Practico Mentoria - Analisis Exploratorio y Curación de Datos

Alumno: Flavio Olivier (Omega)

Importaciones ¶

In [1]:
```
#paquetería...

%matplotlib inline

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import scipy as sp

from sklearn import preprocessing

import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
# Set seed for reproducibility
np.random.seed(0)
```

In [3]:
```
#daterio...

path = 'https://raw.githubusercontent.com/diplodatos2019mentoria/Analisis_Exploratorio_Curacion_Datos/master/'

player_df = pd.read_csv(path + 'Datasets/football_player.csv')
team_df = pd.read_csv(path + 'Datasets/football_team.csv')
match_df = pd.read_csv(path + 'Datasets/football_match.csv')

print("Shape 'player_df' = {}".format(player_df.shape))
print("Shape 'team_df'   = {}".format(team_df.shape))
print("Shape 'match_df'  = {}".format(match_df.shape))
```

```
Shape 'player_df' = (11060, 40)
Shape 'team_df'   = (288, 22)
Shape 'match_df'  = (25979, 12)
```

1. Importacion de los datos

Calculemos el rango de fechas de los partidos

In [4]:
```
match_df['date'].max() - match_df['date'].min()
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-4-b14d418495e8> in <module>
----> 1 match_df['date'].max() - match_df['date'].min()

TypeError: unsupported operand type(s) for -: 'str' and 'str'
```

Indiquemos que la columna `date` es una fechas como indica la documentación

In [5]:
```
# TODO

match_df = pd.read_csv(path + 'Datasets/football_match.csv', parse_dates = ['date'])
match_df['date'].max() - match_df['date'].min()
```

Out[5]:
```
Timedelta('2868 days 00:00:00')
```

2. Etiquetas de variables/columnas: no usar caracteres especiales

Chequar que no haya caracteres fuera de `a-Z` , `0-9` y `_` en los nombres de columnas de los Dataframes:

- `player_df`
- `team_df`
- `match_df`

In [6]:
```
# TODO

#caracteres especiales
```

```
import re

def check(col):
    regex = re.match(r'([a-zA-Z0-9_])+$', col)

    if regex == None:
        validacion = 'INVALIDO'
    else:
        validacion = 'ok'

    return validacion

#side by side...
from IPython.display import display_html

def siamesas(*args):
    html_str = ''
    spaciador = '<table style="min-width: 30px !important;"><tr style="min-width: 30px !important; background:none !important;">
    for df in args:
        html_str += df.to_html() + spaciador

    display_html(html_str.replace('table', 'table style = "display:inline"'), raw = True)
```

In [7]:
```
#validamos nombres de columnas...

dfs = []
cols = []
estados = []

for df in ['player_df', 'team_df', 'match_df']:
    for col in pd.eval(df + '.columns'):
        #print(df, col, '->', check(col))
        dfs.append(df)
        cols.append(col)
        estados.append(check(col))

dfs = pd.DataFrame.from_dict({'tabla':dfs, 'columna':cols, 'valida':estados})
siamesas(dfs.loc[dfs.tabla == 'player_df'], dfs.loc[dfs.tabla == 'team_df'], dfs.loc[dfs.tabla == 'match_df'])
```

	tabla	columna	valida
0	player_df	player name	INVALIDO
1	player_df	birthday	ok
2	player_df	height_m	ok
3	player_df	weight_kg	ok
4	player_df	overall_rating	ok
5	player_df	potential	ok
6	player_df	preferred foot	INVALIDO
7	player_df	crossing	ok
8	player_df	finishing	ok
9	player_df	heading accuracy	INVALIDO
10	player_df	short passing	INVALIDO
11	player_df	volleys	ok

	tabla	columna	valida
40	team_df	team long name	INVALIDO
41	team_df	team short name	INVALIDO
42	team_df	buildUpPlaySpeed	ok
43	team_df	buildUpPlaySpeedClass	ok
44	team_df	buildUpPlayDribblingClass	ok
45	team_df	buildUpPlayPassing	ok
46	team_df	buildUpPlayPassingClass	ok
47	team_df	buildUpPlayPositioningClass	ok
48	team_df	chanceCreationPassing	ok
49	team_df	chanceCreationPassingClass	ok
50	team_df	chanceCreationCrossing	ok
51	team_df	chanceCreationCrossingClass	ok

	tabla	columna	valida
62	match_df	country name	INVALIDO
63	match_df	league name	INVALIDO
64	match_df	season	ok
65	match_df	stage	ok
66	match_df	date	ok
67	match_df	home team long name	INVALIDO
68	match_df	home short long name	INVALIDO
69	match_df	away team long name	INVALIDO
70	match_df	away short long name	INVALIDO
71	match_df	home team goal	INVALIDO
72	match_df	away team goal	INVALIDO
73	match_df	total goal	INVALIDO

In [8]:
```
#reeplazamos espacios por '_'...
#... y re-validamos.

player_df.columns = player_df.columns.str.replace(' ', '_')
team_df.columns = team_df.columns.str.replace(' ', '_')
match_df.columns = match_df.columns.str.replace(' ', '_')

#

dfs = []
cols = []
estados = []

for df in ['player_df', 'team_df', 'match_df']:
    for col in pd.eval(df + '.columns'):
        #print(df, col, '->', check(col))
        dfs.append(df)
        cols.append(col)
        estados.append(check(col))

dfs = pd.DataFrame.from_dict({'tabla':dfs, 'columna':cols, 'valida':estados})
siamesas(dfs.loc[dfs.tabla == 'player_df'], dfs.loc[dfs.tabla == 'team_df'], dfs.loc[dfs.tabla == 'match_df'])
```

	tabla	columna	valida
0	player_df	player_name	ok
1	player_df	birthday	ok
2	player_df	height_m	ok
3	player_df	weight_kg	ok
4	player_df	overall_rating	ok
5	player_df	potential	ok
6	player_df	preferred_foot	ok
7	player_df	crossing	ok
8	player_df	finishing	ok
9	player_df	heading_accuracy	ok
10	player_df	short_passing	ok

	tabla	columna	valida
40	team_df	team_long_name	ok
41	team_df	team_short_name	ok
42	team_df	buildUpPlaySpeed	ok
43	team_df	buildUpPlaySpeedClass	ok
44	team_df	buildUpPlayDribblingClass	ok
45	team_df	buildUpPlayPassing	ok
46	team_df	buildUpPlayPassingClass	ok
47	team_df	buildUpPlayPositioningClass	ok
48	team_df	chanceCreationPassing	ok
49	team_df	chanceCreationPassingClass	ok
50	team_df	chanceCreationCrossing	ok

	tabla	columna	valida
62	match_df	country_name	ok
63	match_df	league_name	ok
64	match_df	season	ok
65	match_df	stage	ok
66	match_df	date	ok
67	match_df	home_team_long_name	ok
68	match_df	home_short_long_name	ok
69	match_df	away_team_long_name	ok
70	match_df	away_short_long_name	ok
71	match_df	home_team_goal	ok
72	match_df	away_team_goal	ok

11	player_df	volleys	ok	51	team_df	chanceCreationCrossingClass	ok	73	match_df	total_goal	ok	

3. Agregar nuevas caracteristicas

Agregar al Dataframe `player_df` una nueva columna que sea `imc` correspondiente al **Indice de Masa Corporal**

Link:

- https://www.texasheart.org/heart-health/heart-information-center/topics/calculadora-del-indice-de-masa-corporal-imc/

In [9]:
```
# TODO

if 'IMC' in player_df.columns:
    player_df.drop('IMC', axis = 1, inplace = True)

player_df.insert(4, 'IMC', player_df['weight_kg'] / player_df['height_m']**2)
player_df.head()
```

Out[9]:

	player_name	birthday	height_m	weight_kg	IMC	overall_rating	potential	preferred_foot	crossing	finishing	...	vision	penalties	marking	standin
0	Aaron Appindangoye	1992-02-29	1.83	84.82	25.327720	63.60	67.60	right	48.60	43.60	...	53.60	47.60	63.80	
1	Aaron Cresswell	1989-12-15	1.70	66.22	22.913495	66.97	74.48	left	70.79	49.45	...	57.45	53.12	69.39	
2	Aaron Doran	1991-05-13	1.70	73.94	25.584775	67.00	74.19	right	68.12	57.92	...	69.38	60.54	22.04	
3	Aaron Galindo	1982-05-08	1.83	89.81	26.817761	69.09	70.78	right	57.22	26.26	...	53.78	41.74	70.61	
4	Aaron Hughes	1979-11-08	1.83	69.85	20.857595	73.24	74.68	right	45.08	38.84	...	46.48	52.96	77.60	

5 rows × 41 columns

4. Tratar valores faltantes

Veamos cuantos valores nulos tenemos

In [10]:
```
player_missing_values_count = player_df.isnull().sum()

player_missing_values_count[player_missing_values_count > 0]
```

Out[10]:
```
volleys           478
curve             478
agility           478
balance           478
jumping           478
vision            478
sliding_tackle    478
dtype: int64
```

In [11]:
```
team_missing_values_count = team_df.isnull().sum()

team_missing_values_count[team_missing_values_count > 0]
```

Out[11]:
```
Series([], dtype: int64)
```

In [12]:
```
match_missing_values_count = match_df.isnull().sum()

match_missing_values_count[match_missing_values_count > 0]
```

Out[12]:
```
Series([], dtype: int64)
```

Algunas tecnicas para tratar los *missing values*:

- **Eliminar** muestras o variables que tienen datos faltantes.
- **Imputar** los valores perdidos, es decir, sustituirlos por estimaciones por ejemplo la `media`, la `moda` ó usando `KNN`.

A) Analizar si es conveniente **Eliminar** las muestras o variables con datos faltantes del Dataframe `player_df`.

B) Aplicar la **Imputacion** usando la `media` o `moda` sobre las columnas con *missing values* del Dataframe `player_df`.

Hint:

- Para la imputacion usando la `media`, `moda` ver el siguiente link:
  https://pandas.pydata.org/pandas-docs/stable/user_guide/missing_data.html

¿Eliminar los *missing values*? Justificar

In [13]:
```
#registros NaN's...

player_df_NaNs = player_df[list(player_df.isnull().sum()[player_df.isnull().sum() > 0].index)][player_df.volleys.isnull()]
player_df_NaNs
```

Out[13]:

	volleys	curve	agility	balance	jumping	vision	sliding_tackle
25	NaN	NaN	NaN	NaN	NaN	NaN	NaN
30	NaN	NaN	NaN	NaN	NaN	NaN	NaN
31	NaN	NaN	NaN	NaN	NaN	NaN	NaN
83	NaN	NaN	NaN	NaN	NaN	NaN	NaN

85	NaN	NaN	NaN	NaN	NaN	NaN	NaN
175	NaN	NaN	NaN	NaN	NaN	NaN	NaN
190	NaN	NaN	NaN	NaN	NaN	NaN	NaN
203	NaN	NaN	NaN	NaN	NaN	NaN	NaN
253	NaN	NaN	NaN	NaN	NaN	NaN	NaN
275	NaN	NaN	NaN	NaN	NaN	NaN	NaN
289	NaN	NaN	NaN	NaN	NaN	NaN	NaN

In [14]:
```
#porcentaje de registros NaN sobre player_df...

print('Porcentual de valores nulos sobre el universo player_df: {:.2f}%'.format(player_df_NaNs.shape[0] / player_df.shape[0] * 1
```

```
Porcentual de valores nulos sobre el universo player_df: 4.32%
```

La opción de eliminar los registros puede ser viable, representan menos del 5% del universo... aunque podríamos intentar una estrategia que evitara esto aplicando K vecinos más cercanos.

In [15]:
```
#KNN...
from sklearn.neighbors import KNeighborsRegressor

col_para_eliminar = list(player_df_NaNs.columns)
idx_para_recordar = list(player_df_NaNs.index)

player_df_notnan = player_df.dropna()
X_train = player_df_notnan.drop(col_para_eliminar, axis = 1).select_dtypes(include ='float')
y_train = player_df_notnan[col_para_eliminar]
X_test = player_df.loc[list(player_df_NaNs.index)].dropna(axis = 1).select_dtypes(include ='float')

#entrenamos regresor...
resultado = []
reg = KNeighborsRegressor(5, weights = 'distance')
for columna in col_para_eliminar:
    modelo = reg.fit(X_train, y_train[columna])

    #predecimos...
    print('predicción de: '+columna)
    prediccion = modelo.predict(X_test)
    print(prediccion)
    print()

    #contatenamos resultados...
    resultado.append(prediccion)
```

```
predicción de: volleys
[53.40793977 50.61075748 54.99453834 59.53346925 68.51797819 49.19050185
 61.08190688 72.94302628 44.7098002  39.64321854 19.51748853 40.44893264
 55.31948659 43.86317554 56.28820827 57.12099293 35.11976765 61.43988715
 53.56577713 49.56218098 30.58393313 44.84964956 66.5965974  12.52207371
 75.3568084  59.55352549 33.26576001 28.84352754 54.8503441  16.65282903
 49.0719208  46.46288288 40.6041672  55.52337536 54.69569119 44.56786323
 38.60307843 36.71054538 26.58060526 58.24994044 21.42603728 53.73064116
 41.23631995 18.26670938 28.93278529 49.95401419 47.70285197 28.17612863
 48.23062693 54.49322354 62.52098399 69.84748667 38.6978352  51.48221874
 35.54364789 55.90882609 30.7658424  42.49816745 22.2386005  53.53646406
 57.44544116 56.11187984 29.50484588 47.17939226 13.89244813 60.5996222
 32.69991184 48.90931457 49.4400109  57.56184906 56.44831847 52.38952105
 55.03616909 45.56854631 64.51968631 28.71763552 48.01889941 48.44545959
 73.01896133 13.35137627 26.39078286 53.33401937 52.97051628 34.32011357
 50.81150336 36.1163914  33.67799338 36.79311099 45.16019272 14.00075582
 44.28867454 38.97376005 37.8858054  44.98131235 71.73483014 43.23100401
 38.28639594 55.90300739 26.98389396 49.72341497 51.93029677 56.49613211
 44.29244238 32.9529299  33.23118512 49.6893257  41.97430741 75.12465588
```

In [16]:
```
#generamos tabla predicciones de las columnas NaN...

predicciones = pd.DataFrame(resultado).T
predicciones.columns = col_para_eliminar
predicciones.index = idx_para_recordar
predicciones
```

Out[16]:

	volleys	curve	agility	balance	jumping	vision	sliding_tackle
25	53.407940	56.621197	66.723442	68.785028	62.870740	65.354919	57.839723
30	50.610757	55.785303	62.172254	72.509349	71.343057	66.938158	72.487838
31	54.994538	52.752182	68.740927	62.502517	63.004874	57.403377	20.607705
83	59.533469	43.444147	58.437773	64.531549	63.726807	50.165564	24.565158
85	68.517978	63.203751	71.612670	69.437473	68.118260	65.457800	30.304778
175	49.190502	50.607659	58.419065	58.363384	57.588054	53.684221	31.923268
190	61.081907	57.058733	74.786733	63.300443	69.626405	62.912750	20.264091
203	72.943026	73.426464	74.031616	72.676383	68.977321	74.716895	25.440000
253	44.709800	46.971950	62.039941	68.802366	65.833661	67.914943	61.231873
275	39.643219	40.254281	51.580657	67.238720	65.534813	61.516999	60.799294
289	19.517489	22.087701	59.921240	64.794621	69.005924	57.194844	23.787594

In [17]:
```
#mapeamos sobre tabla original reemplazando valores NaN's...

player_df.update(predicciones)
player_df
```

Out[17]:

	player_name	birthday	height_m	weight_kg	IMC	overall_rating	potential	preferred_foot	crossing	finishing	...	vision	penalties	markin
0	Aaron Appindangoye	1992-02-29	1.83	84.82	25.327720	63.60	67.60	right	48.60	43.60	...	53.600000	47.60	63.
1	Aaron Cresswell	1989-12-15	1.70	66.22	22.913495	66.97	74.48	left	70.79	49.45	...	57.450000	53.12	69.
2	Aaron Doran	1991-05-13	1.70	73.94	25.584775	67.00	74.19	right	68.12	57.92	...	69.380000	60.54	22.
3	Aaron Galindo	1982-05-08	1.83	89.81	26.817761	69.09	70.78	right	57.22	26.26	...	53.780000	41.74	70.
4	Aaron Hughes	1979-11-08	1.83	69.85	20.857595	73.24	74.68	right	45.08	38.84	...	46.480000	52.96	77.
5	Aaron Hunt	1986-09-04	1.83	73.03	21.807161	77.26	80.15	left	73.89	72.81	...	79.960000	75.59	31.
6	Aaron Kuhl	1996-01-30	1.73	66.22	22.125697	60.57	76.00	right	47.57	31.57	...	60.000000	41.57	51.

In [18]:
```
#validamos existencia de nulos...

player_missing_values_count = player_df.isnull().sum()
player_missing_values_count[player_missing_values_count > 0]
```

Out[18]:
```
Series([], dtype: int64)
```

Imputacion usando Media y Moda

In [19]:
```
#estimamos media de las columnas NaN's...

player_df[player_df_NaNs.columns].mean()
```

Out[19]:
```
volleys           47.078695
curve             50.324995
agility           64.304038
balance           64.534801
jumping           66.010144
vision            56.074348
sliding_tackle    47.189009
dtype: float64
```

In [20]:
```
#imputamos media a los valores NaN's...

player_df[player_df_NaNs.columns] = player_df[player_df_NaNs.columns].fillna(player_df[player_df_NaNs.columns].mean())

player_missing_values_count = player_df.isnull().sum()
player_missing_values_count[player_missing_values_count > 0]
```

Out[20]:
```
Series([], dtype: int64)
```

5. Normalizacion de columnas

Primero que todo la notación:

- $x = [x_1, x_2, \ldots, x_n]$
- μ: Media
- σ: Desviacion Estandar

Ahora normalizaremos algunas de las columnas del Dataframe, para ello usaremos dos tipos de normalizacion:

- Min-Max:

$$z_i = \frac{x_i - min(x)}{max(x) - min(x)}$$

- Z-score

$$z_i = \frac{x_i - \mu}{\sigma}$$

Normalizar la columna `crossing` usando **Min-Max**.

Normalizar la columna `short_passing` usando **Z-score**.

Hints:

- https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html#sklearn.preprocessing.MinMaxScaler
- https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.zscore.html

Min-Max

In [21]:
```
# TODO
from sklearn.preprocessing import MinMaxScaler

pre = player_df['crossing'].head().copy()
minmaxscaler = MinMaxScaler()
player_df['crossing'] = minmaxscaler.fit_transform(player_df['crossing'].values.reshape(-1, 1))
pos = player_df['crossing'].head().copy()

siamesas(pd.DataFrame(pre).rename(columns={'crossing': 'crossing(pre)'}), pd.DataFrame(pos).rename(columns={'crossing': 'crossin
```

	crossing(pre)
0	48.60

	crossing(pos)
0	0.511036

1	70.79	1	0.777231
2	68.12	2	0.745202
3	57.22	3	0.614443
4	45.08	4	0.468810

Z-score

In [22]:
```
# TODO
from scipy import stats

pre = player_df['short_passing'].head().copy()
player_df['short_passing'] = stats.zscore(player_df['short_passing'])
pos = player_df['short_passing'].head().copy()

siamesas(pd.DataFrame(pre).rename(columns={'short_passing': 'short_passing(pre)'}), pd.DataFrame(pos).rename(columns={'short_pas
```

	short_passing(pre)
0	60.60
1	62.27
2	65.12
3	64.70
4	64.76

	short_passing(pos)
0	0.017238
1	0.140868
2	0.351853
3	0.320761
4	0.325202

6. Codificar variables

> Las variables categóricas deben ser etiquetadas como variables numéricas, no como cadenas.

Codificar la variable `country_name` del Dataframe `match_df`

In [23]:
```
# TODO

match_df.country_name.value_counts()
```

Out[23]:
```
France         3040
Spain          3040
England        3040
Italy          3017
Germany        2448
Netherlands    2448
Portugal       2052
Poland         1920
Scotland       1824
Belgium        1728
Switzerland    1422
Name: country_name, dtype: int64
```

More about preprocessing in:

- https://scikit-learn.org/stable/modules/classes.html#module-sklearn.preprocessing

In [24]:
```
#
from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()
xx = le.fit_transform(match_df['country_name'])

if 'country_name_le' in match_df.columns:
    match_df.drop('country_name_le', axis = 1, inplace = True)

match_df.insert(0, 'country_name_le', xx)

set(zip(le.classes_, le.transform(le.classes_)))
```

Out[24]:
```
{('Belgium', 0),
 ('England', 1),
 ('France', 2),
 ('Germany', 3),
 ('Italy', 4),
 ('Netherlands', 5),
 ('Poland', 6),
 ('Portugal', 7),
 ('Scotland', 8),
 ('Spain', 9),
 ('Switzerland', 10)}
```

In [25]:
```
match_df
```

Out[25]:

	country_name_le	country_name	league_name	season	stage	date	home_team_long_name	home_short_long_name	away_team_long_name	awa
0	0	Belgium	Belgium Jupiler League	2008/2009	1	2008-08-17	KRC Genk	GEN	Beerschot AC	
1	0	Belgium	Belgium Jupiler League	2008/2009	1	2008-08-16	SV Zulte-Waregem	ZUL	Sporting Lokeren	
2	0	Belgium	Belgium Jupiler League	2008/2009	1	2008-08-16	KSV Cercle Brugge	CEB	RSC Anderlecht	

3	0	Belgium	Belgium Jupiler League	2008/2009	1	2008-08-17	KAA Gent	GEN	RAEC Mons
4	0	Belgium	Belgium Jupiler League	2008/2009	1	2008-08-16	FCV Dender EH	DEN	Standard de Liège

In [ ]: