Projeto Final | Aprendizado de Máquina

Grupo 8

2022-12-07

1. Classificador

Problema

Temos um problema de classificação multiclasse. Como cada jogador tem mais de uma classificação considerada correta, uma abordagem possível é utilizando modelos *multilabel*, que pode atribuir um ou mais rótulos não-exclusivos para uma mesma observação.

Porém, para simplificar o problema, optamos por tratá-lo apenas como um classificador multiclasse, que atribui dentre as N classes disponíveis, um único rótulo. Para não perder a informação de que mais de uma posição pode estar correta, treinamos o modelo no formato longo e para a predição final, agrupamos por jogador a de maior probabilidade.

Função de risco

$$R(g) := \mathbb{E}[\mathbb{I}(Y \neq g(\boldsymbol{X}))] = \mathbb{P}(Y \neq g(\boldsymbol{X}))$$

Adotamos a função de perda 0-1, comum para problemas de classificação, mas não a única.

Data splitting

O conjunto de dados de treino foi separado em treino e validação, para estimar $\hat{R}(g)$. 70% das observações foram para treino e 30% para validação.

Análise exploratória

Modelos

Foram treinados 5 modelos distintos:

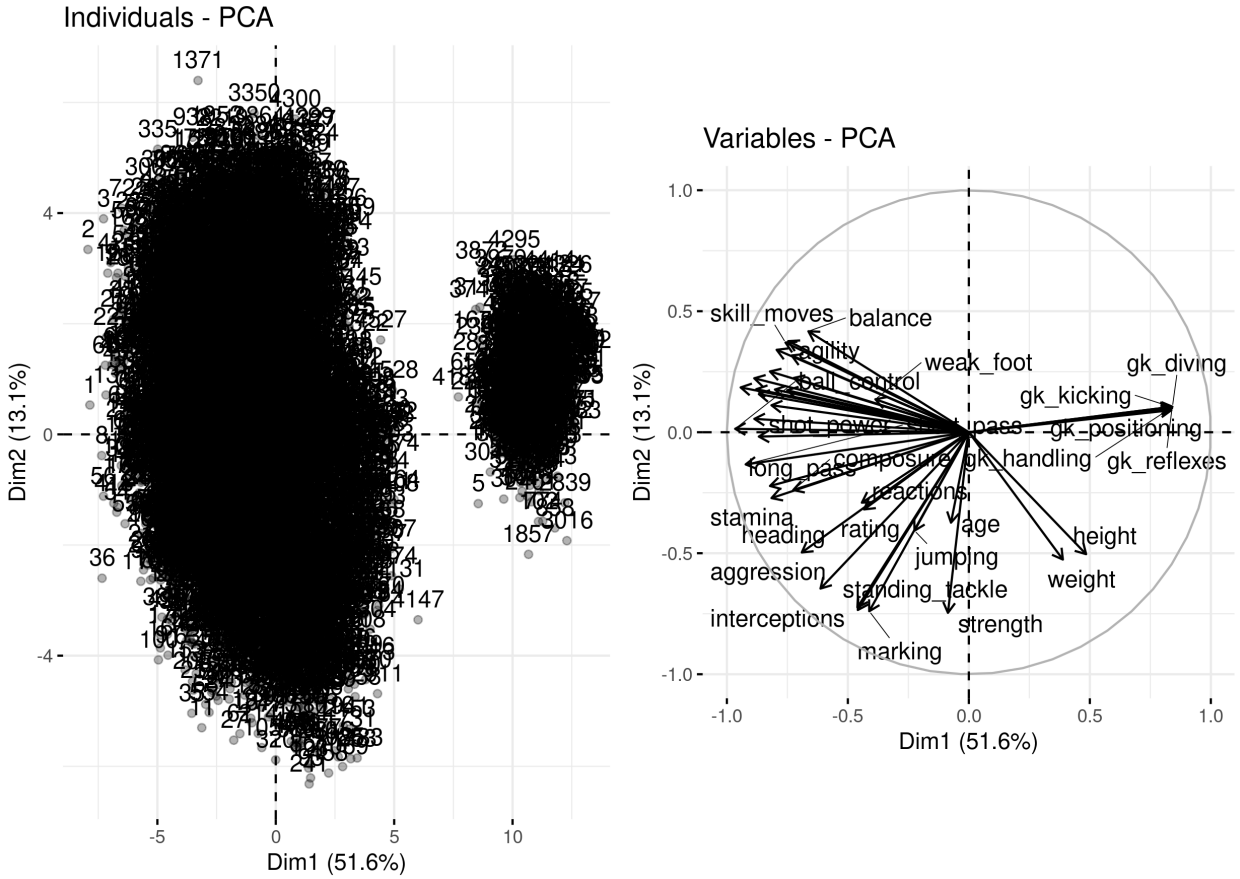
- Regressão multinomial ou *softmax*
- Naive Bayes
- Support Vector Machines
- Árvores de decisão
- Random Forest

O método que forneceu o melhor resultado foi o Random Forest, de acordo com o risco estimado na tabela a seguir.

Modelo	Risco estimado
Naive Bayes	0.3590325
SVM	0.1806500
Regressão multinomial	0.2781557
Árvorede de decisão	0.3862434
Random Forest	0.0090703

Redução de dimensionalidade

Aplicamos a técnica de análise de componente principal (PCA) para redução de dimensionalidade das covariáveis numéricas e há claramente um agrupamento nos dados. Observa-se no gráfico abaixo os primeiros dois componentes, com as observações individuais e a correlação entre as variáveis.



Comentários

O ponto mais difícil foi fazer o tratamento da variável resposta `Preffered_Position`, quebrando as uma ou mais posições de cada jogador. Com mais tempo, seria interessante alterar os *tunning parameters* de cada modelo e também usar uma outra função de risco, como por exemplo a entropia cruzada para multiclasse.

2. Recomendador