Desmistificando o Cartola FC: O Baseline

Ciro Ceissler
RA 108786
ciro.ceissler@gmail.com

Lucas de Souza e Silva
RA 140765
lucasonline1@gmail.com

Matheus Laborão Netto
RA 140765
mln.laborao@gmail.com

Ramon Nepomuceno
RA 192771
ramonn76@gmail.com

I. Introdução

A *baseline* para o projeto do Cartola FC é baseada numa implementação disponível em [1], ela consiste primeiramente no tratamento da base de dados para depois realizar os experimentos. A predição dos melhores jogadores em cada rodada utiliza uma rede neural artificial, no código usado como referência explora-se diferentes topologias, variando o número de camadas escondidas e também a quantidade de neurônios. A codificação foi feita na linguagem Python com o auxílio da biblioteca do `scikit-learn` [2].

II. Limpeza dos Dados

A primeira etapa realizada para implementar o modelo de predição foi analisar os dados fornecidos pela API do cartola, fazendo as limpezas necessárias para criar amostras corretas e relevantes para o preditor. Os dados devem continuar consistente após a limpeza dos dados, ou seja, nenhuma coluna poderá ser adicionada, alterada ou removida. Após uma análise prévia dos dados, alguns problemas foram detectados nas informações dos jogadores:

- todos os scouts com valor NaN (*not a number*).
- coluna 'ClubeID' com valor NaN.
- coluna 'Status' com valor NaN.
- pontuação não equivalente a soma ponderada dos scouts.

A coluna 'atletas.clube_id' tem campos repetidos e divergentes: por exemplo, todos os Atléticos (MG, PR, e GO) são ATL. Além disso, há jogadores com siglas diferentes das equipes que eles jogam (por exemplo, Maicosuel [id: 37851]). A coluna 'athletes.atletas.scout' não é informativa. Os scouts de 2015 dos jogadores são cumulativos, ou seja, os scouts dos jogadores vão sendo somados a cada rodada. Entretanto, a pontuação não é. Isso também causa o repetimento de dados.

As linhas que possuem as inconsistências citadas acima são removidas ou atualizadas. Além destas modificações, outras remoções de dados irrelevantes são realizadas, por exemplo jogadores que não participaram de nenhuma rodada, técnicos e jogadores sem posição, jogadores sem nome, entre outros.

A. Atualização dos scouts cumulativos de 2015

Como dito anteriomente, os dados sobre os scouts de 2015 foram disponibilizados de maneira cumulativa pela fornecidos pela API do cartola, ou seja, os scouts de uma rodada são adicionados aos scouts anteriores a cada nova rodada que um jogador participa. Portanto, foi necessário tirar essa acumulação para cada jogador, de maneira que a representação dos dados ficasse coerente.

Para isso, dada uma rodada específica, os scouts de um jogador são subtraídos do máximo dos scouts de todas as rodadas anteriores. Repare que assim há chance do scout 'Jogo Sem Sofrer Gols (SG)' ser negativo se o jogador não sofrer gols na rodada anterior e sofrer na rodada atual. Quando isso acontece, esse scout é atualizado.

B. Verificação da pontuação com os scouts

Por inconsistência na base de dados fornecida pela API do cartola, alguns jogadores possuiam pontuações que não condiziam com seus scouts. Para esses casos, o jogador é removido da base de dados para evitar qualquer tipo de ruído. Ao final, mais de 4000 jogadores foram removidos.

C. Remoção das linhas duplicadas

A última operação realizada para limpar a base de dados foi apagar as linhas repetidas. A existência de linhas repetidas deve-se ao fato de que a partir da primeira participação de um jogador no campeonato, ele aparece em todas as rodadas subsequentes, mesmo que não tenha jogado. As entradas redundantes não são necessárias ao modelo, por isso foram removidas.

III. Criação das Amostras

Continuando a implementação após a limpeza da base de dados, o próximo passo foi transformar os dados para tornar utilizáveis na criação dos modelos. Desta forma, duas operações são realizadas e descritadas abaixo:

- **Selecionar somente as colunas de interesse:** colunas como 'atletas.nome', 'atletas.foto', etc não são relevantes para criação do modelo. No entanto, colunas como o 'AtletaID' e 'atletas.apelido', mesmo que não utilizadas para treinamento do modelo, são importante para avaliar o resultado e, portanto, também serão consideradas.
- **Converter todos os dados categóricos para numéricos:** as colunas 'Posicao', 'ClubeID', 'opponent' e 'casa' serão convertidas para número.

IV. treinamento do Modelo

O modelo preditor utilizado foi uma rede neural artificial com diferentes configurações para avaliar o desempenho delas. Em resumo, o seu funcionamento acaba sendo bem simples, o modelo recebe como entrada os dados de uma determinada rodada e faz uma predição das pontuações dos jogadores para a próxima rodada.

Para estimar a melhor arquitetura para rede, bem como os hiperparâmetros, foi utilizada a estratégia *GridSearch*. Nesse método, todas as combinações possíveis entre os parâmetros são testados usando uma validação cruzada com 5 *folds*. A combinação de parâmetros que for melhor na média dos *folds*, é considerada a melhor. As seguintes combinações de redes neurais foram utilizadas com a numeração de cada item corresponde ao seu `id`:

1) quatros camadas escondidas cada uma com 50 neurônios.
2) três camadas escondidas sendo os número de neurônios de cada uma delas é, respectivamente, 50, 100, 50.
3) quatro camadas escondidas sendo os número de neurônios de cada uma delas é, respectivamente, 50, 100, 100, 50.
4) apenas uma camada escondida com 128 neurônios.
5) duas camadas escondidas cada uma com 128 neurônios.
6) três camadas escondidas cada uma com 128 neurônios.

Optou-se também por normalizar os dados utilizando a estratégia *MinMaxScaler*, que normaliza cada atributo no intervalo [0-1], e utilizar o método de otimização *adam*. *Adam* é um algoritmo de otimização que pode ser usado em vez do procedimento clássico de descida de gradiente estocástico para atualizar os pesos da rede de forma iterativa com base nos dados de treinamento. Uma vez que trata-se de um problema de regressão, a função de ativação da saída escolhida para a rede foi a função linear e a rede será treinada visando minimizar a Root Mean Squared Error (RMSE).

A Tabela I mostra o resultado da estratégia comentada acima para diferentes configurações de redes neurais, variando a quantidade de camadas escondidas e seus neurônios. Os *scores* negativos obtidos indicam que os modelos avaliados tem um desempenho muito ruim, a documentação do `scikit-learn` informa que o melhor score seria 1.

Table I
EXPERIMENTOS DAS REDES NEURAIS

id	fold 1	fold 2	fold 3	fold 4	fold 5	média
1	-19.514	-22.250	-19.641	-19.224	-17.298	-18.879
2	-22.382	-19.712	-19.404	-19.025	-17.154	-18.601
3	-17.114	-19.669	-22.350	-19.490	-19.038	-18.585
4	-19.791	-19.503	-22.334	-19.013	-17.163	-18.562
5	-22.282	-19.824	-20.116	-17.211	-19.424	-18.592
6	-19.796	-17.251	-19.702	-19.019	-22.412	-18.843

A melhor arquitetura foi a que utiliza apenas uma camada escondida e 128 neurônios, sendo a média do seu *score* igual a -18.562. O resultado mostra que novas arquiteturas precisam ser exploradas para conseguir melhorar este resultado, uma vez que as redes neurais testadas tem comportamento muito similar entre elas. O *score* para o conjunto de validação foi de -18.962 pelos resultados analisados anteriormente esse baixo desempenho já era esperado.

A Figura 1 possui os valores da pontuação reais em azul comparado com os preditos em vermelho. O gráfico mostra que os valores preditos estão abaixo do valor real, mostrando características de uma caso de *underfitting* do modelo.

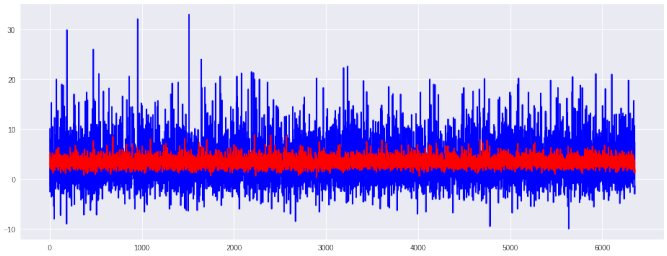

Figure 1. Pontuação Predito vs Real.

V. PROPOSTAS PARA MELHORAR O BASELINE.

A princípio, cogitamos utilizar as seguintes estratégias para melhorar o baseline:

1) Utilização do *Random Forest*. Esse modelo consiste em um método de aprendizado conjunto para classificação, regressão e outras tarefas, que operam construindo uma multiplicidade de árvores de decisão no momento do treinamento e gerando a classe que é o modo das classes ou previsão média das árvores individuais. Essa proposta foi discutida em https://github.com/henriquepgomide/caRtola/issues/33, por esse motivo resolvemos testá-la para avaliar os resultados.
2) Acreditamos também que podemos melhorar os resultados modificando a rede para receber como entrada não só apenas os dados de uma terminada rodada, mas sim um histórico das últimas *n* rodadas. Isso deve-se ao fato de que, pela experiencia que temos assistindo futebol, o desempenho dos jogadores oscila de acordo com um determinado histórico. Faremos essa modificação na esperança que o modelo reconheça esse padrão.
3) Utilizar a estratégia conhecida como one-hot encoding para representar as features categóricas.

REFERENCES

[1] GitHub - Repositório caRtola. https://github.com/henriquepgomide/caRtola/.
[2] scikit-learn: machine learning in Python - http://scikit-learn.org/, Acessado em 18-10-2018.