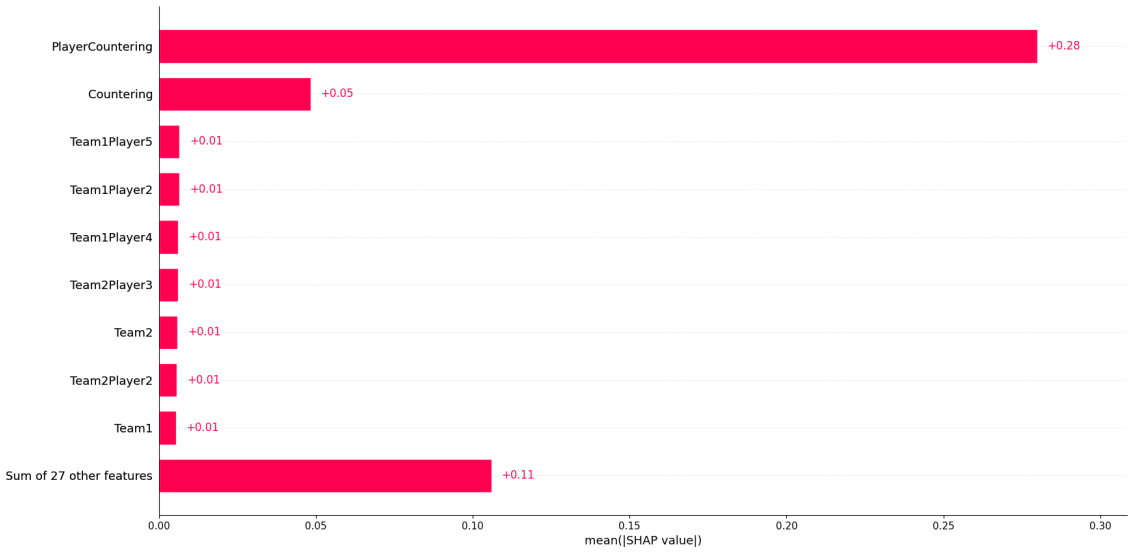

Figura 13: Gráfico em barras dos valores SHAP do modelo de florestas aleatórias.

6 Conclusão

6.1 Considerações Finais

Com os resultados expostos no capítulo anterior, é possível perceber que o problema possui complexidade suficiente para que não haja uma solução direta, especialmente sem o tratamento dos dados originais. Apesar da capacidade computacional, não há atributos suficientes para a modelagem de um sistema capaz de minimizar o impacto humano dentro de cada partida, utilizando somente os dados estatísticos.

Em contrapartida, o processo de engenharia de características é capaz de transformar completamente o problema, cedendo aos modelos métricas sobre o relacionamento de seres humanos entre si e dos personagens uns com os outros. Com isso, é possível atingir um patamar de desempenho extremamente aceitável e, apesar de ainda existirem erros de classificação, isso ocorre pois nenhum *draft* ideal será capaz de superar a imprevisibilidade de um esporte, mesmo que virtual, uma vez que jogadores podem ter uma atuação muito melhor ou pior do que seu adversário em um determinado dia específico, contrariando o que os dados históricos apontavam.

Essa importância da variância humana no resultado da partida possui grande influência na força de cada atributo para os modelos, sendo o mais relevante o quão vitorioso é um jogador sobre cada adversário. Essa força é visualizável através da análise com *Shapley Values* e os gráficos gerados com a biblioteca SHAP, que demonstram não somente quão valioso é esse atributo, mas também os outros gerados com engenharia de características.

6.2 Limitações e Trabalhos Futuros

Os atributos obtidos através de engenharia de características são construídos a partir de uma base histórica que permite gerar conexões entre jogadores e campeões para um jogo ainda seguinte. Apesar disso, o *LoL* recebe atualizações quinzenais, que mudam completamente o mapa do jogo, as características de cada campeão e diversos outros aspectos relevantes à análise de um *draft*.

Portanto, torna-se necessária a constante alimentação de novos dados para

os modelos, pois, por exemplo, um destes modelos treinado com partidas de 2024 certamente não possuirá bons resultados para partidas de 2025, já que o jogo foi alterado drasticamente.

Com isso em mente, as seguintes ideias são consideradas como trabalhos futuros:

- Criação de um sistema de alimentação contínua, utilizando dados de cada nova partida realizada para novos treinamentos;
- Criação de um sistema que possa ler dados parciais de *draft*, de forma a realizar predição e apresentar as probabilidade de vitória, em tempo real, conforme as escolhas de campeões são realizadas.
- Elaboração de um modelo capaz de ler drafts parciais e recomendar decisões futuras, baseado na possível chance de vitória.

Referências

Aggarwal, Charu C. 2023. *Neural Networks and Deep Learning: A Textbook.* 2nd. Springer Publishing Company, Incorporated. ISBN: 3031296419.

Ahani, Ali, Mehrbakhsh Nilashi, Othman Ibrahim, Louis Sanzogni e Scott Weaven. 2019. "Market segmentation and travel choice prediction in Spa hotels through TripAdvisor's online reviews". *International Journal of Hospitality Management* 80:52–77.

Akhmedov, Kodirjon e Anh Huy Phan. 2021. *Machine learning models for DOTA 2 outcomes prediction.* arXiv: 2106.01782 [cs.LG]. https://arxiv.org/abs/2106.01782.

Amherd, Fabian e Elias Rodriguez. 2021. *Heatmap-based Object Detection and Tracking with a Fully Convolutional Neural Network.* https://doi.org/10.48550/arXiv.2101.03541.

Baker, Bowen, Ingmar Kanitscheider, Todor Markov, Yi Wu, Glenn Powell, Bob McGrew e Igor Mordatch. 2020. *Emergent Tool Use From Multi-Agent Autocurricula.* arXiv: 1909.07528 [cs.LG].

Bengio, Yoshua, Réjean Ducharme e Pascal Vincent. 2000. "A neural probabilistic language model". *Advances in neural information processing systems* 13.

Berrar, Daniel. 2018. "Bayes' theorem and naive Bayes classifier". *Encyclopedia of Bioinformatics and Computational Biology: ABC of Bioinformatics.*

Boser, Bernhard E, Isabelle M Guyon e Vladimir N Vapnik. 1992. "A training algorithm for optimal margin classifiers". Em *Proceedings of the fifth annual workshop on Computational learning theory,* 144–152.

Breiman, Leo. 2001. "Random forests". *Machine learning* 45:5–32.

Buijsman, Michiel. 2024. *The global games market will generate $187.7 billion in 2024.* https://newzoo.com/resources/blog/global-games-market-revenue-estimates-and-forecasts-in-2024.

Chen, Yunji, Ling Li, Wei Li, Qi Guo, Zidong Du e Zichen Xu. 2024. "Chapter 2 - Fundamentals of neural networks". Em *AI Computing Systems,* editado por

Yunji Chen, Ling Li, Wei Li, Qi Guo, Zidong Du e Zichen Xu, 17–51. Morgan Kaufmann. ISBN: 978-0-323-95399-3. https://doi.org/https://doi.org/10.1016/B978-0-32-395399-3.00008-1. https://www.sciencedirect.com/science/article/pii/B9780323953993000081.

Cortes, Corinna e Vladimir Vapnik. 1995. "Support-vector networks". *Machine learning* 20:273–297.

Cutler, D. Richard, Thomas C. Edwards Jr., Karen H. Beard, Adele Cutler, Kyle T. Hess, Jacob Gibson e Joshua J. Lawler. 2007. "RANDOM FORESTS FOR CLASSIFICATION IN ECOLOGY". *Ecology* 88 (11): 2783–2792. https://doi.org/https://doi.org/10.1890/07-0539.1. eprint: https://esajournals.onlinelibrary.wiley.com/doi/pdf/10.1890/07-0539.1. https://esajournals.onlinelibrary.wiley.com/doi/abs/10.1890/07-0539.1.

Efron, Bradley e Robert J Tibshirani. 1994. *An introduction to the bootstrap.* Chapman / Hall/CRC.

Esports Charts. 2023. *Faker claims 4th World Championship title at the most popular esports event ever.* https://escharts.com/news/worlds-2023-recap?utm_campaign=news_posts&utm_medium=esc_socials&utm_term=2023_11_19.

Gill, Sunil. 2025. *League Of Legends Player Count & Stats 2025.* https://prioridata.com/data/league-of-legends/.

Grinsztajn, Léo, Edouard Oyallon e Gaël Varoquaux. 2022. *Why do tree-based models still outperform deep learning on tabular data?* arXiv: 2207.08815 [cs.LG]. https://arxiv.org/abs/2207.08815.

Ho, Tin Kam. 1995. "Random decision forests". Em *Proceedings of 3rd International Conference on Document Analysis and Recognition,* volume 1, 278–282 vol.1. https://doi.org/10.1109/ICDAR.1995.598994.

Jiang, Yuchen, Xiang Li, Hao Luo, Shen Yin e Okyay Kaynak. 2022. "Quo vadis artificial intelligence?" *Discover Artificial Intelligence* 2 (1): 4.

Joyce, James. 2021. "Bayes' Theorem". Em *The Stanford Encyclopedia of Philosophy,* Fall 2021, editado por Edward N. Zalta. Metaphysics Research Lab, Stanford University.

Kingma, Diederik P. e Jimmy Ba. 2017. *Adam: A Method for Stochastic Optimization.* arXiv: 1412.6980 [cs.LG]. https://arxiv.org/abs/1412.6980.

Kingsford, Carl e Steven L Salzberg. 2008. "What are decision trees?" *Nature biotechnology* 26 (9): 1011–1013.

Kinkade, Nicholas e K Lim. 2015. "Dota 2 win prediction". *Univ Calif* 1:1–13.

Krizhevsky, Alex, Ilya Sutskever e Geoffrey E Hinton. 2012. "Imagenet classification with deep convolutional neural networks". *Advances in neural information processing systems* 25.

LaValley, Michael P. 2008. "Logistic Regression". *Circulation* 117 (18): 2395–2399. https://doi.org/10.1161/CIRCULATIONAHA.106.682658. eprint: https://www.ahajournals.org/doi/pdf/10.1161/CIRCULATIONAHA.106.682658. https://www.ahajournals.org/doi/abs/10.1161/CIRCULATIONAHA.106.682658.

Learning, Machine. 1997. "Tom mitchell". *Publisher: McGraw Hill.*

Lewis, Michael. 2004. *Moneyball: The art of winning an unfair game.* WW Norton & Company.

Lol Esports. 2018. *2018 Events By the Numbers.* https://nexus.leagueoflegends.com/en-us/2018/12/2018-events-by-the-numbers/.

Magee, John F. 1964. *Decision trees for decision making.* Harvard Business Review Brighton, MA, USA.

Manski, Charles F. 1991. "Regression". *Journal of Economic Literature* 29 (1): 34–50. ISSN: 00220515, acedido em 31 de janeiro de 2025. http://www.jstor.org/stable/2727353.

Mitchell, Tom M. 1997. *Machine learning.*

Noble, William S. 2006. "What is a support vector machine?" *Nature biotechnology* 24 (12): 1565–1567.

Oostwal, Elisa, Michiel Straat e Michael Biehl. 2021. "Hidden unit specialization in layered neural networks: ReLU vs. sigmoidal activation". *Physica A: Statistical Mechanics and its Applications* 564:125517. ISSN: 0378-4371. https://doi.org/https://doi.org/10.1016/j.physa.2020.125517. https://www.sciencedirect.com/science/article/pii/S0378437120308153.

OpenAI. 2022. *Introducing ChatGPT.* https://openai.com/index/chatgpt/.

PwC. 2024. *Perspectives from the Global Entertainment & Media Outlook 2024–2028.* https://www.pwc.com/gx/en/issues/business-model-reinvention/outlook/insights-and-perspectives.html.

Quinlan, J.R. 1990. "Decision trees and decision-making". *IEEE Transactions on Systems, Man, and Cybernetics* 20 (2): 339–346. https://doi.org/10.1109/21.52545.

Rodríguez-Pérez, Raquel e Jürgen Bajorath. 2022. "Evolution of Support Vector Machine and Regression Modeling in Chemoinformatics and Drug Discovery". *Journal of Computer-Aided Molecular Design* 36:1–8. https://doi.org/10.1007/s10822-022-00442-9.

Ruby, Usha e Vamsidhar Yendapalli. 2020. "Binary cross entropy with deep learning technique for image classification". *Int. J. Adv. Trends Comput. Sci. Eng* 9 (10).

Schmidt-Hieber, Johannes. 2020. "Nonparametric regression using deep neural networks with ReLU activation function". *The Annals of Statistics* 48 (4): 1875–1897. https://doi.org/10.1214/19-AOS1875. https://doi.org/10.1214/19-AOS1875.

Schudey, Alexander, Pavel Kasperovich, Adeel Ikram, David Panhans e Lyudmila Matviets. 2023. *Let the Game Begin: How Esports Is Shaping the Future of Live Entertainment.* https://www.bcg.com/publications/2023/how-esports-will-become-future-of-entertainment.

Shannon, Claude E. 1951. "Prediction and entropy of printed English". *Bell system technical journal* 30 (1): 50–64.

Shapley, L. S. 1953. "17. A Value for n-Person Games". Em *Contributions to the Theory of Games, Volume II,* editado por Harold William Kuhn e Albert William Tucker, 307–318. Princeton: Princeton University Press. ISBN: 9781400881970.

https://doi.org/doi:10.1515/9781400881970-018. https://doi.org/10.1515/9781400881970-018.

Shen, Qiyuan. 2022. "A machine learning approach to predict the result of League of Legends". Em *2022 International Conference on Machine Learning and Knowledge Engineering (MLKE),* 38–45. https://doi.org/10.1109/MLKE55170.2022.00013.

Stoltzfus, Jill C. 2011. "Logistic Regression: A Brief Primer". *Academic Emergency Medicine* 18 (10): 1099–1104. https://doi.org/https://doi.org/10.1111/j.1553-2712.2011.01185.x. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1553-2712.2011.01185.x. https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1553-2712.2011.01185.x.

Vaghasiya, Krunal. 2024. *The Latest ChatGPT Statistics and User Trends (2022-2025).* https://wisernotify.com/blog/chatgpt-users/.

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser e Illia Polosukhin. 2017. "Attention is all you need". *Advances in neural information processing systems* 30.

Webb, Geoffrey I, Eamonn Keogh e Risto Miikkulainen. 2010. "Naïve Bayes." *Encyclopedia of machine learning* 15 (1): 713–714.

Wehrl, Alfred. 1978. "General properties of entropy". *Rev. Mod. Phys.* 50 (2): 221–260. https://doi.org/10.1103/RevModPhys.50.221. https://link.aps.org/doi/10.1103/RevModPhys.50.221.