

Fifa practice

L'objectiu d'aquesta pràctica es predir el valor dels jugadors emprant informació dels jugadors

Autor: Carles Serrano Rueda

Any: 2020-2021

Link Github: <https://github.com/ca7les/PracticaMLUIB>

Importam llibraries

Primer de tot importam totes les llibreries que empremem

In [1]:

```
import os

from sklearn.model_selection import train_test_split
from sklearn import linear_model
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score

import pandas as pd
import numpy as np
import warnings
warnings.filterwarnings('ignore')
import matplotlib.pyplot as plt
```

Primer carregam les dades

Empremem la llibreria `pandas` i una funció de la llibreria `os`

In [2]:

```
df = pd.read_csv(os.path.join("../", "in", "/content/fifa.csv"))
df
```

Out[2]:

Unnamed: 0	ID	Name	Age	Photo	Nationality	Flag	Overall	P
0	0	158023	L. Messi	31	https://cdn.sofifa.org/players/4/19/158023.png	Argentina	https://cdn.sofifa.org/flags/52.png	94
1	1	20801	Cristiano Ronaldo	33	https://cdn.sofifa.org/players/4/19/20801.png	Portugal	https://cdn.sofifa.org/flags/38.png	94
2	2	190871	Neymar Jr	26	https://cdn.sofifa.org/players/4/19/190871.png	Brazil	https://cdn.sofifa.org/flags/54.png	92
3	3	193080	De Gea	27	https://cdn.sofifa.org/players/4/19/193080.png	Spain	https://cdn.sofifa.org/flags/45.png	91
4	4	192985	K. De Bruyne	27	https://cdn.sofifa.org/players/4/19/192985.png	Belgium	https://cdn.sofifa.org/flags/7.png	91
...
18202	18202	238813	J. Lundstram	19	https://cdn.sofifa.org/players/4/19/238813.png	England	https://cdn.sofifa.org/flags/14.png	47
18203	18203	243165	N. Christoffersson	19	https://cdn.sofifa.org/players/4/19/243165.png	Sweden	https://cdn.sofifa.org/flags/46.png	47
18204	18204	241638	B. Worman	16	https://cdn.sofifa.org/players/4/19/241638.png	England	https://cdn.sofifa.org/flags/14.png	47
18205	18205	246268	D. Walker-Rice	17	https://cdn.sofifa.org/players/4/19/246268.png	England	https://cdn.sofifa.org/flags/14.png	47
18206	18206	246269	G. Nugent	16	https://cdn.sofifa.org/players/4/19/246269.png	England	https://cdn.sofifa.org/flags/14.png	46

18207 rows × 89 columns



Anàlisi de les dades

En primer lloc, volem saber què està ocorrent amb les dades. Per fer-ho incrementem el nombre de columnes a mostrar. Després d'això utilitzarem una instrucció que ens mostrarà l'estructura de dades.

In [3]:

```
pd.set_option('display.max_columns', None)
```

In [4]:

```
df.head()
```

Out[4]:

Unnamed: 0	ID	Name	Age	Photo	Nationality	Flag	Overall	Potential	
0	0	158023	L. Messi	31	https://cdn.sofifa.org/players/4/19/158023.png	Argentina	https://cdn.sofifa.org/flags/52.png	94	94
1	1	20801	Cristiano Ronaldo	33	https://cdn.sofifa.org/players/4/19/20801.png	Portugal	https://cdn.sofifa.org/flags/38.png	94	94
2	2	190871	Neymar Jr	26	https://cdn.sofifa.org/players/4/19/190871.png	Brazil	https://cdn.sofifa.org/flags/54.png	92	93
3	3	193080	De Gea	27	https://cdn.sofifa.org/players/4/19/193080.png	Spain	https://cdn.sofifa.org/flags/45.png	91	93
4	4	192985	K. De Bruyne	27	https://cdn.sofifa.org/players/4/19/192985.png	Belgium	https://cdn.sofifa.org/flags/7.png	91	92



In [5]:

```
df.describe()
```

										Out[5]:
	Unnamed: 0	ID	Age	Overall	Potential	Special	International Reputation	Weak Foot	Skill Moves	
count	18207.000000	18207.000000	18207.000000	18207.000000	18207.000000	18207.000000	18159.000000	18159.000000	18159.000000	1
mean	9103.000000	214298.338606	25.122206	66.238699	71.307299	1597.809908	1.113222	2.947299	2.361308	
std	5256.052511	29965.244204	4.669943	6.908930	6.136496	272.586016	0.394031	0.660456	0.756164	
min	0.000000	16.000000	16.000000	46.000000	48.000000	731.000000	1.000000	1.000000	1.000000	
25%	4551.500000	200315.500000	21.000000	62.000000	67.000000	1457.000000	1.000000	3.000000	2.000000	
50%	9103.000000	221759.000000	25.000000	66.000000	71.000000	1635.000000	1.000000	3.000000	2.000000	
75%	13654.500000	236529.500000	28.000000	71.000000	75.000000	1787.000000	1.000000	3.000000	3.000000	
max	18206.000000	246620.000000	45.000000	94.000000	95.000000	2346.000000	5.000000	5.000000	5.000000	

Com veim hi ha columnes que les hem d'eliminar ja que no aporten informació útil per poder entrenar el model.

Columnes:

- **Unnamed: 0** -> Es un index que no significa res sobre el jugador que es troba en aquella fila
- **ID** -> Es un identificador que nomes serveix per identificarlo únicament dedins tot el conjunt, per tant no ens interessa.
- **Name** -> Pel mateix motiu que la columna ID no ens interessa saber el nom de un jugador només les seves característiques
- **Photo, Flag, Club Logo** -> No ens interessen les imatges
- **Special** -> Es la suma de totes les columnes que indiquen la qualitat que te un jugador amb una característica, per tant com emprarem aquelles característiques per separat no es necessari tenir la suma, ja que es redundant.
- **Work rate** -> Es una dada molt subjectiva ja que alt o baix no se pot quantificar ni saber quant pot afectar, també depen de es dia que tenguí el jugador, per tant la eliminam. Dos jugadors poden tenir alt a la columna de rendiment alt en atac i alomillor un ho fa molt millor que l'altre i segons aquesta variable son iguals en aquesta característica.
- **Real Face** -> Si el jugador te o no la seva cara real al videojoc no ens interessa
- **Joined** -> El dia que va entrar a competir tampoc es una dada que influeixi en el valor del jugador ja que hi ha jugadors que poden dur 1 any jugant i tenen més valor que un jugador que duu 5 anys. I en el cas a la inversa també passa, hi ha jugadors que duen 3 anys que son joves i un jugador que duu 10 anys te major valor. Per tant aquesta característica no aporta informació per predir el valor del jugador.
- **Contract Valid Until** -> L'any que s'acabarà el contracte amb el club no ens interessa ja que no te res a veure amb valor del jugador. Els jugadors que tenen contracte amb un club solen tenir una clàusula si algun club els vol fixar abans de que acabi el contracte. Pero aquest contracte depen de molt de factors i cada contracte es diferent. Que un contracte duri 5 anys o 1 any no te res a veure amb el valor del jugador. Només ens interessaria quan el jugador ha acabat amb el contracte pero pot passar que el valor del jugador augmenti o disminueixi, així que per facilitar les coses, llevam aquesta columna.
- **LS, ST, RS, LW, LF, CF, RF, RW, LAM, CAM, RAM, LM, LCM, CM, RCM, RM, LWB, LDM, CDM, RDM, RWB, LB, LCB, CB, RCB, RB** -> Son els valors que te un jugador a cada posició del camp, no es una característica que influeixi molt al valor d'un jugador porque si fixes a un extrem dret te serà igual quina puntuació te de porter o de defensa central. (Farem una comprovació de si això es així)

In [6]:

```
df = df.drop(columns = ["Unnamed: 0", "ID", "Name", "Photo", "Flag", "Club Logo", "Special", "Work Rate", "Real F
```

In [7]:

```
df.head()
```

																Out[7]:
	Age	Nationality	Overall	Potential	Club	Value	Wage	Preferred Foot	International Reputation	Weak Foot	Skill Moves	Body Type	Position	Jersey Number	Loaned From	
0	31	Argentina	94	94	FC Barcelona	€110.5M	€565K	Left	5.0	4.0	4.0	Messi	RF	10.0	NaN	
1	33	Portugal	94	94	Juventus	€77M	€405K	Right	5.0	4.0	5.0	C. Ronaldo	ST	7.0	NaN	
2	26	Brazil	92	93	Paris Saint-Germain	€118.5M	€290K	Right	5.0	5.0	5.0	Neymar	LW	10.0	NaN	
3	27	Spain	91	93	Manchester United	€72M	€260K	Right	4.0	3.0	1.0	Lean	GK	1.0	NaN	
4	27	Belgium	91	92	Manchester City	€102M	€355K	Right	4.0	5.0	4.0	Normal	RCM	7.0	NaN	

Una vegada ja tenim les columnes amb les que farem feina, les hem de tractar i estudiar. Podem començar mirant les columnes que tenen NaN.

Tractament de NaNs

In [8]:

```
df.columns[df.isna().any()].tolist()
```

Out[8]:

```
['Club',
 'Preferred Foot',
 'International Reputation',
 'Weak Foot',
 'Skill Moves',
 'Body Type',
 'Position',
 'Jersey Number',
 'Loaned From',
 'Height',
 'Weight',
 'LS',
 'ST',
 'RS',
 'LW',
 'LF',
 'CF',
 'RF',
 'RW',
 'LAM',
 'CAM',
 'RAM',
 'LM',
 'LCM',
 'CM',
 'RCM',
 'RM',
 'LWB',
 'LDM',
 'CDM',
 'RDM',
 'RWB',
 'LB',
 'LCB',
 'CB',
 'RCB',
 'RB',
 'Crossing',
 'Finishing',
 'HeadingAccuracy',
 'ShortPassing',
 'Volleys',
 'Dribbling',
 'Curve',
 'FKAccuracy',
 'LongPassing',
 'BallControl',
 'Acceleration',
 'SprintSpeed',
 'Agility',
 'Reactions',
 'Balance',
 'ShotPower',
 'Jumping',
 'Stamina',
 'Strength',
 'LongShots',
 'Aggression',
 'Interceptions',
 'Positioning',
 'Vision',
 'Penalties',
 'Composure',
 'Marking',
 'StandingTackle',
 'SlidingTackle',
 'GKDivining',
 'GKHandling',
 'GK Kicking',
 'GK Positioning',
 'GK Reflexes',
 'Release Clause']
```

Podem veure que hi ha moltes columnes que tenen NaN

```
df[df.isna().any(axis=1)]
```

In [9]:

```
df[df.Isna().any(axis=1)]
```

Out[9]:

	Age	Nationality	Overall	Potential	Club	Value	Wage	Preferred Foot	International Reputation	Weak Foot	Skill Moves	Body Type	Position	Jersey Number	Loaned From
0	31	Argentina	94	94	FC Barcelona	€110.5M	€565K	Left	5.0	4.0	4.0	Messi	RF	10.0	None
1	33	Portugal	94	94	Juventus	€77M	€405K	Right	5.0	4.0	5.0	C. Ronaldo	ST	7.0	None
2	26	Brazil	92	93	Paris Saint-Germain	€118.5M	€290K	Right	5.0	5.0	5.0	Neymar	LW	10.0	None
3	27	Spain	91	93	Manchester United	€72M	€260K	Right	4.0	3.0	1.0	Lean	GK	1.0	None
4	27	Belgium	91	92	Manchester City	€102M	€355K	Right	4.0	5.0	4.0	Normal	RCM	7.0	None
...
18202	19	England	47	65	Crewe Alexandra	€60K	€1K	Right	1.0	2.0	2.0	Lean	CM	22.0	None
18203	19	Sweden	47	63	Trelleborgs FF	€60K	€1K	Right	1.0	2.0	2.0	Normal	ST	21.0	None
18204	16	England	47	67	Cambridge United	€60K	€1K	Right	1.0	3.0	2.0	Normal	ST	33.0	None
18205	17	England	47	66	Tranmere Rovers	€60K	€1K	Right	1.0	3.0	2.0	Lean	RW	34.0	None
18206	16	England	46	66	Tranmere Rovers	€60K	€1K	Right	1.0	3.0	2.0	Lean	CM	33.0	None

18207 rows × 16 columns



Tractament Loaned From i Club

Primer de tot tractarem la columna Loaned From ja que podem veure que te molts de valors amb NaN. Aquesta columna està molt relacionada amb Club. Ja que un jugador pot pertànyer a un club pero jugar per un altre club perquè està cedit. Que un jugador estigui cedit influeix en el valor perquè no es el mateix que jugui pel club principal que que jugui cedit a un altre club. Pero tampoc podem posar que sigui jugador del club on està cedit perquè un jugador per exemple que sigui del Barça i estigui cedit al Mallorca, té més valor que si fos un jugador del Mallorca simplement. Per tant el que farem serà canviar el club actual ja que la columna Club té el club on està jugant i la columna Loaned from té el club de on ve cedit. Per tant el que farem serà posar el club de on ve cedit com a club i el valor de Loaned From serà 1 si està cedit. D'aquesta manera podrem saber que pertany a un club pero que està cedit, per tant el seu valor no serà tan gran com si realment formes part de la plantilla del club on te contracte.

Primer mirarem quants de NaN te la columna Club


In [10]:

```
df.loc[df.Club != df.Club]
```

Out[10]:

	Age	Nationality	Overall	Potential	Club	Value	Wage	Preferred Foot	International Reputation	Weak Foot	Skill Moves	Body Type	Position	Jersey Number	Loaned From	Height
452	24	Argentina	80	85	NaN	€0	€0	Right	2.0	4.0	4.0	Normal	CM	5.0	NaN	5'11"
538	33	Sweden	80	80	NaN	€0	€0	Right	2.0	4.0	2.0	Normal	LCB	4.0	NaN	6'0"
568	26	Russia	79	81	NaN	€0	€0	Right	1.0	3.0	1.0	Normal	GK	12.0	NaN	6'0"
677	29	Russia	79	79	NaN	€0	€0	Right	2.0	3.0	3.0	Lean	RB	2.0	NaN	5'10"
874	29	Russia	78	78	NaN	€0	€0	Right	2.0	3.0	3.0	Stocky	ST	22.0	NaN	6'0"
...
17197	21	India	55	64	NaN	€0	€0	Right	1.0	2.0	1.0	Normal	GK	1.0	NaN	6'0"
17215	26	Finland	55	57	NaN	€0	€0	Right	1.0	3.0	2.0	Normal	RB	3.0	NaN	6'0"
17339	23	India	54	63	NaN	€0	€0	Right	1.0	3.0	2.0	Normal	NaN	NaN	NaN	5'0"
17436	20	India	54	67	NaN	€0	€0	Right	1.0	3.0	2.0	Normal	NaN	NaN	NaN	6'0"
17539	21	India	53	62	NaN	€0	€0	Right	1.0	3.0	2.0	Lean	NaN	NaN	NaN	6'0"

241 rows × 78 columns



Son un total de 241 jugadors que no tenen club per tant com veim de 18207 jugadors que te el dataframe no suposa una gran p rduda si els llevam

In [11]:

```
df = df.dropna(subset = ['Club'])
df
```

Out[11]:

	Age	Nationality	Overall	Potential	Club	Value	Wage	Preferred Foot	International Reputation	Weak Foot	Skill Moves	Body Type	Position	Jersey Number	Loaned From
0	31	Argentina	94	94	FC Barcelona	�110.5M	�565K	Left	5.0	4.0	4.0	Messi	RF	10.0	NaN
1	33	Portugal	94	94	Juventus	�77M	�405K	Right	5.0	4.0	5.0	C. Ronaldo	ST	7.0	NaN
2	26	Brazil	92	93	Paris Saint-Germain	�118.5M	�290K	Right	5.0	5.0	5.0	Neymar	LW	10.0	NaN
3	27	Spain	91	93	Manchester United	�72M	�260K	Right	4.0	3.0	1.0	Lean	GK	1.0	NaN
4	27	Belgium	91	92	Manchester City	�102M	�355K	Right	4.0	5.0	4.0	Normal	RCM	7.0	NaN
...
18202	19	England	47	65	Crewe Alexandra	�60K	�1K	Right	1.0	2.0	2.0	Lean	CM	22.0	NaN
18203	19	Sweden	47	63	Trelleborgs FF	�60K	�1K	Right	1.0	2.0	2.0	Normal	ST	21.0	NaN
18204	16	England	47	67	Cambridge United	�60K	�1K	Right	1.0	3.0	2.0	Normal	ST	33.0	NaN
18205	17	England	47	66	Tranmere Rovers	�60K	�1K	Right	1.0	3.0	2.0	Lean	RW	34.0	NaN
18206	16	England	46	66	Tranmere Rovers	�60K	�1K	Right	1.0	3.0	2.0	Lean	CM	33.0	NaN

17966 rows × 78 columns



Miram quants de jugadors tenim que estiguin cedits

In [12]:

```
df[df.isna()["Loaned From"]==False]
```

Out[12]:

	Age	Nationality	Overall	Potential	Club	Value	Wage	Preferred Foot	International Reputation	Weak Foot	Skill Moves	Body Type	Position	Jersey Number	Loaned From
28	26	Colombia	88	89	FC Bayern München	€69.5M	€315K	Left	4.0	3.0	4.0	Normal	LAM	10.0	Real Madrid
38	30	Argentina	88	88	Milan	€57M	€245K	Right	4.0	4.0	3.0	Normal	LS	9.0	Juventus
91	29	Brazil	85	85	Guangzhou Evergrande Taobao FC	€37M	€235K	Right	3.0	3.0	3.0	Lean	LDM	9.0	Barcelona
166	24	Brazil	83	90	Guangzhou Evergrande Taobao FC	€36.5M	€18K	Left	2.0	4.0	4.0	Normal	CAM	24.0	Benfica
176	24	Croatia	83	89	Chelsea	€35M	€165K	Right	3.0	4.0	4.0	Normal	LCM	17.0	Real Madrid
...
17978	21	England	51	57	Hamilton Academical FC	€50K	€3K	Right	1.0	2.0	2.0	Lean	ST	16.0	Norwich City
17979	21	China PR	51	60	Guizhou Hengfeng FC	€60K	€2K	Right	1.0	2.0	2.0	Normal	CM	8.0	Tianjin Quanjian
18026	21	China PR	50	59	Guizhou Hengfeng FC	€50K	€2K	Right	1.0	2.0	2.0	Lean	LM	29.0	Jiangsu Suning
18031	20	China PR	50	61	Stabæk Fotball	€40K	€2K	Right	1.0	3.0	2.0	Normal	RB	98.0	Beijing Sinobo Guoan
18056	19	Italy	50	65	Ascoli	€60K	€3K	Left	1.0	3.0	2.0	Lean	CM	27.0	Real Madrid

1264 rows × 16 columns



Podem veure per exemple el jugador 28

```
df.iloc[28] ["Club"]
```

In [13]:

```
'FC Bayern München'
```

Out[13]:

```
df.iloc[28] ["Loaned From"]
```

In [14]:

```
'Real Madrid'
```

Out[14]:

En aquest cas hauriem de canviar el valor de FC Bayern München per Real Madrid i la columna Loaned From posar un 1
Pero abans de fer això posarem tots els NaN a 0

In [15]:

```
df["Loaned From"].fillna(0,inplace = True)
```

Ara canviem el clubs

In [16]:

```
df['Club'] = np.where(df['Loaned From'] != 0, df['Loaned From'], df["Club"])
```

Com podem veure s'han canviat els clubs

In [17]:

```
df.iloc[28] ["Club"]
```

Out[17]:

```
'Real Madrid'
```

In [18]:

```
df.iloc[28] ["Loaned From"]
```



```
'Real Madrid'
```

Out[18]:

Ara hem de posar a 1 totes les files que tinguin un club a la columna de Loaned From.

In [19]:

```
df["Loaned From"] = df["Loaned From"].apply(lambda x: 1 if x!=0 else 0)
```

I canviem el nom de Loaned From a Loaned ja que ara només te sentit si es Loaned o no.

In [20]:

```
df = df.rename(columns={'Loaned From': 'Loaned'})
```

Per comprovar que tot s'hagui fet correctament comprovarem amb el jugador que hem estat mirant

In [21]:

```
df.iloc[28]["Club"]
```

Out[21]:

```
'Real Madrid'
```

In [22]:

```
df.iloc[28]["Loaned"]
```

Out[22]:

```
1
```

Com podem veure ara el jugador te com a club Real Madrid i la columna de ceditos(Loaned) té un 1, que com hem dit abans era el resultat que esperavem

Pero encara no hem acabat amb la columna Club ja que no podem tractar strings, per tant haurem de crear una columna per cada Club de la següent manera:

In [23]:

```
clb = df.pop("Club")
df = pd.concat([df.reset_index(drop=True), pd.get_dummies(clb, prefix='clb').reset_index(drop=True)], axis=1)
```

In [24]:

```
df.head()
```

Out[24]:

	Age	Nationality	Overall	Potential	Value	Wage	Preferred Foot	International Reputation	Weak Foot	Skill Moves	Body Type	Position	Jersey Number	Loaned	Height	Weight
0	31	Argentina	94	94	€110.5M	€565K	Left	5.0	4.0	4.0	Messi	RF	10.0	0	5'7	156
1	33	Portugal	94	94	€77M	€405K	Right	5.0	4.0	5.0	C. Ronaldo	ST	7.0	0	6'2	183
2	26	Brazil	92	93	€118.5M	€290K	Right	5.0	5.0	5.0	Neymar	LW	10.0	0	5'9	150
3	27	Spain	91	93	€72M	€260K	Right	4.0	3.0	1.0	Lean	GK	1.0	0	6'4	168
4	27	Belgium	91	92	€102M	€355K	Right	4.0	5.0	4.0	Normal	RCM	7.0	0	5'11	154



Tractament Nationality

Com veim la Nacionalitat té el mateix problema que tenia Club, són strings i l'hem de fer que siguin diferents columnes. La nacionalitat d'un país si que influeix en el valor d'un jugador perquè per exemple els jugadors de Brasil són més valuosos que un jugador de Canadà

In [25]:

```
nat = df.pop("Nationality")
df = pd.concat([df.reset_index(drop=True), pd.get_dummies(nat, prefix='nat').reset_index(drop=True)], axis=1)
```

In [26]:

```
df.head()
```

Out[26]:

	Age	Overall	Potential	Value	Wage	Preferred Foot	International Reputation	Weak Foot	Skill Moves	Body Type	Position	Jersey Number	Loaned	Height	Weight	LS
0	31	94	94	€110.5M	€565K	Left	5.0	4.0	4.0	Messi	RF	10.0	0	5'7	159lbs	88+2
1	33	94	94	€77M	€405K	Right	5.0	4.0	5.0	C. Ronaldo	ST	7.0	0	6'2	183lbs	91+3
2	26	92	93	€118.5M	€290K	Right	5.0	5.0	5.0	Neymar	LW	10.0	0	5'9	150lbs	84+3
3	27	91	93	€72M	€260K	Right	4.0	3.0	1.0	Lean	GK	1.0	0	6'4	168lbs	NaN
4	27	91	92	€102M	€355K	Right	4.0	5.0	4.0	Normal	RCM	7.0	0	5'11	154lbs	82+3

Tractament Value, Wage i Release Clause

Amb aquestes columnes tenim un problema de format. Hem d'eliminar el símbol de € i les lletres K i M tractar-les com mils i milions.

Abans de començar pero estudiarem els casos de NaN a les respectives columnes

In [27]:

```
df.isna() ["Value"].sum()
```

0
Podem veure que la columna Value no té NaN

Out[27]:

In [28]:

```
df.isna() ["Wage"].sum()
```

0
Podem veure que la columna Wage no té NaN

Out[28]:

In [29]:

```
df.isna()["Release Clause"].sum()
```

1323

Amb la columna Release Clause veim que tenim 1323 valors que son NaN, per tant tractarem aquests jugadors com si no tenguessin clàusula, per tant canviarem els NaN per 0. Els podem tractar com si no tenguessin clàusula ja que pot passar que un jugador no tenguí clàusula, perquè vol dir que ha acabat el contracte. El problema es que el model podria pensar que si és 0 vol dir que no té valor o que es dolent i això no es vera perquè per exemple en Messi pot no tenir clàusula i no es dolent. Pero si li possèsim la mitjana als jugadors que no tenen clàusula podria no ser adient al jugador ja que a ne'n Messi podria tenir una clàusula de 100k€ quan realment és de milions.

Out[29]:

In [30]:

```
df["Release Clause"].fillna("€0",inplace = True)
```

Comprovam que no hagi hagut errors:

In [31]:

```
df.isna()["Release Clause"].sum()
```

Una vegada hem tractat els NaN, definirem una funció que s'encarregui de modificar el format

Out[31]:

In [32]:

```
def value_to_float(x):
    x = x.replace('€', '')
    ret_val = 0.0

    if type(x) == float or type(x) == int:
        ret_val = x
    if 'K' in x:
        if len(x) > 1:
            ret_val = float(x.replace('K', ''))
            ret_val = ret_val * 1000
    if 'M' in x:
        if len(x) > 1:
            ret_val = float(x.replace('M', ''))
```

```
ret_val = ret_val * 1000000.0
return ret_val
```

I ara aplicam aquesta funció a cada valor de cada columna

In [33]:

```
df["Value"] = df["Value"].apply(value_to_float)
df["Wage"] = df["Wage"].apply(value_to_float)
df["Release Clause"] = df["Release Clause"].apply(value_to_float)
```

Ara tenim aquestes tres columnes amb un format útil per el model

In [34]:

```
df.head()
```

Out[34]:

	Age	Overall	Potential	Value	Wage	Preferred Foot	International Reputation	Weak Foot	Skill Moves	Body Type	Position	Jersey Number	Loaned	Height	Weight	
0	31	94	94	110500000.0	565000.0	Left	5.0	4.0	4.0	Messi	RF	10.0	0	5'7	159lbs	8
1	33	94	94	77000000.0	405000.0	Right	5.0	4.0	5.0	C. Ronaldo	ST	7.0	0	6'2	183lbs	9
2	26	92	93	118500000.0	290000.0	Right	5.0	5.0	5.0	Neymar	LW	10.0	0	5'9	150lbs	8
3	27	91	93	72000000.0	260000.0	Right	4.0	3.0	1.0	Lean	GK	1.0	0	6'4	168lbs	
4	27	91	92	102000000.0	355000.0	Right	4.0	5.0	4.0	Normal	RCM	7.0	0	5'11	154lbs	8

Tractament Preferred Foot

Aquesta columna en un primer moment la podríem transformar en 0 o 1 (dreta o esquerra), pero anam a veure quins valors se poden obtenir

In [35]:

```
df["Preferred Foot"].unique()
```

Out[35]:

```
array(['Left', 'Right', nan], dtype=object)
```

Com veim només pot tenir com a valors esquerra o dreta, pero primer hem de tractar es NaN.

In [36]:

```
df.isna()["Preferred Foot"].sum()
```

Out[36]:

```
48
```

Com veim son 48 persones que no tenen valor a n'aquesta columna per tant el que farem serà eliminarlos ja que no es una gran quantitat del dataframe

In [37]:

```
df = df.dropna(subset = ['Preferred Foot'])
```

Comprovam que hagui anat bé

In [38]:

```
df.isna()["Preferred Foot"].sum()
```

Out[38]:

```
0
```

Ara el que farem serà definir una funció que si es dreta retornarà 0 i si es esquerra retornarà 1

In [39]:

```
def peuPreferit(x):
    if x == "Right":
        return 0
    return 1
```

Per acabar aplicam aquesta funció a Preferred Foot

In [40]:

```
df["Preferred Foot"] = df["Preferred Foot"].apply(peuPreferit)
```

In [41]:

```
df.head()
```

Out[41]:

	Age	Overall	Potential	Value	Wage	Preferred Foot	International Reputation	Weak Foot	Skill Moves	Body Type	Position	Jersey Number	Loaned	Height	Weight	
0	31	94	94	110500000.0	565000.0	1	5.0	4.0	4.0	Messi	RF	10.0	0	5'7	159lbs	8
1	33	94	94	77000000.0	405000.0	0	5.0	4.0	5.0	C. Ronaldo	ST	7.0	0	6'2	183lbs	9
2	26	92	93	118500000.0	290000.0	0	5.0	5.0	5.0	Neymar	LW	10.0	0	5'9	150lbs	8
3	27	91	93	72000000.0	260000.0	0	4.0	3.0	1.0	Lean	GK	1.0	0	6'4	168lbs	
4	27	91	92	102000000.0	355000.0	0	4.0	5.0	4.0	Normal	RCM	7.0	0	5'11	154lbs	8

Tractament Body Type

Body Type es una columna que representa el tipus de cos que te el jugador, pero anam a veure quins valors diferents hi ha

In [42]:

```
df["Body Type"].unique()
```

Out[42]:

```
array(['Messi', 'C. Ronaldo', 'Neymar', 'Lean', 'Normal', 'Courtois',  
      'Stocky', 'PLAYER_BODY_TYPE_25', 'Shaqiri', 'Akinfenwa'],  
      dtype=object)
```

Podem distingir 3 tipus de cos generals que són: Lean, Normal i Stocky i després diferents tipus de cos que fan referencia a jugadors individuals. Comprovarem si es tracta així

In [43]:

```
df["Body Type"].value_counts()
```

Out[43]:

```
Normal      10436  
Lean        6351  
Stocky      1124  
Shaqiri      1  
Neymar       1  
Courtois     1  
Messi        1  
PLAYER_BODY_TYPE_25  1  
Akinfenwa    1  
C. Ronaldo   1  
Name: Body Type, dtype: int64
```

Podem veure que existeixen 6 tipus de cos individuals, per exemple n'Akinfenwa té un cos molt gran ja que medeix 1.85 i pesa 102kg o en Shaqiri que medeix 1.65 i pesa 72kg. Però tot i ser coses especials els posarem a normal per simplificar el tractament ja que no influiran molt tenir-los per separat o juntar-los amb "Normal" que es el valor que més freqüència té.

In [44]:

```
def tipoDeCos(x):  
    if x == "Normal":  
        return "Normal"  
    if x == "Lean":  
        return "Lean"  
    if x == "Stocky":  
        return "Stocky"  
    return "Normal"
```

In [45]:

```
df["Body Type"] = df["Body Type"].apply(tipoDeCos)
```

Comprovam si s'ha fet correctament

In [46]:

```
df["Body Type"].unique()
```

array(['Normal', 'Lean', 'Stocky'], dtype=object)

Ara que ja només tenim aquests tres valors haurem de crear una columna per cada valor

Out[46]:

In [47]:

body = df.pop("Body Type")
df = pd.concat([df.reset_index(drop=True), pd.get_dummies(body, prefix='body').reset_index(drop=True)], a

In [48]:

df.head()

Out[48]:

	Age	Overall	Potential	Value	Wage	Preferred Foot	International Reputation	Weak Foot	Skill Moves	Position	Jersey Number	Loaned	Height	Weight	LS	ST	RS
0	31	94	94	110500000.0	565000.0	1	5.0	4.0	4.0	RF	10.0	0	5'7	159lbs	88+2	88+	88+
1	33	94	94	77000000.0	405000.0	0	5.0	4.0	5.0	ST	7.0	0	6'2	183lbs	91+3	91+	91+
2	26	92	93	118500000.0	290000.0	0	5.0	5.0	5.0	LW	10.0	0	5'9	150lbs	84+3	84+	84+
3	27	91	93	72000000.0	260000.0	0	4.0	3.0	1.0	GK	1.0	0	6'4	168lbs	NaN	NaN	NaN
4	27	91	92	102000000.0	355000.0	0	4.0	5.0	4.0	RCM	7.0	0	5'11	154lbs	82+3	82+	82+

Tractament Position

Aquesta columna té el mateix problema que el Club i la Nacionalitat. La posició on juga un jugador és molt important a l'hora de determinar el seu valor per tant farem una columna per cada posició

Pero abans comprovarem que no hi hagi NaN

In [49]:

df.isna()["Position"].sum()

0

Com veim no tenim NaN per tant ja podem crear les columnes

In [50]:

pos = df.pop("Position")
df = pd.concat([df.reset_index(drop=True), pd.get_dummies(pos, prefix='pos').reset_index(drop=True)], axis=1)

In [51]:

df.head()

Out[51]:

	Age	Overall	Potential	Value	Wage	Preferred Foot	International Reputation	Weak Foot	Skill Moves	Jersey Number	Loaned	Height	Weight	LS	ST	RS
0	31	94	94	110500000.0	565000.0	1	5.0	4.0	4.0	10.0	0	5'7	159lbs	88+2	88+2	88+2
1	33	94	94	77000000.0	405000.0	0	5.0	4.0	5.0	7.0	0	6'2	183lbs	91+3	91+3	91+3
2	26	92	93	118500000.0	290000.0	0	5.0	5.0	5.0	10.0	0	5'9	150lbs	84+3	84+3	84+3
3	27	91	93	72000000.0	260000.0	0	4.0	3.0	1.0	1.0	0	6'4	168lbs	NaN	NaN	NaN
4	27	91	92	102000000.0	355000.0	0	4.0	5.0	4.0	7.0	0	5'11	154lbs	82+3	82+3	82+3

Tractament Height

Height té un problema de format ja que té un apòstrof per representar els decimals, per tant hem de canviar-ho per un punt

Primer de tot mirarem si té NaN

In [52]:

df.isna()["Height"].sum()

Out[52]:

0
Veim que no tenim NaN, per tant ja podem canviar es format per tal de que sigui útil pel model

In [53]:

```
def convertHeight(x):  
    x.split("")  
    return float(x[0]+"."+x[2])
```

In [54]:

```
df["Height"] = df["Height"].apply(convertHeight)
```

In [55]:

```
df.head()
```

Out[55]:

	Age	Overall	Potential	Value	Wage	Preferred Foot	International Reputation	Weak Foot	Skill Moves	Jersey Number	Loaned	Height	Weight	LS	ST	RS
0	31	94	94	110500000.0	565000.0	1	5.0	4.0	4.0	10.0	0	5.7	159lbs	88+2	88+2	88+2
1	33	94	94	77000000.0	405000.0	0	5.0	4.0	5.0	7.0	0	6.2	183lbs	91+3	91+3	91+3
2	26	92	93	118500000.0	290000.0	0	5.0	5.0	5.0	10.0	0	5.9	150lbs	84+3	84+3	84+3
3	27	91	93	72000000.0	260000.0	0	4.0	3.0	1.0	1.0	0	6.4	168lbs	NaN	NaN	NaN
4	27	91	92	102000000.0	355000.0	0	4.0	5.0	4.0	7.0	0	5.1	154lbs	82+3	82+3	82+3

Tractament Weight

Weight igual que Height té un problema de format ja que no mos interessa les unitats en que estigui agafat un valor, sempre i quan tots els valors estiguin amb les mateixes unitats. Per tant el que hem de fer es eliminar "lbs".

Primer de tot mirarem els NaN

In [56]:

```
df.isna()["Weight"].sum()
```

Out[56]:

0
No tenim cap NaN, per tant ja li podem aplicar el format

In [57]:

```
def valueLibras(x):  
    x=x.rstrip("lbs")  
    return float(x)
```

In [58]:

```
df["Weight"] = df["Weight"].apply(valueLibras)
```

In [59]:

```
df.head()
```

Out[59]:

	Age	Overall	Potential	Value	Wage	Preferred Foot	International Reputation	Weak Foot	Skill Moves	Jersey Number	Loaned	Height	Weight	LS	ST	RS
0	31	94	94	110500000.0	565000.0	1	5.0	4.0	4.0	10.0	0	5.7	159.0	88+2	88+2	88+2
1	33	94	94	77000000.0	405000.0	0	5.0	4.0	5.0	7.0	0	6.2	183.0	91+3	91+3	91+3
2	26	92	93	118500000.0	290000.0	0	5.0	5.0	5.0	10.0	0	5.9	150.0	84+3	84+3	84+3
3	27	91	93	72000000.0	260000.0	0	4.0	3.0	1.0	1.0	0	6.4	168.0	NaN	NaN	NaN
4	27	91	92	102000000.0	355000.0	0	4.0	5.0	4.0	7.0	0	5.1	154.0	82+3	82+3	82+3

Preparació per la segon prova

In [60]:

```
df_ColumnesPos = df.iloc[:,13:39]
```

In [61]:

```
df_ColumnesPos
```

Out[61]:

	LS	ST	RS	LW	LF	CF	RF	RW	LAM	CAM	RAM	LM	LCM	CM	RCM	RM	LWB	LDM	CDM	RDM	R
0	88+2	88+2	88+2	92+2	93+2	93+2	93+2	92+2	93+2	93+2	93+2	91+2	84+2	84+2	84+2	91+2	64+2	61+2	61+2	61+2	6
1	91+3	91+3	91+3	89+3	90+3	90+3	90+3	89+3	88+3	88+3	88+3	88+3	81+3	81+3	81+3	88+3	65+3	61+3	61+3	61+3	6
2	84+3	84+3	84+3	89+3	89+3	89+3	89+3	89+3	89+3	89+3	89+3	88+3	81+3	81+3	81+3	88+3	65+3	60+3	60+3	60+3	6
3	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	82+3	82+3	82+3	87+3	87+3	87+3	87+3	87+3	88+3	88+3	88+3	88+3	87+3	87+3	87+3	88+3	77+3	77+3	77+3	77+3	7
...
17913	42+2	42+2	42+2	44+2	44+2	44+2	44+2	44+2	45+2	45+2	45+2	44+2	45+2	45+2	45+2	44+2	44+2	45+2	45+2	45+2	4
17914	45+2	45+2	45+2	39+2	42+2	42+2	42+2	39+2	40+2	40+2	40+2	38+2	35+2	35+2	35+2	38+2	30+2	31+2	31+2	31+2	3
17915	45+2	45+2	45+2	45+2	46+2	46+2	46+2	45+2	44+2	44+2	44+2	44+2	38+2	38+2	38+2	44+2	34+2	30+2	30+2	30+2	3
17916	47+2	47+2	47+2	47+2	46+2	46+2	46+2	47+2	45+2	45+2	45+2	46+2	39+2	39+2	39+2	46+2	36+2	32+2	32+2	32+2	3
17917	43+2	43+2	43+2	45+2	44+2	44+2	44+2	45+2	45+2	45+2	45+2	46+2	45+2	45+2	45+2	46+2	46+2	46+2	46+2	46+2	4

17918 rows × 26 columns



In [62]:

```
df.drop(columns = ["LS","ST","RS","LW","LF","CF","RF","RW","LAM","CAM","RAM","LM","LCM","CM","RCM","RM","R",
```

Abans de passar a fer la part de predicció, comprovarem que no ens hem deixat cap columna amb NaN

In [63]:

```
df.columns[df.isna().any()].tolist()
```

Out[63]:

```
[]
```

Ens guardarem una mostra del dataframe per poder fer una segona prova

In [64]:

```
df_2 = df.copy(deep=True)
```

Predicció

Una vegada ha hem fet tot el tractament de les dades, el que volem es predir la columna de **value**

In [65]:

```
val = df.pop("Value")
df
```

Out[65]:

	Age	Overall	Potential	Wage	Preferred Foot	International Reputation	Weak Foot	Skill Moves	Jersey Number	Loaned	Height	Weight	Crossing	Finishing	Heading
0	31	94	94	565000.0	1	5.0	4.0	4.0	10.0	0	5.7	159.0	84.0	95.0	
1	33	94	94	405000.0	0	5.0	4.0	5.0	7.0	0	6.2	183.0	84.0	94.0	
2	26	92	93	290000.0	0	5.0	5.0	5.0	10.0	0	5.9	150.0	79.0	87.0	
3	27	91	93	260000.0	0	4.0	3.0	1.0	1.0	0	6.4	168.0	17.0	13.0	
4	27	91	92	355000.0	0	4.0	5.0	4.0	7.0	0	5.1	154.0	93.0	82.0	
...
17913	19	47	65	1000.0	0	1.0	2.0	2.0	22.0	0	5.9	134.0	34.0	38.0	
17914	19	47	63	1000.0	0	1.0	2.0	2.0	21.0	0	6.3	170.0	23.0	52.0	
17915	16	47	67	1000.0	0	1.0	3.0	2.0	33.0	0	5.8	148.0	25.0	40.0	
17916	17	47	66	1000.0	0	1.0	3.0	2.0	34.0	0	5.1	154.0	44.0	50.0	
17917	16	46	66	1000.0	0	1.0	3.0	2.0	33.0	0	5.1	176.0	41.0	34.0	

17918 rows × 892 columns



Ara ja tenim separada les dades amb la columna que volem predir. El que hem de fer ara es separar les dades amb dades que emprarem per entrenar el model i dades que emprarem per evaluar. Agafarem un 67% de les dades per entrenar un model i un 33% per evaluar-lo

In [66]:

```
X_train, X_test, y_train, y_test = train_test_split(df, val, test_size=0.33, random_state=42)
```

Ara passam a fer la normalització. Agafarem totes aquelles columnes que el valors no siguin 1 o 0. De les columnes que normalitzarem agafarem aquelles columnes on la seva desviació estandard no sigui 0.

In [67]:

```
for i in X_test.columns:
    if i == "Loaned" or i == "Preferred Foot":
        pass
    elif i == "clb_ SSV Jahn Regensburg":
        break
    else:
        if (X_train[i].std() != 0):
            X_test[i] = (X_test[i]-X_train[i].mean())/X_train[i].std()
            X_train[i] = (X_train[i]-X_train[i].mean())/X_train[i].std()
        else:
            X_train.pop(i)
            X_test.pop(i)
```

In [68]:

```
y_test = (y_test-y_train.mean())/y_train.std()
```

In [69]:

```
y_train = (y_train-y_train.mean())/y_train.std()
```

Ara entrenam un model de regressió lineal amb les dades que hem separat abans per entrenar.

In [70]:

```
reg = linear_model.LinearRegression().fit(X_train, y_train)
```

Finalment agafam la mètrica de R2 per la regressió. Empram la implementació de la llibreria [sickit-learn](#). Empram R² ja que es tracta de una regressió linial i volem predir valors. R2 el que fa es emprar el coeficient de correlació de Pearson i l'eleva al quadrat. R2 pot agafar valors entre 0 i 1(tot i que hi ha casos que pot agafar valors negatius), de manera que quan més s'aproxima a 1 millor ha estat la generalització i més encertada serà la predicció

In [71]:

```
preds = reg.predict(X_test)
```

In [72]:

```
r2_score(preds, y_test)
```


-0.002201975243184995

Out[72]:

Ens surt un resultat molt dolent. Pot ser que sigui perquè no s'ha normalitzat bé. Miram amem si hi ha qualche errada

In [73]:

X_test

Out[73]:

	Age	Overall	Potential	Wage	Preferred Foot	International Reputation	Weak Foot	Skill Moves	Jersey Number	Loaned	Height	Weight	Crossing
12491	1.488645	-0.463036	-1.366741	-0.361748	1	-0.285658	0.078287	-0.488667	-1.038409	0	0.457922	-1.142576	-0.607804
7988	0.632663	0.112284	-0.714593	-0.361748	0	-0.285658	0.078287	-0.488667	3.052964	0	0.902818	0.841547	-0.593558
13940	-1.293295	-0.750696	-0.426667	-0.408015	1	-0.285658	-1.430282	-0.488667	0.883296	0	0.457922	0.521527	-0.975809
2616	2.558622	0.975265	0.263630	-0.130409	0	2.235906	0.078287	0.839184	-0.356514	0	0.235474	-1.270584	1.426915
3194	-0.223318	0.831435	1.241853	1.627764	0	-0.285658	0.078287	-0.488667	0.263391	1	0.680370	0.841547	-0.156699
...
10753	-0.009323	0.175376	-0.551556	-0.269212	0	-0.285658	0.078287	-0.488667	0.201401	0	0.680370	-0.438533	0.717019
15553	-0.437314	1.038356	-1.203704	-0.408015	0	-0.285658	-1.430282	-0.488667	0.914428	0	0.457922	1.545591	0.061731
1041	-0.009323	1.550585	1.404890	3.247134	0	2.235906	1.586857	0.839184	1.069268	0	0.680370	-0.054509	1.426915
5996	0.418668	0.399945	-0.388519	-0.222945	0	-0.285658	0.078287	-0.488667	0.077420	0	-1.544110	-0.886561	0.771626
13461	0.632663	-0.606866	-1.529779	-0.408015	0	-0.285658	0.078287	-0.488667	-0.356514	0	-0.209422	-1.718612	0.334768

5913 rows × 892 columns



Tot pareix que s'ha normalitzat bé, anam a veure y_test

In [74]:

y_test

Out[74]:

```
12491    -0.402950
7988     -0.295376
13940    -0.362610
2616     -0.205732
3194      0.224561
...
10753    -0.340199
15553    -0.406535
1041      1.533369
5996     -0.281930
13461    -0.395778
```

Name: Value, Length: 5913, dtype: float64

No sembla que sigui un problema de la normalització per tant provarem d'eliminar columnes ja que com hem vist al implementar one hot encoding, se'ns ha generat una gran quantitat de columnes, sobretot amb els clubs, per tant el que podem fer es eliminar tots els clubs que no tenen molta gent i que per tant no aporten molta informació.

Crearem un dataframe amb les columnes del club per tractarles

In [75]:

```
df.columns.get_loc("clb_ SSV Jahn Regensburg")
```

Out[75]:

47

In [76]:

```
df.columns.get_loc("clb_Śląsk Wrocław")
```

698

Out[76]:

```
df_columnesClub = df.iloc[:,47:699]
```

In [77]:

```
len(df_columnesClub.columns)
```

In [78]:

652

Out[78]:

Tenim 652 columnes que són clubs, com veim són moltes columnes que alomillor no aporten molta informació al model i encanvi ocupen columnes que el model haurà de mirar.

Mirarem de clubs grans quina es la quantitat de jugadors que tenen

In [79]:

```
df_columnesClub["clb_FC Barcelona"].value_counts()
```

Out[79]:

```
0    17877
1         41
Name: clb_FC Barcelona, dtype: int64
```

In [80]:

```
df_columnesClub["clb_Chelsea"].value_counts()
```

Out[80]:

```
0    17874
1         44
Name: clb_Chelsea, dtype: int64
```

Podem veure que més o menys en tenen 40. Anam a veure un club de segona divisió

In [81]:

```
df_columnesClub["clb_RCD Mallorca"].value_counts()
```

Out[81]:

```
0    17892
1         26
Name: clb_RCD Mallorca, dtype: int64
```

Com veim s'ha reduït a 26 jugadors per tant mirarem quants de clubs tindrem si agafam els que tinguin 26 jugadors

Una reducció bona sense perdre molta informació i sense que tenguem un excés de columnes hauria de variar entre 350-450 clubs

In [82]:

```
llistaClubs = []
for c in df_columnesClub.columns:
    if df_columnesClub[c].sum() < 26:
        llistaClubs.append(c)
```

Anam a veure quants de clubs hem llevat

In [83]:

```
len(llistaClubs)
```

Out[83]:

223

Llevam dels dataframes els clubs que s'han seleccionat

In [84]:

```
for c in llistaClubs:
    df.pop(c)
    df_2.pop(c) #ens servirà per fer la segona prova amb les columnes i les posicions
```

In [85]:

```
len(df_columnesClub.columns)-len(llistaClubs)
```

Out[85]:

429

Com veim de 652 clubs que teniem ens hem quedat amb 429. Hem reduït el dataset de manera que han quedat els més significatius.

Anam a fer el mateix amb les nacionalitats ja que també hi ha països on només hi ha un jugador i tenim una columna només per aquests

In [86]:

```
df.head(1)
```

Out[86]:

	Age	Overall	Potential	Wage	Preferred Foot	International Reputation	Weak Foot	Skill Moves	Jersey Number	Loaned	Height	Weight	Crossing	Finishing	HeadingAccur
0	31	94	94	565000.0	1	5.0	4.0	4.0	10.0	0	5.7	159.0	84.0	95.0	7



In [87]:

```
df.columns.get_loc("nat_Afghanistan")
```

Out[87]:

476

In [88]:

```
df.columns.get_loc("nat_Zimbabwe")
```

Out[88]:

638

Com amb els clubs cream un dataframe amb les nacionalitats per tractarles

In [89]:

```
df_columnesNat = df.iloc[:,476:639]
```

In [90]:

```
df_columnesNat
```

Out[90]:

	nat_Afghanistan	nat_Albania	nat_Algeria	nat_Andorra	nat_Angola	nat_Antigua & Barbuda	nat_Argentina	nat_Armenia	nat_Australia	nat_Austria
0	0	0	0	0	0	0	1	0	0	0
1	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0
...
17913	0	0	0	0	0	0	0	0	0	0
17914	0	0	0	0	0	0	0	0	0	0
17915	0	0	0	0	0	0	0	0	0	0
17916	0	0	0	0	0	0	0	0	0	0
17917	0	0	0	0	0	0	0	0	0	0

17918 rows × 163 columns



Cercam les nacionalitats on només hi ha un jugador i les llevam dels dataframes

In [91]:

```
llistaNat = []
for c in df_columnesNat.columns:
    if df_columnesNat[c].sum() < 2:
        llistaNat.append(c)
        df.pop(c)
        df_2.pop(c) #ens servirà per fer la segona prova amb les columnes i les posicions
```

In [92]:

```
len(llistaNat)
```

Out[92]:

25

Veim que podem llevar unes altres 25 columnes, dels països on només hi ha una persona.

Miram com ha quedat el dataframe

In [93]:

df

Out[93]:

	Age	Overall	Potential	Wage	Preferred Foot	International Reputation	Weak Foot	Skill Moves	Jersey Number	Loaned	Height	Weight	Crossing	Finishing	HeadingA
0	31	94	94	565000.0	1	5.0	4.0	4.0	10.0	0	5.7	159.0	84.0	95.0	
1	33	94	94	405000.0	0	5.0	4.0	5.0	7.0	0	6.2	183.0	84.0	94.0	
2	26	92	93	290000.0	0	5.0	5.0	5.0	10.0	0	5.9	150.0	79.0	87.0	
3	27	91	93	260000.0	0	4.0	3.0	1.0	1.0	0	6.4	168.0	17.0	13.0	
4	27	91	92	355000.0	0	4.0	5.0	4.0	7.0	0	5.1	154.0	93.0	82.0	
...
17913	19	47	65	1000.0	0	1.0	2.0	2.0	22.0	0	5.9	134.0	34.0	38.0	
17914	19	47	63	1000.0	0	1.0	2.0	2.0	21.0	0	6.3	170.0	23.0	52.0	
17915	16	47	67	1000.0	0	1.0	3.0	2.0	33.0	0	5.8	148.0	25.0	40.0	
17916	17	47	66	1000.0	0	1.0	3.0	2.0	34.0	0	5.1	154.0	44.0	50.0	
17917	16	46	66	1000.0	0	1.0	3.0	2.0	33.0	0	5.1	176.0	41.0	34.0	

17918 rows × 644 columns



Predicció

Ara provarem si amb l'eliminació d'aquestes columnes, el resultat de la predicció del model millora

In [94]:

```
X_train, X_test, y_train, y_test = train_test_split(df, val, test_size=0.33, random_state=42)
```

In [95]:

```
for i in X_test.columns:
    if i == "Loaned" or i == "Preferred Foot":
        pass
    elif i == "clb_1. FC Heidenheim 1846":
        break
    else:
        if (X_train[i].std()!=0):
            X_test[i] = (X_test[i]-X_train[i].mean())/X_train[i].std()
            X_train[i] = (X_train[i]-X_train[i].mean())/X_train[i].std()
        else:
            X_train.pop(i)
            X_test.pop(i)
```

In [96]:

```
y_test = (y_test-y_train.mean())/y_train.std()
```

In [97]:

```
y_train = (y_train-y_train.mean())/y_train.std()
```

In [98]:

```
reg = linear_model.LinearRegression().fit(X_train, y_train)
```

In [99]:

```
preds = reg.predict(X_test)
```

In [100]:

```
resultatProva1 =r2_score(preds, y_test)
```

In [101]:

```
resultatProva1
```

0.9705253876991475

Out[101]:

Podem veure que després de reduir el número de columnes dins el dataframe ja ens surt un resultat bo, ja que s'aproxima a 1.

Segona prova

Anam a fer una segona prova on emprarem les columnes de la puntuació que té un jugador a cada posició

El que farem serà agafar les columnes que hem guardat anteriorment a df_ColumnesPos i fer un estudi de quin tractament necessita

In [102]:

```
df_ColumnesPos
```

Out[102]:

	LS	ST	RS	LW	LF	CF	RF	RW	LAM	CAM	RAM	LM	LCM	CM	RCM	RM	LWB	LDM	CDM	RDM	R
0	88+2	88+2	88+2	92+2	93+2	93+2	93+2	92+2	93+2	93+2	93+2	91+2	84+2	84+2	84+2	91+2	64+2	61+2	61+2	61+2	61+2
1	91+3	91+3	91+3	89+3	90+3	90+3	90+3	89+3	88+3	88+3	88+3	88+3	81+3	81+3	81+3	88+3	65+3	61+3	61+3	61+3	61+3
2	84+3	84+3	84+3	89+3	89+3	89+3	89+3	89+3	89+3	89+3	89+3	88+3	81+3	81+3	81+3	88+3	65+3	60+3	60+3	60+3	60+3
3	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	82+3	82+3	82+3	87+3	87+3	87+3	87+3	87+3	88+3	88+3	88+3	88+3	87+3	87+3	87+3	88+3	77+3	77+3	77+3	77+3	77+3
...
17913	42+2	42+2	42+2	44+2	44+2	44+2	44+2	44+2	45+2	45+2	45+2	44+2	45+2	45+2	45+2	44+2	44+2	45+2	45+2	45+2	45+2
17914	45+2	45+2	45+2	39+2	42+2	42+2	42+2	39+2	40+2	40+2	40+2	38+2	35+2	35+2	35+2	38+2	30+2	31+2	31+2	31+2	31+2
17915	45+2	45+2	45+2	45+2	46+2	46+2	46+2	45+2	44+2	44+2	44+2	44+2	38+2	38+2	38+2	44+2	34+2	30+2	30+2	30+2	30+2
17916	47+2	47+2	47+2	47+2	46+2	46+2	46+2	47+2	45+2	45+2	45+2	46+2	39+2	39+2	39+2	46+2	36+2	32+2	32+2	32+2	32+2
17917	43+2	43+2	43+2	45+2	44+2	44+2	44+2	45+2	45+2	45+2	45+2	46+2	45+2	45+2	45+2	46+2	46+2	46+2	46+2	46+2	46+2

17918 rows × 26 columns



Veim que tenim hi ha un problema amb el format ja que es un string. El que farem es agafar es els dos nombres i sumar-los

Com veim per tenim jugadors que tenen valors NaN, són els jugadors que ocupen la posició de porter. El que podem fer es calcular la mitjana de tots es jugadors a cada posició i posarli als porters.

Per poder calcular la mitjana primer hem de llevar tots els porters del dataframe. (Amb una posició basta per detectar els porters ja que tenen NaN a totes les posicions)

In [103]:

```
df_noPorters = df_ColumnesPos[df_ColumnesPos.isna()["LS"]==False]
```

In [104]:

```
df_noPorters
```

Out[104]:

	LS	ST	RS	LW	LF	CF	RF	RW	LAM	CAM	RAM	LM	LCM	CM	RCM	RM	LWB	LDM	CDM	RDM	R
0	88+2	88+2	88+2	92+2	93+2	93+2	93+2	92+2	93+2	93+2	93+2	91+2	84+2	84+2	84+2	91+2	64+2	61+2	61+2	61+2	6
1	91+3	91+3	91+3	89+3	90+3	90+3	90+3	89+3	88+3	88+3	88+3	88+3	81+3	81+3	81+3	88+3	65+3	61+3	61+3	61+3	6
2	84+3	84+3	84+3	89+3	89+3	89+3	89+3	89+3	89+3	89+3	89+3	88+3	81+3	81+3	81+3	88+3	65+3	60+3	60+3	60+3	6
4	82+3	82+3	82+3	87+3	87+3	87+3	87+3	87+3	88+3	88+3	88+3	88+3	87+3	87+3	87+3	88+3	77+3	77+3	77+3	77+3	7
5	83+3	83+3	83+3	89+3	88+3	88+3	88+3	89+3	89+3	89+3	89+3	89+3	82+3	82+3	82+3	89+3	66+3	63+3	63+3	63+3	6
...
17913	42+2	42+2	42+2	44+2	44+2	44+2	44+2	44+2	45+2	45+2	45+2	44+2	45+2	45+2	45+2	44+2	44+2	45+2	45+2	45+2	4
17914	45+2	45+2	45+2	39+2	42+2	42+2	42+2	39+2	40+2	40+2	40+2	38+2	35+2	35+2	35+2	38+2	30+2	31+2	31+2	31+2	3
17915	45+2	45+2	45+2	45+2	46+2	46+2	46+2	45+2	44+2	44+2	44+2	44+2	38+2	38+2	38+2	44+2	34+2	30+2	30+2	30+2	3
17916	47+2	47+2	47+2	47+2	46+2	46+2	46+2	47+2	45+2	45+2	45+2	46+2	39+2	39+2	39+2	46+2	36+2	32+2	32+2	32+2	3
17917	43+2	43+2	43+2	45+2	44+2	44+2	44+2	45+2	45+2	45+2	45+2	46+2	45+2	45+2	45+2	46+2	46+2	46+2	46+2	46+2	4

15926 rows × 26 columns



Ara cream una funció que llevi el signe de suma i sumarem els dos números

In [105]:

```
def valorPosicions(v):
    v=v.split('+')
    return float(v[0])+float(v[1])
```

In [106]:

```
for columna in df_noPorters.columns:
    df_noPorters[columna] = df_noPorters[columna].apply(valorPosicions)
```

In [107]:

df_noPorters

Out[107]:

	LS	ST	RS	LW	LF	CF	RF	RW	LAM	CAM	RAM	LM	LCM	CM	RCM	RM	LWB	LDM	CDM	RDM	RWB	LB	LCE
0	90.0	90.0	90.0	94.0	95.0	95.0	95.0	94.0	95.0	95.0	95.0	93.0	86.0	86.0	86.0	93.0	66.0	63.0	63.0	63.0	66.0	61.0	49.0
1	94.0	94.0	94.0	92.0	93.0	93.0	93.0	92.0	91.0	91.0	91.0	91.0	84.0	84.0	84.0	91.0	68.0	64.0	64.0	64.0	68.0	64.0	56.0
2	87.0	87.0	87.0	92.0	92.0	92.0	92.0	92.0	92.0	92.0	92.0	91.0	84.0	84.0	84.0	91.0	68.0	63.0	63.0	63.0	68.0	63.0	50.0
4	85.0	85.0	85.0	90.0	90.0	90.0	90.0	90.0	91.0	91.0	91.0	91.0	90.0	90.0	90.0	91.0	80.0	80.0	80.0	80.0	80.0	76.0	69.0
5	86.0	86.0	86.0	92.0	91.0	91.0	91.0	92.0	92.0	92.0	92.0	92.0	85.0	85.0	85.0	92.0	69.0	66.0	66.0	66.0	69.0	63.0	52.0
...
17913	44.0	44.0	44.0	46.0	46.0	46.0	46.0	46.0	47.0	47.0	47.0	46.0	47.0	47.0	47.0	46.0	46.0	47.0	47.0	47.0	46.0	47.0	47.0
17914	47.0	47.0	47.0	41.0	44.0	44.0	44.0	41.0	42.0	42.0	42.0	40.0	37.0	37.0	37.0	40.0	32.0	33.0	33.0	33.0	32.0	31.0	34.0
17915	47.0	47.0	47.0	47.0	48.0	48.0	48.0	47.0	46.0	46.0	46.0	46.0	40.0	40.0	40.0	46.0	36.0	32.0	32.0	32.0	36.0	35.0	30.0
17916	49.0	49.0	49.0	49.0	48.0	48.0	48.0	49.0	47.0	47.0	47.0	48.0	41.0	41.0	41.0	48.0	38.0	34.0	34.0	34.0	38.0	37.0	33.0
17917	45.0	45.0	45.0	47.0	46.0	46.0	46.0	47.0	47.0	47.0	47.0	48.0	47.0	47.0	47.0	48.0	48.0	48.0	48.0	48.0	48.0	48.0	49.0

15926 rows × 26 columns



Ara ja podem calcular la mitjana de cada columna i després li aplicarem als porters

In [108]:

```
df_ColumnesPos[df_ColumnesPos.isna() ["LS"]==True]
```

Out[108]:

	LS	ST	RS	LW	LF	CF	RF	RW	LAM	CAM	RAM	LM	LCM	CM	RCM	RM	LWB	LDM	CDM	RDM	RWB	LB	LCB
3	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
9	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
18	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
19	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
22	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
...
17889	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
17891	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
17894	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
17905	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
17909	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

1992 rows × 26 columns



In [109]:

```
for column in df_noPorters.columns:
    df_ColumnesPos[df_ColumnesPos.isna()["LS"]==True] = str(df_noPorters.mean()[column].mean())+"0"
```

Per poder aplicar la funció de valorPosicions he hagut d'emplenar els NaN amb "+0"

In [110]:

df_ColumnesPos

Out[110]:

	LS	ST	RS	LW	LF	CF
0	88+2	88+2	88+2	92+2	93+2	93+2
1	91+3	91+3	91+3	89+3	90+3	90+3
2	84+3	84+3	84+3	89+3	89+3	89+3
3	59.84264724350119+0	59.84264724350119+0	59.84264724350119+0	59.84264724350119+0	59.84264724350119+0	59.84264724350119+0
4	82+3	82+3	82+3	87+3	87+3	87+3
...
17913	42+2	42+2	42+2	44+2	44+2	44+2
17914	45+2	45+2	45+2	39+2	42+2	42+2
17915	45+2	45+2	45+2	45+2	46+2	46+2
17916	47+2	47+2	47+2	47+2	46+2	46+2
17917	43+2	43+2	43+2	45+2	44+2	44+2

17918 rows × 7 columns



In [111]:

```
for columna in df_ColumnesPos.columns:
    df_ColumnesPos[columna] = df_ColumnesPos[columna].apply(valorPosicions)
```

In [112]:

df_ColumnesPos

Out[112]:

	LS	ST	RS	LW	LF	CF	RF	RW	LAM	CAM	RAM	LM
0	90.000000	90.000000	90.000000	94.000000	95.000000	95.000000	95.000000	94.000000	95.000000	95.000000	95.000000	93.000000
1	94.000000	94.000000	94.000000	92.000000	93.000000	93.000000	93.000000	92.000000	91.000000	91.000000	91.000000	91.000000
2	87.000000	87.000000	87.000000	92.000000	92.000000	92.000000	92.000000	92.000000	92.000000	92.000000	92.000000	91.000000
3	59.842647	59.842647	59.842647	59.842647	59.842647	59.842647	59.842647	59.842647	59.842647	59.842647	59.842647	59.842647
4	85.000000	85.000000	85.000000	90.000000	90.000000	90.000000	90.000000	90.000000	91.000000	91.000000	91.000000	91.000000
...
17913	44.000000	44.000000	44.000000	46.000000	46.000000	46.000000	46.000000	46.000000	47.000000	47.000000	47.000000	46.000000
17914	47.000000	47.000000	47.000000	41.000000	44.000000	44.000000	44.000000	41.000000	42.000000	42.000000	42.000000	40.000000
17915	47.000000	47.000000	47.000000	47.000000	48.000000	48.000000	48.000000	47.000000	46.000000	46.000000	46.000000	46.000000
17916	49.000000	49.000000	49.000000	49.000000	48.000000	48.000000	48.000000	49.000000	47.000000	47.000000	47.000000	48.000000
17917	45.000000	45.000000	45.000000	47.000000	46.000000	46.000000	46.000000	47.000000	47.000000	47.000000	47.000000	48.000000

17918 rows × 26 columns

Juntar dataframes

Una vegada hem tractat les columnes que voliem incorporar ja ho podem juntar amb la resta

In [113]:

df_2

Out[113]:

	Age	Overall	Potential	Value	Wage	Preferred Foot	International Reputation	Weak Foot	Skill Moves	Jersey Number	Loaned	Height	Weight	Crossing	Finishing
0	31	94	94	110500000.0	565000.0	1	5.0	4.0	4.0	10.0	0	5.7	159.0	84.0	95.0
1	33	94	94	77000000.0	405000.0	0	5.0	4.0	5.0	7.0	0	6.2	183.0	84.0	95.0
2	26	92	93	118500000.0	290000.0	0	5.0	5.0	5.0	10.0	0	5.9	150.0	79.0	84.0
3	27	91	93	72000000.0	260000.0	0	4.0	3.0	1.0	1.0	0	6.4	168.0	17.0	84.0
4	27	91	92	102000000.0	355000.0	0	4.0	5.0	4.0	7.0	0	5.1	154.0	93.0	84.0
...
17913	19	47	65	60000.0	1000.0	0	1.0	2.0	2.0	22.0	0	5.9	134.0	34.0	84.0
17914	19	47	63	60000.0	1000.0	0	1.0	2.0	2.0	21.0	0	6.3	170.0	23.0	84.0
17915	16	47	67	60000.0	1000.0	0	1.0	3.0	2.0	33.0	0	5.8	148.0	25.0	84.0
17916	17	47	66	60000.0	1000.0	0	1.0	3.0	2.0	34.0	0	5.1	154.0	44.0	84.0
17917	16	46	66	60000.0	1000.0	0	1.0	3.0	2.0	33.0	0	5.1	176.0	41.0	84.0

17918 rows × 645 columns

In [114]:

df_ColumnesPos

Out[114]:

	LS	ST	RS	LW	LF	CF	RF	RW	LAM	CAM	RAM	LM
0	90.000000	90.000000	90.000000	94.000000	95.000000	95.000000	95.000000	94.000000	95.000000	95.000000	95.000000	93.000000
1	94.000000	94.000000	94.000000	92.000000	93.000000	93.000000	93.000000	92.000000	91.000000	91.000000	91.000000	91.000000
2	87.000000	87.000000	87.000000	92.000000	92.000000	92.000000	92.000000	92.000000	92.000000	92.000000	92.000000	91.000000
3	59.842647	59.842647	59.842647	59.842647	59.842647	59.842647	59.842647	59.842647	59.842647	59.842647	59.842647	59.842647
4	85.000000	85.000000	85.000000	90.000000	90.000000	90.000000	90.000000	90.000000	91.000000	91.000000	91.000000	91.000000
...
17913	44.000000	44.000000	44.000000	46.000000	46.000000	46.000000	46.000000	46.000000	47.000000	47.000000	47.000000	46.000000
17914	47.000000	47.000000	47.000000	41.000000	44.000000	44.000000	44.000000	41.000000	42.000000	42.000000	42.000000	40.000000
17915	47.000000	47.000000	47.000000	47.000000	48.000000	48.000000	48.000000	47.000000	46.000000	46.000000	46.000000	46.000000
17916	49.000000	49.000000	49.000000	49.000000	48.000000	48.000000	48.000000	49.000000	47.000000	47.000000	47.000000	48.000000
17917	45.000000	45.000000	45.000000	47.000000	46.000000	46.000000	46.000000	47.000000	47.000000	47.000000	47.000000	48.000000

17918 rows × 26 columns

In [115]:

```
df_2 = pd.concat([df_2, df_ColumnesPos], axis=1, sort=False)
```

In [116]:

df_2

Out[116]:

	Age	Overall	Potential	Value	Wage	Preferred Foot	International Reputation	Weak Foot	Skill Moves	Jersey Number	Loaned	Height	Weight	Crossing	Finis
0	31	94	94	110500000.0	565000.0	1	5.0	4.0	4.0	10.0	0	5.7	159.0	84.0	9
1	33	94	94	77000000.0	405000.0	0	5.0	4.0	5.0	7.0	0	6.2	183.0	84.0	9
2	26	92	93	118500000.0	290000.0	0	5.0	5.0	5.0	10.0	0	5.9	150.0	79.0	8
3	27	91	93	72000000.0	260000.0	0	4.0	3.0	1.0	1.0	0	6.4	168.0	17.0	1
4	27	91	92	102000000.0	355000.0	0	4.0	5.0	4.0	7.0	0	5.1	154.0	93.0	8
...
17913	19	47	65	60000.0	1000.0	0	1.0	2.0	2.0	22.0	0	5.9	134.0	34.0	1
17914	19	47	63	60000.0	1000.0	0	1.0	2.0	2.0	21.0	0	6.3	170.0	23.0	9
17915	16	47	67	60000.0	1000.0	0	1.0	3.0	2.0	33.0	0	5.8	148.0	25.0	4
17916	17	47	66	60000.0	1000.0	0	1.0	3.0	2.0	34.0	0	5.1	154.0	44.0	9
17917	16	46	66	60000.0	1000.0	0	1.0	3.0	2.0	33.0	0	5.1	176.0	41.0	1

17918 rows × 671 columns

Segona predicció

Ara passam a fer la predicció d'aquesta prova per veure amem si els resultats han millorat

In [117]:

```
val = df_2.pop("Value")
```

In [118]:

```
X_train, X_test, y_train, y_test = train_test_split(df_2, val, test_size=0.33, random_state=42)
```

In [119]:

```
for i in X_test.columns:
    if i == "Loaned" or i == "Preferred Foot":
        pass
```

```

elif i == "clb_1. FC Heidenheim 1846":
    break
else:
    if (X_train[i].std()!=0):
        X_test[i] = (X_test[i]-X_train[i].mean())/X_train[i].std()
        X_train[i] = (X_train[i]-X_train[i].mean())/X_train[i].std()
    else:
        X_train.pop(i)
        X_test.pop(i)

```

In [120]:

```

for i in df_ColumnesPos.columns:
    if (X_train[i].std()!=0):
        X_test[i] = (X_test[i]-X_train[i].mean())/X_train[i].std()
        X_train[i] = (X_train[i]-X_train[i].mean())/X_train[i].std()
    else:
        X_train.pop(i)
        X_test.pop(i)

```

In [121]:

X_test

Out[121]:

	Age	Overall	Potential	Wage	Preferred Foot	International Reputation	Weak Foot	Skill Moves	Jersey Number	Loaned	Height	Weight	Crossing
12491	1.488645	0.463036	1.366741	0.361748	1	-0.285658	0.078287	0.488667	1.038409	0	0.457922	1.142576	0.607804
7988	0.632663	0.112284	0.714593	0.361748	0	-0.285658	0.078287	0.488667	3.052964	0	0.902818	0.841547	0.593558
13940	1.293295	0.750696	0.426667	0.408015	1	-0.285658	1.430282	0.488667	0.883296	0	0.457922	0.521527	0.975809
2616	2.558622	0.975265	0.263630	0.130409	0	2.235906	0.078287	0.839184	0.356514	0	0.235474	1.270584	1.426915
3194	0.223318	0.831435	1.241853	1.627764	0	-0.285658	0.078287	0.488667	0.263391	1	0.680370	0.841547	0.156699
...
10753	0.009323	0.175376	0.551556	0.269212	0	-0.285658	0.078287	0.488667	0.201401	0	0.680370	0.438533	0.717019
15553	0.437314	1.038356	1.203704	0.408015	0	-0.285658	1.430282	0.488667	0.914428	0	0.457922	1.545591	0.061731
1041	0.009323	1.550585	1.404890	3.247134	0	2.235906	1.586857	0.839184	1.069268	0	0.680370	0.054509	1.426915
5996	0.418668	0.399945	0.388519	0.222945	0	-0.285658	0.078287	0.488667	0.077420	0	1.544110	0.886561	0.771626
13461	0.632663	0.606866	1.529779	0.408015	0	-0.285658	0.078287	0.488667	0.356514	0	0.209422	1.718612	0.334768

5913 rows × 670 columns



In [122]:

```
reg = linear_model.LinearRegression().fit(X_train, y_train)
```

In [123]:

```
preds = reg.predict(X_test)
```

In [124]:

```
resultatProva2=r2_score(preds, y_test)
```

In [125]:

```
resultatProva2
```

0.9705504757867881

Out[125]:

resultatProva1

In [126]:

0.9705253876991475

Out[126]:

resultatProva2-resultatProva1

In [127]:

2.5088087640567913e-05

Out[127]:

Com veim la millora no ha estat molt significativa per tant això indica que les columnes de les posicions no aporten molta informació a l'hora de determinar el valor d'un jugador

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js