Universidade Católica do Salvador
Bacharelado em Engenharia de Software

Danilo Silva Gonçalves

e-Coach: Um sistema de recomendação de equipes e personagens de League of Legends utilizando redes neurais multicamadas

Salvador
2019

Danilo Silva Gonçalves

e-Coach: Um sistema de recomendação de equipes e personagens de League of Legends utilizando redes neurais multicamadas

Trabalho de Conclusão de Curso apresentado à Universidade Católica do Salvador como parte dos requisitos necessários para a obtenção do Título de Engenheiro de Software.
Orientador: Prof. Me. André Brasil Vieira Wyzykowski

Universidade Católica do Salvador
Bacharelado em Engenharia de Software

Salvador
2019

Danilo Silva Gonçalves

e-Coach: Um sistema de recomendação de equipes e personagens de League of Legends utilizando redes neurais multicamadas

Trabalho de Conclusão de Curso apresentado à Universidade Católica do Salvador como parte dos requisitos necessários para a obtenção do Título de Engenheiro de Software.

Comissão Examinadora

Prof. Me. André Brasil Vieira Wyzykowski
Universidade Católica do Salvador
Orientador

Prof. Me. Marcelo Indio dos Reis
Universidade Católica do Salvador

Prof. Ma. Pâmella Arielle Brito de Aquino
Universidade Católica do Salvador

Salvador, 31 de julho de 2019

Dedico este trabalho a todos os envolvidos que ajudaram a concluir.

Agradecimentos

Primeiramente, a Deus pôr ter me dado saúde e força para superar as dificuldades.

Dedico este trabalho a minha mãe Vilma, minha tia Marivalda, meu amigo Zamparoni e ao restante da minha família pelo apoio dado e por ter acompanhado e acreditado em toda minha jornada até o ensino superior.

Agradeço a todos os meus colegas de sala, principalmente Hudson, Deveza, Vinicius e Bruno por todo o acompanhamento no ambiente acadêmico. Aos professores do curso, que repassam todos os dias seus conhecimentos para a formação de bons profissionais.

Agradeço aos meus amigos por estarem sempre presentes, dando conselhos, me apoiando nas decisões e pelo divertimento diário. Aos meus colegas de Discord da Ordem, onde se dispuseram a ajudar em quaisquer problemas que viessem surgir, mesmos distantes, na elaboração deste trabalho.

Agradeço ao meu orientador André Wyzykowski pelas suas sugestões, correções, seu conhecimento e experiência em relação ao assunto para que esse trabalho fosse realizado.

"Não há vergonha em errar, vergonha é não ter dado tudo de si por ter medo de errar."
(Endou Mamoru)

Resumo

A modalidade dos esportes eletrônicos têm estado cada vez mais presente na sociedade, lotando estádios em diversas partes do mundo e conquistando cada vez mais adeptos por onde seus eventos ocorrem. Entre investimentos e premiações milionárias, uma carreira de *cyber* atleta se tornou almejada como objetivo dos jovens da comunidade *gamer*, além de incentivar outras empresas no empreendimento dessa área. Neste sentido, este estudo demonstra a implementação de um algoritmo aplicando redes *Multi Layer Perceptron*, utilizando como base de dados os resultados das partidas ranqueadas de League of Legends. Além disso, explica a execução de experimentos com jogadores em partidas reais com o uso do algoritmo e comparando-os com partidas sem o uso do algoritmo. Sendo assim, tornando possível a realização da predição de qual equipe e personagens que melhor se encaixam para a partida na obtenção da vitória.

Palavras-Chave: League of Legends, Esporte Eletrônico, Aprendizado de Máquina, Inteligência Artificial.

Abstract

The mode of electronic sports has been increasingly present in society, crowding stadiums in different parts of the world and conquering more and more supporters by where their events take place. Between investment and millionaire awards, a career as a cyber athlete has become a goal of young people in the gamer community, as well as encouraging other companies in the area. In this sense, this study demonstrates the implementation of an algorithm applying Multi Layer Perceptron networks, using as a database the results of ranked matches of League of Legends. In addition, it explains the execution of experiments with players in real matches using the algorithm and comparing them to matches without using the algorithm. Thus, making possible the realization of the prediction of which team and characters that best fit for the match in obtaining victory.

Keywords: *League of Legends*, *Electronic Sport*, *Machine Learning*, *Artificial Intelligence.*

Lista de figuras

Lista de tabelas

Lista de Siglas e Abreviaturas

DOTA2	*Defense of the Ancients 2*
LOL	*League of Legends*
CBLOL	*Campeonato Brasileiro de League of Legends*
SVM	*Support Vector Machine*
KNN	*K-nearest Neighbors Algorithm*
MOBA	*Multiplayer Online Battle Arena*
F2P	*Free To Play*
BOT	*Robot*
ARAM	*All Random All Mid*
MD10	*Melhor de 10*
MD3	*Melhor de 3*
MD5	*Melhor de 5*
MD2	*Melhor de 2*
MD1	*Melhor de 1*
MSI	*Mid-Season Invitation*
API	*Application Programming Interface*
IDE	*Integrated Development Environment*
GPU	*Graphics Process Unit*
RAM	*Random Access Memory*
CNTK	*Microsoft Cognitive Toolkit*
RNA	*Rede Neural Artificial*
JSON	*JavaScript Object Notation*
MPL	*Multilayer Perceptron*
AMA	*Abates, mortes e assistências*
GAN	*Generative Adversarial Networks*
cGAN	*Conditional Generative Adversarial Network*

Sumário

1 Introdução

A modalidade esportiva virtual, conhecida como esporte eletrônico ou *e-Sports*, vêm atraindo os olhares do público, principalmente entre os jovens. Consiste no confronto entre equipes, representados por um ou mais jogadores, em uma competição de jogos virtuais.

A realização de suas competições com sólida estrutura física, organizacional - palco, iluminação, merchandising - além de ampla utilização atrativa de tecnologia - efeitos visuais, sons, fogos de artifício, etc. - com premiações que alcançam milhões de dólares.

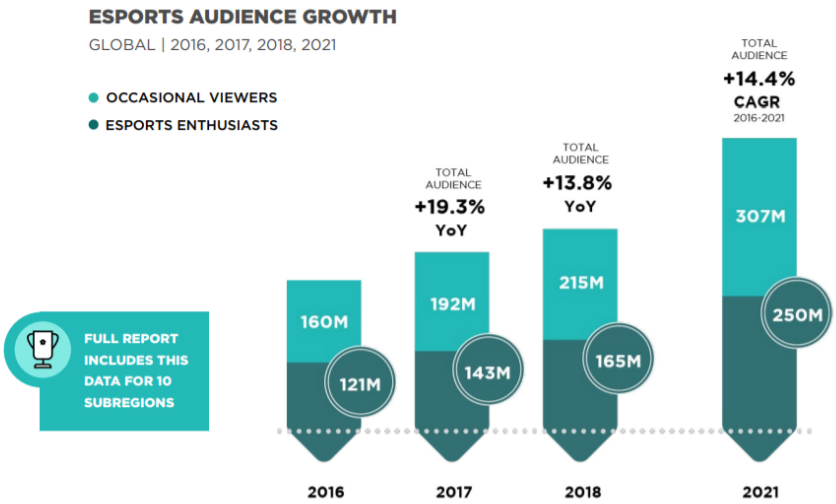
Multidões de torcedores lotam estádios de futebol para assistir e apoiar sua equipe favorita. Toda a infraestrutura desenvolvida para comportar e agradar esse público tende a ser muito bem organizada. As parcerias efetuadas com emissoras de televisão sobre os direitos de exibição de imagem dos jogos foram importantes para o crescimento do número de espectadores.

Mesmo quem não pode prestigiar o evento pessoalmente, não são dele excluídos, porque as competições são transmitidas para todo o mundo através de diversas plataformas de vídeo, conseguindo bater recordes de audiência. Os torcedores e pessoas que desejam conhecer a modalidade acompanham o evento a partir de suas casas, bares, salas de cinemas ou qualquer outro local que exibida os jogos.

Devido a sua visibilidade conquistada, muitas empresas passaram a investir e patrocinar esse mercado, tornando a bonificação maior. A maior premiação de *e-Sports* (SPORTV, 2018), até o momento, ocorreu no *The International* de 2018, campeonato mundial de *Defense of the Ancients 2* (DOTA2), o qual distribuiu mais de 25 milhões de dólares em premiações, dos quais cerca de 11 milhões apenas para o vencedor.

Em 2018, a Newzoo (NEWZOO, 2018), empresa dedicada à inteligência de mercado no setor de games e *e-Sports*, elaborou um relatório prospectivo acerca do negócio envolvendo o esporte eletrônico. O estudo mostra que em 2021 (Figura 1), o nível de audiência chegará a 557 milhões, dos quais 250 milhões são considerados entusiastas e os demais apreciadores ocasionais.

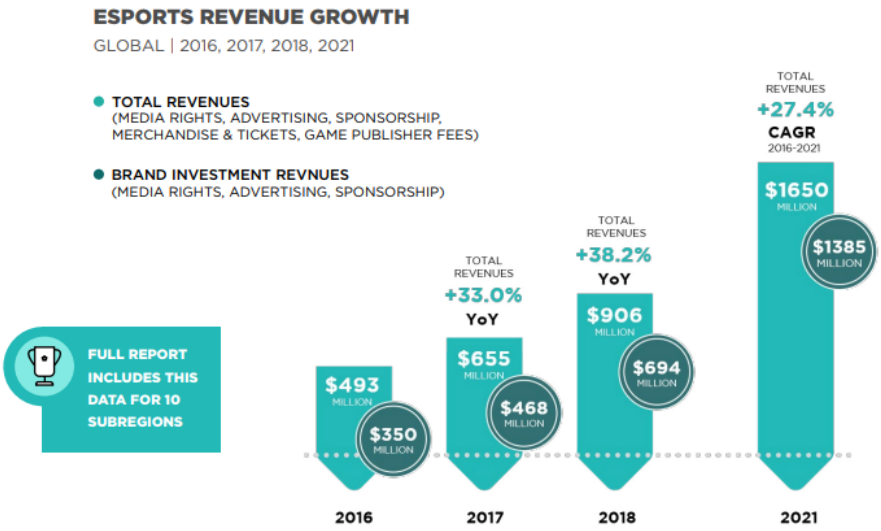
Figura 1 – Crescimento de audiência do esporte eletrônico até 2021.



Fonte: (NEWZOO, 2018)

A Newzoo também apontou neste estudo, a quantidade de receita que possa ser gerada com esse mercado de competições até 2021. Foi suposto que o montante pode alcançar 1.650 milhões de dólares (Figura 2).

Figura 2 – Crescimento de receita do esporte eletrônico até 2021.



Fonte: (NEWZOO, 2018)

Assim como qualquer outro esporte tradicional, no universo *e-Sports* existem equipes competitivas que são compostas por atletas e comissão técnica para conduzir treinamentos e a organização das participações durante o decorrer da temporada. Como no futebol e outros esportes de equipe, há a chamada "janela de transferências de atletas", momento em que as equipes se reformulam através da dispensa ou contratação tanto de novos atletas

quanto de membros da comissão técnica. Em alguns casos, novos talentos que emergiram na temporada e que foram descobertos por "olheiros", acabam por serem inseridos no cenário profissional durante essas transações.

Durante as negociações, alguns dos *cyber* atletas assinam contratos de milhões de dólares, devido ao rendimento conquistado ao longo de sua carreira. Segundo o *eSports Earnings*, existem jogadores que já tiveram contratos com cifras acima de três milhões de dólares (ESPORTSEARNING, 2019).

Tabela 1 – Os cincos atletas com maior rendimento global em competições

Posição	Apelido do Jogador	Nome do Jogador	Total (No geral)
1	KuroKy	Kuro Takhasomi	$ 3,740,477.75
2	Miracle-	Amer Al-Barkawi	$ 3,313,886.88
3	MinD_ContRoL	Ivan Ivanov	$ 3,080,961.36
4	Matumbaman	Lasse Urpalainen	$ 3,080,665.64
5	UNiVeRsE	Saahil Arora	$ 2,996,603.47

Fonte: (ESPORTSEARNING, 2019)

Este multimilionário mercado, o League of Legends (Seção 2.2), conhecido popularmente como LOL, é o mais jogados do mundo (NEWZOO, 2019) e um dos grandes responsáveis por alavancar o cenário competitivo virtual na sociedade. Em 2014, a Riot Games, empresa desenvolvedora do LOL, reportou que 27 milhões de pessoas estavam jogando-o diariamente (FISHER, 2014).

Seus campeonatos são realizados em diversas partes do mundo e seus principais atletas tornaram-se grandes celebridades internacionais. No Brasil ocorre o Campeonato Brasileiro de League of Legends, abreviadamente chamado de CBLOL, no qual o time vencedor recebe uma premiação em dinheiro e ganha a chance de disputar o mundial. O evento CBLOL pode ser visto na Figura 3.

Figura 3 – Final do CBLOL 2018.

Fonte: (RIOT GAMES, 2018c)

Este mundo de investimentos e entretenimento para o desenvolvimento e evolução no *e-Sports* têm agradado muitos jovens e fazendo com que a profissão de *cyber* atleta seja ainda mais almejada pelos admiradores dos *games*.

Um estudo feito por SUN (SUN, 2017), avalia as motivações para se jogar League of Legends e as relaciona aos grupos etários, sexos e grupos de frequências. Da amostra com 111 entrevistados entre homens e mulheres entre 18 até 45 anos ou mais, conclui que cerca de 65,1% jogam LOL por mais de 6 horas por semana. Esta amostra foi analisada à luz das seguintes motivações: *achievement* (um estilo desafiador), *socialization* (voltado para relacionamentos no jogo) e *immersion* (aventurar-se no universo). Como um dos resultados, concluiu que no fator *achievement* houve diferença significativa entre jogadores com 18 a 21 anos e de 22 a 24 anos, ou seja, a vontade de "ser o melhor" é maior entre os jovens.

A escalada para se tornar um atleta profissional começa dentro de jogo. O jogador deve ficar entre os melhores do *ranking*, isto é, jogar muitas partidas ranqueadas para subir de categoria. "Costumamos olhar sempre a *solo queue*, só temos olhos para os *Challengers*", disse Hugo Tristão, um dos diretores de equipe competitivas de League of Legends em entrevista (UOL JOGOS, 2016). Além disso, a participação em campeonatos amadores também pode dar visibilidade ao jogador.

A destreza do jogador deve estar sendo sempre aprimorada. Segundo estudos feitos por ZHENGXING et. al. (ZHENGXING et al., 2017), foram utilizados vários modelos preditivos baseados na decomposição das habilidades do jogador aplicados em dois MOBA's conhecidos, o LOL e DOTA2. Três fatores estão presente no LOL: *player base skill* (habilidade básica de jogador), *champion base skill* (habilidade básica de campeão) e *champion-player specific skill* (habilidade específica de jogador com o campeão). Os

resultados obtidos a partir desta análise para cada jogador específico, possibilita que os mesmos possam aprender a montar sua composição de personagens, utilizando as três fontes de análise indicadas, uma vez que seria possível identificar as vantagens e desvantagens quanto às habilidades. Os jogadores podem até mesmo fazer prognóstico aproximado sobre a partida, logo após concluídas as escolhas das 10 personagens, pelos jogadores.

No entanto, alguns jogadores não conseguem melhorar seu desempenho de jogo para alcançar o cenário competitivo. Em alguns casos, os atletas que já estão inseridos neste meio de competições, não recebem por parte de sua comissão técnica estudos detalhados e consistentes o suficiente que possam servir de base para uma correta montagem de estratégia para desempenhar seu trabalho.

Diante disso, pergunta-se: é possível encontrar uma solução que auxilie na superação deste fator limitante, de modo a aprimorar a performance dos jogadores e profissionais, para a otimização de seus objetivos? Este trabalho pretende justamente, como se verá abaixo, apontar um caminho para tal.

1.1 Aplicabilidade e Motivação

Para melhorar o desempenho dos jogadores de League of Legends, é necessário que uma série de fatores os esteja favorecendo dentro de jogo, tais como abates, ouro conquistado, estruturas destruídas, etc., porém, antes de começar a realizar esses feitos é necessário que jogue com um personagem, por ele escolhido. Esta escolha é determinante para que o *player* tenha bom desempenho e consequente sucesso na conquista dos objetivos do jogo. Mas como fazer a escolha certa? A seguir indica-se um caminho.

1.2 Objetivos

O objetivo deste trabalho é desenvolver um algoritmo capaz de recomendar automaticamente as melhores opções de equipes e personagens para uma partida de League of Legends, com o intuito de aumentar a chance de vitória, dando resposta satisfatória à pergunta acima. Para isso, este objetivo geral precisa ser desdobrado em passos, que se constituem em objetivos específicos.

- Objetivos específicos:

    1. Pré-processamento dos *datasets* de partidas de League of Legends;
    2. Elaborar um modelo preditivo de uma rede neural multicamadas;
    3. Submeter o algoritmo implementado ao teste real com jogadores de League of Legends;

4. Fazer uma análise comparativa entre os resultados de desempenho alcançados pelos jogadores quando usando o algoritmo, com o alcançado sem o uso do mesmo.

2 Fundamentação Teórica

Alguns cientistas da computação já fizeram trabalhos envolvendo algoritmos de *machine learning* e dados de League of Legends.

YIN (YIN, 2018) tinha como objetivo em seu projeto prever os resultados de uma partida antes de seu início. Os algoritmos que ele utilizou para fazer a comparação de sensibilidade, especificidade e validação foram: *Support Vector Machine* (SVM), *Neural Net*, *Random Forest*, *Gradient Boosting* e *Logistic Regression*. Para realização do experimento, utilizou uma quantidade certa de amostras que seriam suficientes para aplicar em cada algoritmo. Cada amostra foi separada em 15 características especificas para estatísticas entre personagem e jogador coletadas da Riot Games API e Champion.gg. O resultado obtido por YIN, apontou que o time vencedor da partida não era decidido nem na seleção de personagens e nem nos acontecimentos aleatórios que poderiam ocorrer durante o jogo, concluindo portanto que a utilização dos algoritmos usados não chegaram aos resultados previstos, frustrando sua hipótese.

Outro trabalho bem semelhante foi o de SANTOS (SANTOS, 2018), que desenvolveu um sistema web de recomendação para formação de um time completo de League of Legends, desenvolvido em linguagem de programação Go. Os dados foram coletados da Riot Games API, porém utilizou uma biblioteca chamada Python Riot-Watcher que opera comandos em Python para reunir as informações. Para seu projeto foi utilizado *K-nearest Neighbors Algorithm* (KNN) para realizar a predição dos campeões, a serem utilizados. Ele fez dois tipos de recomendação:

1. *Intercept Recommendation*: segundo este procedimento, calculam-se as interceptações realizadas por todas as equipes e retorna ao usuário uma quantidade K de listas de recomendações dos campeões possíveis e mais apropriados para a equipe. O próprio autor aponta que um dos problemas com este sistema é que não há como determinar qual das listas geradas é a mais efetiva, ficando a decisão a critério do usuário.

2. *Serendipitous Recommendation* (recomendação por serendipidade[1]): é uma recomendação aleatória, segundo a qual, depois de calcular os itens são geradas uma quantidade K de listas de recomendações e a partir destas são geradas outras. Seria então feita a medição do número de ocorrências que um campeão aparece entre as listas. Ao fim, seriam apontados os cinco campeões que mais apareceram como resultado os processos descritos.

[1] Aquilo que acontece ou é descoberto por acaso, de modo imprevisto, inesperado.

Estes projetos comentados acima estão relacionados com os objetivos deste trabalho, além da utilização da ferramenta Riot Games API que está presente em ambos. Para execução e implementação deste trabalho, foi elaborado uma nova forma de abordagem, com o uso de algoritmo diferente.

2.1 Esportes Eletrônicos

PEREIRA (PEREIRA, 2014), conceitua jogo eletrônico como sendo uma projeção que necessita de uma plataforma para ser exibida e manipulada através de um console. Quando os primeiros *arcades* começaram a ser vendidos, o entretenimento virtual ganhou espaço nas residências e nos espaços comerciais que disponibilizavam várias máquinas, cujos nomes delas derivavam: "fliperamas", onde a juventude se reunia para jogar.

Os fliperamas ficavam lotados, novos jogadores surgiam e a tendência era crescer com o decorrer do tempo. Novos jogos e novas plataformas eram lançadas para a renovação do mercado. Esse período ficou conhecido como Época de Ouro do Arcade e então surgiram as primeiras competições.

> *"Em 1983 foi organizado o primeiro 'Campeonato Mundial de Videogames', em Dallas nos EUA (Kennedy, 1983); os recordes de videogames ganharam notoriedade e foram publicados no Guiness; e no mesmo ano a Twin Galaxies formou o U.S. National Video Game Team, que disputou a primeira competição de fato internacional de videogames com um time formado por britânicos."*
>
> (PEREIRA, 2014)

Os *arcades* foram importantes para o meio de veiculação dos jogos eletrônicos, porém, foi a entrada dos computadores pessoais e a internet que tornou-se possível a jogabilidade *online*. O surgimento das lan houses era a novidade no fim da década de 1990, o que abriu a possibilidade de se jogar em equipe, ou contra uma pessoa em outra parte do planeta. Novas empresas começaram a surgir no mercado, desenvolvendo novos jogos e organizando campeonatos para incentivo de seus jogadores.

Mesmo sendo considerado algo novo na sociedade, o *e-Sports* já conquistou multidões e envolve fortunas, como se disse acima, o que fez com que países tais como China, Estados Unidos e Coreia do Sul, reconhecem os *cyber* atletas e pessoas que atuam na área do esporte eletrônico, como profissões (ESPORTS NEWS, 2019) e algumas universidades já disponibilizam bolsas de estudos para jogadores profissionais (CANALTECH, 2016). Os times de futebol, basquete e outras modalidades desportivas já estão investindo na área *gamer* e participando de competições oficiais.

Figura 4 – Campeonato de Counter-Strike Global Offensive

Fonte: (MEDIUM, 2018)

O Comitê Organizador dos Jogos do Sudeste Asiático anunciou, em 2018, que os esportes eletrônicos farão parte de suas modalidades em 2019 e o LOL é uma das cinco selecionadas pela organização do evento (RIOT GAMES, 2018d).

2.2 League of Legends

League of Legends é um jogo *Multiplayer Online Battle Arena* (MOBA), desenvolvido pela Riot Games e lançado em 2009. A empresa surgiu em 2006, em Los Angeles, nos Estados Unidos, fundada por Brandon "Ryze" Beck e Marc "Tryndamere" Merril, cujos apelidos foram usados para nominar os primeiros personagens do jogo.

No universo de LOL, os jogadores são chamados de invocadores que controlam um personagem, nomeados de "campeões". No início, o jogo constava com apenas 40 personagens e, atualmente, possui mais de 140 campeões. Sua licença de utilização é *Free to Play* (F2P), livre para baixar e jogar.

A própria empresa se preocupa em atrair novos usuários e tentar ajudar o *noob* (iniciante) a aprender a jogar tendo criado, inclusive, uma plataforma de ensino que atende tanto os novatos quanto os jogadores frequentes em busca de aperfeiçoamento, chamada Central de Aprendizado (RIOT GAMES, 2018b).

2.2.1 Campeões, mapas e roles

Campeão é o nome dado ao personagem no mundo de League of Legends os quais são controlados pelos invocadores. Além disso, os campeões contam com sua própria

fábula para serem lidas, possibilitando enredos e narrativas ao jogo, criando afinidade entre campeões e invocadores.

Cada campeão possui um conjunto único de habilidades e uma *role* (papel, função). Existem 6 funções:

- Atiradores: causam dano físico contínuo nas lutas e possuem capacidade de ataques a longa distância;
- Assassinos: causam muito dano em único alvo e possuem grande mobilidade para se movimentarem-se no mapa;
- Tanques: possuem muita resistência e protegem os outros jogadores;
- Lutadores: capazes de golpes corporais, possuem muita resistência e causam danos consideráveis;
- Magos: especialistas em dano mágico; possuem controle de grupo para desestabilizar os times adversários;
- Suportes: personagens versáteis, que podem usar habilidades de *buff* (apoio) aos aliados (cura, escudos, velocidade de movimento, entre outros) ou *debuff* ao inimigo (lentidão, atordoamento, imobilização, entre outros).

O mapa (cenário) de League of Legends é chamado de *Summoner's Rift* (vale dos invocadores), campo de batalha onde se enfrentam a equipe azul e vermelha. Ao observar o mapa da Figura 5, vemos que existem duas extremidades que são as bases dos times; o lado inferior do mapa é o território do time azul e a parte superior, pertence ao time vermelho.

Summoner's Rift possui três rotas: *Top* (Rota Superior), *Mid* (Rota do Meio) e *Bot* (Rota Inferior). A área que fica entre as rotas é chamada de *Jungle* (Selva) povoada de monstros que, quando abatidos, dão bônus ao jogador.

As rotas, ou *lanes* o termo em inglês, possuem linhas de estruturas que se repetem em ambos os territórios, como em um espelho e que funcionam como fortificações de defesa no caminho até chegar à base adversária.

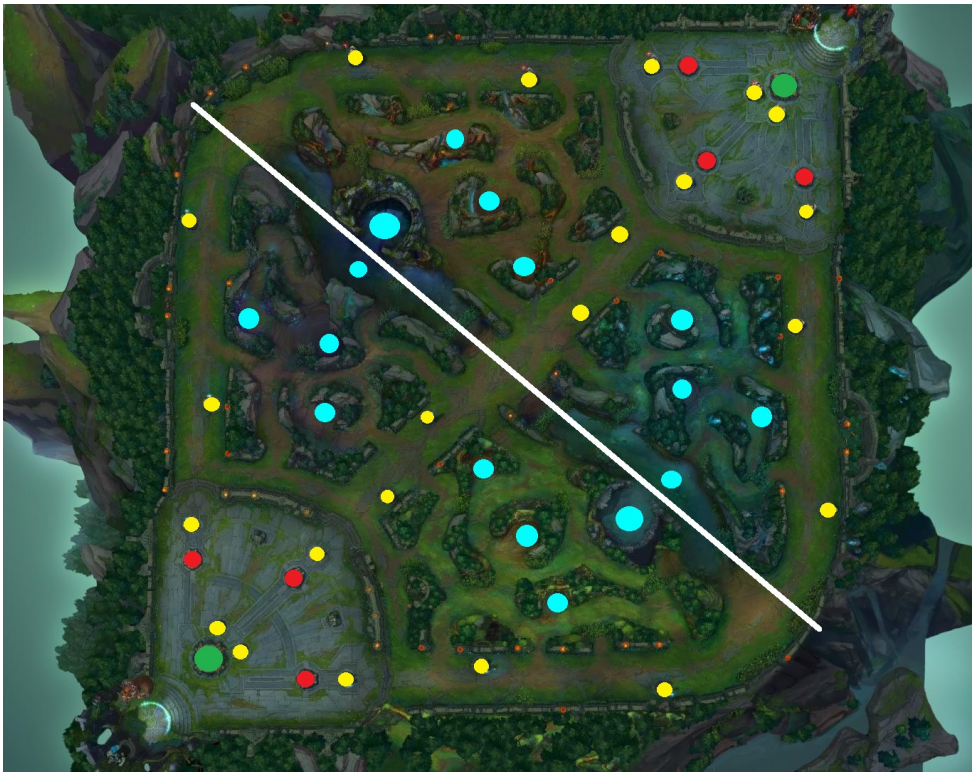
As defesas de cada rota são compostas por duas torres que atacam os inimigos quando próximos. Cada base possui entradas, ou saídas a depender da direção de movimentação do personagem, para cada rota e também possuem suas próprias defesas.

As entradas de cada base contam com uma torre de defesa, igual às torres presentes ao longo das rotas e em seguida um pequeno cristal de energia chamado *Inhibitor* (Inibidor). Para a peça principal do jogo, chamado de Nexus, temos duas torres de defesa. O Nexus é um grande cristal que cria tropas, chamada de *minions* para lutarem ao seu lado. São três ondas de tropas geradas, uma para cada rota. Por fim, ao lado do grande cristal, fica

a fonte por onde retorna ou renascem os campeões e uma loja para compra de itens que os potencializam, conforme abaixo se tratará. Na Figura 5, o significado para as simbologia são:

- Linha branca: divisão do mapa (lado inferior e superior);
- Círculos amarelos: torres de defesa;
- Círculos azuis: monstros da selva;
- Círculos vermelhos: Inibidores;
- Círculos verdes: Nexus;

Figura 5 – Mapa de Summoner's Rift

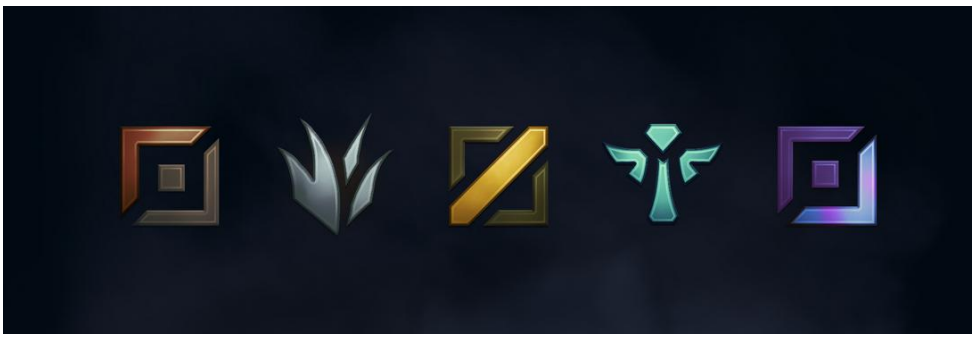


Os *laners* são os jogadores que atuam na rota que lhes é designada pela equipe com características específicas. Os *laners* recebem o mesmo nome, semelhante de sua *role* e sua simbologia está representado na Figura 6:

- *Toplaner* (Topo): jogador da rota do top, normalmente eles são a linha de frente, segurando todo o dano causado pela equipe inimiga e também os contra-atacando;
- *Jungler* (Caçador): jogador que atua na área da Selva e tem como missão abater os monstros presentes e também auxiliar os aliados na rota para remover a pressão causada pelo inimigo ou chegar para o abate;

- *Midlaner* (Meio): jogador da rota do meio, um dos carregadores do time, esta posição é geralmente composta por personagens magos ou assassinos;
- *Ad Carry* (Carregador): jogador da rota inferior é o principal causador de dano aos inimigos;
- *Support* (Suporte): acompanha o *Ad Carry* na rota inferior, e tem como função disponibilizar visão do mapa, atuando como sentinela, possibilitando ver a movimentação do adversário ao longo do mapa.

Figura 6 – Simbologia dos laners



Fonte: (RIOT GAMES, 2018e)

2.2.2 Modos de jogo

O jogo possui uma gama de opções para seus jogadores. Caso o modo que esteja sendo jogado pelo *player* não o esteja agrando, ele pode fazer outras escolhas. Os modos alternativos presentes no LOL são:

- Coop vs IA: jogadores contra *Robot* (BOT), único modo no LOL que não existem adversários reais. Este modo é uma partida onde seu time é formado por outros jogares que também estão conectados a este modo e os oponentes são máquinas. BOT é uma aplicação autônoma que segue padrões para simular uma atividade humana. O nível dos BOT's pode ser determinado pelos jogadores entre o básico e o intermediário;
- Normal: é um modo comum em LOL, porém existem duas formas de criação deste modo que se diferenciam pela maneira de selecionar o campeão. As formas normais são:
    - Escolha às cegas: não existem predefinições, cada jogador escolhe seu personagem e combina com seus colegas as *roles* que deseja jogar. As escolhas do time inimigo só podem ser vistas quando ambos os lados estejam selecionados;

  - Modo Alternado: existem as predefinições de *roles* que o jogador deseja jogar antes de entrar na seleção de campeões. O jogador pode escolher até duas rotas preferenciais. Ao entrar na seleção de campeões vai ser mostrada ao jogador a *role* qual das duas roles foi escolhida para ele, dando início à fase de banimentos, isto é, os personagens que não poderão ser escolhidos pelos dois times. A fase de banimentos podem ter até 5 personagens banidos por time. Depois, inicia-se o momento de escolhas de personagens que são feitas por turnos, intercalados entre os times.

- Ranqueado: os procedimentos são iguais ao modo Normal-Alternado, entretanto, o resultado das partidas jogadas neste modo, classificam o jogador no ranking do jogo; *ranking* no jogo;

- *Twisted Treeline*: um mapa menor, com apenas 2 rotas e uma selva, para partidas envolvendo três jogadores em cada time.

- *All Random All Mid* (ARAM): também possui um mapa reduzido, com apenas uma rota, onde todos os personagens são selecionados aleatoriamente. O objetivo permanece o mesmo;

- Modos Rotativos: são temporários, que variam bastante em seu modo de escolha, objetivo ou atributos dentro de jogo;

- Customizada: o jogador é dono de uma sala com nove vagas para outros jogadores ou BOT, cabendo também a ele decidir as predefinições da partida. O invocador é livre para iniciar a partida quando quiser mesmo que esteja sozinho na sala. Normalmente é usada para testes em benefício próprio do invocador ou para organização de campeonatos amadores.

2.2.3 Seleção de campeões

O LOL consiste em uma partida com 10 participantes, com dois times (times azul e vermelho), compostos de 5 jogadores cada. Para buscar uma partida, o jogador deve iniciar uma fila de busca de partidas que vai selecionar os outros 9 participantes de mesmo nível (mesma liga: ver abaixo) para todos estarem pareados em termos de desempenho. Ao encontrar uma partida, os invocadores entrarão na área da seleção de campeões onde todos os integrantes da rodada deverão fazer suas escolhas e respectivas definições:

- Campeão: o personagem do jogo;

- Runas e talentos: atributos especiais para o personagem selecionado;

- Feitiços de invocador: definir suas habilidades específicas do jogador;

- Rotas (apenas para os modos Coop vs IA/Escolha as Cegas/ *Twisted Treeline*): indicar a rota para seus colegas de equipe.

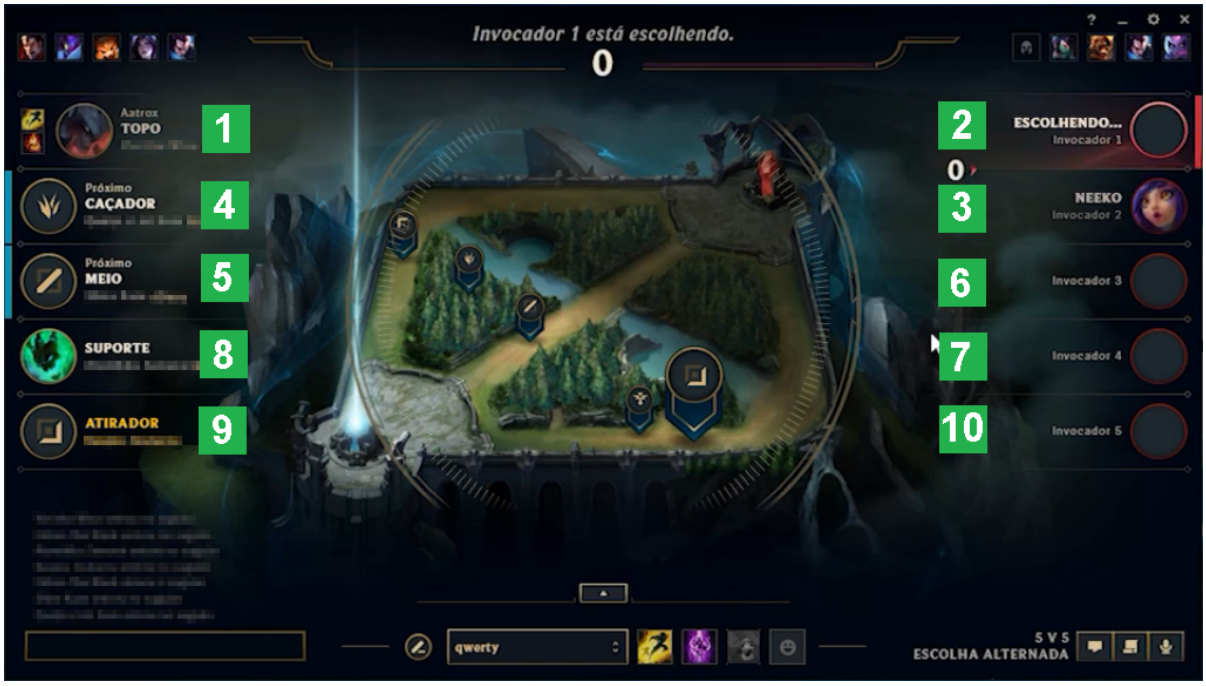
Na Figura 7 vemos o cliente do jogo, que é a interface da seleção de campeão. As numerações na imagem simbolizam:

1. Campeões banidos pelo seu time;
2. Campeões escolhidos pelo time aliado;
3. Personagens disponíveis para escolha;
4. Runas e talentos;
5. Feitiços de invocador;
6. Personagens banidos pelo time inimigo;
7. Campeões escolhidos pelo time inimigo;
8. Etapa da seleção de campeões, podendo ser fase de banimentos ou de escolhas;

Figura 7 – Tela de seleção de campeões

No modo de jogo ranqueado, a seleção de personagens é feito alternadamente entre os times, conforme já foi dito. Na figura a seguir, mostra a ordem das escolhas, após os times terem banidos os personagens.

Figura 8 – Ordem das escolhas na seleção de campeões



2.2.4 Ranking

O sistema de ranqueamento do LOL é chamado de Elo. Ao vencer uma partida ranqueada, o jogador ganhará uma quantidade de pontos com base na taxa de vitórias que o situam nas chamadas ligas. A ordem crescente das ligas é: Ferro, Bronze, Prata, Ouro, Platina, Diamante, Mestre, Grão-Mestre e Desafiante cujos ícones distintivos se encontram na Figura 9.

Figura 9 – Ligas de League of Legends

Fonte: (RIOT GAMES, 2018e)

Cada liga possui quatro níveis, por exemplo: Ouro IV, Ouro III, Ouro II, Ouro I. Tais níveis não existem nas três ligas mais altas da hierarquia nas quais a forma de promoção difere das inferiores.

Para obter ranqueamento e classificação em uma das ligas o jogador deverá jogar o que se denomina de “Melhor de 10 (MD10)”, ou seja, dez partidas ranqueadas. Embora o sistema se chame “Melhor de 10”, o ranqueamento do jogador não se dará a partir

do melhor desempenho obtido em uma das partidas mas do desempenho médio obtido nas dez partidas. Se um jogador se encontra fora do *ranking* e entra em uma partida com jogadores que são detentores de elos mais elevados e o seu desempenho se mostrar inferior ao esperado pelo *ranking* de sua equipe, ou dos adversários, ele será incluído em partidas que exigem menor desempenho até que, uma vez completadas as dez partidas, seja consolidado o *ranking* a que pertencerá, sendo então lhe atribuída um elo. A maior liga que pode ser conquistada por este método (MD10) é a Platina.

Para avançar de nível dentro de uma mesma liga, o jogador deve conquistar 100 pontos. Por exemplo, caso o invocador esteja no Prata III, ele deve conseguir 100 pontos neste nível e em seguida disputar uma série de promoção, que funciona no esquema Melhor de 3 (MD3) para subi para o nível II.

O avanço na escala das liga só ocorre para os níveis iniciais da liga seguinte. Novamente, o jogador deve conseguir os 100 pontos e ganhar a série de promoção. Só que para mudança de liga, em vez de partidas MD3, é exigido o esquema Melhor de 5 (MD5).

Caso o jogador seja derrotado diversas partidas, quando já estiver com zero pontos, existe o risco de rebaixamento da divisão ou elo. Outra forma de queda, pode decorrer de inatividade que se caracteriza quando o jogador passa muito tempo sem jogar filas ranqueadas.

Nas três últimas ligas não existem limites de pontos, ranqueando os jogos entre os melhores e piores colocados. Após passar um tempo determinado de horas, as três ligas são atualizadas. Os piores colocados são rebaixados para a liga anterior e os melhores colocados são elevados para a liga imediatamente superior. Quem se encontra na mais alta liga (desafiante), não tendo mais possibilidade de ascensão deve manter o desempenho em alto nível para evitar o rebaixamento.

2.2.5 Objetivo do jogo

Para conseguir a vitória, o objetivo principal é destruir o Nexus do time inimigo. Ainda assim, a caminhada até a base inimiga não é fácil. Todos os obstáculos descritos na Subseção 2.2.1 devem ser destruído até chegar ao grande cristal.

Para destruir as defesas é necessário que os campeões subam de níveis. Basta ficar próximo aos inimigos abatidos para ganhar experiência e subir de nível, a cada subida o personagem ganha um ponto de habilidade para ser utilizada. Todo início de partida, o personagem começa do nível 1 até chegar a 18, o máximo.

Ao abater tropas, o campeão ganha ouro que pode ser gasto na loja para compra de itens que deixará mais forte. Outras formas de conseguir ouro é derrotando os campeões inimigos, destruindo fortificações ou abatendo monstros da selva.

Quando o *inhibitor* inimigo é demolido, as tropas aliadas ficam mais fortes e ajudam bastante na conquista do objetivo principal, contudo, o *inhibitor* renasce após alguns

minutos.

2.2.6 Atualizações no jogo

A cada duas semanas ocorre uma atualização no *patch* do jogo, modificando diversos atributos como, por exemplo, *status* de personagens e itens, tempo dos objetivos, custo dos itens e habilidades, entre outros. Alguns *patches* são lançados sem intervalo determinado: geralmente são *hotfixes*, para correção de falhas do jogo.

O meta é um estilo de jogo analisada de forma estatística a cada atualização, buscando jogar aproveitando o máximo dos recursos gerando uma maior probabilidade de vitória. Esses *patches* são responsáveis pela formação do meta, que vai definir a estratégia a ser utilizada pelas equipes, com base na viabilidade e percentual da taxa de vitória que o personagem está atualmente no jogo. Os metas podem ser alterados ou não, conforme o impacto que as atualizações causam no LOL.

2.2.7 Competições regionais e internacionais

O League of Legends possui 11 ligas profissionais espalhadas pelo mundo (Tabela 2). As competições regionais são realizadas duas vezes (*Split Summer* e *Split Spring*) por ano, com premiação em dinheiro e conquista de vaga para a disputa internacional.

Tabela 2 – Ligas profissionais de League of Legends

Sigla	Nome da Liga	Região
LCS	League of Legends Championship Series	América do Norte
LEC	League of Legends European Championship	Europa
LCK	League of Legends Champions Korea	Coreia do Sul
LPL	Tencent League of Legends Pro League	China
LMS	League of Legends Master Series	Sudeste Asiático
CBLOL	Campeonato Brasileiro de League of Legends	Brasil
LLA	Liga Latinoamérica	América do Sul e Central
LCL	League of Legends Continental League	Rússia e região
LJL	League of Legends Japan League	Japão
OPL	Oceanic Pro League	Oceania
TCL	Turkish Champions League	Turquia

(RIOT GAMES, 2018d)

No primeiro split (*Summer*), os campeões da primeira etapa são classificados para o *Mid-Season Invitation* (MSI), que é uma modalidade competitiva internacional e que garante, ao vencedor, vaga direta para a etapa mundial.

Na segunda etapa (*Spring*), os times vencedores de suas ligas e do MSI, irão disputar o *League of Legends World Championship* que entrega o título de campeão mundial do

ano e a premiação de milhões de dólares. Além disso, os jogadores campeões mundiais ganham *skins* (visuais para os personagens) para os seus escolhidos.

Algumas regiões são consideradas mais fortes que outras, devido ao seu desempenho em competições internacionais anteriores, assim podem conseguir até duas vagas e classificação direta no campeonato mundial. As regiões consideradas fracas ficam com apenas uma vaga e delas se exige que disputem jogos de classificatórias para a competição internacional, disputando contra outras equipes de mesma categorização.

Ações beneficentes são realizadas durante os eventos mundiais. No *All-Stars* 2018 (RIOT GAMES, 2018a), os jogadores profissionais e *streamers* (pessoas que realizam transmissões ao vivo) da mesma região foram organizados em duplas para disputarem um torneio 2 contra 2 entre regiões, e uma parte do valor arrecadado durante o evento seria destinado a uma instituição de caridade que a dupla vencedora estava representando.

2.2.8 Times competitivos

A sistematização de times competitivos são gerenciadas pela comissões técnicas, que estabelecem para os jogadores uma quantidade de treinos computacionais visando exercitar reflexos, realizar atividades físicas para prevenir problemas relacionados ao sedentarismo e tensões musculares, além de sessões psicológicas para auxiliar na tomada de decisões, trabalho em equipe e lidar com pressões dentro e fora de jogo.

Por conta do jogo estar em mudança constante, as equipes técnicas devem estar sempre atualizadas quanto às novidades dos *patchs*; analisar estratégias de outras equipes nacionais ou internacionais, repassar todas as informações para os atletas que devem treinar as novas habilidades visando os confrontos oficiais.

Em dias de competição, as comissões técnicas devem escalar os atletas e determinar a estratégia a serem seguidas pelos jogadores, com base nos treinos. É útil a criação de uma composição da equipe que melhor se encaixe para a partida em específico.

é o desempenho preditivo de modelos e o segundo é automatizar o processo de modelagem das bases de dados observados ou aprendizado com os dados observados.

2.3 Machine Learning

Machine Learning (Aprendizagem de Máquina) é um ramo da Inteligência Artificial com o objetivo de desenvolver algoritmos para as tomadas de decisões de forma otimizada. Segundo VASCONCELOS (VASCONCELOS, 2017), o aprendizado de máquina possui dois objetivos, o primeiro é o desempenho de modelos preditivos e o segundo é a automatização dos processos de modelagem ou aprendizado de dados. A máquina pode ser treinada para coletar e processar um conjunto de dados sugerindo o que o usuário deseja, ou seja, utilizam-se algoritmos de recomendação.

A Aprendizagem de Máquina pode ser dividida em duas tarefas:

- Aprendizagem Supervisionada: possui dados de entrada e saída, onde a máquina entende como tratar aquelas informações e gerar tanto entradas quanto saídas;
- Aprendizagem Não Supervisionada: possui apenas dados de entrada; então é o próprio algoritmo que define os agrupamentos e os categorias de saída a serem devolvidos.

A utilização de algoritmos de *machine learning* em jogos eletrônicos vem sendo, com frequência, objeto de pesquisas científicas como é o caso de LOPES e BRAGA (LOPES; BRAGA, 2017) que utilizaram o algoritmo de aprendizado *Q-learning* com redes neurais, que faz uso de regras para tomar uma determinada ação ideal para otimização do processo. O objetivo do trabalho deles era desenvolver um programa autônomo para obter a maior pontuação possível jogando Galaga, do console Playstation 2, e outros jogos da plataforma *Atari*.

2.4 Python e Google Colab

A linguagem de programação Python, lançada por Guido van Rossum em 1991, foi idealizada visando facilitar à produtividade e legibilidade do código. Na contemporaneidade, o modelo de desenvolvimento Python integra-se ao *open source*, com as atualizações sendo gerenciadas pela Python Software Foundation, sem fins lucrativos.

O Python aproxima-se do R em relação a sua sintaxe, por causa da concisão de sua escrita e integração com múltiplas bibliotecas e módulos desenvolvidos pela própria comunidade. O código conquistou popularidade entre os grupos científicos podendo ser utilizado em aplicações criadas para o processamento de dados.

Google Colab é uma plataforma disponível em nuvem que executa código Python nas versões 2.7 e 3.6. O desenvolvimento do projeto dentro da plataforma pode ser feito de forma colaborativa, além de possuir uma *Graphics Processing Unit* (GPU) para otimização do código. Por ser uma ferramenta da Google, ela possui integração com outras aplicações da própria empresa.

A aplicação disponibiliza um Jupyter Notebook, exclusivo para análise de dados, cedendo 12 GB RAM, 48 GB de armazenamento em disco e 10 GPU para criação e compartilhamento de *workspaces* comportando códigos de estudos científicos nos processamentos das informações. O serviço Colab é gratuito, evitando que os apreciadores em ciência de dados tenham gastos na compra de um potente computador pessoal.

2.4.1 Keras API (TensorFlow)

Keras API é uma rede neural de alto nível para montagem e treinamento de modelos para aprendizagem profunda de máquina. A biblioteca trabalha em cima de TensorFlow, CNTK ou Theano. Possuindo uma sintaxe simples para implementação e otimização na execução das linhas de comando.

Uma das vantagens de uso do Keras é sua execução aperfeiçoada utilizando CPU e GPU. O Jupyter Notebook é perfeito para trabalhar juntamente com a API. Foi desenvolvida em Python, porque possui compatibilidade, facilidade na depuração e extensão da ferramenta.

2.5 Multi Layer Perceptron (MLP)

Publicada em 1986, o *Multi Layer Perceptron*, traduzindo Perceptron de Múltiplas Camadas, é uma rede neural de aprendizado para solução de problemas de forma linear. Ele surgiu pensado no algoritmo Perceptron. O processo de treinamento, normalmente, é realizado através do algoritmo de aprendizado supervisionado *backpropagation*.

O Perceptron é uma Rede Neural Artificial (RNA), desenvolvida em 1957 por Frank Rosenblatt no Cornell Aeronautical Laboratory. Tal algoritmo de treinamento do perceptron foi o primeiro modelo supervisionado, embora alguns perceptrons fossem auto-organizados, consiste em uma única camada de neurônios com pesos sinápticos e bias ajustáveis (CASTRO; ZUBEN, 2001).

O MLP surgiu com a proposta da elaboração de uma rede perceptron com múltiplas camadas na tentativa de superar os problemas encontrados no aprendizado do Perceptron simples. A necessidade de desenvolver treinamento sofisticado, automatizando as definições para os pesos de rede de neurônios. A diferença entre o MLP e o Perceptron simples é a quantidade de camadas de neurônios.

Backpropagation ou retropropagação do erro, é um algoritmo de treinamento de redes neurais artificiais. Sua forma de inicialização, consiste em um método comum utilizado para o aprendizado das redes, inserindo de forma distribuída os pesos com valores aleatórios e no final do processo gerando uma saída desejada. Durante o treinamento com o algoritmo, a rede realiza um processamento dividido de dois passos (MIGEZ; MACULAN FILHO; XAVIER, 2012):

1. Processamento direto: é apresentado um modelo à camada de entrada da rede. O sinal resultante flui pela rede, entre as camadas intermediárias, até que a resposta será gerada pela camada de saída.

2. Processamento reverso: a saída gerada no primeiro passo é comparada com a desejada para o modelo. Se esta saída não estiver correta, o erro é calculado. Este

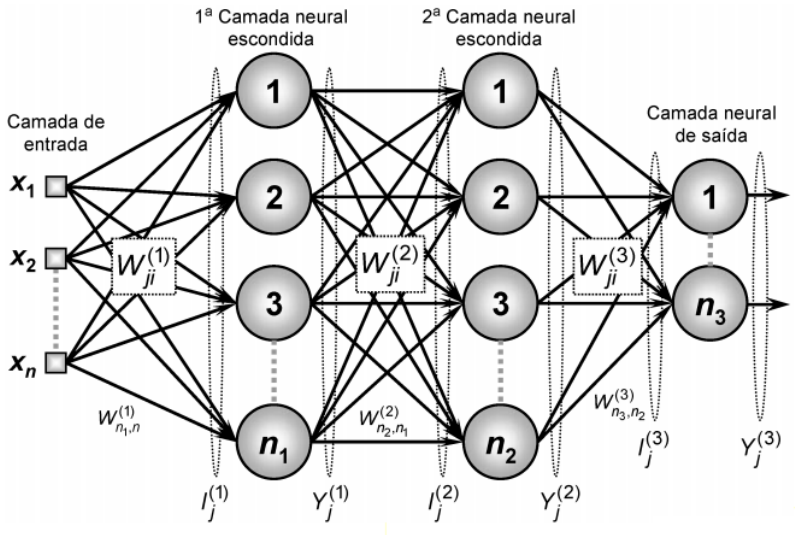
erro é propagado percorrendo o processo reverso da rede, da camada de saída até a de entrada, e os pesos das conexões das unidades das camadas internas vão sendo modificados à medida que a incorreção é retropropagada.

Faz uso de combinações lineares para cálculo do aprendizado. Realiza-se a soma da multiplicação dos sinais de entrada multiplicados aos seus receptivos pesos ($x_1 * w_1 + x_2 * w_2 + ... + x_n * w_n$) e do limiar de ativação (-θ), resultando no potencial de ativação (u), como pode-se ver na fórmula abaixo.

$$u = \sum_{i=1}^{N} Wi * Xi - \theta$$

Segundo os pesquisadores RIGO JUNIOR et al. (RIGO JUNIOR et al., 2017), o MLP é uma rede neural contendo camada de entrada, uma camada de saída e uma ou mais camadas escondidas (podendo serem chamadas também de camadas intermediarias), onde os neurônios de uma camada estão conectados com todos os neurônios das camadas ao lado. Na Figura 10, vemos a estrutura do MLP, sendo que entre as conexões é realizado as combinações lineares para resultar no potencial de ativação e repassados para as camadas adjacentes.

Figura 10 – Estrutura de aprendizado do MLP



(RNA..., 2001)

O *Multi Layer Perceptron* já foi utilizado em diversos trabalhos com objetivo de presumir um resultado. No projeto desenvolvido pelos RIGO JUNIOR et al. (RIGO JUNIOR et al., 2017), a rede MLP é aplicada para estimar as taxas de emissão dos gases: Óxidos de Nitrogênio (NO_X), Amoníaco (NH_3) e Óxido Nitroso (N_2O) em veículos movidos a Diesel pensado no percentual de projeção destes poluentes na atmosfera. Os pesquisadores utilizaram 8–10 camadas de entrada, 4–6 camadas intermediarias e 1 camada de saída

na função *sigmoid* para o aprendizado da rede. Segundos os pesquisadores deste projeto, eles obtiveram sucesso utilizando este método eficiente para prever as taxas de emissão de poluentes.

2.6 Base de Dados (Kaggle)

O Kaggle é uma plataforma, adquirida pela Google em 2017, de *Data Science* com hospedagem para competições públicas, privadas e acadêmicas entre os Cientistas de Dados. Além disso, a plataforma também armazena e disponibiliza *datasets* de diversos conteúdos.

Dentre os datasets presentes no Kaggle, foram encontradas duas bases de dados que continham informações sobre partidas de League of Legends organizadas por região. As informações de ambas as tabelas foram coletas de uma *Application Programming Interface* (API) desenvolvida pela Riot Games, que fornece tais dados armazenados em seus servidores.

Mesmo sendo coletas de um mesmo *framework*, as tabelas possuem uma diferença importante. Uma reúne dados de partidas oficiais competitivas, ou seja, campeonatos da Riot Games, a outra traz dados de partidas ranqueadas de jogadores de LOL.

2.7 Riot Games API

API é um conjunto de rotinas e padrões de programação para acesso a um aplicativo de software ou plataforma baseada na Web. A intenção de desenvolver uma API, é incentivar outros desenvolvedores a implementar produtos associados a seu serviço.

Como citado anteriormente, as tabelas da base de dados coletaram as informações através da Riot Games API. Os dados são transmitidos entre as tecnologias utilizando *JavaScript Object Notation* (JSON), aonde irá ser mostrado toda forma de informação existente naquela partida.

A intenção da Riot Games na liberação da API para a comunidade era criar integrações de projetos que otimizem a experiência do jogador. Diversas aplicações que utilizam a ferramenta já estão disponíveis na Web para uso, como, por exemplo: OP.GG, Blitz APP, Champions.gg, entre outros.

A empresa determina políticas do uso da ferramenta para que não haja utilização ilícita dos dados de seus jogadores ou informações internas do jogo para geração de monetização, alterações no código-fonte ou qualquer outra forma de benefício pessoal. No entanto, a organização disponibiliza materiais para desenvolvedores de ferramentas de apoio aos jogadores que, por sua vez, ao desenvolverem maiores habilidades, alimentam o jogo e possibilitam à Riot Games criar novas possibilidades no âmbito do jogo.

2.8 Linguagem R e RStudio

A linguagem de programação R é responsável pela manipulação e processamento de dados coletados por várias aplicações. Oferece uma gama de opções para estudos estatísticos e geração grafos para uma melhor compreensão dos dados. Criada inicialmente por Ross Ihaka e por Robert Gentleman no departamento de Estatística da Universidade de Auckland, na Nova Zelândia, a linguagem é *open source*, ou seja, a evolução da linguagem é feita com base nas colaborações de várias pessoas.

O RStudio é uma *Integrated Development Environment* (IDE), uma ferramenta visual para a utilização do R, que integra vários mecanismos para o desenvolvimento de um processo de forma eficaz. A IDE é considerada a melhor ferramenta para desenvolvimento de códigos em R, permitindo separar os projetos em diretórios. Além de possuir diversas bibliotecas disponíveis para download diretamente da IDE, permitindo a extensão da linguagem

3 Desenvolvimento

Os tópicos deste capítulo tratam da maneira com que foi desenvolvido o algoritmo; do reconhecimento e manipulação das tabelas do Kaggle até a montagem do código.

3.1 Análise das tabelas e validação dos dados pela Riot Games API

Antes de dar início ao manuseio das tabelas, foi exigido um estudo de todos os elementos presentes nas mesmas, porque não se deve utilizar um dado sem saber o seu significado. Esta etapa do processo foi importante para saber com as colunas que seriam relevantes para o projeto.

O projeto para este trabalho, escolhido no Kaggle, foi criado por CAMPANELLI (CAMPANELLI, 2017), utilizando a base dos dados que continha mais de 180 mil rodadas de League of Legends ranqueadas, coletadas a partir de 2014. Por ter uma amostragem maior, possibilita obter maior precisão na predição dos personagens. A partir das sete tabelas elaboradas por Campanelli, foi procurado identificar o que cada uma delas reunia e quais tipos de dados carregava consigo.

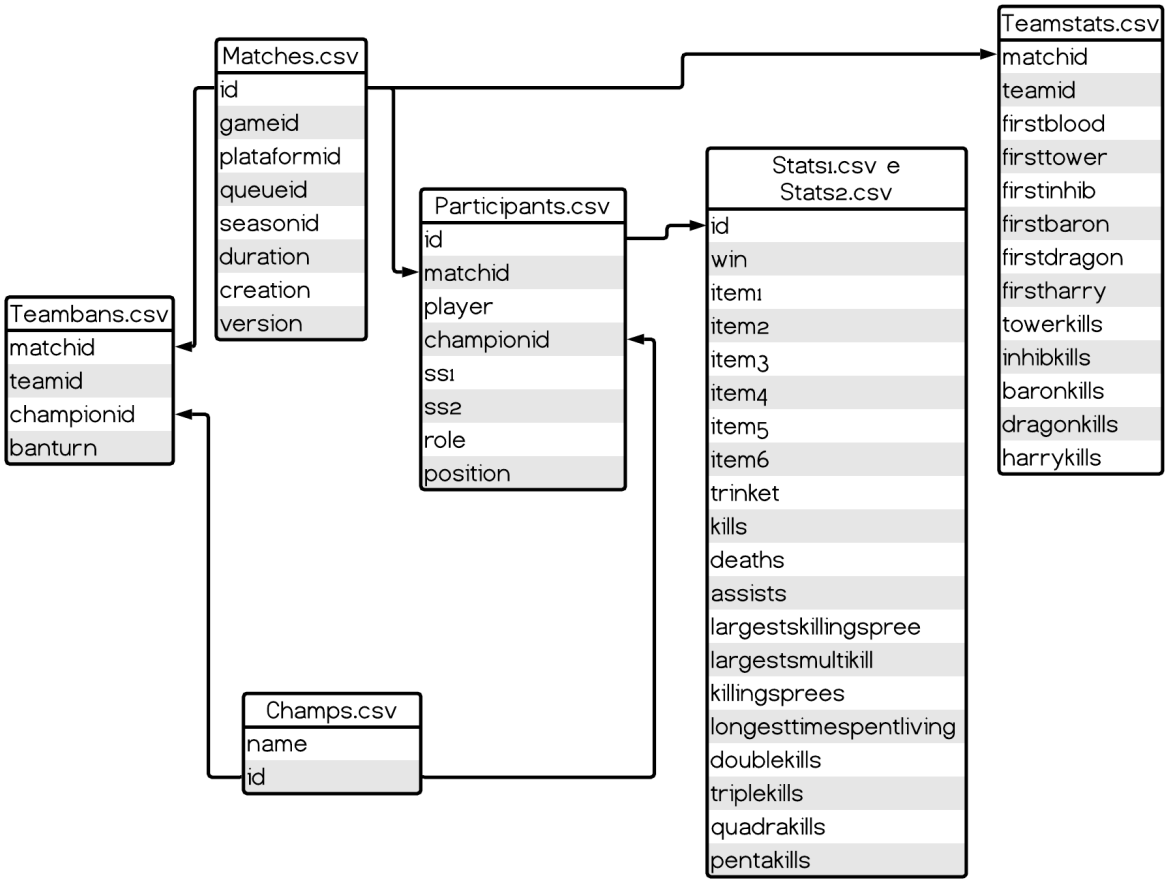
Mesmo que os dados estivessem espalhados entre várias tabelas, existia cruzamento de informações entre as bases através de uma coluna que tinha função de conter um valor identificador para referência. As colunas que estivessem com a abreviatura 'id' sozinha ou entre suas palavras, significava que está mencionando algo ou servia para ser mencionada em outra base. As tabelas foram reconhecidas da seguinte forma:

- Champs.csv: tabela de identificação dos personagens. Campos: *name e id*;
- Matches.csv: tabela de identificação da partida. Campos: *id, gameid, plataformid, queueid, seasonid, duration, creation, version*;
- Participants.csv: tabela de identificação dos jogadores da partida, campeões selecionados e *role position*. Campos: *id, matchid, player, championid, ss1, ss2, role, position*;
- Stats1 e Stats2: uma continuação de linhas da outra e mesmas colunas. Tabela de detalhamento da partida para cada jogador, onde informa o time vencedor, itens comprados, abates, mortes, assistências e entre outros. Campos: *id, win, item1, item2, item3, item4, item5, item6, trinket, kills, deaths, assists, largestskillingspree, largestsmultikill, killingsprees, longesttimespentliving, doublekills, triplekills, quadrakills, pentakills*;

- Teambans.csv: tabela de personagens banidos na partida. Campos: *matchid, teamid, championid e banturn*;
- Teamstats.csv: tabela de objetivos conquistados pelo time na partida. Campos: *matchid, teamid, firstblood, firsttower, firstinhib, firstbaron, firstdragon, firstharry, towerkills, inhibkills, baronkills, dragonkills e harrykills.*

Para um melhor entendimento das tabelas e suas interações, foi montado um diagrama das bases de dados Kaggle e as relações entre seus campos ilustrados na Figura 11.

Figura 11 – Diagrama das bases de dados do Kaggle.



Pensado na integridade da base, outra etapa teve importância no seguimento do estudo, que foi a validação das informações, isto é, se as partidas registradas eram reais. A Riot Games API é a ferramenta essencial para execução desta atividade, porque é a portadora dos históricos de partidas de todos os seus jogadores.

3.2 Manipulação das tabelas no RStudio

As tabelas escolhidas contêm dados do jogo, como, por exemplo: abates, personagens, mortes, assistências, identificador de partida e entre outros. Dispõe uma gama de opções para fazer diversas categorias de processamento de dados.

Alguns elementos não foram utilizados porque não iriam influenciar no resultado que se deseja para este trabalho. Portanto, como queremos realizar predição de melhores escolhas de personagem para a partida, apenas as colunas que contém relação de campeão e rodada foram manipuladas.

As informações estão espalhadas entre múltiplas tabelas, tem casos de que uma das bases é a continuação de outra tabela. Para solução, foi necessário gerar uma nova tabela contendo os dados que estão espalhados. Como vemos no trecho abaixo, utilizou-se uma biblioteca, chamada *sqldf*, do RStudio que utiliza *scripts* SQL para realização da tarefa e no final gerando uma tabela "geral" (Figura 12).

Figura 12 – Reagrupamento de todas as partidas para a tabela geral

```
1 partidas = sqldf::sqldf("SELECT matchid, championid from participants")
2 stats = rbind(stats1, stats2)
3 resultados = sqldf::sqldf("SELECT win from stats")
4 geral = cbind(partidas, resultados)
```

Foi criada uma base de dados com três colunas e mais de 1.800.000 linhas. A coluna *matchid* contém o número identificador da partida, sendo que todo jogo é composto por 10 jogadores. A coluna *championid* traz o número identificador do personagem baseado na Riot Games API. A coluna *win* é referente ao resultado da partida, onde 0 significa derrota para o jogador da mesma linha e 1 representa vitória.

Além disso, mais de 1 milhão de linhas constituem demasiados elementos para serem incluídos apenas uma tabela, o que ficaria custoso no momento de análise dos dados. Para otimização do processo, foram separadas variadas bases de dados para uma mesma partida, ou seja, foi montada uma tabela para cada jogo. No final desta etapa, era esperado uma quantidade acima de 180 mil tabelas.

O processo para a separação de tabelas por partida, foi montado em cima de 3 diferentes *functions* para execução das atividades (Figura 13). O papel de cada *function* será:

1. Separar os jogos pelo *matchid* da partida;
2. Verificar o resultado da partida, onde 0 é para o lado azul e 1 para o vermelho;
3. Salvar as tabelas como nomeação matchid_ladovencedor, por exemplo: 13_0.

Figura 13 – Separação das partidas por tabela

```
coletarDados = function (data, matchid){
  coletar = sqldf::sqldf(paste("SELECT matchid, championid, win from geral
    where matchid = ", matchid))
  time_azul <- subset(coletar, as.numeric(rownames(coletar))<=(length(
    coletar$championid)/2))
  time_verm <- subset(coletar, as.numeric(rownames(coletar))>(length(
    coletar$championid)/2))
  final <- data.frame(cbind(bChampionid=time_azul[,2], rChampionid=
    time_verm[,2],win=time_azul[,3]))
  return(final)
}

verificarVencedor = function(data){
  if(data[1,3]==1){
    valor = 0
    return(valor)
  }else{
    valor = 1
    return(valor)
  }
}

porPartida = function(data){
  for(i in 10:187588){
    coletar = coletarDados(data, i)
    if(is.na(coletar[1,1])){
      print("Sem MatchID")
    }else{
      vencedor = verificarVencedor(coletar)
      coletar = coletar[,-3]
      texto = c(i,"_",vencedor,".csv")
      write.csv(coletar, paste(texto, collapse = ''), row.names = FALSE)
    }
  }
}

porPartida(geral)
```

A máquina levou alguns dias para terminar de separar todos os dados, o processo de separação foi encerrado e como dito anteriormente, era esperado um valor acima de 180 mil arquivos de tabelas. Os resultados da execução destas funções podem ser encontrados no Apêndice A, ilustrados nas Figuras 32 e 36.

3.3 Avaliação da rede MLP

Através dos conjuntos de bibliotecas Keras API, pode-se construir um conjunto de redes neurais artificiais capazes de receber estímulos (valores aleatórios) para que possam processar as informações, repassar o aprendizado entre os neurônios e resultar numa resposta esperada para o usuário.

Inicialmente foi separada uma amostra para uma triagem de validação. Montou-se duas tabelas para comparação dos valores, uma matriz com 10 colunas compostas pelos personagens e outra matriz para resultados, formada de apenas por vetores $[1, 0]$ (significando vitória do time azul) ou $[0, 1]$ (representando vitória do time vermelho). Essas tabelas precisariam ser tratadas antes de entrarem na rede neural.

Para o tratamento, foi utilizado um método chamado *holdout*, que consiste na separação das tabelas em: 70% do conteúdo servindo para treinamento da rede e os outros 30% para uso de testes, como mostrado na Figura 14.

Figura 14 – Preparação do *holdout*

```
x = pd.read_csv("partidas_1000.csv").values
y_train = pd.read_csv("resultados.csv").values

y = np.zeros([1000,2],dtype=np.uint8)

for i in range(0, len(y_train)):
  if(y_train[i] == 0):
    y[i] = [0,1]
  else:
    y[i] = [1,0]

y = y.reshape([1000,2])
x = x.reshape([1000,10])

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.33,
    random_state=42)
```

A validação da rede MLP era necessário para medir a acurácia das redes neurais. Para este projeto, foi estipulado 5 camadas de neurônios para o tratamento dos dados passando dez valores de dimensão. O trecho de código abaixo (Figura 15) mostra como foi gerado o grupo de neuros:

Figura 15 – Elaboração das camadas de Perceptrons da rede neural

```
model = Sequential()
model.add(Dense(256, input_dim=10 , activation='relu'))
model.add(Dense(128, activation='relu'))
model.add(Dense(64, activation='relu'))
model.add(Dense(32, activation='relu'))
model.add(Dense(2, activation='sigmoid'))
```

Com a RNA montadas e amostras separadas, o próximo passo seria injetar os modelos de dados nas redes. O modelo criado como métrica a acurácia. Logo após, criar um *fit* passando as tabelas de amostra, determinando uma quantidade de *epochs* necessárias para o aprendizado da máquina (Figura 16).

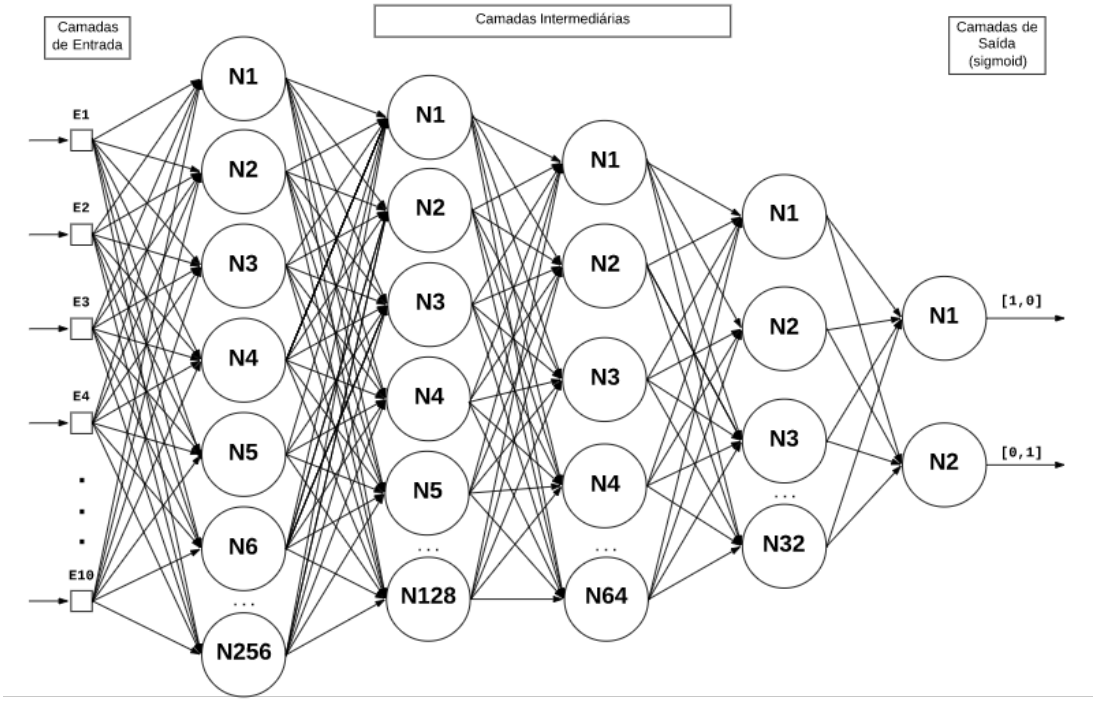
Figura 16 – Treinamento da rede MLP

```
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['
    accuracy'])

model.fit(x_train, y_train, epochs=60, batch_size=10)
```

Resumindo, foi separada dez camadas entradas, para serem injetadas nos neurônios (intermediárias) e gerar um resultado nas saídas em função *sigmoid*. De forma gráfica está mostrada na figura a seguir.

Figura 17 – Modelo Multi Layer Perceptron do projeto.



O modelo foi validado com cinco execuções do *holdout*, alcançando uma precisão média de 99,27% (Tabela 3) de acurácia no treinamento do algoritmo. Quanto maior a quantidade de vezes que o algoritmo é executado, ele aprende um padrão e melhora sua distinção.

Tabela 3 – Resultados de precisão para validação do MLP

Teste 1	Teste 2	Teste 3	Teste 4	Teste 5	Média
96.34%	100%	100%	100%	100%	99.24%

3.4 Implementação do algoritmo no Google Colab

Como dito anteriormente, o Google Colab compila linhas de comando em Python e para a execução desse projeto utilizaram-se as importações de bibliotecas padrões do Python e o uso da Keras API.

Durante a construção do algoritmo, foram separados alguns problemas de implementação, que deveriam ser solucionados:

1. Transformar os campeões em números para processamento otimizado dos modelos Keras API;
2. Gerar vetores aleatórios para cada entrada de dados;
3. Simular o modo alternado de escolhas;
4. Mostrar recomendação para os jogadores.

O primeiro problema já estava solucionado desde a seleção das tabelas do Kaggle. A base de dados Champs.csv possui uma coluna de *id* que contém apenas valores inteiros. Estas categorias de dados, é referente a Riot Games API, onde os seus campeões possuem um identificador em todas as suas aplicações.

No entanto, seria necessário memorizar todos os identificadores passando o valor inteiro para os vetores que iriam ser gerados. Então, criaram-se funções que faziam a transformação do *name* do personagem pela sua *id* (Figura 18).

Figura 18 – Método de transformação dos nomes de campeões em *id*

```
def idChamp(champs, champ):
  cont = 0
  for v in [x[0] for x in champs]:
    if v.lower() == champ.lower():
      return champs[cont][1]
    else:
      cont = cont + 1
```

Na solução do segundo problema assentavam os pilares para o funcionamento do e-Coach. Novamente, a implementação de funções foram montadas para solução da problemática.

Dentre as bibliotecas presentes no Python, existe uma que gera uma faixa de valores aleatorizados dentro de uma parametrização determinada pelo desenvolvedor, chamada de *random*.

A cada entrada de dado feita pelo problema anterior, passaria como parâmetro de seleção aleatória o *dataset* dos Champs.csv, logo o *random* selecionaria os valores naquela faixa de opções parametrizadas, conforme mostrado no trecho de código da Figura 19.

Figura 19 – Método de preenchimento dos vetores com números aleatórios

```
import random as rd

def montarTime(champs, characters, time, pos, pers):
  time.insert(pos, pers)
  pos = pos + 1
  while len(time) != 5:
    valor = rd.choices(characters)
    valor = sum(valor)
    if valor not in time:
      time.insert(pos,valor)
      pos = pos + 1

  return time

def alteracaoTime(champs, characters, time, pos):
  pos = pos + 1
  while pos != 5:
    valor = rd.choices(characters)
    valor = sum(valor)
    if valor not in time:
      time[pos] = valor
      pos = pos + 1

  return time
```

Foi elaborado o método de construção dos vetores, tendo sido os nomes dos personagens transformados em identificadores. O próximo passo foi simular o modo alternado das escolhas, como mostrado na Figura 8. A resposta do problema, está no estabelecimento da sequência adequada de como são chamados os métodos.

A sequência foi a mesma, não importando em qual time os jogadores estivessem jogando. Ao final da rodada de seleção dos personagens, o MLP iria entrar em execução e a lista de vetores passaria pelo processo de predição, comparando-se os *arrays* criados com os resultados, e exibindo as 10 primeiras decisões para o usuário.

4 Experimentos

O objetivo do experimento foi testar o algoritmo em situação real, em partidas com os jogadores, analisando sua eficiência na sugestão de personagens. Para a realização do experimento foi necessário formar um grupo de seis pessoas, das quais cinco participariam jogando no modo Normal-Alternado e uma pessoa faria o monitoramento.

Ainda assim, um problema deveria ser solucionado antes de começar o experimento. A complicação ocorria na tela de seleção de campeões. Quando chega a vez de cada participante escolher o campeão, ele tem apenas 20 segundos para o fazer, do contrário a plataforma do LOL desconecta todos os invocadores e a partida é cancelada, além de penalizar o *player*, que não escolher o personagem a tempo. Como o processamento do algoritmo consumia alguns desses segundos que poderia inviabilizar o experimento, foi separada uma amostra menor para continuação e execução de um aprendizado mais rápido. Além disso, a quantidade de *epochs* foi diminuída para uma execução mais veloz do código.

4.1 Planejamento de experimentação entre o e-Coach e jogadores

Os candidatos foram submetidos à realização um conjunto de 10 partidas, como o MD10 acima descrito, separadas em duas etapas de cinco rodadas cada. A primeira etapa foi destinada à utilização do e-Coach, na qual o participante teria que selecionar apenas dentre personagens sugeridos pelo algoritmo, porém, como o algoritmo precisa de dados de entrada para iniciar o aprendizado, as primeiras seleções eram livres e as possibilidades seguintes seriam estabelecidas pelo algoritmo. Já na segunda etapa todas as escolhas seriam de livre seleção dos integrantes.

Para acompanhar a partida, o monitor utilizou uma função que o LOL disponibiliza para seus jogadores, que é chamado de Modo Espectador no qual qualquer jogador pode assistir a partida de outro *player*, sendo que o invocador deve estar na lista de amizades. O modo espectador tem um atraso de 3 minutos para quem deseja assistir, assim evitando que os invocadores que estão assistindo a partida passem informações em tempo real para quem está jogando a rodada.

4.1.1 Fase de preparação do experimento

Foram selecionados cinco candidatos com diferentes níveis de habilidades em League of Legends. A comunicação no experimento foi feita de forma online, via Discord (software de

comunicação), e requisitado que os *players* iniciassem uma partida no jogo, diretamente de sua própria máquina. Foi solicitado aos candidatos que atendessem a regras e requisitos de execução especificas para o experimento que, entretanto, não alteram as regras dos jogos normais. Caso ocorressem imprevistos externos, como problemas de conexão, quedas de energia ou qualquer outro fator que afetasse os candidatos ou a partida, o experimento seria invalidado para análise e seria executado novamente.

Antes de iniciar a experimentação, os candidatos preencheram o formulário de participação do experimento, sendo informados que nenhum de seus dados pessoais seriam expostos, e a concordância com um termo isentando o organizador da experimentação dos prejuízos que pudessem ser causados a conta do usuário junto ao LOL. Na Subseção 4.1.2 abaixo, está descrito mais detalhes sobre os termos.

Após o fim do experimento, foi colhido o *feedback* dos candidatos sobre o experimento e sugestões que pudessem ser implementadas para aprimorar o código.

4.1.2 Formulário de participação e termo de responsabilidade

Os voluntários assinaram um formulário de participação no experimento, pelo Google Formulários no qual preencheram seus dados pessoais e concordância as propostas do experimento e aprovação dos termos de responsabilidade que isentava o organizador do experimento por prejuízos causados a conta do candidato durante a realização do experimento, como, por exemplo: suspensão permanente de conta, mau comportamento dentro de jogo do participante e entre outros, pois, conforme a Riot Games, tais punições não visam a boa conduta dos jogadores.

Aqueles que prejudicam, ofendem ou praticam alguma atitude negativa no jogo são levados ao Tribunal (uma central para atender os relatórios de comportamento inapropriado dos jogadores enviado pelos membros da sua equipe) e determinado uma punição consoante ao grau de relevância do ato praticado.

4.2 Execução do experimento entre o e-Coach e jogadores

Com os jogadores reunidos em sala Discord, foram informados as regras e os requisitos de execução do experimento:

1. Banimento de campeões novos, porque a base de dados utilizada está com alguns dados atrasados referentes a personagens novos lançados após a última atualização feita no Kaggle;

2. Evitar repetir o mesmo campeão da partida anterior;

3. Para a primeira etapa estariam sujeitos as escolhas do algoritmo;

4. Escolherem entre si qual participante seria o representante que iria comunicar-se com o monitor;

5. Executar o experimento com seriedade, pois, como se trata de um jogo, algumas pessoas possuem comportamento impróprio, porém, ainda seria permitido a sua diversão.

No início da seleção de personagens, a primeira informação a ser trocada entre monitor e jogador era sobre qual time estavam jogando. A escolha do lado, entre azul e vermelho, é feita de forma aleatória pela plataforma do LOL. Em seguida, conforme os banimentos e as escolhas (de forma alternada) foram acontecendo, simultaneamente, um dos candidatos comunicou-se com o monitor infomando os personagens selecionados em tais fases. Conforme a informação era repassada, o monitor foi espelhando as escolhas e preenchendo os campos do algoritmo para realização da predição.

A cada dado de entrada, o algoritmo gerava uma lista de vetores aleatórios sem alterar o(s) valor(es) que já foram inserido para comparar com a base de partidas e seu resultados. Gerando as redes neurais para o aprendizado, criando modelos convolucionais para elaborar a predição, mostrando os resultados na tela. Na Figura 20, vemos os valores de entrada (*input*), vetores de aleatórios gerados de cada inserção, o aprendizado realizado em *epoch* com sua acurácia e a lista de vetores com os valores de saídas (lista de recomendação).

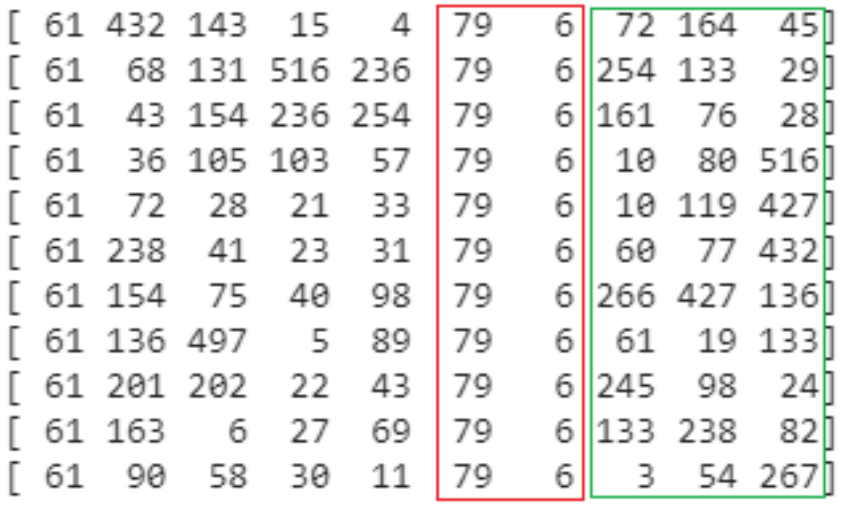
Figura 20 – Etapa de treinamento da MLP durante a primeira fase de execução do e-Coach

```
Qual lado você está jogando? 1 - Azul / 2 - Vermelho
Digite um número: 1
Personagem Azul: orianna
Sorteio Azul 1 = [[61, 432, 143, 15, 4], [61, 13, 27, 201, 157], [61, 12, 201, 107, 33],
Personagem Vermelho: gragas
Sorteio Verm 1 = [[79, 240, 82, 127, 51], [79, 59, 143, 1, 30], [79, 36, 83, 111, 10], [7
Personagem Vermelho: urgot
Sorteio Verm 2 = [[79, 6, 72, 164, 45], [79, 6, 22, 25, 238], [79, 6, 412, 32, 117], [79,
Epoch 56/60
1000/1000 [==============================] - 0s 195us/step - loss: 0.0016 - acc: 1.0000
Epoch 57/60
1000/1000 [==============================] - 0s 190us/step - loss: 0.0014 - acc: 1.0000
Epoch 58/60
1000/1000 [==============================] - 0s 186us/step - loss: 0.0012 - acc: 1.0000
Epoch 59/60
1000/1000 [==============================] - 0s 183us/step - loss: 0.0011 - acc: 1.0000
Epoch 60/60
1000/1000 [==============================] - 0s 190us/step - loss: 9.6026e-04 - acc: 1.0000
1000/1000 [==============================] - 0s 283us/step

acc: 100.00%
[ 61 432 143  15   4  79   6  72 164  45]
[ 61  68 131 516 236  79   6 254 133  29]
[ 61  43 154 236 254  79   6 161  76  28]
[ 61  36 105 103  57  79   6  10  80 516]
[ 61  72  28  21  33  79   6  10 119 427]
[ 61 238  41  23  31  79   6  60  77 432]
[ 61 154  75  40  98  79   6 266 427 136]
[ 61 136 497   5  89  79   6  61  19 133]
[ 61 201 202  22  43  79   6 245  98  24]
[ 61 163   6  27  69  79   6 133 238  82]
[ 61  90  58  30  11  79   6   3  54 267]
```

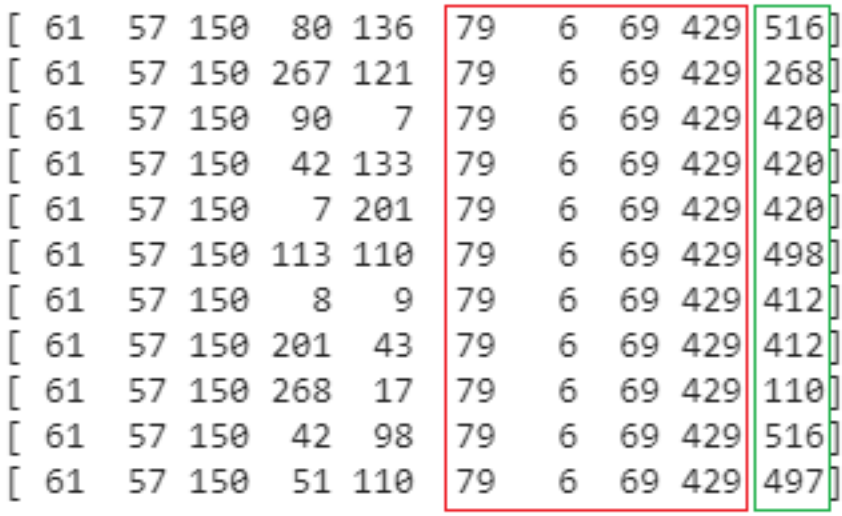
A lista de recomendação mostra 10 vetores com 10 elementos, cada número correspondendo a um campeão, sendo que os das duas equipes estão sendo representados em apenas um vetor. Os cinco primeiros elementos são designados para o time azul e os cinco últimos elementos são para o vermelho. Os primeiros valores do vetor, de ambas os times, são os valores do *input* feito pelo monitor. Por exemplo, para o equipe vermelho, na Figura 21, o destacado com linhas vermelhas são as colunas do *inputs* e o destaque em verde são as colunas para a recomendação do participante.

Figura 21 – Lista de vetores resultante da primeira triagem



Quando o algoritmo realizou a primeira triagem, o monitor repassou para os jogadores as opções de personagens, para que decidissem com qual campeão iriam jogar. Para a segunda triagem do e-Coach, os participantes, novamente, disseram ao monitor os novos personagens selecionados pela equipe inimiga e também quais que eles escolheram. Ao colocar os novos dados repassados, foi feita uma nova e última triagem, gerando outros vetores aleatórios, porém, sem alterar os valores do primeiro e segundo processamento. A Figura 22 segue a mesma explicação da figura anterior.

Figura 22 – Lista de vetores resultante da segunda triagem



Por fim, as últimas entradas foram informadas e o monitor as introduziu no algoritmo, que mostrou os dois times separadamente e depois agrupados para a partida (Figura 23). O papel do monitor, após o fim da seleção de campeões, foi aguardar o final da partida, escutar a comunicação entre os jogadores, observar o decorrer do jogo através do modo espectador e permanecer com o microfone desligado para não interferir.

Figura 23 – Resultado final das triagens

```
Time Azul Completo: [61, 57, 150, 113, 117]
Time Vermelho Completo: [79, 6, 69, 429, 1]
Times Completos: [61, 57, 150, 113, 117, 79, 6, 69, 429, 1]
```

No Apêndice B está simulado uma interação entre o algoritmo e a ordem de seleção de personagem para uma melhor abstração.

4.2.1 Coleta dos resultados

A coleta dos dados finais da partida foi feita pela própria página da Riot Games e no Blitz.gg (um site que utiliza a API para disponibilizar dados e estatísticas das partidas e jogadores de LOL), onde se disponibiliza gráficos de todos os fatores que ocorreram durante a partida. Como o experimento foi separado em duas etapas, precisamos comparar as duas formas, extraindo as informações que resultaram no time vencedor. A tabela a seguir mostra quais os campos de comparação que foi usado e sua descrição.

Tabela 4 – Dados de comparação das etapas e suas descrições

Campos	Descrição
Abates	Abates totais conquistado pela equipe na partida
AMA	O AMA total da equipe
Ouro	Ouro total conquistado pela equipe
Dano total	Dano total causado pela equipe sobre a equipe inimiga
Estruturas destruídas	Fortificações destruídas ao longo do mapa
Objetivos da selva	Monstros da selva para a partida

5 Resultados e Discussão

Este capítulo contém os resultados obtidos pelos experimentos realizados no capítulo anterior. Conforme dito na Subseção 4.2.1, era necessária a verificação dos dados entre partidas, observando as possíveis variáveis que pudessem impactar no final da partida. Foi montado um gráfico comparativo para cada item da Tabela 4 e as subseções abaixo trazem mais detalhes sobre os campos.

5.1 Resultados obtidos

Mesmo com partidas resultando em vitórias e derrotas, os resultados finais obtidos na análise trouxeram dados interessantes. Houve partidas, em ambos os experimentos, que resultaram em domínio total de um dos times, partidas que houve "virada" de tendência de resultados e partidas equilibradas. Tais resultados são importantes para o experimento, porque mostram as diversas situações que podem ocorrer em um confronto. Vejamos abaixo, com detalhes.

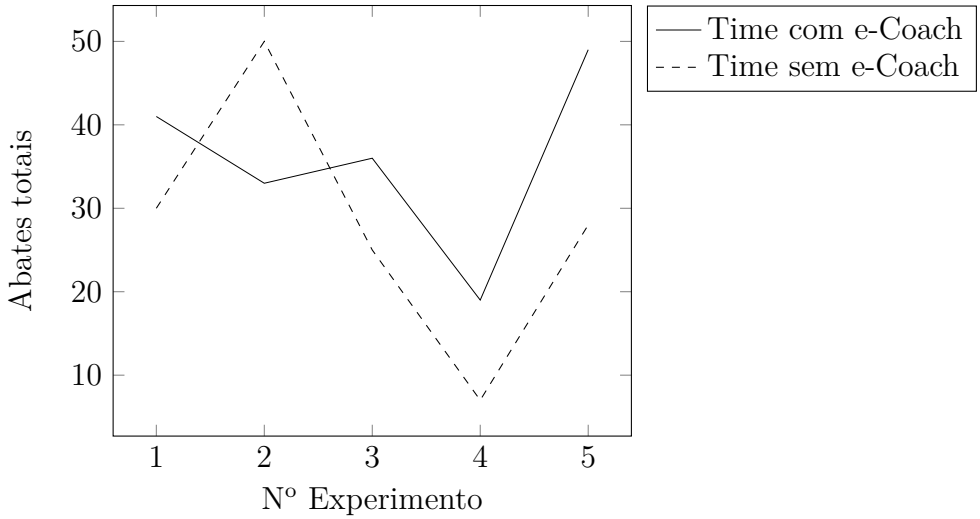
5.1.1 Abates totais

Os abates totais são as quantidades de vezes que o time experimental abateu seus adversários. Este dado é um dos maiores influenciadores de resultado, pois, quando um personagem abate um personagem inimigo, ele consegue ouro que possibilita a compra de itens para tornar mais forte o seu campeão e consequentemente influir poderosamente no resultado do jogo. A quantidade de abates totais obtidos estão representados na Tabela 5 e graficados na Figura 24

Tabela 5 – Tabela de análise de abates da equipe entre os tipos de experimentos

	EXP 1	EXP 2	EXP 3	EXP 4	EXP 5	Média	% Média
Com e-Coach	41	33	36	19	49	35.6	11.19%
Sem e-Coach	30	50	25	7	28	28	8.8%

Figura 24 – Gráfico de análise de abates da equipe entre os tipos de experimentos



Podemos observar que com o uso do algoritmo, obteve-se uma porcentagem média maior de abates, ou seja, as composições montadas pelo e-Coach proporcionou aos participantes uma possibilidade de obter vantagem no decorrer da partida.

5.1.2 Abates, mortes e assistências (AMA)

O AMA é feito a partir de um cálculo do placar individual de cada jogador. Somam-se os placares positivos (os abates com as assistências), pois pode ocorrer que embora a personagem não tenha conseguido abates, pode ter auxiliado de maneira decisiva para que seu colega de equipe conseguisse tal feito, o que conta favoravelmente na partida tanto para a equipe como para cada jogador que tenha participado.. Em seguida, a soma é dividida pelos pontos negativos (são as mortes), porque toda personagem abatida dá vantagem para a equipe inimiga. A fórmula abaixo mostra como é calculado o AMA.

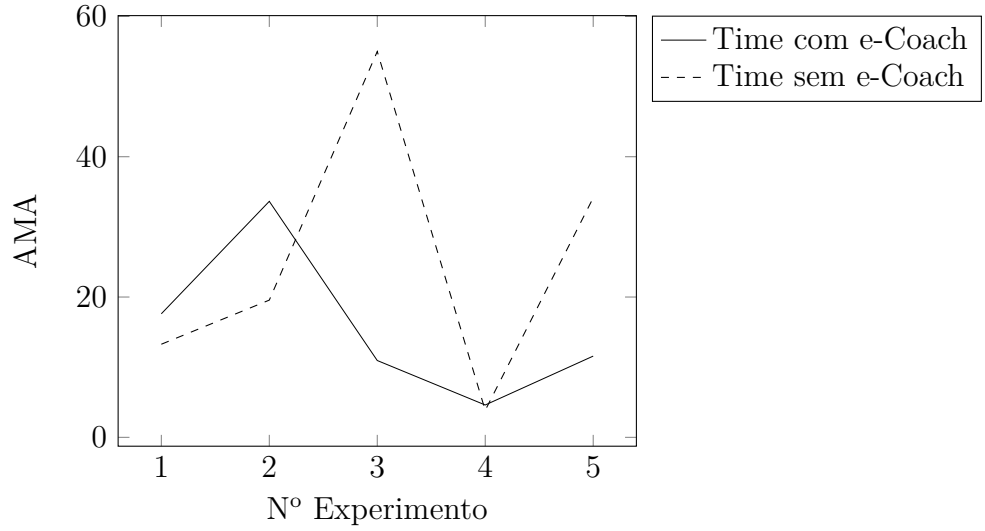
$$AMA = (abates + assistências)/mortes$$

O cálculo de AMA é utilizado entre comentaristas e analistas dos campeonatos de LOL, para identificarem quais dos jogadores profissionais estão realizando uma boa temporada e base para especularem acerca de qual equipe sairá vitoriosa nos confrontos. Para a investigação da experimentação, foi coletado o AMA individual de cada jogador e somados para gerar o AMA do time em cada partida, como mostrado na Tabela 6 e graficado na Figura 25.

Tabela 6 – Tabela de análise do AMA entre os tipos de experimentos

	EXP 1	EXP 2	EXP 3	EXP 4	EXP 5	Média	% Média
Com e-Coach	17.61	33.64	10.95	4.61	11.58	78.39	38.38%
Sem e-Coach	13.27	19.56	55	3.85	34.14	113.58	55.61%

Figura 25 – Gráfico de análise do AMA entre os tipos de experimentos



Quanto maior o AMA do jogador, indica que ele errou menos no decorrer da partida e diminuiu a chance de reviravolta do time adversário. Pode-se perceber que em alguns experimentos, com o e-Coach, o AMA foi superior, comparada ao experimento sem influência do código.

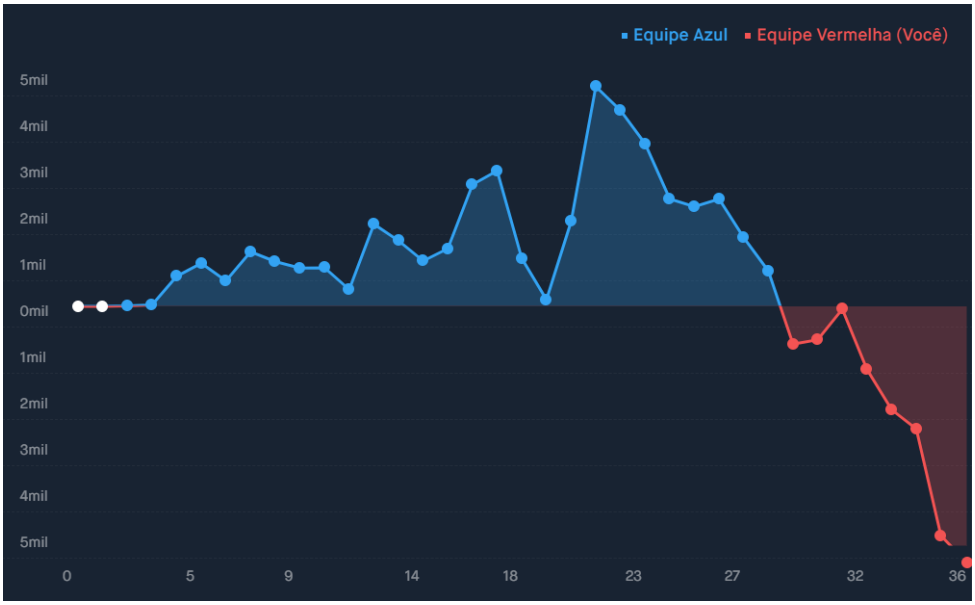
5.1.3 Ouro total

Um dos principais influenciadores de resultado. O ouro é fator resultante de aquisições dos jogadores dentro de jogo: abates inimigos, destruição de fortificações, abates de tropas, conquista de objetivos da selva e entre outros.

Em grande parte dos casos, a diferença de ouro entre as equipes é um fator crucial para o desenrolar da partida, porque a discrepância no montante de ouro obtido entre as equipes, significa que um time está com maior vantagem sobre o adversário.

No entanto, uma das partidas utilizando o algoritmo chamou atenção com uma reviravolta. A figura abaixo foi coletada de um site chamado Blitz.gg, onde foi gerado um gráfico da vantagem de ouro conquistado, em comparação entre as equipes, ao decorrer do tempo:

Figura 26 – Gráfico da diferença de ouro em um dos experimentos



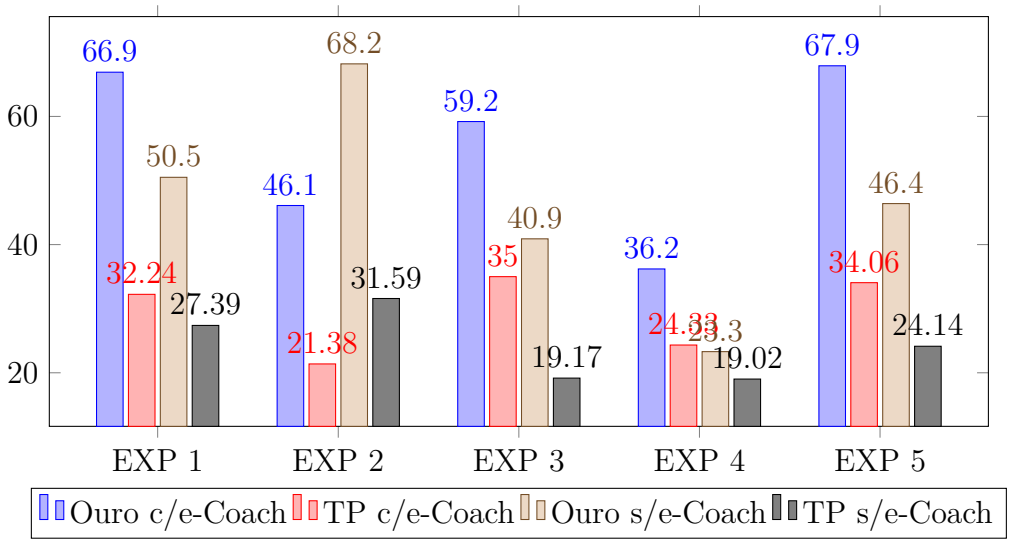
A equipe experimental é representada pela cor vermelha neste experimento. Na maior parte do tempo da partida, os jogadores do time azul conseguiram vantagem em relação ao vermelho, entretanto, a distância entre os pontos, que mostram uma queda, é sinal de que a equipe vermelha conseguiu tomar decisões que diminuíram a chance inimiga de ganhar. Nos minutos finais, quando o lado vermelho conquistou vantagem após falhas inimigas, eles finalizam a partida garantindo o triunfo.

A Tabela 7 o total de ouro obtido nas partidas de cada experimento.

Tabela 7 – Tabela de análise do ouro total entre os tipos de experimentos

	EXP 1	EXP 2	EXP 3	EXP 4	EXP 5	Média	% Média
Com e-Coach	66.9	46.1	59.2	36.2	67.9	55.26	9.07%
Sem e-Coach	50.5	68.2	40.9	23.3	46.4	45.86	10.92%

Figura 27 – Gráfico de análise do ouro total entre os tipos de experimentos



No gráfico de comparação entre experimentos (Figura 27), foi identificado uma composição recomendada que permite acumular muito dinheiro. O fator tempo de partida (TP), tem que ser considerado nesta análise, porque quanto mais tempo dura a partida, mais ouro é gerado.

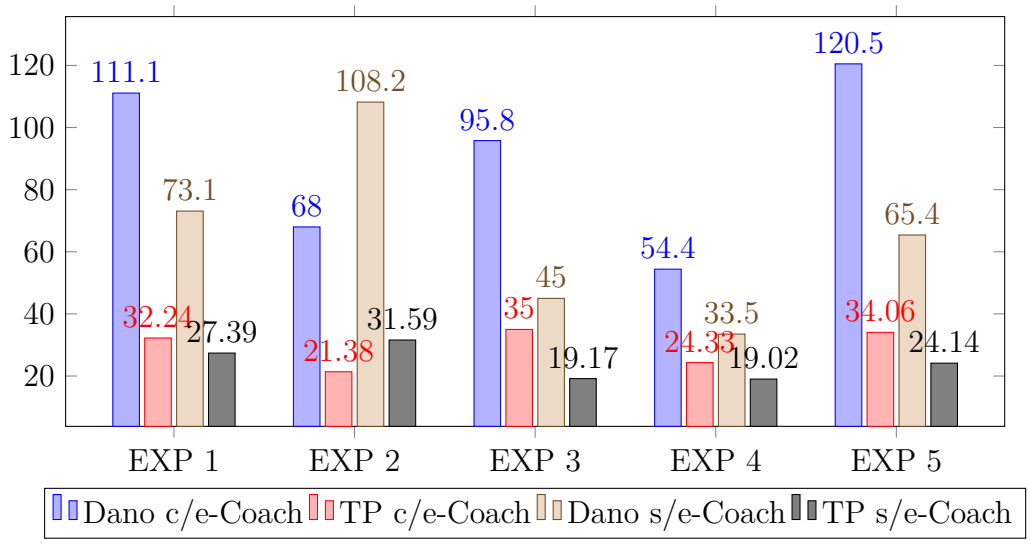
5.1.4 Dano total

O dano total causado pela equipe, significa que quando maior a quantidade de dano, mais forte o time está. Ele é um fator resultante da quantidade de ouro adquirida, porque é o ouro que vai permitir a compra de itens para deixar o personagem mais forte. A quantidade total de dano por experimento, está mostrado na Tabela 8

Tabela 8 – Tabela de análise do dano total causado entre os tipos de experimentos

	EXP 1	EXP 2	EXP 3	EXP 4	EXP 5	Média	% Média
Com e-Coach	111.1	68	95.8	54.4	120.5	89.96	11.60%
Sem e-Coach	73.1	108.2	45	33.5	65.4	65.04	8.39%

Figura 28 – Gráfico de análise do dano total causado entre os tipos de experimentos



O time recomendado pelo algoritmo, além de conseguir muitos abates, causa muito dano em seus adversários, como mostrado na Figura 28. O total de dano causado é vinculado ao tempo de partida, pelo mesmo motivo do fator anterior. A média percentual do dano, também foi inferior comparado ao experimento sem algoritmo.

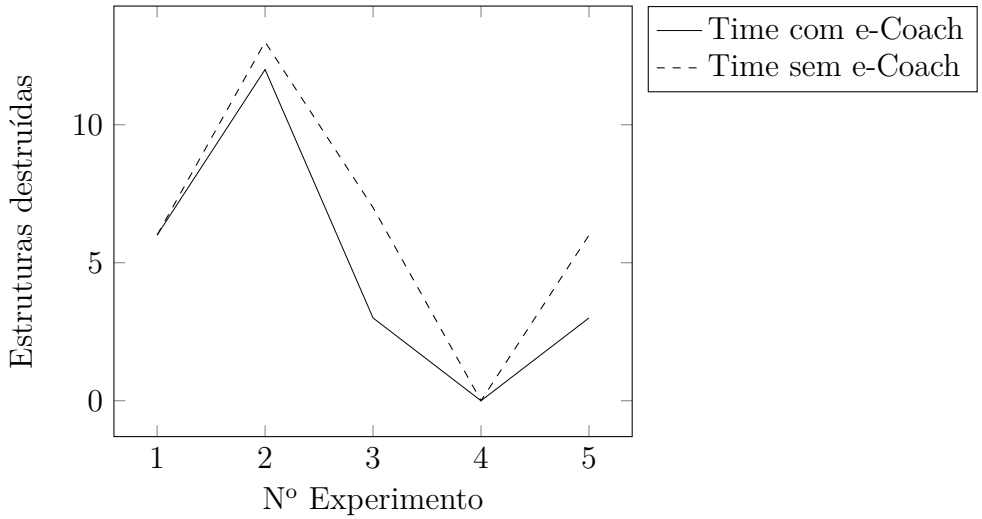
5.1.5 Estruturas destruídas

A derrubada de fortificações mostra o domínio de mapa conquistado pelo time que mais derrubou torres, o que é importante para pressionar a equipe adversária e conseguir chegar mais próximo ao Nexus. As estruturas contabilizadas incluem: torres de defesas e inibidores. A Tabela 9 quantifica as torres destruídas por experimento.

Tabela 9 – Tabela de análise das estruturas destruídas no mapa entre os tipos de experimentos

	EXP 1	EXP 2	EXP 3	EXP 4	EXP 5	Média	% Média
Com e-Coach	6	12	3	0	3	5	8.77%
Sem e-Coach	6	13	7	0	6	6.4	11.22%

Figura 29 – Gráfico de análise das estruturas destruídas no mapa entre os tipos de experimentos



Comparado ao o experimento sem uso do código, olhando o gráfico da Figura 29, a linha permanecem bem próximo ao pontilhado. Não se pode comparar os pontos neste gráfico pela maior quantidade de estruturas destruídas entre os experimentos, porque as partidas podem acabar pela desistência de uma das equipes, que declara o fim da partida sem levar o Nexus. Além disso, a equipe atacante é quem toma as decisões sobre quantidade de torres e inibidores que podem ser derrubadas. Para chegar até o Nexus não é preciso derrubar todas as torres de todas as rotas.

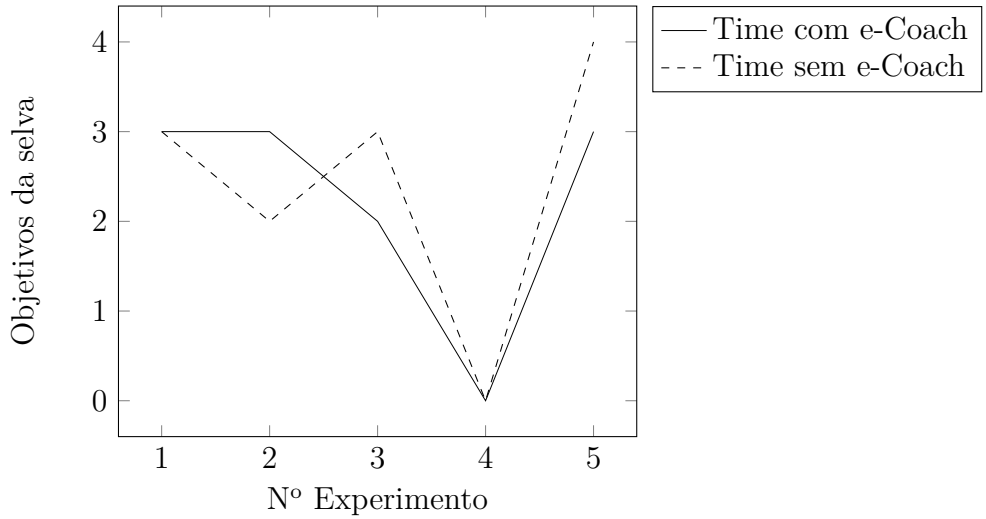
5.1.6 Objetivos da selva

Monstros neutros que se localizam na selva do mapa, quando abatidos podem oferecer bônus para a equipe e, consequentemente, vantagem na partida sobre o time inimigo. Os monstros contabilizados neste experimento incluem: arauto, dragões e barões que são os que mais influem no resultado (Tabela 10). Quanto mais objetivos são conquistados, mais bônus extras são obtidos para os personagens e mais poder.

Tabela 10 – Tabela de análise dos objetivos da selva no mapa entre os tipos de experimentos

	EXP 1	EXP 2	EXP 3	EXP 4	EXP 5	Média	% Média
Com e-Coach	3	3	2	0	3	5	21.73%
Sem e-Coach	3	2	3	0	4	6.4	27.82%

Figura 30 – Gráfico de análise dos objetivos da selva no mapa entre os tipos de experimentos



Os objetivos da selva são avaliados da mesma forma que a destruição das estruturas porque é feita a partir da tomada de decisões pelos jogadores. Alguns podem decidir não levar nenhum objetivo bônus e ganhar ou perder a partida. Também, como anteriormente, algumas partidas podem terminar por desistência de um dos lados. Porém, percebe-se pelo gráfico da Figura 30, que o sentido de um ponto para outro seguem a mesma direção. Acredita-se que esta tendência decorre de uma estratégia adotada pela equipe durante a execução do experimento.

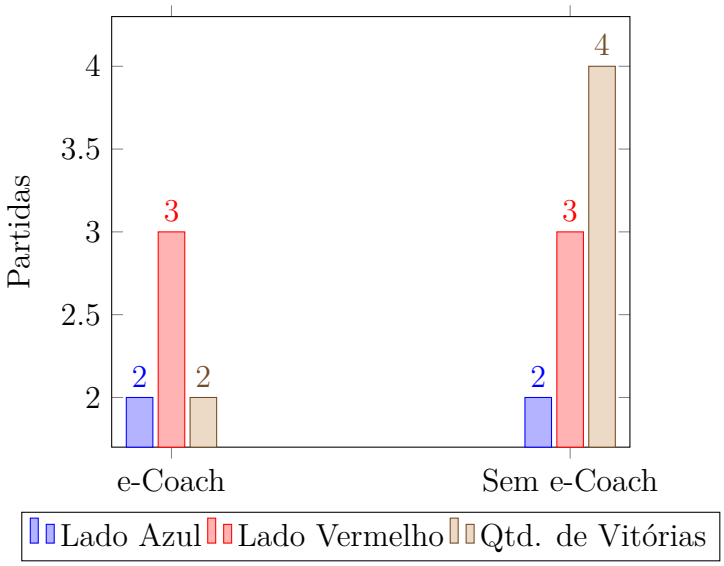
5.1.7 Considerações finais

Apesar de o algoritmo ter indicado escolhas que possibilitariam resultados indicativos de vitória, os dados finais apontaram que isto não foi determinante para que as vitórias se concretizassem, pois, no ambiente do LOL dezenas de outros fatores, dentre os quais sobressaem as decisões subjetivas de cada jogador preponderam sobre a escolha do campeão. A tabela abaixo mostra o resultado das partidas e graficados na Figura 31.

Tabela 11 – Resultados dos experimentos

	EXP 1	EXP 2	EXP 3	EXP 4	EXP 5	% Vitória
Com e-Coach	V	V	D	D	D	40%
Sem e-Coach	V	V	V	D	V	80%

Figura 31 – Gráfico de análise dos experimentos



5.2 Feedback dos jogadores

Após o final do experimento, os candidatos foram convidados a responder um formulário de *feedbacks* na qual se pretendeu que emitissem opiniões sobre as etapas, a recomendação feita pelo algoritmo, o sistema e sugestões de melhorias.

Sobre as etapas, uso do algoritmo e uso sem algoritmo, saber o nível de satisfação dos usuários atuando nas diferentes fases do experimento. Cerca de 80% ficaram mais satisfeitos jogando partidas sem o uso do algoritmo, devido ao critério de "segurança" com os personagens. Os candidatos gostavam de se sentirem mais seguros escolhendo personagens de sua preferência pois julgam que estariam menos propensos a errar, do que personagens recomendados pelo e-Coach. Além disso, alguns citaram ter gostado de experimentar coisas novas com o uso do sistema.

Sobre o e-Coach, foram questionados sobre os pontos que mais chamaram a atenção dos candidatos e a melhoria em sua jogabilidade. Um percentual de 60% destaca o ponto da composição formada pelo algoritmo e 80% acredita no melhor desempenho de jogabilidade. Isso se deve ao fato do algoritmo sugerir personagens, que às vezes fogem da zona de conforto do jogador. Mesmo assim, no decorrer da partida, o candidato pode descobrir do que o novo personagem pode fazer dentro de jogo. Todos os participantes concordam que a composição formada pelo algoritmo resultou na decisão da partida, mesmo sendo vitória ou derrota.

Sobre a construção do sistema, foram perguntados sobre o uso e recomendação da ferramenta de jogadores para outros jogadores que realizam as mesmas ações do e-Coach. Uma taxa de 60% dos candidatos alegam que usariam uma aplicação capaz de realizar a predição e a indicariam para outros *players* e 40% informou que não faria uso de tal

ferramenta, porém, apresentaria para outras pessoas.

Como a Riot Games faz alterações no LOL pensada na comunidade de jogadores, para este projeto não seria diferente. Foi pedido sugestões de como melhor a interação do usuário-sistema. Na Seção 6.1 estão detalhadas as mudanças e acrescentadas algumas sugestões dos usuários.

6 Conclusão

Neste trabalho foi implementado um algoritmo utilizando uma rede MLP para recomendação de times e personagens de League of Legends. Para validação do projeto, foram realizados testes de validação com jogadores em partidas reais.

Para execução do trabalho foi utilizado uma base de dados, disponível no Kaggle, de partidas ranqueadas de League of Legends feito por CAMPANELLI (CAMPANELLI, 2017). Foi realizado o tratamento e realocação das informações contidas nas tabelas.

Os resultados obtidos no experimento mostraram que o algoritmo pode determinar composições que auxiliem no desempenho dos atletas nas partidas conforme demonstram os gráficos estatísticos acima incluídos.

O algoritmo, a despeito de ainda apresentar algumas limitações próprias de uma versão experimental, obteve aprovação de jogadores que sugeriram a sua disponibilidade, viabilizando a possibilidade de desenvolvimento da aplicação para fins colaborativos entre a comunidade de invocadores que desejam melhorar sua jogabilidade ou almejar algo maior, como se tornarem *cyber* atletas em competições nacionais e internacionais.

6.1 Sugestões de trabalhos futuros

Os trabalhos futuros envolvem desenvolver uma aplicação disponível para jogadores e equipes profissionais de League of Legends, fornecendo aos mesmos uma nova maneira de melhorar sua jogabilidade. Tanto os invocadores que desejam se tornar grandes profissionais quanto os que apenas querem melhorar sua habilidade dentro de jogo poderão fazer uso desta ferramenta.

Para o futuro seria importante fazer uso da base de dados com partidas de campeonatos de League of Legends, porque envolve as escolhas de profissionais jogando contra outros profissionais. Essas tabelas podem trazer firmeza na sugestão do personagem.

Antes de executar essas propostas, devem ser feitas algumas correções no algoritmo e aplicação de novos métodos:

1. Reduzir o tempo de processamento para gerar uma recomendação, sem diminuir a quantidade total de amostras;

2. Na saída da recomendação, fazer conversão dos identificadores para seus respectivos nomes;

3. Implementar sistema de banimentos;

4. Implementação do Generative Adversarial Networks (GAN) no projeto, com o objetivo de gerar e recomendar um personagem mais apropriado a um determinado perfil.

O GAN é um modelo gerativo que foi proposto por GOODFELLOW (GOODFELLOW et al., 2014) para estimar modelos generativos através de um processo adversarial que utilizaram MLP para gerar seus neuros para o processamento e *backpropagation* para realização da aprendizagem. Podem ser aplicadas em geração de imagens, vídeos, texto, *text-to-image*.

O modelo GAN está sendo utilizado em estudos de processamento de imagens. Um estudo feito pela Synced (SYNCED, 2017), que atua no campo da inteligência artificial, utilizou *Conditional Generative Adversarial Network* (cGAN) para reconhecimento facial de uma pessoa em diferentes idades. Segundo a organização, o método pode ser usado para procurar pessoas desaparecidas, principalmente crianças, e melhor aplicação em outras pesquisas na área de reconhecimento facial. O uso do GAN no projeto poderia trazer consistência na escolha do personagem. Podendo sugerir uma única recomendação, em vez de várias.

Com as melhorias sendo implementadas, espera-se a utilização do e-Coach em diversas plataformas de dispositivos, com o objetivo de recomendar melhores opções de equipes e personagens para uma partida de LOL em um período desejável de tempo.

Referências Bibliográficas

CAMPANELLI, P. *League of Legends Ranked Matches*. 2017. Disponível em: <https://www.kaggle.com/paololol/league-of-legends-ranked-matches>. Acesso em: 24 mar. 2018. 40, 64

CANALTECH. *Nos EUA, universidade vai oferecer bolsa de até R$ 50 mil para gamers*. 2016. Disponível em: <https://canaltech.com.br/games/nos-eua-universidade-vai-oferecer-bolsa-de-ate-r-50-mil-para-gamers-84917/>. Acesso em: 15 abr. 2019. 24

CASTRO, L. N. de; ZUBEN, F. J. V. Redes neurais artificiais. 48 slides, 2001. Disponível em: <ftp://vm1-dca.fee.unicamp.br/pub/docs/vonzuben/ia006_03/topico5_03.pdf>. Acesso em: 02 jun. 2019. 36

ESPORTS NEWS. *Esports é reconhecido como profissão na China*. 2019. Disponível em: <https://esportsnews.com.br/esports-e-reconhecido-como-profissao-na-china/>. Acesso em: 15 abr. 2019. 24

ESPORTSEARNING. *Highest Overall Earnings*. 2019. Disponível em: <https://www.esportsearnings.com/players>. Acesso em: 12 mar. 2019. 19

FISHER, S. D. The rise of esports: League of legends article series. *White Paper*, Foster Pepper's Media, Entertainment e Games, p. 5, 2014. Disponível em: <https://www.foster.com/documents/foster-pepper-white-paper/riseofesportswhitepaper-fosterpepper.pdf>. Acesso em: 12 mar. 2019. 19

GOODFELLOW, I. J. et al. Generative adversarial networks. *arXiv* — Universidade de Montreal, Montreal, 2014. Disponível em: <arXiv:1406.2661>. Acesso em: 2 jun. 2018. 65

LOPES, R. A. S.; BRAGA, V. G. de M. Um sistema para o aprendizado automático de jogos eletrônicos baseado em redes neurais e q-learning usando interface natural. Universidade de Brasília, Distrito Federal, 2017. Disponível em: <http://bdm.unb.br/bitstream/10483/17240/1/2017_RafaelLopes_VictorGueresi_tcc.pdf>. Acesso em: 18 mai. 2019. 35

MEDIUM. *Campeonato de Counter-Strike Global Offensive em Katowice na Polônia*. 2018. Disponível em: <https://medium.com/csgo/primeiro-major-de-cs-go-de-2019-será-em-katowice-na-polônia-4eabaee7db17>. Acesso em: 12 mar. 2019. 25

MIGEZ, G.; MACULAN FILHO, N.; XAVIER, A. E. Otimização do algoritmo de backpropagation pelo uso da função de ativação bi-hiperbólica. Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2012. Disponível em: <https://www.marinha.mil.br/spolm/sites/www.marinha.mil.br.spolm/files/88683.pdf>. Acesso em: 02 jun. 2019. 36

NEWZOO. *Free 2018 Global Market Report*. 2018. Disponível em: <https://resources.newzoo.com/hubfs/Reports/Newzoo_2018_Global_Esports_Market_

Report_Excerpt.pdf?submissionGuid=47fc7ea2-641b-42d4-b1af-c99577cd0340>. Acesso em: 18 mai. 2019. 17, 18

NEWZOO. *Most Popular Core PC Games Global*. 2019. Disponível em: <https://newzoo.com/insights/rankings/top-20-core-pc-games/>. Acesso em: 10 abr. 2019. 19

PEREIRA, S. K. O videogame como esporte: Uma comparação entre esportes eletrônicos e esportes tradicionais. Universidade de Brasília, Distrito Federal, 2014. Disponível em: <http://bdm.unb.br/bitstream/10483/9385/1/2014_SilvioKazuoPereira.pdf>. Acesso em: 7 mar. 2019. 24

RIGO JUNIOR, L. O. et al. Aplicação de multi-layer perceptron para previsão de emissão de gases derivados de veículos a diesel. *Latin American Journal of Energy Research*, v. 3, n. 2, p. 11, 2017. Disponível em: <http://periodicos.ufes.br/lajer/article/download/15689/pdf>. Acesso em: 02 jun. 2019. 37

RIOT GAMES. *Anúncio do Evento All-Star 2018*. 2018. Disponível em: <https://br.lolesports.com/noticias/219a9b34-7eda-4b22-904a-b22d88aa748e>. Acesso em: 7 out. 2018. 34

RIOT GAMES. *Central de Aprendizado*. 2018. Disponível em: <https://centraldeaprendizado.br.leagueoflegends.com/>. Acesso em: 18 mai. 2019. 25

RIOT GAMES. *A Final da Segunda Etapa do CBLoL 2018 aos olhos do público*. 2018. Disponível em: <https://br.lolesports.com/noticias/final-da-segunda-etapa-do-cblol-2018-aos-olhos-do-publico>. Acesso em: 13 set. 2018. 20

RIOT GAMES. *LoL nos Jogos Asiáticos de 2018*. 2018. Disponível em: <https://br.lolesports.com/noticias/lol-nos-jogos-asiaticos-de-2018>. Acesso em: 10 mai. 2018. 25, 33

RIOT GAMES. *Novidades nas Ranqueadas 2019*. 2018. Disponível em: <https://br.leagueoflegends.com/pt/news/game-updates/competitive/novidades-nas-ranqueadas-2019>. Acesso em: 18 jan. 2019. 28, 31

RNA Perceptron Multicamadas. 14 slides, 2001. Disponível em: <http://www.barbon.com.br/wp-content/uploads/2013/08/RNA_Aula51.pdf>. Acesso em: 02 jun. 2019. 37

SANTOS, J. de D. Building a league of legends team recommender in go. *Medium*, 2018. Disponível em: <https://towardsdatascience.com/building-a-league-of-legends-champions-recommender-system-in-go-and-how-to-deploy-\it-in-the-cloud-1ee7a4fb55ee>. Acesso em: 12 mar. 2019. 23

SPORTV. *Mega-sena dos eSports: lista reúne os campeonatos mais bem pagos de 2018*. 2018. Disponível em: <https://sportv.globo.com/site/e-sportv/noticia/mega-sena-dos-esports-lista-reune-os-campeonatos-mais-bem-pagos-de-2018.ghtml>. Acesso em: 18 mai. 2019. 17

SUN, Y. Motivation to play esports: Case of league of legends. Universidade da Carolina do Sul, 2017. Disponível em: <https://scholarcommons.sc.edu/cgi/viewcontent.cgi?article=5119&context=etd>. Acesso em: 28 mar. 2019. 20

SYNCED. Face aging with conditional generative adversarial networks. *Medium*, 2017. Disponível em: <https://medium.com/syncedreview/face-aging-with-conditional-generative-adversarial-networks-d41076379047>. Acesso em: 02 jun. 2019. 65

UOL JOGOS. *Como se tornar um pro player de "League of Legends"? Especialistas contam*. 2016. Disponível em: <https://jogos.uol.com.br/ultimas-noticias/2016/11/16/como-se-tornar-um-pro-player-de-league-of-legends-especialistas-contam.htm>. Acesso em: 18 mai. 2019. 20

VASCONCELOS, B. F. B. de. Poder preditivo de métodos de machine learning com processos de seleção de variáveis: uma aplicação às projeções de produto de países. 2017. Disponível em: <http://repositorio.unb.br/bitstream/10482/23995/1/2017_BrunoFreitasBoynarddeVasconcelos.pdf>. Acesso em: 18 mai. 2019. 34

YIN, J. League of legends: Predicting wins in champion select with machine learning. *Medium*, 2018. Disponível em: <https://medium.com/@jihan_yin/league-of-legends-predicting-wins-in-champion-select-with-machine-learning-6496523a7ea7>. Acesso em: 12 mar. 2019. 23

ZHENGXING, C. et al. Player skill decomposition in multiplayer online battle arenas. *arXiv* — Universidade da California, Los Angeles, 2017. Disponível em: <arXiv:1702.06253>. Acesso em: 18 mai. 2019. 20

A Apêndice

Figura 32 – Propriedades de pasta das tabelas separadamente

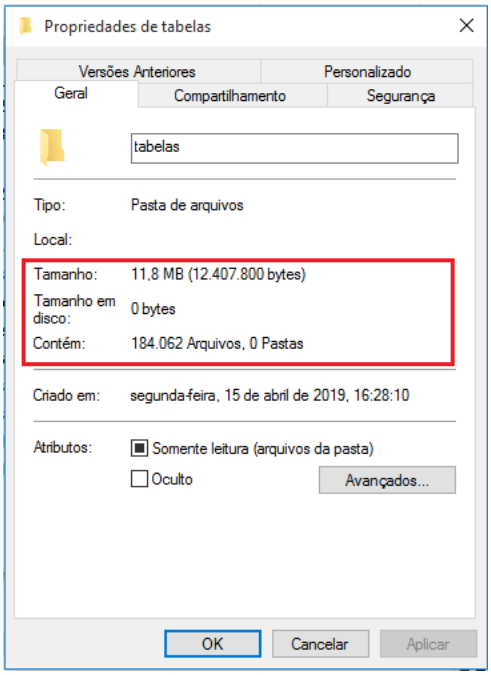
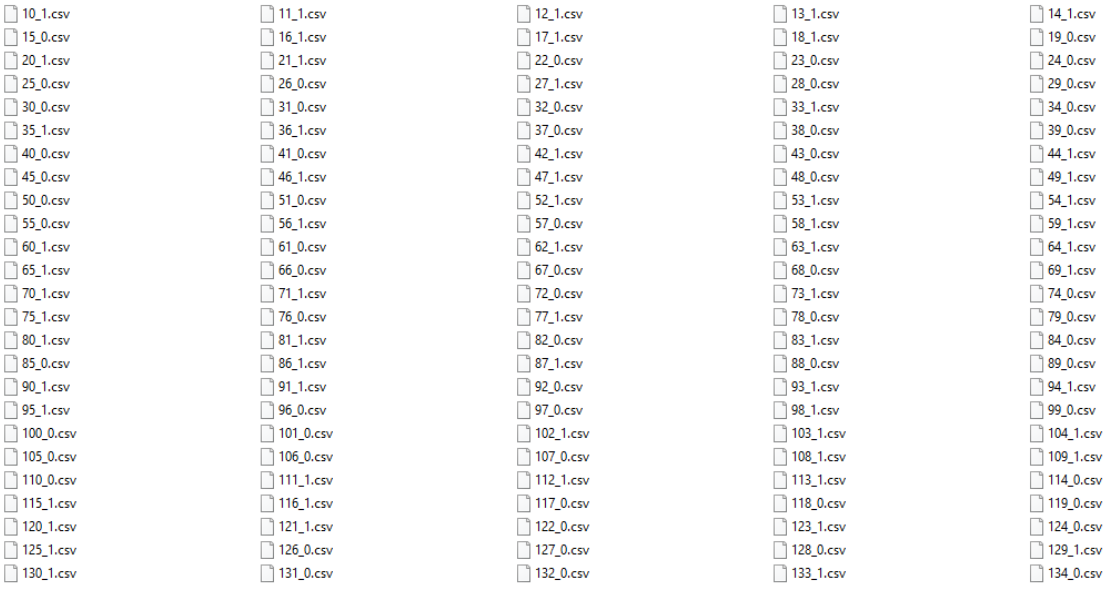


Figura 33 – Parte das tabelas geradas com suas nomenclaturas



B Apêndice

As imagens a seguir demonstram as etapas da simulação. Os destaques em amarelo simbolizam escolhas sem interferência do algoritmo e os destaques em cor verde representa as próximas escolhas que serão sugeridas com interferência do e-Coach. Nesta simulação, o time que estaria utilizando o algoritmo é o azul.

As três primeiras seleções são livres. É necessário esses primeiros valores para a análise e dar as primeiras recomendações.

Figura 34 – Primeira etapa de simulação

Time Azul
?
?
?
?
?

Time Vermelho
?
?
?
?
?

Após as primeiras três inserções feitas, o e-Coach vai criar um sorteio aleatório com uma lista de vetores com quatro sugestões possíveis campeões. Em seguida, a rede neural vai fazer o treinamento do modelo, conforme mostrado na Figura 20 da Seção 4.2, comparando os valores presentes no vetores com os resultados presentes na base de dados e sugerir para o time azul as dez melhores opções de personagens de sua recomendação

obtida. Os jogadores do lado azul, observarão os campeões sugeridos e farão as escolhas de dois deles, como está destacado em verde na figura abaixo.

Depois de fazerem as escolhas, será inserido os personagens escolhidos por ambos os lados durante a execução do código, para a realização da terceira etapa de de processamento e escolhas.

Figura 35 – Segunda etapa de simulação

Time Azul
Mordekaiser
?
?
?
?

Time Vermelho
Thresh
Pyke
?
?
?

Nesta terceira etapa, o algoritmo irá recalcular as novas opções com base nas inserções anteriores feitas. Como o próximo quadro de seleção do time azul, oportuniza a escolha de mais dois campeões, o e-Coach irá recomendar tais escolhas restantes. Posteriormente, com os personagens finais selecionados dos dois lados, os jogadores realizam a fase de preparação para dar início à partida.

Figura 36 – Terceira etapa de simulação

Time Azul
Mordekaiser
Shen
Annie
?
?

Time Vermelho
Thresh
Pyke
Ekko
Kha'Zix
?