Comparação Experimental de Classificadores para Predição de Resultados em League of Legends aos 15 Minutos de Jogo

Rubem Valadares de Almeida

Programa de Pós-Graduação em Informática, Universidade Federal do Espírito Santo (UFES), Vitória, ES, Brasil

Abstract

Este trabalho apresenta uma comparação experimental de cinco algoritmos de aprendizado de máquina para a tarefa de predição de vitória em partidas do jogo League of Legends (LoL). O diferencial deste estudo reside na utilização de um conjunto de dados restrito, contendo apenas informações coletadas aos 15 minutos de partida, simulando um cenário de predição em tempo real com informações parciais. Os classificadores avaliados foram: Árvore de Decisão (DT), K-Nearest Neighbors (KNN), Multi-Layer Perceptron (MLP), Random Forest (RF) e um ensemble customizado denominado Heterogeneous Boosting (HB). O procedimento experimental foi conduzido por meio de validação cruzada aninhada, com 3 repetições de 10 *folds* externos para teste e 4 *folds* internos para otimização de hiperparâmetros via Grid Search. Os resultados indicam que o MLP obteve a maior acurácia média (75.31%), seguido de perto pelo HB (75.16%) e RF (74.66%). Testes de hipótese (t de Nadeau e Bengio e Wilcoxon) confirmaram que, embora o MLP, RF e HB tenham apresentado desempenho estatisticamente superior ao DT e KNN, não houve diferença estatística significativa entre eles, sugerindo que os três são igualmente eficazes para esta tarefa de predição precoce. O código-fonte completo e os dados utilizados estão disponíveis publicamente para reprodução dos experimentos.

Keywords: Aprendizado de Máquina, League of Legends, Reconhecimento de Padrões, Classificação, Sistemas Inteligentes, Predição Esportiva

1. Introdução

Os esportes eletrônicos (e-sports) representam um fenômeno global, movimentando uma indústria bilionária e atraindo milhões de espectadores. Dentre os jogos mais populares, League of Legends (LoL), um Multiplayer Online Battle Arena (MOBA), destaca-se pela sua complexidade estratégica e competitiva. A capacidade de prever o resultado de uma partida com base em seu estado inicial é um problema de grande interesse, tanto para analistas e equipes

Email address: rubem.almeida@aluno.ufes.br (Rubem Valadares de Almeida)

quanto para o mercado de apostas.

Este trabalho foca na predição do time vencedor de uma partida de LoL utilizando um subconjunto restrito de dados, coletados especificamente aos 15 minutos de jogo. Esta abordagem simula um cenário prático onde decisões precisam ser tomadas com informações parciais, muito antes do desfecho da partida.

O objetivo principal é realizar uma comparação experimental rigorosa entre cinco técnicas de classificação: Árvore de Decisão (DT), K-Nearest Neighbors (KNN), Multi-Layer Perceptron (MLP), Random Forest (RF) [4] e um método de ensemble proposto, o Heterogeneous Boosting (HB). Através de um procedimento de validação cruzada aninhada e testes de hipótese, buscamos identificar não apenas o modelo com melhor desempenho médio, mas também avaliar se a superioridade de um método sobre os outros é estatisticamente significativa.

2. Base de Dados

A base de dados utilizada neste estudo é fundamental para a análise e o treinamento dos modelos de aprendizado de máquina. A seguir, detalhamos suas principais características.

2.1. Descrição do Domínio

O estudo é baseado em dados de partidas de League of Legends (LoL), um dos e-sports mais populares do mundo. Em LoL, duas equipes de cinco jogadores competem com o objetivo de destruir a base inimiga. O desempenho das equipes pode ser medido através de diversas métricas, como ouro acumulado, experiência (XP), número de abates, entre outras. A predição de resultados com base em dados coletados nos estágios iniciais da partida é um desafio relevante, pois pode indicar tendências de vitória antes que uma vantagem se torne irreversível.

2.2. Definição das Classes e das Características

O problema é modelado como uma tarefa de classificação binária. A classe alvo, 'result', indica o vencedor da partida, onde '1' representa a vitória da equipe azul e '0' a vitória da equipe vermelha.

Para a Tarefa IV, foram utilizadas exclusivamente 5 características (*features*) que representam a diferença de desempenho entre as duas equipes aos 15 minutos de jogo:

- **golddiffat15**: Diferença de ouro acumulado.
- **xpdiffat15**: Diferença de pontos de experiência.
- **csdiffat15**: Diferença na contagem de minions abatidos (*creep score*).

- **killsdiffat15**: Diferença no número de abates.
- **assistsdiffat15**: Diferença no número de assistências.

Todas as características são numéricas e foram padronizadas (média 0 e desvio padrão 1) antes do treinamento dos modelos para evitar viés de escala.

2.3. Número de Instâncias

O conjunto de dados completo contém 8152 instâncias, onde cada instância corresponde a uma partida única da temporada de 2021.

2.4. Distribuição de Classes

A base de dados apresenta um leve desbalanceamento entre as classes:

- **Classe 1 (Vitória do time azul)**: 4336 instâncias (53.19%).
- **Classe 0 (Vitória do time vermelho)**: 3816 instâncias (46.81%).

Este desbalanceamento foi considerado no procedimento experimental através do uso da validação cruzada estratificada, garantindo que a proporção das classes fosse mantida em cada *fold* de treino e teste.

3. O Método Heterogeneous Boosting

Para esta análise comparativa, foi implementado um método de *ensemble* customizado, o Heterogeneous Boosting (HB). O HB foi projetado para combinar as predições de múltiplos classificadores base heterogêneos, buscando um modelo mais robusto e generalista.

O HB é composto por quatro algoritmos de aprendizado de máquina distintos, implementados a partir da biblioteca Scikit-learn [1] e utilizados com seus parâmetros padrão:

- Árvore de Decisão (DecisionTreeClassifier)
- Naive Bayes Gaussiano (GaussianNB)
- Multi-Layer Perceptron (MLPClassifier)
- K-Nearest Neighbors (KNeighborsClassifier)

O processo de predição do HB se baseia em uma votação majoritária simples. Para uma dada instância, cada um dos N estimadores do *ensemble* (onde N é um hiperparâmetro) faz uma predição. A classe final é aquela que recebe o maior número de votos. Em caso de

empate, o critério de desempate é a classe majoritária observada no conjunto de treinamento, uma estratégia simples para resolver ambiguidades sem adicionar complexidade ao modelo. O pseudocódigo do método é apresentado abaixo.

```
procedure HBEnsemble(X_train, y_train, X_test, n_estimators)
    // Treinamento
    base_learners = [DT, NB, MLP, KNN]
    n_base_learners = length(base_learners)
    trained_models = []

    for i from 1 to n_estimators:
        learner = base_learners[i % n_base_learners]
        model = train(learner, X_train, y_train)
        add model to trained_models

    // Predição
    all_predictions = []
    for model in trained_models:
        predictions = predict(model, X_test)
        add predictions to all_predictions

    final_predictions = []
    for each sample_index:
        votes = get_votes_for_sample(all_predictions, sample_index)
        if tie_in_votes(votes):
            predict majority_class_from_y_train
        else:
            predict most_voted_class

    return final_predictions
end procedure
```

4. Descrição dos Experimentos Realizados e seus Resultados

O procedimento experimental foi desenhado para garantir uma avaliação robusta e imparcial dos classificadores, utilizando implementações da biblioteca Scikit-learn [1] e validação cruzada aninhada para otimização de hiperparâmetros e teste.

A avaliação foi realizada através de 3 repetições de validação cruzada estratificada de 10 *folds* (ciclo externo). Para cada *fold* do ciclo externo, uma busca em grade (Grid Search) com validação cruzada de 4 *folds* (ciclo interno) foi utilizada para encontrar os melhores hiperparâmetros para cada modelo. Os hiperparâmetros testados foram:

- **DT**: `criterion` (`gini`, `entropy`), `max_depth` (5, 10, 15, 25).
- **KNN**: `n_neighbors` (1, 3, 5, 7, 9).
- **MLP**: `hidden_layer_sizes` ((100,), (10,)), `alpha` (0.0001, 0.005), `learning_rate` (`constant`, `adaptive`).
- **RF**: `n_estimators` (5, 10, 15, 25), `max_depth` (10, None).
- **HB**: `n_estimators` (5, 10, 15, 25, 50).

Para garantir a reprodutibilidade, as sementes aleatórias (`random_state`) foram fixadas em `36854321` para a validação cruzada e `13` para os classificadores com componentes estocásticos.

4.1. Resultados dos Classificadores

A Tabela 1 resume o desempenho dos cinco classificadores em termos da acurácia média, desvio padrão e o intervalo de confiança de 95% ao longo das 30 rodadas de teste.

Tabela 1: Resultados de acurácia dos classificadores.

Método	Média	Desvio Padrão	Lim. Inferior (IC95%)	Lim. Superior (IC95%)
DT	0.7500	0.0146	0.7448	0.7552
KNN	0.7362	0.0141	0.7311	0.7412
MLP	0.7531	0.0158	0.7474	0.7587
RF	0.7466	0.0153	0.7411	0.7521
HB	0.7516	0.0168	0.7456	0.7576

O MLP apresentou a maior acurácia média, superando ligeiramente os métodos de *ensemble* HB e RF. O KNN obteve o desempenho mais baixo entre os modelos avaliados. A Figura 1 ilustra a distribuição das acurácias para cada classificador através de um *boxplot*.

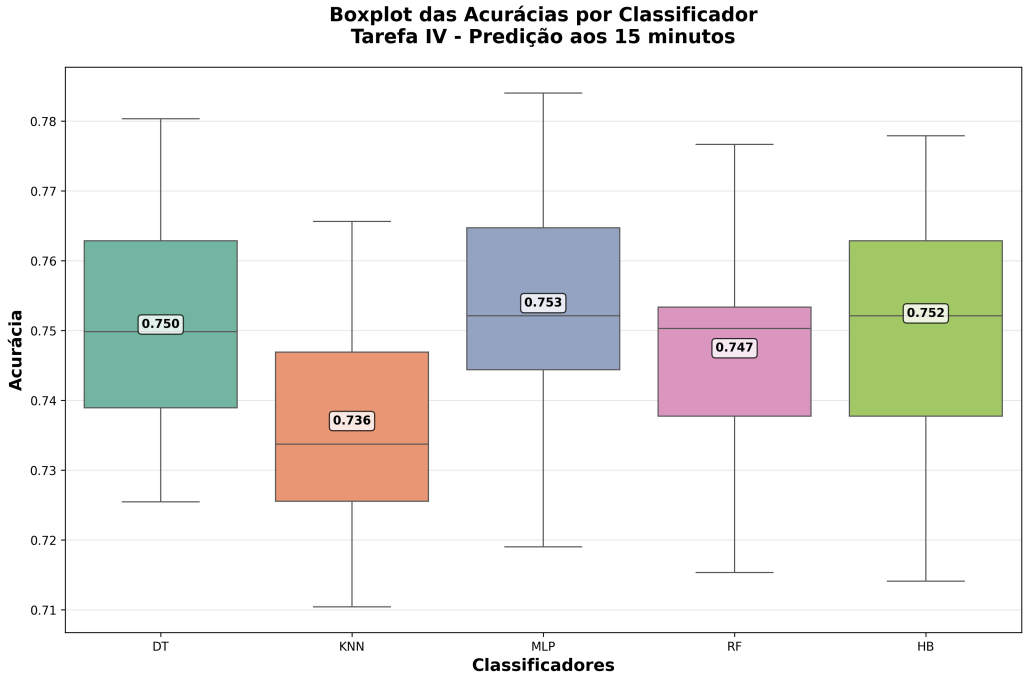

Figura 1: *Boxplot* das acurácias dos classificadores nos 30 *folds* de teste.

4.2. Análise Estatística

Para determinar se as diferenças de desempenho observadas são estatisticamente significativas, foram realizados testes de hipótese pareados: o teste t corrigido de Nadeau e Bengio [2] e o teste de Wilcoxon [3]. A Tabela 2 apresenta os p-valores obtidos, com valores que rejeitam a hipótese nula ($p < 0.05$) em negrito.

Tabela 2: Tabela de p-valores. Triângulo superior: teste t corrigido. Triângulo inferior: teste de Wilcoxon.

	DT	**KNN**	**MLP**	**RF**	**HB**
DT		**0.0000**	0.3014	0.1693	0.6128
KNN	**0.0000**		**0.0000**	**0.0000**	**0.0000**
MLP	**0.0381**	**0.0000**		**0.0016**	0.5898
RF	**0.0002**	**0.0000**	0.0617		**0.0076**
HB	0.2882	**0.0000**	0.3204	**0.0011**	

A análise estatística revela que MLP, RF e HB foram significativamente superiores a DT e KNN. No entanto, a hipótese nula de que não há diferença de desempenho entre MLP, RF e HB não pôde ser rejeitada pela maioria dos testes, indicando que, do ponto de vista estatístico, seus desempenhos são equivalentes para esta tarefa.

5. Conclusões

Este trabalho realizou uma análise experimental detalhada para a predição de resultados em partidas de League of Legends, utilizando apenas dados dos primeiros 15 minutos de jogo.

5.1. Análise geral dos resultados

Os resultados mostraram que é possível atingir uma acurácia de aproximadamente 75% utilizando apenas um subconjunto limitado de *features* do início do jogo, o que demonstra o alto poder preditivo dessas variáveis. O modelo MLP alcançou a maior acurácia média, embora os testes estatísticos não tenham apontado uma superioridade significativa em relação aos *ensembles* RF e HB. Isso sugere que, para esta tarefa específica, os três modelos são escolhas competitivas. Os modelos mais simples, DT e KNN, apresentaram um desempenho estatisticamente inferior, indicando que a complexidade e a capacidade de generalização dos modelos mais avançados são vantajosas.

5.2. Contribuições do Trabalho

A principal contribuição deste estudo é a avaliação rigorosa e comparativa de diferentes abordagens de classificação em um cenário desafiador e prático de predição precoce. Além disso, a implementação e avaliação do *ensemble* Heterogeneous Boosting (HB) demonstrou que a combinação de modelos heterogêneos pode alcançar um desempenho competitivo, equiparável a métodos consagrados como Random Forest [4].

5.3. Melhorias e trabalhos futuros

Como trabalhos futuros, sugere-se a exploração de um conjunto mais amplo de características, incluindo dados específicos de cada jogador ou eventos do mapa. A utilização de técnicas de *deep learning*, como redes neurais recorrentes (RNNs) para analisar a sequência de eventos do jogo, também pode oferecer melhorias de desempenho. Por fim, a aplicação destes modelos em um sistema de predição em tempo real seria um passo natural para validar sua eficácia em um ambiente de produção.

Disponibilidade de Código e Dados

Para garantir a reprodutibilidade dos experimentos e facilitar futuras pesquisas, todo o código-fonte utilizado neste trabalho, incluindo a implementação dos algoritmos, scripts de experimentação e análise estatística, está disponível publicamente no repositório GitHub: `https://github.com/rubemalmeida/ml-lol-match-prediction.git`

O repositório contém:

- Implementação completa do ensemble Heterogeneous Boosting (HB)
- Scripts para validação cruzada aninhada e otimização de hiperparâmetros
- Código para geração de tabelas e gráficos de resultados
- Testes estatísticos (t de Nadeau e Bengio e Wilcoxon)
- Jupyter Notebook com análise exploratória completa
- Documentação detalhada para reprodução dos experimentos
- Dataset utilizado (`jogosLoL2021.csv`) com 8152 partidas da temporada 2021

Referências Bibliográficas

Referências

[1] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: Machine Learning in Python, Journal of Machine Learning Research, 12 (2011) 2825-2830.

[2] C. Nadeau, Y. Bengio, Inference for the Generalization Error, Machine Learning, 52(3) (2003) 239-281.

[3] F. Wilcoxon, Individual comparisons by ranking methods, Biometrics bulletin, 1(6) (1945) 80-83.

[4] L. Breiman, Random Forests, Machine Learning, 45(1) (2001) 5-32.