Nanodegree Engenheiro de Machine Learning

Proposta de projeto final

Ewerton Longo da Silva

15/05/2019

Utilização de aprendizado não supervisionado para melhorar *gameplay* em League of Legends

Histórico do assunto

Cada vez mais os *e-sports* (esportes eletrônicos) vem ganhando mais espaço no mundo e atingindo um grande número de pessoas ao redor do mundo. Esse "*boom*" fez com que crescessem ainda mais a quantidade de eventos realizados em todo o planeta e também fez com os investimentos sobre estes alcançassem a casa de milhões de reais [1]. Curiosamente, ou não, eu sou uma dessas pessoas fissuradas em *e-sports*, sobretudo em um especial chamado League of Legends.

Jogo regularmente o League of Legends, também conhecimento como LoL, desde 2014 e de lá pra cá venho a cada dia tentando melhorar a minha *gameplay* (ou jogabilidade) e desenvolvendo novas estratégias para me tornar melhor. Foi então durante o Nanodegree de Engenheiro de Machine Learning da Udacity que tive a ideia de unir o útil ao agradável, sendo esta: utilizar *machine learning* para ajudar a melhorar a minha *gameplay* em League of Legends.

Após a conclusão deste projeto espero também melhorar, além da minha jogabilidade, as dos meus amigos e também a da minha esposa, pois jogamos juntos diariamente e time unido vem a crescer unido!

Descrição do problema

O objetivo deste projeto é utilizar aprendizado não supervisionado sobre os dados das partidas de League of Legends que joguei e me auxiliar, através de inferências obtidas a partir do modelo de *machine learning* proposto, a melhorar a minha *gameplay*. O projeto será construído com foco no meu usuário do jogo e nas minhas partidas, porém nada impede deste projeto ser utilizado como modelo para análise das partidas de outros jogadores.

Conjuntos de dados e entradas

A Riot, empresa proprietária do League of Legends, disponibiliza API's para desenvolvedores utilizarem alguns dados do jogo para consultas, criação de estatísticas e análises, etc. Para isso eles construíram um portal próprio para estes onde são cedidas documentações e exemplos, além de uma comunidade para que os desenvolvedores troquem experiências [2].

Em pesquisa de como obter esses dados e de como utilizar as API's disponibilizadas pela Riot foi encontrado um *wrapper,* chamado RiotWatcher, construído para a linguagem Python que facilita o acesso aos dados do jogo utilizando "por trás" as próprias API's oficiais da empresa [3]. O RiotWatcher será aplicado neste projeto para a obtenção dos dados.

Os dados que serão utilizados para a construção da análise deste projeto serão de partidas que joguei recentemente. Serão utilizadas muitas *features* destas partidas para a construção do modelo, como por exemplo: quantidade de *kills*, *defeats* e *assists*, tempo de duração de jogo, elo do jogador, posição do jogador no time, score de visão no mapa, entre outras. Todos os atributos das partidas jogadas podem ser encontrados na documentação da API utilizada para recuperar estes dados [4].

Antes de utilizar a API da Riot para recuperar os dados, há a necessidade de realizar um cadastro (processo já realizado) no seu portal do desenvolvedor e gerar uma *Development API Key* para obtenção dos dados. Esta *key* tem um tempo de expiração e sempre que a mesma tiver o seu prazo de validade vencido há a necessidade de gerar um nova.

Descrição da solução

A fim de melhorar a minha *gameplay* será utilizado um algoritmo de aprendizado não supervisionado (ainda não escolhido) que irá inferir quais os principais elementos de uma partida que impactam em uma vitória ou derrota (uma vez que não há empate no League of Legends). A ideia é utilizar também outros

algoritmos de *machine learning* para auxiliar na implementação de identificação de relevância dos atributos dos dados, uma vez que são disponibilizados muitos. O resultado obtido deverá ser metrificado utilizando a *feature* que indica se o resultado da partida foi vitória ou derrota, também disponibilizada via API.

Modelo de referência (benchmark)

Foi realizada uma pesquisa de benchmark e identifiquei que existem vários sites que possuem como objetivo auxiliar de diversas formas um jogador a melhorar o seu desempenho em League of Legends e de conhecer melhor sobre o jogo. São exemplos: More Legends [5], Mobafire [6], Champion GG [7], entre outros.

Contudo foi o Mobalytics [8] aquele que chegou mais próximo deste trabalho justamente por possuir o mesmo objetivo. Esta aplicação analisa os últimos jogos do jogador, gera informações e inclusive direciona o usuário no que precisa melhorar dando *scores* de referência para cada ponto analisado.

Esta plataforma apresenta também muito de seus dados em gráficos e gera um *score* próprio chamado GPI (Gamer Performance Index) com o intuito de auxiliar no entendimento das forças e fraquezas de cada jogador. É relatado no site da aplicação que com base no GPI calculado, *coaches* de nível profissional ou desafiante (*rank* mais alto no League of Legends) auxiliam o jogador a entender qual o melhor caminho para subir cada vez mais seu nível.

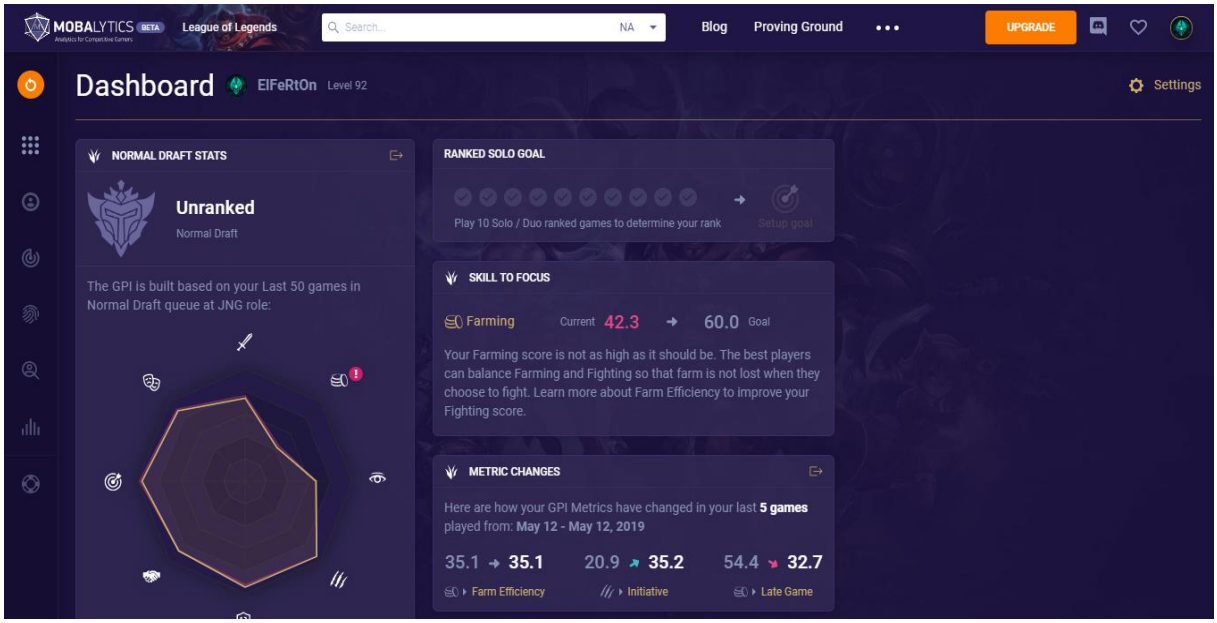

Imagem retira a partir do meu usuário no Mobalytics

Métricas de avaliação

Para avaliar o resultado obtido será utilizada a acurácia para comparar a segmentação gerada com o atributo original do conjunto de dados que identifica se aquela partida foi vitória ou derrota. Será realizada também uma comparação entre os atributos destacados pelo Mobalytics e aqueles também destacados pelo modelo gerado.

Design do projeto

O fluxo de trabalho a ser realizado segue as seguintes etapas:

1. Extração de dados

   Através do *wrapper* citado acima, o RiotWatcher, serão extraídos os dados de jogos que joguei, junto com possíveis outros dados de domínios, como por exemplo de campeões e itens. Esses dados são extraídos em formato de dicionários do Python.

2. Modelagem de dados

   A etapa de modelagem consiste em carregar os dados a partir de dicionários em data frames, para melhor manipulá-los e também os colocar em apenas uma dimensão, quando necessário. A necessidade de colocar os dados em apenas uma dimensão vem do fato de alguns estão presentes em listas ou em dicionários hierarquizados. O atributo que identifica vitória ou derrota também será retirado.

3. Identificação de relevância dos atributos

   Como a quantidade de atributos de uma partida disponibilizados pela API da Riot é grande, há uma necessidade de identificar os mais relevantes e se estes possuem correlações entre si. Para isto a ideia é importar um modelo de aprendizado supervisionado para realizar esta classificação e produzir também uma matriz de dispersão.

4. Pré-processamento de dados

   Nesta etapa será realizado o pré-processamento dos dados para criar uma melhor representação das partidas ao executar um escalonamento dos dados e detectando os discrepantes.

5. Transformação de atributo

   Implementação do PCA para descobrir quais dimensões dos dados melhor maximizam a variância dos atributos envolvidos e redução de dimensionalidade, se necessário.

6. Clusterização

   Implementação de algoritmo de *clustering* para identificar as segmentações presentes nos dados de partidas. A ideia é que sejam aplicados inicialmente apenas dois clusters, caracterizando as vitórias ou derrotas nas partidas.

7. Conclusão

   Após o resultado do *clustering* e análise nos principais atributos que ajudam a caracterizar uma vitória ou derrota em uma partida, será criado um novo atributo a partir da segmentação dos dados e eventualmente compará-lo com o que identifica vitória ou derrota no conjunto de dados original (retirado no início do processo).

Referências Bibliográficas

[1] https://economia.uol.com.br/noticias/redacao/2019/01/20/esports-crescimento-mercado-campeonatos-torneios.htm. **"eSports devem alcançar 450 milhões de pessoas em 2019, diz consultoria".** Acessado em 15/05/2019.

[2] https://developer.riotgames.com/. **"Riot Games API".** Acessado em 15/05/2019.

[3] https://riot-watcher.readthedocs.io/en/latest/. **"RiotWatcher's documentation".** Acessado em 15/05/2019.

[4] https://developer.riotgames.com/api-methods/#match-v4. **"Riot Games API – Match-V4".** Acessado em 15/05/2019.

[5] https://morelegends.com/pt. **"More Legends"**. Acessado em 15/05/2019.

[6] https://www.mobafire.com/. **"Mobafire".** Acessado em 15/05/2019.

[7] https://champion.gg/. **"ChampionGG".** Acessado em 15/05/2019.

[8] https://mobalytics.gg/. **"Mobalytics".** Acessado em 15/05/2019.