Luizalabs

Machine Learning Engineer

Descrição do problema

Para se preparar para o campeonato de Gods Unchained[1], você precisa pensar em uma estratégia na hora de montar seu deck. Para isso, um primeiro passo é classificar os cards como “early” ou “late” game com base em algumas informações relacionadas ao card como “mana”, “attack” e “health”.

Como você é excelente em engenharia de software e tem conhecimento a respeito de machine learning, resolveu criar um modelo para classificar se determinado card se enquadra em um card de “early” ou de “late” game.

Ainda, para auxiliar outras pessoas da comunidade nesse mesmo processo, você decide criar uma API para expor publicamente este modelo.

Desta maneira, o desafio pode ser dividido em duas partes:

Treinamento do modelo

Dado o dataset abaixo contendo informações relacionadas aos cards de Gods Unchained, implemente e treine um modelo para classificar se o card se enquadra em uma estratégia de “early” ou “late” game, com base nas features “mana”, “attack”, “health”, “type” e “god”.

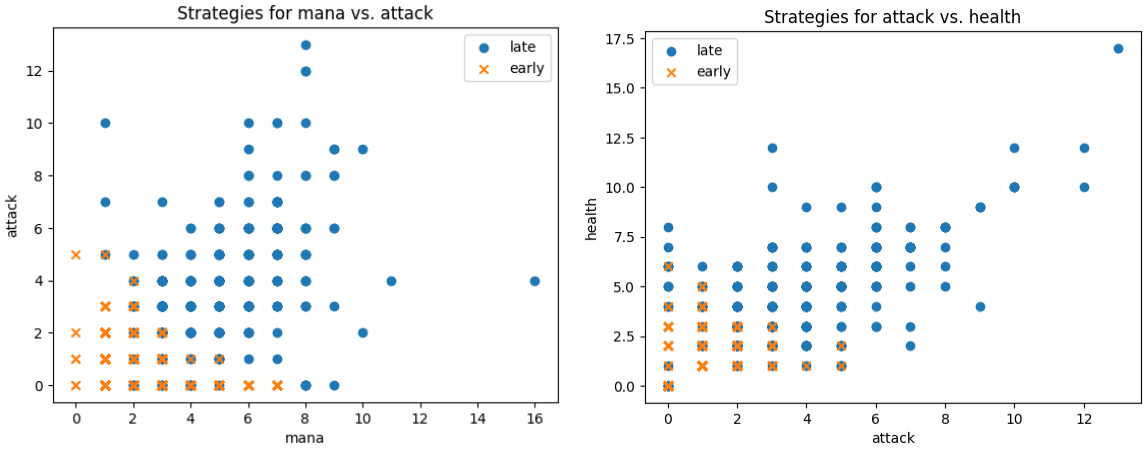
	id	name	mana	attack	health	type	god	strategy
0	1118	Firewine	5	0	0	spell	nature	early
1	1036	Leyhoard Hatchling	10	2	1	creature	magic	late
2	244	Aetherfuel Alchemist	6	4	4	creature	neutral	late

Descrição dos dados:

id (int) - Identificador único da carta no banco de dados do jogo
name (str) - Nome da carta
mana (int) - Custo de mana para colocar a carta na mesa
attack (int) - Dano que a carta causa ao oponente
health (int) - Resistência ao dano ou durabilidade da carta
type (str) - ['spell','creature','weapon','god power']
god (str) - ['death','neutral','deception','nature','light','war','magic']
strategy (str) - Estágio do jogo ['early','late']

[1]Gods Unchained é um card game free-to-play, semelhante a Magic The Gathering, onde os jogadores podem adquirir cards utilizando criptomoedas.

Abaixo seguem algumas visualizações dos dados:



A distribuição estatística dos dados está em constante evolução, portanto, o foco da solução está na criação de um código de simples manutenção por terceiros que permita execuções recorrentes futuras de forma prática.

Exposição do modelo

Com o modelo previamente treinado, implemente uma maneira de expor o mesmo através de uma API. Essa API deve possuir ao menos uma rota, que receba o ID do card e retorne a predição (**strategy**). O ID utilizado para as requisições na API pode vir tanto do arquivo **train.csv** quanto do **test.csv**.

Tenha em mente que esta API deve estar pronta para produção, deve ter fácil manutenção e ser escalável para um grande volume de usuários.

O que é esperado

- Implementar a solução em linguagem Python.
- Disponibilizar o projeto em um repositório git privado.
- Ser possível realizar o treinamento de forma reprodutível.
- Desenvolver a API seguindo os princípios de REST.
- Documentar os pontos do projeto que achar relevante.
- Utilizar quaisquer bibliotecas ou frameworks que julgar necessário.
- Criar um Dockerfile que exponha de maneira simples a API.
- Seguir boas práticas de Orientação a Objetos e de Git.

O que é opcional

- Entrega de EDA, visualização de dados ou análise preditiva.
- Incluir manifestos para deploy da aplicação no Kubernetes.
- Expor a API com algum tipo de documentação (e.g. Swagger, RAML, etc).
- Documentar quaisquer passos extras realizados que achar relevante para explicar sua solução.

O que **não** é necessário

- Qualquer tipo de frontend.

Considerações finais

Será julgado seu poder de síntese, de fazer mais com menos, de arquitetar a solução da forma mais simples e robusta possível, de uma forma que seja manutenível, com qualidade e elegância.

Você pode, em caso de dúvidas relacionadas ao desafio, nos encaminhar um email com essas perguntas.

Esperamos que você consiga concluir o desafio em 1 semana, mas se precisar de mais tempo basta nos informar.

Esperamos que se divirta no processo.

Boa sorte!