Práctica de Aprendizaje Automático

Nombre:Arkadiy Kosyuk NIE:X8048330D

El objetivo de esta practica es conocer el valor de un jugador a partir de los datos proporcionados, por tato debemos hacer un tratamiento de los datos proporcionados y posteriormente hacer la regresion lineal.

Importar librerias

Primero de todo importamos las librerias que vamos a utilizar

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
```

Leemos los datos

Para leer la información emplearemos `pandas` y `os`.

In [2]:

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

Out[2]:

	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	F
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	M
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	M

5 rows × 89 columns

Analisis de datos

Como podemos observar a continuación el dataset contiene varias columnas que son irrelevantes para el entrenamiento, por tanto es necesario eliminar estas columnas, aparte de estas columnas que no aportan informacion irrelevante o de gran importancia tambien he decidido quitar la columna de `Release clause` a causa de la gran cantidad de espacios vacios que tiene, esto causaria un desajuste en los valores finales y si son substituidas no veriamos valores reales por tanto he decidido que es mejor quitarlo directamente del dataset.

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	F
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	M
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	M

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	

In [6]:

```
df = df.drop(columns= ['Unnamed: 0',"ID","Name","Photo" ,"Nationality","Flag", "Club Logo","Preferred Foo
                       "Real Face","Jersey Number","Joined", "Loaned From", "Contract Valid Until","Heig
```

Modificar NaNs

Primero miramos donde tenemos NaNs

In [7]:

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

Out[7]:

```
['Club',
 'International Reputation',
 'Weak Foot',
 'Skill Moves',
 'Position',
 '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',
 'GKDiving',
 'GKHandling',
 'GKKicking',
 'GKPositioning',
 'GKReflexes']
```

Vemos que hay varias columnas que contienen NaN, esto hay que solucionarlo y esto se puede hacer eliminando estos nans del dataset, el problema es que si hay muchos NaNs en la columna y los quitamos el entrenamiento no se hara correctamente, razon por la cual elimine el Release Clause, además tambien hay columnas con Strings y por tanto hay que pasarlo a forma numérica.

In [8]:

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

Out[8]:

	Age	Overall	Potential	Club	Value	Wage	Special	International Reputation	Weak Foot	Skill Moves	Position	LS	ST	RS	LW	LF	CF	RF
452	24	80	85	NaN	€0	€0	2122	2.0	4.0	4.0	CM	71+2	71+2	71+2	75+2	75+2	75+2	75+2
538	33	80	80	NaN	€0	€0	1797	2.0	4.0	2.0	LCB	62+2	62+2	62+2	56+2	58+2	58+2	58+2
568	26	79	81	NaN	€0	€0	1217	1.0	3.0	1.0	GK	NaN	NaN	NaN	NaN	NaN	NaN	NaN
677	29	79	79	NaN	€0	€0	2038	2.0	3.0	3.0	RB	70+2	70+2	70+2	73+2	72+2	72+2	72+2
874	29	78	78	NaN	€0	€0	1810	2.0	3.0	3.0	ST	77+2	77+2	77+2	71+2	74+2	74+2	74+2
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
17197	21	55	64	NaN	€0	€0	838	1.0	2.0	1.0	GK	NaN	NaN	NaN	NaN	NaN	NaN	NaN
17215	26	55	57	NaN	€0	€0	1366	1.0	3.0	2.0	RB	46+2	46+2	46+2	45+2	44+2	44+2	44+2
17339	23	54	63	NaN	€0	€0	1321	1.0	3.0	2.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
17436	20	54	67	NaN	€0	€0	1270	1.0	3.0	2.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
17539	21	53	62	NaN	€0	€0	1247	1.0	3.0	2.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

241 rows × 71 columns

Aqui podemos ver que en comparacion con nuestro dataset hay muy pocas filas con NaN en la columna Club por tanto eliminamos estos jugadores que no se encuentran en un club.

In [9]:

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

Out[9]:

	Age	Overall	Potential	Club	Value	Wage	Special	International Reputation	Weak Foot	Skill Moves	Position	LS	ST	RS	LW	LF	
0	31	94	94	FC Barcelona	€110.5M	€565K	2202	5.0	4.0	4.0	RF	88+2	88+2	88+2	92+2	93+2	93-
1	33	94	94	Juventus	€77M	€405K	2228	5.0	4.0	5.0	ST	91+3	91+3	91+3	89+3	90+3	90-
2	26	92	93	Paris Saint-Germain	€118.5M	€290K	2143	5.0	5.0	5.0	LW	84+3	84+3	84+3	89+3	89+3	89-
3	27	91	93	Manchester United	€72M	€260K	1471	4.0	3.0	1.0	GK	NaN	NaN	NaN	NaN	NaN	Na
4	27	91	92	Manchester City	€102M	€355K	2281	4.0	5.0	4.0	RCM	82+3	82+3	82+3	87+3	87+3	87-
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
18202	19	47	65	Crewe Alexandra	€60K	€1K	1307	1.0	2.0	2.0	CM	42+2	42+2	42+2	44+2	44+2	44-
18203	19	47	63	Trelleborgs FF	€60K	€1K	1098	1.0	2.0	2.0	ST	45+2	45+2	45+2	39+2	42+2	42-
18204	16	47	67	Cambridge United	€60K	€1K	1189	1.0	3.0	2.0	ST	45+2	45+2	45+2	45+2	46+2	46-
18205	17	47	66	Tranmere Rovers	€60K	€1K	1228	1.0	3.0	2.0	RW	47+2	47+2	47+2	47+2	46+2	46-
18206	16	46	66	Tranmere Rovers	€60K	€1K	1321	1.0	3.0	2.0	CM	43+2	43+2	43+2	45+2	44+2	44-

17966 rows × 71 columns

Ahora realizaremos el mismo analisis anterior pero con la columna "Position". Primero miramos si hay muchos NaNs en cuyo caso se decidiria si eliminar la columna o reyenar esos espacios con un valor determinado.

In [10]:

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

Out[10]:

	Age	Overall	Potential	Club	Value	Wage	Special	International Reputation	Weak Foot	Skill Moves	Position	LS	ST	RS	LW	LF	CF	RF

	Age	Overall	Potential	Club	Value	Wage	Special	International Reputation	Weak Foot	Skill Moves	Position	LS	ST	RS	LW	LF	CF	RF
13236	33	62	62	Rochdale	€120K	€1K	1510	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
13237	29	62	62	Boyacá Chicó FC	€300K	€1K	1532	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
13238	35	62	62	Notts County	€140K	€3K	1573	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
13239	20	62	72	Brescia	€425K	€1K	1610	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
13240	24	62	66	Hamilton Academical FC	€400K	€1K	1481	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
13241	21	62	72	Śląsk Wrocław	€425K	€1K	1692	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
13242	23	62	70	Club Atlético Aldosivi	€450K	€2K	1663	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
13243	19	62	78	Everton	€600K	€5K	1328	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
13244	30	62	62	Hobro IK	€230K	€2K	1244	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
13245	21	62	69	HJK Helsinki	€425K	€1K	1549	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
13246	22	62	68	AS Béziers	€425K	€2K	1494	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
13247	28	62	62	SV Mattersburg	€240K	€3K	1630	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
13248	24	62	69	Tranmere Rovers	€375K	€2K	1461	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
13249	27	62	62	Shanghai Greenland Shenhua FC	€250K	€3K	1636	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
13250	29	62	62	Itagüí Leones FC	€300K	€1K	1454	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
13251	34	62	62	NAC Breda	€150K	€2K	1665	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
13252	22	62	70	Malmö FF	€375K	€1K	1587	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
13253	31	62	62	Carlisle United	€200K	€2K	1535	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
13254	17	62	82	VfB Stuttgart	€550K	€2K	1418	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
13255	25	62	65	Hamilton Academical FC	€325K	€1K	1693	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
13256	26	62	62	Dundee FC	€325K	€1K	1712	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
13257	25	62	65	Suwon Samsung Bluewings	€375K	€2K	1536	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
13258	23	62	67	Al Wehda	€350K	€3K	1664	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
13259	27	62	65	CD Palestino	€300K	€1K	1316	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
13260	20	62	69	Albacete BP	€425K	€1K	1574	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
13261	25	62	64	Al Nassr	€300K	€5K	1665	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
13262	24	62	66	TSV 1860 München	€325K	€1K	1625	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
13263	25	62	66	FC Admira Wacker Mödling	€325K	€2K	1354	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
13264	29	62	62	Grenoble Foot 38	€300K	€1K	1620	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
13265	28	62	62	Oldham Athletic	€300K	€3K	1740	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
13266	20	62	73	Hammarby IF	€525K	€1K	1549	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
13267	22	62	70	Itagüí Leones FC	€450K	€1K	1607	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

13268	27	62	61	Miedź Legnica	€110K	€1K	1541	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
13269	26	62	64	Jaguares de Córdoba	€375K	€1K	1552	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
13270	19	62	77	Bologna	€525K	€1K	1141	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
13271	26	62	63	CD Antofagasta	€290K	€1K	1497	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
13272	22	62	70	Dundee FC	€450K	€1K	1614	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
13273	25	62	67	Kristiansund BK	€400K	€1K	1686	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
13274	27	62	62	TSV 1860 München	€325K	€2K	1670	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
13275	23	62	70	Boyacá Chicó FC	€375K	€1K	1542	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
13276	26	62	63	Al Raed	€290K	€3K	1679	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
13277	26	62	62	Deportes Iquique	€325K	€1K	1675	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
13278	17	62	79	FC Utrecht	€550K	€1K	1545	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
13279	22	62	69	Perugia	€350K	€1K	1681	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
13280	19	62	77	Montpellier HSC	€650K	€2K	1478	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
13281	27	62	62	Gyeongnam FC	€300K	€1K	1729	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
13282	25	62	65	Tiburones Rojos de Veracruz	€375K	€2K	1661	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
13283	25	62	66	Guizhou Hengfeng FC	€325K	€2K	1578	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

Como no hay muchas filas con este valor vacio eliminamos los jugadores que tienen NaNs en `Posición`

In [11]:

```
df = df.dropna(subset = ['Position'])
df.head()
```

Out[11]:

	Age	Overall	Potential	Club	Value	Wage	Special	International Reputation	Weak Foot	Skill Moves	Position	LS	ST	RS	LW	LF	CF	
0	31	94	94	FC Barcelona	€110.5M	€565K	2202	5.0	4.0	4.0	RF	88+2	88+2	88+2	92+2	93+2	93+2	9
1	33	94	94	Juventus	€77M	€405K	2228	5.0	4.0	5.0	ST	91+3	91+3	91+3	89+3	90+3	90+3	9
2	26	92	93	Paris Saint-Germain	€118.5M	€290K	2143	5.0	5.0	5.0	LW	84+3	84+3	84+3	89+3	89+3	89+3	8
3	27	91	93	Manchester United	€72M	€260K	1471	4.0	3.0	1.0	GK	NaN	NaN	NaN	NaN	NaN	NaN	
4	27	91	92	Manchester City	€102M	€355K	2281	4.0	5.0	4.0	RCM	82+3	82+3	82+3	87+3	87+3	87+3	8

In [12]:

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

Out[12]:

	Age	Overall	Potential	Club	Value	Wage	Special	International Reputation	Weak Foot	Skill Moves	Position	LS	ST	RS	LW	LF	CF	RI
3	27	91	93	Manchester United	€72M	€260K	1471	4.0	3.0	1.0	GK	NaN	NaN	NaN	NaN	NaN	NaN	Na
9	25	90	93	Atlético Madrid	€68M	€94K	1331	3.0	3.0	1.0	GK	NaN	NaN	NaN	NaN	NaN	NaN	Na
18	26	89	92	FC Barcelona	€58M	€240K	1328	3.0	4.0	1.0	GK	NaN	NaN	NaN	NaN	NaN	NaN	Na
19	26	89	90	Real Madrid	€53.5M	€240K	1311	4.0	2.0	1.0	GK	NaN	NaN	NaN	NaN	NaN	NaN	Na
22	32	89	89	FC Bayern München	€38M	€130K	1473	5.0	4.0	1.0	GK	NaN	NaN	NaN	NaN	NaN	NaN	Na
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	..
18178	18	48	65	Dalkurd FF	€50K	€1K	738	1.0	2.0	1.0	GK	NaN	NaN	NaN	NaN	NaN	NaN	Na
18180	22	48	58	St. Johnstone FC	€40K	€1K	987	1.0	2.0	1.0	GK	NaN	NaN	NaN	NaN	NaN	NaN	Na
18183	44	48	48	Cambridge United	€0	€1K	774	1.0	2.0	1.0	GK	NaN	NaN	NaN	NaN	NaN	NaN	Na
18194	18	47	65	Lecce	€50K	€1K	731	1.0	3.0	1.0	GK	NaN	NaN	NaN	NaN	NaN	NaN	Na
18198	18	47	70	Burton Albion	€60K	€1K	792	1.0	2.0	1.0	GK	NaN	NaN	NaN	NaN	NaN	NaN	Na

1992 rows × 71 columns

Aquí podemos ver una gran cantidad de filas que tienen este parametro vacío como los ST,RS ... En esta ocasión en vez de borrar los jugadores porque borrariamos muchos y en vez de borrar estas columnas lo que realizare sera rellenar estos NaNs con la media de la columna, utilizaremos la media porque es el valor que menos desviara el valor a predecir. Para ello primero cambiaremos el valor de estas columnas a un valor numerico sumando el primer valor con el segundo, para realizarlo definiremos una funcion que dividira el valor en dos quitando asi la suma y posteriormente sumamos los dos valores y lo devolvemos.

In [13]:

```
def nuevo_valor(v):

    if not(v != v):
        valor = v.split('+')
        nValor = float(valor[0]) + float(valor[1])
        return nValor
    else:
        return np.nan
```

Arriba tenemos la funcion que nos devuelve el valor necesario, ahora procedere a crear una lista con las columnas que nos dan problemas y las tratare una por una en caso de que se un NaN hacemos la media de la columna e introducimos el valor

In [14]:

```
list = ["LS","ST","RS","LW","LF","CF","RF","RW","LAM","CAM","RAM","LM","LCM","CM","RCM","RM","LWB","LDM",
for i in list:
    df[i] = df[i].apply(nuevo_valor)
    media=df[i].mean()
    df[i].fillna(media, inplace=True)
df.head()
```

```
C:\Users\kosyu\anaconda3\envs\analisi_dades\lib\site-packages\ipykernel_launcher.py:3:
SettingWithCopyWarning:
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-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  This is separate from the ipykernel package so we can avoid doing imports until
C:\Users\kosyu\anaconda3\envs\analisi_dades\lib\site-packages\pandas\core\series.py:4535:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  downcast=downcast,
```

Out[14]:

	Age	Overall	Potential	Club	Value	Wage	Special	International Reputation	Weak Foot	Skill Moves	Position	LS	ST	RS	
0	31	94	94	FC Barcelona	€110.5M	€565K	2202	5.0	4.0	4.0	RF	90.000000	90.000000	90.000000	94.000
1	33	94	94	Juventus	€77M	€405K	2228	5.0	4.0	5.0	ST	94.000000	94.000000	94.000000	92.000
2	26	92	93	Paris Saint-Germain	€118.5M	€290K	2143	5.0	5.0	5.0	LW	87.000000	87.000000	87.000000	92.000
3	27	91	93	Manchester United	€72M	€260K	1471	4.0	3.0	1.0	GK	59.842647	59.842647	59.842647	61.066
4	27	91	92	Manchester City	€102M	€355K	2281	4.0	5.0	4.0	RCM	85.000000	85.000000	85.000000	90.000

Arriba me sale una advertencia de SettingWithCopyWarning pero como la funcion que realizamos completa el dataframe como lo deseamos no hay ningun error. Despues de realizar estas acciones aun nos faltan modificar varias columnas para que esten valores numericos, primero cambiare los String y posteriormente modificare aquellas columnas que son dinero.

In [15]:

```
clb = df.pop("Club")
posi = df.pop("Position")

df = pd.concat([df.reset_index(drop=True), pd.get_dummies(clb, prefix='clb').reset_index(drop=True)], axi
df = pd.concat([df.reset_index(drop=True), pd.get_dummies(posi, prefix='posi').reset_index(drop=True)], a

df.head()
```

Out[15]:

	Age	Overall	Potential	Value	Wage	Special	International Reputation	Weak Foot	Skill Moves	LS	ST	RS	LW	LF	
0	31	94	94	€110.5M	€565K	2202	5.0	4.0	4.0	90.000000	90.000000	90.000000	94.000000	95.000000	95.00
1	33	94	94	€77M	€405K	2228	5.0	4.0	5.0	94.000000	94.000000	94.000000	92.000000	93.000000	93.00
2	26	92	93	€118.5M	€290K	2143	5.0	5.0	5.0	87.000000	87.000000	87.000000	92.000000	92.000000	92.00
3	27	91	93	€72M	€260K	1471	4.0	3.0	1.0	59.842647	59.842647	59.842647	61.066809	60.750471	60.75
4	27	91	92	€102M	€355K	2281	4.0	5.0	4.0	85.000000	85.000000	85.000000	90.000000	90.000000	90.00

Aqui tenemos la funcion que cabia a numerico los valores.

In [16]:

```
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
```

Aplicaremos esta funcion a cada columna que lo necesite.

In [17]:

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

Out[17]:

	Age	Overall	Potential	Value	Wage	Special	International Reputation	Weak Foot	Skill Moves	LS	ST	RS	LW	LF
0	31	94	94	110500000.0	565000.0	2202	5.0	4.0	4.0	90.000000	90.000000	90.000000	94.000000	95.000000
1	33	94	94	77000000.0	405000.0	2228	5.0	4.0	5.0	94.000000	94.000000	94.000000	92.000000	93.000000
2	26	92	93	118500000.0	290000.0	2143	5.0	5.0	5.0	87.000000	87.000000	87.000000	92.000000	92.000000
3	27	91	93	72000000.0	260000.0	1471	4.0	3.0	1.0	59.842647	59.842647	59.842647	61.066809	60.750471
4	27	91	92	102000000.0	355000.0	2281	4.0	5.0	4.0	85.000000	85.000000	85.000000	90.000000	90.000000

Despues de nuestro tratamiento comprobamos que no haya ningun null en ninguna parte del dataset

In [18]:

```
df.isnull().sum().sum()
```

Out[18]:

```
0
```

Predicción

Ya tenemos nuestro dataset listo para el entrenamiento, por tanto sacamos la columna "Value" ya que es nuestra variable a predecir, realizare el entrenamiento con el 25% del dataset

In [19]:

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

In [20]:

```
X_train, X_test, y_train, y_test = train_test_split(df, val, test_size=0.25, random_state=50)
```

In [21]:

```
len(X_train)
```

Out[21]:

```
13438
```

Ahora entrenamos el modelo de Regresión Lineal

In [22]:

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

Finalmente obtenemos una métrica $R2$ para la regresión, usamos la implementación de scikit-learn.

In [23]:

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

In [24]:

```
preds[0]
```

Out[24]:

```
3539957.0805664062
```

In [25]:

```
y test
```

-_

Out[25]:

```
4020     2700000.0
9164      475000.0
3153     2700000.0
7790      725000.0
9097      250000.0
           ...
10230     600000.0
2879     4000000.0
17646      70000.0
6778      850000.0
9304      600000.0
Name: Value, Length: 4480, dtype: float64
```

In [26]:

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

Out[26]:

```
0.8040241252036251
```

Con todas las modificaciones del data frame tenemos un porcentaje de 80% de acierto, no es un mal valor pero podria mejorarse.

In [ ]: