FIFA pràctica

Volem predir el "value" dels jugadors utilitzant la informació del jugador.

Enllaç al github: https://github.com/xescnova/FIFA-Machine-Learning/tree/main/Practica%203%20IA (https://github.com/xescnova/FIFA-Machine-Learning/tree/main/Practica%203%20IA)

Importar llibreries

Importam les llibreries necesaries

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')
```

Llegim les dades

Per llegir-les utilitzam `pandas` i la utilitat de la llibreria `os` .

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

Anàlisi de dades

Aquestes són les nostres dades i hem de fer un anàlisi

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

In [4]:
```
df.head()
```

Out[4]:

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

Eliminació de columnes

A aquesta part eliminaré les columnes que no aporten informació a la resolució del problema : ID,Nationality,Photo,Preferred Foot,Name, Flag , Club Logo , Real Face , Joined , Jersey Number , Loaned From , Contract Valid Until , Weight , Height.

Totes aquestes columnes excepte "Joined" i "Loaned From" no són necesaries i es trivial. Per altra banda, tampoc ens interesa conèixer l'any en que va entrar la persona al club perqué és independent del seu "Value". Si és un jugador cedit tampoc ens interesa, perque hi ha molt pocs que ho són i el "Value" del jugador no dependrà d'ell.

Weight i Height són atributs que es podria realitzar un tractament. En el cas de Height seria canviar el x'x per x.x perque el programa ho pugui computar i en el cas de Weight seria llevar la part de Xlbs i pasar-ho a kg si es prefereix però he decidit no tractar-les perque són quasi irrelevants per calcular el "Value" del jugador ja que hi ha altres atributs que són més importants.

Work Rate i Body Type es podria fer un tractament similar al que realitzaré per "Club" que consisteix en agafar totes les files diferents i transformar-les en columnes i si el jugador té aquest atribut aleshores asignar-li un 1 i un 0 si no el té. Per Work Rate primer s'hauria de fer un tractament previ perque té 2 valors , un per la primera part i un altre per la segona, aleshores podriem agafar el de la primera part i l'altre eliminar-lo i a continuació realitzar el tractament anterior.Però per calcular el "Value" hi ha atributs més importants i aquests els podem eliminar.

Preferred foot: atribut amb molt poc impacte sobre "Value" que podem eliminar també.

In [5]:
```
df.drop(['ID','Preferred Foot','Name','Nationality','Body Type','Weight','Height'
df
```

Out[5]:

	Unnamed: 0	Age	Overall	Potential	Club	Value	Wage	Special	International Reputation	Wea Foc
0	0	31	94	94	FC Barcelona	€110.5M	€565K	2202	5.0	4.
1	1	33	94	94	Juventus	€77M	€405K	2228	5.0	4.
2	2	26	92	93	Paris Saint-Germain	€118.5M	€290K	2143	5.0	5.
3	3	27	91	93	Manchester United	€72M	€260K	1471	4.0	3.
4	4	27	91	92	Manchester City	€102M	€355K	2281	4.0	5.
...	...	...	...	...	...	...	...	...	...	.
18202	18202	19	47	65	Crewe Alexandra	€60K	€1K	1307	1.0	2.
18203	18203	19	47	63	Trelleborgs FF	€60K	€1K	1098	1.0	2.
18204	18204	16	47	67	Cambridge United	€60K	€1K	1189	1.0	3.
18205	18205	17	47	66	Tranmere Rovers	€60K	€1K	1228	1.0	3.
18206	18206	16	46	66	Tranmere Rovers	€60K	€1K	1321	1.0	3.

18207 rows × 73 columns

Arreglar NaNs

Hi ha valors que són NaN,és a dir "not a number" ,valors nulls. Aquí hi ha les columnes que tenen algun NaN

In [6]:
```
df.columns[df.isna().any()].tolist()
```

Out[6]:
```
['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',
 'Release Clause']
```

Amb la següent instrucció podem veure la informació del nostre dataframe, les columnes Non-Null indiquen el total de jugadors que tenen un valor en aquell atribut.

El total de jugadors són 18207 , per tant , haurem de tractar totes les columnes amb NaN.

Per altra banda, totes les columnes amb Dtype = object hauran de ser convertides a int per tal de poder ser interpretades.

In [7]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 18207 entries, 0 to 18206
Data columns (total 73 columns):
 #   Column                    Non-Null Count  Dtype  
---  ------                    --------------  -----  
 0   Unnamed: 0                18207 non-null  int64  
 1   Age                       18207 non-null  int64  
 2   Overall                   18207 non-null  int64  
 3   Potential                 18207 non-null  int64  
 4   Club                      17966 non-null  object 
 5   Value                     18207 non-null  object 
 6   Wage                      18207 non-null  object 
 7   Special                   18207 non-null  int64  
 8   International Reputation  18159 non-null  float64
 9   Weak Foot                 18159 non-null  float64
 10  Skill Moves               18159 non-null  float64
 11  Position                  18147 non-null  object 
 12  LS                        16122 non-null  object 
 13  ST                        16122 non-null  object 
 14  RS                        16122 non-null  object 
 15  LW                        16122 non-null  object 
 16  LF                        16122 non-null  object 
 17  CF                        16122 non-null  object 
 18  RF                        16122 non-null  object 
 19  RW                        16122 non-null  object 
 20  LAM                       16122 non-null  object 
 21  CAM                       16122 non-null  object 
 22  RAM                       16122 non-null  object 
 23  LM                        16122 non-null  object 
 24  LCM                       16122 non-null  object 
 25  CM                        16122 non-null  object 
 26  RCM                       16122 non-null  object 
 27  RM                        16122 non-null  object 
 28  LWB                       16122 non-null  object 
 29  LDM                       16122 non-null  object 
 30  CDM                       16122 non-null  object 
 31  RDM                       16122 non-null  object 
 32  RWB                       16122 non-null  object 
 33  LB                        16122 non-null  object 
 34  LCB                       16122 non-null  object 
 35  CB                        16122 non-null  object 
 36  RCB                       16122 non-null  object 
 37  RB                        16122 non-null  object 
 38  Crossing                  18159 non-null  float64
 39  Finishing                 18159 non-null  float64
 40  HeadingAccuracy           18159 non-null  float64
 41  ShortPassing              18159 non-null  float64
 42  Volleys                   18159 non-null  float64
 43  Dribbling                 18159 non-null  float64
 44  Curve                     18159 non-null  float64
 45  FKAccuracy                18159 non-null  float64
 46  LongPassing               18159 non-null  float64
 47  BallControl               18159 non-null  float64
 48  Acceleration              18159 non-null  float64
 49  SprintSpeed               18159 non-null  float64
```

```
 50  Agility                 18159 non-null  float64
 51  Reactions               18159 non-null  float64
 52  Balance                 18159 non-null  float64
 53  ShotPower               18159 non-null  float64
 54  Jumping                 18159 non-null  float64
 55  Stamina                 18159 non-null  float64
 56  Strength                18159 non-null  float64
 57  LongShots               18159 non-null  float64
 58  Aggression              18159 non-null  float64
 59  Interceptions           18159 non-null  float64
 60  Positioning             18159 non-null  float64
 61  Vision                  18159 non-null  float64
 62  Penalties               18159 non-null  float64
 63  Composure               18159 non-null  float64
 64  Marking                 18159 non-null  float64
 65  StandingTackle          18159 non-null  float64
 66  SlidingTackle           18159 non-null  float64
 67  GKDiving                18159 non-null  float64
 68  GKHandling              18159 non-null  float64
 69  GKKicking               18159 non-null  float64
 70  GKPositioning           18159 non-null  float64
 71  GKReflexes              18159 non-null  float64
 72  Release Clause          16643 non-null  object
dtypes: float64(37), int64(5), object(31)
memory usage: 10.1+ MB
```

Ens podem fixar que hi ha columnes que tenen el mateix valor però que no és el total, aquest valor és 18159. És a dir,hi ha 48 jugadors dels quals no se sap una serie d'atributs, aquests es poden eliminar ja que 48 jugadors sobre 18207 no és un canvi gran.

Procedim a eliminar-los (podem executar una instrucció que elimini els jugadors que no tenen Marking per exemple ja que seran els mateixos 48 que no tenen l'altra serie de atributs.

In [8]:
```
df = df[df['Marking'].notna()]
df
```

Out[8]:

	Unnamed: 0	Age	Overall	Potential	Club	Value	Wage	Special	International Reputation	Wea Foc
0	0	31	94	94	FC Barcelona	€110.5M	€565K	2202	5.0	4.
1	1	33	94	94	Juventus	€77M	€405K	2228	5.0	4.
2	2	26	92	93	Paris Saint-Germain	€118.5M	€290K	2143	5.0	5.
3	3	27	91	93	Manchester United	€72M	€260K	1471	4.0	3.
4	4	27	91	92	Manchester City	€102M	€355K	2281	4.0	5.
...	...	...	...	...	...	...	...	...	...	.
18202	18202	19	47	65	Crewe Alexandra	€60K	€1K	1307	1.0	2.
18203	18203	19	47	63	Trelleborgs FF	€60K	€1K	1098	1.0	2.
18204	18204	16	47	67	Cambridge United	€60K	€1K	1189	1.0	3.
18205	18205	17	47	66	Tranmere Rovers	€60K	€1K	1228	1.0	3.
18206	18206	16	46	66	Tranmere Rovers	€60K	€1K	1321	1.0	3.

18159 rows × 73 columns

Eliminam els jugadors que no tenen Club , ja que són menys de 100 i no afecten als nostres càlculs. Es podria fer de una altra manera canviant el NaN per *Sense Club* però amb la quantitat que hi ha he preferit eliminar-los.

In [9]:
```
df = df[df['Club'].notna()]
df
```

Out[9]:

	Unnamed: 0	Age	Overall	Potential	Club	Value	Wage	Special	International Reputation	Wea Foc
0	0	31	94	94	FC Barcelona	€110.5M	€565K	2202	5.0	4.
1	1	33	94	94	Juventus	€77M	€405K	2228	5.0	4.
2	2	26	92	93	Paris Saint-Germain	€118.5M	€290K	2143	5.0	5.
3	3	27	91	93	Manchester United	€72M	€260K	1471	4.0	3.
4	4	27	91	92	Manchester City	€102M	€355K	2281	4.0	5.
...	...	...	...	...	...	...	...	...	...	.
18202	18202	19	47	65	Crewe Alexandra	€60K	€1K	1307	1.0	2.
18203	18203	19	47	63	Trelleborgs FF	€60K	€1K	1098	1.0	2.
18204	18204	16	47	67	Cambridge United	€60K	€1K	1189	1.0	3.
18205	18205	17	47	66	Tranmere Rovers	€60K	€1K	1228	1.0	3.
18206	18206	16	46	66	Tranmere Rovers	€60K	€1K	1321	1.0	3.

17918 rows × 73 columns

Hem de modificar els jugadors que tenen "Release Clause" = NaN per un valor igual a 0 ja que si el valor de la clàusula és NaN, significa que no en té i que és igual a 0 i el jugador es pot anar gratis.

In [10]:
```
df['Release Clause'] = df['Release Clause'].fillna('€0K')
df
```

Out[10]:

	Unnamed: 0	Age	Overall	Potential	Club	Value	Wage	Special	International Reputation	Wea Foc
0	0	31	94	94	FC Barcelona	€110.5M	€565K	2202	5.0	4.
1	1	33	94	94	Juventus	€77M	€405K	2228	5.0	4.
2	2	26	92	93	Paris Saint-Germain	€118.5M	€290K	2143	5.0	5.
3	3	27	91	93	Manchester United	€72M	€260K	1471	4.0	3.
4	4	27	91	92	Manchester City	€102M	€355K	2281	4.0	5.
...	...	...	...	...	...	...	...	...	...	.
18202	18202	19	47	65	Crewe Alexandra	€60K	€1K	1307	1.0	2.
18203	18203	19	47	63	Trelleborgs FF	€60K	€1K	1098	1.0	2.
18204	18204	16	47	67	Cambridge United	€60K	€1K	1189	1.0	3.
18205	18205	17	47	66	Tranmere Rovers	€60K	€1K	1228	1.0	3.
18206	18206	16	46	66	Tranmere Rovers	€60K	€1K	1321	1.0	3.

17918 rows × 73 columns

Hi ha un problema amb els porters. Els porters no tenen puntuació a les columnes "LS","RT","ST",etc. i és lògic perquè un porter només juga de porter.

Per això totes les columnes que tenguin NaN al les columnes de "LS","RT","ST",etc. seran emplenades amb 0.

In [11]:
```
df['LS'] = df['LS'].fillna('0')
df['ST'] = df['ST'].fillna('0')
df['RS'] = df['RS'].fillna('0')
df['LW'] = df['LW'].fillna('0')
df['LF'] = df['LF'].fillna('0')
df['CF'] = df['CF'].fillna('0')
df['RF'] = df['RF'].fillna('0')
df['RW'] = df['RW'].fillna('0')
df['LAM'] = df['LAM'].fillna('0')
df['CAM'] = df['CAM'].fillna('0')
df['RAM'] = df['RAM'].fillna('0')
df['LM'] = df['LM'].fillna('0')
df['LCM'] = df['LCM'].fillna('0')
df['CM'] = df['CM'].fillna('0')
df['RCM'] = df['RCM'].fillna('0')
df['RM'] = df['RM'].fillna('0')
df['LWB'] = df['LWB'].fillna('0')
df['LDM'] = df['LDM'].fillna('0')
df['CDM'] = df['CDM'].fillna('0')
df['RDM'] = df['RDM'].fillna('0')
df['RWB'] = df['RWB'].fillna('0')
df['LB'] = df['LB'].fillna('0')
df['LCB'] = df['LCB'].fillna('0')
df['CB'] = df['CB'].fillna('0')
df['RCB'] = df['RCB'].fillna('0')
df['RB'] = df['RB'].fillna('0')
```

Transformació de columnes a int

"Value" , "Wage" y "Release Clause" son columnes amb expressions que el programa no sap interpretar per això les hem de transformar

Utilitzarem el següent mètode:

In [12]:
```
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
```

In [13]:
```
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)
```

Hem de modificar els valors de "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" perque duen un + i el programa no els sap interpretar.

Hem de crear el següent mètode que suma els dos valors. Exemple -> 88+2 = 90

In [14]:
```
#funció que suma el str per exemple: 88+2 = 90 i el transforma a int, sino és str
def skillSuma(valor):
    if type(valor) == str:
        s1 = valor[0:2]
        s2 = valor[-1]
        valor = int(s1) +int(s2)
        return valor
    else:
        return 0
```

In [15]:
```
    skillColumnes = ['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']


    for columna in skillColumnes:
        df[columna] = df[columna].apply(skillSuma)
```

In [16]:
```
df.head()
```

Out[16]:

ernational eputation	Weak Foot	Skill Moves	Position	LS	ST	RS	LW	LF	CF	RF	RW	LAM	CAM	RAM	LM	LCM
5.0	4.0	4.0	RF	90	90	90	94	95	95	95	94	95	95	95	93	86
5.0	4.0	5.0	ST	94	94	94	92	93	93	93	92	91	91	91	91	84
5.0	5.0	5.0	LW	87	87	87	92	92	92	92	92	92	92	92	91	84
4.0	3.0	1.0	GK	0	0	0	0	0	0	0	0	0	0	0	0	0
4.0	5.0	4.0	RCM	85	85	85	90	90	90	90	90	91	91	91	91	90

In [17]:
```
clb = df.pop("Club")

df = pd.concat([df.reset_index(drop=True), pd.get_dummies(clb, prefix='clb').rese
```

In [18]:
```
df['clb_1. FC Heidenheim 1846'].value_counts()
```

Out[18]:
```
0    17890
1       28
Name: clb_1. FC Heidenheim 1846, dtype: int64
```

In [19]:
```
pos = df.pop("Position")

df = pd.concat([df.reset_index(drop=True), pd.get_dummies(pos, prefix='pos').rese
```

In [20]:
```
df.head()
```

Out[20]:

(s	clb_AFC Wimbledon	clb_AIK	clb_AJ Auxerre	clb_AS Béziers	clb_AS Monaco	clb_AS Nancy Lorraine	clb_AS Saint-Étienne	clb_AZ Alkmaar	clb_Aalborg BK	clb_Aar
)	0	0	0	0	0	0	0	0	0	
)	0	0	0	0	0	0	0	0	0	
)	0	0	0	0	0	0	0	0	0	
)	0	0	0	0	0	0	0	0	0	
)	0	0	0	0	0	0	0	0	0	

In [21]:
```
df['pos_RW'].value_counts()
```

Out[21]:
```
0    17553
1      365
Name: pos_RW, dtype: int64
```

In [22]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 17918 entries, 0 to 17917
Columns: 749 entries, Unnamed: 0 to pos_ST
dtypes: float64(40), int64(31), uint8(678)
memory usage: 21.3 MB
```

Predicció

Finalment volem predir el **value** .Per això ho treim del dataset amb pop("Value") i aplicam el model.

In [23]:
```
val = df.pop("Value")
df
```

Out[23]:

	Unnamed: 0	Age	Overall	Potential	Wage	Special	International Reputation	Weak Foot	Skill Moves	LS	ST
0	0	31	94	94	565000.0	2202	5.0	4.0	4.0	90	90
1	1	33	94	94	405000.0	2228	5.0	4.0	5.0	94	94
2	2	26	92	93	290000.0	2143	5.0	5.0	5.0	87	87
3	3	27	91	93	260000.0	1471	4.0	3.0	1.0	0	0
4	4	27	91	92	355000.0	2281	4.0	5.0	4.0	85	85
...	...	...	...	...	...	...	...	...	...	...	...
17913	18202	19	47	65	1000.0	1307	1.0	2.0	2.0	44	44
17914	18203	19	47	63	1000.0	1098	1.0	2.0	2.0	47	47
17915	18204	16	47	67	1000.0	1189	1.0	3.0	2.0	47	47
17916	18205	17	47	66	1000.0	1228	1.0	3.0	2.0	49	49
17917	18206	16	46	66	1000.0	1321	1.0	3.0	2.0	45	45

17918 rows × 748 columns

Separam el data set en dos conjunts, un per fer el test i l'altre per entrenar amb el model.

In [24]:
```
X_train, X_test, y_train, y_test = train_test_split(df, val, test_size=0.33, rand
```

In [25]:
```
len(X_train)
```

Out[25]:
```
12005
```

Entrenam un LinearRegression model.

In [26]:
```
reg = linear_model.LinearRegression().fit(X_train, y_train)
```

Finalment obtenim una métrica R^2 per a la regressío, utilitz la implementació de sickit-learn (https://scikit-learn.org/stable/modules/generated/sklearn.metrics.r2_score.html).

In [27]:
```
preds = reg.predict(X_test)
```

In [28]:
```
preds[0]
```

Out[28]:
```
-447513.5713709798
```

In [29]:
```
y_test[0]
```

Out[29]:
```
110500000.0
```

In [30]:
```
r2_score(preds, y_test)
```

Out[30]:
```
0.9672990111410212
```