PracticaFifa

January 28, 2021

1 Practica Fifa

1.1 Intelůligència Artificial. Grup 1 UIB

Lluc Valdés Carrasco

Repository Github

Importam les llibreries necesaries

[1777]:
```
import os
from IPython.display import set_matplotlib_formats
set_matplotlib_formats('pdf', 'svg')
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
from sklearn import preprocessing

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

Llegim les dades de el .csv

[1778]:
```
df = pd.read_csv(os.path.join("..", "in", "/content/fifa.csv"))
```

Anam a veure com son les dades.

[1779]:
```
df.head()
```

[1779]:
```
   Unnamed: 0      ID  ... GKReflexes  Release Clause
0           0  158023  ...        8.0          226.5M
1           1   20801  ...       11.0          127.1M
2           2  190871  ...       11.0          228.1M
3           3  193080  ...       94.0          138.6M
4           4  192985  ...       13.0          196.4M

[5 rows x 89 columns]
```

Aquí podem veure que hi ha moltes columnes que no ens donaran cap informació com poden ser les tres primeres, la de Photo, la de Flag o la de Club Logo així que les llevarem.

[1780]:
```
df = df.drop(columns =  ["Unnamed: 0","ID","Name","Photo","Flag","Club␣
↪Logo"],axis=1)
```

No seran les úniques columnes que acabarem llevant, però eren les que no tenien res a veure amb el valor del jugador.

És necessari canviar totes les columnes com value o wage, ja que ara mateix són strings i volem que siguin números.

Això o farem amb la funció feta pel professor.

[1781]:
```
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
```

[1782]:
```
df["Value"] = df["Value"].apply(value_to_float)
df["Wage"] = df["Wage"].apply(value_to_float)
df["Release Clause"].fillna("0",inplace = True)
df["Release Clause"] = df["Release Clause"].apply(value_to_float)
```

També hi ha un problema amb la columna Loaned From, ja que té molts de NaN (Si un jugador no està cedit surt NaN).

Podríem fer dues coses, eliminar la columna. O si és probable que hi hagi una relació amb el preu del jugador guardar aquesta informació d'un altra manera.

Jo he optat per la primera solució.

[1783]:
```
df = df.drop(columns =  ["Loaned From"],axis=1)
```

Tant la columna club com a nacionalitat s'han de modificar per poder emprar-les, i que la informació que donin sigui útil. Per això emprarem onehot encoding.

[1784]:
```
clb = df.pop("Club")
df = pd.concat([df.reset_index(drop=True), pd.get_dummies(clb, prefix='clb').
 ↪reset_index(drop=True)], axis=1, sort=False)
nat = df.pop("Nationality")
df = pd.concat([df.reset_index(drop=True), pd.get_dummies(nat, prefix='nat').
 ↪reset_index(drop=True)], axis=1, sort=False)
df.head()
```

[1784]:
```
   Age  Overall  Potential  ...  nat_Wales  nat_Zambia  nat_Zimbabwe
0   31       94         94  ...          0           0             0
1   33       94         94  ...          0           0             0
2   26       92         93  ...          0           0             0
3   27       91         93  ...          0           0             0
4   27       91         92  ...          0           0             0

[5 rows x 895 columns]
```

[1759]:
```
df.corr()
```

[1759]:
```
                     Age   Overall  ...  nat_Zambia  nat_Zimbabwe
Age             1.000000  0.452350  ...   -0.013810      0.009868
Overall         0.452350  1.000000  ...   -0.003272      0.013660
Potential      -0.253312  0.660939  ...    0.008953      0.005028
Value           0.075022  0.627085  ...   -0.005093      0.001768
Wage            0.141145  0.571926  ...   -0.006355     -0.003506
...                  ...       ...  ...         ...           ...
nat_Uzbekistan  0.009826  0.001914  ...   -0.000233     -0.000280
nat_Venezuela  -0.010526  0.009060  ...   -0.001352     -0.001625
nat_Wales      -0.006978 -0.025667  ...   -0.001879     -0.002258
nat_Zambia     -0.013810 -0.003272  ...    1.000000     -0.000594
nat_Zimbabwe    0.009868  0.013660  ...   -0.000594      1.000000

[860 rows x 860 columns]
```

2 Anem a mirar quines columnes queden que necessitin algun canvi:

PreferedFoot: Hauria de passar de ser un string a una altra cosa, es podria emprar one-hot.

Work Rate: Fer dues columnes una per atac que es el primer valor, i un altre per defensa que és el segon valor, on Low = 1, Medium = 2, High = 3.

Body Type: Eliminarla o fer one-hot.

Real Face: Canviar els yes per 1 i el no per 0, és probable que els jugadors que tenen photo sigui perquè són coneguts.

Jersey Number: Abans he decidit deixar-ho perquè alomillor els nombres més baixos tenien valors més alts. Ja que solen ser els jugadors titulars, però com es pot veure dalt la correlació és de 0,09 així que no és molt útil. Així que ho llevaré.

Joined: Segurament no ens sigui molt útil (Messi va entrar molt prest i és molt car i en Ronaldo fa poc i també és molt car) així que ho llevarem.

Contract Valid Until: Tampoc ens afectarà molt, ja que no hi ha molts d'anys de diferència als contractes així que tambe ho llevarem.

Height i Weight: Els modificarem perquè no siguin strings

Position: Podem llevar aquesta columna si arreglam les de LS, ST, etc. Per la qual cosa hauríem de fer la mitja i posar-la als porters. Així que les llevarem i farem one hot de position

Prefered Foot

[1760]:
```
pf = df.pop("Preferred Foot")
```

```
df = pd.concat([df.reset_index(drop=True), pd.get_dummies(pf, prefix='pf').
↪reset_index(drop=True)], axis=1, sort=False)
```

Work Rate

[1761]:
```
def esforçAtac(x):

  x = x.split('/ ')
  if(x[0] == "Low"):
    return 1
  if(x[0] == "Medium"):
    return 2
  if(x[0] == "High"):
    return 3
```

[1762]:
```
def esforçDefensa(x):
  x = x.split('/ ')
  if(x[1] == "Low"):
    return 1
  if(x[1] == "Medium"):
    return 2
  if(x[1] == "High"):
    return 3
```

[1763]:
```
df["Work Rate"].fillna("Medium/ Medium",inplace = True)
EA = df.pop("Work Rate")
ED = EA.copy()
```

[1764]:
```
df["EsforçAtac"] = EA.apply(esforçAtac)
df["EsforçDefensa"] = ED.apply(esforçDefensa)
```

Body Type

[1765]:
```
bt = df.pop("Body Type")
df = pd.concat([df.reset_index(drop=True), pd.get_dummies(bt, prefix='bt').
↪reset_index(drop=True)], axis=1, sort=False)
```

Real Face

[1766]:
```
df["Real Face"].unique()
```

[1766]:
```
array(['Yes', 'No', nan], dtype=object)
```

[1767]:
```
df["Real Face"].fillna("No",inplace = True)
df["Real Face"].replace({"Yes": 1, "No": 0}, inplace=True)
```

Jersey Number, Joined, Release Clause

[1768]:
```
df = df.drop(columns =  ["Jersey Number", "Joined", "Contract Valid␣
↪Until"],axis=1)
```

Heigth i Weight

[1769]:
```
def convertHeight(x):
  x.split("'")
  y = float(x[0]+"."+x[2])*30.48
```

```
  return y
```

[1770]:
```
def convertWeight(x):
  x = x.replace('lbs','')
  y = float(x)/2.205
  return y
```

[1771]:
```
df.dropna(subset=['Height'], inplace = True)
df["Height"] = df["Height"].apply(convertHeight)

df.dropna(subset=['Weight'], inplace = True)
df["Weight"] = df["Weight"].apply(convertWeight)
```

Position

[1772]:
```
df = df.drop(columns = ␣
 ↪["LS","ST","RS","LW","LF","CF","RF","RW","LAM","CAM","RAM","LM","LCM","CM","RCM","RM","LWB"
p = df.pop("Position")
df = pd.concat([df.reset_index(drop=True), pd.get_dummies(p, prefix='p').
 ↪reset_index(drop=True)], axis=1, sort=False)
df
```

[1772]:

	Age	Overall	Potential	Value	...	p_RS	p_RW	p_RWB	p_ST
0	31	94	94	110500000.0	...	0	0	0	0
1	33	94	94	77000000.0	...	0	0	0	1
2	26	92	93	118500000.0	...	0	0	0	0
3	27	91	93	72000000.0	...	0	0	0	0
4	27	91	92	102000000.0	...	0	0	0	0
...	...	...	...	...	...	...	...	...	...
18154	19	47	65	60000.0	...	0	0	0	0
18155	19	47	63	60000.0	...	0	0	0	1
18156	16	47	67	60000.0	...	0	0	0	1
18157	17	47	66	60000.0	...	0	1	0	0
18158	16	46	66	60000.0	...	0	0	0	0

[18159 rows x 903 columns]

3 Predicció

Normalització

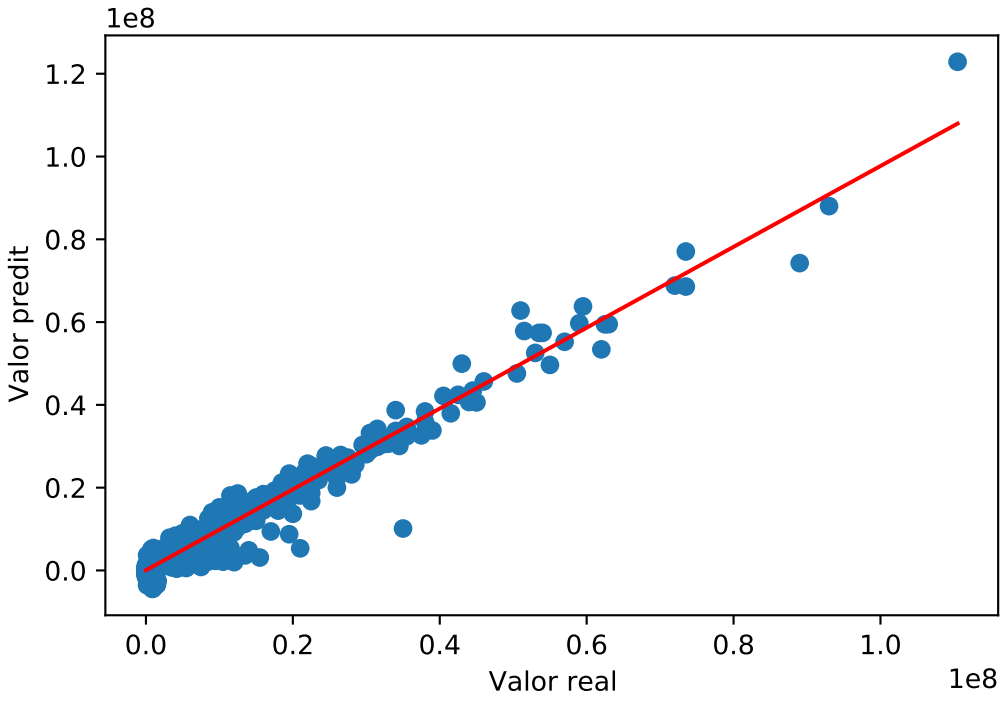
[1773]:
```
val = df.pop("Value")
#Es podria fer cross-validation pero el resultat es suficientment bo
X_train, X_test, y_train, y_test = train_test_split(df, val, test_size=0.33,␣
 ↪random_state=42)
reg = linear_model.LinearRegression().fit(X_train, y_train)
preds = reg.predict(X_test)
```

[1774]:
```
r2_score(preds, y_test)
```

[1774]:
```
0.9650461946761562
```

Això vol dir que aquesta predicció és bona, ja que com més s'atraca a 1 més petit és l'error, sent 1 una predicció sense error.

[1775]:
```
plt.scatter(y_test,preds)
plt.xlabel('Valor real')
plt.ylabel('Valor predit')
plt.plot(np.unique(y_test),np.poly1d(np.polyfit(y_test, preds, 1))(np.
 ↪unique(y_test)),color = 'red')
plt.show()
```



Podem veure com en general les prediccions tenen un marge d'error bastant petit, a part d'algun outlier com pot ser en Neymar que és el punt de més dalt a la dreta.

[1776]:
```
%%capture
!wget -nc https://raw.githubusercontent.com/brpy/colab-pdf/master/colab_pdf.py
from colab_pdf import colab_pdf
colab_pdf('PracticaFifa.ipynb')
```