Predição de vitória no jogo eletrônico League of Legends baseado nas estatísticas da partida aos 10 minutos

Felipe F. R. Melo, Thales M. Leijoto

Ciência da Computação – Universidade Federal do São João Del-Rei (UFSJ)

felipefrmelo@hotmail.com, thalesmradl@gmail.com

1. Introdução

League of Legends (LoL) é um jogo multiplayer online de batalha em arena (MOBA) desenvolvido pela Riot Games em 2009. Em uma partida comum, cinco jogadores lutam contra outro personagens controlados por outro grupo, chamados campeões em um mapa com três rotas e uma selva. Existem muitos objetivos que devem ser destruídos durante o jogo, como torres, barões, dragões, que dá buffs aos campeões e permite que o principal objetivo, denominado nexus, seja destruído.

Em partidas de LoL, principalmente as denominadas ranqueadas, cada equipe tem condições de vitória que dependem dos campeões escolhidos, vantagem de ouro, ou destruição de objetivos. Um exemplo de condição de vitória é quando uma equipe possui vantagem de ouro, portanto seus campeões tem mais itens e possui mais poder que a adversária, aproveitando isso em lutas e aumentando a vantagem.

Neste trabalho, usamos informações sobre o que aconteceu nos primeiros 10 minutos de uma partida e tentamos prever o resultado de um jogo, que costuma ter duração média de 30 minutos, usando um modelo de classificação, uma classe de técnicas de aprendizado de máquina supervisionado.

Diante do exposto, os principais questionamentos que este trabalho busca responder são:

- Quais os objetivos mais importantes de serem cumpridos em uma partida até os seus 10 minutos, para que o time se sagre vencedor?
- É possível prever o time vencedor da partida apenas com base nas estatísticas de jogo apresentadas até os 10 minutos?

A principal motivação para a elaboração desse trabalho é pensando no cenário competitivo de League of Legends, onde existem grandes equipes que contam com um time de analistas que estudam a fundo o jogo, a fim de aumentar o conhecimento do jogo e repassar a informação aos jogadores da equipe. Sendo as possíveis informações levantadas neste estudo de grande valia para o entendimento estatístico de quais são os aspectos do jogo que devem ser priorizados na fase inicial da partida (os primeiros 10 minutos). Por ser um jogo extremamente competitivo, onde cada partida reserva suas surpresas, e por atrair

multidões, existem muitos sites de apostas que realizam apostas envolvendo os campeonatos e partidas promovidas pela Riot Games. Sendo este segmento de apostas, outro setor que pode ser beneficiado por estudos deste tipo, principalmente caso exista uma aposta hipotética do time que vai ganhar a partida, e seja possível apostar até a partida atingir 15 minutos de duração.

Este trabalho está organizado da seguinte forma: Seção II descreve a metodologia usada neste trabalho; A seção III mostra os resultados dos experimentos e a Seção IV resume nossas conclusões e discute possíveis trabalhos futuros.

2. Metodologia

Para a manipulação dos dados, aplicação dos algoritmos e visualização, foi usada a linguagem python, e as bibliotecas *pandas*, *matplotlib*, *seaborn* e *pycaret*. Esta última uma linguagem de aprendizado de máquina que simplifica a construção de um *pipeline* de um modelo preditivo.

2.1. Os dados

O conjunto de dados usado neste trabalho está disponível no Kaggle e contém informações sobre aproximadamente 10000 partidas ranqueadas de League of Legends, todas elas compostas por jogadores de nível elevado dentro do jogo (Diamante I adiante). Contém 19 atributos de cada time dos primeiros 10 minutos de uma partida, incluindo: vantagem de ouro, torres destruídas, campeões abatidos, número de assistências, monstros épicos abatidos, etc.

2.2. Pré-processamento dos dados

A etapa de pré-processamento dos dados foi bem simples. Não há dados faltantes na base, sendo necessário apenas a remoção de atributos redundantes, estes detectados através da observação de colinearidade perfeita entre atributos na matriz de correlação de Pearson. A matriz de correlação mede a correlação linear entre dois atributos. O valor da correlação pode variar entre -1 e 1, sendo que quando os valores são exatamente -1 ou 1, nós temos atributos redundantes. Como por exemplo os atributos de diferença de gold, "*blueGoldDiff*" e "*redGoldDiff*", onde que se "*blueGoldDiff*" é igual a +5000, o atributo "*redGoldDiff*" será igual a -5000, tendo um coeficiente de Pearson de -1. O problema da multicolinearidade é que quando há uma forte relação linear entre X1 e X2 (multicolinearidade) pode ficar muito difícil identificar os efeitos isolados de X1 e X2 sobre Y. Ou seja, a maior parcela da variabilidade de Y é explicada pelo efeito conjunto de X1 e X2. Sendo assim, para todos os grupos de atributos redundantes, foi mantido apenas um atributo, a fim de remover a redundância. Foi também realizada a criação de um dois novos atributos, o "*redKda*" e o "*blueKda*", que é baseado na somatória dos abates e assistências de um time e o resultado dividido pelo número de mortes desse mesmo time. A análise conjunta desses atributos é

comumente usada nos esportes eletrônicos como medida de desempenho e pode contribuir positivamente para a análise.

2.3. Classificação

A classificação é uma categoria do aprendizado de máquina supervisionado na qual o algoritmo tenta aprender as regras que mapeiam as entradas nas saídas corretas. Como essa técnica é muito usada quando a saída é do tipo binária, ela é muito atraente para o nosso problema.

Na classificação binária, um detalhe que deve ser levado em consideração é se a base de dados está balanceada. Uma base de dados desbalanceada pode ser definida pela pequena incidência de uma categoria (classe minoritária) em comparação com a outra (classe majoritária). Na maioria dos casos, isso faz com que tenhamos muitas informações a respeito da categoria mais incidente, e menos da minoritária. A consequência desse desequilíbrio é que o modelo terá uma tendência a dar muitos "alarmes falsos". Ou seja, na prática ele irá responder muito bem às entradas para a classe majoritária, mas terá um desempenho inferior para a minoritária. A Figura 1 apresenta a distribuição das classes dentro da base de dados. Como pode-se observar, as classes estão perfeitamente balanceadas, podendo partir para a construção do modelo, sem se preocupar em implementar técnicas de balanceamento de classes.

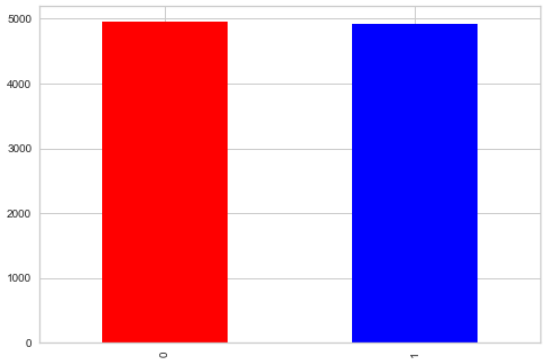

Figura 1. Distribuição de classes alvos da classificação na base de dados.

Para as etapas de treinamento e testes, a base foi dividida em 80% dos dados para treino e 20% para teste. Foi utilizado o método de validação cruzada com 5 folds, isso é, os dados foram divididos em 5 partes de tamanho iguais, com amostras mutuamente exclusivas. Depois, 5 iterações de treinamento foram realizadas, tal que em cada iteração uma diferente parte foi usada para teste e as outras 4 para treinamento.

Para avaliar a qualidade do modelo, foi utilizado a métrica AUC. Esta medida utiliza a curva ROC que é uma curva de probabilidade. Ela é criada traçando a taxa verdadeiro-positivo contra a taxa de falsos-positivos. Ou seja, o número de vezes que o classificador acertou a predição contra o número de vezes que o classificador errou a predição. O AUC representa o grau ou medida de separabilidade. Quanto maior o AUC,

melhor o modelo está em prever 0s como 0s e 1s como 1s. Quanto maior a AUC, melhor o modelo está em distinguir entre vitória do time azul e derrota do time azul (vitória do time vermelho).

No aprendizado de máquina, para capturar indicadores úteis e obter um resultado mais preciso, tende-se a adicionar o máximo de features possível logo de início. Porém, a partir de um certo ponto, o desempenho do modelo pode começar a diminuir com o aumento do número de features. Este fenômeno é frequentemente referido como "The Curse of Dimensionality" (A Maldição da Dimensionalidade). A maldição da dimensionalidade ocorre porque a densidade do conjunto de dados diminui exponencialmente com o aumento da dimensionalidade. Quando continuamos adicionando features sem aumentar o número de amostras do dataset de treinamento, a dimensionalidade do espaço de features aumenta e se torna cada vez mais esparsa. Devido a essa dispersão, torna-se muito mais fácil encontrar uma solução "perfeita" para o modelo de aprendizado de máquina, o que muito provavelmente leva ao problema do overfitting.

Nesse sentido existe a técnica de feature selection. A premissa central ao usar feature selection é que os dados contêm algumas features que são redundantes ou irrelevantes e, portanto, podem ser removidos sem levar a muitas perdas de informações. Dado o grande número de features na base de dados e a elevada quantidade de features que representam conceitos similares, foi realizada uma seleção de features.

Através da função *compare_models()* do *pycaret,* obtivemos as pontuações estimadas das performances dos modelos da biblioteca na nossa base de dados. Foi constatado que o *Logistic Regression* possui o maior valor para a métrica AUC igual a 0.8076. Sendo assim, o modelo de regressão logística foi o escolhido para realizar a classificação.

Realizou-se em seguida a otimização dos parâmetros, que consiste em variar os parâmetros do estimador, buscando encontrar a combinação que resulte no melhor resultado para a métrica desejada. Nesta etapa também é realizada a validação cruzada com 5 folds.

E por fim, de fato, é realizada a predição do modelo já treinado e ajustado nos dados separados para teste.

3. Resultados e Discussões

Neste capítulo serão apresentados os resultados das análises e do modelo de classificação, bem como suas respectivas discussões.

3.1. Análise Descritiva da Base de Dados

Antes de discutir de fato o modelo de classificação, é interessante realizar um estudo preliminar feito em cima dos atributos da base de dados, buscando organizar, resumir, compreender e descrever os aspectos importantes presentes no mesmo.

Primeiramente, foram analisadas as correlações dos atributos em função do atributo ‘winner’ que tem por valores 0 e 1. Sendo 1 em caso de vitória do time azul e 0 em caso de vitória do time vermelho. A Figura 2 apresenta esta matriz de correlação.

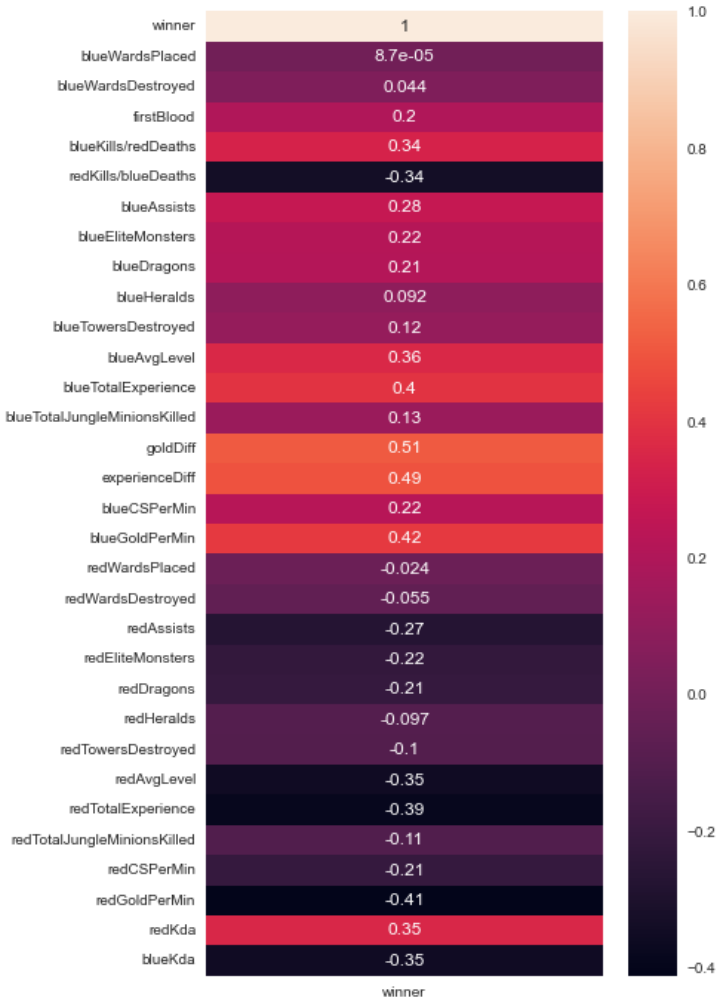

Figura 2. Mapa de correlação dos atributos com o atributo ‘winner’.

À primeira vista, observamos que os atributos que mais se correlacionam linearmente com a vitória ou não-vitória são os atributos de diferença de ouro e diferença de experiência, o que de certa forma é esperado, pois estes dois atributos pode ser considerado como uma generalização de todos os outros. Uma vez que todas as ações dentro do jogo representado pelos atributos deste estudo, geram ouro e/ou experiência à equipe que a realizou, como por exemplo o abate de um dragão, o time que abater o dragão ganha um bônus de ouro e experiência. As Figuras 3 ilustra a relação entre a diferença de ouro entre os times e o número de vitórias, enquanto que a Figura 4 ilustra a relação entre a diferença de experiência entre os times e o número de vitórias. Por fim, a Figura 5 une estes dois atributos em um único gráfico, podendo ver o comportamento de seus valores em função de qual equipe ganhou a partida.

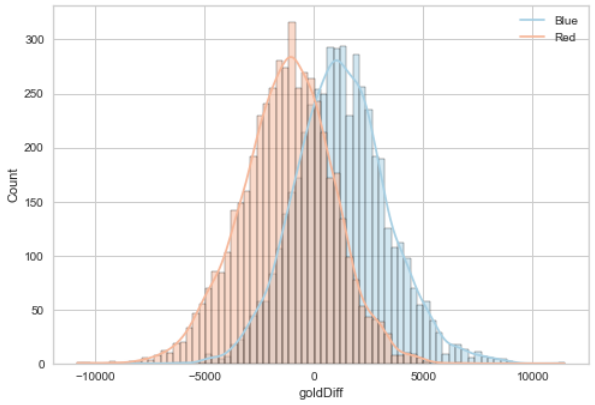

Figura 3. Relação entre a diferença de ouro entre os times e o número de vitórias.

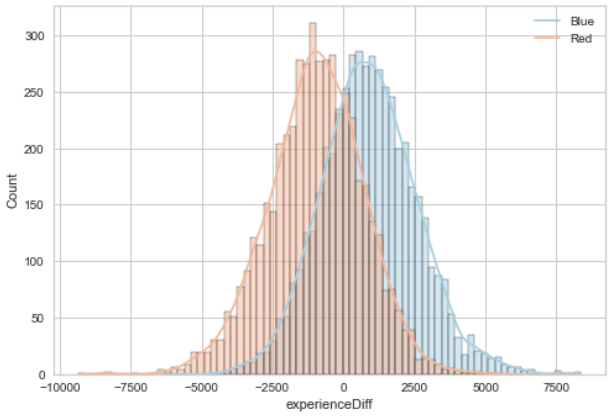

Figura 4. Relação entre a diferença de experiência entre os times e o número de vitórias.

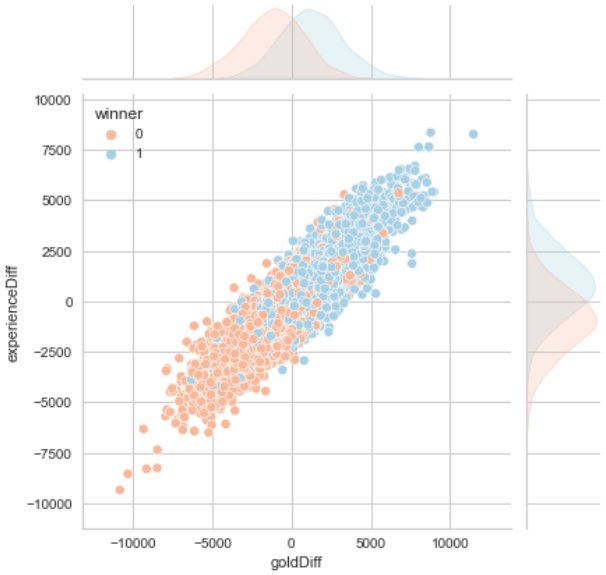

Figura 5. Relação entre a diferença de ouro e a diferença de experiência em função de qual time ganhou a partida.

Pela observação das figuras, notamos a importância desses dois atributos na condição de vitória de uma equipe. Visualmente, pode-se dizer que para ambos os atributos, quando a

diferença ultrapassa os 5000, as chances de vitória da equipe que está com essa diferença são muito altas.

Outro atributo que chama atenção é o "*firstBlood*", que retorna qual equipe fez o primeiro abate da partida, 1 se foi a equipe azul e 0 se foi a equipe vermelha. Do ponto de vista da correlação de Pearson, ele não tem uma correlação linear tão forte com a condição de vitória, tendo um valor de 0,2. Contudo, muitos jogadores relatam que o time que pega o primeiro abate conta uma vantagem mental, uma vez que os integrantes da equipe, principalmente o jogador que sofreu o primeiro abate, pode ficar dessintonizado no jogo, perdendo a concentração e etc. aumentando as chances de derrota. Neste sentido, a Figura 6 apresenta a relação entre a equipe que conquistou o primeiro abate e as condições de vitória. É notável um certo aumento na quantidade de vitórias, para ambas as equipes, quando tal equipe realiza o primeiro abate na partida.

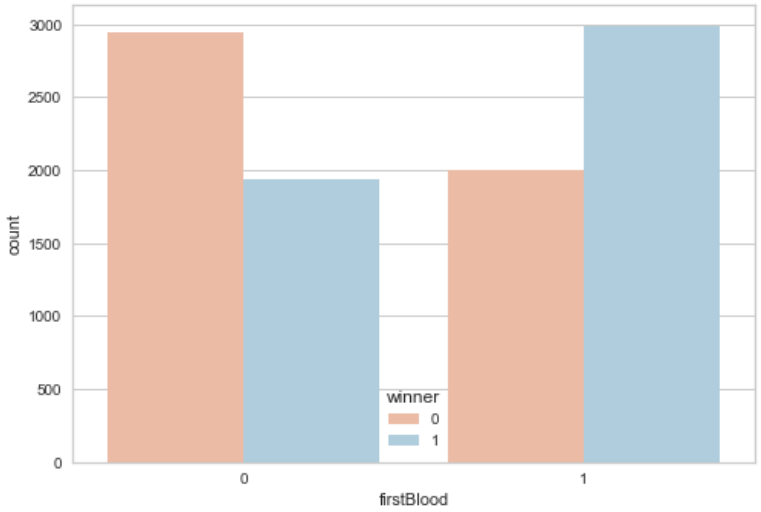

Figura 6. Relação entre a condição de conquistar o primeiro abate com o número de vitórias.

3.2. Classificação

Tendo treinado o modelo, realizado a otimização dos parâmetros e a otimização da quantidade de features, deve-se executar o modelo para os dados separados para teste. A Tabela 1 apresenta o resultado da execução do modelo para os 20% dos dados separados para treino.

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	Logistic Regression	0.7343	0.8153	0.7273	0.7204	0.7238	0.4679	0.4679

A Figura 7 ilustra a matriz de confusão resultado da classificação. A matriz de confusão é uma métrica voltada para modelos de classificação e tem como objetivo calcular a quantidade de falso positivo e falso negativo; e de verdadeiro positivo e verdadeiro negativo, além de te fornecer a acurácia e sensibilidade. Observamos através da matriz de confusão,

que o modelo obteve uma taxa de acerto similar entre as duas classes, demonstrando que o modelo está equilibrado.

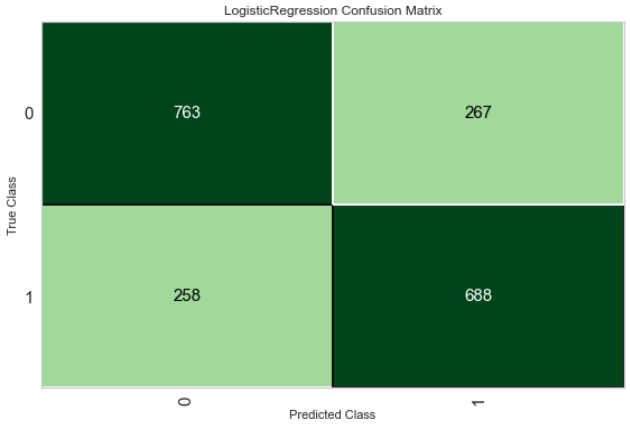

Figura 7. Matriz de confusão da classificação.

A Figura 8 apresenta o relatório da classificação para as métricas de precisão, recall e f1-score. Pela métrica de precisão, que é a razão entre os verdadeiros positivos e os verdadeiros positivos mais os falsos positivos, observa-se valores promissores, como 72% para a classe 0 e 74% para a classe 1.

Figura 8. Relatório de Classificação.

Relacionado aos atributos de entrada do classificador, uma análise interessante de ser feita é a importância de cada atributo na tarefa de classificação. A Figura 9 traz a importância de cada um dos atributos na classificação feita pelo *Logistic Regression*. Com a figura confirmamos algumas observações já feitas anteriormente: os atributos relacionados à ouro e os atributos relacionados à experiência tem um alto grau de importância nas decisões do modelo preditivo.

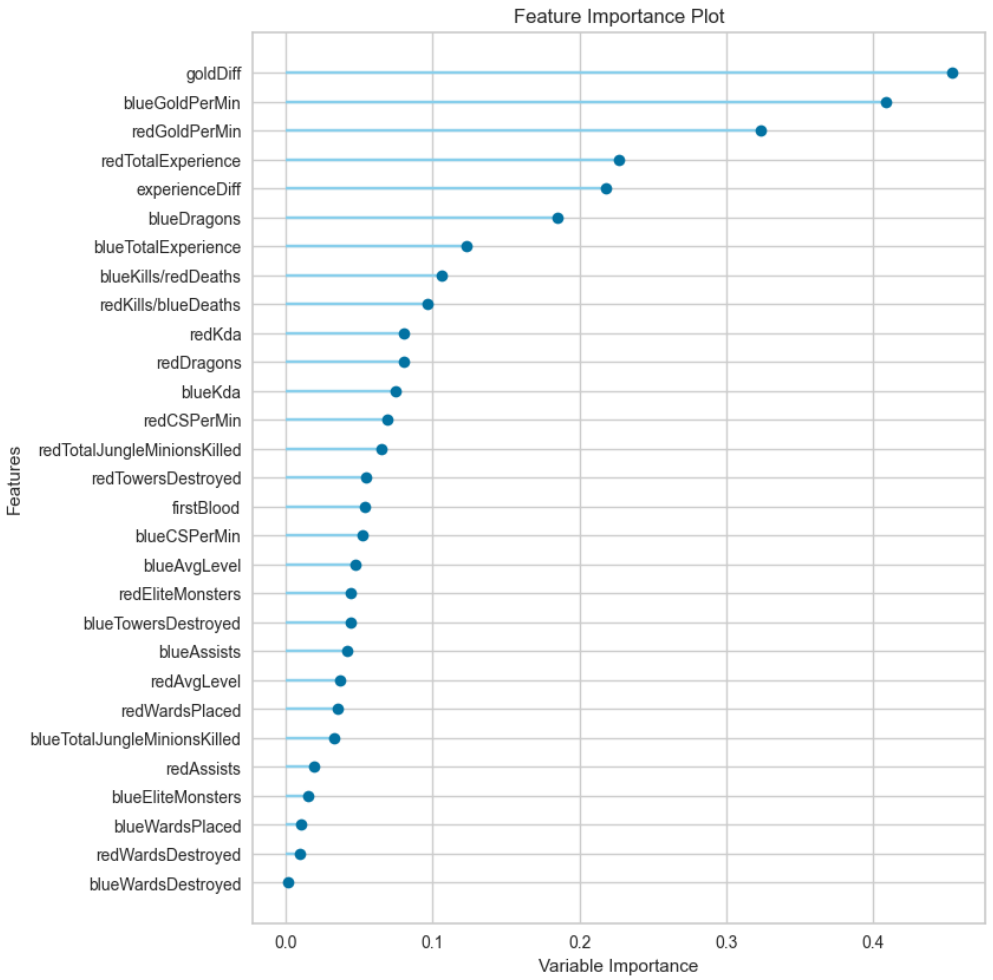

Figura 9. Importância dos atributos na classificação.

A Figura 10 apresenta o processo de Recursive Feature Selection aplicado ao modelo. O objetivo do Recursive Feature Selection é selecionar features recursivamente considerando conjuntos de features cada vez menores. Primeiro, o estimador é treinado no conjunto inicial de features e a importância de cada feature é obtida (usando os coeficientes de uma regressão logística, por exemplo). Em seguida, as features menos importantes são eliminadas do conjunto atual de features. O procedimento é recursivamente repetido no conjunto obtido até restar 1 feature. Por fim, o conjunto de features com o melhor score é selecionado. Observamos que o conjunto de 8 features foi o conjunto que obteve o melhor resultado, tendendo a perder qualidade para uma maior quantidade de features. Resultado esperado, dado a quantidade elevada de features que representam conceitos similares, como já foi dito anteriormente.

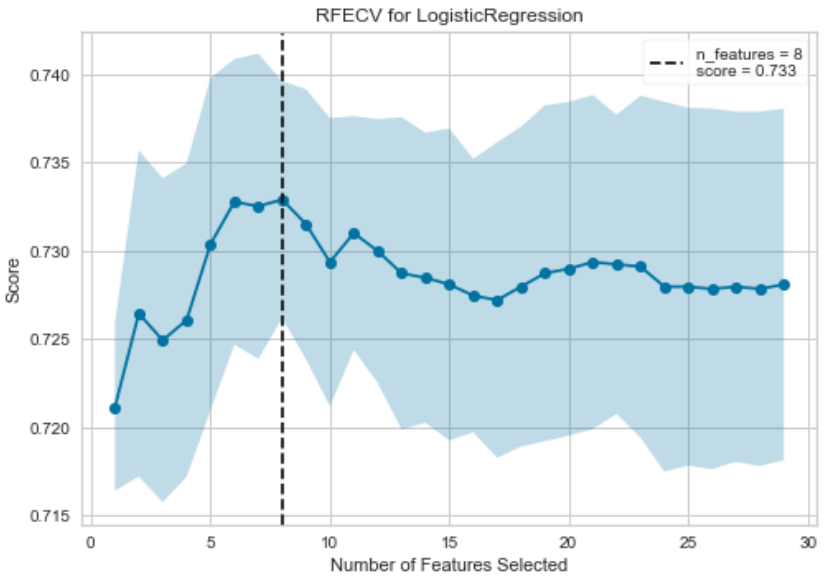

Figura 10. Recursive Feature Selection.

4. Conclusões

Este trabalho buscou predizer a equipe vencedora de uma partida de League of Legends com base nas estatísticas da partida obtida aos 10 minutos de jogo. Os resultados obtidos foram satisfatórios, mostrando que, de fato, existem certos atributos que aumentam as chances de vitória da equipe, destaque para os atributos relacionados ao ouro e à experiência conquistada pela equipe até os 10 minutos. É entendível que estes atributos tenham uma maior importância nas condições de vitória uma vez que os outros atributos indiretamente estão relacionados aos atributos de ouro e experiência. Inclusive, esta grande correlação entre os atributos da base, foi evidenciada pela etapa de *feature selection*, onde foi selecionado apenas as 8 features mais relevantes ao classificador das 29 features totais.

A classificação obteve bons resultados, tendo uma precisão média de 72% para os dados de teste, mostrando que os primeiros 10 minutos de partida podem ser decisivos para o restante da partida que pode durar mais de 1 hora.

Como trabalhos futuros, podemos citar a adição dos campeões selecionados por cada equipe ao estimador. Esta informação pode ser obtida acessando a API do League Of Legends, passando o ID da partida, que está presente nesta base de dados. Os campeões selecionados pela equipe, no LoL, costuma ser um fator determinante na dinâmica da partida, principalmente levando em consideração que a cada nova atualização do jogo, certos campeões costumam ficar “desbalanceados”, isto é, ficam mais fortes que o restante dos campeões do jogo, podendo levar a uma grande vantagem na partida.

5. Referências

R. Ani, V. Harikumar, A. K. Devan and O. S. Deepa, "Victory prediction in League of Legends using Feature Selection and Ensemble methods," 2019 International Conference on Intelligent Computing and Control Systems (ICCS), 2019, pp. 74-77, doi: 10.1109/ICCS45141.2019.9065758.

“O que é AUC e ROC nos modelos de Machine Learning.” Medium, 21 May 2019. Disponível em: <https://medium.com/@eam.avelar/o-que-%C3%A9-auc-e-roc-nos-modelos-de-machine-learning-2e2c4112033d>.

“Multicolinearidade.” Unicamp. Disponível em: <https://www4.eco.unicamp.br/docentes/gori/images/arquivos/EconometriaI/Econometria_Cap10_Multicolinearidade.pdf>.

“Feature Selection. Olá! No post de hoje falaremos sobre… | by Guilherme Duarte | Datarisk.io.” Medium, 19 August 2020. Disponível em: <https://medium.com/datarisk-io/feature-selection-24338ce39067>.