Relatório do Projeto:

Modelos de Previsão de Partidas Ranqueadas de LoL

Introdução:

O projeto desenvolvido trata-se da construção de algoritmos que recebem uma base de dados contendo diferentes partidas do jogo “League of Legends” (LoL) e informações e valores estatísticos sobre os distintos aspectos e elementos presentes em cada uma. O objetivo é que, através da análise desses dados, os programas desenvolvidos consigam prever corretamente qual é o time vencedor: o azul ou o vermelho.

Visando um desenvolvimento mais prático e de fácil compreensão, separei o projeto em 4 partes principais: a análise exploratória dos dados, que se resume em entender as informações e corrigir inconsistências, o pré-processamento, onde os dados são alterados para que a máquina consiga entendê-los melhor, a modelagem, em que são criados os modelos ou programas que recebem essas informações reestruturadas, as entendem e realizam as previsões do objetivo e, por fim, a etapa de avaliação, em que é verificado por métricas e medidas a capacidade dos programas de atingir o objetivo de prever o time vencedor.

Tratamento de Dados:

A primeira etapa do desenvolvimento do projeto consiste em carregar os dados da nossa base no ambiente virtual e verificar se existiam valores faltantes nas colunas ou se alguma informação havia sido incorretamente inserida. Visto que não havia nenhuma dessas inconsistências presentes, não precisei alterar nada inicialmente e prossegui no compreendimento das informações presentes e reparei que, apesar de termos 40 variáveis diferentes, tínhamos 19 pares de colunas que representavam a mesma informação, porém para times diferentes. Isso é, existia, por exemplo, uma variável com o número de abates do time azul e outra com o número de abates do time vermelho.

Vendo que possuíamos muitas informações e que isso dificultaria o processo dos algoritmos para realizar as previsões, exigindo mais tempo e poder computacional, decidi combinar esses pares em apenas uma única variável que representaria a diferença entre os valores de cada time, ou seja, as variáveis de quantidade de mortes de cada time, por exemplo, se passariam a ser uma variável que representaria a diferença entre a quantidade de mortes dos times em cada partida registrada. Além disso, removi algumas variáveis que não representavam nenhuma informação importante ao projeto, como a de ID da partida, ou que repetiam uma informação já fornecida por outra variável, como a que indica se o time vermelho obteve o primeiro abate ou não, que fornecia a mesma informação que a variável que indicava se o time azul tinha tido o primeiro abate, mas com os valores inversos.

Após a criação dessas variáveis de diferenças entre as informações de pares, também criei algumas outras: a diferença de KDA (cálculo que mede o desempenho da equipe de acordo com a soma da quantidade de abates e assistências dividida

pela quantidade de mortes) entre os times, além de 3 colunas de proporção: uma do KDA dos times, a segunda da de sentinelas colocadas pelos times e a última de sentinelas destruídas pelos times.

Com isso, a base de dados passa a ter 22 variáveis:

- A diferença de abates entre os times
- A diferença de mortes entre os times
- A diferença de assistências entre os times
- A diferença de ouro obtido entre os times
- A diferença de ouro obtido por minuto entre os times
- A diferença de experiência obtida entre os times
- A diferença de nível médio da equipe entre os times
- A diferença de “minions” eliminados entre os times
- A diferença de “minions” da selva eliminados entre os times
- A diferença de “minions” eliminados por minuto entre os times
- A diferença de dragões obtidos entre os times
- A diferença de arautos obtidos entre os times
- A diferença de monstros de elite obtidos entre os times
- A diferença de sentinelas destruídos entre os times
- A diferença de sentinelas colocados entre os times
- A diferença de torres destruídas entre os times
- A diferença de KDA entre os times
- A proporção de KDA dos times
- A proporção de sentinelas colocados dos times
- A proporção de sentinelas destruídas dos times
- O time que obteve o primeiro abate
- **Variável Alvo**: O time vencedor

Tratamento de Outliers:

A partir dessas informações, procurei entender melhor a distribuição dos valores em cada um dos elementos para identificar a presença de “outliers”, ou seja, valores extremos que fogem muito do padrão. É importante encontrar esses casos, pois os algoritmos desenvolvidos podem ter um impacto muito negativo ao se depararem com eles, resultando em previsões menos precisas e incorretas, então é necessário tratá-los corretamente.

Para o caso desse projeto, eu notei que muitas das variáveis tinham valores que estavam acima ou abaixo dos padrões apresentados. A solução que escolhi foi a de não realizar nenhum tratamento, pois após o uso de métodos que buscavam reduzir o impacto dos outliers, percebi que os resultados acabaram sendo inferiores a quando nada era feito. Como os valores não estavam incorretos, apenas com variações diferentes e que faziam sentido por estarmos falando de um jogo que tem diferentes estratégias a serem tomadas, essa escolha me pareceu a mais lógica.

Correlação das Variáveis:

Com os valores corrigidos e tratados, parti para uma análise de correlação das variáveis. Aqui, quantifiquei a relação entre cada uma das variáveis, variando de –1 a 1, para poder entender se o aumento ou a queda de um valor de um elemento específico indicava como outras variáveis iam se diferenciar. De forma mais simples, se o a correlação de duas variáveis fosse mais próxima de –1, isso indica que enquanto o valor de uma delas cai, a outra aumenta, se a correlação fosse mais próxima de 1, indica que ambas sobem ou caem “juntas” e, se fosse mais próxima de 0, isso indica que a queda ou aumento de uma variável não indica nenhum comportamento específico ou esperado no valor de outra variável.

Essa análise me permitiu verificar variáveis que não possuíam nenhuma correlação significativa, principalmente em relação à variável que indica qual time vencedor, e, portanto, não seriam úteis para os modelos de previsão, apenas fornecendo informações desnecessárias e aumentando a complexidade e o tempo. Retirei também algumas variáveis que apresentavam correlação perfeita com alguma outra informação, já que seriam duas variáveis diferentes que forneceriam os mesmos dados e isso poderia causar confusão para o algoritmo.

Dessa forma, mantive apenas 12 variáveis além do alvo, que possuíam informações únicas e relevantes para o objetivo de prever qual o time vencedor, o que permitiu que diminuíssemos a quantidade de dados que o algoritmo vai receber, mas sem prejudicar a sua qualidade e as previsões resultantes.

Separação da Base em Treino e Teste:

O próximo passo foi separar a base de dados total em duas duplas diferentes: a dupla das bases de treino e a dupla das bases de teste, em que uma das bases da dupla receberá apenas a variável que queremos prever corretamente e a outra contém todas as outras informações presentes, conhecidas como variáveis independentes.

A separação em treino e teste serve para garantirmos o bom funcionamento do código. Primeiro, a máquina recebe as bases de treino para estudar os padrões dos dados fornecidos e aprender as características que indicam qual time vencedor. Com esse processo feito, para garantir que os modelos conseguem prever o objetivo com dados novos, é passada a base de teste apenas com as variáveis independentes para que ele identifique qual time venceu. Posteriormente, para medir a capacidade do algoritmo, os resultados são comparados com o gabarito, que é a base de teste contendo apenas a variável de qual time vencedor.

De forma resumida, separamos as variáveis para que os modelos possam prever sem terem noção do resultado, permitindo resultados mais realistas, e separamos em treino e teste para o modelo, primeiramente, compreender as informações e suas características e, depois, para averiguar se esse aprendizado foi bem efetuado e consegue entender novas informações, como se primeiro estudasse para uma prova e depois a fizesse.

Padronização e Balanceamento:

No próximo momento, realizei a padronização das variáveis independentes. Esse processo consiste em colocar todos os valores presentes em uma mesma escala, visando facilitar a compreensão do modelo e evitar a má interpretação do algoritmo de valores que podem ter uma variação muito pequena ou muito grande em comparação com os outros elementos presentes na base. Com isso, garanto que as variáveis contribuam de maneira igualitária para as previsões.

Por fim, antes de prosseguir para a etapa de modelagem, em que construo de fato os modelos de previsão de time vencedor, me certifico de que a base de treino está balanceada para os possíveis resultados que os algoritmos podem obter ou seja, verifico se tenho o mesmo número de casos em que o time azul ganhou e casos que o time vermelho foi o vitorioso. Essa verificação é importante, porque se tivermos muitos mais casos de uma equipe ganhadora do que de outra, o algoritmo vai acabar entendendo exageradamente essas informações majoritárias e não terá aprendido o suficiente sobre os casos minoritários e, portanto, não conseguirá identificá-los.

Para o caso desse projeto, percebi que a quantidade de vitórias entre os times era praticamente a mesma, com um pouco mais de casos em que o vermelho vencia. Por essa diferença ser bem pequena, concluí que não seria necessário implementar dados sintéticos em que o time azul ganhava para balancear o treinamento. Assim, pude seguir em frente com a modelagem sem haver o risco de enviesamento do código para um dos resultados possíveis.

Modelagem:

A etapa de modelagem é onde construí propriamente os códigos de previsão para o elemento que desejo, ou seja, a parte em que foram criados os modelos que identificam o time vencedor de cada partida registrada. Utilizei 4 tipos diferentes de modelos, com características e vantagens distintas, para comparar os resultados e escolher qual o mais eficaz, mas é importante ressaltar que todos passaram pelo mesmo passo a passo: utilizei de ferramentas que buscam as melhores configurações para cada um, estabeleci esses resultados encontrados nos modelos oficiais e, então, comecei o processo de aprendizado através da base de treino e conclui com a efetuação de previsões com dados inéditos da base de teste. Os 4 modelos escolhidos foram:

- **Regressão Logística**: Esse modelo é mais simples e prático de ser aplicado, então não requer muito tempo ou poder computacional, e é comumente utilizado para projetos de classificação binária, ou seja, quando o que queremos descobrir pode ter 2 resultados, o que é o caso desse projeto, que quer prever se o time vencedor foi o azul ou o vermelho.
- **Random Forest**: Um modelo levemente mais complexo, mas com maior confiabilidade, pois se trata de um conjunto de modelos mais simples, chamados de Árvores de Decisão, que possuem diferenças entre si ao analisar os dados. Ao final, ocorre uma “votação” com os resultados de cada um desses

modelos para definir qual previsão majoritária para cada caso recebido para ser retornada oficialmente para nós.

- **SVM**: Naturalmente otimizado, esse modelo busca separar os dados em suas respectivas classificações da maneira mais eficaz possível, encontrando o fator de separação ideal para aquele caso. Esse é um dos modelos mais robustos e utilizados em projetos de ciências de dados por ser resistente a outliers, ter uma boa capacidade de generalização e conseguir captar relações entre as variáveis menos evidentes. Contudo, esse modelo requer maior tempo e poder computacional.

- **XGBoost**: Também extremamente otimizado, o modelo de XGBoost consiste também em vários modelos simples de Árvore de Decisão. Aqui, uma primeira árvore apresenta seus resultados e a segunda árvore identifica os erros que ela teve e busca corrigir e apresentar novos resultados que também serão identificados e corrigidos por uma nova árvore e assim subsequentemente até chegar a um ponto em que os resultados todos são juntados para fornecer as melhores previsões possíveis. Esse modelo é muito robusto, pouco sensível a outliers e “overfitting”, mas é, consequentemente, mais demorado e custoso computacionalmente.

Avaliação dos Resultados:

Para entender qual modelo foi o mais satisfatório, a etapa de avaliação conta, principalmente com o uso das métricas de F1-Score, pois indica com maior equilíbrio a capacidade de identificação das classes possíveis e de previsões corretas do código; a acurácia, que mostra a taxa de acerto de forma geral; a curva ROC e AUC Score, para entender o quão eficaz o modelo é para o objetivo, além de “cross validation”, que garante que o modelo oficial não tenha sofrido de problemas, como enviesamento, por causa da forma como os dados foram estruturados e organizados ao serem passados ao modelo e gráficos de matriz de confusão, que é uma demonstração visual das classificações que o modelo vez, mostrando a quantidade numérica de previsões feitas para cada categoria, quantas estavam corretas e quantas foram categorizadas incorretamente e para qual categoria elas foram previstas.

Tendo estabelecido essas métricas, foi possível notar que o desempenho dos modelos ao se depararem com dados novos, ou seja, a base de teste, foi extremamente parecido, com acurácias, F1-Score, curva ROC e AUC Score ficando entre 0,72 e 0,73, o que indica que, em média, 73% dos casos são previstos corretamente. A maior mudança ocorre ao compararmos as métricas para os dados de treino, isso é, aqueles que os modelos usaram para aprender: vemos que o modelo de Random Forest apresentou métricas bem melhores do que as suas de teste.

Isso significa que esse modelo teve maior dificuldade de lidar com informações inéditas e, portanto, se adaptou de maneira exagerada aos dados de aprendizado.

Com isso, por ter mostrado essa tendência problemática, o Random Forest é o menos adequado para o objetivo que buscamos concluir.
Em relação aos 3 modelos restantes, a maior diferença entre eles é perceptível através das matrizes de confusão: os modelos de Regressão Logística e XGBoost apresentaram quantidade de acertos e erros de cada possível classe extremamente similares, com o último mostrando apenas um pouco mais de acertos e sendo levemente mais equilibrado para ambas as classes. O SVM, por outro lado, mostrou uma menor diferença entre a quantidade de erros para as previsões de cada classe, mas ainda mantendo uma boa taxa de acerto.

Com isso, é possível afirmar que esses 3 modelos são boas opções para serem aplicados de maneira oficial: a regressão logística apresentou um desempenho tão bom quanto outros modelos mais robustos, mesmo sendo mais simples, o que a faz ser ideal para obter bons resultados mais rapidamente, sem a necessidade de um grande poder computacional, o XGBoost, por sua vez, é mais lento, mas sua maior resistência a "outliers" e "overfitting" podem resultar em menos problemas futuros com novas amostras de dados, constantemente atualizadas, do que com modelos mais simples, com uma maior facilidade de adaptação. Por fim, o SVM é mais complexo e de difícil interpretação, mas aparentou ter uma maior consistência de uma maneira geral, o que o tornaria ideal para cenários em que a capacidade de previsão do modelo de forma ampla é o mais importante. Com isso, esses resultados permitem que a escolha possa ser feita de acordo com as condições em que os modelos seriam aplicados e, por tanto, varia de acordo com a demanda, o que permite que a manutenção e melhora dos algoritmos possam ser feitas de maneira mais específica para cada caso.

Conclusão:

Como o objetivo é identificar corretamente o time vencedor, seja lá qual seja, o modelo de SVM é o mais adequado e seria a escolha ideal. Ainda assim, as capacidades gerais de todos os modelos ainda deixam a desejar, o que permite a conclusão de que, para maior certeza sobre o resultado de cada partida, seriam necessários maiores detalhes sobre os jogadores de cada equipe, como suas classes e outros dados estatísticos. Além disso, apesar da criação de novas variáveis através das informações originais da base de dados, é possível que a criação de outras informações possa ser feita para melhorar as previsões ou até, de maneira inversa, o uso apenas das variáveis de maior importância para os modelos apresentados, sendo necessário o desenvolvimento contínuo do projeto, mesmo após essa conclusão inicial. Dessa forma, buscarei melhorar esse projeto para alcançar melhores resultados e publicarei no GitHub as atualizações futuras.