INSPER

Ciência dos Dados

Projeto 3

Classificador Automático de Times

André Matarazzo e Gustavo Pierre

ÍNDICE

Introdução: Contexto, objetivo e foco

Recentemente, a renomada EA Sports lançou o FIFA 19, o clássico jogo de videogame de futebol conhecido e amado por muitos "gamers" ao redor do globo. Dentre a mais nova e diferenciada interface, o jogo permite montar times, trocar jogadores e ingressar em diferentes campeonatos e torneios. Por mais que muito fácil para aqueles que já jogavam e entendem de futebol, o jogo pode demonstrar dificuldades aqueles novos à ele, principalmente a quem não tem experiência em futebol como um todo. Na formação de times especificamente, muitas vezes estes "gamers" não sabem em que posição colocar os jogadores, assim como se confundem na própria posição com que o jogador deve jogar.

Com o objetivo de facilitar e melhorar a experiência desses players, os engenheiros programadores André Matarazzo e Gustavo Pierre criaram um formador automático de times, que solucionaria de vez os problemas de players inexperientes. Basta selecionar os jogadores, e o programa encontra a posição destes, monta formação do time e devolve o time completo.

Abaixo, os visionários Decão e Gupi demonstraram o processo de criação do programa, com o intuito de ensinar aspirantes programadores e demonstrar gratidão pelo seu sucesso.

Parte 1: Encontrando o DataSet e iniciando análises exploratórias

Para iniciar a análise e se aproximar do objetivo desejado, um DataSet foi adquirido. Uma vez que FIFA 19 é um jogo novo e um DataSet de seus dados não está disponível na internet, o DataSet lógico a ser utilizado foi o do jogo FIFA 18, tendo em mente que o programa criado por este também possibilitaria a classificação de jogadores no FIFA 19.

DataSet: https://www.kaggle.com/thec03u5/fifa-18-demo-player-dataset

Começamos importando as bibliotecas necessárias, carregando os dados e, em seguida, olhando e aperfeiçoando o DataSet:

```
%matplotlib inline
import pandas as pd
import numpy as np
import math as math
from scipy import stats
import matplotlib.pyplot as plt
import seaborn as sns

fifa18 = pd.read_excel('FIFA.xlsx')
fifa18_2 = fifa18

fifa18.head()

del fifa18['Photo']
del fifa18['Flag']
del fifa18['Club Logo']
del fifa18['Special']

fifa18.columns

fifa18.describe()

fifa18.dtypes
```

Percebendo a existência de colunas em formato object, todas foram limpadas e transformadas em float, para que a classificação futura possa ser feita. Então, o primeiro DataFrame foi criado.

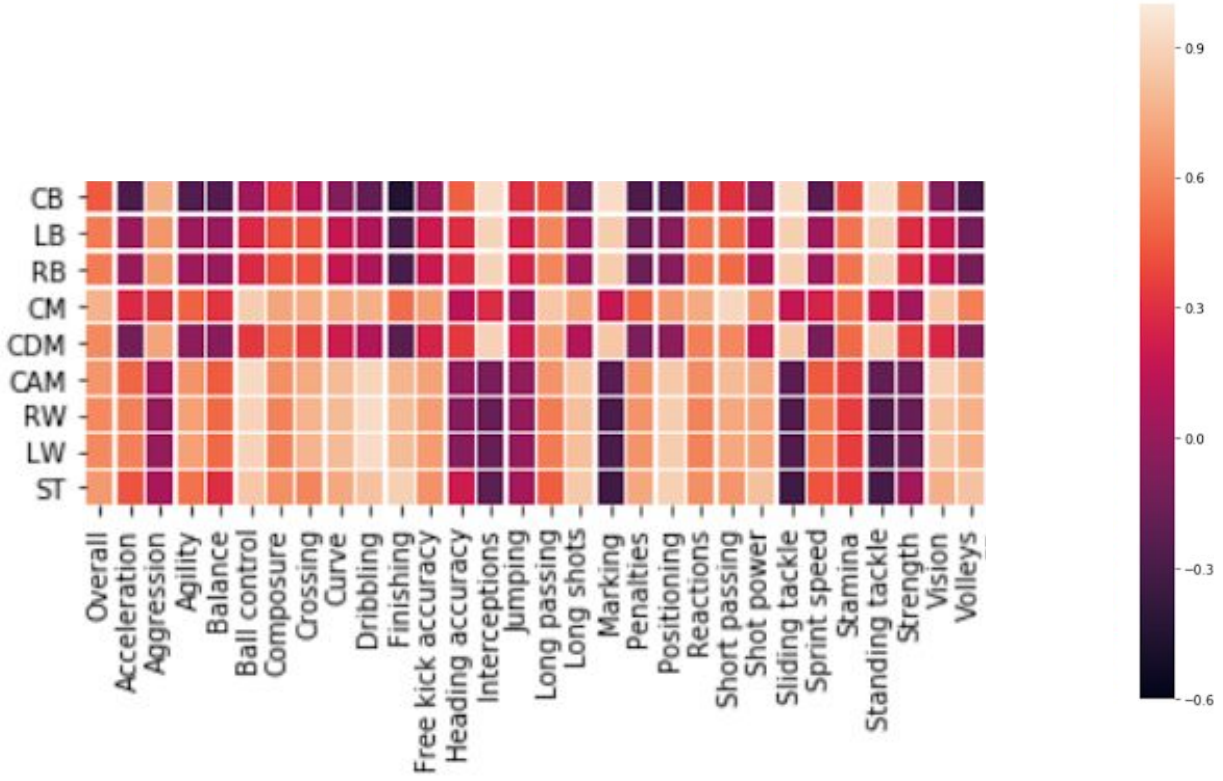
```
from pandas import DataFrame

def troca_obj_float (coluna):
    fifa18[coluna] = fifa18[coluna].astype('float64')

fifa18 = DataFrame(fifa18, columns=['Acceleration', 'Aggression', 'Agility', 'Balance',
       'Ball control', 'Composure', 'Crossing', 'Curve', 'Dribbling',
       'Finishing', 'Free kick accuracy', 'GK diving', 'GK handling',
       'GK kicking', 'GK positioning', 'GK reflexes', 'Heading accuracy',
       'Interceptions', 'Jumping', 'Long passing', 'Long shots', 'Marking',
       'Penalties', 'Positioning', 'Reactions', 'Short passing', 'Shot power',
       'Sliding tackle', 'Sprint speed', 'Stamina', 'Standing tackle',
       'Strength', 'Vision', 'Volleys', 'CAM', 'CB', 'CDM', 'CF', 'CM', 'ID',
       'LAM', 'LB', 'LCB', 'LCM', 'LDM', 'LF', 'LM', 'LS', 'LWB', 'RAM', 'RB', 'RCB', 'RCM', 'RDM', 'RF', 'RM',
       'RS', 'RW', 'RWB','ST','LW'])

for x in fifa18:
    coluna=troca_obj_float(x)
```

Análise Exploratória

Tentando atingir o objetivo desejado, foi iniciada a análise exploratória. Inicialmente, quatro categorias principais foram estabelecidas: Goleiro, Zagueiros, Meio-Campistas e Atacantes.

Porém, após tais serem concluídas, foi necessário criar subcategorias, tais que mudam na vida real baseado na formação criada pelo técnico. Como cada formação mudaria a quantidade de jogadores em cada posição, foi necessário escolher uma formação única, para então aumentar de forma relevante a acurácia do classificador. Sendo o mais comum, a formação escolhida foi 4-3-3 (1 goleiro, 4 zagueiros, 3 meio-campistas e 3 atacantes).

Sabendo em quais categorias os jogadores poderiam se encaixar, um heat map foi feito, com a intenção de visualizar quais os melhores atributos de cada posição. O heat map aperfeiçoado pode ser visto abaixo.



Conseguindo separar visualmente os maiores atributos para cada posição, a correlação em si foi isolada, com o intuito de confirmar os melhores atributos para cada posição.

Goleiro:

GK diving	GK handling	GK kicking	GK positioning	GK reflexes
0.969429	0.964298	0.964298	0.968807	0.972755

-

Zagueiros:

Central-Back (CB)(x2):

CB	Aggression	Interceptions	Marking	Sliding tackle	Standing tackle
	0.748338	0.941310	0.94122	0.929456	0.9498

Left-Back (LB):

LB	Interceptions	Marking	Sliding tackle	Standing tackle
	0.895004	0.86694	0.877306	0.885623

Right-Back (RB):

RB	Interceptions	Marking	Sliding tackle	Standing tackle
	0.895004	0.86694	0.877306	0.885623

Meio-Campistas:

Centre Attacking Midfield (CAM):

CAM	Ball control	Crossing	Curve	Dribbling	Finishing	Free kick accuracy	Long shots
	0.921189	0.732099	0.7961	0.908719	0.772853	0.702388	0.831632

Positioning	Short passing	Shot power	Vision	Volleys

0.852649	0.802186	0.722553	0.88631	0.756350	

Centre Midfield (CM):

CAM	Ball control	Composure	Crossing	Curve	Dribbling	Long passing	Long shots
	0.873643	0.713268	0.73308	0.733	0.724634	0.841507	0.709441

Reactions	Short passing	Vision
0.732892	0.916952	0.841507

Centre Defensive Midfield (CDM) :

CDM	Aggression	Interceptions	Marking	Sliding tackle	Standing tackle
	0.711130	0.893164	0.844273	0.838502	0.866010

Atacantes:

Left Winger (LW):

LW	Agility	Ball control	Crossing	Curve	Dribbling	Finishing	Longshots
	0.697810	0.899655	0.771786	0.7958	0.93399	0.793891	0.816996

Positioning	Short passing	Shot power	Vision	Volleys

0.871166	0.725103	0.700408	0.83074	0.76134

Right Winger (RW):

RW	Agility	Ball control	Crossing	Curve	Dribbling	Finishing	Longshots
	0.697810	0.899655	0.771786	0.7958	0.93399	0.793891	0.816996

Positioning	Short passing	Shot power	Vision	Volleys
0.871166	0.725103	0.700408	0.83074	0.76134

Striker (ST):

ST	Ball control	Curve	Dribbling	Finishing	Long shots	Penalties	Positioning
	0.846925	0.72928	0.825582	0.881245	0.860435	0.729116	0.883531

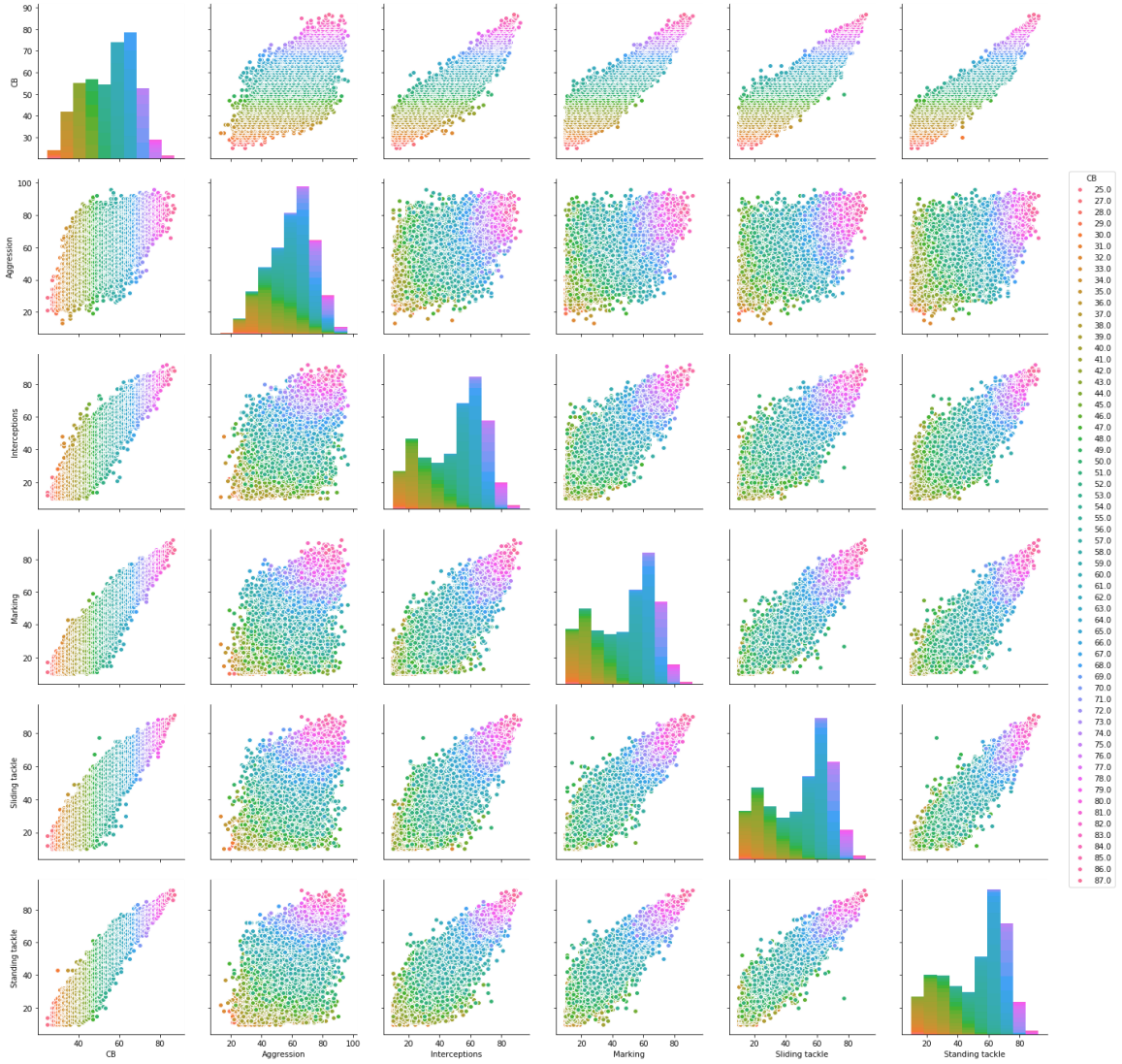
Shot power	Vision	Volleys
0.822876	0.75529	0.830853

Checando que todos os atributos escolhidos possuíam uma correlação mínima de $\simeq 0.7$ e uma média mínima dos melhores atributos de cada posição de $\simeq 0.8$, tais atributos foram

confirmados como os principais de cada posição, e mais um passo foi feito para o início do classificador.

Ainda assim, a análise numérica das correlações de cada posição demonstrou um erro: a incapacidade do sistema de diferenciar posições iguais porém de lados diferentes, especificamente LB e RB (Left Back e Right Back) e LW e RW (Left Winger e Right Winger). Ainda que um erro, tal foi aceitável, logo que tais posições exercem funções quase idênticas e não podem ser diferenciadas por dados (como por exemplo pé-dominante; este quesito é distribuído de forma quase igual dentre os "Wingers" e os zagueiros laterais. Por exemplo, existem quase um número igual de Left Wingers canhotos e destros).

Para ajudar na análise dos dados, plotamos os scatter plots de cada posição e seus principais atributos. Também no gráfico é possível visualizar as diferentes cores que correspondem a um maior reconhecimento a essa posição, por exemplo, 0.99 seria o jogador perfeito para essa posição, portanto quanto mais alto for, mais o jogador se encaixa na posição.



Parte 2: Categorização e classificador

Tendo conhecimento do funcionamento básico da relação atributo/posição, a metodologia de regressão logística foi escolhida, seguida pela utilização do RandomForest.

Regressão Logística

A metodologia principal do projeto envolveu a utilização de uma regressão logística, um método de classificação simples e ideal para os dados disponíveis e objetivo do trabalho como um todo. "A **regressão logística** é uma técnica estatística que tem como objetivo produzir, a partir de um conjunto de observações, um modelo que permita a predição de valores tomados por uma variável categórica, frequentemente binária, a partir de uma série de variáveis explicativas contínuas e/ou binárias."[1]

[1] https://pt.wikipedia.org/wiki/Regress%C3%A3o_log%C3%ADstica

Representando tal de forma matemática, a técnica implica que uma suposta variável X tem distribuição Bernoulli, e possui probabilidade de sucesso binária (ou seja, é possível duas respostas: X=1 como sucesso e X=0 como fracasso).

Além disso, um modelo logístico proporciona vantagens, tais como:

1. Facilidade para lidar com variáveis independentes categóricas
2. Fornecimento de resultados em termos de probabilidade
3. Facilidade de classificação de indivíduos em categorias
4. Não requer uma grande quantidade de suposições
5. Possui um alto grau de confiabilidade[2]

Assim sendo, uma regressão logística foi a escolha ideal para ser utilizada neste trabalho.

Mas como a regressão logística pode ser aplicada em Python?

Para a implementação em Python é necessário dividir o DataFrame em duas partes. Chamamos de Treino e Teste(Tudo aleatório). Definindo uma função de classificação genérica, leva um modelo como entrada e determina a porcentagem de precisão. O código faz previsões nos dados da parte Treino e então partimos para a regressão logística que fará a previsão das posições.

[2] https://edisciplinas.usp.br/pluginfile.php/3769787/mod_resource/content/1/09_RegressaoLogistica.pdf

```
Y = fifa18_analiseGeral["Posição"]
X = fifa18_analiseGeral
del X["Posição"]
```

```
RANDOM_SEED = 5

np.random.seed(RANDOM_SEED)


from sklearn.model_selection import train_test_split

X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.33, random_state=RANDOM_SEED)
```

```
X_train.shape, Y_train.shape, X_test.shape, Y_test.shape
```

RandomForest

RandomForest é um algoritmo de aprendizagem que, a partir da combinação de "árvores de decisão", aumenta o resultado geral. Desta forma, o algoritmo adiciona aleatoriedade ao modelo, e tal "diversidade" amplia a acurácia, precisão e estabilidade da previsão do modelo em geral.

Para um modelo como o explorado neste trabalho, a utilidade do RandomForest envolve sua facilidade para medir a significância relativa de cada atributo, conseguindo "medir" sua importância na previsão final como um todo. Uma vez que atributos "menos importantes" são identificados, tal permite a diminuição na quantidade de atributos, que resultam em uma maior probabilidade de *overfitting* (ajuste realizado pelo programa para atender as características de cada atributo).[3]

O Random Forest usa a mesma base de Teste e Treino usada na Regressão logística.

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score

model = RandomForestClassifier()
model.fit(X_train, Y_train)
Y_pred = model.predict(X_test)
print('Acurácia: {}'.format(accuracy_score(Y_test, Y_pred)))
```

Resultados e conclusão:

Pensando na questão de resultados, tais podem ser vistos abaixo.

[3] https://medium.com/machina-sapiens/o-algoritmo-da-floresta-aleat%C3%B3ria-3545f6babdf8

col_0	CAM	CB	CDM	CM	LB	LW	LWB	RB	RW	RWB	ST
Posição											
CAM	91	1	36	73	2	73	0	1	67	0	49
CB	1	778	28	35	40	0	1	30	1	0	0
CDM	26	75	148	164	19	5	0	9	3	0	1
CM	52	43	165	297	15	29	0	8	21	0	5
LB	2	86	24	27	161	15	1	94	4	1	0
LW	65	2	25	58	43	145	0	16	133	0	95
LWB	0	4	3	3	14	5	0	6	1	0	0
RB	0	90	23	21	122	16	2	100	10	0	1
RW	57	8	26	60	32	152	0	18	150	0	94
RWB	1	3	3	3	13	1	0	10	3	0	0
ST	30	2	6	14	0	65	0	2	65	0	637

Na maioria dos casos, o programa foi capaz de julgar corretamente a posição do jogador, como pode ser visto nas relações diretas entre as próprias posições (sempre é o maior número daquela posição). Ainda assim, houveram erros, que podem ser classificados da seguinte maneira:

> **Erros leves:** erros leves incluem classificações não na posição e lado exato, mas somente na posição (por exemplo um meio-campista Centre Attacking Midfield foi confundido com um meio campista Centre Midfield). Este foi o segundo caso que mais ocorreu durante a classificação.
>
> **Erros médios:** erros médios foram os terceiros mais comum. Este incluía erros de posição, mas de posições em que os atributos principais eram parecidos. Um exemplo é CAM (Centre Attacking Midfield com LW, Left Winger).
>
> **Erros graves:** erros graves foram os erros mais incomuns, e que dificilmente ocorreram. Estes incluíam erros completo de posição, como a classificação de um zagueiro (CB por exemplo) com um atacante (LW por exemplo)

Tendo tal análise em mente, o projeto pode ser considerado bem sucedido, e capaz de cumprir a função desejada. Por mais que a acurácia final atingida foi de 47%, esta indica somente os acertos diretos, sem incluir erros leves e médios que são aceitáveis uma vez que não afetariam o time de forma drástica. Considerando erros leves como acerto, o programa teria uma acurácia de ≃ 92%, e considerando erros leves e médios, a acurácia seria de ≃ 99%. Assim sendo, o programa compriu seu propósito de forma muito efetiva, e o objetivo de criar um classificador automático de times foi gerado.

Fontes:

https://pt.wikipedia.org/wiki/Regress%C3%A3o_log%C3%ADstica
https://edisciplinas.usp.br/pluginfile.php/3769787/mod_resource/content/1/09_RegressaoLogistica.pdf

https://medium.com/machina-sapiens/o-algoritmo-da-floresta-aleat%C3%B3ria-3545f6babdf8
http://www.portalaction.com.br/analise-de-regressao/411-modelo-estatistico
https://www.kaggle.com/thec03u5/fifa-18-demo-player-dataset
https://scikit-learn.org/stable/auto_examples/linear_model/plot_sparse_logistic_regression_20newsgroups.html#sphx-glr-auto-examples-linear-model-plot-sparse-logistic-regression-20newsgroups-py