PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS
NÚCLEO DE EDUCAÇÃO A DISTÂNCIA
Pós-graduação *Lato Sensu* em Ciência de Dados e Big Data

Lucas Marchon de Souza Lima
Marcelio Antonio Prado dos Santos

TÍTULO DO PROJETO

Análises e Co-relações entre o Cartola FC e o Mundo Real

Rio de Janeiro
2019

Lucas Marchon de Souza Lima
Marcelio Antonio Prado dos Santos

TÍTULO DO PROJETO

Análises e Co-relações entre o Cartola FC e o Mundo Real

Trabalho de Conclusão de Curso apresentado ao Curso de Especialização em Ciência de Dados e Big Data como requisito parcial à obtenção do título de especialista.

Rio de Janeiro
2019

SUMÁRIO

1. Introdução

1.1. Contextualização

Nosso TCC é focado em analisar e fazer co-relações com algumas métricas dos jogadores do CartolaFC e o desempenho dos mesmos no Campeonato Brasileiro de Futebol.

O CartolaFC é um fantasy game criado pela Globo.com no qual os cartoleiros (como são conhecidas as pessoas que jogam o game) montam seus times com jogadores de futebol inscritos oficialmente na Série A do Campeonato Brasileiro.

O game utiliza scouts baseados nos scouts oficiais da CBF. Isso significa que o game se baseia na súmula dos jogos para computar suas pontuações. Porém, alguns scouts são passíveis de interpretação, exemplo, roubadas de bola e defesas difíceis. Essas métricas são computadas “manualmente” pelos scouters que trabalham na Globo.com.

Nosso principal objetivo é analisar algumas variáveis de performance dos jogadores no game e verificarmos se as mesmas estão relacionadas com a desempenho do atleta na Série A do Campeonato Brasileiro de Futebol.

1.2. O problema proposto

Analisar os dados de um jogo de futebol tem como princial objetivo melhorar a performance dos atletas, imaginar um eficniente esquema tático para a equipe e contribuir para que o treinador possa tomar boas decisões.

As pranchetas e planilhas (muitas vezes feitas à mão) com alguns dados dos jogos, estão dando lugar a softwares que em poucos segundos oferecem milhões de informações, as quais uma pessoa levaria muitos dias para calcular. Essa tecnologia benéfica tem chamado cada vez mais a atenção dos profissionais da área esportiva, que têm interesse em aprender como ocorre a análise de desempenho no futebol.

Levamos em consideração os dados do Cartola FC para analisar o desempenho dos atletas, visto que o game utiliza os scouts das súmulas das partidas fornecidos pela CBF. Esses dados são praticamente os mesmos dados que uma equipe profissional possui de seus jogadores. Claramente uma equipe profissional possui diversos outros tipos de dados com relação às performances dos atletas e do time como um todo, mas os principais dados (finalização à gol, impedimentos, assistências, faltas sofridas e etc...) são os mesmos que o Cartola FC possui.

O escopo dos dados utilizados para análise compreende todas as métricas dos jogadores entre os anos de 2014 e 2017. Com base nesses dados, vamos aplicar os conceitos relacionados aos tratamentos dos dados, machine learning e exploraração dos dados. Apresentaremos os dados através de dashboards sendo possível analisar e tomar decisões em cima de dados agregados e/ou filtrados.

Através dos dashboards e do modelo de machine learning será possível entender melhor o desempenho do atleta e até pensar em montagem e/ou reposição para o elenco.

2. Coleta de Dados

Os dados que foram utilizados para o desenvolvimento do trabalho foram separados em datasets:

- atletas
    - 2014_jogadores.csv
    - 2015_jogadores.csv
    - 2016_jogadores.csv
    - 2017_jogadores.csv
- scouts
    - base_bruta_scouts.csv
- times
    - times_ids.csv

A seguir, os detalhamentos das origens dos dados, formatos e relacionamentos. Os dados dos atletas, scouts e times foram obtidos no GitHub pelo link: https://github.com/henriquepgomide/caRtola

As bases desse link foram extraídas diretamente da API do Cartola FC:

```
https://api.cartolafc.globo.com/atletas/pontuados
https://api.cartolafc.globo.com/partidas
https://api.cartolafc.globo.com/clubes
```

Todas as bases foram disponibilizadas em formato .csv usando a vírgula como separador. Abaixo segue o dicionário para os datasets atletas, times e scouts:

ATLETAS:

2014_jogadores.csv
2015_jogadores.csv
2016_jogadores.csv
2017_jogadores.csv

Nome do campo	Descrição	Tipo
ID	ID único do atleta	Int
Apelido	Nome/Apelido do atleta	String
ClubeID	ID único do clube que o atleta pertence	Int
PosicaoID	ID único da posição que o atleta joga	Int

TIMES:

times_ids.csv

Nome do campo	Descrição	Tipo
nome.cbf	Nome do clube na CBF	String
nome.cartola	Nome do clube no Cartola FC	String
nome.completo	Nome completo do Clube	String
cod.older	ID do clube em anos anteriores	Int
cod.2017	ID do clube no ano de 2017	Int
cod.2018	ID do clube no ano de 2018	Int

id	ID final do clube	Int
abreviacao	Sigla do clube. Ex: FLA, FLU, BOT	String
escudos.60x60	Link do repositório com a imagem do escudo do clube no tamanho 60x60	String
escudos.45x45	Link do repositório com a imagem do escudo do clube no tamanho 45x45	String
escudos.30x30	Link do repositório com a imagem do escudo do clube no tamanho 30x30	String

SCOUT:

2017_dados_agregados.csv (nome original) depois renomeamos o arquivo para base_bruta_scouts.csv

Nome do campo	Descrição	Tipo
AtletaID	id do jogador	Int
Rodada	número da rodada do Brasileirão	Int
ClubeID	clube do jogador	Int
Participou	indica se o jogador participou daquela rodada	Booleano
Posicao	posição do jogador	String
Jogos	qtde. de jogos que o jogador participou até aquela rodada	Int
Pontos	pontuação do jogador	Float
PontosMedia	média da pontuação do jogador	Flaot
Preco	preço do jogador	Float
PrecoVariacao	variação de preço	Float
FS	faltas sofridas	Int
PE	passes errados	Int
A	assistências	Int
FT	finalizações na trave	Int

FD	finalizações defendidas	Int
FF	finalizações para fora	Int
G	gols	Int
I	impedimentos	Int
PP	pênaltis perdidos	Int
RB	roubadas de bola	Int
FC	faltas cometidas	Int
GC	gols contra	Int
CA	cartões amarelo	Int
CV	cartões vermelho	Int
SG	jogos sem sofrer gols	Int
DD	defesas difíceis	Int
DP	defesas de pênalti	Int
GS	gols sofridos	Int
ano	ano dos dados	Int
Apelido	nome/apelido do jogador	String
Status	status do jogador	String
avg.Points	média de pontos do jogador	Float
avg.last05	média de pontos do jogador nas últimas 5 rodadas	Float
avg.FS	média de faltas sofridas	Float
avg.FS.l05	média de faltas sofridas nas últimas 5 rodadas	Float
avg.PE	média de passes errados	Float
avg.PE.l05	média de passes errados nas últimas 5 rodadas	Float
avg.A	média de assistências	Float
avg.A.l05	média de assistências nas últimas 5 rodadas	Float
avg.FT	média de finalizações na trave	Float

avg.FT.l05	média de finalizações na trave nas últimas 5 rodadas	Float
avg.FD	média de finalizações defendidas	Float
avg.FD.l05	média de finalizações defendidas nas últimas 5 rodadas	Float
avg.FF	média de finalizações para fora	Float
avg.FF.l05	média de finalizações para fora nas últimas 5 rodadas	Float
avg.G	média de gols	Float
avg.G.l05	média de gols nas últimas 5 rodadas	Float
avg.I	média de impedimentos	Float
avg.I.l05	média de impedimentos nas últimas 5 rodadas	Float
avg.PP	média de pênaltis perdidos	Float
avg.PP.l05	média de pênaltis perdidos nas últimas 5 rodadas	Float
avg.RB	média de roubadas de bola	Float
avg.RB.l05	média de roubadas de bola nas últimas 5 rodadas	Float
avg.FC	média de faltas cometidas	Float
avg.FC.l05	média de faltas cometidas nas últimas 5 rodadas	Float
avg.GC	média de gols contra	Float
avg.GC.l05	média de gols contra nas últimas 5 rodadas	Float
avg.CA	média de cartões amarelos	Float
avg.CV.l05	média de cartões vermelhos nas últimas 5 rodadas	Float
avg.SG	média de jogos sem sofrer gols	Float
avg.SG.l05	média de jogos sem sofrer gols nas últimas 5 rodadas	Float
avg.DD	média de defesas difíceis	Float
avg.DD.l05	média de defesas difíceis nas últimas 5 rodadas	Float

avg.DP	média de defesas de pênalti	Float
avg.DP.l05	média de defesas de pênalti nas últimas 5 rodadas	Float
avg.GS	média de gols sofridos	Float
avg.GS.l05	média de gols sofridos nas últimas 5 rodadas	Float
risk_points	desvio-padrão da pontuação do jogador	Float
mes	mês que a partida ocorreu	Int
dia	dia que a partida ocorreu	Int
home.score.x	placar to time visitante	Int
away.score.x	placar do time da casa	Int
pred.home.score	estimativa de força de ataque do time do jogador	Float
pred.away.score	estimativa de força de defesa do time do jogador	Float
home.attack	estimativa de gols para o time da casa	Float
home.defend	estimativa de gols para o time visitante	Float
variable	indica se o jogador é do time da casa ou visitante	String

Após a coleta das bases mencionadas acima, montamos outras três bases para cruzarmos informações relevantes para as análises. A base a seguir é a base composta pelos atletas que formaram a seleção do Cartola FC no seu respectivo ano. Seguem links abaixo:

Seleção do Cartola 2014:

http://globoesporte.globo.com/cartola-fc/noticia/2014/12/cartola-14-conca-e-maior-pontuador-e-daniel-tem-recorde-numa-so-rodada.html

Seleção do Cartola 2015:
http://globoesporte.globo.com/cartola-fc/noticia/2015/12/cartolao-confira-todos-os-principais-dados-do-game-do-brasileirao-2015.html

Seleção do Cartola 2016:

http://globoesporte.globo.com/cartola-fc/noticia/2016/12/cartolao-2016-veja-os-principais-dados-do-game-no-ano-marinho-se-destaca.html

Seleção do Cartola 2017:

https://globoesporte.globo.com/cartola-fc/noticia/com-a-maior-media-geral-hernanes-comanda-selecao-do-cartola-2017-confira.ghtml

SELEÇÕES DO CARTOLA

selecoes_cartola_2014_2017

Nome do campo	Descrição	Tipo
id_atleta	ID único do atleta	Int
atleta	Nome do atleta	String
id_clube	ID único do clube que o atleta pertence	Int
clube	Nome do clube que o atleta pertence	String
id_posicao	ID único da posição que o atleta joga	Int
posicao	Nome da posição que o atleta joga	String
ano	Partição com a informação do ano em que o atleta fez parte da seleção do CartolaFC.	Int

Outra base que precisamos montar, foi a base composta pelos atletas que formaram a seleção do Campeonato Brasileiro Série A no seu respectivo ano (eleições realizadas por jornalistas, atletas, técnicos e a própria CBF). Segue link abaixo:

https://globoesporte.globo.com/futebol/brasileirao-serie-a/noticia/selecao-do-brasileirao-relembre-todas-as-premiacoes-da-historia-da-competicao.ghtml

SELEÇÕES DO CAMPEONATO BRASILEIRO

selecoes_brasileiro_2014_2017

Nome do campo	Descrição	Tipo
craque_campeonato	Indica se o atleta foi eleito o craque do campeonato. 1 = Craque, 0 = Não foi o craque	Int
id_atleta	ID único do atleta	Int
atleta	Nome do atleta	String
clube	Nome do clube que o atleta pertence	String
id_posicao	ID único da posição que o atleta joga	Int
posicao	Nome da posição que o atleta joga	String
ano	Partição com a informação do ano em que o atleta fez parte da seleção do CartolaFC.	Int

Por fim, montamos a base com a classificação dos clubes no Campeonato Brasileiro Série A. Segue link abaixo:

https://www.google.com/search?q=classifica%C3%A7%C3%A3o+campeonato+brasileiro&rlz=1C1GCEB_enBR843BR843&oq=classifica%C3%A7%C3%A3o+campeonato+brasileiro&aqs=chrome.0.0l2j69i59j69i60l3.5281j0j4&sourceid=chrome&ie=UTF-8#sie=lg;/g/11fhwkn08h;2;/m/0fnk7q;st;fp;1;;

CLASSIFICAÇÃO DO CAMPEONATO BRASILEIRO SÉRIE A

brasileiros_2014_2017

Nome do campo	Descrição	Tipo
classificacao	Posição do clube na tabela de classificação	Int
clube	Nome do clube que o atleta pertence	String
pontos	Total de pontos que o clube fez no campeonato	Int
ano	Ano em que o clube atingiu a classificação e pontuação.	Int

3. Processamento/Tratamento de Dados

Como base para nosso projeto, utilizamos diversos arquivos com informações sobre o cartola, entre o período de 2014 e 2017. Informações sobre os atletas, scouts, times e posições. Utilizamos o Pyspark para a leitura e tratamento das bases.

Importação das bibliotecas e configurações das variáveis dos diretórios do projeto.

```
from pyspark import *
from pyspark.sql import SparkSession
from pyspark.sql.functions import *
import datetime as dt
import pandas as pd

sc = SparkContext()
sc.setLogLevel("ERROR")
conf = SparkConf().setAppName('Cartola')
spark = SparkSession.builder.config(conf=conf).enableHiveSupport().getOrCreate()

# Diretórios
base = '/home/marcelio/Git/test_local/cartola'
base_atleta = base + '/bases/atletas'
base_scout = base + '/bases/scouts'
base_time = base + '/bases/times'
base_atleta = base + '/bases/atletas'
base_posicao = base + '/bases/posicao'
base_tmp = base + '/bases/tmp'
```

Efetuamos a leitura do arquivo de scouts, removemos as colunas que não eram necessárias, renomeamos/padronizamos as colunas do dataframe, tratamos as descrições de posições. Temos um total de 43.821 registros.

```
##########
# Scouts
##########
try:
    df_scouts = spark.read.csv(base_scout + '/base_bruta_scouts.csv', header=True, sep=",", \
                               encoding="ISO-8859-1", inferSchema=False)

    df_scouts = df_scouts.drop('Participou','Status','avg.Points','avg.last05','avg.FS','avg.FS.l05','avg.PE',
                               'avg.PE.l05','avg.A','avg.A.l05','avg.FT','avg.FT.l05','avg.FD','avg.FD.l05','avg.FF',
                               'avg.FF.l05','avg.G','avg.G.l05','avg.I','avg.I.l05','avg.PP','avg.PP.l05','avg.RB',
                               'avg.RB.l05','avg.FC','avg.FC.l05','avg.GC','avg.GC.l05','avg.CA','avg.CV.l05',
                               'avg.SG','avg.SG.l05','avg.DD','avg.DD.l05','avg.DP','avg.DP.l05','avg.GS',
                               'avg.GS.l05','risk_points','mes','dia','home.score.x','away.score.x','pred.home.score'
                               ,'pred.away.score','home.attack','home.defend','variable', 'Apelido')

    df_scouts = df_scouts.withColumnRenamed('AtletaID','id_atleta').withColumnRenamed('Rodada','rodada') \
        .withColumnRenamed('ClubeID','id_clube').withColumnRenamed('Posicao','posicao') \
        .withColumnRenamed('Pontos','pontos').withColumnRenamed('PontosMedia','pontos_media') \
        .withColumnRenamed('Preco','preco').withColumnRenamed('PrecoVariacao','preco_variacao') \
        .withColumnRenamed('FS','falta_sofrida').withColumnRenamed('PE','passe_errado') \
        .withColumnRenamed('A','assistencia').withColumnRenamed('FT','finalizacao_trave') \
        .withColumnRenamed('FD','finalizacao_defendida') \
        .withColumnRenamed('FF','finalizacao_fora').withColumnRenamed('G','gol').withColumnRenamed('I','impedimento') \
        .withColumnRenamed('PP','penalti_perdido').withColumnRenamed('RB','roubada_bola') \
        .withColumnRenamed('FC','falta_cometida').withColumnRenamed('GC','gol_contra') \
        .withColumnRenamed('CA','cartao_amarelo').withColumnRenamed('CV','cartao_vermelho') \
        .withColumnRenamed('SG','jogo_sem_sofrer_gol').withColumnRenamed('DD','defesa_dificil') \
        .withColumnRenamed('DP','defesa_penalti').withColumnRenamed('GS','gol_sofrido')
```

```
52
53      df_scouts = df_scouts.withColumn('id_atleta', col('id_atleta').cast('int')) \
54          .withColumn('rodada', col('rodada').cast('int')) \
55          .withColumn('pontos_media', col('pontos_media').cast('float')) \
56          .withColumn('preco', col('preco').cast('float')) \
57          .withColumn('preco_variacao', col('preco_variacao').cast('float')) \
58          .withColumn('falta_sofrida', col('falta_sofrida').cast('int')) \
59          .withColumn('passe_errado', col('passe_errado').cast('int')). \
60          withColumn('assistencia', col('assistencia').cast('int')) \
61          .withColumn('finalizacao_trave', col('finalizacao_trave').cast('int')) \
62          .withColumn('finalizacao_defendida', col('finalizacao_defendida').cast('int')) \
63          .withColumn('finalizacao_fora', col('finalizacao_fora').cast('int')) \
64          .withColumn('gol', col('gol').cast('int')) \
65          .withColumn('impedimento', col('impedimento').cast('int')) \
66          .withColumn('penalti_perdido', col('penalti_perdido').cast('int')) \
67          .withColumn('roubada_bola', col('roubada_bola').cast('int')) \
68          .withColumn('falta_cometida', col('falta_cometida').cast('int')) \
69          .withColumn('gol_contra', col('gol_contra').cast('int')) \
70          .withColumn('cartao_amarelo', col('cartao_amarelo').cast('int')) \
71          .withColumn('cartao_vermelho', col('cartao_vermelho').cast('int')) \
72          .withColumn('jogo_sem_sofrer_gol', col('jogo_sem_sofrer_gol').cast('int')) \
73          .withColumn('defesa_dificil', col('defesa_dificil').cast('int')) \
74          .withColumn('defesa_penalti', col('defesa_penalti').cast('int')) \
75          .withColumn('gol_sofrido', col('gol_sofrido').cast('int')) \
76          .withColumn('ano', col('ano').cast('int')) \
77          .withColumn('id_clube', col('id_clube').cast('int'))
78
79      df_scouts = df_scouts.withColumn('posicao', \
80                           when(col('posicao') == 'ata','Atacante') \
81                          .when(col('posicao') == 'gol','Goleiro') \
82                          .when(col('posicao') == 'lat','Lateral') \
83                          .when(col('posicao') == 'mei','Meia') \
84                          .when(col('posicao') == 'tec','Tecnico') \
85                          .when(col('posicao') == 'zag','Zagueiro'))
86
87      print('\n*** SCOUTS ***')
88      print('Total de registros na base de scouts: {}'.format(df_scouts.count()))
89
90      df_scouts = df_scouts.distinct()
91      print('\nTotal de registros distintos na base de scouts: {}'.format(df_scouts.count()))
92
93      df_scouts.printSchema()
94      df_scouts.show(50)
95
96  except:
97      print('Erro no processo Scouts: {}'.format(sys.exc_info()[0]))
98      raise
99  else:
100     print('Processo Scouts executado com sucesso !\n')
```

Log:

*** SCOUTS ***

Total de registros na base de scouts: 43821

Total de registros distintos na base de scouts: 43821

Para a base de times, excluímos as colunas que não precisamos, renomeamos algumas colunas no dataframe, tratamos o id do clube, para casos de alguns clubes que trocaram de nome e/ou código em anos diferentes e convertemos a coluna chave para inteiro. Com o tratamento dos times, o total de registros foi de 47 para 43.

```
#########
# Times
#########
try:

    df_times = spark.read.csv(base_time + '/times_ids.csv', header=True, sep=",", encoding="ISO-8859-1", \
                              inferSchema=False)

    df_times = df_times.drop('nome.cbf').drop('nome.completo').drop('cod.older').drop('cod.2017') \
               .drop('cod.2018').drop('abreviacao').drop('escudos.60x60').drop('escudos.45x45').drop('escudos.30x30')

    df_times = df_times.withColumnRenamed('nome.cartola','clube').withColumnRenamed('id','id_clube')

    print('\n*** TIMES ***')
    print('Total de registros na base de times: {}'.format(df_times.count()))
    df_times = df_times.withColumn('clube', \
                                   when(col('id_clube') == '293','Atlhetico-PR') \
                                   .when(col('id_clube') == '354', 'Ceará') \
                                   .otherwise(col('clube') )
                                  )

    df_times = df_times.distinct()

    print('Total de registros distintos na base de times: {}'.format(df_times.count()))

    df_times = df_times.withColumn('id_clube', col('id_clube').cast('int'))
    df_times = df_times.distinct()

    df_times.printSchema()
    df_times.show()

except:
    print('Erro no processo Times: {}'.format(sys.exc_info()[0]))
    raise
else:
    print('Processo Times executado com sucesso !\n')
```

Log:

*** TIMES ***

Total de registros na base de times: 47

Total de registros distintos na base de times: 43

Com isso concluímos o tratamento da base_times e o dataset ficou da seguinte maneira:

base_times

Nome do campo	Descrição	Tipo
id_clube	ID único do clube que o atleta pertence	Int
clube	Nome do clube que o atleta pertence	String

No tratamento da base de atletas, juntamos as quatro bases de atletas, convertemos alguns tipos que seriam utilizados posteriormente como chaves, criamos o campo posição de acordo com seu respectivo id, criamos um dataframe temporário para fazer um join mais a frente com a tabela de scouts, inserimos os respectivos anos e reordenamos dataframe.

```
##########
# Atletas
##########
try:
    df_atletas_2014 = spark.read.csv(base_atleta + '/2014_jogadores.csv', header=True, sep=",", \
                                     encoding="ISO-8859-1", inferSchema=False)
    df_atletas_2015 = spark.read.csv(base_atleta + '/2015_jogadores.csv', header=True, sep=",", \
                                     encoding="ISO-8859-1", inferSchema=False)
    df_atletas_2016 = spark.read.csv(base_atleta + '/2016_jogadores.csv', header=True, sep=",", \
                                     encoding="ISO-8859-1", inferSchema=False)
    df_atletas_2017 = spark.read.csv(base_atleta + '/2017_jogadores.csv', header=True, sep=",", \
                                     encoding="ISO-8859-1", inferSchema=False)

    df_atletas_2014 = df_atletas_2014.withColumnRenamed('ID','id_atleta') \
                                     .withColumnRenamed('Apelido', 'atleta') \
                                     .withColumnRenamed('ClubeID', 'id_clube') \
                                     .withColumnRenamed('PosicaoID','id_posicao')


    df_atletas_2015 = df_atletas_2015.withColumnRenamed('ID','id_atleta') \
                                     .withColumnRenamed('Apelido', 'atleta') \
                                     .withColumnRenamed('ClubeID', 'id_clube') \
                                     .withColumnRenamed('PosicaoID','id_posicao')

    df_atletas_2016 = df_atletas_2016.withColumnRenamed('ID','id_atleta') \
                                     .withColumnRenamed('Apelido', 'atleta') \
                                     .withColumnRenamed('ClubeID', 'id_clube') \
                                     .withColumnRenamed('PosicaoID','id_posicao')

    df_atletas_2017 = df_atletas_2017.withColumnRenamed('AtletaID','id_atleta') \
                                     .withColumnRenamed('Apelido', 'atleta') \
                                     .withColumnRenamed('ClubeID', 'id_clube') \
                                     .withColumnRenamed('PosicaoID','id_posicao')


    df_atletas_2014 = df_atletas_2014.withColumn('ano', lit(2014))
    df_atletas_2015 = df_atletas_2015.withColumn('ano', lit(2015))
    df_atletas_2016 = df_atletas_2016.withColumn('ano', lit(2016))
    df_atletas_2017 = df_atletas_2017.withColumn('ano', lit(2017))

```

```
181
182     print('\n*** ATLETAS ***')
183     # Union com os atletas de todos os anos
184     print('Juntando bases de atletas ...')
185     df_atletas = df_atletas_2014.unionAll(df_atletas_2015).unionAll(df_atletas_2016).unionAll(df_atletas_2017)
186
187     df_atletas = df_atletas.withColumn('id_clube', col('id_clube').cast('int')) \
188                    .withColumn('id_atleta', col('id_atleta').cast('int'))
189
190     df_atletas = df_atletas.withColumn('posicao', \
191                           when(col('id_posicao') == '1', 'Goleiro') \
192                          .when(col('id_posicao') == '2', 'Lateral') \
193                          .when(col('id_posicao') == '3', 'Zagueiro') \
194                          .when(col('id_posicao') == '4', 'Meia') \
195                          .when(col('id_posicao') == '5', 'Atacante') \
196                          .when(col('id_posicao') == '6', 'Tecnico'))
197
198     print('Total de registros na base de atletas: {}'.format(df_atletas.count()))
199
200     df_atletas = df_atletas.distinct()
201     print('Total de registros distintos na base de atletas: {}'.format(df_atletas.count()))
202
203     df_atletas_tmp = df_atletas.drop('id_clube').drop('clube').drop('id_posicao').drop('posicao').drop('ano')
204
205     df_atletas.select('id_atleta','atleta','id_clube','id_posicao','posicao','ano')
206     df_atletas.printSchema()
207     df_atletas.show(50)
208
209 except:
210     print('Erro no processo Atletas: {}'.format(sys.exc_info()[0]))
211     raise
212 else:
213     print('Processo Atletas executado com sucesso !\n')
```

Log:

*** ATLETAS ***

Juntando bases de atletas ...

Total de registros na base de atletas: 3811

Total de registros distintos na base de atletas: 3811

Após a conclusão do tratamento na base_atleta, a mesma ficou da seguinte maneira:

base_atletas

Nome do campo	Descrição	Tipo
id_atleta	ID único do atleta	Int
atleta	Nome/Apelido do atleta	String
id_clube	ID único do clube que o atleta pertence	Int
id_posicao	ID único da posição que o atleta joga	Int

No programa principal, fizemos um join entre o dataframe df_scouts e df_times para trazer o nome dos clubes, depois fizemos um join com a base de atletas através do id para buscarmos os nomes dos atletas, reordenamos nosso dataframe e geramos os arquivos finais utilizando o Pandas. Após o tratamento dos registros distintos, a base foi de 395.305 registros para 47.493 registros.

```
###############
## Principal
###############
try:
    print('DF_SCOUTS:')
    df_scouts.printSchema()

    print('DF_TIMES:')
    df_times.printSchema()

    print('DF_ATLETAS:')
    df_atletas.printSchema()

    # Adiciona nome do clube
    # df = df_scouts.join(df_times, on='id_clube', how='left')

    df = df_scouts.alias('a').join(df_times.alias('b'), col('b.id_clube') == col('a.id_clube')).select(
        [col('a.id_atleta'),col('a.rodada'),col('a.id_clube'),col('a.posicao'),\
         col('a.Jogos'),col('a.pontos'),col('a.pontos_media'),col('a.preco'),\
         col('a.preco_variacao'),col('a.falta_sofrida'),col('a.passe_errado'),\
         col('a.assistencia'),col('a.finalizacao_trave'),col('a.finalizacao_defendida'),\
         col('a.finalizacao_fora'),col('a.gol'),col('a.impedimento'),col('a.penalti_perdido'),\
         col('a.roubada_bola'),col('a.falta_cometida'),col('a.gol_contra'),col('a.cartao_amarelo'),\
         col('a.cartao_vermelho'),col('a.jogo_sem_sofrer_gol'),col('a.defesa_dificil'),\
         col('a.defesa_penalti'),col('a.gol_sofrido'),col('a.ano')] + [col('b.clube')])

    df = df.alias('a').join(df_atletas.alias('b'), col('b.id_atleta') == col('a.id_atleta')).select(
        [col('a.id_atleta'), col('a.rodada'), col('a.id_clube'), col('a.clube'), \
         col('a.Jogos'), col('a.pontos'), col('a.pontos_media'), col('a.preco'), \
         col('a.preco_variacao'), col('a.falta_sofrida'), col('a.passe_errado'), \
         col('a.assistencia'), col('a.finalizacao_trave'), col('a.finalizacao_defendida'), \
         col('a.finalizacao_fora'), col('a.gol'), col('a.impedimento'), col('a.penalti_perdido'), \
         col('a.roubada_bola'), col('a.falta_cometida'), col('a.gol_contra'), col('a.cartao_amarelo'), \
         col('a.cartao_vermelho'), col('a.jogo_sem_sofrer_gol'), col('a.defesa_dificil'), \
         col('a.defesa_penalti'), col('a.gol_sofrido'), col('a.ano')] + [col('b.posicao')])

```

```
252
253         df = df.withColumn('id_clube', col('id_clube').cast('int'))
254
255         df = df.join(df_atletas_tmp, on='id_atleta', how='left')
256         df = df.drop('Jogos')
257
258         df.printSchema()
259
260         print('\n*** DF_FINAL ***')
261         print('\nQuantidade de registros no dataframe (antes): {}\n'.format(df.count()))
262         df = df.distinct()
263         print('Quantidade de registros no dataframe (depois): {}\n'.format(df.count()))
264
265         df = df.select('id_atleta','atleta','rodada','id_clube','clube','posicao','pontos','pontos_media',\
266                        'preco','preco_variacao','falta_sofrida','passe_errado','assistencia',\
267                        'finalizacao_trave','finalizacao_defendida','finalizacao_fora','gol','impedimento',\
268                        'penalti_perdido','roubada_bola','falta_cometida','gol_contra','cartao_amarelo',\
269                        'cartao_vermelho','jogo_sem_sofrer_gol','defesa_dificil','defesa_penalti','gol_sofrido','ano')
270
271         print('DF_FINAL:')
272         df.printSchema()
273         df.show(5)
274
275         # Gravando arquivos com resultados
276         df_pd_scouts = df.toPandas()
277         df_pd_scouts.to_csv(path_or_buf='/home/marcelio/Git/test_local/cartola/bases/resultados/scout.csv', \
278                             sep=';',header=True, mode='w')
279
280         df_pd_times = df_times.toPandas()
281         df_pd_times.to_csv(path_or_buf='/home/marcelio/Git/test_local/cartola/bases/resultados/times.csv', \
282                             sep=';',header=True, mode='w')
283
284         df_pd_atletas = df_atletas.toPandas()
285         df_pd_atletas.to_csv(path_or_buf='/home/marcelio/Git/test_local/cartola/bases/resultados/atletas.csv', \
286                             sep=';',header=True, mode='w')
287
288     except:
289         print('Erro: {}'.format(sys.exc_info()[0]))
290         raise
291     else:
292         print('\nFim: {}\n'.format(dt.datetime.today().strftime('%m/%d/%Y, %H:%M:%S')))
```

Logs:

*** DF_FINAL ***

Quantidade de registros no dataframe (antes): 395305

Quantidade de registros no dataframe (depois): 47493

Com a finalização dos tratamentos realizados na base_scouts, a mesma ficou da seguinte maneira:

base_scout

Nome do campo	Descrição	Tipo
id_atleta	ID único do atleta	Int
atleta	Nome do atleta	String
rodada	Número da rodada do campeonato	Int
id_clube	ID único do clube que o atleta pertence	Int
clube	Nome do clube que o atleta pertence	String
id_posicao	ID único da posição que o atleta joga	Int
posicao	Nome da posição que o atleta joga	String
pontos	Pontos que o atleta fez na rodada do CartolaFC	Double
media_pontos	Média de pontos do atleta no CartolaFC	Double
preco	Preço do atleta no CartolaFC	Double
variacao_preco	Variação do preço do atleta na rodada do CartolaFC	Double
falta_sofrida	Quantidade de faltas que o atleta sofreu na rodada do Cartola FC	Int
passe_errado	Quantidade de passes errados que o atleta deu na rodada do Cartola FC	Int
assistencia	Quantidade de assistências que o atleta deu na rodada do Cartola FC	Int
finalizacao_trave	Quantidade de finalizações na trave que o atleta deu na rodada do Cartola FC	Int
finalizacao_defendida	Quantidade de finalizações defendidas que o atleta teve na rodada do Cartola FC	Int
finalizacao_fora	Quantidade de finalizações pra fora que o atleta teve na rodada do Cartola FC	Int
gol	Quantidade de gols que o atleta fez na rodada do Cartola FC	Int

impedimento	Quantidade de impedimentos que o atleta teve na rodada do Cartola FC	Int
penalti_perdido	Quantidade de penaltis perdidos que o atleta teve na rodada do Cartola FC	Int
roubada_bola	Quantidade de roubadas de bola que o atleta teve na rodada do Cartola FC	Int
falta_cometida	Quantidade de faltas cometidas que o atleta teve na rodada do Cartola FC	Int
gol_contra	Quantidade de gols contra que o atleta fez na rodada do Cartola FC	Int
cartao_amarelo	Quantidade de cartões amarelos que o atleta recebeu na rodada do Cartola FC	Int
cartao_vermelho	Quantidade de cartões vermelhos que o atleta recebeu na rodada do Cartola FC	Int
jogo_sem_sofrer_gol	Retorna 1 ou 0 quando o atleta que faz parte da defesa fica um jogo sem levar gols. 1 = jogo sem sofrer gols 0 = jogo que sofreu gols	Int
defesa_dificil	Quantidade de defesas difíceis que o atleta (goleiro) fez na rodada do Cartola FC	Int
defesa_penalti	Quantidade de defesas de penaltis que o atleta (goleiro) fez na rodada do Cartola FC	Int
gol_sofrido	Quantidade de gols sofridos que o atleta (goleiro) teve na rodada do Cartola FC	Int
ano	Informação do ano em que ocorreu a pontuação	Int

Mesmo após a conclusão dos tratamentos, ainda encontramos um número baixo de registros duplicados. A princípio o compando “drop_duplicates” resolveria o problema, mas descobrimos que o registro duplicado era um pouco mais complexos e foi necessário retirar esses registros manualmente. O número de registros duplicados excluídos manualmente foi de 342 registros.

4. Análise e Exploração dos Dados

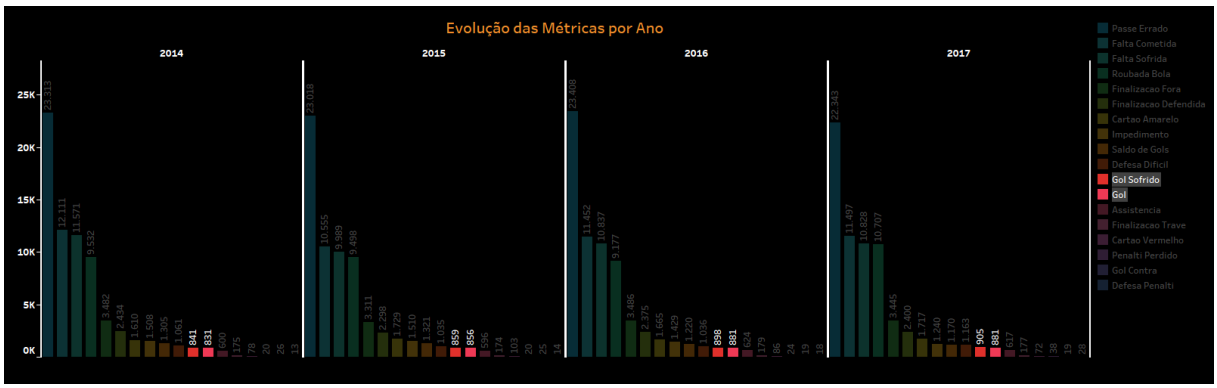
Utilizamos a ferramenta Tableau para a exploração e visualização dos dados por possuir licença gratuita de um ano para estudantes e por ser uma das líderes de mercado segundo o quadrante Gartner's:

Após o final do tratamento dos dados descritos no item 3, as bases foram disponibilizas em arquivos .csv e, com isso, os mesmos foram importados para o Tableau para a criação das análises e dashboards.

4.1 Evolução das Métricas por Ano

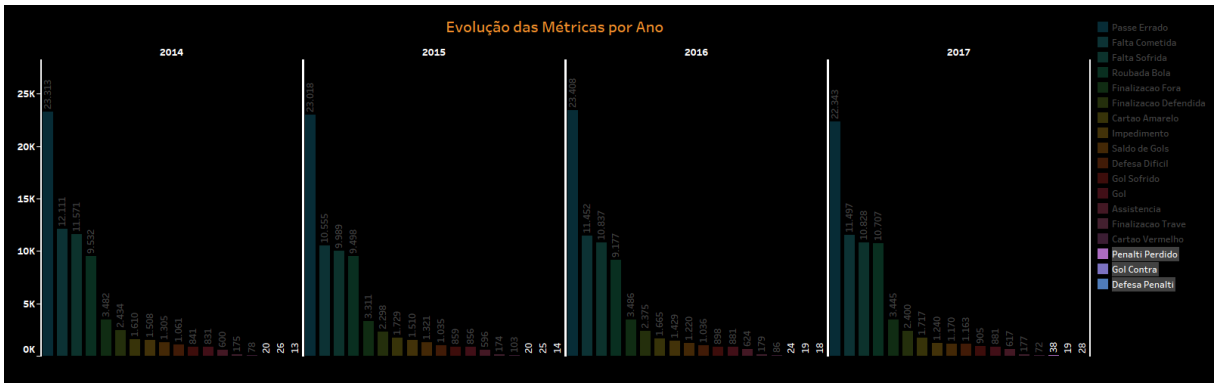
Fizemos um gráfico de barras para acompanhar a evolução das métricas dos atletas com o passar dos anos. Conseguimos observar alguns dados interessantes com o gráfico de barra abaixo.

Filtrando somente as métricas “Gol” (Gol feito pelos atletas) e “Gol sofrido” (Gols que os goleiros sofrem) reparamos que de 2014 até 2017 o número de gols vem aumentando gradativamente.

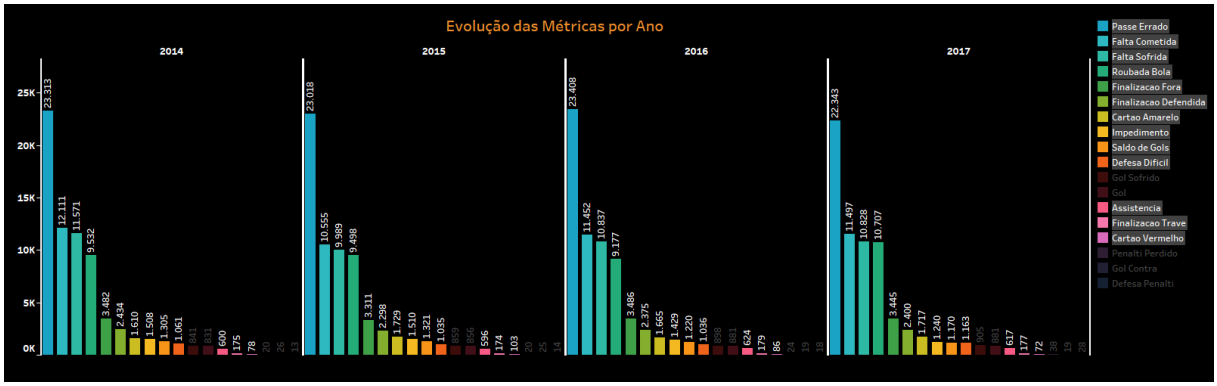


Na imagem abaixo, filtramos as métricas “Penalti Perdido”, “Gol Contra” e “Defesa de Penalti”, números da esquerda pra direita respectivamente. Percebemos que o número de penaltis perdidos vem aumentando (20, 20, 24 e 38) e o número de defesas de penaltis também (13, 14, 18 e 28). Conclui-se que o nível técnico dos goleiros vem melhorando.

Em contra-partida, o número de gols contra vem diminuindo com o passar dos anos. Em 2014 foram 26, baixou em 2015 e 2016 até chegar a um total de 19 gols contra em 2017.



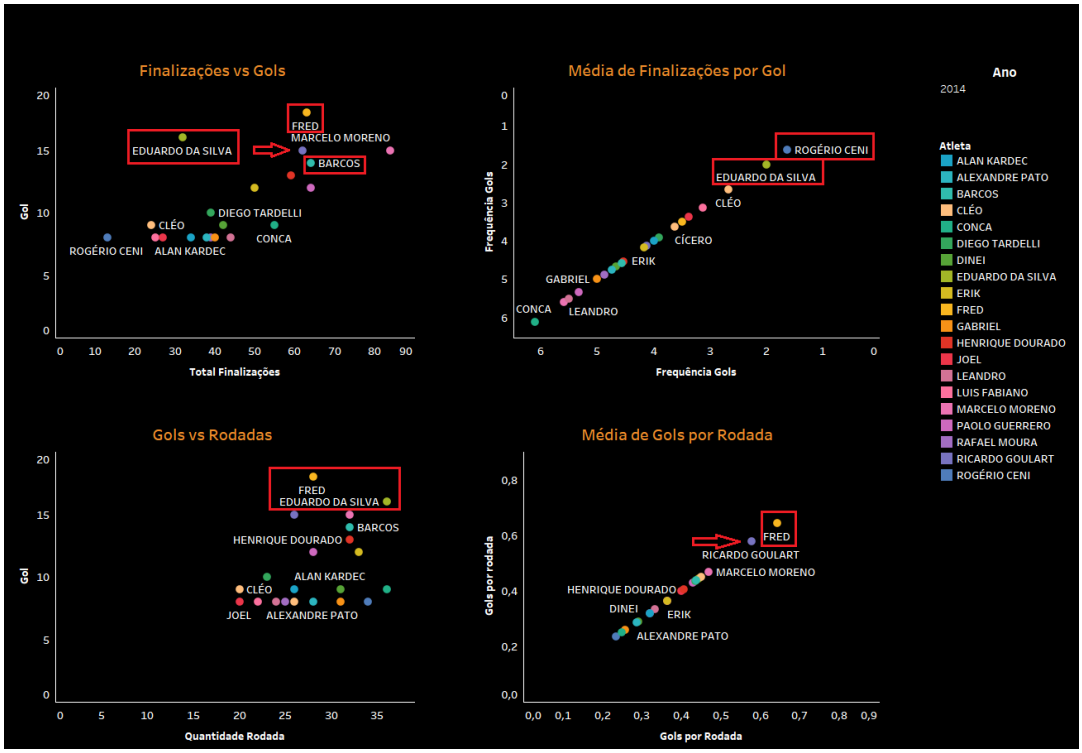
As demais métricas se mantiveram na média sem nenhuma grande diferença de um ano para o outro. Talvez seja interessante mencionar que o número de passes errados caiu em mais de mil de 2016 para 2017 (parece um número pequeno, mas se tratando de passes errados significa que os jogadores estão melhorando nesse fundamento). Outra métrica que talvez valha citar é a de falta cometida que de 2014 para 2015 caiu o número em pouco mais de 1500 (de 12.111 para 10.555), isso mostra que os jogadores de marcação conseguiram roubar mais a bola fazendo menos faltas. De 2016 para 2017 a roubada de bola também teve um salto de 9.177 para 10.707 reforçando como os jogadores de marcação melhoraram nesse fundamento.



4.2 Gols

Analisando o desempenho dos atletas na principal métrica do futebol fica claro que em sua grande maioria os atacantes são os responsáveis por fazer mais gols. Porém, nem todos os principais artilheiros são os que tem uma melhor média de finalização. Como podem ver na imagem abaixo (Ano 2014), os atacantes Fred, Marcelo Moreno, Eduardo da Silva e Barcos são os que mais somaram Gols no Campeonato, com a exceção do atleta Ricardo Goulart (indicado pela seta vermelha) que é meio-campo.

O atleta que mais finalizou foi o Marcelo Moreno (Cruzeiro), mas o que mais marcou gols foi o Fred (Fluminense).

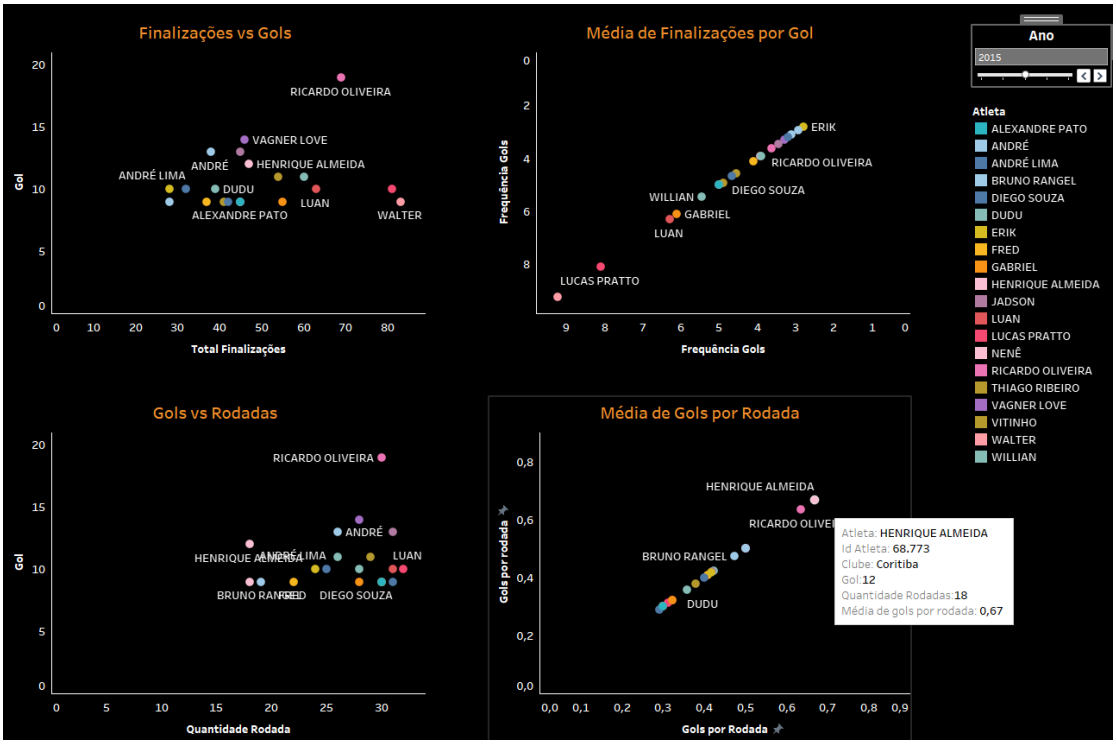


Ainda sobre a imagem acima, o atleta que possui a melhor média de finalização por gol é o Rogério Ceni (Goleiro do São Paulo). Ele é famoso por suas cobranças de faltas e pênaltis. Mas aí você vai questionar, mas o Rogério Ceni não só bate faltas e pênaltis?! Sim, porém, a cada 1,63 finalizações ele faz um gol, ou seja, mesmo que só bata faltas e pênaltis, ele acaba tendo um excelente aproveitamento nesses fundamentos. Após o Rogério Ceni, o que possui a melhor média de chutes por gol é o atacante Eduardo da Silva (Flamengo). A cada 2 chutes ele faz um gol.

Já no quesito média de gols por rodada quem “ganha” é o atacante Fred (Fluminense), ele faz 0,64 gols a cada rodada. Logo em seguida vem o meio-campo Ricardo Goulart (Cruezeiro) com média de 0,58 gols por partida.

Verificando todos os anos (2014 a 2017) o atleta que possui a melhor média de finalizações por gol é Rogério Ceni com 1,63 chutes para cada gol. Se excluírmos o Rogério, a melhor média de finalizações por gol fica com o Eduardo da Silva que chutou 32 vezes e fez 16 gols em 36 rodadas, média de 2 chutes para fazer um gol.

Já na média de gols por rodadas (nos anos de 2014 a 2017) quem possui o melhor número é o atacante Henrique Almeida (Coritiba) que no ano de 2015 jogou 18 partidas e fez 12 gols, uma média de 0,67 gols por partida.



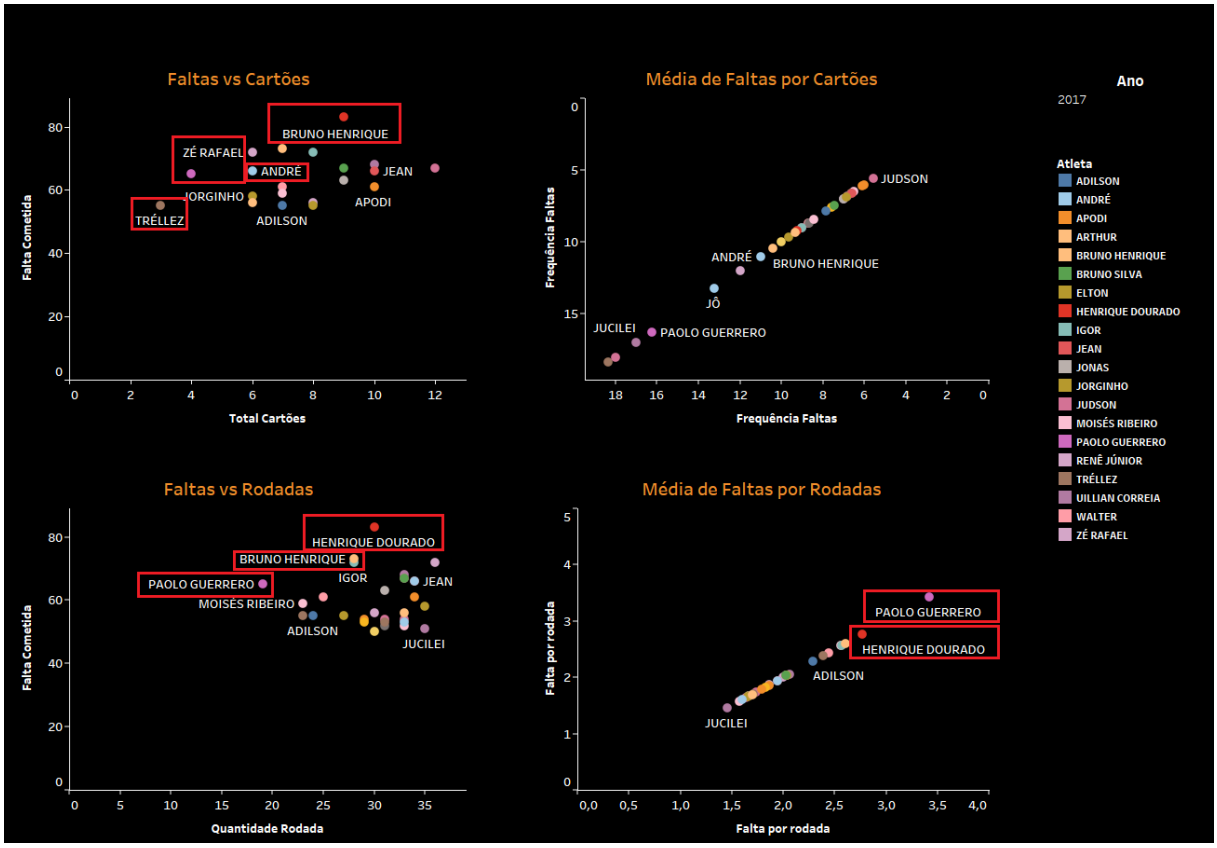
4.3 Faltas

Fizemos uma análise para descobrir qual o jogador que não vale a pena ser escalado no game por fazer muitas faltas e levar cartões. Essas métricas (falta cometida, cartão amarelo e cartão vermelho) dão pontos negativos para os atletas e, com isso, o time que foi montado pode não ir tão bem na rodada. Filtrando somente o ano de 2017, observamos que os atacantes são os atletas que fizeram mais faltas. Normalmente o atacante não sabe marcar um atleta adversário e sim marcar gols, talvez por isso o número alto de faltas para os atletas nessa posição.

O atleta que mais fez faltas foi o Henrique Dourado (Fluminense) com um total de 83 faltas e 9 cartões (amarelos e/ou vermelhos), é dele também a segunda pior média de faltas por rodada com 2,77 faltas a cada rodada (o “campeão nesse

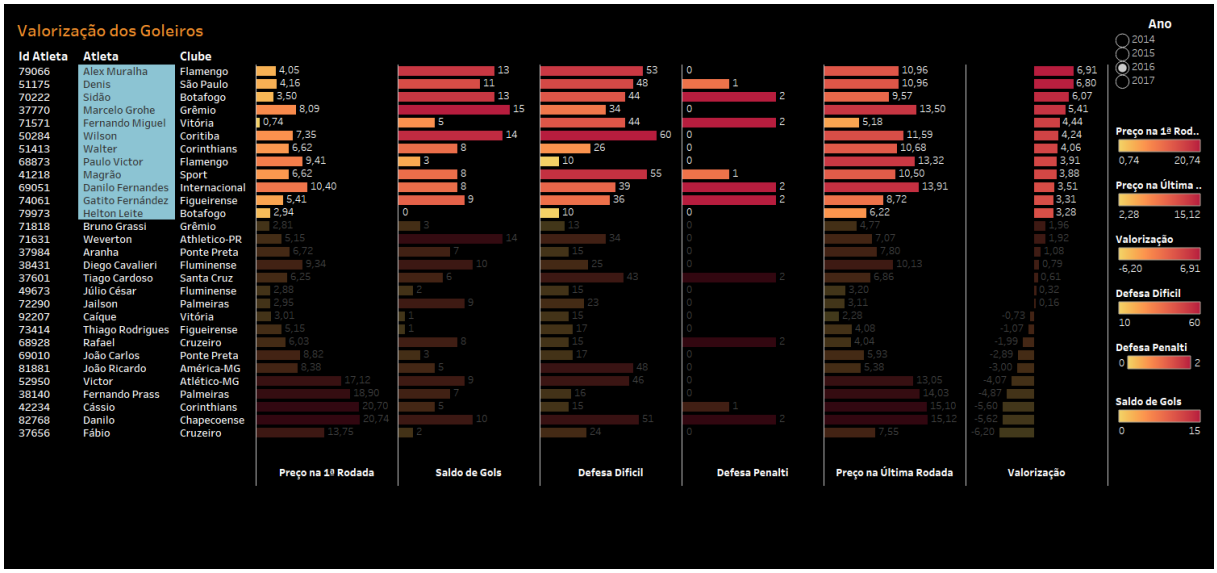
quesito é Paolo Guerrero com 3,42 faltas a cada rodada). Outro atacante que também fez muita falta foi Bruno Henrique (Santos) com um total de 73 faltas e 7 cartões (amarelos e/ou vermelhos).

O atleta mais “violento” é o Judson (Avaí) com um total de 67 faltas cometidas e 12 cartões em 33 rodadas. A cada 5,58 faltas ele leva um cartão (vermelho ou amarelo).

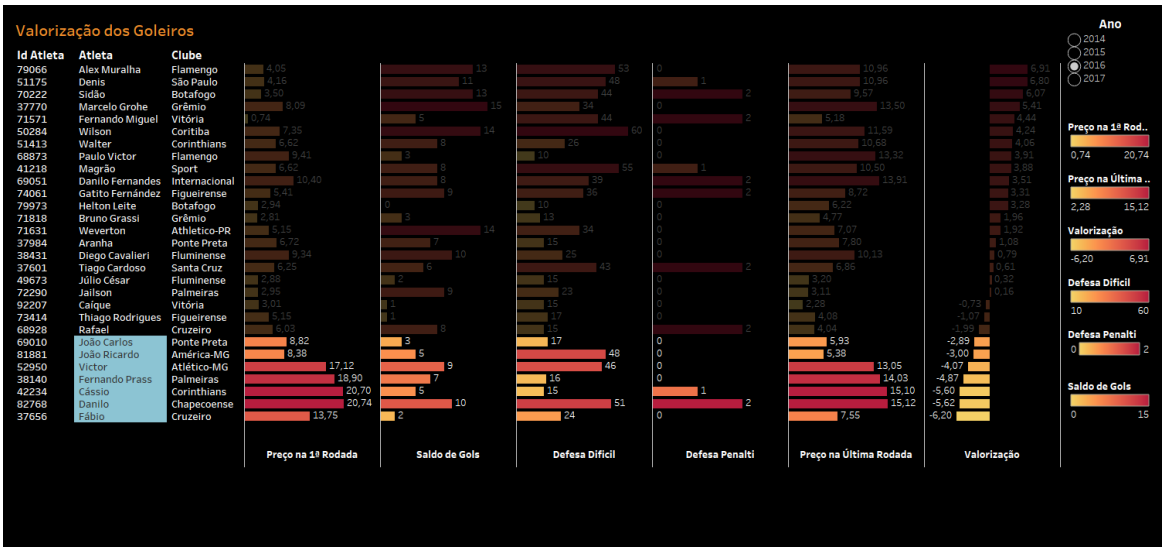


4.4 Valorização dos Goleiros

Analisamos o desempenho e a valorização (em cartoletas) do goleiro no CartolaFC e concluímos que o goleiro que na primeira rodada possui um preço baixo (na faixa de 10 cartoletas) e tem um bom desempenho, no final do game ele terá valorizado bastante. Vide imagem abaixo onde filtramos somente o ano de 2016. Percebe-se claramente que os goleiros que começaram com um preço baixo tiveram uma grande valorização.



Em contra partida, os goleiros que começaram o game com um valor muito alto (pouco acima da faixa de 10 cartoletas), tiveram uma desvalorização mesmo (as vezes) tendo um bom desempenho.



4.5 Pontuação dos Meio Campistas

O objetivo de analisar os meio-campsitas era para tentar entender se os melhores meio-campistas (os 3 que fazem parte da seleção do CartolaFC, normalmente no esquema 4-3-3) são os que dão mais assistências e fazem mais gols (os meia clássicos) ou são os que roubam mais bola (os volantes)?

No gráfico abaixo, temos os 25 melhores meio campistas (que mais pontuaram no game) e, com essa quantidade, conseguimos ter uma idéia de quais são as métricas que mais contribuem para que eles alcancem suas respectivas pontuações. Filtramos somente o ano de 2015 e os 3 melhores da posição são: Jadson (Corinthians), Otávio (Athletico-PR) e Diego Souza (Sport).

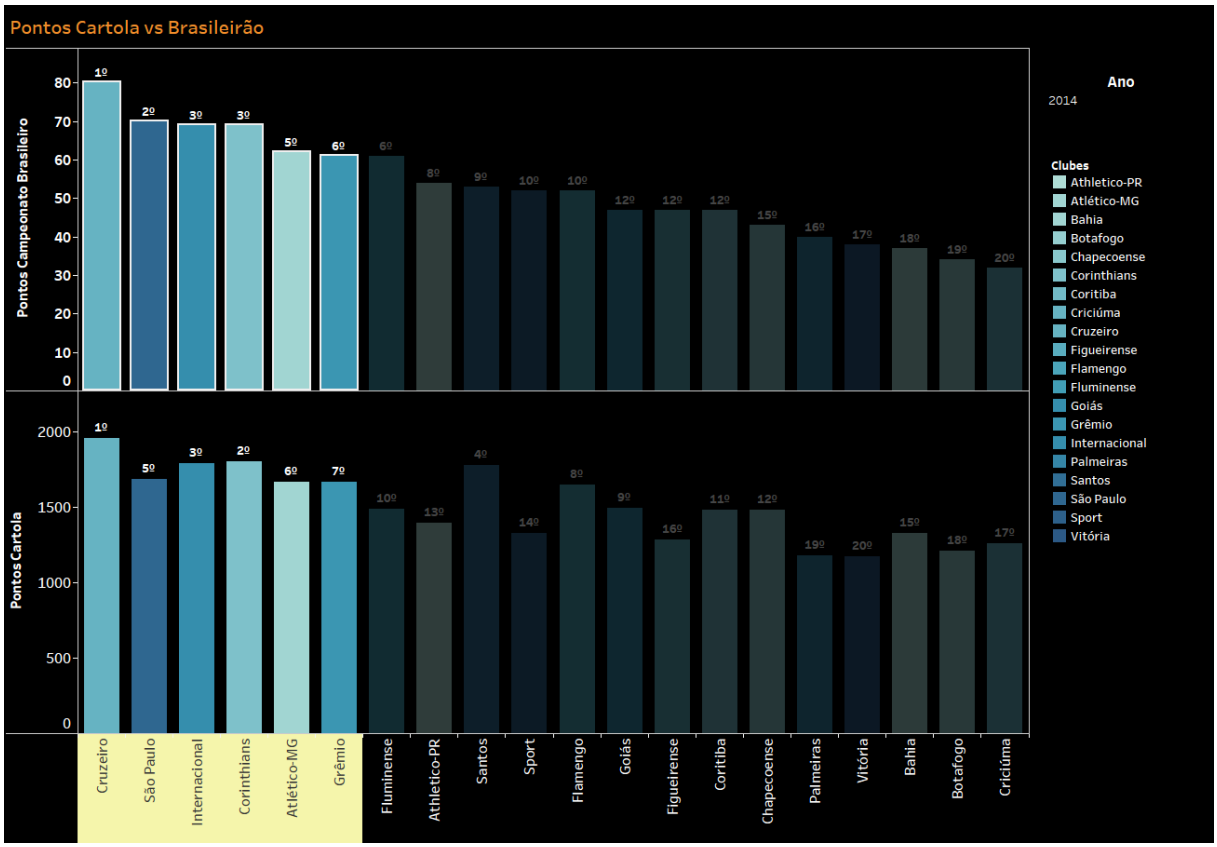
Percebam que o Jadson deu um total de 12 assistências e fez 13 gols, essas métricas somadas deram a ele 164 pontos (importante ressaltar que a soma dos pontos de roubadas de bola, assistências e gols ilustradas no gráfico acima não são exatamente o total de pontos do atleta, visto que o mesmo possui pontuações em outras métricas, em alguns cenários pontuações negativas). Ou seja, somente com assistências e gols o Jadson já conseguiu uma ótima pontuação, mas será que somente essas métricas são necessárias para o atleta ficar no top3 da sua posição? Vamos mostrar que não.

O atleta Otávio (Athletico-PR) conseguiu roubar 96 bolas dos adversários, isso deu a ele um total de 163,2 pontos. Perceberam? Somente com roubadas de bola o Otávio já atingiu a pontuação do Jadson, imagina se ele ainda tivesse dado um número razoável de assistências e feito gols? Com certeza seria o top1. Mas nesse ano, o Otávio só deu uma assistência e não fez nenhum gol, acabou figurando no top3 graças ao fundamento de roubada de bola.

Nos outros anos os cenários são parecidos, encontraremos os meias clássicos e volantes entre os top3.

4.6 Pontos no Cartola vs Pontos no Brasileirão

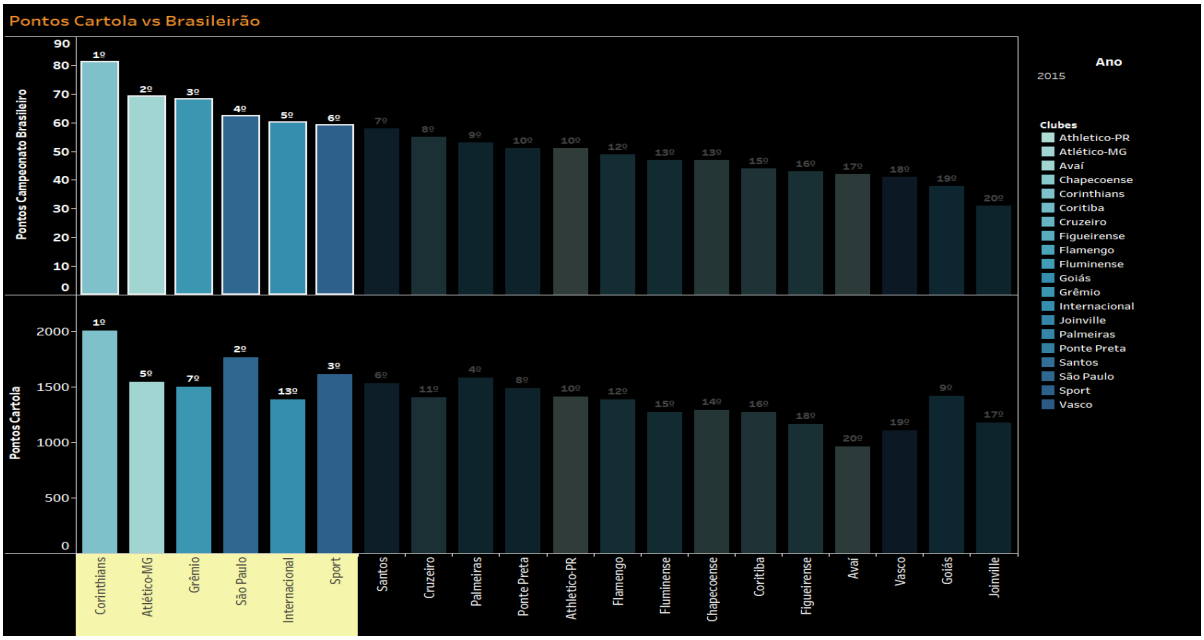
Aqui faremos a nossa primeira co-relação com o mundo real, iremos comparar se a classificação do clube no Cartola FC (ranking com a soma dos pontos dos jogadores do clube) está relacionada com a classificação do clube no Campeonato Brasileiro Série A. No gráfico abaixo, com o ano de 2014 filtrado, comparamos a classificação do campeonato com a classificação no game.



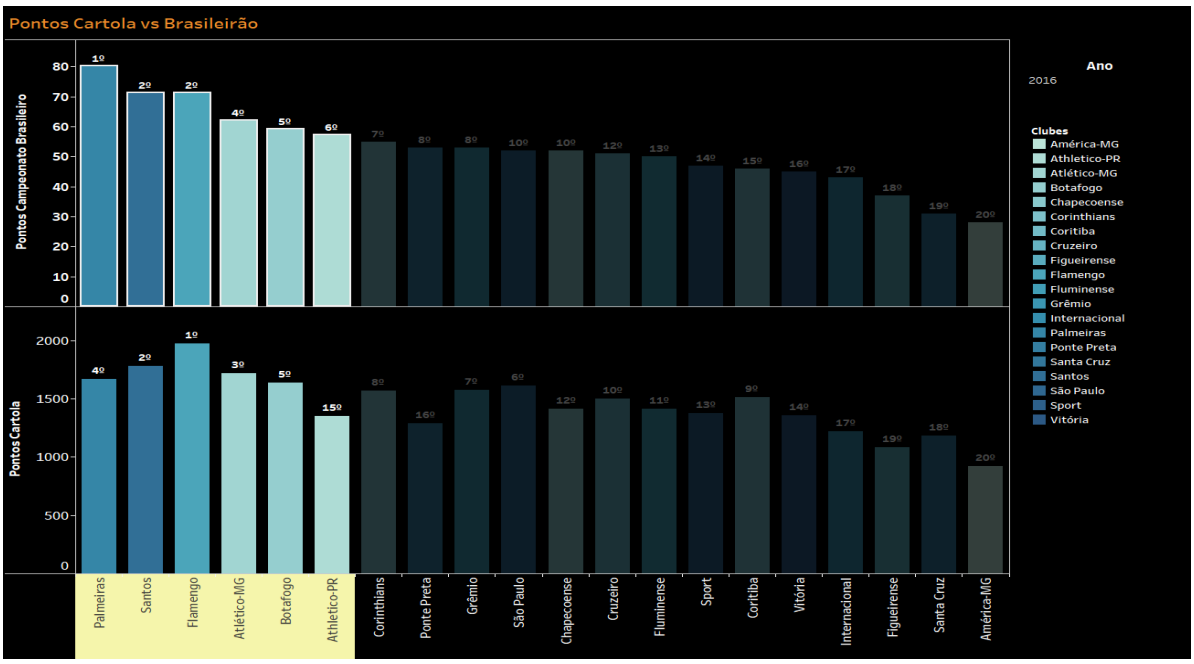
A conclusão que a gente chega é que o clube que fica no top 6 da classificação do Campeonato Brasileiro também fica no top 6 da classificação do Cartola FC. Resumindo, se o time vence vários jogos no Campeonato Brasileiro é natural que os jogadores desse mesmo time pontuem bem no Cartola (é natural, mas nem sempre é regra, vide as exceções no gráfico).

Para os demais anos o cenário se repete:

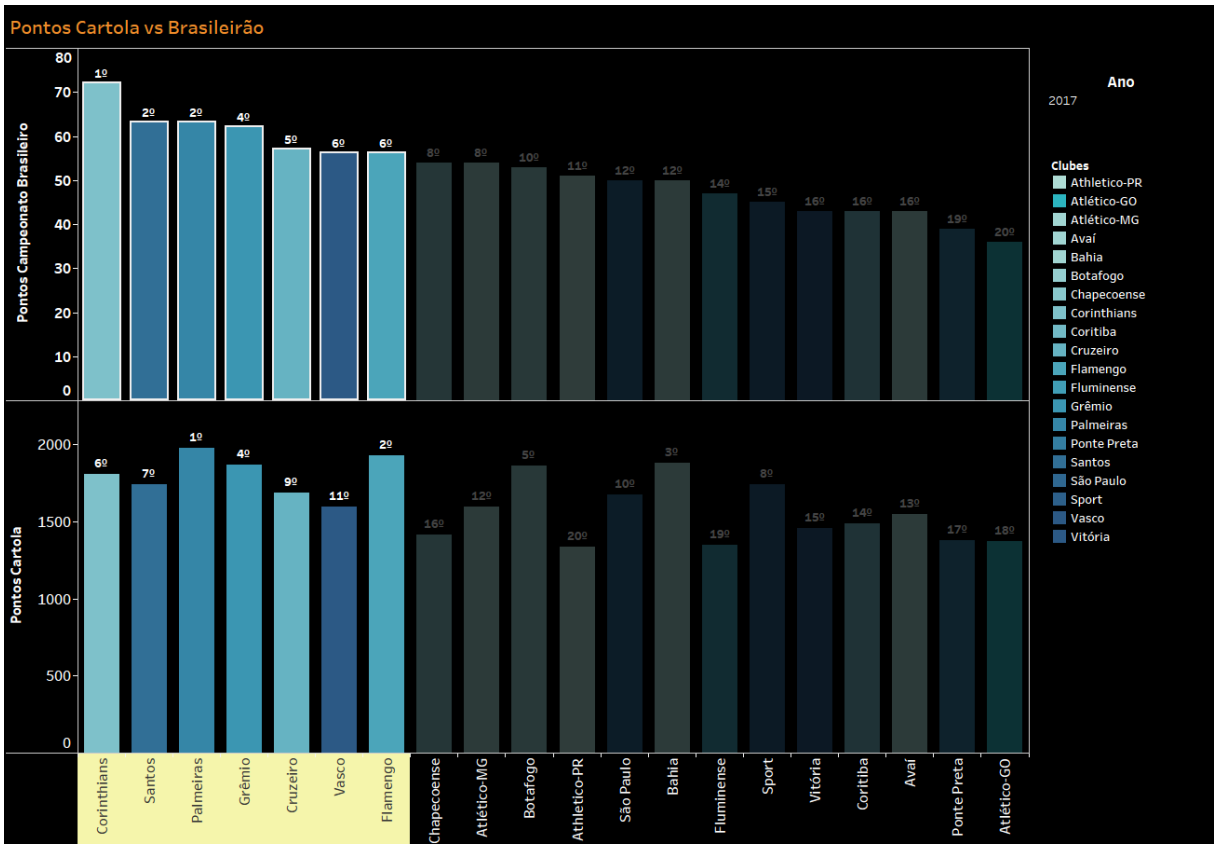
2015



2016

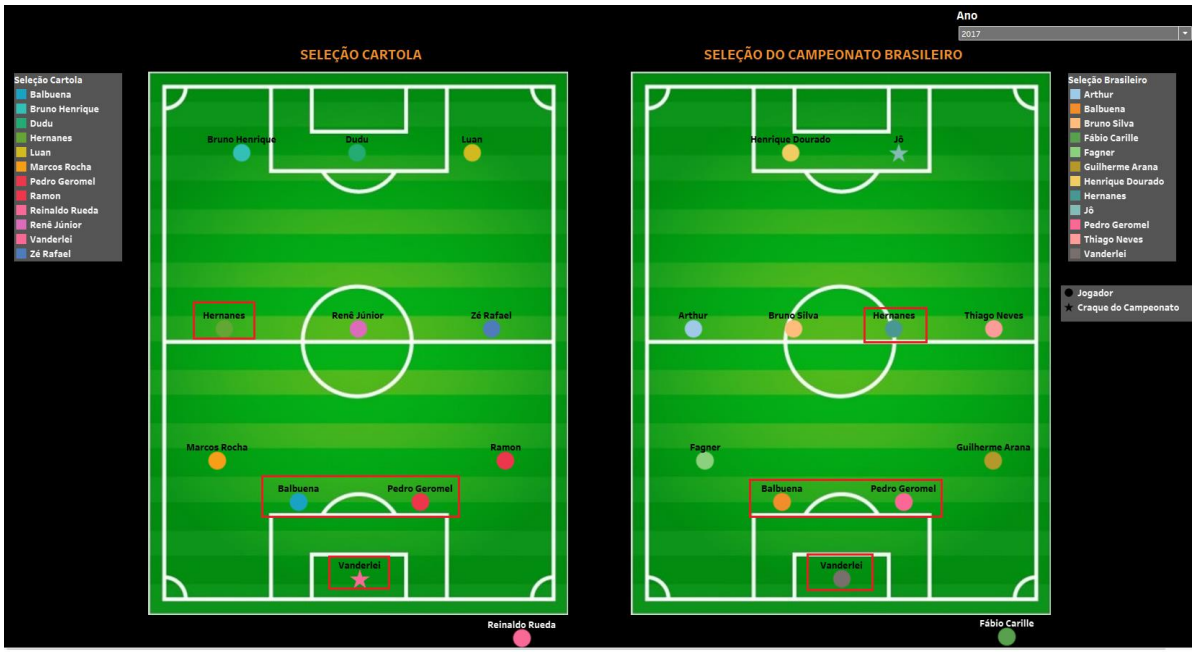


2017


Pontos Cartola vs Brasileirão
Pontos Campeonato Brasileiro
80
70
60
50
40
30
20
10
0
1º
2º
2º
4º
5º
6º
6º
8º
8º
10º
11º
12º
12º
14º
15º
16º
16º
16º
19º
20º
Pontos Cartola
2000
1500
1000
500
0
6º
7º
1º
4º
9º
11º
2º
16º
12º
5º
20º
10º
3º
19º
8º
15º
14º
13º
17º
18º
Corinthians
Santos
Palmeiras
Grêmio
Cruzeiro
Vasco
Flamengo
Chapecoense
Atlético-MG
Botafogo
Athletico-PR
São Paulo
Bahia
Fluminense
Sport
Vitória
Coritiba
Avaí
Ponte Preta
Atlético-GO
Ano
2017
Clubes
Athletico-PR
Atlético-GO
Atlético-MG
Avaí
Bahia
Botafogo
Chapecoense
Corinthians
Coritiba
Cruzeiro
Flamengo
Fluminense
Grêmio
Palmeiras
Ponte Preta
Santos
São Paulo
Sport
Vasco
Vitória

4.7 Seleção do Cartola vs Seleção do Brasileirão

Aqui fazemos a nossa segunda co-relação. Comparamos a seleção do Cartola (Composta pelos maiores pontuadores de cada posição, normalmente no esquema tático 4-3-3) com a seleção do Campeonato Brasileiro (eleita por jornalistas, técnicos, jogadores e CBF, normalmente no esquema tático 4-4-2). Também comparamos se o craque do campeonato (também decido por eleição) foi o maior pontuador do game.



Na imagem acima, filtramos somente o ano de 2017 e colocamos lado a lado ambas as seleções. Reparem que dos 12 atletas (jogadores e técnico), temos um total de 4 jogadores (Hernanes – meio-campo, Balbuena e Pedro Geromel - zagueiros e Vanderlei - goleiro))que fazem parte de ambos os times. O Jô (jogador do Corinthians eleito craque do campeonato) nem figura na seleção do Cartola. No game o jogador que mais pontuou foi o Vanderlei (Goleiro do Santos).

Abaixo ilustramos as seleções do ano de 2016 e temos um total de 3 atletas (jogadores e técnico) nos dois times. Gabriel Jesus (eleito craque do campeonato) até figura na seleção do Cartola, mas não foi o maior pontuador.

Em 2015 o cenário é igual a 2016, um total de 3 atletas (entre jogadores e técnico) em ambas as seleções. E novamente o craque do campeonato (Renato Augusto) nem figura na seleção do game.

5. Criação de Modelos de Machine Learning

Decidimos por utilizar o algoritmo K-Nearest Neighbors Classification (que em português seria algo como “Classificação K-Vizinhos mais próximos) que é um aprendiz de classificação simples e poderoso.

O KNN é um dos muitos algoritmos (de aprendizagem supervisionada) usados no campo de data mining e machine learning, ele é um classificador onde o aprendizado é baseado “no quão similar” é um dado (um vetor) do outro. Este algoritmo pode ser aplicado em diversos segmentos de negócio, logo também se aplica em diversos problemas como finanças, saúde, ciência política, reconhecimento de imagem e reconhecimento de vídeos. Ele também pode ser usado tanto para classificação quanto para regressão.

No nosso modelo, iremos utilizar a classificação, para prever qual a posição (lateral, zagueiro, meio campo ou atacante) de um atleta baseado nos scouts (falta_sofrida, falta_cometida, impedimento, gol, assistência, etc...)

O dataset base_scout possui um total de 44020 registros entre os anos de 2014 e 2017, para o modelo vamos filtrar somente o ano de 2014.

No código abaixo vamos usar as bibliotecas de machine learning do pyspark, Pandas e Numpy. A execução do algoritmo pode ser realizada diretamente no *jupyter notebook*.

Trecho do código mostrando a leitura do dataframe scouts_2014.

Ferramentas para análise

```
In [1]: # análise e processamento de dados

        import pandas as pd
        import numpy as np

        # plotagem

        import seaborn as sns
        import matplotlib.pyplot as plt
        %matplotlib inline
```

Ingestão de Dados

```
In [2]: scouts_2014 = pd.read_excel("../data/ML-Cartola.xlsx")
```

Trecho do código mostrando que verificamos os valores nulos nas variáveis.

```
In [3]: # verificando os valores nulos

        scouts_2014.isna().sum()

Out[3]: id_atleta               0
        atleta                  0
        rodada                  0
        id_clube                0
        clube                   0
        posicao                 0
        pontos                  0
        pontos_media            0
        preco                   0
        preco_variacao          0
        falta_sofrida           0
        passe_errado            0
        assistencia             0
        finalizacao_trave       0
        finalizacao_defendida   0
        finalizacao_fora        0
        gol                     0
        impedimento             0
        penalti_perdido         0
        roubada_bola            0
        falta_cometida          0
        gol_contra              0
        cartao_amarelo          0
        cartao_vermelho         0
        jogo_sem_sofrer_gol     0
        defesa_dificil          0
        defesa_penalti          0
        gol_sofrido             0
        ano                     0
        dtype: int64
```

Trecho do código mostrando que verificamos o número de registros de cada variável, se a mesma era nula e o tipo da variável.

```
In [4]: # conhecendo os tipos de dados e outras coisas do conjunto de dados

        scouts_2014.info()

        <class 'pandas.core.frame.DataFrame'>
        RangeIndex: 11123 entries, 0 to 11122
        Data columns (total 29 columns):
        id_atleta               11123 non-null int64
        atleta                  11123 non-null object
        rodada                  11123 non-null int64
        id_clube                11123 non-null int64
        clube                   11123 non-null object
        posicao                 11123 non-null object
        pontos                  11123 non-null float64
        pontos_media            11123 non-null float64
        preco                   11123 non-null float64
        preco_variacao          11123 non-null float64
        falta_sofrida           11123 non-null int64
        passe_errado            11123 non-null int64
        assistencia             11123 non-null int64
        finalizacao_trave       11123 non-null int64
        finalizacao_defendida   11123 non-null int64
        finalizacao_fora        11123 non-null int64
        gol                     11123 non-null int64
        impedimento             11123 non-null int64
        penalti_perdido         11123 non-null int64
        roubada_bola            11123 non-null int64
        falta_cometida          11123 non-null int64
        gol_contra              11123 non-null int64
        cartao_amarelo          11123 non-null int64
        cartao_vermelho         11123 non-null int64
        jogo_sem_sofrer_gol     11123 non-null int64
        defesa_dificil          11123 non-null int64
        defesa_penalti          11123 non-null int64
        gol_sofrido             11123 non-null int64
        ano                     11123 non-null int64
        dtypes: float64(4), int64(22), object(3)
        memory usage: 2.5+ MB
```

No trecho abaixo nós selecionamos as colunas de interesse para a realização do modelo: posicao, pontos, preco, preco_variacao, falta_sofrida, passe_errado, assistencia, finalizacao_trave, finalizacao_defendida, finalizacao_fora, gol, impedimento, penalti_perdido, roubada_bola, falta_cometida, gol_contra, cartao_amarelo e cartao_vermelho.

Excluímos as posições de “Goleiro” e “Tecnico” do modelo. O goleiro por possuir métricas específicas para ele (defesa_difícil, defesa_penalti e gol_sofrido) e o técnico por pontuar em cima das médias de todos os seus atletas.

Como o modelo de machine learning geralmente lida com valores numéricos, convertemos as posições restantes em números usando a função .map(mapping).

Modelando

É possível prever a posição de um atleta através de seus scouts?

In [5]:
```
# Load libraries
from sklearn import datasets
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from sklearn import neighbors
```

Como a modelagem e os modelos de aprendizado de máquina geralmente lidam com valores numéricos, converteremos valores categóricos em valores numéricos, usando uma função de mapa no objeto Data Frame.

In [6]:
```
# selecionando as colunas de interesse

interest_columns = ['posicao', 'pontos', 'preco', 'preco_variacao',
                    'falta_sofrida', 'passe_errado', 'assistencia', 'finalizacao_trave',
                    'finalizacao_defendida', 'finalizacao_fora', 'gol', 'impedimento',
                    'penalti_perdido', 'roubada_bola', 'falta_cometida', 'gol_contra',
                    'cartao_amarelo', 'cartao_vermelho']


scouts_2014 = scouts_2014.loc[:, interest_columns]


# Para essa análise, excluiremos os goleiros (por terem métricas específicas para eles) e os tecnicos.

scouts_2014 = scouts_2014[(scouts_2014["posicao"] != "Goleiro") & (scouts_2014["posicao"] != "Tecnico")]
```

In [7]:
```
# mapeando posições nomeadas e seus IDs apropriados

mapping = {
    "Meia": 4,
    "Atacante": 5,
    "Lateral": 2,
    "Zagueiro": 3,
}

scouts_2014["posicao"] = scouts_2014["posicao"].map(mapping)
```

Nesse trecho a gente cria o conjunto de treinamento e teste, instancia e treina o classificador definindo a classificação com valor = 10 e usando o algortimo KNN. Por fim, printamos o desempenho do classificador que é de 0,67.

Primeira Abordagem

Usando dados brutos - sem escala - para prever a posição dos jogadores no campo.

In [8]:
```
# Crição do conjuntos de treinamento e teste

X_train, X_test, y_train, y_test = train_test_split(scouts_2014.drop("posicao", axis="columns"),
                                                    scouts_2014["posicao"],
                                                    test_size=0.2,
                                                    random_state=42)
```

In [9]:
```
# instanciando e treinando o classificador

clf = neighbors.KNeighborsClassifier(10, weights = 'uniform')

trained_model = clf.fit(X_train, y_train)
```

In [10]:
```
# O classificador tem bom desempenho nos dados de treinamento?

print("Models' performance on training data: {}".format(trained_model.score(X_train, y_train)))
```

```
Models' performance on training data: 0.6690479283023769
```

Conclusão

Os modelos cujo desempenho nos dados de treinamento não atinge o desempenho máximo não podem ser tão ruins. Há uma troca que pode ser explorada entre desempenho muito bom em treinamento (sobreajuste) e muito pior (subajuste).

Nesse contexto, 0,67 como pontuação geral é melhor que nada!

No trecho abaixo, temos o resultado incial da precisão na previsão das posições dos atletas, lembrando que a posição 2 = lateral, 3 = zagueiro, 4 = meio campo e 5 = atacante.

Plotagem do Relatório de Classificação

In [11]:
```
y_pred = trained_model.predict(X_test)

print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           2       0.25      0.38      0.31       330
           3       0.29      0.30      0.30       324
           4       0.48      0.46      0.47       780
           5       0.51      0.35      0.42       491

    accuracy                           0.39      1925
   macro avg       0.38      0.37      0.37      1925
weighted avg       0.42      0.39      0.40      1925
```

Aqui tentamos uma segunda abordagem definindo o valor da classificação = 3. Verificamos que houve uma pequena melhora no índice do desempenho do classificador que agora é de 0,69.

Segunda Abordagem

O Scout "Scouts" valoriza o melhor desempenho dos modelos?

In [12]:
```
sc = StandardScaler()

# escalando os dados de treinamento

X = scouts_2014.drop("posicao", axis="columns")
X_std = sc.fit_transform(X)
```

```
E agora, o mesmo processo repetido como visto acima
```

In [13]:
```
# Criação do treinamento de dados e teste
X_train, X_test, y_train, y_test = train_test_split(X_std,
                                                    scouts_2014["posicao"],
                                                    test_size=0.2,
                                                    random_state=42)

# Treinando o modelo

clf = neighbors.KNeighborsClassifier(3, weights = 'uniform')
trained_model = clf.fit(X_train, y_train)
```

In [14]:
```
trained_model.score(X_train, y_train)
```

Out[14]:
```
0.6880114300558514
```

No trecho abaixo, temos o resultado final da precisão na previsão das posições dos atletas. Percebe-se que comparando com a primeira abordagem, houve uma leve melhora no desempenho do modelo, mas ainda há o que estudar e melhorar.

Plotagem do Relatório de Classificação

In [15]:
```
# O classificador executa bem os dados de treinamento?

print("Models' performance on training data: {}".format(trained_model.score(X_train, y_train)))
```

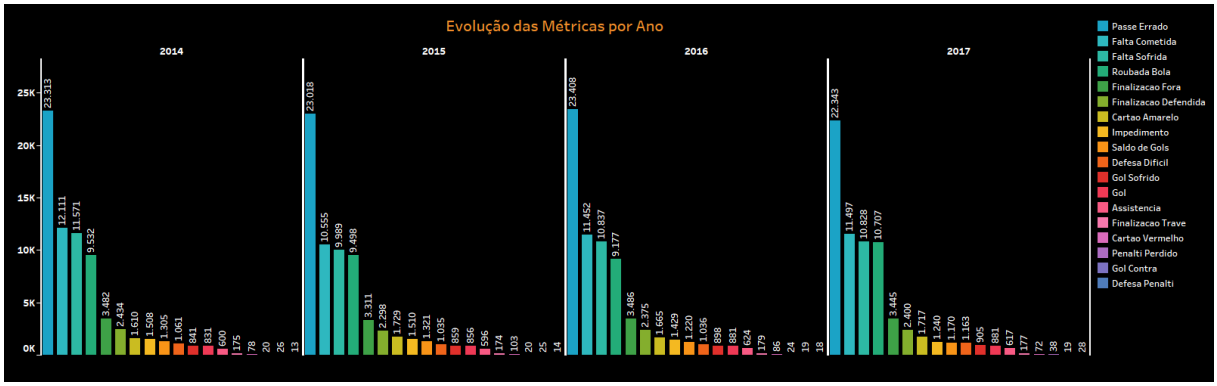
```
Models' performance on training data: 0.6880114300558514
```

In [17]:
```
y_pred = trained_model.predict(X_test)

print(classification_report(y_test, y_pred))
```

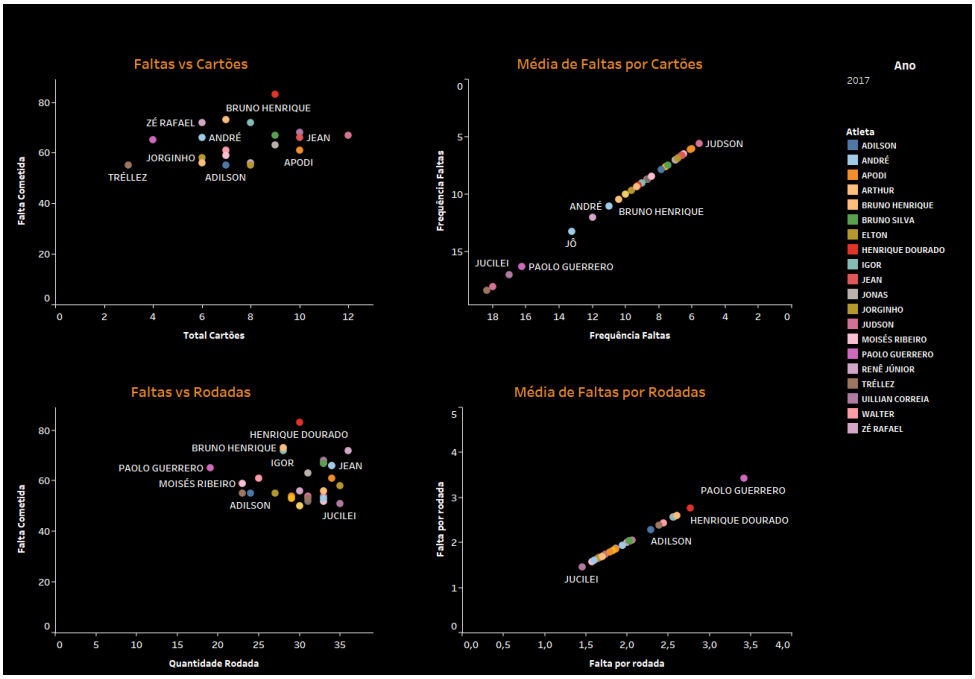
```
              precision    recall  f1-score   support

           2       0.28      0.38      0.32       330
           3       0.33      0.29      0.31       324
           4       0.50      0.48      0.49       780
           5       0.49      0.44      0.47       491

    accuracy                           0.42      1925
   macro avg       0.40      0.40      0.40      1925
weighted avg       0.43      0.42      0.43      1925
```

6. Apresentação dos Resultados

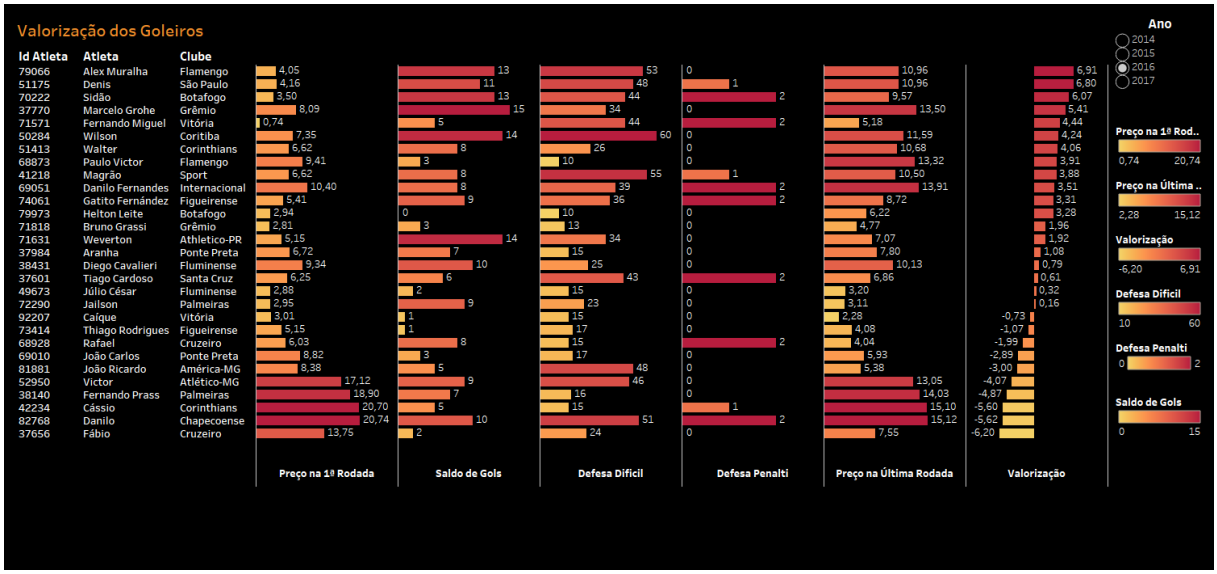
Depois de todas as análise realizadas no tópico 4 concluímos que houve evolução em algumas métricas de performance dos atletas com o passar dos anos, sejam elas aumentadas (defesa de penalti, gols e outras) ou diminuídas (exemplo: faltas cometidas, gols contra e outras).



Concluímos também que nem sempre vale a pena escalar um atacante que faça gols mas que também faça muitas faltas. Esse atleta vai pontuar positivamente quando fizer gols (que não ocorre em todas as rodadas), mas vai pontuar negativamente quando fizer faltas (essa métrica acontece com uma frequência maior).

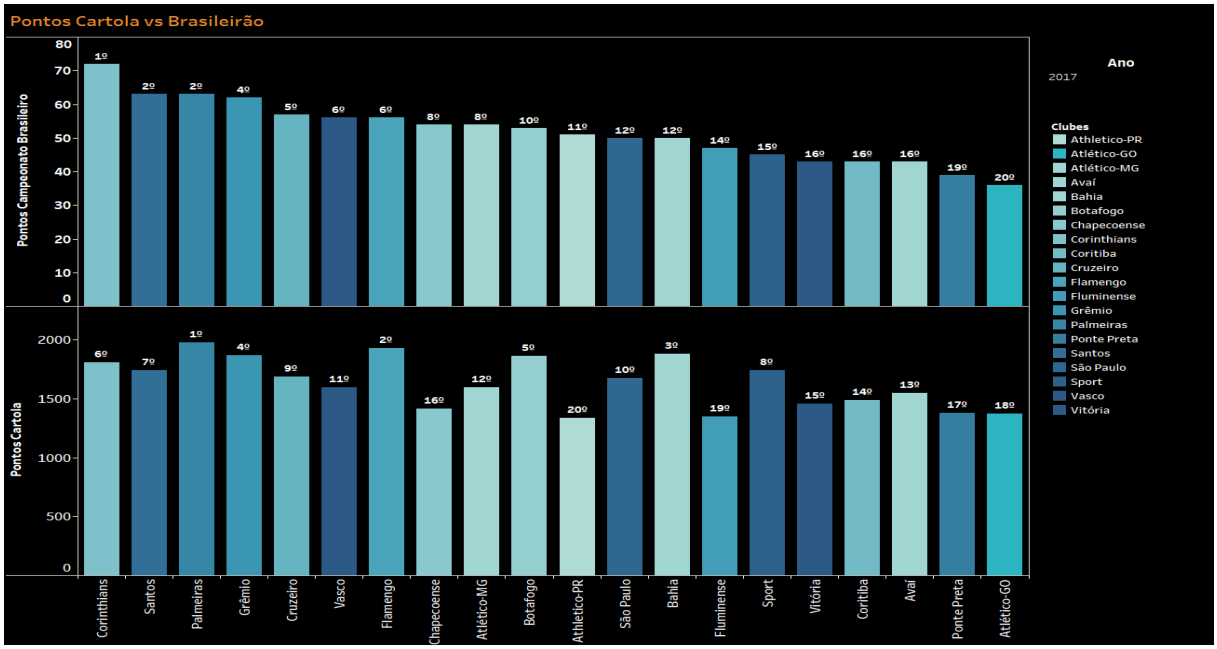


Se o objetivo do Cartoleiro for acumular cartoletas (como são chamadas as moedas do cartola), vale mais a pena escalar um goleiro que tenha um preço baixo e que costume ter um bom desempenho. Ao final do campeonato esse goleiro terá valorizado bastante (em cartoletas).

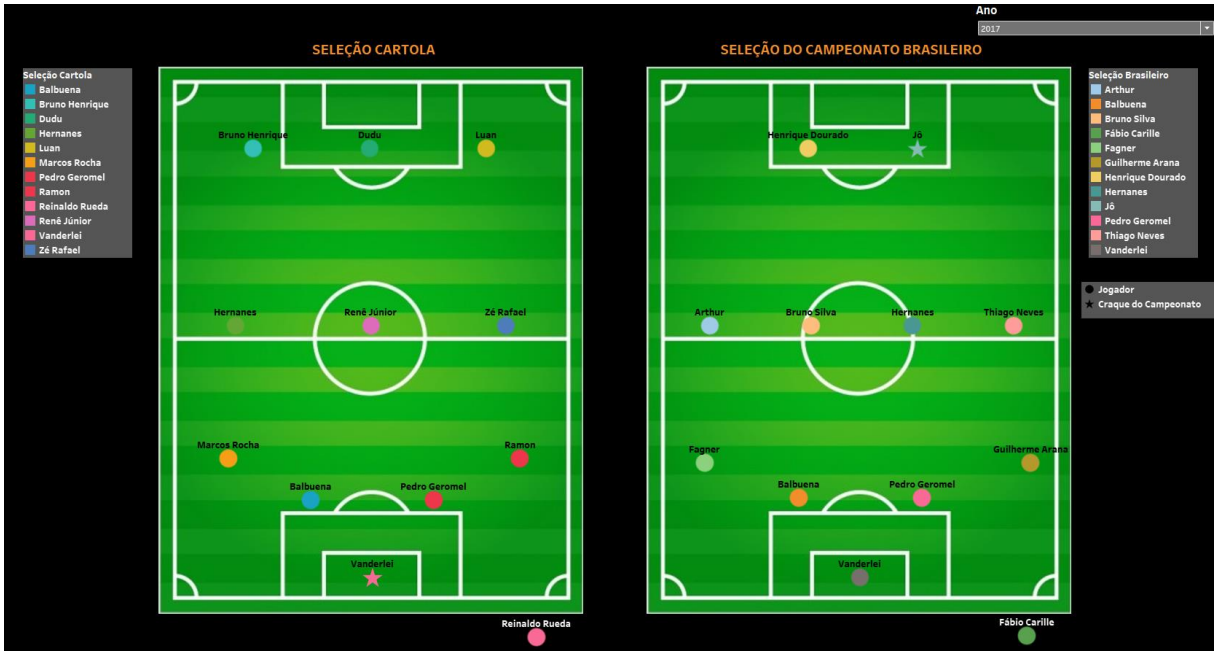


Outra importante conclusão foi que os melhores pontuadores da posição de meio campo são normalmente os que dão mais assistências e fazem gols, mas também os que roubam bastante bolas. Em alguns casos os volantes figuram em maior número entre os maiores pontuadores do que os meio campistas clássicos.

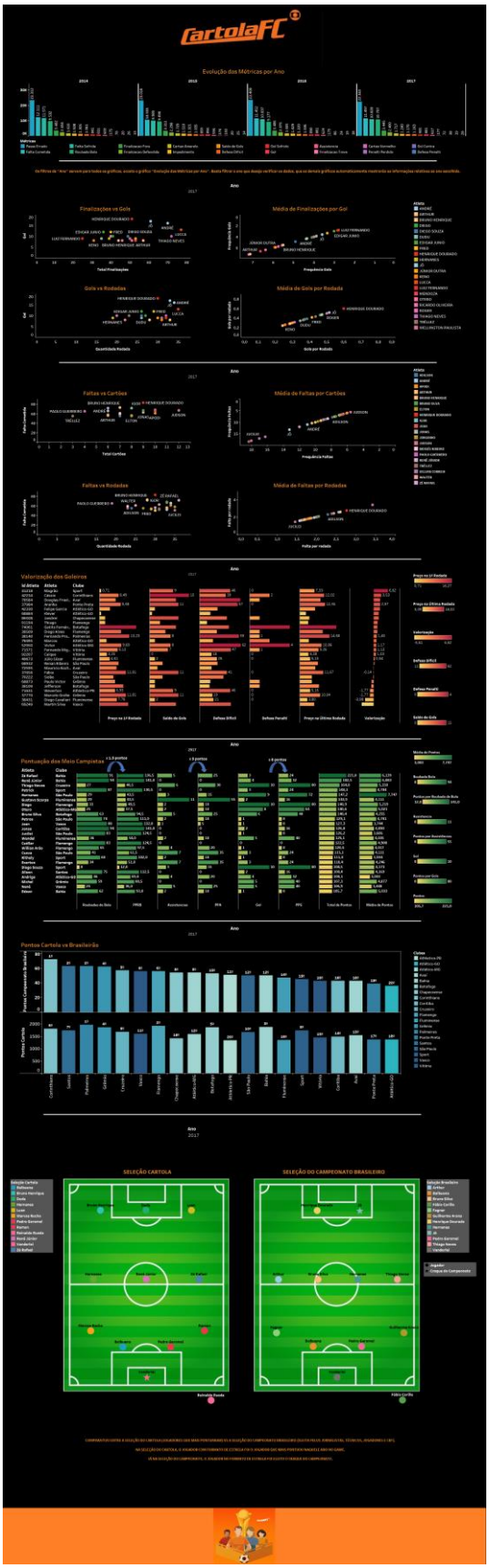
Nas nossas duas co-relações com o mundo real, concluímos que o desempenho do clube na classificação do campeonato brasileiro influencia na classificação desse mesmo clube no Cartola FC (soma de pontos de todos os jogadores do clube). Normalmente o clube do top 6 na classificação do campeonato fica no top 6 na classificação do game.



Pra finalizar, nossa última co-relação, verificamos que em média cerca de 4 atletas que compõe a seleção do Campeonato Brasileiro também compõe a seleção do Cartola FC.



Abaixo um dashboard com todas essas análises e co-relações feitas no projeto.



7. Links

No link do GoogleDrive disponibilizado abaixo estão todas as nossas bases, os códigos e os dashboards utilizados para esse projeto:

https://drive.google.com/drive/folders/11nCw2hX0Dv6gQtbtJSHiCnsBkRYtN6i2?usp=sharing

Uma cópia deste documento e o vídeo com a apresentação estão no repositório:
documento

Os dados brutos estão no repositório:
dados_brutos

Os dasboards estão no repositório:
dashboards

Os códigos estão no repositório:
fonte

Os dados tratados estão no repositório:
dados_tratados