Universidad Nacional de Córdoba - Facultad de Matemática, Astronomía, Física y Computación

Diplomatura en Ciencia de Datos, Aprendizaje Automático y sus Aplicaciones

Practico Mentoria - Analisis y Visualizacion de Datos

Autor: Melania Omonte

Importaciones

In [9]:

```
%matplotlib inline

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

import seaborn as sns
import scipy as sp
from scipy import stats

from collections import OrderedDict
from IPython.display import display
from scipy.stats import variation, zscore, kstest, norm

from pylab import *

import warnings
warnings.filterwarnings('ignore')
```

In [10]:

```
sns.set_style("whitegrid")
sns.set_context('talk')
```

In [11]:

```
# Seteamos una semilla para Reproducibilidad
np.random.seed(1)
```

Carga de los Datasets

In [12]:

```
#dataset = pandas.read_csv('C:\Diplomatura\DataSets\hfi_cc_2018.csv')
#dataset.shape

#df_player = pd.read_csv('C:\Diplomatura\DataSets\football_player.csv',encoding='utf-8')
df_player = pd.read_csv('football_player.csv')
df_team = pd.read_csv('football_team.csv')
df_match = pd.read_csv('football_match.csv')
```

Exploremos un poco los Datasets

Players Dataset

In [13]:

```
print("Shape = {}".format(df_player.shape))
```

```
Shape = (9925, 44)
```

In [14]:

```
df_player.sample(10)
```

Out[14]:

	player_name	birthday	age	height_m	weight_kg	imc	overall_rating	potential	preferred_foot	attacking_work_rate	...	vision
858	Ariel Borysiuk	1991-07-29	24	1.80	69.85	21.48	66.12	74.38	right	medium	...	70.46
8529	Sava Miladinovic Bento	1991-01-02	25	1.83	72.12	21.56	58.00	64.43	right	medium	...	59.64
2527	Dusan Tadic	1988-11-20	27	1.80	76.20	23.43	78.16	81.88	left	medium	...	84.32
8473	Samuel Souprayen	1989-02-18	27	1.88	74.84	21.18	64.24	71.76	left	medium	...	46.52
1958	Daniele Croce	1982-09-09	33	1.73	68.04	22.81	67.68	67.68	right	high	...	68.47
4555	John Arne Riise	1980-09-24	35	1.88	82.10	23.24	76.32	77.64	left	high	...	64.14
8408	Saidy Janko	1995-10-22	20	1.78	69.85	22.10	62.13	76.53	right	high	...	41.00
3743	Helder Postiga	1982-08-02	33	1.80	76.20	23.43	76.04	76.93	right	high	...	67.15
2314	Denzel Slager	1993-05-02	23	1.83	81.19	24.28	61.50	70.75	left	high	...	47.00
2992	Fernando Marcal	1989-02-19	27	1.78	72.12	22.81	70.88	75.41	left	high	...	51.53

10 rows × 44 columns

In [15]:

```
df_player.dtypes
```

Out[15]:

```
player_name              object
birthday                 object
age                       int64
height_m                float64
weight_kg               float64
imc                     float64
overall_rating          float64
potential               float64
preferred_foot           object
attacking_work_rate      object
defensive_work_rate      object
crossing                float64
finishing               float64
heading_accuracy        float64
short_passing           float64
volleys                 float64
dribbling               float64
curve                   float64
free_kick_accuracy      float64
long_passing            float64
ball_control            float64
acceleration            float64
sprint_speed            float64
agility                 float64
reactions               float64
balance                 float64
shot_power              float64
jumping                 float64
stamina                 float64
strength                float64
```

```
long_shots              float64
aggression              float64
interceptions           float64
positioning             float64
vision                  float64
penalties               float64
marking                 float64
standing_tackle         float64
sliding_tackle          float64
gk_diving               float64
gk_handling             float64
gk_kicking              float64
gk_positioning          float64
gk_reflexes             float64
dtype: object
```

Teams Dataset

In [16]:

```
print("Shape = {}".format(df_team.shape))
```

```
Shape = (288, 22)
```

In [17]:

```
df_team.sample(10)
```

Out[17]:

	team_long_name	team_short_name	buildUpPlaySpeed	buildUpPlaySpeedClass	buildUpPlayDribblingClass	buildUpPlayPassing	bu
276	BSC Young Boys	YB	53.83	Balanced	Little	63.00	
98	VfL Wolfsburg	WOL	61.33	Balanced	Little	51.33	
186	Widzew Łódź	LOD	65.25	Balanced	Little	62.75	
73	FC Sochaux-Montbéliard	SOC	61.33	Balanced	Little	46.00	
269	RC Celta de Vigo	CEL	48.67	Balanced	Little	49.67	
226	Heart of Midlothian	HEA	59.60	Balanced	Little	60.00	
35	Middlesbrough	MID	62.67	Balanced	Little	55.83	
154	Vitesse	VIT	42.00	Balanced	Little	39.00	
253	Real Betis Balompié	BET	52.33	Balanced	Little	40.67	
108	Karlsruher SC	KAR	57.40	Balanced	Little	47.40	

10 rows × 22 columns

In [18]:

```
df_team.dtypes
```

Out[18]:

```
team_long_name                    object
team_short_name                   object
buildUpPlaySpeed                 float64
buildUpPlaySpeedClass             object
buildUpPlayDribblingClass         object
buildUpPlayPassing               float64
buildUpPlayPassingClass           object
buildUpPlayPositioningClass       object
chanceCreationPassing            float64
chanceCreationPassingClass        object
chanceCreationCrossing           float64
chanceCreationCrossingClass       object
```

```
chanceCreationShooting              float64
chanceCreationShootingClass          object
chanceCreationPositioningClass       object
defencePressure                     float64
defencePressureClass                 object
defenceAggression                   float64
defenceAggressionClass               object
defenceTeamWidth                    float64
defenceTeamWidthClass                object
defenceDefenderLineClass             object
dtype: object
```

Matchs Dataset

In [19]:

```
print("Shape = {}".format(df_match.shape))
```

```
Shape = (25979, 15)
```

In [20]:

```
df_match.sample(10)
```

Out[20]:

	country_name	league_name	season	stage	date	home_team_long_name	home_short_long_name	away_team_long_name
15289	Netherlands	Netherlands Eredivisie	2014/2015	28	2015-03-20	FC Utrecht	UTR	NAC Breda
6697	France	France Ligue 1	2013/2014	11	2013-10-26	Valenciennes FC	VAL	Évian Thonon Gaillard FC
15489	Netherlands	Netherlands Eredivisie	2015/2016	17	2015-12-19	Heracles Almelo	HER	FC Groningen
2579	England	England Premier League	2010/2011	18	2011-01-26	Liverpool	LIV	Fulham
10264	Italy	Italy Serie A	2008/2009	1	2008-08-31	Torino	TOR	Lecce
17444	Poland	Poland Ekstraklasa	2015/2016	14	2015-10-30	Górnik Łęczna	LEC	Cracovia
11088	Italy	Italy Serie A	2010/2011	16	2010-12-12	Brescia	BRE	Sampdoria
467	Belgium	Belgium Jupiler League	2009/2010	30	2010-03-21	Standard de Liège	STL	KAA Gent
17525	Poland	Poland Ekstraklasa	2015/2016	23	2016-02-21	Polonia Bytom	GOR	Ruch Chorzów
15773	Poland	Poland Ekstraklasa	2008/2009	15	2008-11-22	GKS Bełchatów	BEL	Jagiellonia Białystok

In [21]:

```
df_match.dtypes
```

Out[21]:

```
country_name             object
league_name              object
season                   object
stage                     int64
date                     object
home_team_long_name      object
home_short_long_name     object
away_team_long_name      object
away_short_long_name     object
home_team_goal            int64
away_team_goal            int64
```

```
total_goal              int64
B365H                 float64
B365D                 float64
B365A                 float64
dtype: object
```

Ejercicios

> Ejercicio 1

Calcular los siguientes Estadisticos:

```
Moda
Media
Mediana
Desviacion Estandar
Minimo y Maximo
```

de variables como el 'Shot Power' y 'Long Shots de los jugadores.

Ver si responden a alguna distribución conocida.

Moda de 'Shot Power' y Distribucion (Histograma)

In [22]:

```
## Remuevo las columnas que sean categoricas nominales.
nonordinals_cols_to_remove =
['player_name','birthday','preferred_foot','attacking_work_rate','defensive_work_rate']
df_player_range = df_player.drop(columns = nonordinals_cols_to_remove)
```

In [23]:

```
## Calculo el rango para cada columna.
pd.DataFrame([df_player_range.min(), df_player_range.max(), df_player_range.max() -
df_player_range.min()],
                     index=['min','max','rango']).transpose()
```

Out[23]:

	min	max	rango
age	17.00	47.00	30.00
height_m	1.57	2.08	0.51
weight_kg	53.07	110.22	57.15
imc	17.87	30.87	13.00
overall_rating	47.00	92.19	45.19
potential	51.00	95.23	44.23
crossing	6.00	89.36	83.36
finishing	5.00	92.23	87.23
heading_accuracy	8.00	93.11	85.11
short_passing	10.57	95.18	84.61
volleys	3.75	90.79	87.04
dribbling	5.14	96.46	91.32
curve	6.67	92.57	85.90
free_kick_accuracy	7.00	93.60	86.60
long_passing	11.33	94.16	82.83
ball_control	9.00	95.77	86.77
acceleration	15.00	95.79	80.79
sprint_speed	17.00	95.70	78.70

	min	max	rango
agility	21.00	94.67	73.67
reactions	28.00	92.54	64.54
balance	20.00	94.31	74.31
shot_power	9.92	93.08	83.16
jumping	21.00	94.31	73.31
stamina	16.00	93.18	77.18
strength	23.12	95.00	71.88
long_shots	6.00	89.88	83.88
aggression	11.00	93.00	82.00
interceptions	7.00	91.04	84.04
positioning	4.00	93.20	89.20
vision	8.00	95.68	87.68
penalties	9.43	89.57	80.14
marking	5.00	89.67	84.67
standing_tackle	6.00	90.20	84.20
sliding_tackle	6.00	94.37	88.37
gk_diving	1.94	89.86	87.92
gk_handling	3.26	82.90	79.64
gk_kicking	3.26	87.13	83.87
gk_positioning	3.26	90.16	86.90
gk_reflexes	3.26	90.95	87.69

Interpretacion:

De la tabla anterior, podemos observar que:

Se obtuvieron las variables asociadas a los jugadores entre 17 y 47 años.

Con respecto a las variable shot_power, sus valores varian entre 9.92 a 93.08 para los 9925 futbolistas que tiene el data set.

Con respecto a las variable long_shots, sus valores varian entre 6.00 a 89.88 para los 9925 futbolistas que tiene el data set.

In [24]:

```
## Remuevo valores NaN
df_player_cleaned = df_player[['age','preferred_foot','shot_power', 'long_shots']].dropna()
```

In [25]:

```
## Una pequeña descripcion para saber con que valores estoy tratando
df_player_cleaned.describe()
```

Out[25]:

	age	shot_power	long_shots
count	9925.000000	9925.000000	9925.000000
mean	28.207456	59.672008	50.919684
std	5.106009	15.287306	17.356235
min	17.000000	9.920000	6.000000
25%	24.000000	52.270000	38.800000
50%	28.000000	63.030000	55.150000
75%	32.000000	70.570000	64.040000
max	47.000000	93.080000	89.880000

In [26]:

```
## Medidas descriptivas para los campos shot_power y long_shots para el data set df_player
raw = {'shot_power': [df_player_cleaned.shot_power.median(), df_player_cleaned.shot_power.mean(), d
f_player_cleaned.shot_power.std()],
       'long_shots': [df_player_cleaned.long_shots.median(), df_player_cleaned.long_shots.mean(), c
f_player_cleaned.long_shots.std()]}
pd.DataFrame(raw, index=['mediana','media','desviacion'])
```

Out[26]:

	shot_power	long_shots
mediana	63.030000	55.150000
media	59.672008	50.919684
desviacion	15.287306	17.356235

Vale la pena calcular la moda?

In [29]:

```
## shot_power
## Cuantas modas hay
shot_power_mode_count = df_player_cleaned.shot_power.mode().count()

## Valor de la moda o modas.
shot_power_mode = df_player_cleaned.shot_power.mode()

## Frecuencias de shot_power.
shot_power_frec = pd.value_counts(df_player_cleaned.shot_power).to_frame().reset_index()
shot_power_frec.columns = ['shot_power','frecuencia']

## long_shots
## Cuantas modas hay
long_shots_mode_count = df_player_cleaned.long_shots.mode().count()

## Valor de la moda o modas.
long_shots_mode = df_player_cleaned.long_shots.mode()

## Frecuencias de long_shots.
long_shots_frec = pd.value_counts(df_player_cleaned.long_shots).to_frame().reset_index()
long_shots_frec.columns = ['long_shots','frecuencia']

## Armo cuadro con resutados.
raw = {
    'mode_count': [shot_power_mode_count, long_shots_mode_count],
    'mode': [shot_power_mode[0] if shot_power_mode_count == 1 else None, long_shots_mode[0] if long
_shots_mode_count == 1 else None ]
}
pd.DataFrame(raw, index=['shot_power','long_shots'])
```

Out[29]:

	mode_count	mode
shot_power	1	63.0
long_shots	1	59.0

In [39]:

```
long_shots_mode2 = df_player_cleaned.long_shots.mode()
shot_power_mode2 = df_player_cleaned.shot_power.mode()
long_shots_mode2
shot_power_mode2
long_shots_mode_count
```

Out[39]:

1

Interpretacion
Poder de tiro (shots_power) y tiros libres (longs_shots)

Si observamos la media para ambos grupos vemos que son bastante parecidos. Los coeficientes de variacion indican que las observaciones en el long_shots presentan una leve mayor dispersion con respecto a la media que el grupo de shots_power.

Si observamos la mediana para shot_power el valor central observado es de 59.672008 y para long_shots es de 50.919684

¿Validamos si tiene sentido calcular la moda?

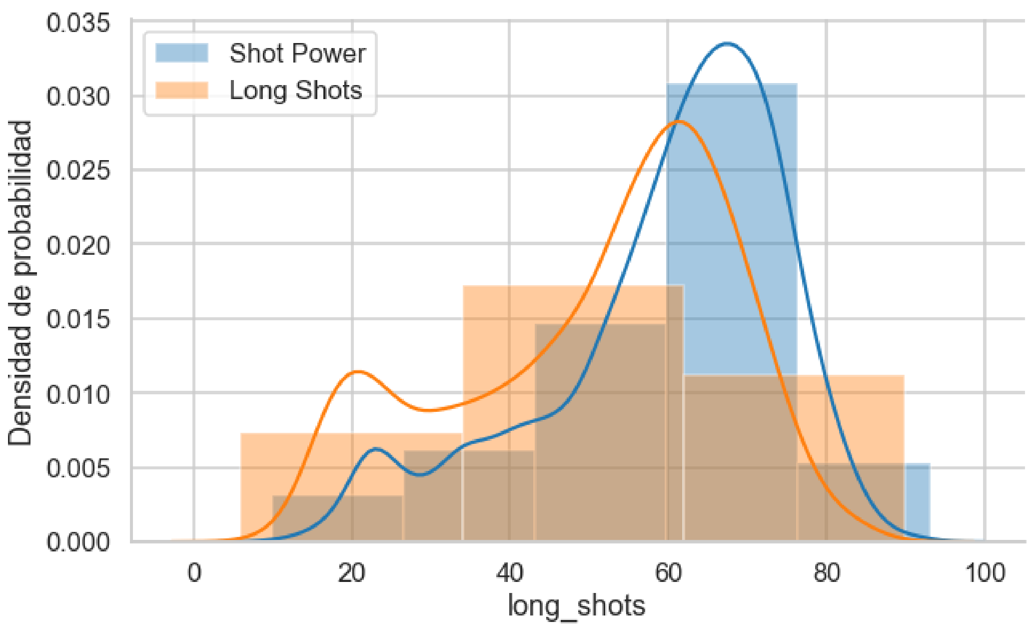
Segun lo que observamos,que ambos campos tienen una sola moda, las cuales son: shots_power=63 long_shots=59

In [32]:

```
### Distribucion que responde este campo

### plt.figure(figsize=(10,6))
### sns.distplot(df_player_cleaned.long_shots, kde=True, bins=10, label='Shot Power')
### sns.despine()

### plt.ylabel('Densidad de probabilidad')
### plt.xlabel('Shot Power')
### plt.legend()

plt.figure(figsize=(10,6))

## Grafico la distribucion del puntaje "shot power"
sns.distplot(df_player_cleaned.shot_power, kde=True, bins=5, label='Shot Power')

## Grafico la distribucion del puntaje "long shots"
sns.distplot(df_player_cleaned.long_shots, kde=True, bins=3, label='Long Shots')
sns.despine()

plt.ylabel('Densidad de probabilidad')
plt.legend()
```

Out[32]:

```
<matplotlib.legend.Legend at 0x19dabe79dd8>
```



In [130]:

```
#Asimetría y Curtosis de pf_identity Global:
print('shot_power:')
print(stats.skew(df_player_cleaned['shot_power'].dropna()), '(Asimetría)')
print(stats.kurtosis(df_player_cleaned['shot_power'].dropna()), '(Curtosis)')
print()

#Asimetría y Curtosis de hf_score Global:
```

```
print('long_shots:')
print(stats.skew(df_player_cleaned['long_shots'].dropna()), '(Asimetría)')
print(stats.kurtosis(df_player_cleaned['long_shots'].dropna()), '(Curtosis)')
print()
```

```
shot_power:
-0.9082444399727565 (Asimetría)
0.2820089529401022 (Curtosis)

long_shots:
-0.5317533149305995 (Asimetría)
-0.7076078929372929 (Curtosis)
```

Interpretacion:

Shot_power
Sigue una distribucion normal, con un leve sesgo a la izquierda (asimetria negativa). Su cola de distribución se alargan a la izquierda, con valores inferiores a su media.

Long_shots
Sigue una distribucion normal, con un leve sesgo a la izquierda (asimetria negativa), tiene un sesgo mucho menos que el campo shot power. Su cola de distribución se alargan a la izquierda, con valores inferiores a su media.

> Ejercicio 2

Realizar un Análisis de valores atípicos (outliers) de las variables anteriores.

In [36]:

```
## Outliers por grupo usando x veces la distancia a la media.

df_player_cleaned['shot_power_zscore'] = zscore(df_player_cleaned["shot_power"])
df_player_cleaned['long_shots_zscore'] = zscore(df_player_cleaned["long_shots"])

shot_power_zscore_outliers = df_player_cleaned["shot_power_zscore"].apply(
    ## Si zscore es menor 2.5 o mayor a 2.5, el valor observado esta en el 5% mas chico o mas gran
de.
  lambda x: x <= -2.5 or x >= 2.5
)
long_shots_zscore_outliers = df_player_cleaned["long_shots_zscore"].apply(
    ## Si zscore es menor 2.5 o mayor a 2.5, el valor observado esta en el 5% mas chico o mas gran
de.
  lambda x: x <= -2.5 or x >= 2.5
)
```

In [45]:

```
###Outliers de shots_power usando "x" veces la distancia la media:
df_player_cleaned[shot_power_zscore_outliers][["age","shot_power"]]
```

Out[45]:

	age	shot_power
88	31	19.60
169	33	20.76
263	31	18.82
285	35	19.41
365	39	21.29
378	38	20.73
436	25	21.37
454	31	19.38

	age	shot_power
653	25	20.94
839	21	21.00
893	27	19.63
1055	29	18.22
1056	26	17.96
1070	33	21.05
1098	25	18.17
1254	28	20.30
1260	34	17.00
1310	40	9.92
1378	42	20.19
1473	44	16.50
1625	22	20.00
1677	24	19.00
1713	23	21.00
1731	29	18.52
1748	32	16.65
1756	30	21.00
1805	27	15.17
1880	36	21.25
1914	32	19.54
2097	25	20.56
...	...	...
8567	29	18.73
8601	34	21.00
8605	34	20.86
8619	38	15.86
8694	23	20.29
8697	29	19.95
8848	23	20.43
8866	25	21.40
8940	29	21.00
8968	44	21.00
9025	35	17.67
9043	34	18.08
9105	41	15.89
9231	31	19.91
9262	20	20.18
9304	30	18.84
9335	27	17.07
9353	24	20.00
9425	22	21.00
9477	19	18.00
9508	24	21.00
9607	27	20.00
9627	21	20.00
9675	27	18.00
9715	23	17.78
9716	32	20.35
9725	32	18.07

	age	shot_power
9865	24	21.17
9878	23	14.53
9884	38	15.69

189 rows × 2 columns

In [43]:

```

```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-43-9dd066ee7bf6> in <module>
----> 1 df_player_cleaned[shot_power_zscore_outliers].max()

NameError: name 'shot_power_zscore_outliers' is not defined
```

In [46]:

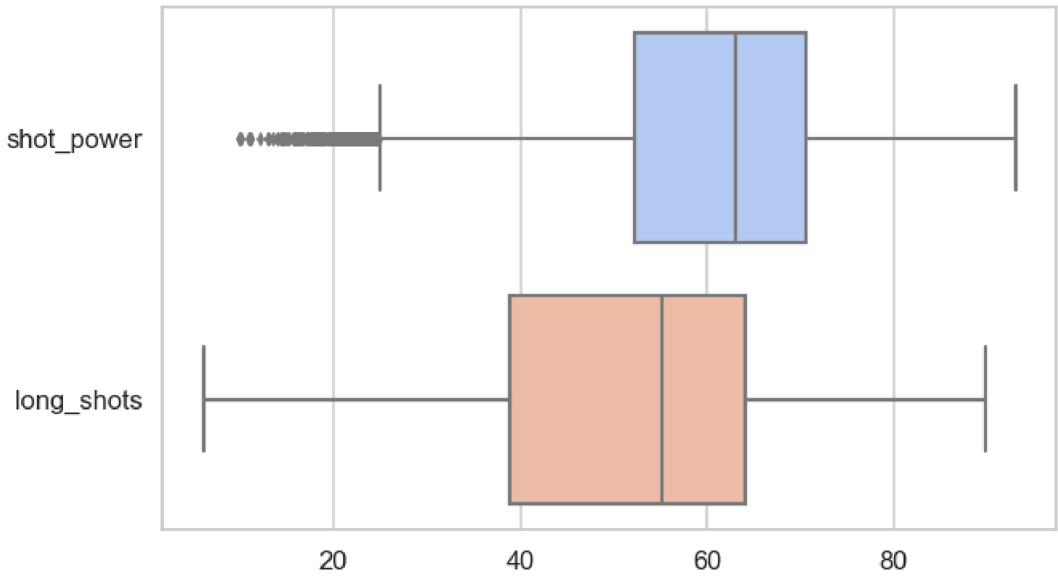
```
###Outliers de long_shots usando "x" veces la distancia la media:
df_player_cleaned[long_shots_zscore_outliers][["age","long_shots"]]
```

Out[46]:

	age	long_shots
4840	20	7.00
6745	21	6.00
8266	41	7.43

In [30]:

```
#Para observar valores atípicos visualizamos el gráfico de caja...
plt.figure(figsize=(10,6))
data = df_player_cleaned[['shot_power', 'long_shots']]
sns.boxplot(data = data, orient="h", palette="coolwarm")
#sns.stripplot(data=data, color='black')
plt.show()
```



In [149]:

```
print(data.quantile(0.25))
print()
print(data.quantile(0.75))
```

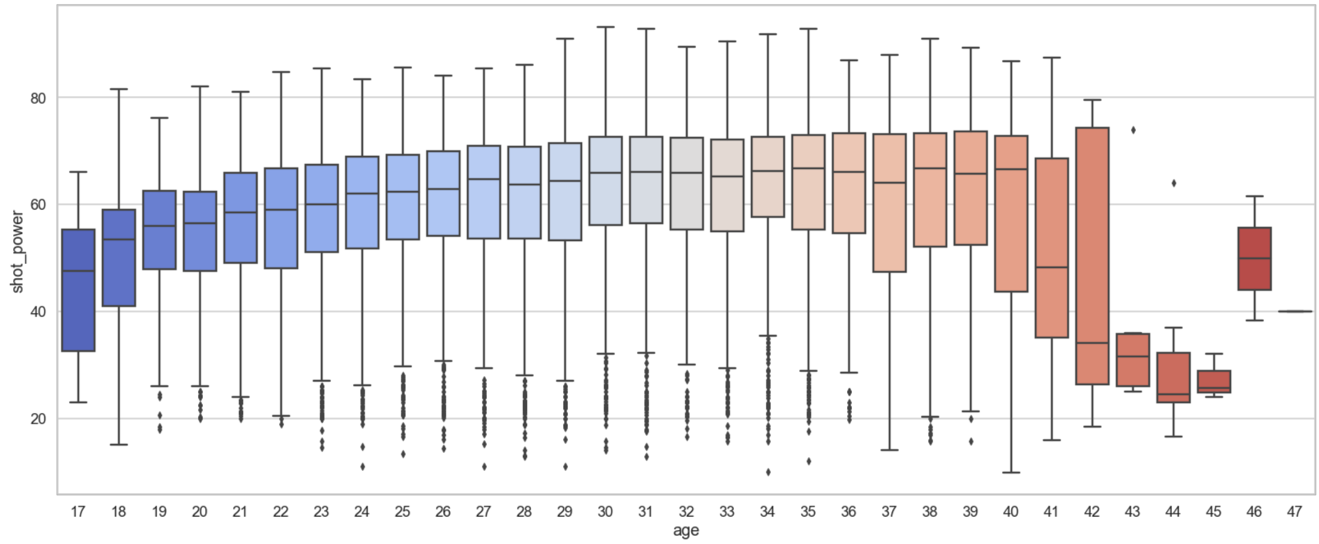
```
shot_power     52.27
long_shots     38.80
Name: 0.25, dtype: float64

shot_power     70.57
long_shots     64.04
Name: 0.75, dtype: float64
```

In [62]:

```python
## Box plot para shot_power
plt.figure(figsize=(25,10))
sns.boxplot(x="age", y="shot_power", data=df_player_cleaned, palette="coolwarm")
```

Out[62]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x16c0bbb9320>
```

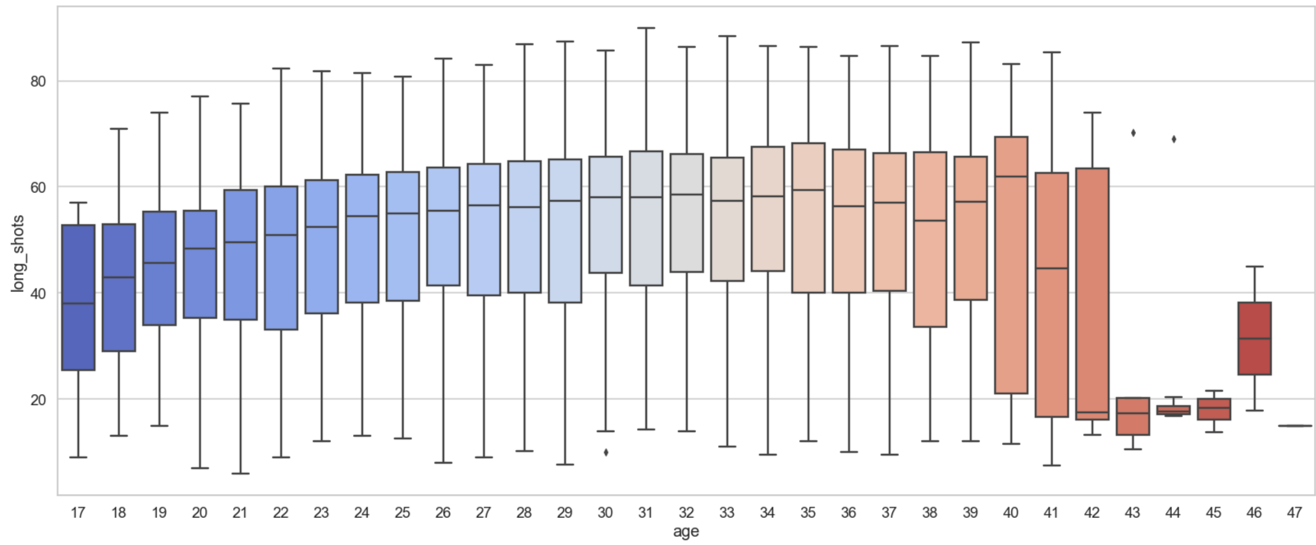


In [40]:

```python
## Box plot para shot_power
plt.figure(figsize=(25,10))
sns.boxplot(x="age", y="long_shots", data=df_player_cleaned, palette="coolwarm")
```

Out[40]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x19dabecc2b0>
```



In [55]:

```
###tips = sns.load_dataset("tips")
```

In [56]:

```
#Calculamos límites para Valores atípicos de los Quartiles:

AtMin_shot_power = data.quantile(0.25)[0] - (1.5 * (data.quantile(0.75)[0] - data.quantile(0.25)[0]
))
AtMax_shot_power = data.quantile(0.75)[0] + (1.5 * (data.quantile(0.75)[0] - data.quantile(0.25)[0]
))

AtMin_long_shots = data.quantile(0.25)[1] - (1.5 * (data.quantile(0.75)[1] - data.quantile(0.25)[1]
))
AtMax_long_shots = data.quantile(0.75)[1] + (1.5 * (data.quantile(0.75)[1] - data.quantile(0.25)[1]
))

print('límites para shot power: (', AtMin_shot_power, ',', AtMax_shot_power,')')
print('límites para long shots: (', AtMin_long_shots, ',' , AtMax_long_shots,')')
```

```
límites para shot power: ( 24.820000000000018 , 98.01999999999998 )
límites para long shots: ( 0.9399999999999835 , 101.90000000000002 )
```

In [58]:

```
#Valores Atípicos para shot_power

df_player_cleaned.loc[(df_player_cleaned['shot_power'] <= AtMin_shot_power) | (df_player_cleaned['
shot_power'] >= AtMax_shot_power)]
```

Out[58]:

	age	preferred_foot	shot_power	long_shots
88	31	right	19.60	19.20
169	33	right	20.76	14.41
227	42	right	22.00	15.78
245	17	right	23.00	9.00
258	24	left	24.62	18.88
263	31	right	18.82	18.82
272	27	right	24.20	14.33
277	32	right	23.35	42.00
285	35	right	19.41	31.94
365	39	right	21.29	12.64
378	38	right	20.73	16.69
380	26	right	21.88	20.08
422	23	right	23.00	12.54
436	25	right	21.37	17.26
454	31	right	19.38	19.75
473	34	right	23.75	19.28
486	34	right	22.10	17.71
507	27	right	24.00	19.94
606	20	right	21.67	19.67
653	25	right	20.94	16.44
663	34	right	22.65	20.48
678	29	right	24.09	19.00
684	42	right	22.00	14.85
714	26	right	22.00	12.00
761	32	right	24.21	20.07
789	28	right	24.40	20.65

	age	preferred_foot	shot_power	long_shots
800	20	right	22.40	20.60
822	31	right	21.71	18.36
834	33	right	21.50	18.50
839	21	right	21.00	17.00
...	...	...	...	...
9259	22	right	22.00	16.00
9262	20	right	20.18	22.45
9284	28	right	23.18	16.27
9304	30	right	18.84	17.63
9320	31	left	23.00	15.00
9335	27	right	17.07	21.87
9353	24	right	20.00	25.00
9381	36	right	23.06	16.28
9420	26	right	22.00	16.00
9425	22	right	21.00	25.00
9455	20	right	24.00	23.29
9477	19	right	18.00	17.00
9486	26	right	22.32	20.32
9508	24	right	21.00	13.00
9549	28	left	22.07	25.07
9588	37	right	23.05	14.42
9607	27	left	20.00	25.00
9627	21	right	20.00	21.00
9661	34	right	22.88	17.40
9675	27	right	18.00	18.00
9676	29	left	23.00	22.33
9711	33	right	23.12	24.59
9715	23	right	17.78	17.56
9716	32	right	20.35	17.70
9725	32	right	18.07	20.07
9800	22	right	23.62	17.25
9865	21	right	21.17	18.50
9866	24	right	21.72	16.89
9878	23	right	14.53	12.53
9884	38	right	15.69	17.75

428 rows × 4 columns

In [59]:

```
#Valores Atípicos para long_Shots:

df_player_cleaned.loc[(df_player_cleaned['long_shots'] <= AtMin_long_shots) | (df_player_cleaned['
long_shots'] >= AtMax_long_shots)]
```

Out[59]:

age	preferred_foot	shot_power	long_shots

In [65]:

```
#Los outliers de hf_score son de grupo: Syria (Middle East & North Africa), aumentando en número s
i el análisis es por Región.
#Sugeriríamos eliminarlos si el foco fuera GLOBAL (son pocos), analizarlos -tal vez ajustarlos- si
fuera REGIONAL.
```

Ejercicio 3

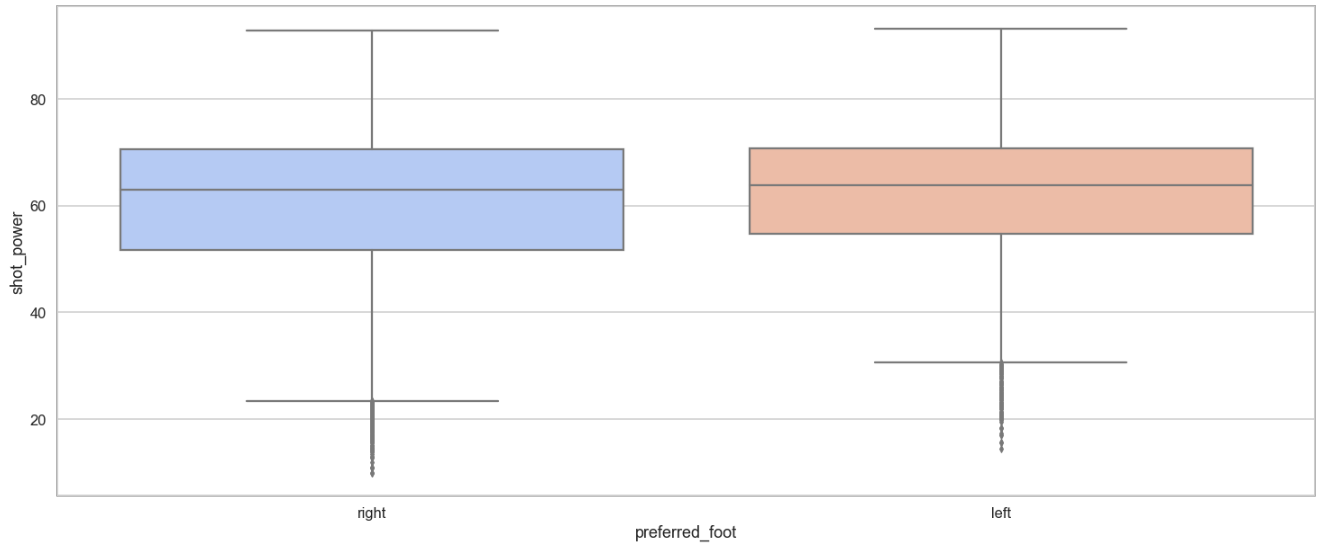
Explicar cómo varía el analisis hecho anteriormente cuando se desglosan por la pierna hábil.
Comparar cualitativamente y gráficamente ambas distribuciones.

In [60]:

```
## Box plot para pf_identity
plt.figure(figsize=(25,10))
sns.boxplot(x="preferred_foot", y="shot_power", data=df_player_cleaned, palette="coolwarm")
```

Out[60]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x16c0bb02c88>
```

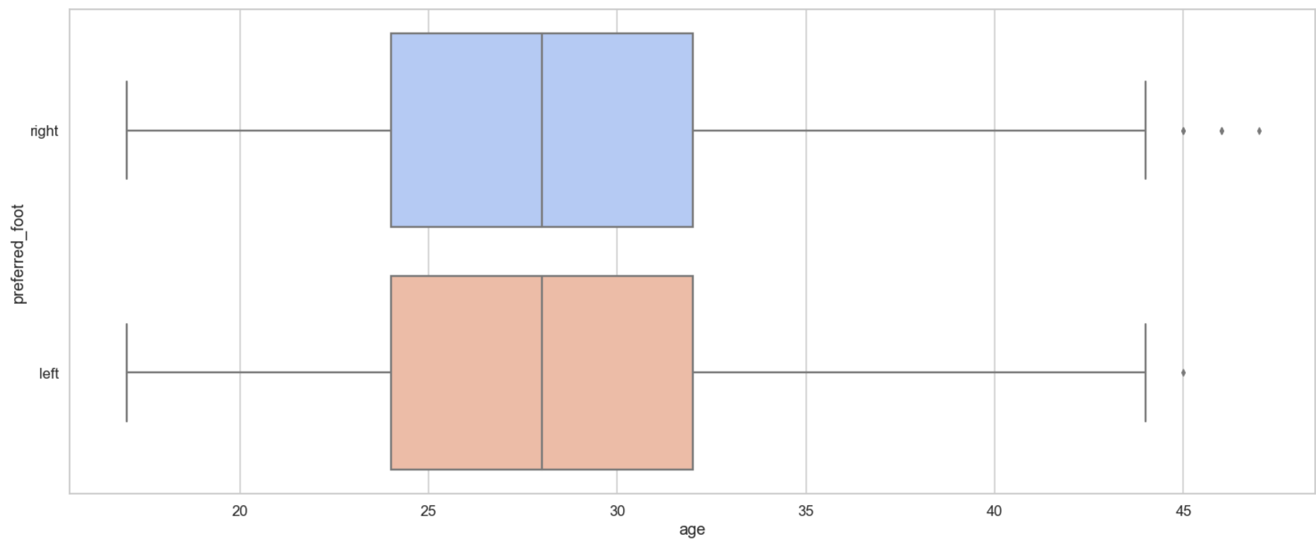


In [41]:

```
## Box plot para pf_identity
plt.figure(figsize=(25,10))
sns.boxplot(x="age", y="preferred_foot", data=df_player_cleaned, palette="coolwarm")
```

Out[41]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x19dabecc630>
```

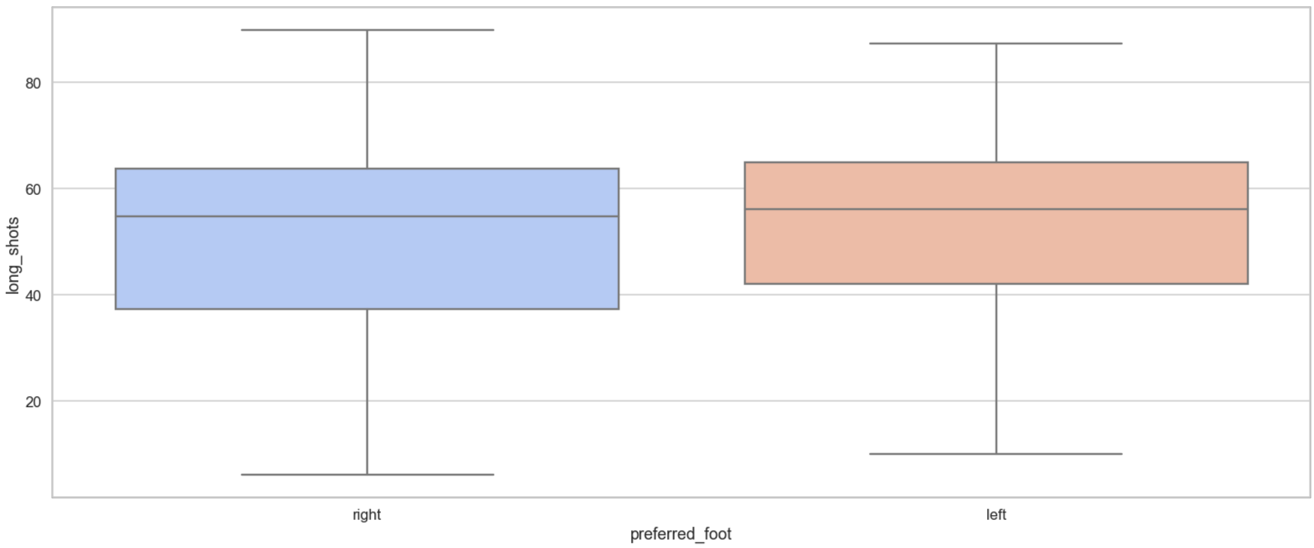


In [61]:

```
## Box plot para pf_identity
plt.figure(figsize=(25,10))
sns.boxplot(x="preferred_foot", y="long_shots", data=df_player_cleaned, palette="coolwarm")
```

Out[61]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x16c0bbf5a90>
```

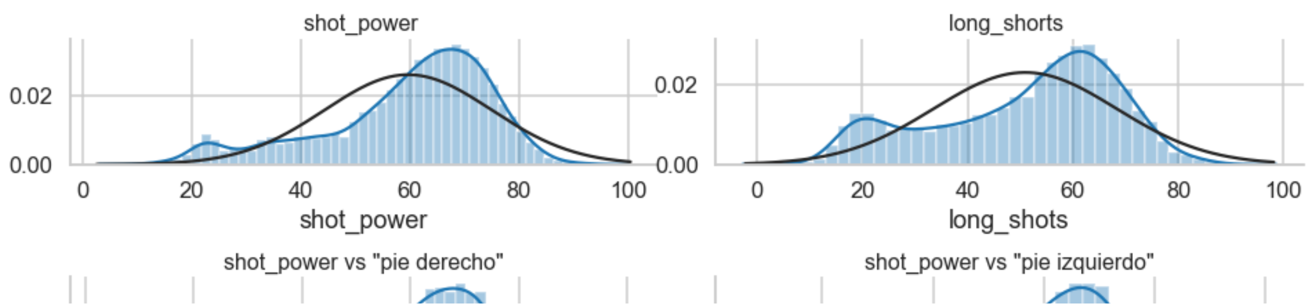


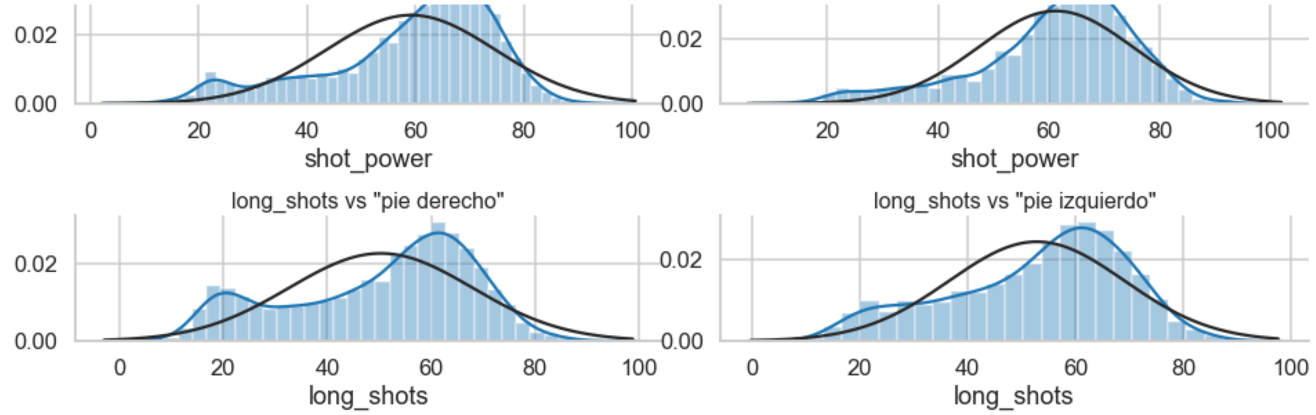
> Ejercicio 4

Graficar la correlacion de los features de los jugadores.
Calcular la correlacion entre los features 'Shot Power' y 'Long Shots' desglosando por la pierna habil.

In [100]:

```
#Comparamos shot_power y long_shots con su pie habil con sus "normales".

fig, axes = plt.subplots(3, 2, figsize = (16, 8))
plt.subplots_adjust(hspace = 0.9, wspace = 0.1)

sns.distplot(df_player_cleaned['shot_power'].dropna(), fit = stats.norm, ax = axes[0,0]).set_title(
'shot_power ', fontsize = 16)
sns.distplot(df_player_cleaned['long_shots'].dropna(), fit = stats.norm, ax = axes[0,1]).set_title(
'long_shorts ', fontsize = 16)

sns.distplot(df_player_cleaned.loc[df_player_cleaned['preferred_foot'] == 'right']
['shot_power'].dropna(), fit = stats.norm, ax = axes[1,0]).set_title('shot_power vs "pie derecho"',
fontsize = 16)
sns.distplot(df_player_cleaned.loc[df_player_cleaned['preferred_foot'] == 'left']['shot_power'].dro
pna(), fit = stats.norm, ax = axes[1,1]).set_title('shot_power vs "pie izquierdo"', fontsize = 16)

sns.distplot(df_player_cleaned.loc[df_player_cleaned['preferred_foot'] == 'right']
['long_shots'].dropna(), fit = stats.norm, ax = axes[2,0]).set_title('long_shots vs "pie derecho"',
fontsize = 16)
sns.distplot(df_player_cleaned.loc[df_player_cleaned['preferred_foot'] == 'left']['long_shots'].dro
pna(), fit = stats.norm, ax = axes[2,1]).set_title('long_shots vs "pie izquierdo"', fontsize = 16)

sns.despine()
```



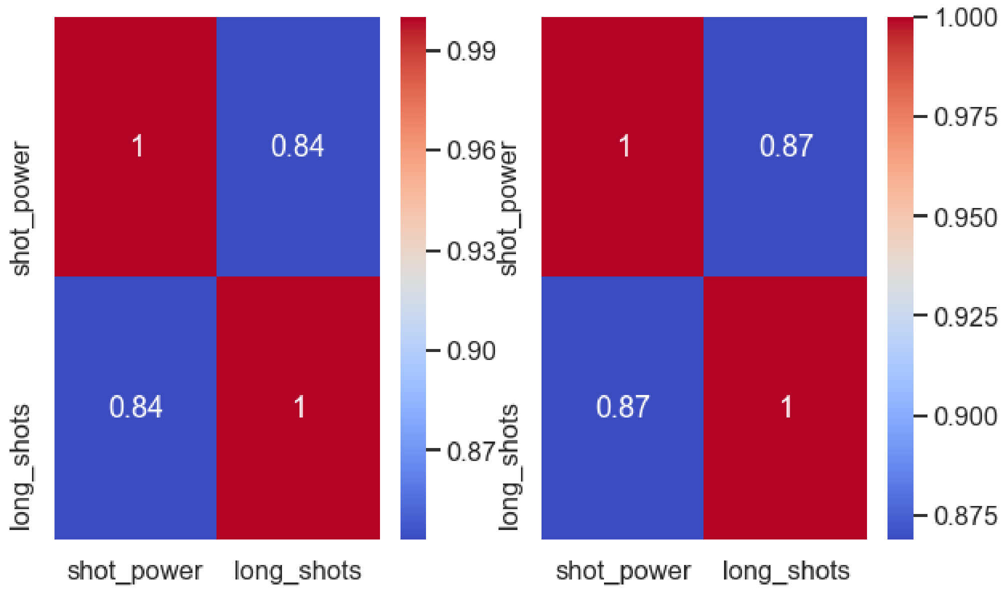


In [69]:

```
plt.figure(figsize=(10,6))

# Correlacion entre zurdos

plt.subplot(1,2,1)
sns.heatmap(df_player[df_player.preferred_foot=="left"][["shot_power", "long_shots"]].corr(method=
"pearson"),
            annot=True, cmap="coolwarm")

# Correlacion entre diestros
plt.subplot(1,2,2)
sns.heatmap(df_player[df_player.preferred_foot=="right"][["shot_power", "long_shots"]].corr(method
="pearson"),
            annot=True, cmap="coolwarm")
```

Out[69]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x19daefe9b38>
```



In [51]:

```
# coeficiente de correlación
df_player_corr = df_player[df_player.preferred_foot=="right"][['shot_power','long_shots']].dropna(
)
df_player_corr.corr()
```

Out[51]:

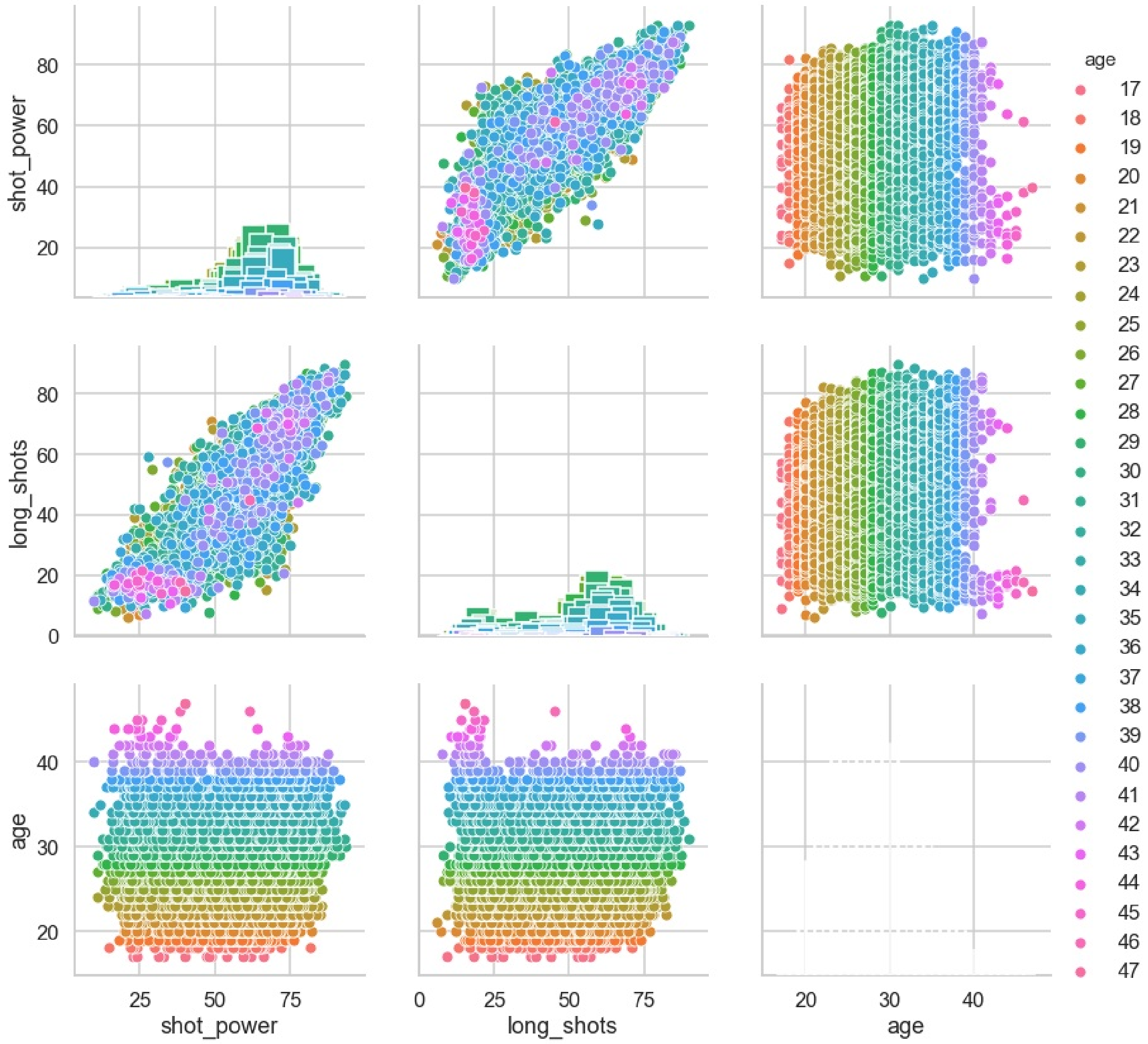
	shot_power	long_shots
shot_power	1.00000	0.86886
long_shots	0.86886	1.00000

long_shots 0.00000 1.00000

shot_power long_shots

In [60]:

```
sns.pairplot(data=df_player, vars = ['shot_power','long_shots','age'], height = 4, hue = 'age', dia
g_kind = 'hist')

plt.show()
```



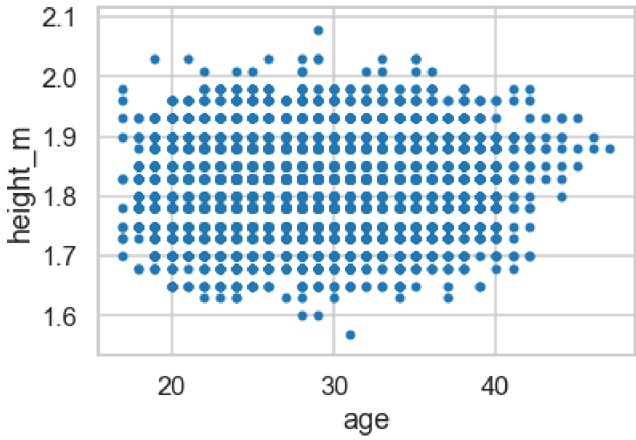
Interpretacion
dfdsfd

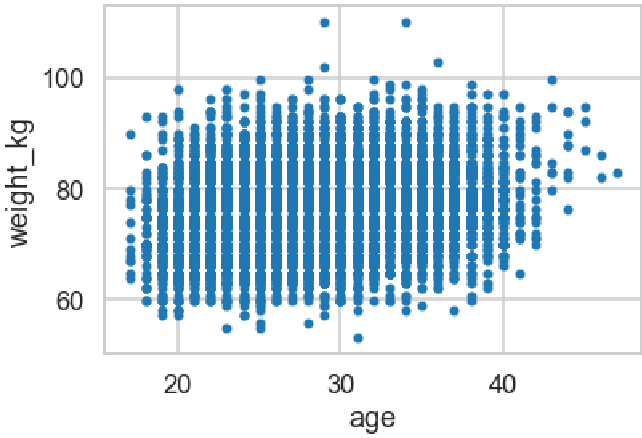
> Ejercicio 5

Graficar la correlacion de los entre los features 'Weight' y 'Age' de los jugadores. Que conclusiones se obtienen?
Graficar la correlacion de los entre los features 'Height' y 'Age' de los jugadores. Que conclusiones se obtienen?

In [52]:

```
# diagrama de dispersión

disp_height= df_player.plot(kind='scatter', x='age', y='height_m')
disp_weight= df_player.plot(kind='scatter', x='age', y='weight_kg')
```





In [54]:

```
# coeficiente de correlación
df_player_corr_height = df_player[['height_m','age']].dropna()
df_player_corr_height.corr()
```

Out[54]:

	height_m	age
height_m	1.000000	0.077192
age	0.077192	1.000000

In [55]:

```
# coeficiente de correlación
df_player_corr_weight = df_player[['weight_kg','age']].dropna()
df_player_corr_weight.corr()
```

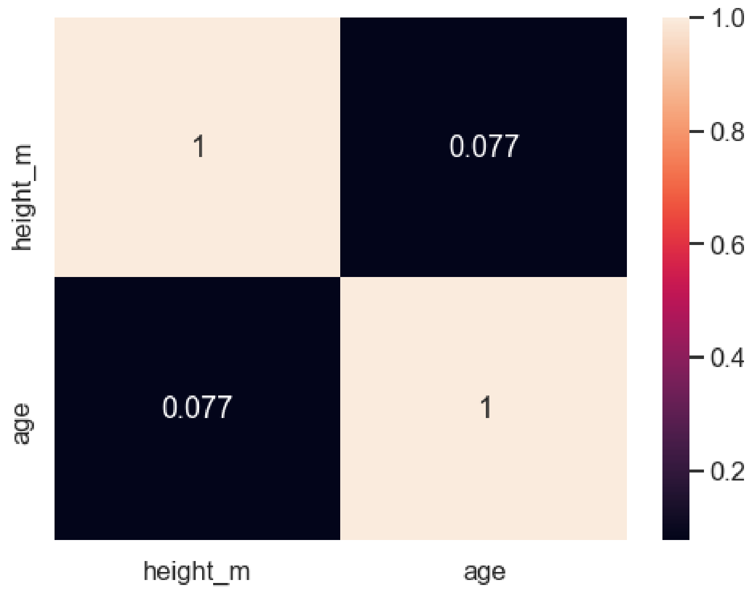
Out[55]:

	weight_kg	age
weight_kg	1.000000	0.199906
age	0.199906	1.000000

In [67]:

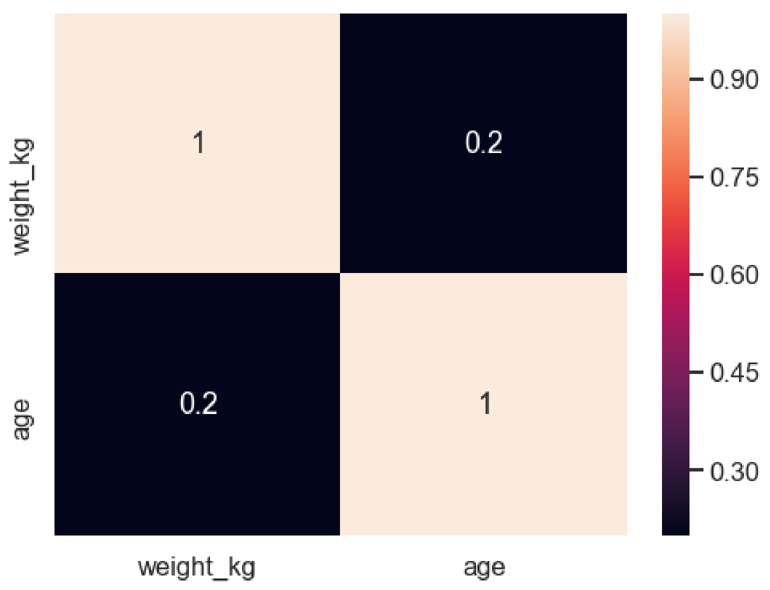
```
#
plt.figure(figsize=(8, 6))

sns.heatmap(df_player_corr[['height_m','age']].corr(method = 'pearson'), annot=True)
```

```
plt.show()
```



In [68]:

```
#
plt.figure(figsize=(8, 6))

sns.heatmap(df_player_corr[['weight_kg','age']].corr(method = 'pearson'), annot=True)

plt.show()
```



In [80]:

```
datos_minimo=df_player['age'].min()
datos_maximo=df_player['age'].max()


pd.DataFrame([df_player['age'].min(), df_player['age'].max()],
                    index=['min','max']).transpose()
```

Out[80]:

	min	max
0	17	47

Interpretacion Ejercicio 5
Correlacion entre Altura y Edad (Height y age) La correlacion entre la altura y la edad es bastante baja, lo que nos sugiere que estas variables no están relacionadas para este set de datos, esto tal vez se deba a que la edad minima del jugador es de 17 años y la edad maxima es de 47, o sea no son edades de maximo crecimiento de las personas.

Correlacion entre Peso y Edad (weight y age)

> Ejercicio 6

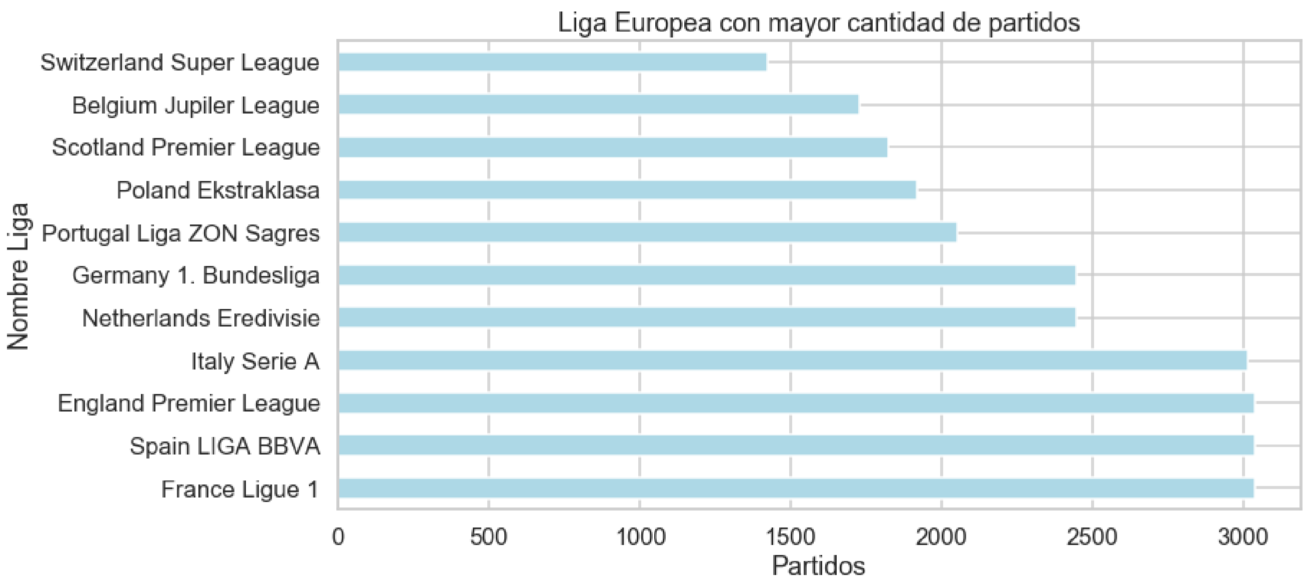
Liga Europea con mayor cantidad de partidos

In [271]:

```
partidos_por_liga = df_match["league_name"].value_counts()
plt.figure(figsize=(12,6))
partidos_por_liga.plot(kind="barh",color='lightblue')
plt.xlabel('Partidos')
plt.ylabel('Nombre Liga')
plt.title('Liga Europea con mayor cantidad de partidos')
```

Out[271]:

```
Text(0.5, 1.0, 'Liga Europea con mayor cantidad de partidos')
```



In [137]:

```
print(partidos_por_liga)
print("\nLiga/s  Europea/s con mayor cantidad de
partidos:\n{}".format(partidos_por_liga[partidos_por_liga==partidos_por_liga.max()]))
```

```
France Ligue 1              3040
Spain LIGA BBVA             3040
England Premier League      3040
Italy Serie A               3017
Netherlands Eredivisie      2448
Germany 1. Bundesliga       2448
Portugal Liga ZON Sagres    2052
Poland Ekstraklasa          1920
Scotland Premier League     1824
Belgium Jupiler League      1728
Switzerland Super League    1422
Name: league_name, dtype: int64

Liga/s  Europea/s con mayor cantidad de partidos:
France Ligue 1            3040
Spain LIGA BBVA           3040
England Premier League    3040
```

Name: league_name, dtype: int64

Interpretacion Ejercicio 6
Existen 3 ligas europeas que comparten el podio con mayor cantidad de partidos, para encontrar estos valores buscamos los valores maximos de partidos por liga

> Ejercicio 7

Top 10 de Equipos con mayor cantidad de goles convertidos: Total, Local y Visitante

In [87]:

```
total_local = df_match.groupby(["home_team_long_name"])[["home_team_long_name", "home_team_goal"]]
.sum()
total_visitante = df_match.groupby(["away_team_long_name"])[["away_team_long_name",
"away_team_goal"]].sum()
total_equipo = pd.concat([total_local, total_visitante], axis=1)
total_equipo["total_equipo_gol"] = total_equipo["home_team_goal"] + total_equipo["away_team_goal"]

# Los ordeno de mayor a menor
podio_total_local_gol = total_equipo.sort_values(["home_team_goal"], ascending=False)[0:10]
["home_team_goal"]
podio_total_visitante_gol = total_equipo.sort_values(["away_team_goal"], ascending=False)[0:10]
["away_team_goal"]
podio_total_equipo_gol = total_equipo.sort_values(["total_equipo_gol"], ascending=False)[0:10]
["total_equipo_gol"]

display(podio_total_local_gol,podio_total_visitante_gol,podio_total_equipo_gol)
print("\nPodio Total Goles de Local:")
print(podio_total_local_gol)
print("\nPodio Total Goles De Visitante:")
print(podio_total_visitante_gol)
print("\nTotal Goles:")
print(podio_total_equipo_gol)
```

```
Podio Total Goles de Local:
Real Madrid CF       505
FC Barcelona         495
Celtic               389
FC Bayern Munich     382
PSV                  370
Manchester City      365
Ajax                 360
FC Basel             344
Manchester United    338
Chelsea              333
Name: home_team_goal, dtype: int64

Podio Total Goles De Visitante:
FC Barcelona         354
Real Madrid CF       338
Celtic               306
Ajax                 287
PSV                  282
FC Basel             275
FC Bayern Munich     271
Arsenal              267
Borussia Dortmund    253
Chelsea              250
Name: away_team_goal, dtype: int64

Total Goles:
FC Barcelona         849
Real Madrid CF       843
Celtic               695
FC Bayern Munich     653
PSV                  652
Ajax                 647
FC Basel             619
```

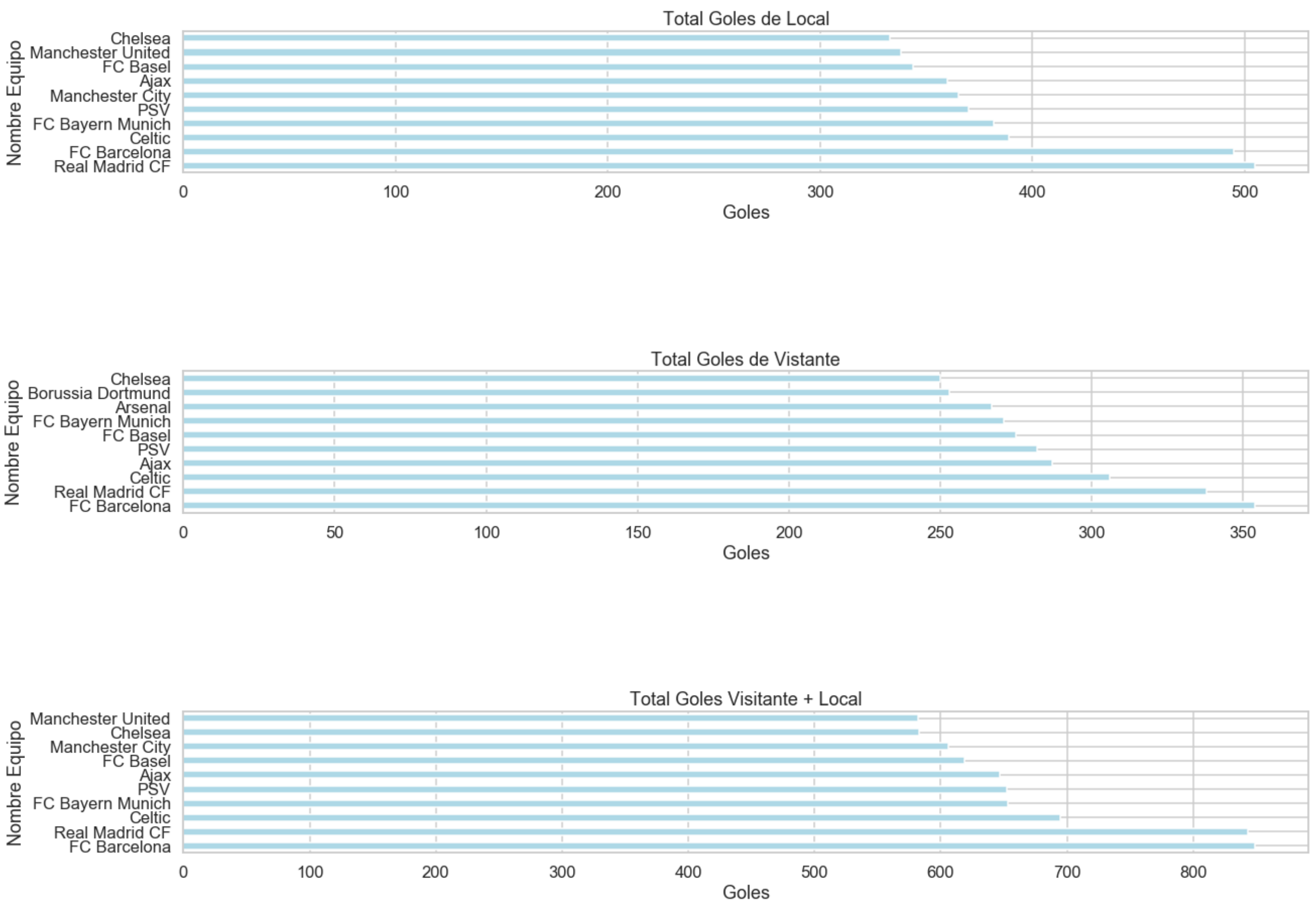
```
Manchester City      606
Chelsea              583
Manchester United    582
Name: total_equipo_gol, dtype: int64
```

In [97]:

```python
plt.figure(figsize=(20,15))

plt.subplot(5,1,1)
podio_total_local_gol.plot(kind='barh',color='lightblue')
plt.xlabel('Goles')
plt.ylabel('Nombre Equipo')
plt.title('Total Goles de Local')

plt.subplot(5,1,3)
podio_total_visitante_gol.plot(kind='barh', color='lightblue')
plt.xlabel('Goles')
plt.ylabel('Nombre Equipo')
plt.title('Total Goles de Vistante')

plt.subplot(5,1,5)
podio_total_equipo_gol.plot(kind='barh', title="Total Goles",color='lightblue')
plt.xlabel('Goles')
plt.ylabel('Nombre Equipo')
plt.title('Total Goles Visitante + Local')
```

Out[97]:

```
Text(0.5, 1.0, 'Total Goles Visitante + Local')
```



Interpretacion Ejercicio 7

Luego de realizar los calculos, asumimos que los 10 equipos con mas cantidad de goles son:

FC Barcelona	849
Real Madrid CF	843
Celtic	695

FC Bayern Munich	653
PSV	652
Ajax	647
FC Basel	619
Manchester City	606
Chelsea	583
Manchester United	582

Estando en el podio, el Equipo de FC Barcelona

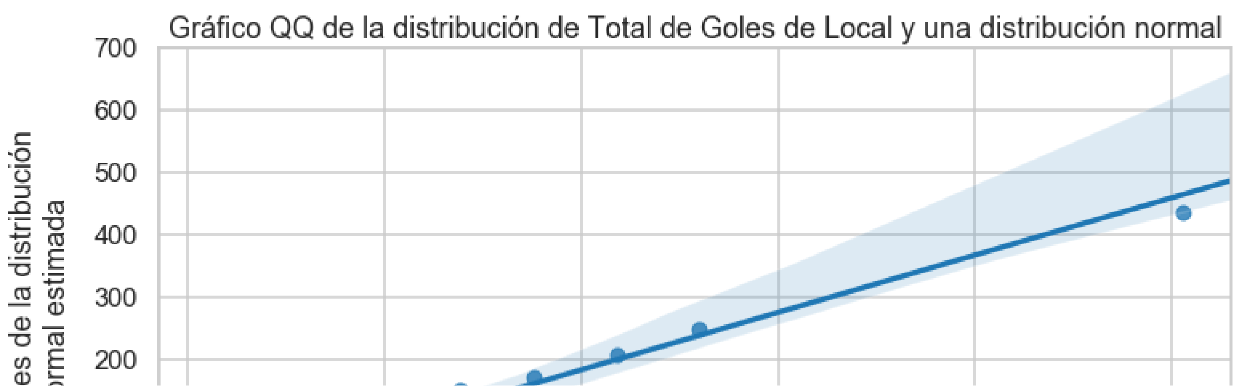
> Ejercicio 8

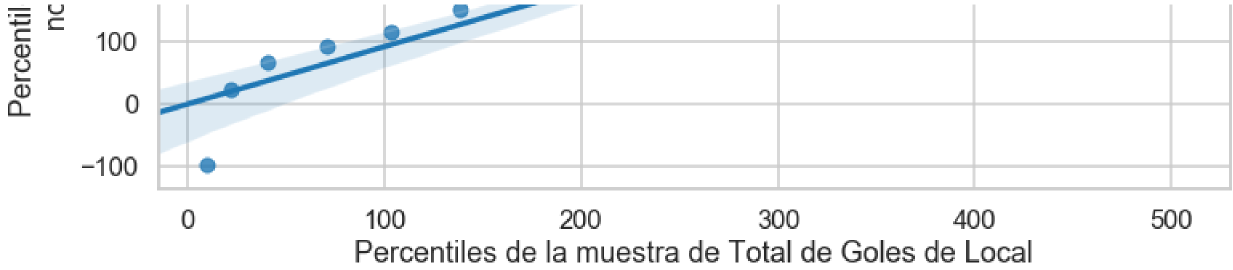
Distribucion de Cantidad de goles convertidos: Total, Local y Visitante Hacer un histograma modificar esto.

In [98]:

```
#Gráfico QQ para home_team_goal Global:

loc, scale = stats.norm.fit(total_equipo["home_team_goal"].dropna())
norm_dist = stats.norm(loc, scale)

percs = np.linspace(0,100,10)  # Creamos 10 puntos percentiles igualmente distribuidos entre 0 y 10
0.

home_team_goal_global_sample = np.percentile(total_equipo["home_team_goal"].dropna(), percs)
home_team_goal_global_norm = np.percentile(norm_dist.rvs(len(total_equipo["home_team_goal"].dropna(
))), percs)

plt.figure(figsize=(12,6))

sns.regplot(x = home_team_goal_global_sample, y = home_team_goal_global_norm)

plt.xlabel('Percentiles de la muestra de Total de Goles de Local')
plt.ylabel('Percentiles de la distribución \n normal estimada')
plt.title('Gráfico QQ de la distribución de Total de Goles de Local y una distribución normal')

x = np.linspace(np.min((home_team_goal_global_sample.min(), home_team_goal_global_norm.min())), np.
max((home_team_goal_global_sample.max(), pf_identity_global_norm.max())))
plt.plot(x,x, color='r', ls="--")

sns.despine()
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-98-c031873add87> in <module>
     17 plt.title('Gráfico QQ de la distribución de Total de Goles de Local y una distribución norm
al')
     18 
---> 19 x = np.linspace(np.min((home_team_goal_global_sample.min(), home_team_goal_global_norm.min(
))), np.max((home_team_goal_global_sample.max(), pf_identity_global_norm.max())))
     20 plt.plot(x,x, color='r', ls="--")
     21 

NameError: name 'pf_identity_global_norm' is not defined
```





In [92]:

```
#Gráfico QQ para pf_identity Global:

loc, scale = stats.norm.fit(total_equipo["away_team_goal"].dropna())
norm_dist = stats.norm(loc, scale)

percs = np.linspace(0,100,10)  # Creamos 10 puntos percentiles igualmente distribuidos entre 0 y 10
0.

away_team_goal_global_sample = np.percentile(total_equipo["away_team_goal"].dropna(), percs)
away_team_goal_global_norm = np.percentile(norm_dist.rvs(len(total_equipo["away_team_goal"].dropna(
))), percs)

plt.figure(figsize=(12,6))

sns.regplot(x = way_team_goal_global_sample, y = way_team_goal_global_norm)

plt.xlabel('Percentiles de la muestra de Total de Goles de Visitante')
plt.ylabel('Percentiles de la distribución \n normal estimada')
plt.title('Gráfico QQ de la distribución de Total de Goles de Visitante y una distribución normal'
)

x = np.linspace(np.min((away_team_goal_global_sample.min(), away_team_goal_global_norm.min())), np.
max((away_team_goal_global_sample.max(), away_team_goal_global_norm.max())))
plt.plot(x,x, color='r', ls="--")

sns.despine()
```

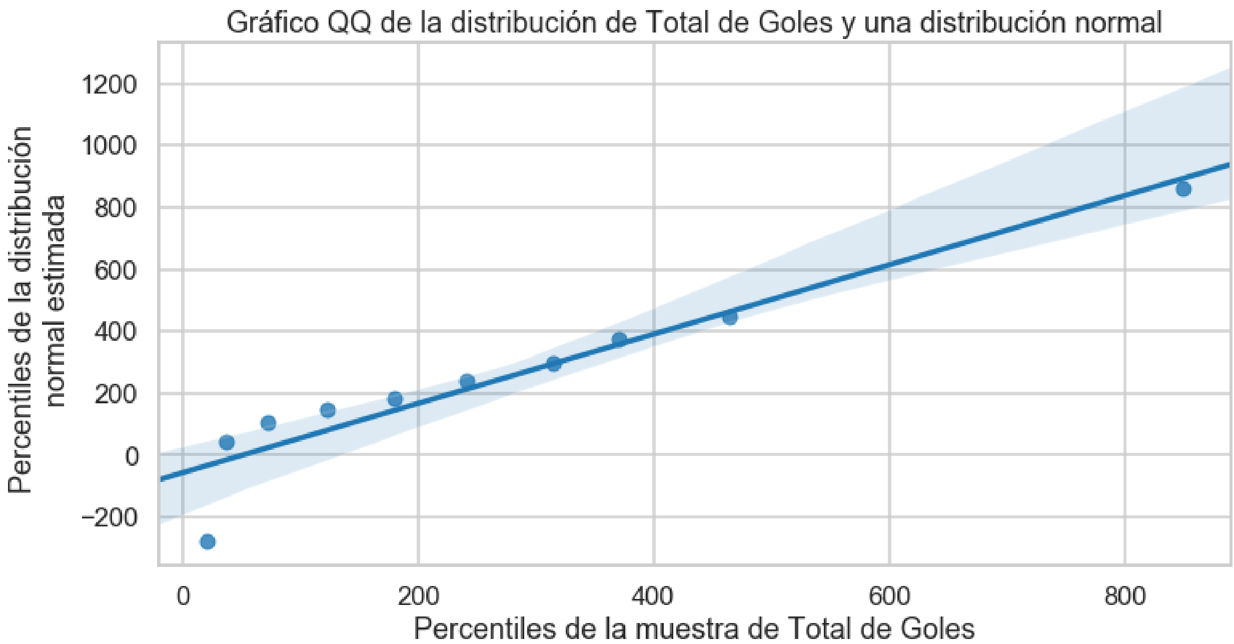
```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-92-84ae87919403> in <module>
     11 plt.figure(figsize=(12,6))
     12 
---> 13 sns.regplot(x = way_team_goal_global_sample, y = way_team_goal_global_norm)
     14 
     15 plt.xlabel('Percentiles de la muestra de Total de Goles de Visitante')

NameError: name 'way_team_goal_global_sample' is not defined


<Figure size 864x432 with 0 Axes>
```

In [93]:

```
#Gráfico QQ para pf_identity Global:

loc, scale = stats.norm.fit(total_equipo["total_equipo_gol"].dropna())
norm_dist = stats.norm(loc, scale)

percs = np.linspace(0,100,10)  # Creamos 10 puntos percentiles igualmente distribuidos entre 0 y 10
0.

total_equipo_gol_global_sample = np.percentile(total_equipo["total_equipo_gol"].dropna(), percs)
total_equipo_gol_global_norm =
np.percentile(norm_dist.rvs(len(total_equipo["total_equipo_gol"].dropna())), percs)

plt.figure(figsize=(12,6))

sns.regplot(x =total_equipo_gol_global_sample, y =total_equipo_gol_global_norm)

plt.xlabel('Percentiles de la muestra de Total de Goles')
plt.ylabel('Percentiles de la distribución \n normal estimada')
plt.title('Gráfico QQ de la distribución de Total de Goles y una distribución normal')
```

```
plt.title('Gráfico QQ de la distribución de Total de Goles y una distribución normal')

x = np.linspace(np.min((total_equipo_gol_global_sample.min(),total_equipo_gol_global_norm.min())), np.max((total_equipo_gol_global_sample.max(), pf_identity_global_norm.max())))
plt.plot(x,x, color='r', ls="--")

sns.despine()
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-93-3bfa7b112210> in <module>
     17 plt.title('Gráfico QQ de la distribución de Total de Goles y una distribución normal')
     18 
---> 19 x = np.linspace(np.min((total_equipo_gol_global_sample.min(),total_equipo_gol_global_norm.min())), np.max((total_equipo_gol_global_sample.max(), pf_identity_global_norm.max())))
     20 plt.plot(x,x, color='r', ls="--")
     21 

NameError: name 'pf_identity_global_norm' is not defined
```



> Ejercicio 9

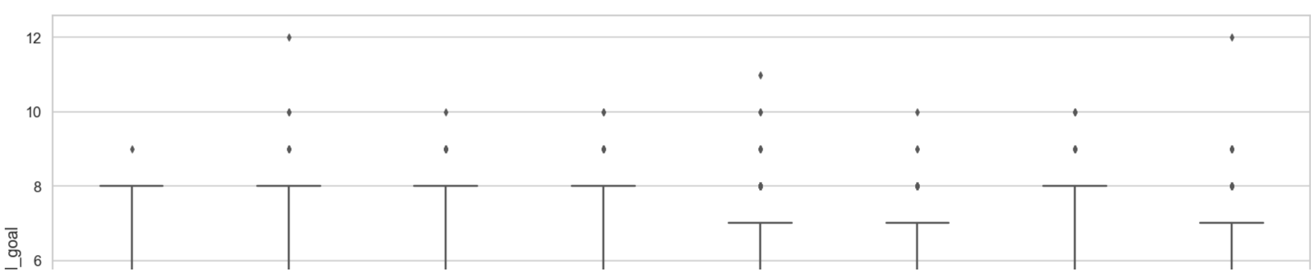
Boxplot de Goles por Temporada

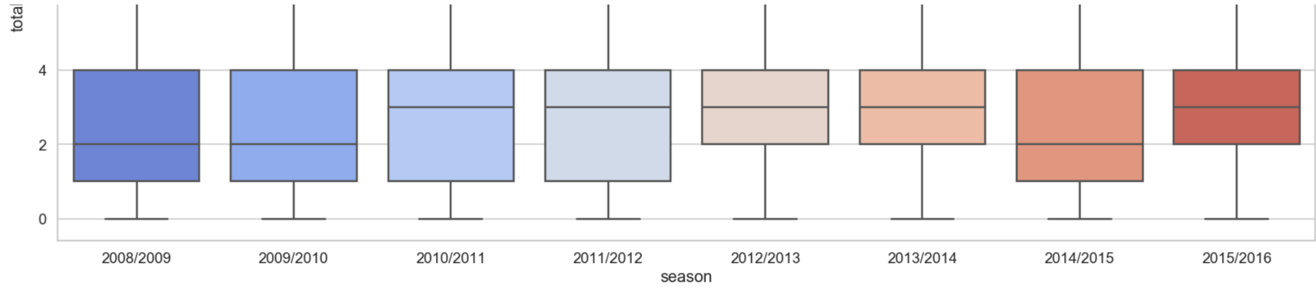
In [94]:

```
plt.figure(figsize=(25,10))

temporada=df_match["season"]
total=df_match["total_goal"]

sns.boxplot(x=temporada, y=total, palette="coolwarm")
```

Out[94]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x19dad3e6f60>
```





In [ ]:

```
Interpretacion a definir
ver que sucedio en las temporads 2009 y 2010
```

> Ejercicio 10

Resumen de Goles convertidos por Temporada: Total, Local y Visitante

In [95]:

```
goles_por_temporada = df_match[["season", "home_team_goal", "away_team_goal", "total_goal"]].groupb
y("season").sum()
goles_por_temporada.sort_values(by = 'season', ascending = False)
```

Out[95]:

season	home_team_goal	away_team_goal	total_goal
2015/2016	5135	4027	9162
2014/2015	5055	3842	8897
2013/2014	4787	3602	8389
2012/2013	5053	3986	9039
2011/2012	5064	3683	8747
2010/2011	5048	3701	8749
2009/2010	4978	3654	8632
2008/2009	5007	3665	8672

In [102]:

```
goles_por_temporada = df_match[["season", "home_team_goal", "away_team_goal", "total_goal"]].groupb
y("season").sum()
goles_por_temporada.sort_values(by = 'total_goal', ascending = False)
```

Out[102]:

season	home_team_goal	away_team_goal	total_goal
2015/2016	5135	4027	9162
2012/2013	5053	3986	9039
2014/2015	5055	3842	8897
2010/2011	5048	3701	8749
2011/2012	5064	3683	8747
2008/2009	5007	3665	8672
2009/2010	4978	3654	8632
2013/2014	4787	3602	8389

> Ejercicio 11

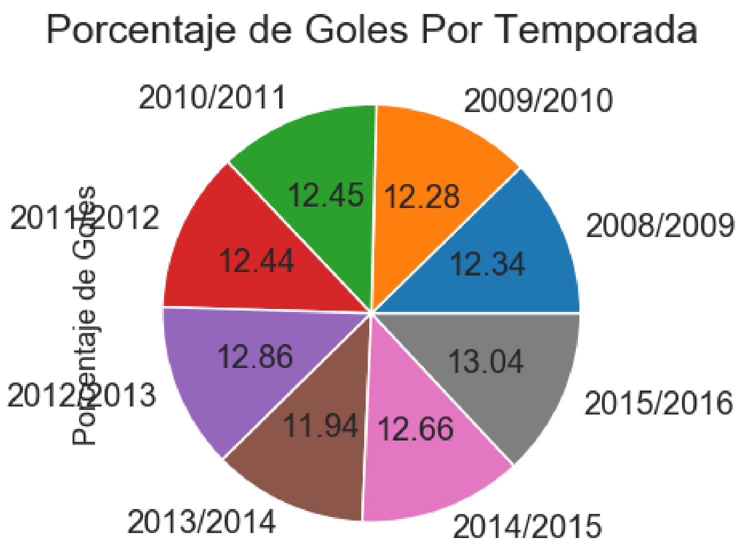
Proporciones de los resultados de los partidos

In [100]:

```
goles_por_temporada["goles_local"] = df_match[["season", "home_team_goal"]].groupby("season").sum()
goles_por_temporada["goles_visitante"] = df_match[["season", "away_team_goal"]].groupby("season").s
um()
goles_por_temporada["goles_total"] = df_match[["season", "total_goal"]].groupby("season").sum()



# Calculating average win percentage for each side:
result_avg_prop = pd.DataFrame((goles_por_temporada["goles_local"]+
goles_por_temporada["goles_visitante"] + goles_por_temporada["goles_total"]) / 3,
                                columns = ['Porcentaje de Goles'])


# Plots average win percentage as a pie chart.
ax = result_avg_prop.plot(kind='pie', figsize =[6,6],autopct='%.2f', y='Porcentaje de Goles', fonts
ize =20,
                           legend = False)
ax.set_title('Porcentaje de Goles Por Temporada', size=25)
#result_avg_prop
```

Out[100]:

```
Text(0.5, 1.0, 'Porcentaje de Goles Por Temporada')
```



Interpretacion
Lo que podemos observar con el grafico de Barras, que existe una leve diferencia entre los totales de Goles por temporada. La temporada con mas Goles corresponde a la Temporada 2015/2016 y la temporada con menor cantidad de Goles, corresponde a la temporada 2013/2014. Si observamos el punto 10, y la correspondiente tabla, se valida lo que se visualiza con el grafico del Ejercicio 11.

In [ ]:

In [ ]: