Practico Mentoria - Analisis Exploratorio y Curación de Datos

Autor: Melania Omonte

Importaciones

In [181]:

```
%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 [150]:

```
# Seteamos una semilla para Reproducibilidad
np.random.seed(0)
```

Carga de los Datasets

In [151]:

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

Exploremos un poco los Datasets

Players Dataset

In [152]:

```
print("Shape 'player_df' = {}".format(player_df.shape))
player_df.sample(5)
```

```
Shape 'player_df' = (11060, 40)
```

Out[152]:

	player name	birthday	height_m	weight_kg	overall_rating	potential	preferred foot	crossing	finishing	heading accuracy	...	vision	penalt
1534	Carlos Acuna	1988-06-23	1.78	71.21	67.33	71.81	right	51.52	67.43	68.86	...	44.00	66
7238	Max Christiansen	1996-09-25	1.88	83.91	64.09	74.45	right	47.73	37.73	60.82	...	56.45	47
10999	Zakaria M'Sila	1992-04-06	1.78	74.84	59.00	65.10	left	57.30	50.90	50.20	...	54.30	54
2669	Dimitrija Lazarevski	1982-09-23	1.78	74.84	59.00	61.00	left	51.00	38.00	54.00	...	NaN	61

	player name	birthday	height_cm	weight_kg	overall_rating	potential	preferred foot	crossing	finishing	heading accuracy	...	vision	penalt
1403	Bruma	1994-10-24	170	66	75.00	80.00	right	57.00	54.00	51.09	...	68.00	71

5 rows × 40 columns

Team Dataset

In [153]:

```
print("Shape 'team_df'   = {}".format(team_df.shape))
team_df.sample(5)
```

```
Shape 'team_df'   = (288, 22)
```

Out[153]:

	team long name	team short name	buildUpPlaySpeed	buildUpPlaySpeedClass	buildUpPlayDribblingClass	buildUpPlayPassing	buildUpPlayPassing
185	Lechia Gdańsk	LGD	50.83	Balanced	Little	48.33	
222	FC Penafiel	PEN	54.00	Balanced	Normal	39.00	
198	Pogoń Szczecin	POG	55.67	Balanced	Little	42.00	
197	Podbeskidzie Bielsko-Biała	POD	62.00	Balanced	Little	58.50	
173	Excelsior	EXC	57.67	Balanced	Little	60.00	

5 rows × 22 columns

Match Dataset

In [154]:

```
print("Shape 'match_df'  = {}".format(match_df.shape))
match_df.sample(5)
```

```
Shape 'match_df'  = (25979, 12)
```

Out[154]:

	country name	league name	season	stage	date	home team long name	home short long name	away team long name	away short long name	home team goal	away team goal	total goal
12042	Italy	Italy Serie A	2012/2013	35	2013-05-05 00:00:00	Catania	CAT	Siena	SIE	3	0	3
8575	Germany	Germany 1. Bundesliga	2010/2011	25	2011-03-05 00:00:00	VfB Stuttgart	STU	FC Schalke 04	S04	1	0	1
9067	Germany	Germany 1. Bundesliga	2012/2013	12	2012-11-17 00:00:00	Eintracht Frankfurt	EFR	FC Augsburg	AUG	4	2	6
13165	Italy	Italy Serie A	2015/2016	34	2016-04-21 00:00:00	Milan	ACM	Carpi	CAP	0	0	0
14904	Netherlands	Netherlands Eredivisie	2013/2014	2	2013-08-10 00:00:00	Heracles Almelo	HER	PEC Zwolle	ZWO	1	3	4

Exploremos un poco los Datasets y sus correspondientes Tipos

Players Dtypes

Players Dtypes

In [155]:

```
player_df.dtypes
```

Out[155]:

```
player name            object
birthday               object
height_m              float64
weight_kg             float64
overall_rating        float64
potential             float64
preferred foot         object
crossing              float64
finishing             float64
heading accuracy      float64
short passing         float64
volleys               float64
dribbling             float64
curve                 float64
free kick accuracy    float64
long passing          float64
ball control          float64
acceleration          float64
sprint speed          float64
agility               float64
reactions             float64
balance               float64
shot power            float64
jumping               float64
stamina               float64
strength              float64
long shots            float64
aggression            float64
interceptions         float64
positioning           float64
vision                float64
penalties             float64
marking               float64
standing tackle       float64
sliding tackle        float64
gk_diving             float64
gk_handling           float64
gk_kicking            float64
gk_positioning        float64
gk_reflexes           float64
dtype: object
```

Match Dtypes

In [156]:

```
match_df.dtypes
```

Out[156]:

```
country name            object
league name             object
season                  object
stage                    int64
date                    object
home team long name     object
home short long name    object
away team long name     object
away short long name    object
home team goal           int64
away team goal           int64
total goal               int64
dtype: object
```

Team Dtypes

In [157]:

```
team_df.dtypes
```

Out[157]:

```
team long name                    object
team short name                   object
buildUpPlaySpeed                 float64
buildUpPlaySpeedClass             object
buildUpPlayDribblingClass         object
buildUpPlayPassing               float64
buildUpPlayPassingClass           object
buildUpPlayPositioningClass       object
chanceCreationPassing            float64
chanceCreationPassingClass        object
chanceCreationCrossing           float64
chanceCreationCrossingClass       object
chanceCreationShooting           float64
chanceCreationShootingClass       object
chanceCreationPositioningClass    object
defencePressure                  float64
defencePressureClass              object
defenceAggression                float64
defenceAggressionClass            object
defenceTeamWidth                 float64
defenceTeamWidthClass             object
defenceDefenderLineClass          object
dtype: object
```

---

1. Importacion de los datos

Calculemos el rango de fechas de los partidos

Antes de calcular el rango de fechas de los partidos, debemos validar que tipo de objeto es la fecha

In [158]:

```
match_df.dtypes['date']
```

Out[158]:

```
dtype('O')
```

Como la fecha es un campo del tipo object, no podremos calcular el rango solicitado, por lo tanto tendremos que cambiar el tipo, asi podemos generar el valor solicitado.

Modificamos el tipo "date", para poder calcular el rango

In [159]:

```
match_df2 = pd.read_csv("./Datasets/football_match.csv", parse_dates=["date"])
```

Visualizamos que haya cambiado el tipo "date"

In [160]:

```
match_df2.dtypes['date']
```

Out[160]:

Out[160]:

```
dtype('<M8[ns]')
```

Validamos que se cambio el tipo a datetime64[ns]

Realizamos la diferencia, para poder calcular el rango solicitado

In [161]:

```
match_df2['date'].max() - match_df2['date'].min()
```

Out[161]:

```
Timedelta('2868 days 00:00:00')
```

Rta: El rango de fechas entre partidos es 2868 dias.

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`

Exploramos los Datasets y validamos que no hayan caracteres fuera de lo solicitado

Match DataSet

In [162]:

```
match_df.columns[~match_df.columns.str.match(r'^(\w+)$')]
```

Out[162]:

```
Index(['country name', 'league name', 'home team long name',
       'home short long name', 'away team long name', 'away short long name',
       'home team goal', 'away team goal', 'total goal'],
      dtype='object')
```

Chequeamos que existen varias columnas que tienen caracteres fuera de "a-Z, 0-9 y _" en el dataset match.

Player DataSet

In [163]:

```
player_df.columns[~player_df.columns.str.match(r'^(\w+)$')]
```

Out[163]:

```
Index(['player name', 'preferred foot', 'heading accuracy', 'short passing',
       'free kick accuracy', 'long passing', 'ball control', 'sprint speed',
       'shot power', 'long shots', 'standing tackle', 'sliding tackle'],
      dtype='object')
```

Chequeamos que existen varias columnas que tienen caracteres fuera de "a-Z, 0-9 y _" en el dataset player.

Team DataSet

In [164]:

```
team_df.columns[~team_df.columns.str.match(r'^(\w+)$')]
```

Out[164]:

```
Index(['team long name', 'team short name'], dtype='object')
```

Chequeamos que existen 2 columnas que tienen caracteres fuera de "a-Z, 0-9 y _" en el dataset team.

Reemplazamos los valores fuera de "a-Z, 0-9 y _" en el dataset team

In [165]:

```
team_df.columns = team_df.columns.str.replace(' ', '_')
team_df.head()
```

Out[165]:

	team_long_name	team_short_name	buildUpPlaySpeed	buildUpPlaySpeedClass	buildUpPlayDribblingClass	buildUpPlayPassing	build
0	KRC Genk	GEN	56.33	Balanced	Little	44.33	
1	Beerschot AC	BAC	46.00	Balanced	Little	41.50	
2	SV Zulte-Waregem	ZUL	55.50	Balanced	Little	52.67	
3	Sporting Lokeren	LOK	64.00	Balanced	Little	53.50	
4	KSV Cercle Brugge	CEB	53.67	Balanced	Little	44.17	

5 rows × 22 columns

Validamos que se hayan reemplazado bien los campos en el dataset Team

In [166]:

```
team_df.columns[~team_df.columns.str.match(r'^(\w+)$')]
```

Out[166]:

```
Index([], dtype='object')
```

Validamos que se reemplazaron exitosamente los campos, ya que la consulta anterior no nos devuelve ningun campo.

Reemplazamos los valores fuera de "a-Z, 0-9 y _" en el dataset Player

In [167]:

```
player_df.columns = player_df.columns.str.replace(' ', '_')
player_df.head()
```

Out[167]:

	player_name	birthday	height_m	weight_kg	overall_rating	potential	preferred_foot	crossing	finishing	heading_accuracy	...	vis
0	Aaron Appindangoye	1992-02-29	1.83	84.82	63.60	67.60	right	48.60	43.60	70.60	...	53
1	Aaron Cresswell	1989-12-15	1.70	66.22	66.97	74.48	left	70.79	49.45	52.94	...	57
2	Aaron Doran	1991-05-13	1.70	73.94	67.00	74.19	right	68.12	57.92	58.69	...	69

	player_name	birthday	height_cm	weight_kg	overall_rating	potential	preferred_foot	crossing	finishing	heading_accuracy	...	vi
3	Aaron Galindo	1982-05-08										
4	Aaron Hughes	1979-11-08	1.83	69.85	73.24	74.68	right	45.08	38.84	73.04	...	46

5 rows × 40 columns

Validamos que se hayan reemplazado bien los campos en el dataset Player

In [168]:

```
player_df.columns[~player_df.columns.str.match(r'^(\w+)$')]
```

Out[168]:

```
Index([], dtype='object')
```

Validamos que se reemplazaron exitosamente los campos, ya que la consulta anterior no nos devuelve ningun campo.

Reemplazamos los valores fuera de "a-Z, 0-9 y _" en el dataset Match

In [169]:

```
match_df.columns = match_df.columns.str.replace(' ', '_')
match_df.head()
```

Out[169]:

	country_name	league_name	season	stage	date	home_team_long_name	home_short_long_name	away_team_long_name	av
0	Belgium	Belgium Jupiler League	2008/2009	1	2008-08-17 00:00:00	KRC Genk	GEN	Beerschot AC	
1	Belgium	Belgium Jupiler League	2008/2009	1	2008-08-16 00:00:00	SV Zulte-Waregem	ZUL	Sporting Lokeren	
2	Belgium	Belgium Jupiler League	2008/2009	1	2008-08-16 00:00:00	KSV Cercle Brugge	CEB	RSC Anderlecht	
3	Belgium	Belgium Jupiler League	2008/2009	1	2008-08-17 00:00:00	KAA Gent	GEN	RAEC Mons	
4	Belgium	Belgium Jupiler League	2008/2009	1	2008-08-16 00:00:00	FCV Dender EH	DEN	Standard de Liège	

Validamos que se hayan reemplazado bien los campos en el dataset Match

In [170]:

```
match_df.columns[~match_df.columns.str.match(r'^(\w+)$')]
```

Out[170]:

```
Index([], dtype='object')
```

Validamos que se reemplazaron exitosamente los campos, ya que la consulta anterior no nos devuelve ningun campo.

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 [171]:

```
from sklearn import preprocessing
player_df_mod = pd.read_csv('./Datasets/football_player.csv')
player_df_mod.head()
```

Out[171]:

	player name	birthday	height_m	weight_kg	overall_rating	potential	preferred foot	crossing	finishing	heading accuracy	...	vision	penalties
0	Aaron Appindangoye	1992-02-29	1.83	84.82	63.60	67.60	right	48.60	43.60	70.60	...	53.60	47.60
1	Aaron Cresswell	1989-12-15	1.70	66.22	66.97	74.48	left	70.79	49.45	52.94	...	57.45	53.12
2	Aaron Doran	1991-05-13	1.70	73.94	67.00	74.19	right	68.12	57.92	58.69	...	69.38	60.54
3	Aaron Galindo	1982-05-08	1.83	89.81	69.09	70.78	right	57.22	26.26	69.26	...	53.78	41.74
4	Aaron Hughes	1979-11-08	1.83	69.85	73.24	74.68	right	45.08	38.84	73.04	...	46.48	52.96

5 rows × 40 columns

Saco el Cuadrado de la altura, para poder sacar el IMC. Una vez definido el cuadrado, calculo el IMC

In [172]:

```
altura_cuadrado=player_df_mod['height_m']**2
def imc(peso,altura):
  #  return peso / (altura*altura)
    return peso / (altura_cuadrado)
```

Agrego la columna IMC en el dataFrame Player

In [173]:

```
player_df_mod['IMC'] = player_df_mod.apply(lambda x: imc(x.weight_kg, x.height_m), axis=1)
player_df_mod.describe()
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
~\AppData\Local\Continuum\anaconda3\lib\site-packages\pandas\core\indexes\base.py in get_loc(self,
key, method, tolerance)
   2656             try:
-> 2657                 return self._engine.get_loc(key)
   2658             except KeyError:

pandas/_libs/index.pyx in pandas._libs.index.IndexEngine.get_loc()

pandas/_libs/index.pyx in pandas._libs.index.IndexEngine.get_loc()

pandas/_libs/hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashTable.get_item()

pandas/_libs/hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashTable.get_item()

KeyError: 'IMC'

During handling of the above exception, another exception occurred:

KeyError                                  Traceback (most recent call last)
~\AppData\Local\Continuum\anaconda3\lib\site-packages\pandas\core\internals\managers.py in
set(self, item, value)
   1052         try:
-> 1053             loc = self.items.get loc(item)
```

```
   1054         except KeyError:

~\AppData\Local\Continuum\anaconda3\lib\site-packages\pandas\core\indexes\base.py in get_loc(self,
key, method, tolerance)
   2658             except KeyError:
-> 2659                 return self._engine.get_loc(self._maybe_cast_indexer(key))
   2660         indexer = self.get_indexer([key], method=method, tolerance=tolerance)

pandas/_libs/index.pyx in pandas._libs.index.IndexEngine.get_loc()

pandas/_libs/index.pyx in pandas._libs.index.IndexEngine.get_loc()

pandas/_libs/hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashTable.get_item()

pandas/_libs/hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashTable.get_item()

KeyError: 'IMC'

During handling of the above exception, another exception occurred:

ValueError                                Traceback (most recent call last)
<ipython-input-173-d9a6d2e6b330> in <module>
      1
----> 2 player_df_mod['IMC'] = player_df_mod.apply(lambda x: imc(x.weight_kg, x.height_m), axis=1)
      3 player_df_mod.describe()

~\AppData\Local\Continuum\anaconda3\lib\site-packages\pandas\core\frame.py in __setitem__(self,
key, value)
   3368         else:
   3369             # set column
-> 3370             self._set_item(key, value)
   3371
   3372     def _setitem_slice(self, key, value):

~\AppData\Local\Continuum\anaconda3\lib\site-packages\pandas\core\frame.py in _set_item(self, key,
value)
   3444         self._ensure_valid_index(value)
   3445         value = self._sanitize_column(key, value)
-> 3446         NDFrame._set_item(self, key, value)
   3447
   3448         # check if we are modifying a copy

~\AppData\Local\Continuum\anaconda3\lib\site-packages\pandas\core\generic.py in _set_item(self, ke
y, value)
   3170
   3171     def _set_item(self, key, value):
-> 3172         self._data.set(key, value)
   3173         self._clear_item_cache()
   3174

~\AppData\Local\Continuum\anaconda3\lib\site-packages\pandas\core\internals\managers.py in
set(self, item, value)
   1054         except KeyError:
   1055             # This item wasn't present, just insert at end
-> 1056             self.insert(len(self.items), item, value)
   1057             return
   1058

~\AppData\Local\Continuum\anaconda3\lib\site-packages\pandas\core\internals\managers.py in insert(
self, loc, item, value, allow_duplicates)
   1156
   1157         block = make_block(values=value, ndim=self.ndim,
-> 1158                            placement=slice(loc, loc + 1))
   1159
   1160         for blkno, count in _fast_count_smallints(self._blknos[loc:]):

~\AppData\Local\Continuum\anaconda3\lib\site-packages\pandas\core\internals\blocks.py in
make_block(values, placement, klass, ndim, dtype, fastpath)
   3093         values = DatetimeArray._simple_new(values, dtype=dtype)
   3094
-> 3095     return klass(values, ndim=ndim, placement=placement)
   3096
   3097

~\AppData\Local\Continuum\anaconda3\lib\site-packages\pandas\core\internals\blocks.py in __init__(
self, values, placement, ndim)
     85             raise ValueError(
```

```
     86                 'Wrong number of items passed {val}, placement implies '
---> 87                 '{mgr}'.format(val=len(self.values), mgr=len(self.mgr_locs)))
     88
     89     def _check_ndim(self, values, ndim):

ValueError: Wrong number of items passed 11060, placement implies 1
```

Grafico de distribucion de IMC

In [ ]:

```
plt.figure(figsize=(10,6))

## Grafico la distribucion de Masa Corporal"
sns.distplot(player_df_mod.IMC, kde=True, bins=5, label='IMC')

plt.ylabel('Densidad de Masa Corporal')
plt.legend()
```

Visualizamos los valores atipicos para el calculo realizado de IMC

In [ ]:

```
#Para observar valores atípicos visualizamos el gráfico de caja...
plt.figure(figsize=(10,6))
data = player_df_mod[['IMC']]
sns.boxplot(data = data, orient="h", palette="coolwarm")
#sns.stripplot(data=data, color='black')
plt.show()
```

4. Tratar valores faltantes

Veamos cuantos valores nulos tenemos

In [ ]:

```
player_missing_values_count = player_df.isnull().sum()

player_missing_values_count[player_missing_values_count > 0]
```

Tenemos 478 valores nulos en 7 columnas del DataFrame Player

In [ ]:

```
len( player_df.dropna())/len(player_df)
```

In [ ]:

```
len(player_df.dropna(subset=['volleys']))/len(player_df)
```

In [ ]:

```
player_df[player_df.volleys.isnull()]
```

Eliminamos los valores nulos

In [ ]:

```
player_df_mod = player_df_mod.dropna(subset=['volleys'])
```

In [ ]:

```
player_df_mod.describe()
```

Validamos que se hayan validado esos valores nulos

In [ ]:

```
player_missing_values_count = player_df_mod.isnull().sum()

player_missing_values_count[player_missing_values_count > 0]
```

Validamos que se eliminaron exitosamente los campos nulos, ya que no devuelve ningun valor la consulta realizada.

In [ ]:

```
player_df[player_df.volleys.isnull()]
```

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

Elimino los valores en un dataset modificado

Para eliminar los valores missing, debemos realizar un analisis sobre esos campos.Las columnas con valores missing son: volleys 478 curve 478 agility 478 balance 478 jumping 478 vision 478 sliding_tackle 478

Una de las soluciones para resolver este problema podria ser llenar estos los valores con ceros. Pero esta solucion no seria la mas optima, porque por ejemplo para un jugador, tendriamos que su agilidad es cero, y no seria representativo.

Otra de las opciones para resolver este problema es decidir eliminar estos valores, suponiendo que los valores missing pueden ser errores, de esta manera subsanamos dicho error.

Otra de las opciones es la imputacion de la Media o Moda, segun corresponda, analisis detallado mas abajo.

Imputacion usando Media y Moda

Reemplazo de Valores Faltantes usando la moda

In [ ]:

```
# Rellenamos usando la Moda
# player_df.fillna(player_df.mode(), inplace=True)

player_df_reemplazo_nan_moda = player_df

for column in ['volleys','agility','curve','balance','jumping','vision','sliding_tackle']:
    player_df_reemplazo_nan_moda[column].fillna(player_df_reemplazo_nan_moda[column].mode()[0], inp
lace=True)

player_df_reemplazo_nan_moda
```

Validamos que se hayan reemplazado bien los valores y que no hayan valores missing:

In [ ]:

```
player_missing_values_count = player_df.isnull().sum()

player_missing_values_count[player_missing_values_count > 0]
```

Se comprueba exitosamente que no hay valores missing, una vez que se reemplazaron los datos por su moda.

Reemplazo de Valores Faltantes usando la Media

In [ ]:

```
# Rellenamos usando la Moda
player_df_reemplazo_nan_media = player_df
player_df_reemplazo_nan_media.fillna(player_df_reemplazo_nan_media.mean(), inplace=True)
```

In [174]:

```
player_missing_values_count = player_df_reemplazo_nan_media.isnull().sum()

player_missing_values_count[player_missing_values_count > 0]
```

Out[174]:

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

Se comprueba exitosamente que no hay valores missing, una vez que se reemplazaron los datos por su media.

5. Normalizacion de columnas

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

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

Normalizando la columna crossing, usando Min-Max

Normalizamos la columna y mostramos un listado antes de la normalizacion y despues de la misma

In [175]:

```
# TODO
print(player_df.crossing.head(10))

scaler = preprocessing.MinMaxScaler()
player_df[["crossing"]] = scaler.fit_transform(player_df[["crossing"]])

print(player_df.crossing.head(10))
```

```
0    48.60
1    70.79
2    68.12
3    57.22
4    45.08
5    73.89
6    47.57
7    78.04
8    12.00
9    63.89
Name: crossing, dtype: float64
0    0.511036
```

```
1    0.777231
2    0.745202
3    0.614443
4    0.468810
5    0.814419
6    0.498680
7    0.864203
8    0.071977
9    0.694458
Name: crossing, dtype: float64
```

Normalizamos la columna short_passing usando Z-score

Normalizamos la columna y mostramos un listado antes de la normalizacion y despues de la misma.

In [176]:

```
print(player_df["short_passing"].head(10))

scaler = preprocessing.MinMaxScaler()
player_df[["short_passing"]] = sp.stats.zscore(player_df[["short_passing"]])

print(player_df["short_passing"].head(10))
```

```
0    60.60
1    62.27
2    65.12
3    64.70
4    64.76
5    78.26
6    63.57
7    76.27
8    23.00
9    68.95
Name: short_passing, dtype: float64
0    0.017238
1    0.140868
2    0.351853
3    0.320761
4    0.325202
5    1.324605
6    0.237107
7    1.177285
8   -2.766282
9    0.635387
Name: short_passing, dtype: float64
```

6. Codificar variables

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

Codificar la variable `country_name` del Dataframe `match_df`

La columna "country_name" antes de ser etiquetada como variable numerica

In [177]:

```
print(set(match_df["country_name"]))
```

```
{'Portugal', 'Belgium', 'Netherlands', 'France', 'England', 'Scotland', 'Italy', 'Poland',
'Switzerland', 'Spain', 'Germany'}
```

Etiqueto como variables numericas a la columna "country_name"

In [178]:

```
le = preprocessing.LabelEncoder()
match_df[["country_name"]] = le.fit_transform(match_df[["country_name"]])
```

Visualizo la columna, con los datos ya transformados

In [179]:

```
print(set(match_df["country_name"]))
```

```
{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
```

A la columna "country_name", la etiquedamos con variables numericas

In [ ]:

```

```