In [2]:

```
import numpy as np
import pandas as pd
from pandas import read_csv
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
from matplotlib.pyplot import figure
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import r2_score
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import MinMaxScaler
from sklearn.ensemble import RandomForestRegressor
import seaborn as sns
import math
import statistics as stat
from sklearn.neural_network import MLPRegressor

import pymongo
from bson.son import SON
```

NIVEL 1 Y 2

- Ejercicios:

Crea una base de datos NoSQL utilizando MongoDB. Añádele algunos datos de ejemplo que te permitan comprobar que eres capaz de procesar la información de manera básica.

Conecta la base de datos NoSQL **a Python** utilizando por ejemplo pymongo.

Carga algunas consultas sencillas a un **Pandas Dataframe**.

He creado una base de datos en MongoDB a la que he llamado Sprint14, a modo de prueba de conocimeinto y manejo de Mongo. Las tres tablas no tienen relación entre ellas.

En la base de datos hay tres colecciones: *Estrellas, venta y fútbol*.

Las colecciones *estrellas* y *venta* han sido hechas por consola de mongodb, mientras que la de *fútbol* ha sido creada importando un CSV mediante mongoinmport.

Procederemos a conectar con la base de datos, explorar las colecciones, a la vez que las exportamos en Datasets.

In [3]:

```
myclient = pymongo.MongoClient("mongodb://localhost:27017/")# accedo a las bases de datso
```

- Miramos las bases de datos que hay y nos conectamos a *Sprint14*

In [4]:

```
print(myclient.list_database_names())# miramos las bases de datos que hay
```

```
['admin', 'config', 'local', 'nueva', 'sprint14']
```

In [5]:

```
db = myclient["sprint14"]# nos conectamos a Sprint14
```

- **Miramos las colecciones**:

In [6]:

```
print(db.list_collection_names())
```

```
['estrellas', 'venta', 'futbol']
```

In [7]:

```
cursor= db.estrellas.find({})
for i in cursor:
    print(i)
```

```
{'_id': ObjectId('62c4715e3265cbfc573d3490'), 'estrella': 'Betelgeuse', 'Radio': 887.0, 'color': 'roja', 'tip
o': 'supergigante'}
{'_id': ObjectId('62c4715e3265cbfc573d3491'), 'estrella': 'Vega', 'Radio': 2.2, 'color': 'blanca', 'tipo': 'gig
ante blanca'}
{'_id': ObjectId('62c4715e3265cbfc573d3492'), 'estrella': 'Sagitario A', 'Radio': 17250.0, 'tipo': 'agujero neg
ro supermasivo'}
{'_id': ObjectId('62c4715e3265cbfc573d3493'), 'estrella': 'Aldebarán', 'Radio': 44.2, 'color': 'naranja', 'tip
o': 'Gigante Naranja'}
```

In [8]:

```
cursor= db.venta.find({}).limit(5)
for i in cursor:
    print(i)
```

```
{'_id': ObjectId('62c4171656aaf51e8a250e47'), 'PEDIDO_IDpedido': 2, 'TRABAJADOR_Idtrabajador': 10, 'cantidad':
1, 'precio': 20, 'PRODUCTO_Idproducto': 9}
{'_id': ObjectId('62c4171656aaf51e8a250e48'), 'PEDIDO_IDpedido': 3, 'TRABAJADOR_Idtrabajador': 3, 'cantidad':
1, 'precio': 68, 'PRODUCTO_Idproducto': 1}
{'_id': ObjectId('62c4171656aaf51e8a250e49'), 'PEDIDO_IDpedido': 4, 'TRABAJADOR_Idtrabajador': 7, 'cantidad':
2, 'precio': 130, 'PRODUCTO_Idproducto': 1}
{'_id': ObjectId('62c4171656aaf51e8a250e4a'), 'PEDIDO_IDpedido': 1, 'TRABAJADOR_Idtrabajador': 2, 'cantidad':
1, 'precio': 72, 'PRODUCTO_Idproducto': 1}
{'_id': ObjectId('62c4171656aaf51e8a250e4b'), 'PEDIDO_IDpedido': 5, 'TRABAJADOR_Idtrabajador': 10, 'cantidad':
1, 'precio': 40, 'PRODUCTO_Idproducto': 2}
```

In [9]:

```
cursor= db.futbol.find({})
# al ser una colección muy grande directamente la pasamos a DF.
df=pd.DataFrame(list(cursor))
df.head()
```

Out[9]:

	_id	id	player_fifa_api_id	player_api_id	date	overall_rating	potential	preferred_foot	attacking_work
0	62c46095f90303fa262c601e	131504	131505	186839	119435	2012-08-31 00:00:00	64.0	71.0	right	me
1	62c46095f90303fa262c601f	66212	66213	192451	166963	2009-08-30 00:00:00	37.0	42.0	right	me
2	62c46095f90303fa262c6020	51377	51378	201991	411616	2014-10-10 00:00:00	69.0	73.0	right	me
3	62c46095f90303fa262c6021	131774	131775	172743	38907	2014-05-30 00:00:00	69.0	73.0	right	
4	62c46095f90303fa262c6022	38881	38882	146121	41294	2014-01-17 00:00:00	73.0	73.0	right	me

5 rows × 44 columns

In [10]:

```
df.shape# miramos las dimensiones
```

Out[10]:

```
(200, 44)
```

- De esta última tabla buscamos aquellos jugadores que son zurdos.

In [12]:

```
cursor= db.futbol.find({"preferred_foot": "left"})
# al ser una colección muy grande directamente la pasamos a DF.
df=pd.DataFrame(list(cursor))
df
```

Out[12]:

	_id	id	player_fifa_api_id	player_api_id	date	overall_rating	potential	preferred_foot	attacking_wor
0	62c46095f90303fa262c6025	130313	130314	189403	139673	2014-11-14 00:00:00	71.0	75.0	left	
1	62c46095f90303fa262c6036	52576	52577	183427	78324	2014-02-07 00:00:00	74.0	79.0	left	m
2	62c46095f90303fa262c6038	108420	108421	190696	157376	2011-08-30 00:00:00	63.0	68.0	left	m
3	62c46095f90303fa262c603c	165499	165500	51100	25594	2016-01-07 00:00:00	78.0	78.0	left	
4	62c46095f90303fa262c603d	56914	56915	45601	30679	2009-08-30 00:00:00	82.0	86.0	left	m
5	62c46095f90303fa262c603e	128948	128949	162240	41008	2016-02-04 00:00:00	80.0	80.0	left	m
6	62c46095f90303fa262c603f	83406	83407	165190	22929	2012-08-31 00:00:00	76.0	82.0	left	m
7	62c46095f90303fa262c6041	132793	132794	7738	46343	2011-02-22 00:00:00	64.0	71.0	left	m
8	62c46095f90303fa262c6045	168252	168253	123621	30905	2014-04-11 00:00:00	77.0	77.0	left	
9	62c46095f90303fa262c6047	153133	153134	156585	31014	2013-03-22 00:00:00	73.0	78.0	left	m
10	62c46095f90303fa262c6048	143264	143265	212187	488298	2015-03-27 00:00:00	62.0	70.0	left	m
11	62c46095f90303fa262c604b	42876	42877	170589	39844	2009-08-30 00:00:00	68.0	81.0	left	m
12	62c46095f90303fa262c6050	151457	151458	7826	30843	2016-04-21 00:00:00	81.0	81.0	left	m
13	62c46095f90303fa262c6054	92331	92332	217647	179090	2015-01-09 00:00:00	72.0	75.0	left	
14	62c46095f90303fa262c605c	83498	83499	204713	242094	2015-01-30 00:00:00	75.0	83.0	left	m
15	62c46095f90303fa262c6064	60065	60066	214421	323438	2014-02-21 00:00:00	63.0	66.0	left	
16	62c46095f90303fa262c6069	107579	107580	190403	181574	2011-08-30 00:00:00	62.0	64.0	left	
17	62c46095f90303fa262c6073	108925	108926	176903	49866	2015-01-16 00:00:00	76.0	76.0	left	m
18	62c46095f90303fa262c607a	69508	69509	177638	41725	2014-02-14 00:00:00	68.0	71.0	left	m
19	62c46095f90303fa262c607e	68403	68404	184480	113005	2013-09-20 00:00:00	68.0	73.0	left	m
20	62c46095f90303fa262c607f	145762	145763	176253	68732	2007-02-22 00:00:00	58.0	71.0	left	m
21	62c46095f90303fa262c6081	46150	46151	202654	163618	2012-08-31 00:00:00	64.0	64.0	left	m
22	62c46095f90303fa262c6082	100514	100515	196921	193952	2010-08-30 00:00:00	62.0	78.0	left	m
23	62c46095f90303fa262c6084	143088	143089	181307	39081	2009-08-30 00:00:00	64.0	69.0	left	m
24	62c46095f90303fa262c608a	31286	31287	165429	26263	2013-03-22 00:00:00	70.0	74.0	left	m
25	62c46095f90303fa262c608d	81366	81367	199639	211020	2015-07-03 00:00:00	67.0	71.0	left	m
26	62c46095f90303fa262c608f	161026	161027	190286	150649	2014-10-24 00:00:00	78.0	84.0	left	
27	62c46095f90303fa262c6091	130316	130317	189403	139673	2014-09-18 00:00:00	71.0	76.0	left	
28	62c46095f90303fa262c6095	66366	66367	193651	127140	2016-04-28 00:00:00	62.0	67.0	left	m
29	62c46095f90303fa262c6096	71204	71205	49204	30788	2007-02-22 00:00:00	76.0	82.0	left	m
30	62c46095f90303fa262c609b	135214	135215	216688	202713	2014-02-07 00:00:00	66.0	74.0	left	m
31	62c46095f90303fa262c609e	118618	118619	202424	206743	2011-08-30 00:00:00	66.0	76.0	left	m
32	62c46095f90303fa262c60b0	174881	174882	186890	111994	2010-08-30 00:00:00	69.0	75.0	left	
33	62c46095f90303fa262c60b6	6690	6691	178393	41269	2013-09-20 00:00:00	81.0	85.0	left	m
34	62c46095f90303fa262c60b7	171494	171495	162145	42579	2008-08-30 00:00:00	74.0	80.0	left	m
35	62c46095f90303fa262c60b8	19322	19323	201114	206830	2014-07-18 00:00:00	68.0	82.0	left	m
36	62c46095f90303fa262c60bc	22966	22967	121591	40197	2013-10-25 00:00:00	73.0	73.0	left	
37	62c46095f90303fa262c60c0	39365	39366	172207	24234	2009-02-22 00:00:00	67.0	79.0	left	
38	62c46095f90303fa262c60c1	124550	124551	205501	186676	2015-07-03 00:00:00	68.0	68.0	left	m
39	62c46095f90303fa262c60c4	101416	101417	222492	530859	2015-05-15 00:00:00	66.0	79.0	left	
40	62c46095f90303fa262c60c6	118803	118804	205362	213485	2014-11-14 00:00:00	76.0	82.0	left	m
41	62c46095f90303fa262c60c9	15965	15966	188487	74291	2009-08-30 00:00:00	61.0	64.0	left	m
42	62c46095f90303fa262c60cc	32847	32848	187644	114339	2015-04-10 00:00:00	70.0	72.0	left	m
43	62c46095f90303fa262c60cd	104840	104841	210363	114672	2014-09-18 00:00:00	70.0	70.0	left	m
44	62c46095f90303fa262c60d0	124378	124379	168459	279742	2007-02-22 00:00:00	58.0	65.0	left	m
45	62c46095f90303fa262c60d2	46634	46635	110283	37450	2016-03-03 00:00:00	73.0	73.0	left	m
46	62c46095f90303fa262c60d7	158174	158175	208421	309334	2016-04-28 00:00:00	77.0	86.0	left	m
47	62c46095f90303fa262c60d8	108664	108665	192324	163236	2010-08-30 00:00:00	67.0	75.0	left	m
48	62c46095f90303fa262c60d9	159614	159615	199832	212959	2010-08-30 00:00:00	57.0	78.0	left	m
49	62c46095f90303fa262c60da	139218	139219	52091	32569	2014-09-18 00:00:00	80.0	80.0	left	
50	62c46095f90303fa262c60de	156287	156288	210747	477604	2013-09-20 00:00:00	62.0	71.0	left	m
51	62c46095f90303fa262c60e0	177760	177761	183488	114746	2014-09-18 00:00:00	71.0	74.0	left	m
52	62c46095f90303fa262c60e2	127121	127122	224314	415500	2016-05-05 00:00:00	66.0	72.0	left	m
53	62c46095f90303fa262c60e4	125819	125820	5818	30616	2007-08-30 00:00:00	76.0	81.0	left	

54 rows × 44 columns

- **Búsquedas con filtros y pipelines**

Procederemos a hacer algunas búsquedas con filtros(por ejemplo, buscar aquellos documentos con el atributo Xi< 0) y pipeline para agrupar.

- Buscamos aquella estrella en la tabla estrellas que contenga la palabara "agujero negro" mediante un filtro de expresión regular

In [22]:

```
cursor= db.estrellas.find( { "tipo": { "$regex": "negro" } } )
print(list(cursor))
```

```
[{'_id': ObjectId('62c4715e3265cbfc573d3492'), 'estrella': 'Sagitario A', 'Radio': 17250.0, 'tipo': 'agujero ne
gro supermasivo'}]
```

- También podemos buscar la misma estrella buscando la estrella con Null en color, y sólo miramos la columnas estrella y tipo. Sólo mostramos dos columnas

In [37]:

```
cursor= db.estrellas.find( { "color": None},{"estrella":1, "_id":0, "tipo":1})
print(list(cursor))
```

```
[{'estrella': 'Sagitario A', 'tipo': 'agujero negro supermasivo'}]
```

- En la tabla de futbolistas, buscamos aquellos jugadores con la columna "Potencial" mayor a 87 y **seleccionamos sólo unas pocas columnas** que queremos ver...

In [35]:

```
cursor= db.futbol.find({"potential": {"$gt":75}},{"player_fifa_api_id":1, "_id":0,"potential":1,"preferred_foot

df=pd.DataFrame(list(cursor))
df.head()
```

Out[35]:

	player_fifa_api_id	potential	preferred_foot
0	137427	76.0	right
1	188612	80.0	right
2	211293	79.0	right
3	189779	77.0	right
4	1257	82.0	right

En los dos últimos casos hemos puesto explícitamente **"_id":0** para que no apareciese _id en la búsqueda

Lo siguiente que haremos es un **GROUP BY** en la tabla de *ventas*. Agruparemos por IDtrabajador, y sumaremos su resultados

In [55]:

```
cursor= db.venta.aggregate([{"$group": {"_id":"$TRABAJADOR_Idtrabajador", "total":{"$sum":"$precio"} }}])

df=pd.DataFrame(cursor)
df
```

Out[55]:

	_id	total
0	8	90
1	1	90
2	6	170
3	14	130
4	13	470
5	5	235
6	4	70
7	3	128
8	2	167
9	7	200
10	10	60
11	9	330

In [ ]: