

JOSÉ LUIS JORDÀ MARTÍN - 41543242K

Práctica FIFA

El objetivo de esta práctica es predecir el valor de cualquier jugador en función de un dataset con numerosos atributos, por lo que tendremos que seleccionar las características útiles a la hora de calcular el mismo, eliminando y transformando aquellas que no sean adecuadas.

Repositorio GitHub: <https://github.com/Wilisete/fifa-practice>

Importación de librerías

En primer lugar importaremos las librerías necesarias y leeremos el fichero con toda la información.

```
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
```

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


```

5 rows × 89 columns



Analisis de la información

En primer lugar, tenemos que determinar que atributos nos aportan información útil a la hora del cálculo del valor. Por ejemplo, las dos primeras columnas no nos aportan nada. Entre otros, también eliminaremos atributos como la nacionalidad, la foto del jugador, si tiene la cara real adaptada en el juego, el tipo de cuerpo o valores relativos a su contrato o su club de origen en un traspaso.

Esto nos simplificará los datos a tratar y de esta manera tendremos únicamente aquellos que podamos utilizar de manera directa o a través de transformaciones adicionales.

A lo largo de la práctica se explicará el por qué de la elección de cada atributo, además de una pequeña reflexión de por qué he intentado incluir el mayor número de variables determinantes.

```
In [3]:  
pd.set_option('display.max_columns', None)  
df.head()
```

Out[3]:

	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 N
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 N



Eliminación de los valores no necesarios

En este apartado borraremos todos los atributos que no serán tomados en cuenta o bien porque no son numéricos o bien porque no aportan información relevante. Posteriormente procesaré los datos que no puedan ser utilizados de manera directa como números para que puedan ser procesados. También podríamos aplicar la predicción sobre un array que indicara las columnas aptas, pero eliminarlos simplificaba bastante el proceso.

In [4]:

```
df.drop(['Unnamed: 0', 'Name', 'Nationality', 'ID', 'Photo', 'Club Logo', 'Flag', 'Real Face', 'Body Type', 'Contract Valid Until'],
       axis=1).head()
```

Out[4]:

	Age	Overall	Potential	Club	Value	Wage	Special	Preferred Foot	International Reputation	Weak Foot	Skill Moves	Position	Height	Weight	LS	ST
0	31	94	94	FC Barcelona	€110.5M	€565K	2202	Left	5.0	4.0	4.0	RF	5'7	159lbs	88+2	88+2
1	33	94	94	Juventus	€77M	€405K	2228	Right	5.0	4.0	5.0	ST	6'2	183lbs	91+3	91+3
2	26	92	93	Paris Saint-Germain	€118.5M	€290K	2143	Right	5.0	5.0	5.0	LW	5'9	150lbs	84+3	84+3
3	27	91	93	Manchester United	€72M	€260K	1471	Right	4.0	3.0	1.0	GK	6'4	168lbs	NaN	NaN
4	27	91	92	Manchester City	€102M	€355K	2281	Right	4.0	5.0	4.0	RCM	5'11	154lbs	82+3	82+3



In [5]:

```
## Análisis de las variables de nuevo. Comentado para evitar alargar el documento innecesariamente
## df.describe
```

NaNs

Ahora obtendremos una lista de los valores que contienen NaN (aquellos que no tienen un valor establecido)

In [6]:

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

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

Procedemos a obtener aquellos jugadores que no tienen un club determinado.

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

Out[7]:

	Age	Overall	Potential	Club	Value	Wage	Special	Preferred Foot	International Reputation	Weak Foot	Skill Moves	Position	Height	Weight	LS	ST	RS
452	24	80	85	NaN	€0	€0	2122	Right	2.0	4.0	4.0	CM	5'11	165lbs	71+2	71+2	71+2
538	33	80	80	NaN	€0	€0	1797	Right	2.0	4.0	2.0	LCB	6'4	185lbs	62+2	62+2	62+2
568	26	79	81	NaN	€0	€0	1217	Right	1.0	3.0	1.0	GK	6'2	176lbs	NaN	NaN	NaN
677	29	79	79	NaN	€0	€0	2038	Right	2.0	3.0	3.0	RB	5'10	154lbs	70+2	70+2	70+2
874	29	78	78	NaN	€0	€0	1810	Right	2.0	3.0	3.0	ST	6'5	201lbs	77+2	77+2	77+2
...
17197	21	55	64	NaN	€0	€0	838	Right	1.0	2.0	1.0	GK	6'2	176lbs	NaN	NaN	NaN
17215	26	55	57	NaN	€0	€0	1366	Right	1.0	3.0	2.0	RB	6'4	187lbs	46+2	46+2	46+2
17339	23	54	63	NaN	€0	€0	1321	Right	1.0	3.0	2.0	NaN	5'9	143lbs	NaN	NaN	NaN
17436	20	54	67	NaN	€0	€0	1270	Right	1.0	3.0	2.0	NaN	6'0	168lbs	NaN	NaN	NaN
17539	21	53	62	NaN	€0	€0	1247	Right	1.0	3.0	2.0	NaN	6'3	174lbs	NaN	NaN	NaN

241 rows × 75 columns

Ahora eliminaremos el atributo club para los jugadores sin él, ya que nos reduce los problemas y hay una proporción mucho mayor de jugadores con club en comparativa con aquellos sin él.

Además, aplicamos una conversión que convertirá el valor de ciertas variables, entre ellas club, a ceros y unos, dependiendo del valor que tengan determinado. Esto nos servirá para, por ejemplo, poder tener en cuenta factores como la pierna buena del jugador.

In [8]:

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

Ahora hemos solucionado el problema que teníamos con el club, pero seguimos teniendo otros valores que o bien contienenen NaNs o bien no están en el formato adecuado. Por ello, definimos una función auxiliar que utilizaremos para convertir el valor y el sueldo de un jugador, denominada value_to_float.

In [9]:

```
def value_to_float(x):
    """
    From K and M to float.

    """
    x = str(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
```

Ahora aplicamos la función al atributo "Value".

Value

Aplicamos la función de conversión con el valor del jugador, para así poder eliminar la nomenclatura "K" y "M". Este valor posteriormente será eliminado a la hora de realizar la predicción, y se comparará con el obtenido.

In [10]:

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

Out[10]:

	Age	Overall	Potential	Value	Wage	Special	International Reputation	Weak Foot	Skill Moves	Height	Weight	LS	ST	RS	LW	LF	CF	
0	31	94	94	110500000.0	€565K	2202		5.0	4.0	4.0	5'7	159lbs	88+2	88+2	88+2	92+2	93+2	9
1	33	94	94	77000000.0	€405K	2228		5.0	4.0	5.0	6'2	183lbs	91+3	91+3	91+3	89+3	90+3	9
2	26	92	93	118500000.0	€290K	2143		5.0	5.0	5.0	5'9	150lbs	84+3	84+3	84+3	89+3	89+3	8
3	27	91	93	72000000.0	€260K	1471		4.0	3.0	1.0	6'4	168lbs	NaN	NaN	NaN	NaN	NaN	
4	27	91	92	102000000.0	€355K	2281		4.0	5.0	4.0	5'11	154lbs	82+3	82+3	82+3	87+3	87+3	87+3

Wage

Hacemos lo mismo con el atributo Wage, ya que necesita de la misma conversión, por lo que convertimos el sueldo a un valor computable.

He optado por incluirlo dentro del conjunto de datos evaluables porque muchas veces va ligado con el rendimiento del jugador. Además, para evaluar el valor de un jugador no nos basta con evaluar únicamente su valoración y sus características físicas. El objetivo es tratar de incluir todas aquellas variables que nos puedan ayudar a predecir con mayor exactitud la valoración del mismo. Hay variables como la capacidad de liderazgo que no son tangibles, por lo que necesitamos recurrir a toda la información posible.

In [11]:

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

Out[11]:

	Age	Overall	Potential	Value	Wage	Special	International Reputation	Weak Foot	Skill Moves	Height	Weight	LS	ST	RS	LW	LF	CF	
0	31	94	94	110500000.0	565000.0	2202		5.0	4.0	4.0	5'7	159lbs	88+2	88+2	88+2	92+2	93+2	93+2
1	33	94	94	77000000.0	405000.0	2228		5.0	4.0	5.0	6'2	183lbs	91+3	91+3	91+3	89+3	90+3	90+3
2	26	92	93	118500000.0	290000.0	2143		5.0	5.0	5.0	5'9	150lbs	84+3	84+3	84+3	89+3	89+3	89+3
3	27	91	93	72000000.0	260000.0	1471		4.0	3.0	1.0	6'4	168lbs	NaN	NaN	NaN	NaN	NaN	
4	27	91	92	102000000.0	355000.0	2281		4.0	5.0	4.0	5'11	154lbs	82+3	82+3	82+3	87+3	87+3	87+3

Cláusula de rescisión

También aplicamos la misma función al atributo de la cláusula de rescisión.

La cláusula de rescisión puede ser útil ya que un jugador "estrella" por norma general tendrá una cláusula astronómica, y esto nos puede ayudar a guiarlos y a determinar el tipo de jugador que estamos evaluando.

In [12]:

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

Out[12]:

	Age	Overall	Potential	Value	Wage	Special	International Reputation	Weak Foot	Skill Moves	Height	Weight	LS	ST	RS	LW	LF	CF	
0	31	94	94	110500000.0	565000.0	2202		5.0	4.0	4.0	5'7	159lbs	88+2	88+2	88+2	92+2	93+2	93+2
1	33	94	94	77000000.0	405000.0	2228		5.0	4.0	5.0	6'2	183lbs	91+3	91+3	91+3	89+3	90+3	90+3
2	26	92	93	118500000.0	290000.0	2143		5.0	5.0	5.0	5'9	150lbs	84+3	84+3	84+3	89+3	89+3	89+3
3	27	91	93	72000000.0	260000.0	1471		4.0	3.0	1.0	6'4	168lbs	NaN	NaN	NaN	NaN	NaN	
4	27	91	92	102000000.0	355000.0	2281		4.0	5.0	4.0	5'11	154lbs	82+3	82+3	82+3	87+3	87+3	87+3

Arreglo de las valoraciones

Ahora tenemos un nuevo "problema", y es que las valoraciones de los jugadores para determinadas posiciones viene dado como una suma, por lo que no puede ser computado de manera directa, por lo que creamos una función auxiliar para convertir estos valores.

Simplemente suma los dos valores y reemplaza el valor por el adecuado. Además, sustituye los valores no numéricos por ceros, para evitar futuros conflictos. También se ha testeado el sustituir estos por un valor aleatorio dentro del rango del resto, por la media +- la variación... Pero finalmente he considerado que era más adecuado sustituirlos por cero al no tener un valor determinado en el dataset original.

In [13]:

```
#Array auxiliar que contiene todas las posiciones a evaluar.  
posiciones = ["LS","ST","RS","LW","LF","CF","RF","RW","LAM","CAM","RAM","LM","LCM","CM","RCM","RM","LWB",  
  
#Convierte una suma de valores en el valor correspondiente  
def valoracionNueva(valor):  
    if (valor!=valor): #Sustituye los NaN por ceros  
        return 0  
    else:  
        arr = valor.split('+')  
        v1 = float(arr[0])  
        v2 = float(arr[1])  
        nuevoValor = v1 + v2  
    return nuevoValor  
  
for posicion in posiciones:  
    df[posicion]=df[posicion].apply(valoracionNueva)
```

Height

He considerado que la altura es un factor a tener en cuenta a la hora de determinar el valor de un jugador, por lo que he realizado una conversión a centímetros, para tratar con esta más fácilmente. Un ejemplo de esto es un delantero centro, que tiende a ser el rematador de los centros, por lo que la altura y el valor son proporcionales (en una medida muy pequeña, pero el objetivo es evaluar el mayor número de variables que afecten al valor del jugador).

Convertimos los valores de altura a centímetros, ya que los originales estaban en pies, y tenían un formato que no era apto para procesarlo directamente.

In [14]:

```
#Convierte una altura en feet a una en centímetros.  
def heightConverter(ht):  
    height = ht.split("'")  
    feet = float(height[0])  
    inches = float(height[1])  
    return int(30.48 * feet + 2.54 * inches)  
  
df=df.dropna(subset=['Height'])  
df["Height"] = df["Height"].apply(heightConverter)  
df.head()
```

In [15]:

Out[15]:

	Age	Overall	Potential	Value	Wage	Special	International Reputation	Weak Foot	Skill Moves	Height	Weight	LS	ST	RS	LW	LF	CF	RF
0	31	94	94	110500000.0	5650000.0	2202	5.0	4.0	4.0	170	159lbs	90.0	90.0	90.0	94.0	95.0	95.0	95.0
1	33	94	94	77000000.0	405000.0	2228	5.0	4.0	5.0	187	183lbs	94.0	94.0	94.0	92.0	93.0	93.0	93.0
2	26	92	93	118500000.0	290000.0	2143	5.0	5.0	5.0	175	150lbs	87.0	87.0	87.0	92.0	92.0	92.0	92.0
3	27	91	93	72000000.0	260000.0	1471	4.0	3.0	1.0	193	168lbs	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	27	91	92	102000000.0	355000.0	2281	4.0	5.0	4.0	180	154lbs	85.0	85.0	85.0	90.0	90.0	90.0	90.0



Weight

Nos "cargaremos" el molesto texto que precede al peso de los jugadores, mediante una función auxiliar que únicamente tendrá en cuenta los tres primeros caracteres del peso del jugador.

Esto nos permite tener en cuenta esta variable para nuestra predicción, ya que considero que es una variable que puede ser determinante en ciertas posiciones. Por ejemplo, por norma general los defensas centrales suelen ser corpulentos, y por ende pesan más, mientras que los jugadores de banda, tienden a ser más bajos y ágiles.

In [16]:

```
def weight(w):
    w = w[0:3]
    return int(w)
```

In [17]:

```
df["Weight"] = df["Weight"].apply(weight)
df.head()
```

Out[17]:

	Age	Overall	Potential	Value	Wage	Special	International Reputation	Weak Foot	Skill Moves	Height	Weight	LS	ST	RS	LW	LF	CF	RF
0	31	94	94	110500000.0	565000.0	2202	5.0	4.0	4.0	170	159	90.0	90.0	90.0	94.0	95.0	95.0	95.0
1	33	94	94	770000000.0	405000.0	2228	5.0	4.0	5.0	187	183	94.0	94.0	94.0	92.0	93.0	93.0	93.0
2	26	92	93	118500000.0	290000.0	2143	5.0	5.0	5.0	175	150	87.0	87.0	87.0	92.0	92.0	92.0	92.0
3	27	91	93	72000000.0	260000.0	1471	4.0	3.0	1.0	193	168	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	27	91	92	102000000.0	355000.0	2281	4.0	5.0	4.0	180	154	85.0	85.0	85.0	90.0	90.0	90.0	90.0



Predicción final

Eliminaremos la columna relativa al valor, ya que nuestro objetivo es acercarnos lo máximo posible a esta mediante la predicción del modelo de regresión lineal.

Después actualizaremos las variables reg ajustándola al modelo y finalmente compararemos nuestra score predicha con la real, y el valor que obtengamos cuanto mayor sea, más precisa habrá sido. El escenario utópico sería que este fuera prácticamente uno.

In [18]:

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

In [19]:

```
X_train, X_test, y_train, y_test = train_test_split(df, val, test_size=0.25, random_state=48)
```

In [20]:

```
len(X_train)
```

Out[20]:

13438

Entrenamos el modelo de regresión lineal.

In [21]:

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

Obtenemos \$R^2\$ como regresión, implementada desde [scikit-learn](#).

In [22]:

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

Out[22]:

```
array([1160394.00146346, 1131153.99474172, 293852.47384825, ...,
       1061513.9381384, -879908.22095644, 256153.8725264])
```

Comparativa y puntuación final.

In [23]:

```
preds[0]
```

Out[23]:

1160394.0014634617

In [24]:

```
y_test
```

Out[24]:

```
10172      775000.0
7517       850000.0
13688      375000.0
11309      600000.0
2          118500000.0
...
17371      80000.0
16284      140000.0
6487       1100000.0
15415      20000.0
14561      400000.0
Name: Value, Length: 4480, dtype: float64
```

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

In [25]:

Out[25]:

```
0.96541217513217
```

Finalmente, obtenemos una puntuación de 0.965, esto quiere decir que nuestra predicción es un 96.5% precisa, por lo que es bastante cercano al objetivo.

Cabe mencionar que esta predicción varía ligeramente en función del valor de la variable \$randomState\$.