UNIVERSIDADE FEDERAL DO RIO DE JANEIRO
INSTITUTO DE COMPUTAÇÃO
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

BRENO PONTES DA COSTA

PREDIÇÃO DE RESULTADOS EM PARTIDAS DE LEAGUE OF LEGENDS USANDO REDES NEURAIS E ANÁLISE SHAP

RIO DE JANEIRO
2023

BRENO PONTES DA COSTA

PREDIÇÃO DE RESULTADOS EM PARTIDAS DE LEAGUE OF LEGENDS USANDO REDES NEURAIS E ANÁLISE SHAP

Trabalho de conclusão de curso de graduação apresentado ao Instituto de Computação da Universidade Federal do Rio de Janeiro como parte dos requisitos para obtenção do grau de Bacharel em Ciência da Computação.

Orientador: Prof. Geraldo Bonorino Xexéo

Co-orientador:

RIO DE JANEIRO

2023

CIP - Catalogação na Publicação

C837p

Costa, Breno Pontes da
Predição de resultados em partidas de League of Legends usando redes neurais e análise SHAP / Breno Pontes da Costa. -- Rio de Janeiro, 2023.
57 f.

Orientador: Geraldo Bonorino Xexéo.
Trabalho de conclusão de curso (graduação) - Universidade Federal do Rio de Janeiro, Instituto de Computação, Bacharel em Ciência da Computação, 2023.

1. Redes neurais. 2. Inteligência artificial. 3. Mineração de dados. 4. League of Legends. 5. SHAP. I. Xexéo, Geraldo Bonorino, orient. II. Título.

Elaborado pelo Sistema de Geração Automática da UFRJ com os dados fornecidos pelo(a) autor(a), sob a responsabilidade de Miguel Romeu Amorim Neto - CRB-7/6283.

BRENO PONTES DA COSTA

PREDIÇÃO DE RESULTADOS EM PARTIDAS DE LEAGUE OF LEGENDS USANDO REDES NEURAIS E ANÁLISE SHAP

Trabalho de conclusão de curso de graduação apresentado ao Instituto de Computação da Universidade Federal do Rio de Janeiro como parte dos requisitos para obtenção do grau de Bacharel em Ciência da Computação.

Aprovado em 23 de agosto de 2023

BANCA EXAMINADORA:

Geraldo Bonorino Xexéo
D.Sc. (UFRJ)

Carolina Gil Marcelino
D.Sc. (UFRJ)

Leandro Ouriques Mendes de Carvalho
M.Sc. (Marinha do Brasil)

Dedico este trabalho a minha família e minha namorada, pois me deram força nesta longa caminhada de estudos.

AGRADECIMENTOS

Gostaria de agradecer ao meu orientador, Geraldo Bonorino Xexéo, à minha namorada, Juliana do Espírito Santo Veloso, e ao meu co-orientador, Lincoln Magalhães Costa, por me ajudarem no estudo, desenvolvimento e revisão do texto, o que tornou possível a conclusão deste trabalho.

"Ensinar não é transferir conhecimento, mas criar as possibilidades para a sua própria produção ou a sua construção."

Paulo Freire

RESUMO

League of Legends é um dos jogos competitivos que mais movimenta dinheiro em campeonatos oficiais. Devido ao tamanho retorno financeiro das competições, das apostas esportivas e da complexidade do jogo, tornam-se interessantes estudos que busquem as melhores estratégias ou que contribuam para o desenvolvimento das equipes. Este trabalho de conclusão de curso tem como objetivo observar trabalhos anteriores da área, desenvolver duas redes neurais artificiais para prever um time vitorioso nos tempos de 10 e 15 minutos, respectivamente, estudar para buscar os atributos que mais influenciam na decisão destas redes e, por fim, comparar resultados com outros estudos sobre o assunto. Os dados que baseiam este trabalho foram extraídos da plataforma (ELIXIR, 2023), com partidas do ano de 2023, de janeiro a abril. O modelo de rede neural utilizado é a MultiLayer Perceptron, além disso, para estudar os atributos que mais influenciaram nas decisões das redes, foi usada a biblioteca SHAP. A acurácia da rede neural desenvolvida no tempo de 10 minutos é de 72,03%. Os atributos que mais influenciaram na decisão foram a diferença de *mineons*, ou monstros da selva abatidos entre os *junglers* e a quantidade de ouro do *top* do time vermelho. Já no tempo de 15 minutos, a acurácia é de 74,68%, sendo a diferença de ouro entre os *bots* e de *mineons* abatidos entre os *junglers* os atributos que mais influenciaram. Conclui-se então que o *jungler*, nos primeiros minutos, deve focar em abater mais monstros da selva para aumentar as chances de vitória. Além disso, o *bot* e o *mid* também devem focar no abate de *mineons*.

Palavras-chave: redes neurais; Inteligência Artificial; mineração de dados; League of Legends; SHAP.

ABSTRACT

League of Legends is one of the competitive games that moves more money in official championships. Due to the financial return of competitions, sports betting and the complexity of the game, studies that seek the best strategies or that contribute to the development of teams become interesting. This course completion work aims to observe previous works in the area, develop two artificial neural networks to predict a winning team in the times of 10 and 15 minutes, respectively, study them to look for the attributes that most influence their decision and finally, compare results with other studies on the subject. The data on which this work is based were extracted by the (ELIXIR, 2023) platform, with departures from the year 2023, from January to April. The neural network model used is the MultiLayer Perceptron, in addition, to study the attributes that most influenced the decisions of the networks, the SHAP library was used. The accuracy of the neural network developed in 10 minutes is 72.03%. The attributes that most influenced the decision were the difference in mineons or jungle monsters killed between the junglers and the amount gold in red team top laner. At 15 minutes, the accuracy is 74.68%, with the difference in gold between bots and the difference in mineons and monsters killed among junglers the attributes that most influenced. It follows then that the jungler, in the first few minutes, should focus on killing more jungle monsters to increase the chances of victory. Furthermore, bot and mid should also focus on killing mineons.

Keywords: neural network; artificial intelligence; data mining; League of Legends; SHAP.

LISTA DE ILUSTRAÇÕES

LISTA DE CÓDIGOS

SUMÁRIO

1 INTRODUÇÃO

Com o crescimento da indústria de jogos e a relevância dos esportes eletrônicos (eSports) no mercado, estudos relacionados se tornam cada vez mais frequentes e relevantes. Segundo (WAGNER, 2006): ”“eSports” é uma área de atividades desportivas em que pessoas desenvolvem e treinam habilidades mentais ou físicas no uso das tecnologias de informação e comunicação.”

No ano de 2021, a premiação para o campeonato mundial de League of Legends foi de US$ 2,5 milhões. Devido ao tamanho retorno financeiro, equipes investem cada vez mais em estratégias e treinamentos a fim de se tornarem mais competitivas neste ambiente. Ferramentas estatísticas ou preditivas se tornam grandes aliadas no processo de estudo de melhores estratégias, treinamento de jogadores e apostas online.

1.1 INTRODUÇÃO AO LEAGUE OF LEGENDS

Segundo (RIOT, 2022): “League of Legends é um jogo de estratégia em que duas equipes de cinco poderosos campeões se enfrentam para destruir a base uma da outra. Escolha entre mais de 140 campeões para realizar jogadas épicas, assegurar abates e destruir torres conforme você luta até a vitória.”. Nesse jogo, cada integrante do time terá um campeão, que é um personagem jogável do League of Legends. Cada um possui habilidades e propósitos diferentes na partida. O objetivo de cada time é destruir o *Nexus* do time inimigo, este se encontra no interior da base do time rival.

Em um instante que antecede a partida acontece a fase de *picks* e *bans*, que consiste no seguinte protocolo:

1. Time azul bane um campeão;
2. Time vermelho bane um campeão;
3. Time azul bane um campeão;
4. Time vermelho bane um campeão;
5. Time azul bane um campeão;
6. Time vermelho bane um campeão;
7. Time azul escolhe um campeão para compor a equipe;
8. Time vermelho escolhe dois campeões para comporem a equipe;
9. Time azul escolhe dois campeões para comporem a equipe;

10. Time vermelho escolhe um campeão para compor a equipe;
11. Time vermelho bane um campeão;
12. Time azul bane um campeão;
13. Time vermelho bane um campeão;
14. Time azul bane um campeão;
15. Time vermelho escolhe um campeão para compor a equipe;
16. Time azul escolhe dois campeões para comporem a equipe;
17. Time vermelho escolhe um campeão para compor a equipe.

É importante ressaltar que, uma vez que o campeão é banido do jogo, o mesmo não poderá ser escolhido por nenhuma equipe. Além disso, não é possível banir um campeão duas vezes, também não é possível dois jogadores escolherem o mesmo campeão. Após a fase de *picks* e *bans*, é iniciada a partida.

Para chegar até a base rival, é necessário destruir certas torres do time inimigo e neste processo, o jogador, junto ao seu time, deverá se fortalecer com itens e com avanço de nível. Em uma partida, cada jogador recebe uma quantia de ouro por segundo, que será usado na compra de itens. Também é possível conseguir dinheiro abatendo torres inimigas, tropas inimigas (chamadas de *minions*), campeões inimigos e completando objetivos globais, que consistem em abater monstros espalhados pelo mapa.

Todos os campeões começam a partida no nível 1, podendo chegar até o nível 18. Abates a campeões inimigos, tropas inimigas, monstros neutros e estruturas recompensam o campeão escolhido com experiência e ouro, que serão necessárias para subir de nível e comprar novos itens durante a partida.

Cada jogador possui um papel que consiste em uma posição na qual ele se encontra no time. Os papéis disponíveis são *top*, *mid*, *jungler*, *adc* e *support*. A Figura 1 ilustra o mapa do jogo. A cada trinta segundos, o Nexus de cada time libera três hordas de *minions*, cada horda de um time anda por uma das três rotas no sentido do Nexus inimigo.

Os objetivos globais mencionados anteriormente consistem em abater dragões, arauto ou barão. O abate desses monstros por uma equipe recompensa com o aumento de habilidades durante o jogo, dependendo do monstro abatido. Por exemplo, o time que derrotar o barão ganha retorno de base acelerado, aumento no poder de habilidade e no dano de ataque. Além disso, as tropas do time que fez o abate são fortalecidas com a presença de algum jogador do próprio time.

Figura 1 – Mapa do League of Legends

1.1.1 Papeis de cada jogador

O jogador da rota do topo, o *top*, geralmente utiliza campeões de alta durabilidade com capacidade balanceada de causar dano no time inimigo. Esta é a rota mais distante dos possíveis eventos do início da partida, como a luta pelo dragão. Grande parte de seu ouro é adquirido através do abate de minions inimigos.

Na rota do meio se encontra o *mid*. Nesta rota é comum campeões mais frágeis, porém, com alto potencial de causar dano no time inimigo. Nesta rota é comum também campeões do tipo "assassino", que são capazes de abater rapidamente campeões frágeis do time inimigo.

O *jungler* se encontra na selva, não estando presente em nenhuma das rotas. A maior parte de seu recurso vem de monstros neutros da selva. Buscam aparecer de surpresa nas rotas para conseguir um abate, ou ajudar o seu time de outras formas.

O papel do *bot* é causar muito dano no time inimigo, em contra partida, os campeões que atuam neste papel são frágeis. Assim como o *top* e *mid*, maior parte do seu recurso vem de *minions* da rota inferior, além disso, é comum campeões que atacam de longa distância.

Por fim, o *sup* se encontra na rota inferior junto com o *bot*. Possuem habilidades de cura, proteção ou "controle de grupo", que consiste em paralisar, atordoar ou diminuir a velocidade de um ou mais campeões inimigos. Os itens do *sup* são baratos, sendo assim, não é necessário abater *minions* para evoluir na partida.

1.2 OBJETIVOS

Este trabalho de conclusão de curso busca utilizar redes neurais do tipo *MultiLayer Perceptron* para prever o vencedor de uma partida profissional de League of Legends, contribuindo na área de estudo de esportes eletrônicos. Serão utilizados apenas dados

obtidos durante a partida, ou seja, qualquer dado pré-partida será desconsiderado, como desempenho do jogador no campeonato e *picks* e *bans*. Serão analisados os desempenhos de cada jogador nos tempos de 10 e 15 minutos. Para uma melhor análise da rede neural desenvolvida, será buscado também quais atributos influenciaram da decisão da rede.

1.2.1 Objetivos específicos

1. Analisar trabalhos anteriores sobre o uso de aprendizado de máquina nos esportes em geral, esportes eletrônicos e, por fim, no *League of Legends*;
2. Desenvolver duas redes neurais que visam prever um vitorioso em uma partida de *League of Legends*, aos 10 e 15 minutos de partida;
3. Utilizar a ferramenta SHAP para analisar as rede neurais e buscar os atributos que mais influenciaram na decisão das redes;
4. Comparar os resultados obtidos com outros trabalhos relacionados na área.

1.2.2 Corpo do projeto final

1. O capítulo 1 visa introduzir o problema, apresentar o objeto de estudo e o que será feito neste projeto de conclusão de curso.
2. O capítulo 2 apresenta uma breve fundamentação teórica sobre aprendizado de máquina e redes neurais, explicando o histórico da área e o funcionamento das redes Perceptron e Multilayer Perceptron. Além disso, se aborda o problema de entender certas redes neurais e o funcionamento da ferramenta SHAP. Por fim, apresenta formas de avaliar redes neurais e faz uma revisão de estudos correlatos na área.
3. O capítulo 3 mostra como é feito o desenvolvimento da rede neural, apresentando a base de dados estudada, as ferramentas e metodologias utilizadas. Além disso, são apresentados os resultados das redes neurais e uma comparação com outros trabalhos de temas correlatos.
4. O capítulo 4 mostra uma explicação das redes neurais utilizando SHAP. Além disso, é feita uma análise dos resultados obtidos.
5. Por fim, o capítulo 5 apresenta uma conclusão deste trabalho de conclusão de curso. Além disso, são apresentados possíveis trabalhos futuros.

2 APRENDIZADO DE MÁQUINA COM REDES NEURAIS

2.1 INTRODUÇÃO AO APRENDIZADO DE MÁQUINA

Desde a sua criação, o computador tem como objetivo auxiliar o ser humano na resolução de problemas de diversos tipos, desde cálculos matemáticos até o reconhecimento de rostos em fotos. O objetivo do campo de estudo de aprendizado de máquina é desenvolver sistemas computacionais com capacidade de melhorar seu desempenho automaticamente a partir das suas experiências. Esses sistemas coletam ou recebem dados de forma externa e os utilizam como base para a melhora na tomada de decisão.

Em termos gerais, algoritmos de aprendizado de máquina podem ter seus métodos de aprendizado classificados em supervisionados e não supervisionados. No aprendizado supervisionado, o conjunto de dados usados para ensinar o sistema possui a saída desejada para cada dado. Algoritmos com essa metodologia de aprendizado são muito utilizados em problemas de predição e classificação. Já no aprendizado não supervisionado, a base de dados usada para ensinar o sistema não possui atributo de saída desejada. Estes algoritmos são comumente utilizados em problemas de reconhecimento de padrões e agrupamento. Dentro do campo de estudo de aprendizado de máquina existe a área de redes neurais artificiais (RNA). Os RNAs são algoritmos que possuem como inspiração o neurônio humano e buscam utilizar diferentes métodos de aprendizagem para auxiliar na solução de problemas.

2.2 REDES NEURAIS

O primeiro modelo matemático de uma rede neural artificial foi introduzido em (MCCULLOCH; PITTS, 1944). Nesse modelo, os sinais de entrada assumem valores binários e possuem o mesmo peso, tais entradas são somadas e o resultado é submetido a uma função degrau, com saídas 0 ou 1. Em (HEBB, 1949) é apresentado um método para treinamento de uma rede neural artificial, intitulado “Hebb ’s rule”. Já em (ROSENBLATT, 1958), é apresentada a rede Perceptron, que consiste em um único neurônio e possui como metodologia de treinamento a Hebb ’s rule, que é supervisionado.

2.3 REDE PERCEPTRON

A estrutura da rede Perceptron consiste em x_i entradas com seus respectivos pesos sinápticos w_i, necessários para a quantificação da importância de cada entrada na rede, um limiar de ativação θ e uma função de ativação g para o resultado da rede. A equação 2.1 mostra uma representação matemática do neurônio perceptron enquanto a Figura 2 permite uma melhor visualização do fluxo do neurônio.

$$\begin{cases} u = \sum_{i=1}^{n} w_i \cdot x_i - \theta \\ y = g\,(u) \end{cases} \tag{2.1}$$

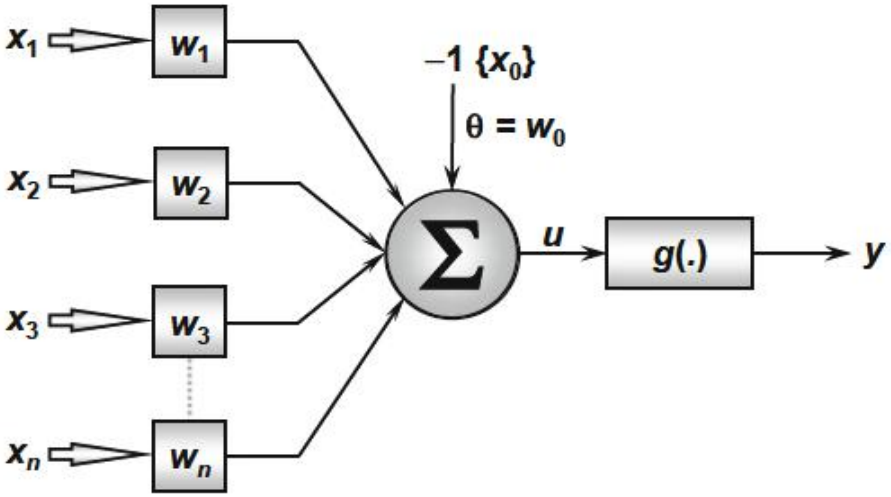

Figura 2 – Ilustração de uma rede Perceptron. Fonte: (SILVA et al., 2016, p. 30).

.

Devido à simplicidade de sua estrutura e metodologia de treinamento, a rede perceptron foi ganhando notoriedade no meio acadêmico. Já Widrow e Hoff (1960), apresentam a rede ADALINE, que também possui apenas uma camada com um único neurônio. Esses autores também apresentam sua forma de treinamento, intitulada "Delta Rule". Essa metodologia consiste em ajustar os pesos e o limiar de ativação para diminuir o erro entre a saída desejada e a encontrada.

É válido ressaltar que as redes apresentadas se limitam a resolver problemas lineares (BLOCK, 1970). O problema do ou-exclusivo (XOR), não linear, é conhecido por expor tal limitação. Sejam quatro pontos (x, y) com os valores $A = (0, 0)$, $B = (0, 1)$, $C = (1, 0)$ e $D = (1, 1)$ em um plano, como na Figura 3:

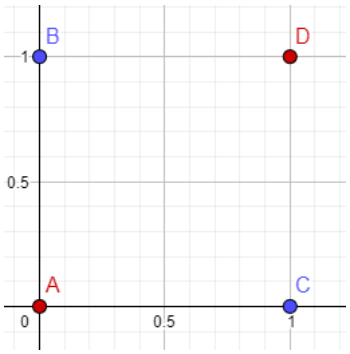

Figura 3 – Gráfico com os pontos $A = (0, 0)$, $B = (0, 1)$, $C = (1, 0)$ e $D = (1, 1)$.

O objetivo é classificar os pontos de acordo com o resultado da operação $x \otimes y$, onde que $\otimes$ representa o ou-exclusivo. Ou seja, um grupo terá como resultado da operação o valor 1, representado pela cor azul, e o outro grupo terá como resultado o valor 0, representado pela cor vermelha. A rede perceptron (composta por um único neurônio perceptron) encontra uma linha que divide o espaço em duas partes. No entanto, não será possível classificar os grupos de forma linear, pois este é um problema de natureza não linear.

2.4 MULTI-LAYER PERCEPTRON

Como apresentado em (SILVA et al., 2016), as redes Multilayer Perceptron (MLP) são capazes de resolver problemas de natureza não lineares. Elas consistem em camadas de neurônios perceptron e se dividem em camada de entrada, camadas ocultas e camada de saída. Em uma MLP existirá apenas uma camada de entrada e uma camada de saída, além disso podem conter uma ou mais camadas ocultas. A quantidade de camadas em uma rede é chamada de profundidade e a quantidade de neurônios em uma camada é chamada de largura.

2.4.1 Princípio operacional da MLP

A camada de entrada recebe os sinais sinápticos a serem estudados. Os sinais de saída de cada camada são utilizados como sinais de entrada para a camada seguinte. A propagação da informação é feita sempre em uma única direção, partindo da camada de entrada e seguindo no sentido da camada de saída.

O algoritmo de aprendizado da MLP, chamado de *backpropagation*[1], é dividido em duas etapas. A primeira etapa consiste no envio dos sinais na camada de entrada, propagando para todas as camadas subsequentes. O objetivo desta etapa é buscar todos os sinais de saída da rede neural. Nesta etapa, os pesos sinápticos e o limiar de ativação se mantêm os mesmos. Ao final, os valores de saída da rede são comparados com os valores esperados, calculando, assim, o erro para aquele atributo.

A segunda etapa do treinamento consiste em ajustar os pesos e limiares de cada neurônio da rede de acordo com o erro. Sendo assim, a execução destas etapas uma ou mais vezes de forma alternada, possibilita que os parâmetros da rede se ajustem, diminuindo o somatório de erros da rede.

Para minimizar o erro da rede neural, é utilizado o método gradiente descendente[2], que consiste em buscar de forma iterativa parâmetros que minimizem a função de erro. Para casos de redes com muitos neurônios, é interessante o uso de uma variante deste algoritmo, conhecida como gradiente descendente estocástico (*sgd*, na sigla em inglês).

[1] Uma explicação mais detalhada pode ser vista em (SILVA et al., 2016)
[2] Uma explicação mais detalhada pode ser vista em (SILVA et al., 2016)

Esta visa utilizar amostras aleatórias para somatório de erro, não utilizando todas as amostras e, assim, otimizando o processo de convergência para o menor erro.

2.4.2 Número de camadas e neurônios

Um dos desafios de trabalhar com redes neurais Multilayer Perceptron é a escolha da quantidade de camadas e de neurônios na rede. Apesar de não existir uma fórmula definida que retorne números exatos de acordo com o problema, existem técnicas que auxiliam no processo de busca. Panchal et al. (2011) mostra em seu artigo que apesar da possibilidade de se trabalhar com múltiplas camadas ocultas, a maioria dos problemas não exige que se trabalhe com mais de duas camadas, além das camadas de entrada e saída. Este artigo também apresenta alguns princípios que visam auxiliar na testagem de neurônios para melhores resultados. São eles:

- O número de neurônios ocultos deve estar entre os tamanho da camada de entrada e o tamanho da camada de saída;
- O número de neurônios ocultos deve ser 2/3 do tamanho da camada de entrada, mais o tamanho da camada de saída;
- O número de neurônios ocultos deve ser menor que duas vezes o tamanho da camada de entrada.

2.4.3 Explicando a rede neural

Para fins de estudos e melhorias na rede neural, é de extrema importância que se entenda o seu funcionamento e quais parâmetros estão influenciando na tomada de decisão da rede. Devido à necessidade, diversas técnicas para explicar modelos de aprendizado de máquina foram desenvolvidas. Neste trabalho, a ferramenta que será utilizada será a SHAP. Em suma, a SHAP utiliza os valores Shapley, apresentados em (KARLIN; SHAPLEY, 1951) para buscar os atributos que mais influenciaram na decisão da rede neural.

2.5 AVALIAÇÃO DA CLASSIFICAÇÃO

Uma forma de avaliar a qualidade de rede neurais é a matriz de confusão, que reúne medidas que buscam avaliar modelos de inteligência artificial com métodos de treinamento supervisionados, sendo aplicada principalmente na avaliação de modelos de classificação.

Como apresentado em (CASTRO; BRAGA, 2011), modelos de classificação podem obter previsões corretas, Verdadeiras Positivas (TP) ou Verdadeiras Negativas (TN), e falsas, Falsas Positivas (FP) e Falsas Negativas (FN). Sendo assim, as medidas utilizadas na matriz de confusão são: acurácia, precisão, *recall* e F1-score.

Acurácia é a porcentagem de acerto do modelo preditivo, sendo calculada pelo quociente do número de previsões corretas com o número de todas as previsões feitas, como mostra a equação 2.2.

$$accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{2.2}$$

Como mostra (CASTRO; BRAGA, 2011), *recall* é a taxa de verdadeiros positivos, podendo ser calculada pelo quociente da quantidade de verdadeiros positivos com a soma de verdadeiros positivos com falsos positivos, como mostra a equação 2.3.

$$recall = \frac{TP}{TP + FN} \tag{2.3}$$

A precisão é uma medida que mostra a qualidade de predição de uma certa classe, podendo ser calculada pela fórmula na equação 2.4.

$$precision = \frac{TP}{TP + FP} \tag{2.4}$$

Por fim, F1-score reúne as métricas *recall* e precisão, fazendo uma média harmônica entre elas, como mostra a equação 2.5.

$$F1score = 2 \cdot \frac{precision \times recall}{precision + recall} \tag{2.5}$$

Uma vez que o modelo utilizado para previsão neste trabalho é classificatório, a matriz de confusão será importante para avaliar a qualidade da rede neural desenvolvida.

2.6 TRABALHOS CORRELATOS

Ao se tratar da utilização de aprendizado de máquina para prever resultados de esportes em geral, torna-se possível fazer um paralelo entre esportes mais tradicionais com os eSports. Isso porque, em muitos casos, ambas as modalidades possuem uma vasta variedade de atributos a serem estudados, alguns deles como, pontuação, tempo de jogo, desempenho de um indivíduo do time, entre outros.

(FIALHO; MANH aES; TEIXEIRA, 2019) fazem uma análise bibliográfica de estudos que utilizaram modelos de aprendizado de máquina para prever resultados de esportes como futebol, futebol americano e basquete. É concluído, então, que a utilização de aprendizado de máquina para prever resultados de esportes em geral é válida, podendo resultar em boas acurácias quando utilizado com bons *datasets*.

No estudo de (DO et al., 2021), é utilizado rede neural para prever um time vitorioso na fase de *picks* e *bans*, analisando apenas partidas não profissionais de *League of Legends*. Neste estudo, o modelo utilizado obteve uma acurácia de 75%. Chegou-se também a conclusão que jogadores que utilizavam campeões nos quais tinham mais experiência, tiveram mais chances de vencer a partida.

Em (COSTA et al., 2021) foram utilizados algoritmos de aprendizado de máquina para o mesmo objetivo de (DO et al., 2021). A análise também se basou na fase de *picks* e *bans*, porém, utilizando partidas de campeonatos profissionais e oficiais. As metodologias utilizadas foram: Regressão Logística, Árvores de Decisão, *Naive Bayes*, *k-Nearest Neighbors*, *RandomForest*, *Support Vector Machines*. Os resultados obtidos pelo algoritmos de Regressão Logística sugerem que o problema é linearmente separável. O parâmetro utilizado para avaliar a performance geral das metodologias usadas foi o **AUC**, sigla em inglês de *area under a receiver operating characteristic (ROC) curve*. Os modelos utilizando Random Forest e Linear Regression obtiveram uma AUC de 0.97, mostrando que é possível prever resultados de partidas profissionais utilizando apenas dados anteriores à partida.

Já (SILVA; PAPPA; CHAIMOWICZ, 2018) utilizaram redes neurais recorrentes para prever um time vencedor durante a partida. Foram analisados os tempos entre 0 a 25 minutos de partida, tendo como base dados como ouro de cada campeão, objetivos conquistados por cada equipe entre outras informações possíveis de coletas em tempo de partida. Nos tempos entre 10 a 15 minutos, a rede neural desenvolvida obteve uma acurácia de 76.29%. Já nos tempos de 5 a 10 minutos, a acurácia foi de 68.69%.

Por fim, em (SOUZA, 2017), foram utilizados os algoritmos de *Random Forest* e Regressão Logística com os mesmos objetivo de previsão de (DO et al., 2021) e (SILVA; PAPPA; CHAIMOWICZ, 2018). Foram utilizadas partidas do *elo desafiante* (divisão dos 200 melhores jogadores de cada região) e estudados os intervalos de 0 a 10 minutos, 10 a 20 minutos, 20 a 30 minutos, 30 até o final da partida e o tempo inteiro da partida. No intervalo de 0 a 10 minutos, a melhor acurácia obtida pelo algoritmo de *Random Forest* foi de 70.66%. Já a Regressão Logística proporcionou uma melhor acurácia de 70.94%. No intervalo de 20 a 30 minutos, a *Random Forest* obteve uma melhor acurácia de 78.83%, enquanto a Regressão Logística proporcionou uma melhor acurácia de 78.96%.

3 DESENVOLVIMENTO

3.1 BASE DE DADOS

A base de dados foi obtida da plataforma (ELIXIR, 2023), que reúne dados diversos sobre *League of Legends* desde 2015. A base estudada contém partidas de diversas ligas do ano de 2023, até o mês de abril. Foi necessária uma transformação nesta base, pois algumas partidas estavam com dados faltantes. Além disso, foi possível reestruturar a base para um formato tabular, com cada linha da tabela contendo uma partida e cada coluna apresentando um atributo da partida em específico. Esse formato foi necessário para facilitar o carregamento dos dados na rede neural. Os algoritmos utilizados para o tratamento dos dados foram escritos em Python utilizando a biblioteca nativa CSV, e podem ser encontrados nos anexos A e B.

Os atributos utilizados para a rede neural de cada time (azul e vermelho) nos estudos de 10 minutos foram:

- Ouro de cada jogador aos 10 minutos;
- XP de cada jogador aos 10 minutos;
- *Creep Score* de cada jogador aos 10 minutos;
- Diferença de ouro de cada jogador para o jogador de mesmo papel no time oponente aos 10 minutos;
- Diferença de XP de cada jogador para o jogador de mesmo papel no time oponente aos 10 minutos;
- Diferença de *Creep Score* de cada jogador para o jogador de mesmo papel no time oponente aos 10 minutos;
- Abates de cada jogador aos 10 minutos;
- Assistência de cada jogador aos 10 minutos;
- Mortes de cada jogador aos 10 minutos.

Para a rede neural de 15 minutos, foram usados os mesmos atributos, porém considerando o tempo de 15 minutos.

3.2 AMBIENTE COMPUTACIONAL

O ambiente de trabalho escolhido foi o Google Colab[1], que disponibiliza um ambiente de desenvolvimento na nuvem com 12,7 GB de RAM e 107 GB de disco. Além disso, foi utilizada a linguagem Python na versão 3.7. O repositório com as redes neurais se encontram no Github[2].

Como mencionado em capítulos anteriores, a rede neural utilizada será a MultiLayer Perceptron. É possível modelar o estudo como um problema de classificação com dois grupos possíveis. Um grupo será composto por partidas em que o time azul venceu e terá como valor 1. Já outro grupo será composto por partidas nas quais o grupo vermelho venceu e terá o valor 0. Este tipo de modelagem torna redes do tipo MLP úteis para as classificações.

A biblioteca Sklearn, apresentada em (PEDREGOSA et al., 2011) foi utilizada para: execução da Multilayer Perceptron, separar a base de dados em testes e treinamento, normalização de dados e cálculo de acurácia da rede neural. Além disso, a biblioteca Pandas, apresentada em (MCKINNEY et al., 2010), foi utilizada para ler o arquivo CSV que continha os dados de cada partida.

3.3 PROCESSO DE TREINAMENTO

Para treinar as redes neurais e avaliar resultados, as base de dados foram divididas em trinta por cento para testes e setenta por cento para treinamento. A classe **MLPClassifier** do Sklearn foi utilizada para treinar as redes neurais. Para cálculo de acurácia, a função **accuracy_score** da mesma biblioteca foi utilizada. Já o algoritmo de otimização de pesos utilizado foi o gradiente descendente estocástico (sgd, na sigla utilizada como parâmetro), explicado no capítulo 2.

A função de ativação escolhida foi a *Relu*, que possui fórmula apresentada na Equação 3.1

$$f(x) = \max(0, x) \tag{3.1}$$

Neste trabalho foram utilizadas duas planilhas com dados, uma com dados de 10 minutos e outra com dados de 15 minutos. O processo de treinamento é o mesmo para os as duas redes neurais, alterando apenas a base de dados a ser estudada. A Figura 4 apresenta um gráfico BPMN do processo de treinamento e análise das redes neurais.

Em um primeiro momento é necessária a importação das bibliotecas que serão utilizadas no experimento, como é mostrado na Figura 5:

Com as bibliotecas importadas, é possível usar o Pandas para importar a base de dados a ser estudada. É necessário, também, criar um array de strings com as colunas

[1] Disponível em: https://colab.research.google.com
[2] Disponível em: https://github.com/Brenopcosta/mlp-lol-analysis

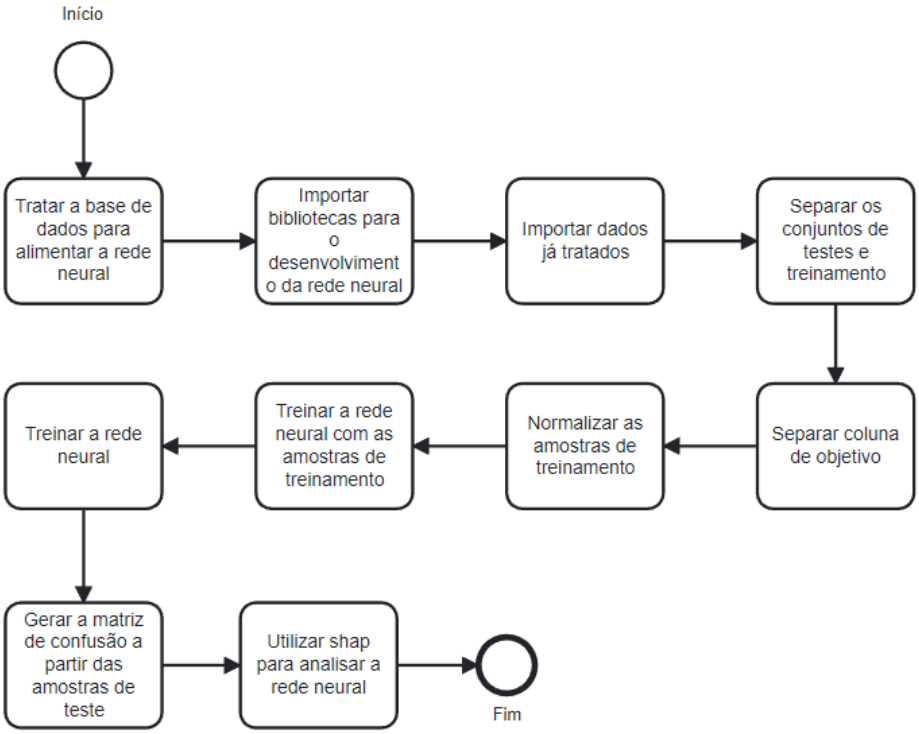

Figura 4 – Gráfico BPMN do processo de desenvolvimento e estudo da rede neural.

```
#Importando bibliotecas

from sklearn.neural_network import MLPClassifier
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report

import numpy as np
import pandas as pd
```

Figura 5 – Bibliotecas importadas no ambiente.

que serão utilizadas no estudo. A coluna *result* será utilizada para indicar o vencedor da partida. A Figura 6 e a Figura 7, a seguir, apresentam o processo de leitura e separação de colunas para os tempos de 10 e 15 minutos, respectivamente.

Como apresentado anteriormente, para cada base de dados foi utilizado trinta por cento das partidas para testes e setenta por cento para treinamento das respectivas redes neurais. Para isto foi utilizada a função **train_test_split** da biblioteca Sklearn como mostra a Figura 8.

Para melhorar a qualidade do treinamento, é interessante a normalização dos dados de entrada, logo, foi utilizada a função **StandardScaler** do Sklearn, como mostra a Figura 9.

Com a base de dados carregada, normalizada e com os conjuntos de treinamento e testes bem definidos, pode-se iniciar o treinamento da rede neural. Será utilizada a

```
lolGameDataframe = pd.read_csv('/content/drive/MyDrive/TCC/inputData10min.csv')
colNames = ['b_top_goldAt10', 'r_top_goldAt10', 'b_jg_goldAt10', 'r_jg_goldAt10', 'b_mid_goldAt1
print(len(colNames))
attributes = lolGameDataframe[colNames]
target = lolGameDataframe['result']
```

```
90
```

Figura 6 – Separação dos atributos a serem estudados no tempo de 10 minutos.

```
lolGameDataframe = pd.read_csv('/content/drive/MyDrive/TCC/inputData15min.csv')
colNames = ['b_top_goldAt15', 'r_top_goldAt15', 'b_jg_goldAt15', 'r_jg_goldAt15', 'b_mid_goldA
print(len(colNames))
attributes = lolGameDataframe[colNames]
target = lolGameDataframe['result']
```

```
90
```

Figura 7 – Separação dos atributos a serem estudados no tempo de 15 minutos.

```
test_size = 0.3
random_state = 0

x_train, x_test, y_train, y_test = train_test_split(attributes,
                                                    target,
                                                    test_size = test_size,
                                                    random_state = random_state)
```

Figura 8 – Separação de amostra para testes e treinamento.

função **MLPClassifier** como mostrada nas figuras Figura 10 e Figura 11.

3.4 RESULTADOS E ACURÁCIA DA REDE NEURAL

O número de neurônios na camada de entrada foi noventa para ambas as redes. Ou seja, um neurônio de entrada para cada atributo a ser estudado. Já na camada de saída, foi necessário apenas um neurônio, uma vez que há apenas um atributo de saída neste estudo.

Para a quantidade de neurônios em camadas ocultas, foram testados os valores entre 1 a 180, baseados nos princípios apresentados em (PANCHAL et al., 2011). A partir dos testes, para o tempo de 10 minutos, a melhor acurácia obtida foi com 2 neurônios na camada oculta. Já no tempo de 15 minutos, 11 neurônios na camada oculta resultaram melhor eficácia. As figuras Figura 12 e Figura 13 possibilitam uma melhor visualização da testagem.

Para o cálculo da acurácia e de outros dados importantes para o estudo da eficácia da rede neural, foi utilizada a função **classification_report** do Sklearn. Sendo assim, para

```
sc = StandardScaler()

sc.fit(x_train)

x_train_std = sc.transform(x_train)

sc.fit(x_test)
x_test_std = sc.transform(x_test)
```

Figura 9 – Normalização de conjuntos para treino e testes.

```
[ ]  classifier = MLPClassifier(hidden_layer_sizes=(2),
                                max_iter=800000,
                                activation = 'relu',
                                solver='sgd',
                                random_state=1
                                )
     classifier.fit(x_train_std,
             y_train)

     y_pred = classifier.predict(x_test_std)

     print("accuracia: {0:.2f}%".format(accuracy_score(y_test, y_pred)*100))
```

Figura 10 – Treinamento da rede neural de 10 minutos.

```
[ ]  classifier = MLPClassifier(hidden_layer_sizes=(11),
                                max_iter=800000,
                                activation = 'relu',
                                solver='sgd',
                                random_state=1)
     classifier.fit(x_train_std,
             y_train)

     y_pred = classifier.predict(x_test_std)

     print("accuracia: {0:.2f}%".format(accuracy_score(y_test, y_pred)*100))
```

Figura 11 – Treinamento da rede neural de 15 minutos.

o tempo de 10 minutos, a rede neural obteve uma acurácia de 72,03% como mostrado no gráfico de confusão na Figura 14. O tempo para padronização dos dados e treinamento da rede neural foi de 1 segundo.

Já no tempo de 15 minutos de jogo, a acurácia foi de 74.68% como apresentado no gráfico de confusão na Figura 15. Assim como na rede neural de 10 minutos, o tempo de

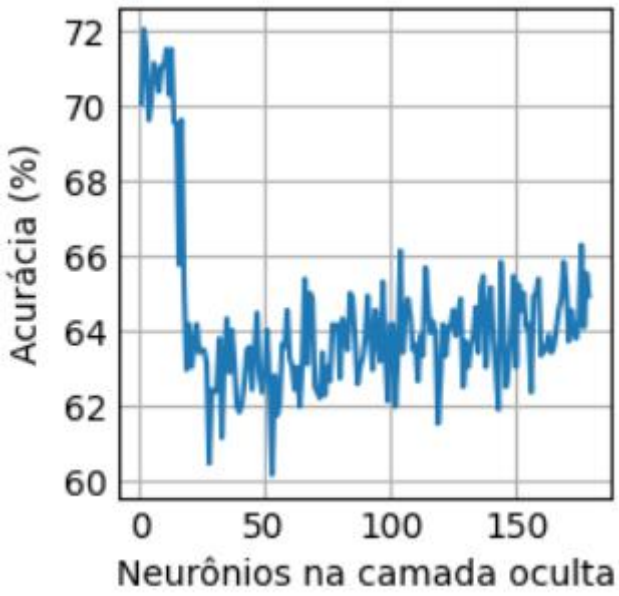

Figura 12 – Gráfico quantidade de neurônios na camada oculta X acurácia da rede neural, em 10 minutos de partida.

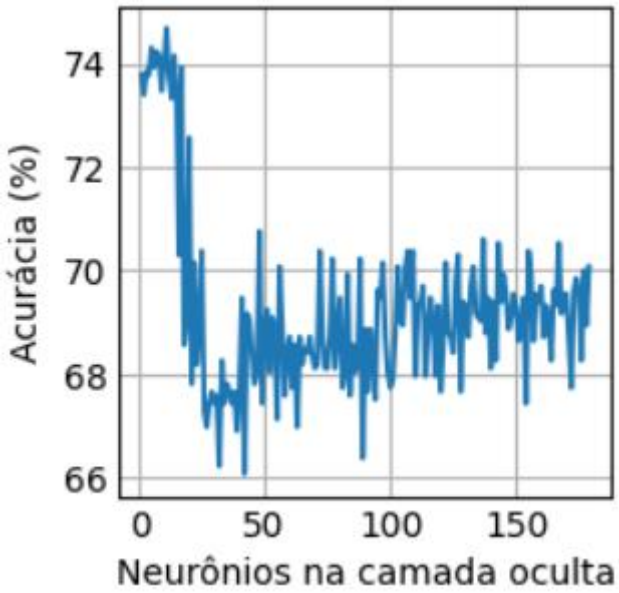

Figura 13 – Gráfico quantidade de neurônios na camada oculta X acurácia da rede neural, em 15 minutos de partida.

padronização dos dados e treinamento também foi de 1 segundo.

É importante observar que no tempo de 10 minutos, a precisão e o recall diminuíram em relação aos mesmos atributos nos tempos de 15 minutos.

```
print(classification_report(y_test, y_pred))
```

	precision	recall	f1-score	support
0	0.68	0.72	0.70	604
1	0.75	0.72	0.74	719
accuracy			0.72	1323
macro avg	0.72	0.72	0.72	1323
weighted avg	0.72	0.72	0.72	1323

Figura 14 – Resultados da rede neural no tempo de 10 minutos.

```
print(classification_report(y_test, y_pred))
```

	precision	recall	f1-score	support
0	0.72	0.73	0.73	604
1	0.77	0.76	0.76	719
accuracy			0.75	1323
macro avg	0.74	0.75	0.75	1323
weighted avg	0.75	0.75	0.75	1323

Figura 15 – Resultados da rede neural no tempo de 15 minutos.

4 EXPLICANDO A REDE NEURAL

Com a função **summary_plot** da biblioteca SHAP apresentada no capítulo 2, foi possível analisar os atributos que mais influenciaram na decisão das redes neurais. A Figura 16 mostra a importação da biblioteca no ambiente e a utilização das funções necessárias para a execução da análise.

```
import shap

shap.initjs()

explainer = shap.KernelExplainer(classifier.predict_proba, x_train)
shap_values = explainer.shap_values(x_test.iloc[0:92,:], nsamples=92)
shap.summary_plot(shap_values, x_test.iloc[0:92,:])
```

Figura 16 – Utilização da biblioteca SHAP para analisar a rede neural.

Na rede neural que avalia a partida no tempo de 10 minutos, a partir da análise apresentada na Figura 17, os três atributos que mais influenciaram foram **r_jg_csdiffat10**, ou seja, diferença de abate de *minions* e monstros da selva entre os *junglers*. Em segunda e terceira posição vem os atributos **r_top_golddiffat10** e **r_bot_botcsdiffat10**, que são a diferença de ouro entre os *tops* e de abate de *minions* entre os *bots* aos 10 minutos respectivamente.

Junglers recebem a maior parte de sua experiência e ouro abatendo monstros da selva. Como explicado no capítulo 1, estes abates serão necessários para subir de nível, liberando novas habilidades e possibilitando a compra de novos itens. Na maioria dos casos, as habilidades de um campeão usado na *jungle* servem para imobilizar, dificultar a mobilidade ou chegar no inimigo mais rápido (com um salto, aumento de velocidade, etc..). Além disso, nos minutos iniciais do jogo, os *junglers* realizam o *gank*, que consiste em aparecer de surpresa em uma rota buscando abater um campeão inimigo. Um *gank* bem-sucedido no início do jogo pode impactar positivamente no andamento da partida, uma vez que o campeão aliado que realizou o abate durante o processo (seja o *jungler* ou não) estará com vantagem em relação ao time inimigo. Pode-se, enfim, concluir que, o *jungler* que abater mais monstros da selva terá mais experiência e ouro, logo, estará mais apto a ajudar sua equipe nos primeiros minutos de partida, podendo impactar positivamente no andamento do jogo. Os campeões de papel *top* muitas vezes fazem um equilíbrio entre dano e utilidade, na maioria dos casos, tem facilidade de segurar muito dano ou até mesmo chegar com facilidade em um campeão inimigo. Normalmente o primeiro item de um *top* já influencia o suficiente para que o campeão impacte no jogo.

Já na rede neural que avalia a partida aos 15 minutos, na Figura 18 é possível observar que o atributo que mais influenciou foi o **r_bot_golddiffat15**, ou seja, a dife-

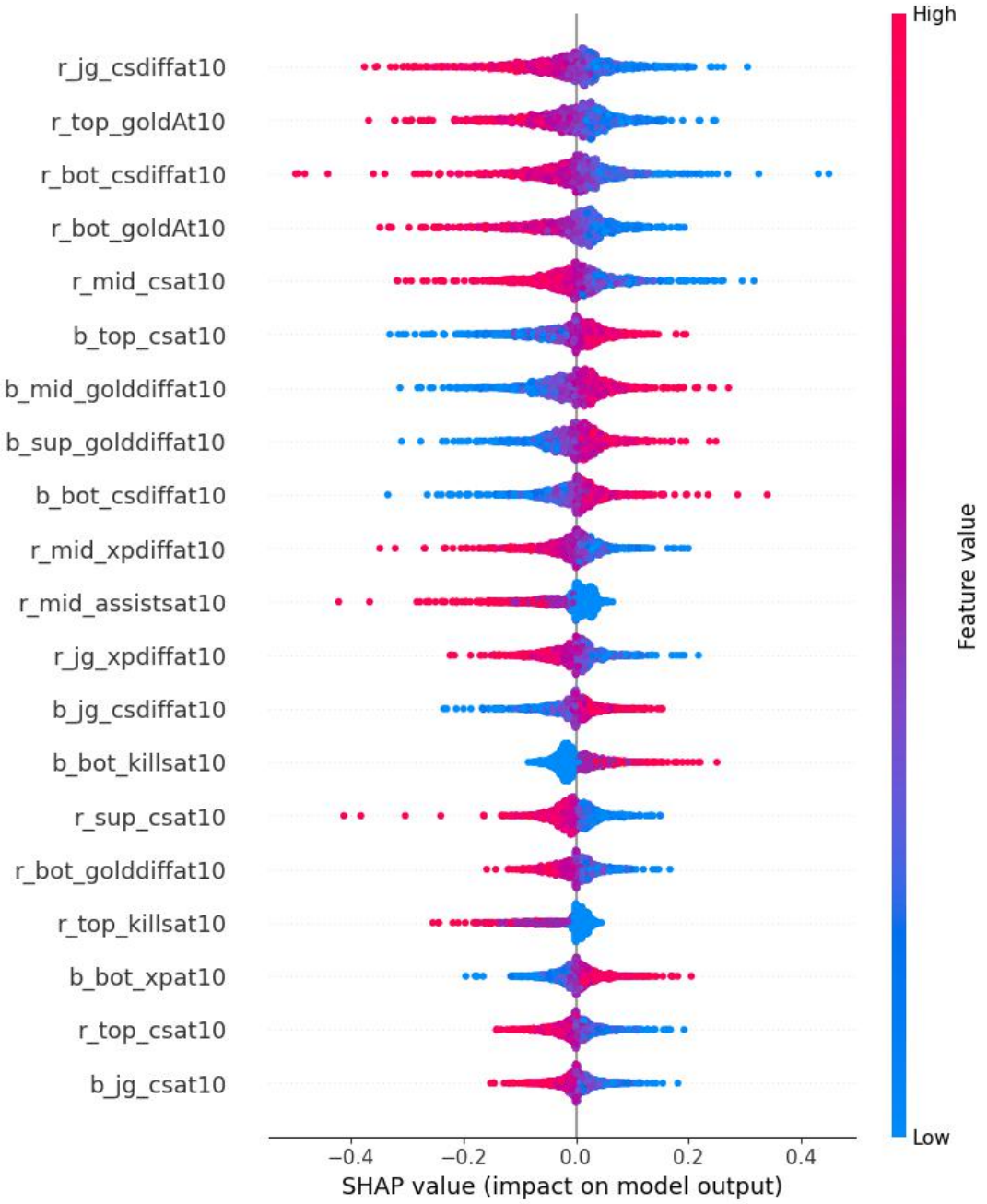

Figura 17 – Análise do SHAP para o tempo de 10 minutos.

rença de ouro entre os *bots* dos dois times, na perspectiva do time vermelho, seguido do **b_jg_csdiffat15**, que é a diferença de abates de *minions* entre os *junglers* na perspectiva do time azul. Na maioria dos casos, os campeões da lane *mid* e *bot* são os responsáveis por causar mais dano no time inimigo durante a partida. Estes são campeões com alto potencial de ataque, porém baixa defesa. Nos minutos iniciais da partida, não possuem ataque nem defesa o suficiente para impactar no andamento do jogo, dependendo na maioria dos casos do *jungler* e do suporte aliado para realizar abates. Sendo assim, neste período, estes focam em abater *minions* do time inimigo para subir de nível e comprar

itens. Contudo, próximo aos 15 minutos de jogo, estes campeões já possuem nível, itens, e consequentemente, dano para impactar na partida. Conclui-se então que, nos 15 minutos de partida, o *bot* ou *mid* que tiver mais ouro (consequentemente com mais itens), terá uma vantagem em relação ao time inimigo, podendo influenciar positivamente no rumo da partida. Conclui-se também que a diferença de abates de *minions* entre os *junglers* ainda é relevante para o andamento da partida.

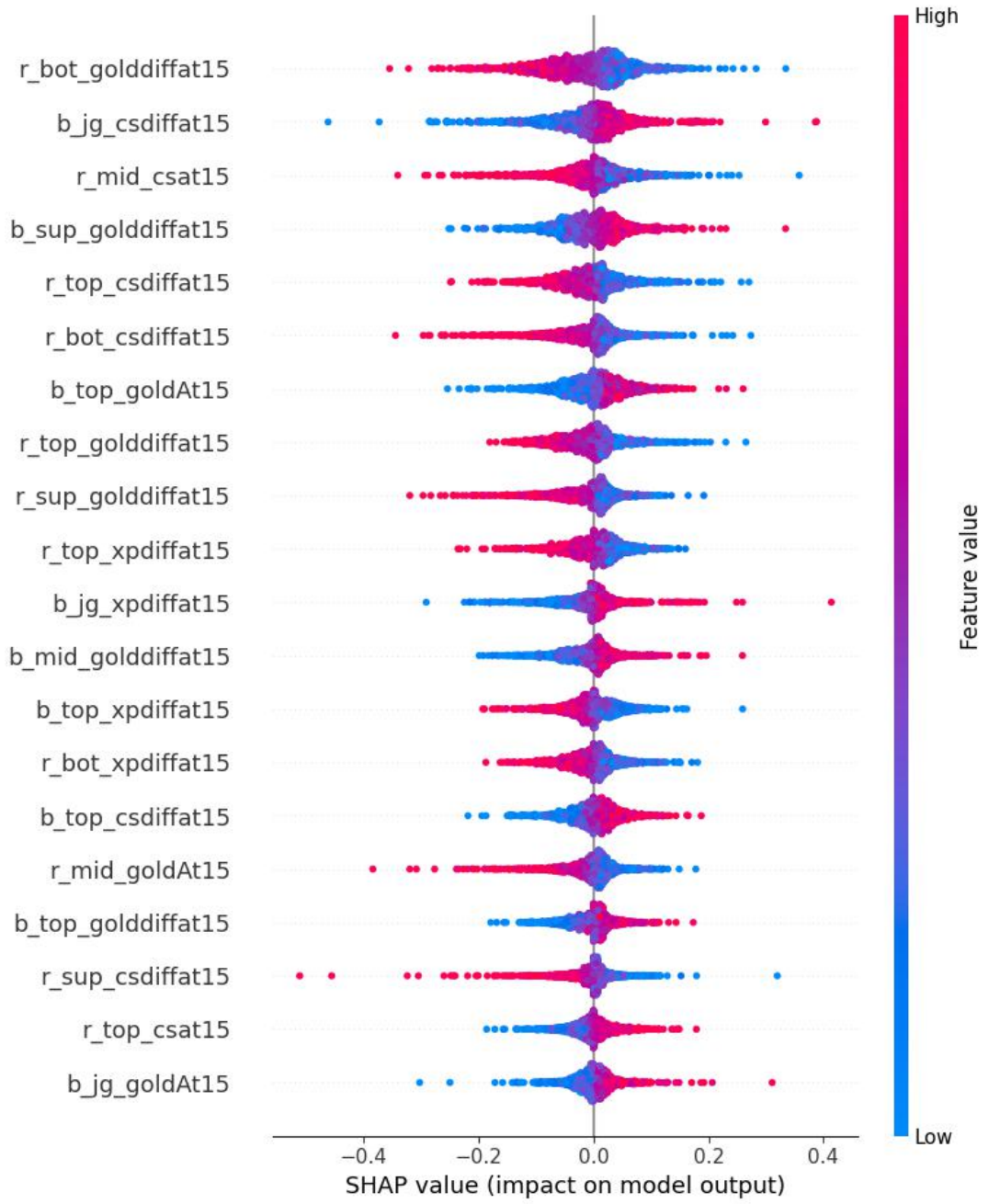

Figura 18 – Análise do SHAP para o tempo de 15 minutos.

5 CONCLUSÃO

As redes neurais MLP desenvolvidas obtiveram uma acurácia de 72,03% para os 10 minutos de partida e 74,68% para os 15 minutos. Comparando os resultados com outros trabalhos correlatos, a acurácia obtida nos 10 minutos teve uma média um pouco acima a obtida no mesmo tempo em (SILVA; PAPPA; CHAIMOWICZ, 2018) e em (SOUZA, 2017), porém aos 15 minutos inferior a (SILVA; PAPPA; CHAIMOWICZ, 2018), mostrando que as redes neurais desenvolvidas também são interessantes para análises de partidas de League of Legends, uma vez que as acurácias obtidas são semelhantes as de outros trabalhos. Além disso, graças ao algoritmo em python desenvolvido para transformação da base de (ELIXIR, 2023), foi possível a criação de uma nova base para análise. De qualquer forma, foi possível através da ferramenta SHAP concluir que o *jungler* que abater mais monstros da selva nos 15 primeiros minutos de jogo terá maior probabilidade de levar a vitória para o seu time. Isso porque o mesmo terá mais nível e maior potencial de obter um *gank* bem-sucedido em relação ao *jungler* do time oposto. Já nos 15 primeiros minutos, o *bot* e *mid* que tiverem mais ouro e *minions* abatidos, consequentemente, finalizarão os itens mais cedo que o oponente de mesmo papel, e irão impactar mais na partida, aumentando, assim, a probabilidade de vitória.

5.1 ATENDIMENTO AOS OBJETIVOS ESPECÍFICOS

1. No Capítulo 2 foram analisados trabalhos correlatos de aprendizado de máquina nos esportes em geral e em *League of Legends*. Com base nestes trabalhos, foi possível entender como algoritmos de aprendizado de máquina são eficientes na previsão por um vitorioso em uma partida;

2. No Capítulo 3 foram desenvolvidas duas redes neurais *Multilayer Perceptron* que visam prever um vitorioso em uma partida de *League of Legends* nos tempos de 10 e 15 minutos, com as acurácias de 72,03% e 74,68% respectivamente;

3. Já no Capítulo 4, com a ferramenta SHAP, foi possível identificar a importância dos atributos **r_jg_csdiffat10** e **r_top_golddiffat10** no tempo de 10 minutos e dos atributos **r_bot_golddiffat15** e **b_jg_csdiffat15** no tempo de 15 minutos, além disso foi feita uma análise de suas importâncias.

4. Na conclusão deste trabalho, foi apresentada uma comparação com trabalhos correlatos.

5.2 TRABALHOS FUTUROS

É sugerido para trabalhos futuros estudos com redes neurais que englobam dados pré-partida, uma vez que como apontado em (DO et al., 2021) e (COSTA et al., 2021), esses dados são relevantes e somados aos obtidos durante a partida, podendo apresentar resultados mais precisos. Outro trabalho interessante é avaliar mais tempos de partida, uma vez que alguns campeões impactam mais no final da partida. Junto a isso, utilizar mais atributos como tipos de dragões conquistados até um determinado tempo, itens comprados por jogador e, por fim, analisar como o desempenho de um jogador no ano em questão influencia no resultado da partida.

REFERÊNCIAS

BLOCK, H. A review of "perceptrons: An introduction to computational geometry". **Information and Control**, v. 17, n. 5, p. 501–522, 1970. ISSN 0019-9958. Disponível em: https://www.sciencedirect.com/science/article/pii/S0019995870904092.

CASTRO, C.; BRAGA, A. Supervised learning with imbalanced data sets: An overview. **Sba: Controle & Automação Sociedade Brasileira de Automatica**, v. 22, p. 441–466, 10 2011.

COSTA, L. M. et al. Feature analysis to league of legends victory prediction on the picks and bans phase. In: IEEE, 2021. **2021 IEEE Conference on Games (CoG)**. Copenhagen, Denmark, 2021. p. 01–05.

DO, T. D. et al. Using machine learning to predict game outcomes based on player-champion experience in league of legends. In: ACM, 16., 2021, Montreal, Canadá. **The 16th International Conference on the Foundations of Digital Games (FDG) 2021**. Montreal, Canadá, 2021. p. 1–5.

ELIXIR, O. **Oracle's Elixir**. 2023. Disponível em: https://oracleselixir.com/about. Acesso em: 2023-02-23.

FIALHO, G.; MANH aES, A.; TEIXEIRA, J. P. Predicting sports results with artificial intelligence–a proposal framework for soccer games. **Procedia Computer Science**, Elsevier, v. 164, p. 131–136, 2019.

HEBB, D. O. **The organization of behavior: A neuropsychological theory**. New York: Wiley, 1949. Hardcover. ISBN 0-8058-4300-0.

KARLIN, S.; SHAPLEY, L. S. **Geometry of moment spaces**. RAND Corp Santa Monica CA, 1951.

MCCULLOCH, W. S.; PITTS, W. A logical calculus of the ideas immanent in nervous activity. **The Journal of Symbolic Logic**, Cambridge University Press, v. 9, n. 2, p. 49–50, 1944. Reprinted from Bulletin of mathematical biophysics, vol. 5 (1943), pp. 115–133.

MCKINNEY, W. et al. Data structures for statistical computing in python. In: SciPy, 9., 2010, Austin, Texas. **Proceedings of the 9th Python in Science Conference**. Austin, Texas, 2010. v. 445, p. 51–56.

PANCHAL, G. et al. Behaviour analysis of multilayer perceptrons with multiple hidden neurons and hidden layers. **International Journal of Computer Theory and Engineering**, v. 3, n. 2, p. 332–337, 2011.

PEDREGOSA, F. et al. Scikit-learn: Machine learning in python. **Journal of machine learning research**, v. 12, n. Oct, p. 2825–2830, 2011.

RIOT. **Como Jogar - League of Legends**. 2022. Disponível em: https://www.leagueoflegends.com/pt-br/how-to-play/. Acesso em: 2022-10-29.

ROSENBLATT, F. The perceptron: a probabilistic model for information storage and organization in the brain. **Psychological review**, American Psychological Association, v. 65, n. 6, p. 386, 1958.

SILVA, A. L. C.; PAPPA, G. L.; CHAIMOWICZ, L. Continuous outcome prediction of league of legends competitive matches using recurrent neural networks. **SBC-Proceedings of SBCGames**, p. 2179–2259, 2018.

SILVA, I. da et al. **Artificial Neural Networks: A Practical Course**. Springer International Publishing, 2016. ISBN 9783319431628. Disponível em: https://books.google.com.br/books?id=DL_mDAAAQBAJ.

SOUZA, R. T. d. Aplicaçao de algoritmos classificadores para previsao de vitória em uma partida de league of legends. 2017.

WAGNER, M. G. On the scientific relevance of esports. In: CSREA PRESS, Las Vegas, NV. **International conference on internet computing**. Las Vegas, NV, 2006. p. 437–442. Tradução nossa.

WIDROW, B.; HOFF, M. E. **Adaptive switching circuits**. Stanford, CA: Stanford University, Stanford Electronics Labs, 1960. Disponível em: https://www-isl.stanford.edu/~widrow/papers/c1960adaptiveswitching.pdf. Acesso em: 2023-08-23.

APÊNDICES

APÊNDICE A – CÓDIGO EM PYTHON UTILIZADO PARA TRANSFORMAR DADOS NO TEMPO DE 10 MINUTOS

```
import csv

def main():
    outputDataHeader = []

    colNames = ['b_top_goldAt10', 'r_top_goldAt10', 'b_jg_goldAt10', '
       r_jg_goldAt10', 'b_mid_goldAt10', 'r_mid_goldAt10', '
       b_bot_goldAt10', 'r_bot_goldAt10', 'b_sup_goldAt10', '
       r_sup_goldAt10', 'b_top_xpat10', 'r_top_xpat10', 'b_jg_xpat10', '
       r_jg_xpat10', 'b_mid_xpat10', 'r_mid_xpat10', 'b_bot_xpat10', '
       r_bot_xpat10', 'b_sup_xpat10', 'r_sup_xpat10', 'b_top_csat10', '
       r_top_csat10', 'b_jg_csat10', 'r_jg_csat10', 'b_mid_csat10', '
       r_mid_csat10', 'b_bot_csat10', 'r_bot_csat10', 'b_sup_csat10', '
       r_sup_csat10','b_top_golddiffat10', 'r_top_golddiffat10', '
       b_jg_golddiffat10', 'r_jg_golddiffat10', 'b_mid_golddiffat10', '
       r_mid_golddiffat10', 'b_bot_golddiffat10', 'r_bot_golddiffat10',
       'b_sup_golddiffat10', 'r_sup_golddiffat10', 'b_top_xpdiffat10', '
       r_top_xpdiffat10', 'b_jg_xpdiffat10', 'r_jg_xpdiffat10', '
       b_mid_xpdiffat10', 'r_mid_xpdiffat10', 'b_bot_xpdiffat10', '
       r_bot_xpdiffat10', 'b_sup_xpdiffat10', 'r_sup_xpdiffat10', '
       b_top_csdiffat10',  'r_top_csdiffat10', 'b_jg_csdiffat10', '
       r_jg_csdiffat10', 'b_mid_csdiffat10', 'r_mid_csdiffat10', '
       b_bot_csdiffat10', 'r_bot_csdiffat10', 'b_sup_csdiffat10', '
       r_sup_csdiffat10', 'b_top_killsat10', 'r_top_killsat10', '
       b_jg_killsat10', 'r_jg_killsat10', 'b_mid_killsat10', '
       r_mid_killsat10', 'b_bot_killsat10' ,'r_bot_killsat10', '
       b_sup_killsat10', 'r_sup_killsat10', 'b_top_assistsat10', '
       r_top_assistsat10', 'b_jg_assistsat10', 'r_jg_assistsat10', '
       b_mid_assistsat10', 'r_mid_assistsat10', 'b_bot_assistsat10', '
       r_bot_assistsat10', 'b_sup_assistsat10', 'r_sup_assistsat10', '
       b_top_deathsat10', 'r_top_deathsat10', 'b_jg_deathsat10', '
       r_jg_deathsat10', 'b_mid_deathsat10', 'r_mid_deathsat10', '
       b_bot_deathsat10', 'r_bot_deathsat10', 'b_sup_deathsat10', '
       r_sup_deathsat10', 'result']

    with open('2023_LoL_esports_match_data_from_OraclesElixir.csv', 'r')
        as teste_csv:
        with open('outputData10min.csv','w') as output:
            csvFile =  csv.DictReader(teste_csv, delimiter = ',')
            csvOutputFile = csv.DictWriter(output,delimiter = ',',
               dialect='unix', fieldnames = colNames)
            csvOutputFile.writeheader()
```

```
matchDataRowCounter = 1

matchRowDicitonary = {1: "top", 2:"jg", 3:"mid", 4:"bot", 5:
    "sup"}

matchData = {}

totalMatch = 0

for row in csvFile:
    if matchDataRowCounter == 12:
        matchDataRowCounter = 1
        if row["datacompleteness"] == "complete":
            csvOutputFile.writerow(matchData)
        totalMatch+=1
        matchData = {}
        print(totalMatch)
        continue

    elif matchDataRowCounter == 11:
        if row["datacompleteness"] != "complete":
            matchDataRowCounter+=1
            continue
        matchData["result"] = row["result"]
        matchDataRowCounter+=1
        continue

    elif matchDataRowCounter in matchRowDicitonary.keys():
        if row["datacompleteness"] != "complete":
            matchDataRowCounter+=1
            continue

        matchData["b_"+matchRowDicitonary[
            matchDataRowCounter]+"_goldAt10"] = row["goldat10
            "]
        matchData["r_"+matchRowDicitonary[
            matchDataRowCounter]+"_goldAt10"] = row["
            opp_goldat10"]

        matchData["b_"+matchRowDicitonary[
            matchDataRowCounter]+"_xpat10"] = row["xpat10"]
        matchData["r_"+matchRowDicitonary[
            matchDataRowCounter]+"_xpat10"] = row["opp_xpat10
            "]

        matchData["b_"+matchRowDicitonary[
            matchDataRowCounter]+"_csat10"] = row["csat10"]
```

```
matchData["r_"+matchRowDicitonary[
    matchDataRowCounter]+"_csat10"] = row["opp_csat10
    "]

matchData["b_"+matchRowDicitonary[
    matchDataRowCounter]+"_golddiffat10"] = row["
    golddiffat10"]
matchData["r_"+matchRowDicitonary[
    matchDataRowCounter]+"_golddiffat10"] = int(row["
    golddiffat10"])*-1

matchData["b_"+matchRowDicitonary[
    matchDataRowCounter]+"_xpdiffat10"] = row["
    xpdiffat10"]
matchData["r_"+matchRowDicitonary[
    matchDataRowCounter]+"_xpdiffat10"] = int(row["
    xpdiffat10"])*-1

matchData["b_"+matchRowDicitonary[
    matchDataRowCounter]+"_csdiffat10"] = row["
    csdiffat10"]
matchData["r_"+matchRowDicitonary[
    matchDataRowCounter]+"_csdiffat10"] = int(row["
    csdiffat10"])*-1

matchData["b_"+matchRowDicitonary[
    matchDataRowCounter]+"_killsat10"] = row["
    killsat10"]
matchData["r_"+matchRowDicitonary[
    matchDataRowCounter]+"_killsat10"] = row["
    opp_killsat10"]

matchData["b_"+matchRowDicitonary[
    matchDataRowCounter]+"_assistsat10"] = row["
    assistsat10"]
matchData["r_"+matchRowDicitonary[
    matchDataRowCounter]+"_assistsat10"] = row["
    opp_assistsat10"]

matchData["b_"+matchRowDicitonary[
    matchDataRowCounter]+"_deathsat10"] = row["
    deathsat10"]
matchData["r_"+matchRowDicitonary[
    matchDataRowCounter]+"_deathsat10"] = row["
    opp_deathsat10"]
matchDataRowCounter+=1
```

```
                continue
            else:
                matchDataRowCounter+=1
                continue


main()
```

APÊNDICE B – CÓDIGO EM PYTHON UTILIZADO PARA TRANSFORMAR DADOS NO TEMPO DE 15 MINUTOS

```
import csv

def main():
    outputDataHeader = []

    colNames = ['b_top_goldAt15', 'r_top_goldAt15', 'b_jg_goldAt15', '
       r_jg_goldAt15', 'b_mid_goldAt15', 'r_mid_goldAt15', '
       b_bot_goldAt15', 'r_bot_goldAt15', 'b_sup_goldAt15', '
       r_sup_goldAt15', 'b_top_xpat15', 'r_top_xpat15', 'b_jg_xpat15', '
       r_jg_xpat15', 'b_mid_xpat15', 'r_mid_xpat15', 'b_bot_xpat15', '
       r_bot_xpat15', 'b_sup_xpat15', 'r_sup_xpat15', 'b_top_csat15', '
       r_top_csat15', 'b_jg_csat15', 'r_jg_csat15', 'b_mid_csat15', '
       r_mid_csat15', 'b_bot_csat15', 'r_bot_csat15', 'b_sup_csat15', '
       r_sup_csat15','b_top_golddiffat15', 'r_top_golddiffat15', '
       b_jg_golddiffat15', 'r_jg_golddiffat15', 'b_mid_golddiffat15', '
       r_mid_golddiffat15', 'b_bot_golddiffat15', 'r_bot_golddiffat15',
       'b_sup_golddiffat15', 'r_sup_golddiffat15', 'b_top_xpdiffat15', '
       r_top_xpdiffat15', 'b_jg_xpdiffat15', 'r_jg_xpdiffat15', '
       b_mid_xpdiffat15', 'r_mid_xpdiffat15', 'b_bot_xpdiffat15', '
       r_bot_xpdiffat15', 'b_sup_xpdiffat15', 'r_sup_xpdiffat15', '
       b_top_csdiffat15',  'r_top_csdiffat15', 'b_jg_csdiffat15', '
       r_jg_csdiffat15', 'b_mid_csdiffat15', 'r_mid_csdiffat15', '
       b_bot_csdiffat15', 'r_bot_csdiffat15', 'b_sup_csdiffat15', '
       r_sup_csdiffat15', 'b_top_killsat15', 'r_top_killsat15', '
       b_jg_killsat15', 'r_jg_killsat15', 'b_mid_killsat15', '
       r_mid_killsat15', 'b_bot_killsat15' ,'r_bot_killsat15', '
       b_sup_killsat15', 'r_sup_killsat15', 'b_top_assistsat15', '
       r_top_assistsat15', 'b_jg_assistsat15', 'r_jg_assistsat15', '
       b_mid_assistsat15', 'r_mid_assistsat15', 'b_bot_assistsat15', '
       r_bot_assistsat15', 'b_sup_assistsat15', 'r_sup_assistsat15', '
       b_top_deathsat15', 'r_top_deathsat15', 'b_jg_deathsat15', '
       r_jg_deathsat15', 'b_mid_deathsat15', 'r_mid_deathsat15', '
       b_bot_deathsat15', 'r_bot_deathsat15', 'b_sup_deathsat15', '
       r_sup_deathsat15', 'result']

    with open('2023_LoL_esports_match_data_from_OraclesElixir.csv', 'r')
        as teste_csv:
        with open('outputData15min.csv','w') as output:
            csvFile =  csv.DictReader(teste_csv, delimiter = ',')
            csvOutputFile = csv.DictWriter(output,delimiter = ',',
               dialect='unix', fieldnames = colNames)
            csvOutputFile.writeheader()
```

```
matchDataRowCounter = 1

matchRowDicitonary = {1: "top", 2:"jg", 3:"mid", 4:"bot", 5:
    "sup"}

matchData = {}

totalMatch = 0

for row in csvFile:
    if matchDataRowCounter == 12:
        matchDataRowCounter = 1
        if row["datacompleteness"] == "complete":
            csvOutputFile.writerow(matchData)
        totalMatch+=1
        matchData = {}
        print(totalMatch)
        continue

    elif matchDataRowCounter == 11:
        if row["datacompleteness"] != "complete":
            matchDataRowCounter+=1
            continue
        matchData["result"] = row["result"]
        matchDataRowCounter+=1
        continue

    elif matchDataRowCounter in matchRowDicitonary.keys():
        if row["datacompleteness"] != "complete":
            matchDataRowCounter+=1
            continue

        matchData["b_"+matchRowDicitonary[
            matchDataRowCounter]+"_goldAt15"] = row["goldat15
            "]
        matchData["r_"+matchRowDicitonary[
            matchDataRowCounter]+"_goldAt15"] = row["
            opp_goldat15"]

        matchData["b_"+matchRowDicitonary[
            matchDataRowCounter]+"_xpat15"] = row["xpat15"]
        matchData["r_"+matchRowDicitonary[
            matchDataRowCounter]+"_xpat15"] = row["opp_xpat15
            "]

        matchData["b_"+matchRowDicitonary[
            matchDataRowCounter]+"_csat15"] = row["csat15"]
```

```
matchData["r_"+matchRowDicitonary[
    matchDataRowCounter]+"_csat15"] = row["opp_csat15
    "]

matchData["b_"+matchRowDicitonary[
    matchDataRowCounter]+"_golddiffat15"] = row["
    golddiffat15"]
matchData["r_"+matchRowDicitonary[
    matchDataRowCounter]+"_golddiffat15"] = int(row["
    golddiffat15"])*-1

matchData["b_"+matchRowDicitonary[
    matchDataRowCounter]+"_xpdiffat15"] = row["
    xpdiffat15"]
matchData["r_"+matchRowDicitonary[
    matchDataRowCounter]+"_xpdiffat15"] = int(row["
    xpdiffat15"])*-1

matchData["b_"+matchRowDicitonary[
    matchDataRowCounter]+"_csdiffat15"] = row["
    csdiffat15"]
matchData["r_"+matchRowDicitonary[
    matchDataRowCounter]+"_csdiffat15"] = int(row["
    csdiffat15"])*-1

matchData["b_"+matchRowDicitonary[
    matchDataRowCounter]+"_killsat15"] = row["
    killsat15"]
matchData["r_"+matchRowDicitonary[
    matchDataRowCounter]+"_killsat15"] = row["
    opp_killsat15"]

matchData["b_"+matchRowDicitonary[
    matchDataRowCounter]+"_assistsat15"] = row["
    assistsat15"]
matchData["r_"+matchRowDicitonary[
    matchDataRowCounter]+"_assistsat15"] = row["
    opp_assistsat15"]

matchData["b_"+matchRowDicitonary[
    matchDataRowCounter]+"_deathsat15"] = row["
    deathsat15"]
matchData["r_"+matchRowDicitonary[
    matchDataRowCounter]+"_deathsat15"] = row["
    opp_deathsat15"]
matchDataRowCounter+=1
```

```
                continue
            else:
                matchDataRowCounter+=1
                continue


main()
```

ANEXOS

ANEXO A – ATRIBUTOS USADOS PARA ESTUDO NO TEMPO DE 10 MINUTOS

Nome do atributo	Descrição	Exemplo
b_top_goldAt10	Ouro do campeão no top do time azul aos 10 minutos	5407
r_top_goldAt10	Ouro do campeão no top do time vermelho aos 10 minutos	4659
b_jg_goldAt10	Ouro do campeão na jungle do time azul aos 10 minutos	6974
r_jg_goldAt10	Ouro do campeão na jungle do time vermelho aos 10 minutos	4854
b_mid_goldAt10	Ouro do campeão no mid do time azul aos 10 minutos	6591
r_mid_goldAt15	Ouro do campeão no mid do time vermelho aos 10 minutos	5013
b_bot_goldAt15	Ouro do campeão no bot do time azul aos 10 minutos	5202
r_bot_goldAt15	Ouro do campeão no bot do time vermelho aos 10 minutos	5078
b_sup_goldAt15	Ouro do campeão suporte do time azul aos 10 minutos	3853
r_sup_goldAt15	Ouro do campeão suporte do time vermelho aos 10 minutos	3405
b_top_xpat10	Experiência do campeão no top do time azul aos 10 minutos	7536
r_top_xpat10	Experiência do campeão no top do time vermelho aos 10 minutos	7592
b_jg_xpat10	Experiência do campeão na jungle do time azul aos 10 minutos	8232
r_jg_xpat10	Experiência do campeão na jungle do time vermelho aos 10 minutos	4827
b_mid_xpat10	Experiência do campeão no mid do time azul aos 10 minutos	7827
r_mid_xpat10	Experiência do campeão no mid do time vermelho aos 10 minutos	7473

b_bot_xpat10	Experiência do campeão no bot do time azul aos 10 minutos	5053
r_bot_xpat10	Experiência do campeão no bot do time vermelho aos 10 minutos	4951
b_sup_xpat10	Experiência do campeão suporte do time azul aos 10 minutos	4681
r_sup_xpat10	Experiência do campeão suporte do time vermelho aos 10 minutos	4231
b_top_csat10	Abates a mineons do campeão no top do time azul aos 10 minutos	114
r_top_csat10	Abates a mineons do campeão no top do time vermelho aos 10 minutos	118
b_jg_csat10	Abates a mineons do campeão na jungle do time azul aos 10 minutos	146
r_jg_csat10	Abates a mineons do campeão na jungle do time vermelho aos 10 minutos	84
b_mid_csat10	Abates a mineons do campeão no mid do time azul aos 10 minutos	158
r_mid_csat10	Abates a mineons do campeão no mid do time vermelho aos 10 minutos	143
b_bot_csat10	Abates a mineons do campeão no bot do time azul aos 10 minutos	130
r_bot_csat10	Abates a mineons do campeão no bot do time vermelho aos 10 minutos	120
b_sup_csat10	Abates a mineons do campeão suporte do time azul aos 10 minutos	28
r_sup_csat10	Abates a mineons do campeão suporte do time vermelho aos 10 minutos	25
b_top_golddiffat10	Diferença de ouro entre o top do time azul e o top do time vermelho aos 10 minutos	748
r_top_golddiffat10	Diferença de ouro entre o top do time vermelho e o top do time azul aos 10 minutos	-748
b_jg_golddiffat10	Diferença de ouro entre o jungler do time azul e o jungler do time vermelho aos 10 minutos	2120
r_jg_golddiffat10	Diferença de ouro entre o jungler do time vermelho e o jungler do time azul aos 10 minutos	-2120
b_mid_golddiffat10	Diferença de ouro entre o mid do time azul e o mid do time vermelho aos 10 minutos	1578

r_mid_golddiffat10	Diferença de ouro entre o mid do time vermelho e o mid do time azul aos 10 minutos	-1578
b_bot_golddiffat10	Diferença de ouro entre o bot do time azul e o bot do time vermelho aos 10 minutos	124
r_bot_golddiffat10	Diferença de ouro entre o bot do time vermelho e o bot do time azul aos 10 minutos	-124
b_sup_golddiffat10	Diferença de ouro entre o suporte do time azul e o suporte do time vermelho aos 10 minutos	448
r_sup_golddiffat10	Diferença de ouro entre o suporte do time vermelho e o suporte do time azul aos 10 minutos	-448
b_top_xpdiffat10	Diferença de experiência entre o top do time azul e o top do time vermelho aos 10 minutos	-56
r_top_xpdiffat10	Diferença de experiência entre o top do time vermelho e o top do time azul aos 10 minutos	56
b_jg_xpdiffat10	Diferença de experiência entre o jungler do time azul e o jungler do time vermelho aos 10 minutos	3405
r_jg_xpdiffat10	Diferença de experiência entre o jungler do time vermelho e o jungler do time azul aos 10 minutos	-3405
b_mid_xpdiffat10	Diferença de experiência entre o mid do time azul e o mid do time vermelho aos 10 minutos	354
r_mid_xpdiffat10	Diferença de experiência entre o mid do time vermelho e o mid do time azul aos 10 minutos	-354
b_bot_xpdiffat10	Diferença de experiência entre o bot do time azul e o bot do time vermelho aos 10 minutos	2
r_bot_xpdiffat10	Diferença de experiência entre o bot do time vermelho e o bot do time azul aos 10 minutos	-102
b_sup_xpdiffat10	Diferença de experiência entre o suporte do time azul e o suporte do time vermelho aos 10 minutos	450
r_sup_xpdiffat10	Diferença de experiência entre o suporte do time vermelho e o suporte do time azul aos 10 minutos	-450

b_top_csdiffat10	Diferença de abates a mineons entre o top do time azul e o top do time vermelho aos 10 minutos	-4
r_top_csdiffat10	Diferença de abates a mineons entre o top do time vermelho e o top do time azul aos 10 minutos	4
b_jg_csdiffat10	Diferença de abates a mineons entre o jungler do time azul e o jungler do time vermelho aos 10 minutos	62
r_jg_csdiffat10	Diferença de abates a mineons entre o jungler do time vermelho e o jungler do time azul aos 10 minutos	-62
b_mid_csdiffat10	Diferença de abates a mineons entre o mid do time azul e o mid do time vermelho aos 10 minutos	15
r_mid_csdiffat10	Diferença de abates a mineons entre o mid do time vermelho e o mid do time azul aos 10 minutos	-15
b_bot_csdiffat10	Diferença de abates a mineons entre o bot do time azul e o bot do time vermelho aos 10 minutos	10
r_bot_csdiffat10	Diferença de abates a mineons entre o bot do time vermelho e o bot do time azul aos 10 minutos	-10
b_sup_csdiffat10	Diferença de abates a mineons entre o suporte do time azul e o suporte do time vermelho aos 10 minutos	3
r_sup_csdiffat10	Diferença de abates a mineons entre o suporte do time vermelho e o suporte do time azul aos 10 minutos	-3
b_top_killsat10	Diferença de abates entre o top do time azul e o top do time vermelho aos 10 minutos	2
r_top_killsat10	Diferença de abates entre o top do time vermelho e o top do time azul aos 10 minutos	0
b_jg_killsat10	Diferença de abates entre o jungler do time azul e o jungler do time vermelho aos 10 minutos	3

r_jg_killsat10	Diferença de abates entre o jungler do time vermelho e o jungler do time azul aos 10 minutos	2
b_mid_killsat10	Diferença de abates entre o mid do time azul e o mid do time vermelho aos 10 minutos	2
r_mid_killsat10	Diferença de abates entre o mid do time vermelho e o mid do time azul aos 10 minutos	0
b_bot_killsat10	Diferença de abates entre o bot do time azul e o bot do time vermelho aos 10 minutos	0
r_bot_killsat10	Diferença de abates entre o bot do time vermelho e o bot do time azul aos 10 minutos	1
b_sup_killsat10	Diferença de abates entre o suporte do time azul e o suporte do time vermelho aos 10 minutos	1
r_sup_killsat10	Diferença de abates entre o suporte do time vermelho e o suporte do time azul aos 10 minutos	0
b_top_assistsat10	Diferença de assistências entre o top do time azul e o top do time vermelho aos 10 minutos	0
r_top_assistsat10	Diferença de assistências entre o top do time vermelho e o top do time azul aos 10 minutos	1
b_jg_assistsat10	Diferença de assistências entre o jungler do time azul e o jungler do time vermelho aos 10 minutos	2
r_jg_assistsat10	Diferença de assistências entre o jungler do time vermelho e o jungler do time azul aos 10 minutos	0
b_mid_assistsat10	Diferença de assistências entre o mid do time azul e o mid do time vermelho aos 10 minutos	3
r_mid_assistsat10	Diferença de assistências entre o mid do time vermelho e o mid do time azul aos 10 minutos	0
b_bot_assistsat10	Diferença de assistências entre o bot do time azul e o bot do time vermelho aos 10 minutos	4
r_bot_assistsat10	Diferença de assistências entre o bot do time vermelho e o bot do time azul aos 10 minutos	1
b_sup_assistsat10	Diferença de assistências entre o suporte do time azul e o suporte do time vermelho aos 10 minutos	4

r_sup_assistsat10	Diferença de assistências entre o suporte do time vermelho e o suporte do time azul aos 10 minutos	2
b_top_deathsat10	Diferença de mortes entre o top do time azul e o top do time vermelho aos 10 minutos	1
r_top_deathsat10	Diferença de mortes entre o top do time vermelho e o top do time azul aos 10 minutos	1
b_jg_deathsat10	Diferença de mortes entre o jungler do time azul e o jungler do time vermelho aos 10 minutos	0
r_jg_deathsat10	Diferença de mortes entre o jungler do time vermelho e o jungler do time azul aos 10 minutos	3
b_mid_deathsat10	Diferença de mortes entre o mid do time azul e o mid do time vermelho aos 10 minutos	0
r_mid_deathsat10	Diferença de mortes entre o mid do time vermelho e o mid do time azul aos 10 minutos	0
b_bot_deathsat10	Diferença de mortes entre o bot do time azul e o bot do time vermelho aos 10 minutos	2
r_bot_deathsat10	Diferença de mortes entre o bot do time vermelho e o bot do time azul aos 10 minutos	2
b_sup_deathsat10	Diferença de mortes entre o suporte do time azul e o suporte do time vermelho aos 10 minutos	0
r_sup_deathsat10	Diferença de mortes entre o suporte do time vermelho e o suporte do time azul aos 10 minutos	2
result	Resultado da partida. 1 para o time azul e 0 para o time vermelho.	1

ANEXO B – ATRIBUTOS USADOS PARA ESTUDO NO TEMPO DE 15 MINUTOS

Nome do atributo	Descrição	Exemplo
b_top_goldAt15	Ouro do campeão no top do time azul aos 15 minutos	5407
r_top_goldAt15	Ouro do campeão no top do time vermelho aos 15 minutos	4659
b_jg_goldAt15	Ouro do campeão na jungle do time azul aos 15 minutos	6974
r_jg_goldAt15	Ouro do campeão na jungle do time vermelho aos 15 minutos	4854
b_mid_goldAt15	Ouro do campeão no mid do time azul aos 15 minutos	6591
r_mid_goldAt15	Ouro do campeão no mid do time vermelho aos 15 minutos	5013
b_bot_goldAt15	Ouro do campeão no bot do time azul aos 15 minutos	5202
r_bot_goldAt15	Ouro do campeão no bot do time vermelho aos 15 minutos	5078
b_sup_goldAt15	Ouro do campeão suporte do time azul aos 15 minutos	3853
r_sup_goldAt15	Ouro do campeão suporte do time vermelho aos 15 minutos	3405
b_top_xpat15	Experiência do campeão no top do time azul aos 15 minutos	7536
r_top_xpat15	Experiência do campeão no top do time vermelho aos 15 minutos	7592
b_jg_xpat15	Experiência do campeão na jungle do time azul aos 15 minutos	8232
r_jg_xpat15	Experiência do campeão na jungle do time vermelho aos 15 minutos	4827
b_mid_xpat15	Experiência do campeão no mid do time azul aos 15 minutos	7827
r_mid_xpat15	Experiência do campeão no mid do time vermelho aos 15 minutos	7473

b_bot_xpat15	Experiência do campeão no bot do time azul aos 15 minutos	5053
r_bot_xpat15	Experiência do campeão no bot do time vermelho aos 15 minutos	4951
b_sup_xpat15	Experiência do campeão suporte do time azul aos 15 minutos	4681
r_sup_xpat15	Experiência do campeão suporte do time vermelho aos 15 minutos	4231
b_top_csat15	Abates a mineons do campeão no top do time azul aos 15 minutos	114
r_top_csat15	Abates a mineons do campeão no top do time vermelho aos 15 minutos	118
b_jg_csat15	Abates a mineons do campeão na jungle do time azul aos 15 minutos	146
r_jg_csat15	Abates a mineons do campeão na jungle do time vermelho aos 15 minutos	84
b_mid_csat15	Abates a mineons do campeão no mid do time azul aos 15 minutos	158
r_mid_csat15	Abates a mineons do campeão no mid do time vermelho aos 15 minutos	143
b_bot_csat15	Abates a mineons do campeão no bot do time azul aos 15 minutos	130
r_bot_csat15	Abates a mineons do campeão no bot do time vermelho aos 15 minutos	120
b_sup_csat15	Abates a mineons do campeão suporte do time azul aos 15 minutos	28
r_sup_csat15	Abates a mineons do campeão suporte do time vermelho aos 15 minutos	25
b_top_golddiffat15	Diferença de ouro entre o top do time azul e o top do time vermelho aos 15 minutos	748
r_top_golddiffat15	Diferença de ouro entre o top do time vermelho e o top do time azul aos 15 minutos	-748
b_jg_golddiffat15	Diferença de ouro entre o jungler do time azul e o jungler do time vermelho aos 15 minutos	2120
r_jg_golddiffat15	Diferença de ouro entre o jungler do time vermelho e o jungler do time azul aos 15 minutos	-2120
b_mid_golddiffat15	Diferença de ouro entre o mid do time azul e o mid do time vermelho aos 15 minutos	1578

r_mid_golddiffat15	Diferença de ouro entre o mid do time vermelho e o mid do time azul aos 15 minutos	-1578
b_bot_golddiffat15	Diferença de ouro entre o bot do time azul e o bot do time vermelho aos 15 minutos	124
r_bot_golddiffat15	Diferença de ouro entre o bot do time vermelho e o bot do time azul aos 15 minutos	-124
b_sup_golddiffat15	Diferença de ouro entre o suporte do time azul e o suporte do time vermelho aos 15 minutos	448
r_sup_golddiffat15	Diferença de ouro entre o suporte do time vermelho e o suporte do time azul aos 15 minutos	-448
b_top_xpdiffat15	Diferença de experiência entre o top do time azul e o top do time vermelho aos 15 minutos	-56
r_top_xpdiffat15	Diferença de experiência entre o top do time vermelho e o top do time azul aos 15 minutos	56
b_jg_xpdiffat15	Diferença de experiência entre o jungler do time azul e o jungler do time vermelho aos 15 minutos	3405
r_jg_xpdiffat15	Diferença de experiência entre o jungler do time vermelho e o jungler do time azul aos 15 minutos	-3405
b_mid_xpdiffat15	Diferença de experiência entre o mid do time azul e o mid do time vermelho aos 15 minutos	354
r_mid_xpdiffat15	Diferença de experiência entre o mid do time vermelho e o mid do time azul aos 15 minutos	-354
b_bot_xpdiffat15	Diferença de experiência entre o bot do time azul e o bot do time vermelho aos 15 minutos	102
r_bot_xpdiffat15	Diferença de experiência entre o bot do time vermelho e o bot do time azul aos 15 minutos	-102
b_sup_xpdiffat15	Diferença de experiência entre o suporte do time azul e o suporte do time vermelho aos 15 minutos	450
r_sup_xpdiffat15	Diferença de experiência entre o suporte do time vermelho e o suporte do time azul aos 15 minutos	-450

b_top_csdiffat15	Diferença de abates a mineons entre o top do time azul e o top do time vermelho aos 15 minutos	-4
r_top_csdiffat15	Diferença de abates a mineons entre o top do time vermelho e o top do time azul aos 15 minutos	4
b_jg_csdiffat15	Diferença de abates a mineons entre o jungler do time azul e o jungler do time vermelho aos 15 minutos	62
r_jg_csdiffat15	Diferença de abates a mineons entre o jungler do time vermelho e o jungler do time azul aos 15 minutos	-62
b_mid_csdiffat15	Diferença de abates a mineons entre o mid do time azul e o mid do time vermelho aos 15 minutos	15
r_mid_csdiffat15	Diferença de abates a mineons entre o mid do time vermelho e o mid do time azul aos 15 minutos	-15
b_bot_csdiffat15	Diferença de abates a mineons entre o bot do time azul e o bot do time vermelho aos 15 minutos	10
r_bot_csdiffat15	Diferença de abates a mineons entre o bot do time vermelho e o bot do time azul aos 15 minutos	-10
b_sup_csdiffat15	Diferença de abates a mineons entre o suporte do time azul e o suporte do time vermelho aos 15 minutos	3
r_sup_csdiffat15	Diferença de abates a mineons entre o suporte do time vermelho e o suporte do time azul aos 15 minutos	-3
b_top_killsat15	Diferença de abates entre o top do time azul e o top do time vermelho aos 15 minutos	2
r_top_killsat15	Diferença de abates entre o top do time vermelho e o top do time azul aos 15 minutos	0
b_jg_killsat15	Diferença de abates entre o jungler do time azul e o jungler do time vermelho aos 15 minutos	3

r_jg_killsat15	Diferença de abates entre o jungler do time vermelho e o jungler do time azul aos 15 minutos	2
b_mid_killsat15	Diferença de abates entre o mid do time azul e o mid do time vermelho aos 15 minutos	2
r_mid_killsat15	Diferença de abates entre o mid do time vermelho e o mid do time azul aos 15 minutos	0
b_bot_killsat15	Diferença de abates entre o bot do time azul e o bot do time vermelho aos 15 minutos	0
r_bot_killsat15	Diferença de abates entre o bot do time vermelho e o bot do time azul aos 15 minutos	1
b_sup_killsat15	Diferença de abates entre o suporte do time azul e o suporte do time vermelho aos 15 minutos	1
r_sup_killsat15	Diferença de abates entre o suporte do time vermelho e o suporte do time azul aos 15 minutos	0
b_top_assistsat15	Diferença de assistências entre o top do time azul e o top do time vermelho aos 15 minutos	0
r_top_assistsat15	Diferença de assistências entre o top do time vermelho e o top do time azul aos 15 minutos	1
b_jg_assistsat15	Diferença de assistências entre o jungler do time azul e o jungler do time vermelho aos 15 minutos	2
r_jg_assistsat15	Diferença de assistências entre o jungler do time vermelho e o jungler do time azul aos 15 minutos	0
b_mid_assistsat15	Diferença de assistências entre o mid do time azul e o mid do time vermelho aos 15 minutos	3
r_mid_assistsat15	Diferença de assistências entre o mid do time vermelho e o mid do time azul aos 15 minutos	0
b_bot_assistsat15	Diferença de assistências entre o bot do time azul e o bot do time vermelho aos 15 minutos	4
r_bot_assistsat15	Diferença de assistências entre o bot do time vermelho e o bot do time azul aos 15 minutos	1
b_sup_assistsat15	Diferença de assistências entre o suporte do time azul e o suporte do time vermelho aos 15 minutos	4

r_sup_assistsat15	Diferença de assistências entre o suporte do time vermelho e o suporte do time azul aos 15 minutos	2
b_top_deathsat15	Diferença de mortes entre o top do time azul e o top do time vermelho aos 15 minutos	1
r_top_deathsat15	Diferença de mortes entre o top do time vermelho e o top do time azul aos 15 minutos	1
b_jg_deathsat15	Diferença de mortes entre o jungler do time azul e o jungler do time vermelho aos 15 minutos	0
r_jg_deathsat15	Diferença de mortes entre o jungler do time vermelho e o jungler do time azul aos 15 minutos	3
b_mid_deathsat15	Diferença de mortes entre o mid do time azul e o mid do time vermelho aos 15 minutos	0
r_mid_deathsat15	Diferença de mortes entre o mid do time vermelho e o mid do time azul aos 15 minutos	0
b_bot_deathsat15	Diferença de mortes entre o bot do time azul e o bot do time vermelho aos 15 minutos	2
r_bot_deathsat15	Diferença de mortes entre o bot do time vermelho e o bot do time azul aos 15 minutos	2
b_sup_deathsat15	Diferença de mortes entre o suporte do time azul e o suporte do time vermelho aos 15 minutos	0
r_sup_deathsat15	Diferença de mortes entre o suporte do time vermelho e o suporte do time azul aos 15 minutos	2
result	Resultado da partida. 1 para o time azul e 0 para o time vermelho.	1