Previsão de vitória em uma rodada de CS:GO

Universidade Federal do Rio de Janeiro

Bernardo Maiorano Vieira
Thiago Guimarães Rebello Mendonça de Alcântara

06 de agosto de 2022

Contents

1 Resumo

Este artigo versa sobre um projeto realizado durante a disciplina de Redes Neurais na UFRJ no período de 2022/1. Aqui, trate-se do problema de um *dataset* do kaggle que fornece dados reais de partidas do jogo CS:GO e quem ganhou dado round (que são micro-partidas dentro de uma partida de CS:GO). A ideia, então, é: dado um certo momento do jogo, é possível prever, utilizando Redes Neurais, quem será vitorioso no round? Assim, este trabalho discorre desde o pré-processamento dos dados deste dataset até a criação de um modelo que possui 85% de acurácia na previsão de quem ganhará o round. É válido já ressaltar que os pontos fortes deste trabalho foram insights acerca do dataset específico que permitiram um pré-processamento inteligente dos dados e a utilização de técnicas que foram passadas durante o curso de redes neurais para otimizar a rede final criada no projeto.

2 Introdução

Counter-Strike: Global Offensive é um dos mais populares jogos online de combate com armas de fogo. Em Counter-Strike, a cada nova partida, dez jogadores são distribuídos em duas equipes com cinco jogadores cada. As partidas são divididas em rodadas, e acabam assim que uma das equipes acumular 16 rodadas vencidas. Para vencer uma rodada, a equipe Terrorista (T) precisa armar e explodir uma bomba, ou eliminar todos os jogadores da equipe Contra-Terrorista (CT). Já os contra-terroristas precisam desarmar a bomba ou eliminar todos os jogadores terroristas para vencer a rodada. O objetivo deste trabalho é treinar e avaliar a capacidade de diferentes modelos de aprendizado de máquina de prever a equipe que irá ganhar uma rodada. Para isso, os modelos são treinados com informações de momentos em várias rodadas, bem como a equipe vencedora dessas rodadas. Por exemplo, sabendo que num momento uma equipe possui um número muito superior de jogadores vivos em relação à outra, espera-se que os modelos prevejam que essa será a equipe vencedora dessa rodada.

2.1 Descrição do problema

Nosso trabalho trata de um problema de classificação binária. Os registros são momentos em diferentes rodadas e os atributos são informações sobre o estado de cada equipe naquele momento — p. ex., jogadores restantes, dinheiro disponível para os jogadores comprarem armas e granadas, vida restante dos jogadores etc. A variável-alvo é a equipe que venceu a rodada, e pode assumir dois valores: CT ou T.

Figure 1: Imagem do jogo Counter-Strike: Global Offensive.

3 Dataset e tecnologia

O *Dataset* utilizado está disponível no site Kaggle[1]. Esse *Dataset* possui 97 colunas e 122410 registros, porém nem todas as colunas foram utilizadas para o treino dos modelos. A Tabela 1 apresenta os principais atributos.

time_left	Tempo restante até o fim da rodada[2]
map	Mapa em que a rodada está sendo jogada
ct_score	Rodadas vencidas pelos CT
t_score	Rodadas vencidas pelos T
bomb_planted	Se a bomba já foi armada
ct_health	Vida restante dos jogadores CT
t_health	Vida restante dos jogadores T
ct_money	Dinheiro disponível para os CT comprarem armas e granadas
t_money	Dinheiro disponível para os T comprarem armas e granadas
ct_helmets	Quanto jogadores CT possuem capacete à prova de balas
t_helmets	Quanto jogadores T possuem capacete à prova de balas
ct_weapon_X	Contador da arma X para os CT
t_weapon_X	Contador da arma X para os T

Table 1: Principais atributos.

O Google Colab[3] foi utilizado como ambiente de desenvolvimento, no qual as biliotecas python do sckit-learn[4] foram utilizadas para parte do pré-processamento dos dados e a criação das redes neurais foi feita a partir da biblioteca Keras[5]. O código usado para esse trabalho pode ser acessado através do Google Colab ou do Github. O *Dataset* é importado para o código através da biblioteca Pandas[6].

4 Metodologia

4.1 Apresentação da solução do problema proposto

A solução do problema é realiza a partir do uso de uma Rede Neural MLP em um dataset pré-processado para prever quem é o vencedor em um dado momento de um round de CS:GO.

4.1.1 Métricas de avaliação

Este trabalho lida com um problema de classificação binária para um *dataset* balanceado em relação ao número de registros para cada classe. Portanto, a principal métrica de avaliação observada é a acurácia dos modelos. A acurácia neste trabalho é definida como a porcentagem de classificações corretas sobre o total de classificações. Esse valor é apresentado junto a uma matriz de confusão, que contém a porcentagem de acerto específica para cada uma das duas classes.

4.2 Descrição teórica (matemática) do modelo utilizado

Uma Rede Neural é composta por nós (neurônios) independentes e conectados que se comunicam em cascata. A Figura 2 representa bem a estrutura básica de uma rede neural. Nela existe a camada de entrada (à esquerda), as camadas intermediárias, e a camada final (à direita).

[1] https://www.kaggle.com/christianlillelund/csgo-round-winner-classification
[2] Se o tempo acabar, a equipe CT vence a rodada, a equipe T precisa vencer antes que o tempo acabe
[3] https://colab.research.google.com/
[4] https://scikit-learn.org/
[5] https://keras.io/
[6] https://pandas.pydata.org/

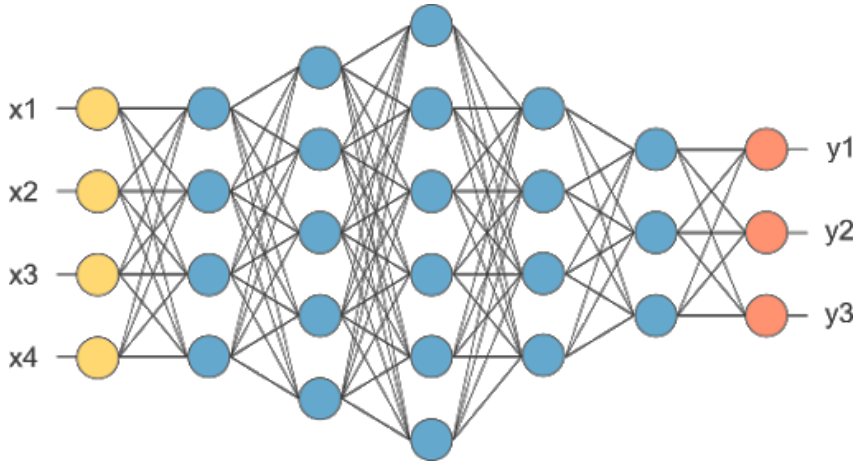

Figure 2: Rede Neural.

A estrutura de um neurônio é apresentada na Figura 3.

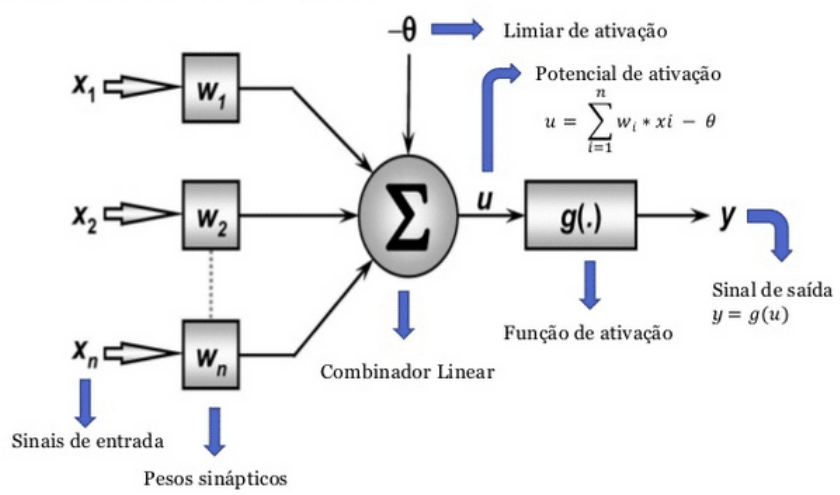

Figure 3: Modelo básico de um Perceptron.

Cada neurônio calcula uma função linear em cima do vetor de entrada. O resultado dessa função linear é usado como entrada de uma função de ativação de natureza não-linear (p.ex. função sigmoide).

Para um neurônio da camada k, sabendo que n é a quantidade de neurônios da camada anterior, e que w_{ij} são os pesos do neurônio, o resultado do neurônio antes da função de ativação é:

$$p_j^k = b_j^k + \sum_{i=1}^{n} w_{ij} h_i^{k-1} \tag{1}$$

O resultado final do neurônio após a função de ativação g é dado por:

$$f_j^k = g(p_j^k) \tag{2}$$

Para encontrar os pesos ideais que minimizem a função custo do modelo, é aplicado o método do gradiente descendente (ou outro similar). Este método visa otimizar a função matemática gerada pela junção dos neurônios e funções de ativação. Para isso, o método do gradiente descendente calcula os

pesos de forma iterativa de modo a modificar a função custo na direção contrária ao vetor gradiente da função.

5 Análise dos Dados

5.1 Visualização e Caracterização dos dados

5.1.1 Verificação de balanceamento

O primeiro aspecto do conjunto de dados analisado é o balanceamento de registros. Idealmente, em um problema de classificação, os registros devem ser distribuídos de forma uniforme entre as classes do problema. Caso contrário, métodos de tratamento de dados precisam ser utilizados para evitar que os modelos realizem o aprendizado de forma a melhorar desproporcionalmente o desempenho das classes mais recorrentes. Neste trabalho, verificamos que o *dataset* já estava suficientemente balanceado. Em específico, 49% dos registros são de vitórias dos CT enquanto 51% são de vitórias dos T.

5.1.2 Tipos de variáveis

O *dataset* deste trabalho possui variáveis dos seguintes tipos: variáveis binárias, categóricas e numéricas discretas (inteiros). Apenas duas variáveis são binárias: 'round_winner' que é a nossa variável alvo, e indica a equipe vencedora, e 'bomb_planted', que indica se a bomba já foi armada pelos T. A variável 'map', que indica o nome do mapa (cenário) da partida é a única variável categórica. As variáveis binárias são transformadas em zero ou um, processo conhecido como binarização. Já a variável categórica passa pelo processo de one-hot encoding, no qual a variável é dividida em várias outras binárias, cada uma indicando se o registro possui aquela categoria da variável original.

5.1.3 Tratamento de variáveis menos relevantes

O *dataset* original possui seis variáveis constantes. Essas variáveis, no geral, indicam a quantidade de jogadores utilizando um modelo de arma especifico para cada equipe. Isso ocorre porque as partidas desse *dataset* são de campeonatos, nos quais os jogadores raramente compram certas armas por serem ineficazes. Além dessas seis variáveis, outras variáveis de armas e granadas quase constantes foram agrupadas. Para realizar esse processo, foram selecionadas variáveis de armas e granadas que possuíam o mesmo valor em pelo menos 90% dos registros e agrupadas em novas variáveis para os Terrorista e Contra-Terrorista, respectivamente. Portanto, após o agrupamento e remoção destes dados, restaram apenas 52 atributos no *dataset* (incluindo a variável alvo).

5.2 Histograma das variáveis

A Figura 4 apresenta os histogramas gerados para cada variável do *dataset*. É possível observar que a maioria das variáveis apresenta distribuição assimétrica e que existe uma variação muito grande entre as ordens de grandeza das variáveis. Enquanto 't_money' varia aproximadamente entre 0 e 70000, 't_helmets' varia apenas entre 0 e 5. Este é um exemplo de discrepância entre variáveis. Para evitar que os modelos privilegiem variáveis de maior escala, foi utilizado o método MinMax Scaler, definido na seguinte equação, onde x é a variável e i o registro.

$$x_i = \frac{x_i - min(x)}{max(x) - min(x)} \tag{3}$$

O MinMax Scaler mapeia as variáveis para o intervalo $[0, 1]$, resolvendo o problema de diferentes escalas. A Figura 5 apresenta a distribuição das variáveis após a transformação. Podemos ver na Figura que as variáveis passam a ter distribuição mais próxima da distribuição normal, o que pode ajudar o aprendizado de alguns modelos.

Figure 4: Histogramas de todas as variáveis restantes.

5.2.1 Matriz de correlação

A matriz de correlação do *dataset*, gerada após os tratamentos, é apresentada na Figura 6. Uma entrada (i, j) da matriz indica o coeficiente de Pearson entre os índices i e j, sendo que os índices da matriz são as variáveis do *dataset*. O coeficiente de Pearson, por sua vez, indica, de forma aproximada, se existe uma relação linear entre esses dois atributos. É possível observar na Figura essas relações entre alguns atributos. Por exemplo, 'ct_players_alive' está positivamente relacionado com 'ct_health'. Essa relação entre vida total e quantidade de jogadores vivos é facilmente compreendida, assim como outras relações presentes no *dataset*. Embora, existam essas dependências lineares entre variáveis, optou-se por não aplicar o método de Análise de Componentes Principais (ACP).

5.2.2 Análise de *outliers*

A última etapa de análise do *dataset* realizada é a remoção de *outliers*. *Outliers* são pontos discrepantes, ou seja, pontos com características muito diferentes do restante dos dados. Esses *Outliers* podem influenciar negativamente o aprendizado dos modelos. Isso ocorre, sobretudo, devido ao efeito de *overfitting*. No *overfitting*, o modelo aprende a classificar os *outliers*, e como esses *outliers* não representam bem a distribuição real dos dados do problema, o modelo acaba tendo seu desempenho afetado quando usado

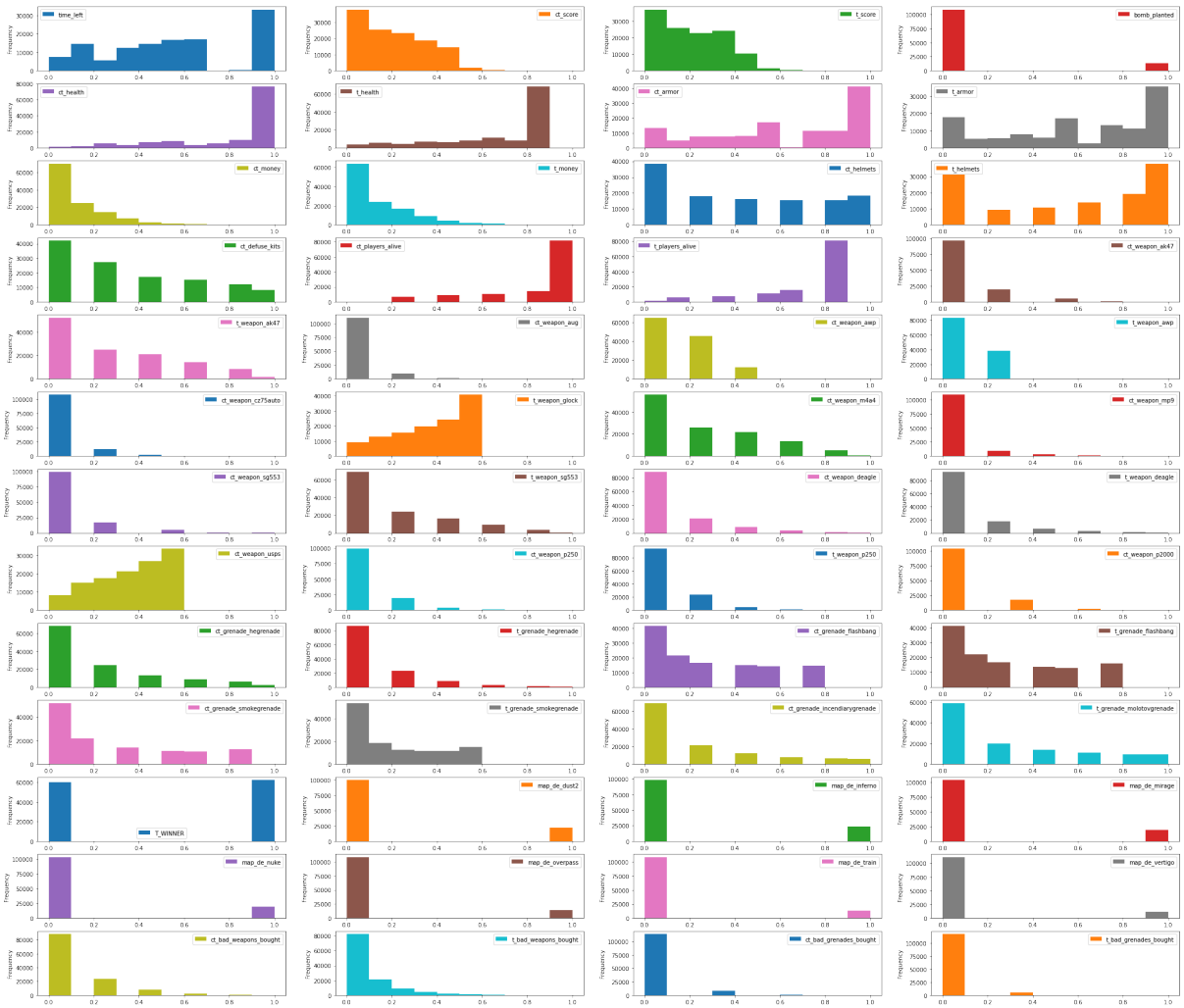

Figure 5: Histogramas das variáveis após MinMax Scaler.

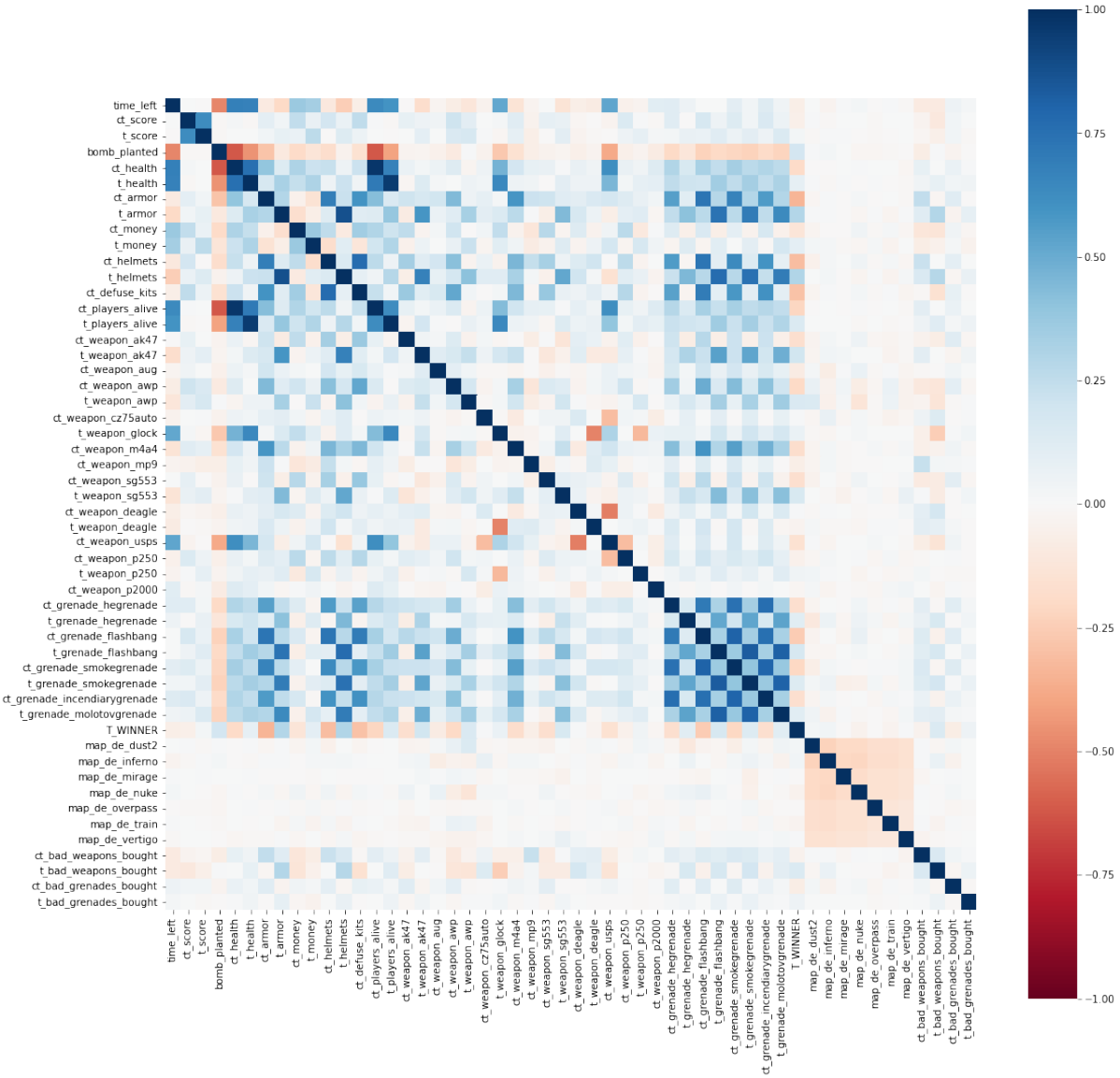

Figure 6: Matriz de correlação.

para classificar registros fora do conjunto de treino. Para identificar os *outliers*, calculou-se a distância média de cada registro aos outros. Essas distâncias são então ordenadas e usadas para comparar os registros que possuem distância muito grande comparada aos outros registros, e, portanto, podem ser considerados *outliers*.

Outliers ocasionalmente aparecem devido a uma má coleta ou tratamento dos dados. Porém, os registros do *dataset* selecionado é confiável pois apenas representam o estado atual do jogo, e por isso não foi feito nenhum tipo de tratamento para remoção de *outliers*.

6 Treinamento e escolha da Rede Neural

6.1 Descrição do procedimento de treinamento e validação

Inicialmente, utilizamos a técnica de validação cruzada para encontrar os melhores parâmetros da rede. Na validação cruzada, o *dataset* é dividido em dez partes, nas quais os registros são distribuídos de forma aleatória. Os modelos, então, são treinados dez vezes, alternando o conjunto de teste entre as dez divisões. O conjunto de treino é composto dos registros restantes, que não são utilizados no teste. Por fim, obtém-se dez métricas de avaliação, que são usadas para calcular uma média e desvio padrão. Após isso, utilizamos a técnica de *treino-teste-validação* para testar ainda mais e estressar os parâmetros descobertos anteriormente para gerar os modelos finais e validá-los aproveitando o máximo de dados para treinamento e ainda assim conseguir testá-los de forma eficiente. Foram utilizados 70% dos dados para o treino e 30% para o teste e validação divididos igualmente.

7 Resultados

Após testar diversas topologias, as melhores topologias encontradas foram duas: MLP com 4 camadas densas com 200 neuronios cada e MLP com 2 camadas de 200 densas - 1 camada dropout (10%) - 2 camadas de 200 densas. A ideia de utilizar dropout veio do fato da utilização deste número grande neurônio de forma a diminuir o overfitting. A primeira, que não utiliza dropout, teve resultados levemente inferiores a rede utilizando dropout nesse sentido. É válido ressaltar que ao tentarmos incluir mais dropouts na rede, esta perdeu muito a capacidade de generalização, assim, chegamos nesse ponto bom com apenas uma camada dropout no meio das camadas intermediárias - assim, melhorando a performance dos modelos nas etapas de teste e validação.

Modelo	Acurácia treino	Acurácia teste	Acurácia validação
(200,200,200,200) sem dropout	96.62%	83.65%	83.98%
(200,200,200,200) com dropout	95.27%	85.08%	84.49%

Table 2: Resultados dos principais modelos de Redes Neurais.

8 Conclusões

Este trabalho realiza estudos de diversas arquiteturas de redes neurais aplicadas a um problema de classificação binária. Antes de treinar os modelos, são realizados diversos pré-processamentos, como ajuste da escala dos registros, variáveis constantes etc.

Com o modelo funcional e bem ajustado, futuramente será possível aproveitar o modelo para criar um sistemas que realize a predição do vencedor da partida em tempo real e utilizar tal sistema em campeonatos ou até mesmo pelas próprias equipes como forma de estudar e compreender as variáveis mais importantes na disputa de uma partida.

Para nós ficou claro durante o processo de desenvolvimento do projeto como os detalhes e o pré-processamento são passos importantes para se obter uma rede interessante. Inicialmente, havíamos

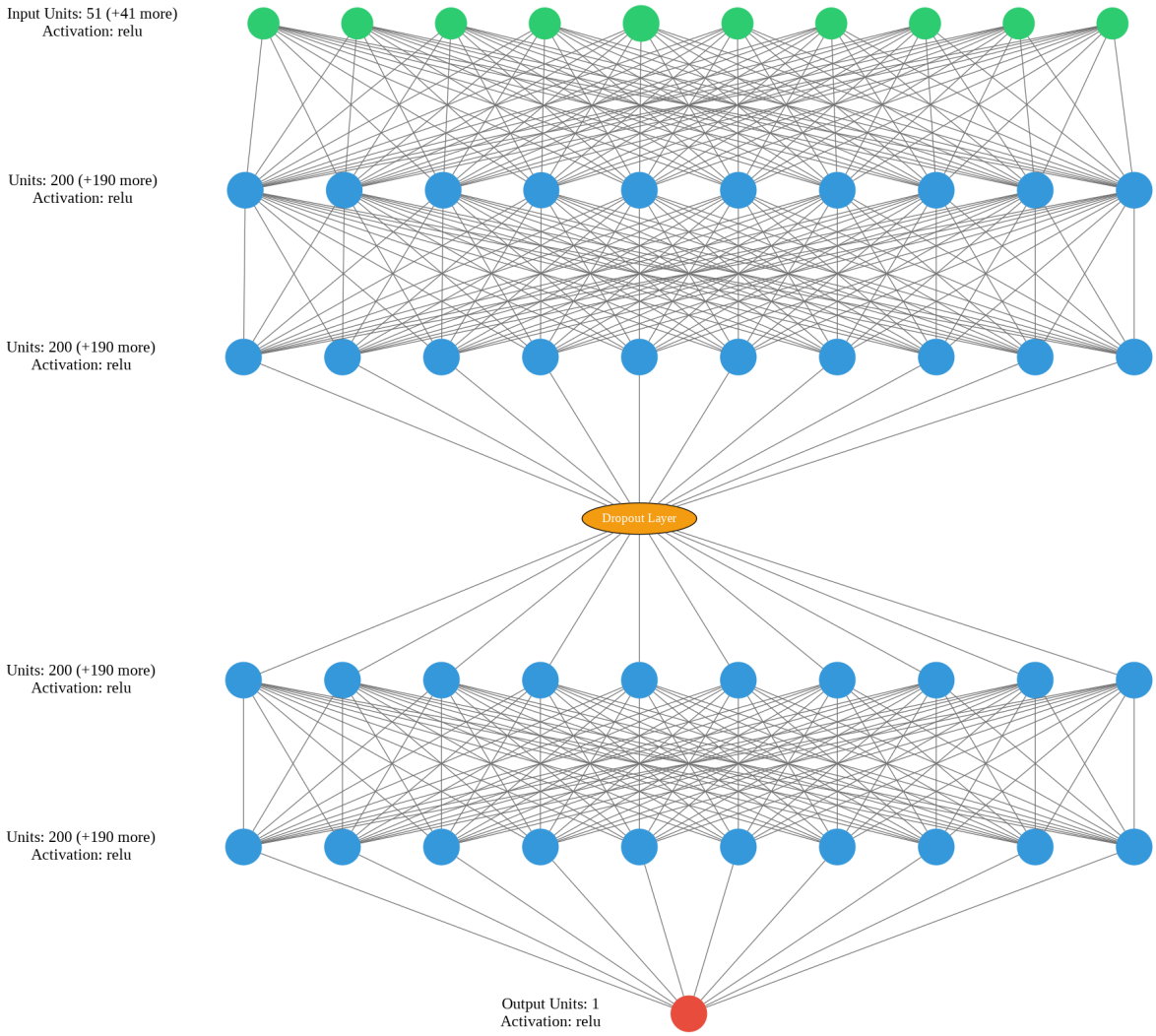

Figure 7: Representação gráfica da melhor rede encontrada

treinado a rede realizando pouco pré-processamento - obtivemos resultados pífios nesse momento, nossos melhores resultados não passaram da casa dos 75% de acurácia - o que foi um balde de água fria.

Assim, a ideia de agregar colunas para não perdermos a informação e ao mesmo tempo facilitar o trabalho de treinamento da rede provou-se uma grande estratégia, de forma esse foi um grande diferencial que nos permitiu obter estes ótimos resultados.

Outro ponto interessante foi a utilização da técnica de dropout no dataset. Embora não tenhamos conseguido nos aproveitar tanto assim desta técnica, conseguimos ao menos diminuir de forma relevante a distância entre os resultados do treinamento do teste e da validação, demonstrando o quanto essa técnica, mesmo com utilização singela, é capaz de diminuir o overfitting.

9 Referências

https://www.kaggle.com/datasets/christianlillelund/csgo-round-winner-classification
https://en.wikipedia.org/wiki/Counter-Strike:_Global_Offensive
https://en.wikipedia.org/wiki/Perceptron
https://www.tensorflow.org/api_docs/python/tf/keras/layers/Dropout