Fernanda Tostes Marana – 4471070
Matheus Aparecido do Carmo Alves – 9791114
Thais Bianchini – 9791010

10-Minutes MOBA: Previsão de vitória em League of Legends utilizando Redes Neurais

Projeto 1 da Disciplina SCC0276 - Aprendizado de Máquina

Professora Doutora Roseli Aparecida Francelin Romero

Universidade de São Paulo - USP

Instituto de Ciências Matemáticas e de Computação - ICMC

Bacharelado em Ciências de Computação
São Carlos, SP

15 de maio de 2020

Sumário

1 Introdução

League of Legends (LoL) representa um dos mais populares jogos eletrônicos da atualidade. Desenvolvido pela Riot Games, o jogo é distribuído gratuitamente e, em 2016, contava com uma comunidade de 100 milhões de jogadores ativos mensalmente (1). Além de representar uma opção de divertimento casual, o jogo apresenta um cenário competitivo responsável por organizar campeonatos que atingem 4 milhões de espectadores e fomentam a industria dos *Eletronic-Sports* (E-Sports) (2).

O jogo é classificado como um *Multiplayer Online Battle Arena* (MOBA) que, usualmente, é caracterizado como um jogo de estratégia no qual duas equipes formadas por cinco jogadores batalham entre si a fim de destruir a base inimiga. Em *League of Legends*, o objetivo final é destruir o cristal dentro da base inimiga, chamado de *Nexus*.

Antes de se alcançar o *Nexus*, um time deve destruir, no mínimo, 3 torres de rota, 1 inibidor (estrutura que inibe a fortificação de soldados do time adversário, chamado *minions*) e as duas torres que protegem o *Nexus*. O mapa (conhecido como *Summoner's Rift*) possui 3 rotas disponíveis para avanço ao objetivo do jogo: rota inferior, do meio e o topo. A Figura 1 apresenta o mapa do jogo com indicações de rotas e estruturas dentro do jogo separado por equipe.

Estas estruturas representam os principais objetivos do jogo, os quais devem ser cumpridos para vitória. Contudo, existem objetivos secundários que agregam vantagens para o time que os cumprem e favorecem a vitória. Entre eles são os abates de soldados inimigos, monstros neutros e adversários. De maneira direta, todos esses abates favorecem a economia do time (que aumenta sua quantidade de recurso disponível para compra de itens). Os itens, por sua vez, aumentam a capacidade destrutiva dos jogadores e, portanto, facilitam a conquista dos objetivos principais.

Frente a este contexto, o objetivo deste projeto é capacitar um algoritmo de aprendizado de máquina para previsão de vitória e derrota em uma partida de *League of Legends*. Para isso, será utilizado um conjunto de dados que contém as informações de conquista de objetivos (primários e secundários) para ambas as equipes nos 10 primeiros minutos de partida. O conjunto de dados fornece, aproximadamente, as informações de 10 mil partidas de alto nível e 38 parâmetros para análise destes 10 minutos de partida. Em outras palavras, as partidas são compostas por jogadores de elo diamante à mestre (classificações de desempenho destes dentro do modo competitivo do jogo), que possuem grande domínio das mecânicas e conhecimento de estratégias dentro do jogo, além de informações relevantes sobre as decisões tomadas pelos jogadores a coleta de dados.

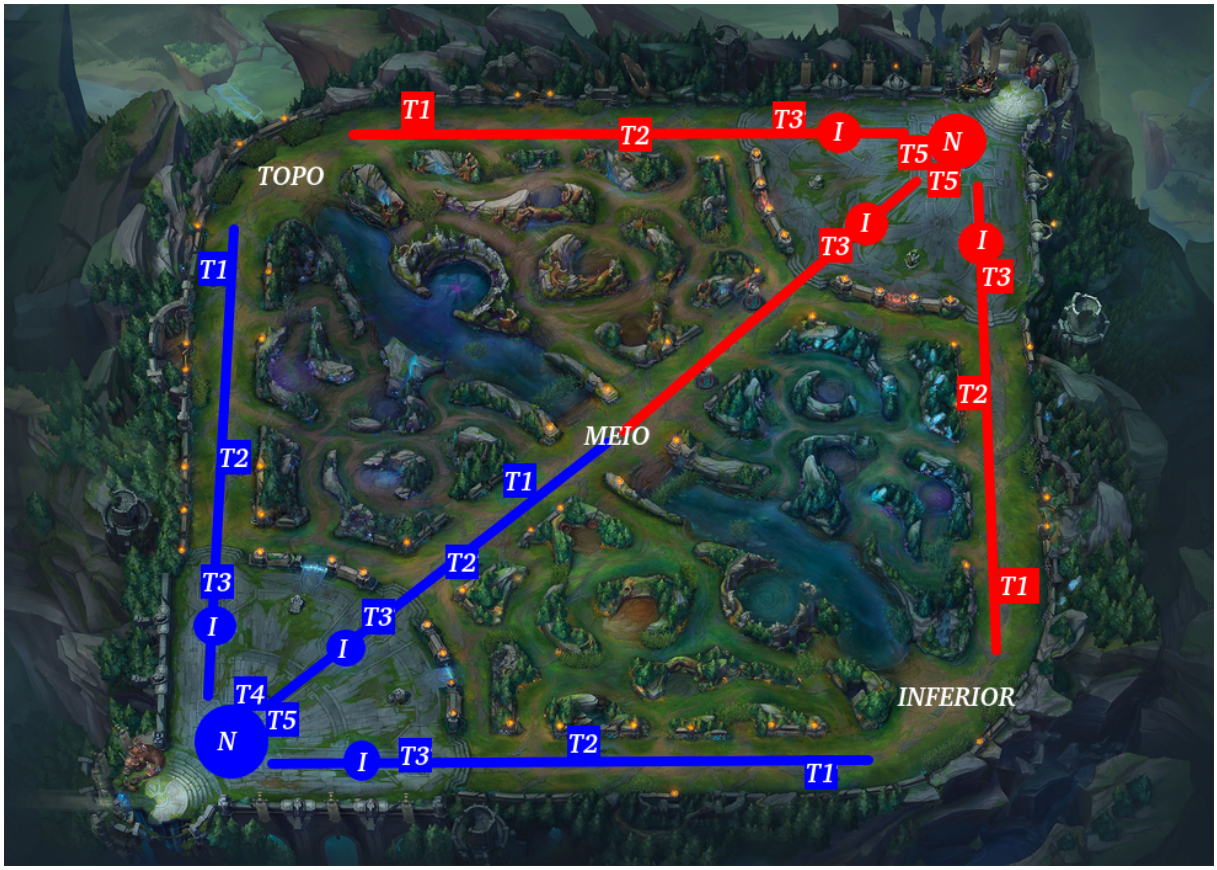


Figura 1 – Mapa de *Summoner's Rift* com a indicação de rotas e estruturas dentro do jogo. As indicações são feitas em Azul e em Vermelho para se distinguir os times participantes. As etiquetas com prefixo "T" representam as torres, "I" representam os inibidores e "N" os *Nexus*.

O conjuntos de dados está disponível gratuitamente na plataforma do *Keagle*[1] e a implementação da proposta a ser apresentada neste documento está disponível publicamente no GitHub[2].

Por fim, este documento está organizado e será apresentado da seguinte forma: na Seção 2 serão apresentados os trabalhos relacionados que abordam ou resolvem um problema similar ao proposto; na Seção 3 será conduzida uma descrição dos dados presentes na base de dados utilizada; na Seção 4 será descrita a metodologia empregada para a resolução do problema proposto; na Seção 5, os resultados obtidos serão apresentados e discutidos; por fim, na Seção 6 uma conclusão será feita com base nas informações e resultados apresentados ao longo deste documento.

[1] Kagle-League of Legends 10 min:
<https://www.kaggle.com/bobbyscience/league-of-legends-diamond-ranked-games-10-min>

[2] GitHub-10 minutes MOBA (League of Predictions):
<https://github.com/Micanga/league_of_predictions>

2 Trabalhos Relacionados

Realizar previsões em cenários que possuem agentes de parâmetros desconhecidos e disponibilizam informações parciais (sejam temporais ou espaciais) representa um relevante desafio dentro de pesquisas em computação. Para o desenvolvimento, teste e aplicação de técnicas que lidam com tais limitações, um dos objetos de estudo utilizados são os jogos do estilo MOBA.

Em (3), é proposto um estudo de caso do jogo Dota 2 (outro popular MOBA disponibilizado gratuitamente pela *Valve*). Considerando somente a escolha do herói e a duração do jogo, o modelo proposto busca determinar quais as chances de vitória para cada time de acordo com essas escolhas. Note que estas escolhas ocorrem antes do início da partida de fato. Utilizando o algoritmo Naive Bayes combinado à validação cruzada dos dados, o modelo apresentou uma precisão de 77% na predição de vitória. Frente a estes resultados, acredita-se que é possível se alcançar resultados similares para resolução de problemas dentro do mesmo contexto. Portanto, neste projeto, prospecta-se se desenvolver um modelo preditivo que consiga se equiparar ao resultado alcançado em (3), porém aplicando, somente, a informação dos 10 minutos iniciais de partida (i.e., sem conhecimento de escolhas pré-partida).

Outra forma de se aplicar essas limitações de conhecimento ao contexto de predição é considerar uma abordagem de aprendizado online. Em (4), esta abordagem é aplicada e um modelo de predição em tempo real para partidas de nível profissional é proposto. Considerando o cenário competitivo do jogo Dota 2, o objetivo deste trabalho foi definir se, utilizando dados mistos de partidas profissionais e amadoras, é possível construir um modelo capaz de oferecer predições precisas para partidas profissionais. O modelo proposto é uma Florestas de Decisão Aleatória e Regressão Logística (*Random Florest*) que alcança uma acurácia de 74.59% nas predições. Esta acurácia média é atingida considerando-se cenários cujo o algoritmo observa informações da partida por um intervalo de 20 a 30 minutos antes da classificação. Neste documento, a proposta é se alcançar um resultado similar, considerando partidas competitivas de LoL e, com somente 10 minutos de partidas, apontar um vencedor.

Semelhantemente à proposta presente neste documento, (5) tem como objetivo a predição de vitória em partidas de *League of Legends*. Porém, este trabalho utiliza dados coletados antes e durante uma partida, visando realizar a classificação por meio do aprendizado combinado destes dois tipos de informação. Dentre os modelos propostos, o que apresenta maior acurácia, de 95.52%, é uma Floresta de Decisão Aleatória. O resultado é obtido pela aplicação do método de validação cruzada. A proposta neste documento

não visa utilizar conhecimento anterior a partida ou dados após 10 minutos de jogo, diferenciando-se deste trabalho trivialmente.

(6) utiliza uma rede neural recorrente e intervalo de 5 minutos de dados (em uma partida de League of Legends) para treiná-la. Os 5 minuto escolhido definem os intervalos entre os minutos 0 a 5, 5 a 10, 10 a 15, 15 a 20 e 20 a 25 de jogo. A melhor acurácia obtida antes dos 10 minutos iniciais foi de $69,68\%$. Já para intervalos após os 10 minutos iniciais, a acurácia atingiu valores de 83.56% de acerto. Note que este trabalho propõe uma análise móvel da proposta deste documento, prevendo o resultado de uma partida com base na variação de recursos dentro de um intervalo de tempo em uma partida. Sucintamente, este projeto propõe uma análise sobre os valores brutos dos parâmetros (conquistas dos times) até os 10 minutos (ao invés de abordar a variância destes parâmetros) e, prospecta-se que está abordagem supere a margem de acerto apresentada para os 10 minutos iniciais.

Por fim, uma outra abordagem, apresentada em (7), possui como principal objetivo a criação de um sistema de recomendação para guiar o jogador na escolha dos heróis que farão parte de um time no jogo DOTA 2. Para então testar a eficácia desse sistema baseado em regras de associação, uma rede neural foi construída para prever o resultado da partida, de acordo com os heróis escolhidos. De acordo com o apresentado, o trabalho apresenta uma acurácia de 88.63% nas predições em caso onde suas recomendações foram seguidas. Este documento propõe a predição, porém sem a utilização de informações adicionais (como análise de “Escolhas e Banimentos”, anterior a partida).

3 Descrição da base de dados

O conjunto de dados utilizados para realização deste projeto, "*League of Legends Diamond Ranked Games (10 min)*", está disponível gratuitamente na plataforma Keagle.

Este conjunto de dado contém informações estatísticas dos 10 primeiros minutos de partida de, aproximadamente 10 mil jogos (no modo competitivo do jogo, "*Solo Queue*") de elos altos (intervalo definido entre os elos Diamante I à Mestre). Essa característica é importante pois define-se que os jogadores possuem um nível de jogabilidade parecido.

Cada jogo é único e possui um "gameId" associado, que permite a realização de buscas na API da Riot para validação das informações ou coleta de informações adicionais.

O conjunto apresenta 19 parâmetros por time, em um total de 38 parâmetros, coletados após 10 minutos de jogo. Esses parâmetros incluem abates de inimigos, mortes em jogo, ouro, experiência dos jogadores (dentro do jogo), nível, entre outros. Ademais, a classificação é dada binariamente na coluna *blueWins* tal que o valor 1 indica que o time azul ganhou a partida e o valor 0 que o time vermelho obteve vitória. Não existem valores faltantes dentro deste conjunto de dados.

Explicitamente, a Figura 2 apresenta parte dos dados e parâmetros existentes na base e a Tabela 1 descreve todos os parâmetros contidos neste conjunto.

	gameId	blueWins	blueWardsPlaced	blueWardsDestroyed	blueFirstBlood	blueKills	blueDeaths	blueAssists	blueEliteMonsters	blueDragons	...
0	4519157822	0	28	2	1	9	6	11	0	0	...
1	4523371949	0	12	1	0	5	5	5	0	0	...
2	4521474530	0	15	0	0	7	11	4	1	1	...
3	4524384067	0	43	1	0	4	5	5	1	0	...
4	4436033771	0	75	4	0	6	6	6	0	0	...
...	...	...	...	...	...	...	...	...	...	...	...
9874	4527873286	1	17	2	1	7	4	5	1	1	...
9875	4527797466	1	54	0	0	6	4	8	1	1	...
9876	4527713716	0	23	1	0	6	7	5	0	0	...
9877	4527628313	0	14	4	1	2	3	3	1	1	...
9878	4523772935	1	18	0	1	6	6	5	0	0	...

9879 rows × 40 columns

Figura 2 – Data Frame gerado apresentando, parcialmente, os dados e parâmetros da base de dados.

Note que todos os dados são numéricos (inteiros), logo nenhum tratamento, quanto a representação de palavras por números, precisa ser feito para a aplicação da base em algoritmos de aprendizado de máquina tradicionais.

Coluna	**Parâmetro**	**Descrição**
0	gameId	identificador da partida.
1	blueWins	valor alvo da classificação que indica a vitória do time azul (1) ou vermelho (0).
2/21	blueWardsPlaces/ redWardsPlaces	número de sentinelas usadas pelo time.
3/22	blueWardsDestroyed/ redWardsDestroyed	número de sentinelas inimigas destruídas pelo time.
4/23	blueFirstBlood/ redFirstBlood	valor binário que indica o o primeiro abate da partida.
5/24	blueKills/ redKills	número de abates realizados pelo time.
6/25	blueDeaths/redDeaths	número de mortes do time.
7/26	blueAssists/ redAssists	número de assistências realizadas pelo time.
8/27	blueEliteMonsters/ redEliteMonsters	número de monstros de elite abatidos pelo time.
9/28	blueDragons/ redDragons	número de dragões abatidos pelo time.
10/29	blueHeralds/ redHeralds	número de *Heralds* abatidos pelo time.
11/30	blueTowersDestroyed/ redTowersDestroyed	número de torres destruídas pelo time.
12/31	blueTotalGold/redTotalGold	total de ouro da equipe.
13/32	blueAvgLevel/redAvgLevel	nível médio do time.
14/33	blueTotalExperience/ redTotalExperience	total de experiência adquirida pelo time.
15/34	blueTotalMinionsKilled/ redTotalMinionsKilled	número de *minions* abatidos pelo time.
16/35	blueTotalJungleMinionsKilled/ redTotalJungleMinionsKilled	número de monstros da selva abatidos pelo time.
17/36	blueGoldDiff/ redGoldDiff	diferença de ouro entre os times.
18/37	blueExperienceDiff/ redExperienceDiff	diferença de experiência entre os times.
19/38	blueCSPerMin/ redCSPerMin	número de *minions* abatidos por minuto pelo time.
20/39	blueGoldPerMin/ redGoldPerMin	ouro por minuto arrecadado pelo time.

Tabela 1 – Tabela que apresenta todos os parâmetros disponíveis e sua discrição.

4 Metodologia

O primeiro passo definido para desenvolvimento metodológico deste projeto foi a realização de uma análise do conjunto de dados a ser utilizado. Desta forma, separou-se o conjunto entre as classes disponíveis (vitória do time azul e vitória do time vermelho) para análise da estratificação dos dados quanto a classe. A Figura 3 apresenta essa distribuição dos dados entre as classes.

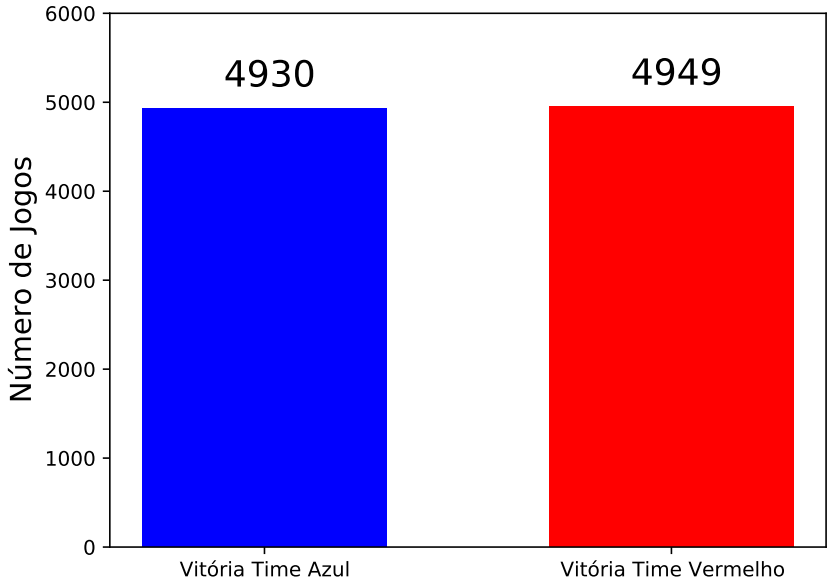


Figura 3 – Gráfico de barras com a comparação entre o número de vitórias do time azul e vermelho no conjunto de dados utilizado.

Pela análise do gráfico de barras, é possível notar que o conjunto de dados alvo é desbalanceado, isto é, possui quantidades diferentes de exemplos para cada classe. Esta característica incorpora ao problema a necessidade da elaboração de uma estratégia de desenvolvimento que não prejudique o treinamento ou utilização futura do modelo. Desta forma, neste projeto será aplicada a técnica de validação "*Stratified K-Fold Cross-Validation*".

O *K-Fold Cross-Validation* é um processo iterativo de treino e teste de algoritmos de aprendizado de máquinas o qual realiza a validação do modelo pelo cruzamento de K conjuntos dados para treinamento e teste. Isto é, através da definição de K conjuntos de dados (os quais representam subconjuntos do conjunto de dados original), realiza-se, de maneira iterativa e cruzada, o treinamento e teste do modelo selecionando-se um dos K conjuntos para teste e os outros K-1 treinamento. A Figura 4 apresenta um esquema do processo de *K-Fold Cross-Validation*. Note que neste processo, obtém se como resultado final para validação do modelo a média de desempenho obtido nas K classificações realizadas (ponderada ou não).

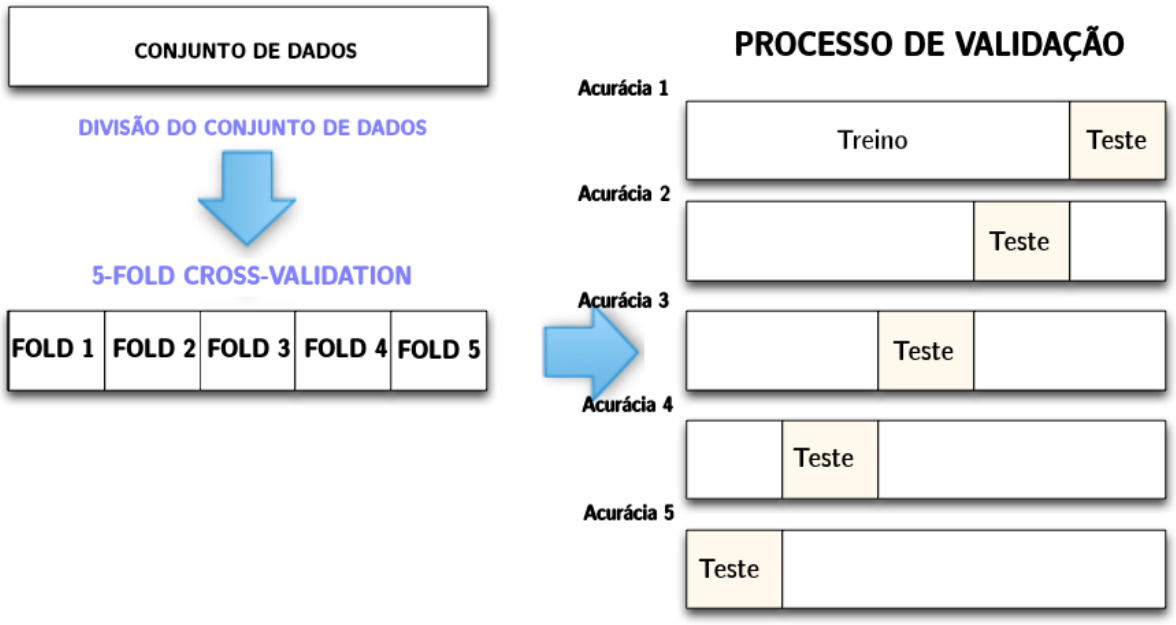


Figura 4 – Esquema que ilustra o processo de K-Fold Cross-Validation.

Analogamente, o "*Stratified K-Fold Cross Validation*" aplica o mesmo processo definido pelo *K-Fold Cross-Validation*, contudo ele garante que os dados serão estratificados pela classe. Em outras palavras, garante-se que todos os K-Folds definidos possuirão a mesma quantidade de exemplos para cada classe existente. Esta abordagem possibilita a utilização de modelos treinados sobre base de dados com quantidades desiguais de exemplos por classe e evita a criação de um modelo enviesado.

Escolhida a abordagem para validação do modelo, iniciou-se a escolha dos algoritmos de aprendizado de máquinas para resolução do problema. De maneira direta, optou-se pela implementação de uma rede neural *Multi-Layer Perceptron* (MLP) dado a natureza do problema e finalidade do modelo (classificação).

Frente a esta escolha, o passo essencial para utilização desta rede sobre o problema é a realização de uma análise de possíveis arquiteturas para representação, aprendizado e classificação do problema. A fim de se alcançar resultados satisfatórios, foram definidas 24 arquiteturas distintas para a realização de testes. A Tabela 2 apresenta todas as arquiteturas MLPs construídas, que possuem variação no (i) número de camadas escondidas, (ii) número de neurônios nas camadas e, (iii) no método de otimização de pesos aplicado dentro da rede durante o processo de treinamento.

Para escolha da melhor rede neural, será aplicado o método *Stratified K-Fold Cross Validation* sobre todas as arquiteturas, coletadas suas acurácias médias e desvio padrão (sobre o conjunto de testes) para comparação. A rede com maior acurácia média e menor desvio padrão será selecionada como arquitetura principal do modelo. Para garantir resultados comparáveis, foram utilizados os mesmos K-Folds para todas as arquiteturas.

# Identificador da Arquitetura	**Arquitetura MLP**	**Otimização de Pesos**
#01	3:3:3:3:3:3:1	gradiente estocástico
#02	3:3:6:6:6:3:1	gradiente estocástico
#03	3:3:6:9:6:3:1	gradiente estocástico
#04	4:4:4:4:4:4:1	gradiente estocástico
#05	4:4:8:8:8:4:1	gradiente estocástico
#06	4:4:8:12:8:4:1	gradiente estocástico
#07	38:38:38:38:38:38:1	gradiente estocástico
#08	38:38:76:76:76:38:1	gradiente estocástico
#09	38:38:76:128:76:38:1	gradiente estocástico
#10	38:38:76:152:152:152:76:36:19:9:4:2:1	gradiente estocástico
#11	38:38:76:152:152:152:152:76:36:19:9:4:2:1	gradiente estocástico
#12	38:38:38:38:38:38:1	gradiente estocástico
#13	38:38:76:76:76:38:2:1	gradiente estocástico
#14	38:38:76:152:152:152:76:36:19:9:4:2:1	gradiente estocástico
#15	3:3:6:6:6:3:1	Quasi-Newton
#16	3:3:6:9:6:3:1	Quasi-Newton
#17	3:3:6:9:12:12:12:9:6:3:2:1	Quasi-Newton
#18	4:4:8:8:8:4:1	Quasi-Newton
#19	4:4:8:12:8:4:1	Quasi-Newton
#20	4:4:8:12:12:12:8:4:2:1	Quasi-Newton
#21	38:38:38:38:38:38:1	Quasi-Newton
#22	38:38:76:76:76:38:1	Quasi-Newton
#23	38:38:76:128:76:38:1	Quasi-Newton
#24	38:38:76:152:152:152:76:36:19:9:4:2:1	Quai-Newton

Tabela 2 – Arquiteturas MLP projetadas para teste, validação e aplicação no prooblema.

Por fim, considerando-se que a melhor arquitetura está entre o conjunto definido para testes, outro passo que pode aprimorar os resultados obtidos na classificação do problema é a aplicação de técnicas de refinamento (pré-processamento) dos dados. Para este projeto, foram definidas duas abordagens para pré-processamento e teste: (i) utilizando a técnica de normalização via Média e Desvio Padrão, e; (ii) pela aplicação do método de Análise de Componentes Principais (ACP).

(i) Normalização via Média e Desvio Padrão

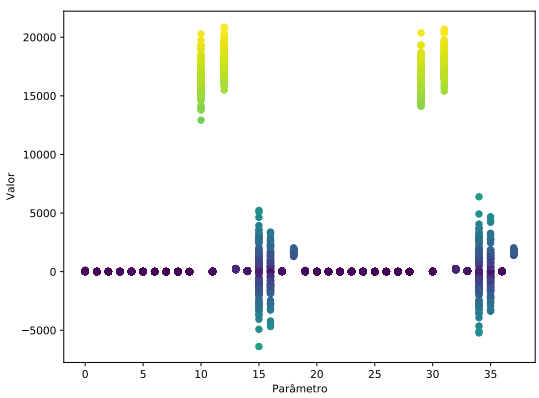
A normalização via média e desvio padrão busca retirar vieses que podem existir em um conjunto de dados que será utilizado para treinamento de uma rede neural. Especificamente, em conjunto de dados onde a ordem numérica dos parâmetros é distinta, aqueles que possuem ordem muito maior ou menor do que outros podem enviesar a classificação para resultados memorizados ou equivocados, devido ao ponderamento “injusto” definido durante o processo de atualização dos pesos na rede. Para evitar esses problemas durante o treinamento, a técnica de normalização via

Média e Desvio Padrão define que todos os parâmetros devem ser centrados sobre a média e respeitar o intervalo de desvio padrão. A Equação 4.1 apresenta a transformação feita sobre os dados para garantir estas propriedades e, consequentemente, a normalização dos dados.

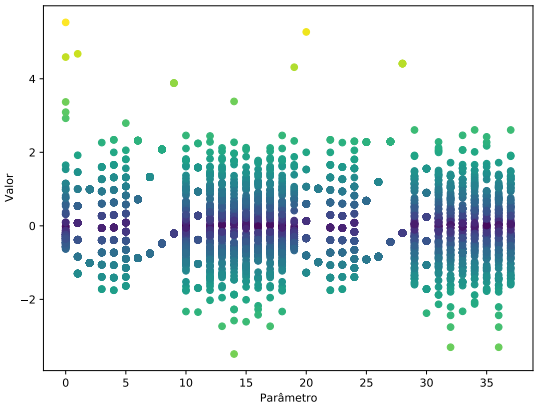
$$z_{\theta,i} = (x_{\theta,i} - \mu_\theta)/\sigma_\theta \qquad , i = 0, 1, ..., |\theta|, \ \forall \ \theta \in \Theta \tag{4.1}$$

sendo $x_{\theta,i}$ o valor do parâmetro θ para o i-ésimo exemplo contido no conjunto de dados, μ_θ a média dos valores para o parâmetro θ, σ o desvio padrão dos valores para o parâmetro θ e $z_{\theta,i}$ o valor normalizado de $x_{\theta,i}$.

Esta transformação aplicada, sobre a base de dados deste projeto, pode ser visualizada na Figura 5, sendo a Figura 5a responsável por apresentar o intervalo sem a normalização Média Desvio-Padrão sobre os dados e, a Figura 5b, responsável por apresentar o intervalo após normalização dos dados.



(a) Antes da normalização.



(b) Após normalização.

Figura 5 – Intervalo de abrangência para cada parâmetro do conjunto de dados antes e após a aplicação da normalização via Média e Desvio Padrão. Foram utilizados as 100 primeiras amostras para construção do gráfico.

Como é possível notar, o intervalo antes da normalização variava de -5000 à 20000, além de definir grupos de intervalos para cada parâmetro. Com a normalização, o intervalo está compreendido entre os valores -3 e 5, apresentando uma melhor distribuição dos valores durante todo o intervalo. Com essa abordagem, a base de dados a ser utilizada poderá superar os problemas apresentados anteriormente.

(ii) Aplicação da técnica de ACP

A técnica de Análise de Componentes Principais (ACP) é responsável por aplicar um algoritmo de análise estatística sobre um conjunto de dados permitindo a redução da dimensionalidade de seu espaço paramétrico. Esta redução de dimensionalidade acarreta em uma perda de informação, contudo o ACP permite um ajuste estatístico

para minimizar esta perda o quanto for desejável. Isso é feito por meio da seleção de parâmetros e variáveis que maximizam a variância do conjunto de dados (ou seja, apresentam informações diferentes para diferentes exemplos) e não definem fortes correlações dentro da base.

Descritivamente, a aplicação da técnica sobre o conjunto de dados deste projeto (que contem 38 parâmetros) gerou a remoção de 34 parâmetros do conjunto; reduzindo o tamanho do conjunto de parâmetros para 4. Está redução gerou uma perda de, somente, 1% da informação original do banco de dados. Isto é, com 4 parâmetros é possível apresentar os exemplos pertencentes a base com 99% da informação da base original. Esta redução possui como principal vantagem uma melhora associada à eficiência de treinamento e teste do algoritmo de aprendizado de máquina, além de permitir a remoção de possíveis parâmetros que confundem o modelo durante o aprendizado. A Figura 6 mostra a curva de informação cumulativa associada ao número de parâmetros na base de dados.

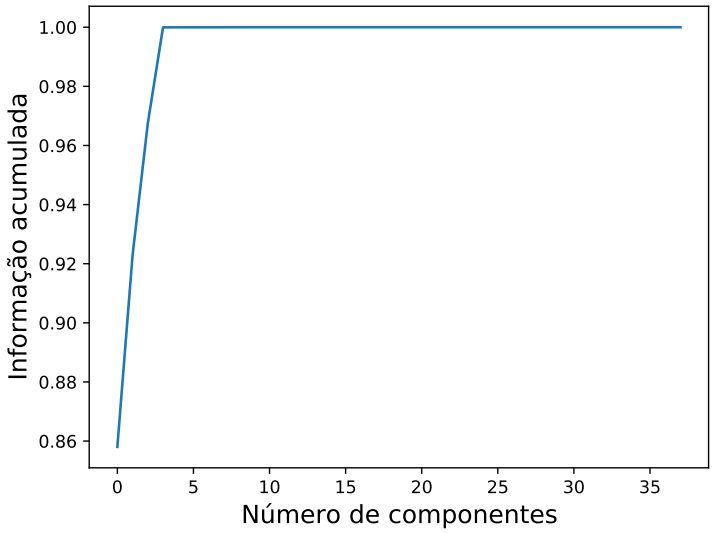


Figura 6 – Curva de informação cumulativa com o aumento do número de parâmetros gerada pela aplicação do algoritmo ACP.

Como se pode observar, os primeiros componentes detém grande parte da informação existente na base de dados enquanto os outros aumentam vagarosamente esta quantidade de informação.

Os parâmetros que descrevem 99% da informação são:

1. ***redGoldDiff*:** diferença entre a economia do time vermelho comparado à economia time azul;
2. ***redExperienceDiff*:** diferença entre a experiência total do time vermelho comparado à experiencia total do time azul;

3. ***blueExperienceDiff*:** diferença entre a experiência total do time azul comparado à experiencia total do time vermelho;
4. ***redTotalGold*:** economia total associada às conquistas do time vermelho.

A escolha desses parâmetros pelo algoritmo ACP apresenta uma sumarização importante sobre os parâmetros escolhidos para montagem do conjunto de dados original. De maneira direta, a união desses parâmetros será, ao final, a representação de quanto dinheiro e experiência foram coletados por cada time até os 10 minutos de jogo. Analisando os outros parâmetros, todos eles possuem como recompensa, em partida, uma quantidade de ouro e/ou uma quantidade de experiência associada ao se completar este objetivo (primário ou secundário). Desta forma, é notavelmente lógico se sumarizar esses dados sobre parâmetros que descrevem estas recompensas e não, somente, sobre a quantidade de objetivos completados.

Descritas as técnicas de pré-processamento de dados, inicia-se a fase de escolha da arquitetura (mesclando pré-processamento de dados, a arquitetura em si e a acurácia obtida) e a fase de coleta de resultados, que serão posteriormente analisados.

5 Resultados

Primeiramente, coletou-se os resultados de classificação para todas as arquiteturas de redes definidas na Seção 4. De maneira direta, a Tabela 3 apresenta os 7 melhores resultados alcançados ordenados de forma decrescente pela acurácia obtida.

ID	Arquitetura MLP	Pré Processamento	Otimização de Pesos	Acurácia Teste
#05	4:4:8:8:8:4:1	PCA(0.99)	gradiente estocástico	0.710 ± 0.005
#08	38:38:76:76:76:38:1	Nenhum	gradiente estocástico	0.701 ± 0.014
#02	3:3:6:6:6:3:1	PCA(0.95)	gradiente estocástico	0.676 ± 0.089
#07	38:38:38:38:38:38:1	Nenhum	gradiente estocástico	0.674 ± 0.045
#09	38:38:76:128:76:38:1	Nenhum	gradiente estocástico	0.675 ± 0.043
#01	3:3:3:3:3:3:1	PCA(0.95)	gradiente estocástico	0.671 ± 0.087
#13	38:38:76:76:76:38:2:1	*Scaling*	gradiente estocástico	0.659 ± 0.006

Tabela 3 – Acurácia de teste obtida na aplicação de diferentes Arquiteturas MLP com diferentes pré-processamentos de dados para classificação. Os resultados estão ordenados de forma decrescente pela acurácia obtida.

Feita esta coleta e frente a uma análise das melhores acurácias obtidas, definiu-se a arquitetura (4:4:8:8:8:4:1) como base para execução de novos experimentos. Desta forma, o primeiro passo foi garantir que esta arquitetura não oferece um resultado enviesado ou memorizado. Para isto, foi feito um estudo sobre a curva de aprendizado do modelo com a variação do número de amostras disponíveis para treinamento. A Figura 7 apresenta o resultado deste estudo.

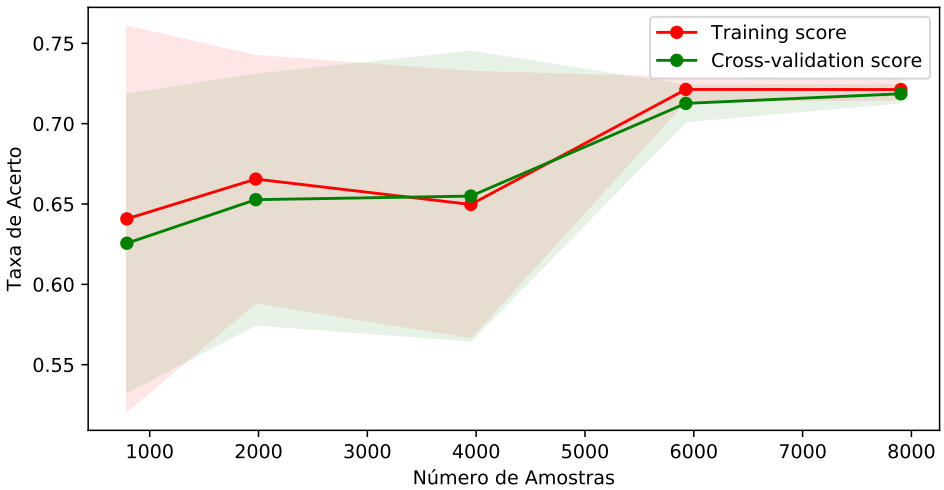


Figura 7 – Curva de aprendizado da MLP com a variação de número de amostras para treinamento (que representam 10%, 25%, 50%, 75% e 100%, respectivamente).

Como se pode notar, a curva de treino e de teste encontram-se próximas e convergindo a um ponto em comum. Este fato mostra uma generalização do conhecimento via treino que melhora a representação de qualquer caso teste futuro. Ademais, com o aumento do número de amostras disponíveis para treino, o desvio padrão do resultado diminui, aumentando a confiabilidade sobre o grau de generalidade da rede.

Após a analise do grau de generalidade da rede, partimos para os resultados de classificação. Para uma primeira avaliação, a rede foi treinada e classificou o conjunto de teste binariamente entre vitória do time azul e vitória do time vermelho. A Figura 8 apresenta o resultado obtido neste teste na representação de uma matriz de confusão.

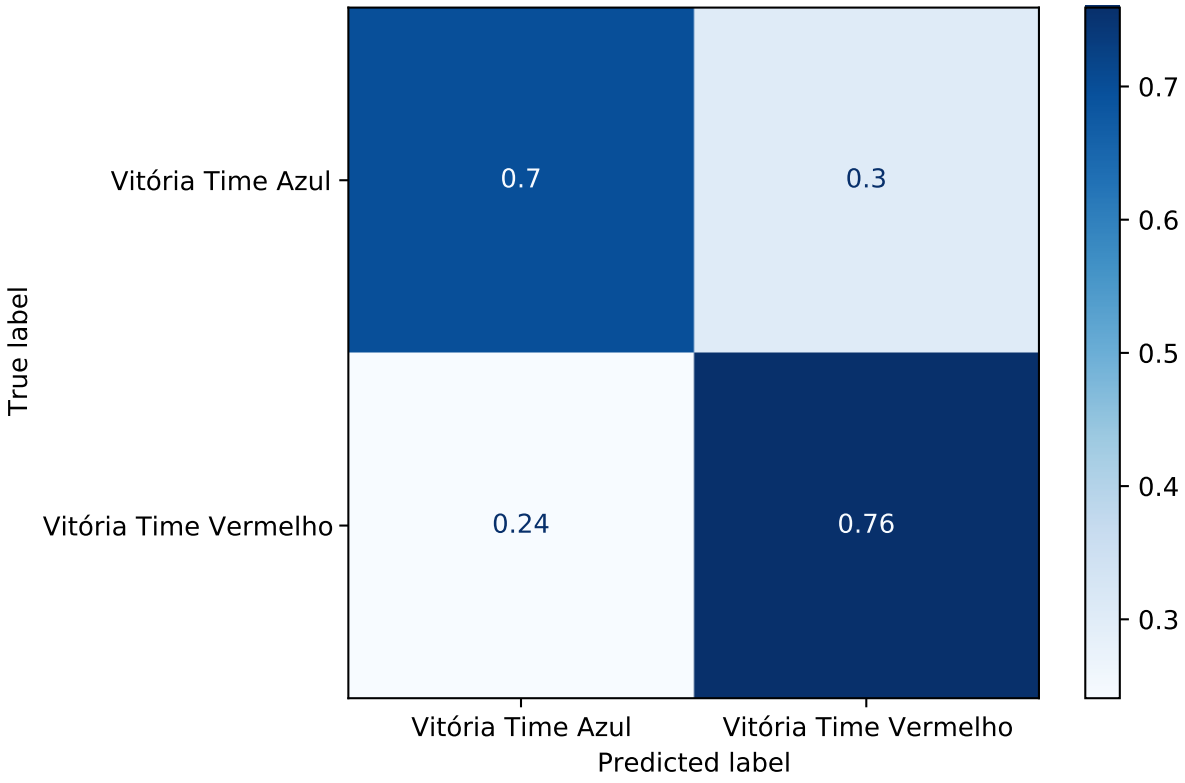


Figura 8 – Matriz de confusão da melhor arquitetura de rede encontrada para o problema. A classificação neste caso é binária.

Note que o resultado neste classificação binária gerou uma acurácia de 70% na classificação de exemplos onde o time azul ganhou e 76% em exemplos onde o time vermelho obteve vitória. Este resultado é satisfatório, dado que foi possível classificar corretamente 73% dos casos de teste com, somente, a informação dos 10 primeiros minutos das partida.

Reitera-se que esta taxa de acerto é relevante, uma vez que dado o contexto de estudo (partidas de LoL) existem parâmetros que não são possíveis de medir ou registrar devido sua natureza ocasional (como falhas na internet de jogadores ou interferências durante a partida) ou intrínseca à capacidade de resposta humana (como a realização de boas jogadas ou erros mecânicos dos jogadores).

Como próximo passo, decidiu-se testar a rede utilizando uma predição probabilística para todos os casos de teste, visando agregar um valor mais semântico aos resultados. Este resultado pode ser usado por jogadores para avaliação de seu desempenho em tempo real, considerando um cenário onde este modelo retornaria seu resultado de classificação durante uma partida. A Figura 9 apresenta o resultado obtido na predição probabilística.

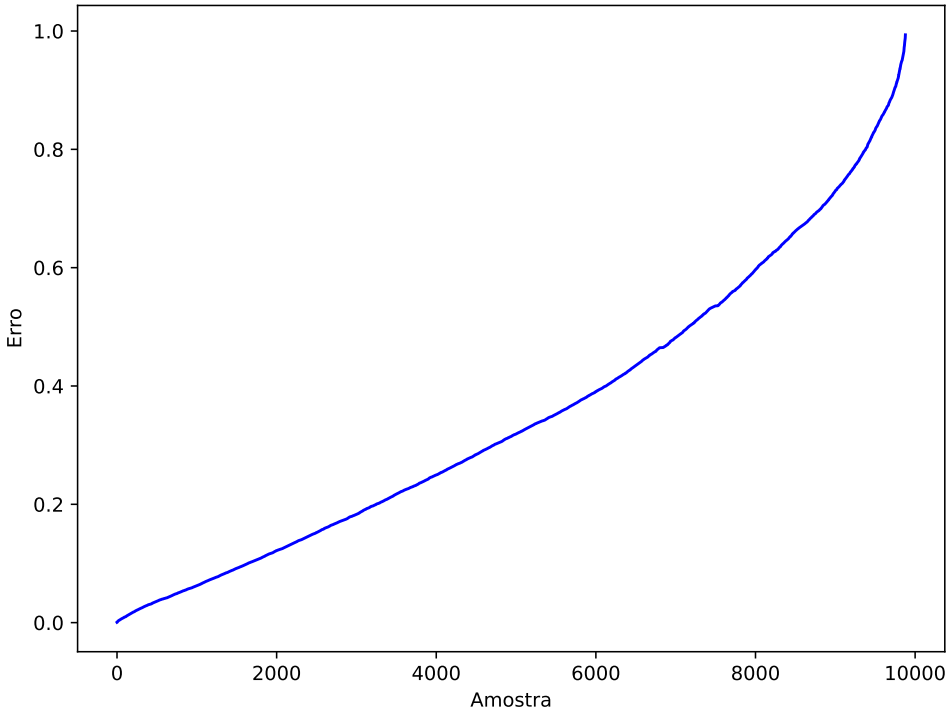


Figura 9 – Taxa de erro na predição probabilística. As amostras estão organizadas de maneira ordenada pelo erro na predição.

Por fim, analisando esse gráfico, pode-se observar que cerca de 1/3 das amostras receberam uma classificação com erro menor ou igual a 25% de erro (considerando a classificação probabilística). Em outras palavras, tem-se que que, com 10 minutos de jogo, o modelo MLP é capaz de acertar com 75% ou mais de certeza a vitória de um time ou outro para 1/3 dos jogos analisados. Dividindo a taxa de probabilidade de ganhar em 4 intervalos, a Tabela 4 apresenta o número de jogos dentro de cada intervalo.

Intervalo	**Confiança do modelo**	**Número de amostras**
1	0-25%	677
2	25-50%	2038
3	50-75%	3217
4	75-100%	3947

Tabela 4 – Quantificação do número de amostras que pertencem a cada intervalo de confiança da predição do modelo dado a classificação probabilística do modelo MLP e o erro (percentual) associado a cada predição.

6 Conclusão

League of Legends é um dos maiores e mais populares jogos da atualidade. Denominado um jogo MOBA, para se jogar LoL é necessário domínio mecânico, conhecimento técnico e estratégico sobre o jogo, além da capacidade de se trabalhar em equipe a fim de conquistar a vitória; buscando coletar recursos e completar objetivos que favoreçam o desempenho de seu time como um todo.

Em jogos como este, diversos fatores (sejam eles mensuráveis ou não) podem alterar o resultado final de uma partida. Esses fatores podem ser, por exemplo, a conquista de um objetivo secundário durante a partida, melhor comunicação entre os jogadores ou problemas com internet, no final todos influenciaram e modificaram o resultado. Frente a isso, um problema interessante que emerge neste contexto é a possibilidade de se prever o resultado de um jogo utilizando informações mensuráveis e parciais das partidas.

Desta forma, este projeto propõe a resolução deste problema através da aplicação de algoritmos de aprendizados de máquina com informações parciais. Para isso, será utilizada uma base de dados disponível na plataforma *Keagle* que apresenta um conjunto de dados com a informações dos 10 minutos iniciais de, aproximadamente, 10 mil partidas de alto nível no modo competitivo de LoL.

Através da implementação e refinamento de uma rede neural MLP e dos dados, foi proposto um modelo preditivo de vitória ou derrota sobre o contexto definido. O modelo apresenta uma acurácia de 71% para os casos disponíveis no conjunto de dados cuja confiança de classificação (ou probabilidade de vitória/derrota) é maior ou igual à 75% para cerca de 40% dos casos. Estes resultados são relevantes e superam as acurácias de publicações em cenários semelhantes, utilizando informações parciais para treinamento e classificação das partidas e não são considerados parâmetros latentes (não mensuráveis, como capacidade mecânica do jogador ou condições de jogo) para enriquecimento da predição.

Este projeto permite o desenvolvimento de diversas plataformas de suporte para jogadores e analistas de partidas em *League of Legends*. Como trabalho futuro, propõe se o desenvolvimento de um método integrado que enriqueça a predição de vitória ou derrota considerando parâmetros pré-partida. Em outras palavras, pretende-se aplicar técnicas de mineração de dados para avaliação de escolhas feitas quanto qualidade de *Escolhas e Banimentos*, escolha de campeões e sinergia de time antes do inicio do jogo. Com isso, prospecta-se melhorar a capacidade classificatória do modelo proposto neste documento e disponibilizar um sistema de avaliação e de recomendação para a comunidade de LoL.

Referências

1 RIOT games reveals league of legends has 100 million monthly players. 2016. <https://www.forbes.com/sites/insertcoin/2016/09/13/riot-games-reveals-league-of-legends-has-100-million-monthly-players/>. Acessado: 10 de Maio de 2020. Citado na página 2.

2 MUNDIAL de LoL teve o maior pico de espectadores entre os torneios disputados em 2019. 2020. <https://www.espn.com.br/esports/artigo/_/id/6480355/mundial-de-lol-teve-o-maior-pico-de-espectadores-entre-os-torneios-disputados-em-2019>. Acessado: 10 de Maio de 2020. Citado na página 2.

3 Almeida, C. E. M. et al. Prediction of winners in moba games. In: *2017 12th Iberian Conference on Information Systems and Technologies (CISTI)*. [S.l.: s.n.], 2017. p. 1–6. Citado na página 4.

4 HODGE, V. J. et al. Win prediction in multi-player esports: Live professional match prediction. *IEEE Transactions on Games*, York, 2019. Citado na página 4.

5 R, A. et al. Victory prediction in league of legends using feature selection and ensemble methods. In: *2019 International Conference on Intelligent Computing and Control Systems (ICCS)*. [S.l.: s.n.], 2019. p. 74–77. Citado na página 4.

6 SILVA, A. L.; PAPPA, G. L.; CHAIMOWICZ, L. Continuous outcome prediction of league of legends competitive matches using recurrent neural networks. In: . [S.l.: s.n.], 2018. Citado na página 5.

7 HANKE, L.; CHAIMOWICZ, L. A recommender system for hero line-ups in moba games. In: *AIIDE*. [S.l.: s.n.], 2017. Citado na página 5.