Carlos Llompart López

Inteligencia Artificial

Práctica 3. FIFA practice

Miquel Miró Nicolau

https://github.com/cllompart/PRACTICA-FIFA

Fifa practice

We want to predict the *value* of the players using the information of the player.

Import libraries

First of all we import the libraries we're gonna use.

Read the data

To read the data we're gonna use `pandas` and an utility from the library `os`.

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_

import pandas as pd
import numpy as np
```

Creamos una base que contrendrá la constrrucción del dataset con los datos de entrenamiento relativos a la variable objetivo 'Valor'. De esta manera, comprobaremos como mejoran esos datos en nuestro modelo de predicción.

In [2]:
```
df = pd.read_csv(os.path.join("..", "in", "fifa.csv"))
base = df
base.head()
```

Out[2]:

	Unnamed: 0	ID	Name	Age	Photo	Nationality
0	0	158023	L. Messi	31	https://cdn.sofifa.org/players/4/19/158023.png	Argentina
1	1	20801	Cristiano Ronaldo	33	https://cdn.sofifa.org/players/4/19/20801.png	Portugal
2	2	190871	Neymar Jr	26	https://cdn.sofifa.org/players/4/19/190871.png	Brazil
3	3	193080	De Gea	27	https://cdn.sofifa.org/players/4/19/193080.png	Spain

4	4	192985	K. De Bruyne	27	https://cdn.sofifa.org/players/4/19/192985.png	Belgium

5 rows × 89 columns

Ya tenemos los jugadores cargados en el data frame

Now we select one columns. The result will be an object from the `pandas` library a `series` object. This object can be converted to a more standard `numpy` array.

In [3]:
```
name = base["Name"]
name
```

Out[3]:
```
0                  L. Messi
1          Cristiano Ronaldo
2                 Neymar Jr
3                    De Gea
4              K. De Bruyne
                  ...
18202           J. Lundstram
18203    N. Christoffersson
18204             B. Worman
18205         D. Walker-Rice
18206             G. Nugent
Name: Name, Length: 18207, dtype: object
```

Data analysis

First of all we want to known what is happening to the data. To do so we increase the number of columns to show. After that we're gonna use a special instruction that will show us the data structure.

In [4]:
```
pd.set_option('display.max_columns', None)
```

In [5]:
```
base.head()
```

Out[5]:

	Unnamed: 0	ID	Name	Age	Photo	Nationality
0	0	158023	L. Messi	31	https://cdn.sofifa.org/players/4/19/158023.png	Argentina
1	1	20801	Cristiano Ronaldo	33	https://cdn.sofifa.org/players/4/19/20801.png	Portugal
2	2	190871	Neymar Jr	26	https://cdn.sofifa.org/players/4/19/190871.png	Brazil
3	3	193080	De Gea	27	https://cdn.sofifa.org/players/4/19/193080.png	Spain
4	4	192985	K. De Bruyne	27	https://cdn.sofifa.org/players/4/19/192985.png	Belgium

In [6]:
```
base.describe()
```

Out[6]:

	Unnamed: 0	ID	Age	Overall	Potential	Specia
count	18207.000000	18207.000000	18207.000000	18207.000000	18207.000000	18207.000000
mean	9103.000000	214298.338606	25.122206	66.238699	71.307299	1597.809908
std	5256.052511	29965.244204	4.669943	6.908930	6.136496	272.586016
min	0.000000	16.000000	16.000000	46.000000	48.000000	731.000000
25%	4551.500000	200315.500000	21.000000	62.000000	67.000000	1457.000000
50%	9103.000000	221759.000000	25.000000	66.000000	71.000000	1635.000000
75%	13654.500000	236529.500000	28.000000	71.000000	75.000000	1787.000000
max	18206.000000	246620.000000	45.000000	94.000000	95.000000	2346.000000

We can see that the two first columns do not provide any meaningful information so we can delete them.

In [7]:
```
base = base.iloc[:, 2:]
base
```

Out[7]:

	Name	Age	Photo	Nationality	
0	L. Messi	31	https://cdn.sofifa.org/players/4/19/158023.png	Argentina	https://cdn.s
1	Cristiano Ronaldo	33	https://cdn.sofifa.org/players/4/19/20801.png	Portugal	https://cdn.s
2	Neymar Jr	26	https://cdn.sofifa.org/players/4/19/190871.png	Brazil	https://cdn.s
3	De Gea	27	https://cdn.sofifa.org/players/4/19/193080.png	Spain	https://cdn.s
4	K. De Bruyne	27	https://cdn.sofifa.org/players/4/19/192985.png	Belgium	https://cdn.
...	...	...	...	...	
18202	J. Lundstram	19	https://cdn.sofifa.org/players/4/19/238813.png	England	https://cdn.s
18203	N. Christoffersson	19	https://cdn.sofifa.org/players/4/19/243165.png	Sweden	https://cdn.s
18204	B. Worman	16	https://cdn.sofifa.org/players/4/19/241638.png	England	https://cdn.s
18205	D. Walker-Rice	17	https://cdn.sofifa.org/players/4/19/246268.png	England	https://cdn.s
18206	G. Nugent	16	https://cdn.sofifa.org/players/4/19/246269.png	England	https://cdn.s

18207 rows × 87 columns

Con la funcion describe(), fijándonos en la desviación típica, podemos comprobar que existen datos que no nos proporcionan información importante para evaluar el valor de un jugador. Esta información solo provocará que empeore nuestro modelo de entrenamiento

Por lo tanto, eliminaremos las columnas : Nombre, Foto, Bandera, Logo del Club,Cara Real y Dorsal.

In [8]:
```
base.drop(['Name','Photo','Flag','Club Logo','Real Face','Jersey Number'
base
```

Out[8]:

	Age	Nationality	Overall	Potential	Club	Value	Wage	Special	Preferred Foot
0	31	Argentina	94	94	FC Barcelona	€110.5M	€565K	2202	Left
1	33	Portugal	94	94	Juventus	€77M	€405K	2228	Right
2	26	Brazil	92	93	Paris Saint-Germain	€118.5M	€290K	2143	Right
3	27	Spain	91	93	Manchester United	€72M	€260K	1471	Right
4	27	Belgium	91	92	Manchester City	€102M	€355K	2281	Right
...	...	...	...	...	...	...	...	...	...
18202	19	England	47	65	Crewe Alexandra	€60K	€1K	1307	Right
18203	19	Sweden	47	63	Trelleborgs FF	€60K	€1K	1098	Right
18204	16	England	47	67	Cambridge United	€60K	€1K	1189	Right
18205	17	England	47	66	Tranmere Rovers	€60K	€1K	1228	Right
18206	16	England	46	66	Tranmere Rovers	€60K	€1K	1321	Right

18207 rows × 81 columns

Por otro lado, las características Joined and Loaned From son más importantes que las anteriores a la hora de construir el modelo de predicción, pero no lo suficiente para contar con ellas. Por lo tanto, las eliminaremos

In [9]:
```
base.drop(['Joined','Loaned From','Contract Valid Until'], axis = 1,inpl
base.head()
```

Out[9]:

	Age	Nationality	Overall	Potential	Club	Value	Wage	Special	Preferred Foot	Inter Re
0	31	Argentina	94	94	FC Barcelona	€110.5M	€565K	2202	Left	
1	33	Portugal	94	94	Juventus	€77M	€405K	2228	Right	
2	26	Brazil	92	93	Paris Saint-Germain	€118.5M	€290K	2143	Right	
3	27	Spain	91	93	Manchester United	€72M	€260K	1471	Right	

4	27	Belgium	91	92	Manchester City	€102M	€355K	2281	Right	

Gestión de valores no numericos

La mayoría de datos no están en el formato correcto, en float y contienen NaNs por ello, habrá que tratarlos.

Observamos que hay valores que no están en el formato correcto, entre ellos : Valor, Salario, Cláusula de recisión, Peso, Altura

El valor, salario y cláusula de recisión están en formato monetario, en € concretamente. Por otro lado, las abreviaturas 'K' y 'M' representan mil euros y millón de euros respectivamente.

Esta función nos transforma value a float.

In [10]:
```
def value_to_float(x):
    """
    From K and M to float.

    """
    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
```

Esta función nos transforma wage a float.

In [11]:
```
def wage_to_float(x):
    """
    From K and M to float.

    """
    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
```

Esta función nos transforma release clause a float.

In [12]:
```
def releaseclause_to_float(x):
    """
    From K and M to float.

    """
    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
```

Las funciones funcionaron bien en las columnas Valor y Salario,como se puede comprobar abajo.Por otro lado, en la clausula de recisión me dió error pues contiene valores NaN.

In [13]:
```
base["Value"] = base["Value"].apply(value_to_float)
base["Wage"] = base["Wage"].apply(wage_to_float)
base.head()
```

Out[13]:

	Age	Nationality	Overall	Potential	Club	Value	Wage	Special	Preferred Foo
0	31	Argentina	94	94	FC Barcelona	110500000.0	565000.0	2202	Lef
1	33	Portugal	94	94	Juventus	77000000.0	405000.0	2228	Righ
2	26	Brazil	92	93	Paris Saint-Germain	118500000.0	290000.0	2143	Righ
3	27	Spain	91	93	Manchester United	72000000.0	260000.0	1471	Righ
4	27	Belgium	91	92	Manchester City	102000000.0	355000.0	2281	Righ

Se puede comprobar con esta función.

In [14]:
```
base.loc[base['Release Clause'].isnull()]
```

Out[14]:

	Age	Nationality	Overall	Potential	Club	Value	Wage	Special	Prefe
28	26	Colombia	88	89	FC Bayern München	69500000.0	315000.0	2171	
38	30	Argentina	88	88	Milan	57000000.0	245000.0	1965	F
91	29	Brazil	85	85	Guangzhou Evergrande	37000000.0	235000.0	2245	F

					Taobao FC				
166	24	Brazil	83	90	Guangzhou Evergrande Taobao FC	36500000.0	18000.0	2142	
176	24	Croatia	83	89	Chelsea	35000000.0	165000.0	2115	F
...	...	...	...	...	...	...	...	...	
17979	21	China PR	51	60	Guizhou Hengfeng FC	60000.0	2000.0	1459	F
18026	21	China PR	50	59	Guizhou Hengfeng FC	50000.0	2000.0	1440	F
18031	20	China PR	50	61	Stabæk Fotball	40000.0	2000.0	1278	F
18056	19	Italy	50	65	Ascoli	60000.0	3000.0	1450	
18183	44	England	48	48	Cambridge United	0.0	1000.0	774	F

1564 rows × 78 columns

Como gestioné esos NaNs? Al haber una gran cantidad de jugadores con esta característica y sobretodo por ser relativa al valor, no era una opción viable eliminarlos.Por ello, para resolverlo, sustituí(imputé) los NaNs por 0

In [15]:
```
base["Release Clause"].fillna('0€', inplace=True)
```

Comprobamos como la función funciona correctamente, pues en estos jugadores la característica Release Clause es igual a 0 €

In [16]:
```
base.loc[base['Release Clause'] == '0€']
```

Out[16]:

	Age	Nationality	Overall	Potential	Club	Value	Wage	Special	Prefe
28	26	Colombia	88	89	FC Bayern München	69500000.0	315000.0	2171	
38	30	Argentina	88	88	Milan	57000000.0	245000.0	1965	F
91	29	Brazil	85	85	Guangzhou Evergrande Taobao FC	37000000.0	235000.0	2245	F
166	24	Brazil	83	90	Guangzhou Evergrande Taobao FC	36500000.0	18000.0	2142	
176	24	Croatia	83	89	Chelsea	35000000.0	165000.0	2115	F
...	...	...	...	...	...	...	...	...	
17979	21	China PR	51	60	Guizhou Hengfeng FC	60000.0	2000.0	1459	F
18026	21	China PR	50	59	Guizhou	50000.0	2000.0	1440	F

					Hengfeng FC				
18031	20	China PR	50	61	Stabæk Fotball	40000.0	2000.0	1278	F
18056	19	Italy	50	65	Ascoli	60000.0	3000.0	1450	
18183	44	England	48	48	Cambridge United	0.0	1000.0	774	F

1564 rows × 78 columns

Ahora si podemos aplicar la función que transforma los valores monetarios de la cláusula de recisión a float

In [17]:
```
base["Release Clause"] = base["Release Clause"].apply(wage_to_float)
```

Comprobamos que en la última columna la característica Cláusula de reicisión es float

In [18]:
```
base.head()
```

Out[18]:

	Age	Nationality	Overall	Potential	Club	Value	Wage	Special	Preferred Foo
0	31	Argentina	94	94	FC Barcelona	110500000.0	565000.0	2202	Lef
1	33	Portugal	94	94	Juventus	77000000.0	405000.0	2228	Righ
2	26	Brazil	92	93	Paris Saint-Germain	118500000.0	290000.0	2143	Righ
3	27	Spain	91	93	Manchester United	72000000.0	260000.0	1471	Righ
4	27	Belgium	91	92	Manchester City	102000000.0	355000.0	2281	Righ

Ahora me fijo en otras características que tampoco están en el formato adecuado como el peso y la altura que pueden ser importantes a la hora de predecir el valor del jugador. Por ello, las transformaremos a float con estas funciones.

In [19]:
```
def height_to_float(x):
    """
    x'x -> x.x
    """
    x = x.split("'")

    if(len(x[1])==1): x = float(x[0]) + float(x[1])/10
    else:
        y = float(x[1])
        y = y/100
        x = float(x[0])+y

    return x
```

Tras intentar realizar, lo mismo con la altura nos da el mismo error conocido anteriormente en la cláusula de recisión.Tenemos jugadores con altura NaNs

In [20]:
```
base.loc[base['Height'].isnull()]
```

Out[20]:

	Age	Nationality	Overall	Potential	Club	Value	Wage	Special	Preferred Foo
13236	33	Scotland	62	62	Rochdale	120000.0	1000.0	1510	NaN
13237	29	Nicaragua	62	62	Boyacá Chicó FC	300000.0	1000.0	1532	NaN
13238	35	England	62	62	Notts County	140000.0	3000.0	1573	NaN
13239	20	Italy	62	72	Brescia	425000.0	1000.0	1610	NaN
13240	24	England	62	66	Hamilton Academical FC	400000.0	1000.0	1481	NaN
13241	21	Poland	62	72	Śląsk Wrocław	425000.0	1000.0	1692	NaN
13242	23	Argentina	62	70	Club Atlético Aldosivi	450000.0	2000.0	1663	NaN
13243	19	England	62	78	Everton	600000.0	5000.0	1328	NaN
13244	30	Denmark	62	62	Hobro IK	230000.0	2000.0	1244	NaN
13245	21	Brazil	62	69	HJK Helsinki	425000.0	1000.0	1549	NaN
13246	22	France	62	68	AS Béziers	425000.0	2000.0	1494	NaN
13247	28	Austria	62	62	SV Mattersburg	240000.0	3000.0	1630	NaN
13248	24	Northern Ireland	62	69	Tranmere Rovers	375000.0	2000.0	1461	NaN
13249	27	China PR	62	62	Shanghai Greenland Shenhua FC	250000.0	3000.0	1636	NaN
13250	29	Paraguay	62	62	Itagüí Leones FC	300000.0	1000.0	1454	NaN
13251	34	Belgium	62	62	NAC Breda	150000.0	2000.0	1665	NaN
13252	22	Albania	62	70	Malmö FF	375000.0	1000.0	1587	NaN
13253	31	Scotland	62	62	Carlisle United	200000.0	2000.0	1535	NaN
13254	17	Germany	62	82	VfB Stuttgart	550000.0	2000.0	1418	NaN
13255	25	Germany	62	65	Hamilton Academical FC	325000.0	1000.0	1693	NaN
13256	26	England	62	62	Dundee FC	325000.0	1000.0	1712	NaN
13257	25	Korea Republic	62	65	Suwon Samsung Bluewings	375000.0	2000.0	1536	NaN
13258	23	Saudi Arabia	62	67	Al Wehda	350000.0	3000.0	1664	NaN
13259	27	Chile	62	65	CD Palestino	300000.0	1000.0	1316	NaN

13260	20	Spain	62	69	Albacete BP	425000.0	1000.0	1574	NaN
13261	25	Saudi Arabia	62	64	Al Nassr	300000.0	5000.0	1665	NaN
13262	24	Germany	62	66	TSV 1860 München	325000.0	1000.0	1625	NaN
13263	25	Austria	62	66	FC Admira Wacker Mödling	325000.0	2000.0	1354	NaN
13264	29	France	62	62	Grenoble Foot 38	300000.0	1000.0	1620	NaN
13265	28	England	62	62	Oldham Athletic	300000.0	3000.0	1740	NaN
13266	20	Sweden	62	73	Hammarby IF	525000.0	1000.0	1549	NaN
13267	22	Colombia	62	70	Itagüí Leones FC	450000.0	1000.0	1607	NaN
13268	37	Poland	62	62	Miedź Legnica	100000.0	1000.0	1542	NaN
13269	26	Colombia	62	64	Jaguares de Córdoba	375000.0	1000.0	1552	NaN
13270	19	Brazil	62	77	Bologna	525000.0	1000.0	1141	NaN
13271	26	Chile	62	63	CD Antofagasta	290000.0	1000.0	1497	NaN
13272	22	Scotland	62	70	Dundee FC	450000.0	1000.0	1614	NaN
13273	25	Estonia	62	67	Kristiansund BK	400000.0	1000.0	1686	NaN
13274	27	Italy	62	62	TSV 1860 München	325000.0	2000.0	1670	NaN
13275	23	Venezuela	62	70	Boyacá Chicó FC	375000.0	1000.0	1542	NaN
13276	26	Saudi Arabia	62	63	Al Raed	290000.0	3000.0	1679	NaN
13277	26	Chile	62	62	Deportes Iquique	325000.0	1000.0	1675	NaN
13278	17	Netherlands	62	79	FC Utrecht	550000.0	1000.0	1545	NaN
13279	22	Italy	62	69	Perugia	350000.0	1000.0	1681	NaN
13280	19	France	62	77	Montpellier HSC	650000.0	2000.0	1478	NaN
13281	27	Korea Republic	62	62	Gyeongnam FC	300000.0	1000.0	1729	NaN
13282	25	Mexico	62	65	Tiburones Rojos de Veracruz	375000.0	2000.0	1661	NaN
13283	25	China PR	62	66	Guizhou Hengfeng FC	325000.0	2000.0	1578	NaN

Observamos que esta lista escasa de jugadores contiene excesivos NaN, por lo tanto, es mejor eliminarlos directamente ya que nos darán también problemas con la altura .

Además van a perjudicar a nuestro dataset de entrenamiento pues no contienen información.

In [21]:
```
base = base.dropna(subset = ['Height'])
```

In [22]:
```
base["Height"] =base["Height"].apply(height_to_float)
base.head()
```

```
C:\phyton\envs\analisi_dades\lib\site-packages\ipykernel_launcher.py:1: S
ettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-do
cs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  """Entry point for launching an IPython kernel.
```

Out[22]:

	Age	Nationality	Overall	Potential	Club	Value	Wage	Special	Preferre Foo
0	31	Argentina	94	94	FC Barcelona	110500000.0	565000.0	2202	Lef
1	33	Portugal	94	94	Juventus	77000000.0	405000.0	2228	Righ
2	26	Brazil	92	93	Paris Saint-Germain	118500000.0	290000.0	2143	Righ
3	27	Spain	91	93	Manchester United	72000000.0	260000.0	1471	Righ
4	27	Belgium	91	92	Manchester City	102000000.0	355000.0	2281	Righ

In [23]:
```
def weight_to_float(x):
    """
    From lbs to float.

    """
    if 'lbs' in x:
            ret_val = float(x.replace('lbs', ' '))
    return ret_val
```

In [24]:
```
base["Weight"] = base["Weight"].apply(weight_to_float)
base.head()
```

```
C:\phyton\envs\analisi_dades\lib\site-packages\ipykernel_launcher.py:1: S
ettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-do
cs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  """Entry point for launching an IPython kernel.
```

Out[24]:

	Age	Nationality	Overall	Potential	Club	Value	Wage	Special	Preferre Foo
0	31	Argentina	94	94	FC Barcelona	110500000.0	565000.0	2202	Lef
1	33	Portugal	94	94	Juventus	77000000.0	405000.0	2228	Righ

2	26	Brazil	92	93	Paris Saint-Germain	118500000.0	290000.0	2143	Righ
3	27	Spain	91	93	Manchester United	72000000.0	260000.0	1471	Righ
4	27	Belgium	91	92	Manchester City	102000000.0	355000.0	2281	Righ

Ahora, comprobamos que existen valores como Nacionalidad,Club,Tipo de Cuerpo, Posición y Pie Preferente que no tienen valores númericos y son valores categóricos. Para ello, aplicaremos la técnica(hot-encoding) usada basándome en la del enunciado con Club y convertiremos los valores a binarios. De esta manera, nos evitaremos problemas con NaNs y podremos convertir los valores no númericos a númericos

In [25]:
```
wf = base.pop("Nationality")
base = pd.concat([base.reset_index(drop=True), pd.get_dummies(wf, prefix=
filtro_Pos = [col for col in base if col.startswith('nat')]
base.drop(columns=filtro_Pos)

wf = base.pop("Club")
base = pd.concat([base.reset_index(drop=True), pd.get_dummies(wf, prefix=
filtro_Pos = [col for col in base if col.startswith('clb')]
base.drop(columns=filtro_Pos)

wf = base.pop("Body Type")
base = pd.concat([base.reset_index(drop=True), pd.get_dummies(wf, prefix=
filtro_Pos = [col for col in base if col.startswith('bdy')]
base.drop(columns=filtro_Pos)

wf = base.pop("Position")
base = pd.concat([base.reset_index(drop=True), pd.get_dummies(wf, prefix=
filtro_Pos = [col for col in base if col.startswith('pos')]
base.drop(columns=filtro_Pos)

wf = base.pop("Preferred Foot")
base = pd.concat([base.reset_index(drop=True), pd.get_dummies(wf, prefix=
filtro_Pos = [col for col in base if col.startswith('foot')]
base.drop(columns=filtro_Pos)

base.head()
```

Out[25]:

	Age	Overall	Potential	Value	Wage	Special	International Reputation	Weak Foot	Skill Moves	
0	31	94	94	110500000.0	565000.0	2202	5.0	4.0	4.0	Mec Me
1	33	94	94	77000000.0	405000.0	2228	5.0	4.0	5.0	
2	26	92	93	118500000.0	290000.0	2143	5.0	5.0	5.0	Me
3	27	91	93	72000000.0	260000.0	1471	4.0	3.0	1.0	Mec Me
4	27	91	92	102000000.0	355000.0	2281	4.0	5.0	4.0	

Otros valor no número es Work Rate, este se divide según su valoración en ataque y defensa, el valor antes de la barra es el de ataque y el valor de despues de la barra es el de defensa. Sigue este formato --> "Valoración en ataque"/ "Valoracion en Defensa".Este valor puede ser "High","Medium" o "Low".
Para ello, a través de esta función, dividiremos en dos columnas el valor en ataque y el valor en defensa, de esta manera tendremos más facilidad de tratarlo.

In [26]:
```
base[['Attack Work Rate','Defense Work Rate']] = base['Work Rate'].str.s
base.pop("Work Rate")
base.head()
```

Out[26]:

	Age	Overall	Potential	Value	Wage	Special	International Reputation	Weak Foot	Skill Moves	Hei
0	31	94	94	110500000.0	565000.0	2202	5.0	4.0	4.0	5
1	33	94	94	77000000.0	405000.0	2228	5.0	4.0	5.0	6
2	26	92	93	118500000.0	290000.0	2143	5.0	5.0	5.0	5
3	27	91	93	72000000.0	260000.0	1471	4.0	3.0	1.0	6
4	27	91	92	102000000.0	355000.0	2281	4.0	5.0	4.0	5

Y ahora volveremos a aplicar la técnica utilizada anteriormente para convertir valores categóricos a binarios

In [27]:
```
wf = base.pop("Attack Work Rate")
base = pd.concat([base.reset_index(drop=True), pd.get_dummies(wf, prefix
filtro_Pos = [col for col in base if col.startswith('AWR')]
base.drop(columns=filtro_Pos)

wf = base.pop("Defense Work Rate")
base = pd.concat([base.reset_index(drop=True), pd.get_dummies(wf, prefix
filtro_Pos = [col for col in base if col.startswith('DWR')]
base.drop(columns=filtro_Pos)
base.head()
```

Out[27]:

	Age	Overall	Potential	Value	Wage	Special	International Reputation	Weak Foot	Skill Moves	Hei
0	31	94	94	110500000.0	565000.0	2202	5.0	4.0	4.0	5
1	33	94	94	77000000.0	405000.0	2228	5.0	4.0	5.0	6
2	26	92	93	118500000.0	290000.0	2143	5.0	5.0	5.0	5
3	27	91	93	72000000.0	260000.0	1471	4.0	3.0	1.0	6
4	27	91	92	102000000.0	355000.0	2281	4.0	5.0	4.0	5

Observamos que deberíamos tratar los valores de rendimiento del jugador en cada posición(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) . Para ello, simplemte sumamos los valores para que quede en un solo valor número. Con esta función se sumarán los dos valores:

In [28]:
```
def add_pos(x):
    """
    Suma de los valores.

    """

    if(x != x):
        return np.nan

    values = x.split('+')

    return (float(values[0])+float(values[1]))
```

Utilizamos dicha función para aplicarla a cada posición posible

In [29]:
```
base["LS"] = base["LS"].apply(add_pos)
base["ST"] = base["ST"].apply(add_pos)
base["RS"] = base["RS"].apply(add_pos)
base["LW"] = base["LW"].apply(add_pos)
base["LF"] = base["LF"].apply(add_pos)
base["CF"] = base["CF"].apply(add_pos)
base["RF"] = base["RF"].apply(add_pos)
base["RW"] = base["RW"].apply(add_pos)
base["LAM"] = base["LAM"].apply(add_pos)
base["CAM"] = base["CAM"].apply(add_pos)
base["RAM"] = base["RAM"].apply(add_pos)
base["LM"] = base["LM"].apply(add_pos)
base["LCM"] = base["LCM"].apply(add_pos)
base["CM"] = base["CM"].apply(add_pos)
base["RCM"] = base["RCM"].apply(add_pos)
base["RM"] = base["RM"].apply(add_pos)
base["LWB"] = base["LWB"].apply(add_pos)
base["LDM"] = base["LDM"].apply(add_pos)
base["CDM"] = base["CDM"].apply(add_pos)
base["RDM"] = base["RDM"].apply(add_pos)
base["RWB"] = base["RWB"].apply(add_pos)
base["LB"] = base["LB"].apply(add_pos)
base["LCB"] = base["LCB"].apply(add_pos)
base["CB"] = base["CB"].apply(add_pos)
base["RCB"] = base["RCB"].apply(add_pos)
base["RB"] = base["RB"].apply(add_pos)
base.head()
```

Out[29]:

	Age	Overall	Potential	Value	Wage	Special	International Reputation	Weak Foot	Skill Moves	Hei
0	31	94	94	110500000.0	565000.0	2202	5.0	4.0	4.0	5
1	33	94	94	77000000.0	405000.0	2228	5.0	4.0	5.0	6
2	26	92	93	118500000.0	290000.0	2143	5.0	5.0	5.0	5
3	27	91	93	72000000.0	260000.0	1471	4.0	3.0	1.0	6
4	27	91	92	102000000.0	355000.0	2281	4.0	5.0	4.0	5

Por último, vemos como David de Gea (y el resto de porteros) tiene NaNs en estos valores, para ello podríamos sustuir estos modificadores de posición por ceros, pero no me parece la solución más óptima. Una posible solución, es rellenar las NaNs con las

medias de las columnas, no será la solución más eficiente pero es una posible solución alterna a tener que imputar ceros.

In [30]:
```
base.fillna(base.mean(), inplace=True)
base.head()
```

Out[30]:

	Age	Overall	Potential	Value	Wage	Special	International Reputation	Weak Foot	Skill Moves	Hei
0	31	94	94	110500000.0	565000.0	2202	5.0	4.0	4.0	5
1	33	94	94	77000000.0	405000.0	2228	5.0	4.0	5.0	6
2	26	92	93	118500000.0	290000.0	2143	5.0	5.0	5.0	5
3	27	91	93	72000000.0	260000.0	1471	4.0	3.0	1.0	6
4	27	91	92	102000000.0	355000.0	2281	4.0	5.0	4.0	5

Prediction

Finalmente, ya tenemos el dataset construido con los datos que consideros mas relativos para la variable objetivo "Valor".

In [31]:
```
val = base.pop("Value")
base
```

Out[31]:

	Age	Overall	Potential	Wage	Special	International Reputation	Weak Foot	Skill Moves	Height	Weig
0	31	94	94	565000.0	2202	5.0	4.0	4.0	5.70	159
1	33	94	94	405000.0	2228	5.0	4.0	5.0	6.20	183
2	26	92	93	290000.0	2143	5.0	5.0	5.0	5.90	150
3	27	91	93	260000.0	1471	4.0	3.0	1.0	6.40	168
4	27	91	92	355000.0	2281	4.0	5.0	4.0	5.11	154
...	...	...	...	...	...	...	...	...	...	
18154	19	47	65	1000.0	1307	1.0	2.0	2.0	5.90	134
18155	19	47	63	1000.0	1098	1.0	2.0	2.0	6.30	170
18156	16	47	67	1000.0	1189	1.0	3.0	2.0	5.80	148
18157	17	47	66	1000.0	1228	1.0	3.0	2.0	5.10	154
18158	16	46	66	1000.0	1321	1.0	3.0	2.0	5.10	176

18159 rows × 931 columns

Now we have only the club information, in hot-econding style. At the same time we have the data we want to predict in a numeric format, the value information. So now we can use this data to train a model. But first we need to split the data to being able to known its true performance.

In [32]:
```
X_train, X_test, y_train, y_test = train_test_split(base, val, test_size=
```

In [33]:
```
len(X_train)
```

Out[33]:
```
12166
```

Now we train a LinearRegression model.

In [34]:
```
reg = linear_model.LinearRegression().fit(X_train, y_train)
```

Finally we get a metric R^2 for the regression, we use the implementation from sickit-learn.

In [35]:
```
preds = reg.predict(X_test)
```

In [36]:
```
preds[0]
```

Out[36]:
```
707282.3607195723
```

In [37]:
```
y_test[0]
```

Out[37]:
```
110500000.0
```

In [38]:
```
r2_score(y_test, preds)
```

Out[38]:
```
0.9641759434065622
```

Obtenemos un 96% de fiabilidad, un valor muy cercano al 100%, lo cual nos demuestra que he contruido el dataset base con datos de entrenamiento relativos a la variable objetivo Valor