Aplicação do Trabalho do Pagerank

ERICK BRITO
B41254@fgv.edu.br

GERMANO ANDRADE
B41314@fgv.edu.br

23 de Novembro de 2020

5 Aplicação em uma base de dados

A base escolhida trata sobre o Campeonato Brasileiro de Futebol das temporadas de 2013 a 2020. Ela contém todas as partidas realizadas e, dentre outros dados, os nomes dos times, data da partida, ano, total de gols de cada time, vencedor da partida, etc.

[1]:
```
#Importando pacotes necessários
import pandas as pd
import numpy as np
```

[2]:
```
#Importando base de dados
br = pd.read_csv("../Databases/BRA.csv")
```

[3]:
```
#Criando uma função para filtrar a base por ano (temporada) e turno.
#Também escolhemos algumas colunas mais interessantes para a gente.

def filtrar_base(ano, turno):
    filtro = br["Season"] == ano
    if turno == 1:
        return br[filtro][["Home", "Away", "HG", "AG", "Res"]].
 ↪head(190)
    elif turno == 2:
```

```
        return br[filtro][["Home", "Away", "HG", "AG", "Res"]].
↪tail(190)
```

Agora vamos, filtrar a base apenas pelo primeiro turno do ano de 2017

[4]:
```
br = filtrar_base(2017, 1)
```

e ficamos com

[5]:
```
br.head()
```

[5]:

	Home	Away	HG	AG	Res
1900	Flamengo RJ	Atletico-MG	1.0	1.0	D
1901	Corinthians	Chapecoense-SC	1.0	1.0	D
1902	Fluminense	Santos	3.0	2.0	H
1903	Avai	Vitoria	0.0	0.0	D
1904	Bahia	Atletico-PR	6.0	2.0	H

Nesse momento, estamos rumo à criação da matriz.
Primeiramente, vamos criar um *DataFrame* apenas com as colunas de ***Winner*** e ***Loser***.

[6]:
```
winner = []
loser = []

for index, row in br.iterrows():
        if row['Res'] == "H":
                winner.append(br.loc[index, 'Home'])
                loser.append(br.loc[index, 'Away'])
        elif row['Res'] == "A":
                winner.append(br.loc[index, 'Away'])
                loser.append(br.loc[index, 'Home'])

        elif row["Res"] == "D":
                winner.append(br.loc[index, 'Home'])
                loser.append(br.loc[index, 'Away'])

                winner.append(br.loc[index, 'Away'])
                loser.append(br.loc[index, 'Home'])
```

Feito isso, temos as colunas da nossa tabela apenas com os vencedores e perdedores em cada coluna.

[7]:
```
#Criando DataFrame
w_l = pd.DataFrame({"Winner": winner, "Loser": loser})

w_l.head()
```

[7]:

	Winner	Loser
0	Flamengo RJ	Atletico-MG
1	Atletico-MG	Flamengo RJ
2	Corinthians	Chapecoense-SC
3	Chapecoense-SC	Corinthians
4	Fluminense	Santos

Nesse ponto, estamos começando a desenhar a nossa matriz de transição

[8]:
```
# 'index' se refere aos times distintos que participaram dessa␣
 ↪edição do brasileirão.
# ela já está em ordem alfabética
index = sorted(w_l["Winner"].unique())

#Criação do DataFrame que servirá como matriz de transição
matriz = pd.DataFrame(index=index, columns=index)

#Mostrando uma submatriz 5x5 com apenas as primeiras 5 linhas e 5␣
 ↪colunas da matriz.
matriz.head()[matriz.columns[:5]]
```

[8]:

	Atletico GO	Atletico-MG	Atletico-PR	Avai	Bahia
Atletico GO	NaN	NaN	NaN	NaN	NaN
Atletico-MG	NaN	NaN	NaN	NaN	NaN
Atletico-PR	NaN	NaN	NaN	NaN	NaN
Avai	NaN	NaN	NaN	NaN	NaN
Bahia	NaN	NaN	NaN	NaN	NaN

Criando uma função que verifica se o time da coluna perdeu para o time que está a linha.
Caso isso ocorra, o **NaN** é substituido pelo número 1.

[9]:
```
def verifica_time(valor, coluna):

        for cada_indice in w_l.index:
                if w_l.loc[cada_indice, "Loser"] == coluna and w_l.
 ↪loc[
                    cada_indice, "Winner"] == valor.name:
                        return 1

#Criando matriz de transição apenas com 1 e 0
transicao = pd.DataFrame()
for i in matriz:
        transicao[i] = matriz.apply(verifica_time, args=(i, )).
 ↪fillna(0)

transicao.head()[transicao.columns[:5]]
```

[9]:

	Atletico GO	Atletico-MG	Atletico-PR	Avai	Bahia
Atletico GO	0.0	0.0	0.0	1.0	0.0
Atletico-MG	1.0	0.0	0.0	1.0	0.0
Atletico-PR	1.0	1.0	0.0	1.0	0.0
Avai	0.0	0.0	0.0	0.0	1.0
Bahia	1.0	1.0	1.0	1.0	0.0

Nesse ponto, podemos finalizar nossa matriz **A**, dividindo cada coluna pela sua soma

[10]:
```
finalizada = pd.DataFrame()
for i in transicao:
        finalizada[i] = transicao[i] / sum(transicao[i])

finalizada.head()[finalizada.columns[:5]].round(3)
```

[10]:

	Atletico GO	Atletico-MG	Atletico-PR	Avai	Bahia
Atletico GO	0.000	0.000	0.000	0.067	0.000
Atletico-MG	0.062	0.000	0.000	0.067	0.000
Atletico-PR	0.062	0.077	0.000	0.067	0.000
Avai	0.000	0.000	0.000	0.000	0.077
Bahia	0.062	0.077	0.083	0.067	0.000

Finalmente, podemos transformar o *DataFrame* para um *array* **numpy**, ou seja, transformá-la em um formato de matriz como estamos habituados.

[11]:
```
A = finalizada.to_numpy()
```

Como visto, para melhores resultados, podemos calcular a matriz $M = (1-p) \cdot A + p \cdot B$, onde $B = \frac{1}{n} \cdot \begin{bmatrix} 1 & \dots & 1 \\ \vdots & \ddots & \vdots \\ 1 & \dots & 1 \end{bmatrix}$ e p é escolhido igual a 0.15.

[12]:
```
p = 0.15
n = A.shape[0]

B = (1/n)*np.ones((n, n))
M = (1-p)*A + p*B
```

[13]:
```
#Essa função retorna uma tupla contendo dois `np.arrays`, onde o
#primeiro se refere aos autovalores e o segundo, aos autovetores␣
 ↪correspondentes.
w, v = np.linalg.eig(M)

#Autovetor correspondente ao autovalor 1
vetor_pagerank = v[:, 0].real

#Tabela contendo o nome dos times e o pagerank de cada um.
pagerank = pd.DataFrame({
    "Times": index,
    "Pagerank": np.absolute(vetor_pagerank)
})

#Ordenando a tabela de acordo com o pagerank
pagerank.sort_values(by="Pagerank", ascending=False, inplace=True)

pagerank
```

[13]:

	Times	Pagerank
7	Corinthians	0.329069
10	Flamengo RJ	0.283475

15	Santos	0.254503
12	Gremio	0.247502
3	Avai	0.246925
9	Cruzeiro	0.245660
2	Atletico-PR	0.238486
11	Fluminense	0.237158
17	Sport Recife	0.228049
8	Coritiba	0.218695
6	Chapecoense-SC	0.216743
5	Botafogo RJ	0.216452
13	Palmeiras	0.210694
14	Ponte Preta	0.203577
1	Atletico-MG	0.196003
4	Bahia	0.179209
18	Vasco	0.173636
16	Sao Paulo	0.172670
19	Vitoria	0.160491
0	Atletico GO	0.123012

Agora, filtrando pelo ano de 2018:

[14]:
```
br = filtrar_base(2018, 1)
```

[14]:

	Times	Pagerank
11	Gremio	0.289166
13	Palmeiras	0.287783
16	Sao Paulo	0.285372
12	Internacional	0.274504
9	Flamengo RJ	0.256528
6	Chapecoense-SC	0.245159
10	Fluminense	0.218257
1	Atletico-MG	0.212538
3	Bahia	0.211568
17	Sport Recife	0.209950
4	Botafogo RJ	0.207756
8	Cruzeiro	0.206139
2	Atletico-PR	0.200230
5	Ceara	0.200108
18	Vasco	0.196409
7	Corinthians	0.192012
0	America MG	0.184698
14	Parana	0.184136
15	Santos	0.182909
19	Vitoria	0.162806

E então, pelo ano de 2019:

[15]:
```
br = filtrar_base(2019, 1)
```

[15]:

	Times	Pagerank
8	Corinthians	0.296163
16	Palmeiras	0.294798
18	Sao Paulo	0.293707
3	Bahia	0.277608
10	Flamengo RJ	0.277288
15	Internacional	0.263889
14	Gremio	0.249776
17	Santos	0.247148
19	Vasco	0.206318
9	Cruzeiro	0.193982
5	CSA	0.193178
1	Atletico-MG	0.190953
6	Ceara	0.188077
0	Athletico-PR	0.186296
11	Fluminense	0.180202
12	Fortaleza	0.171392
4	Botafogo RJ	0.169045
2	Avai	0.166602
13	Goias	0.161079
7	Chapecoense-SC	0.154503