Data Analysis with the European Soccer Database

Import Libraries

In [2]:
```
import sqlite3
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
from sklearn.cluster import KMeans
from sklearn.preprocessing import scale
```

Connection Database:

The basic structure of the query is very simple: You define what you want to see after the SELECT, * means all possible columns You choose the table after the FROM You add the conditions for the data you want to use from the table(s) after the WHERE

In [3]:
```
path = "../input/"
database = path + 'database.sqlite'

conn = sqlite3.connect(database)
tables = pd.read_sql("""SELECT *
                        FROM sqlite_master
                        WHERE type='table';""", conn)
print("Connection Successful",conn)
```

```
Connection Successful <sqlite3.Connection object at 0x78b7884193b0>
```

Load and Read the Data

Now, we will need to read the dataset using the commands below.

Note: Make sure Database connection before you run the data load and read code below.

df is a variable pointing to a pandas data frame.

In [4]:
```
df = pd.read_sql_query("SELECT * FROM Player_Attributes", conn)
df
```

Out[4]:

	id	player_fifa_api_id	player_api_id	date	overall_rating	potential	preferred_foot	attacking_work_rate	defensive_work_rate	crossing	finishing	heading_accuracy	short_passing	volleys	dribbling	curve	free_kick_accurac
0	1	218353	505942	2016-02-18 00:00:00	67.0	71.0	right	medium	medium	49.0	44.0	71.0	61.0	44.0	51.0	45.0	39.
1	2	218353	505942	2015-11-19 00:00:00	67.0	71.0	right	medium	medium	49.0	44.0	71.0	61.0	44.0	51.0	45.0	39.
2	3	218353	505942	2015-09-21 00:00:00	62.0	66.0	right	medium	medium	49.0	44.0	71.0	61.0	44.0	51.0	45.0	39.
3	4	218353	505942	2015-03-20 00:00:00	61.0	65.0	right	medium	medium	48.0	43.0	70.0	60.0	43.0	50.0	44.0	38.
4	5	218353	505942	2007-02-22 00:00:00	61.0	65.0	right	medium	medium	48.0	43.0	70.0	60.0	43.0	50.0	44.0	38.
5	6	189615	155782	2016-04-21 00:00:00	74.0	76.0	left	high	medium	80.0	53.0	58.0	71.0	40.0	73.0	70.0	69.
6	7	189615	155782	2016-04-07 00:00:00	74.0	76.0	left	high	medium	80.0	53.0	58.0	71.0	32.0	73.0	70.0	69.
7	8	189615	155782	2016-01-07 00:00:00	73.0	75.0	left	high	medium	79.0	52.0	57.0	70.0	29.0	71.0	68.0	69.
8	9	189615	155782	2015-12-24 00:00:00	73.0	75.0	left	high	medium	79.0	51.0	57.0	70.0	29.0	71.0	68.0	69.
9	10	189615	155782	2015-12-17 00:00:00	73.0	75.0	left	high	medium	79.0	51.0	57.0	70.0	29.0	71.0	68.0	69.
10	11	189615	155782	2015-10-16 00:00:00	73.0	77.0	left	high	medium	79.0	51.0	57.0	70.0	29.0	71.0	68.0	69.
11	12	189615	155782	2015-09-25 00:00:00	74.0	78.0	left	high	medium	79.0	51.0	57.0	70.0	29.0	71.0	68.0	69.
12	13	189615	155782	2015-09-21 00:00:00	73.0	77.0	left	medium	medium	79.0	51.0	57.0	70.0	29.0	67.0	68.0	69.
13	14	189615	155782	2015-01-09 00:00:00	71.0	75.0	left	medium	medium	78.0	50.0	56.0	69.0	28.0	66.0	67.0	68.
14	15	189615	155782	2014-12-05 00:00:00	71.0	77.0	left	medium	medium	78.0	50.0	56.0	69.0	28.0	66.0	67.0	68.
15	16	189615	155782	2014-11-07	71.0	77.0	left	medium	medium	78.0	50.0	56.0	69.0	28.0	66.0	67.0	68.

	id	player_fifa_api_id	player_api_id	date	overall_rating	potential	preferred_foot	attacking_work_rate	defensive_work_rate	crossing	finishing	heading_accuracy	short_passing	volleys	dribbling	curve	free_kick_accuracy
				00:00:00													
16	17	189615	155782	2014-09-18 00:00:00	70.0	77.0	left	medium	medium	77.0	50.0	51.0	67.0	28.0	66.0	67.0	68.
17	18	189615	155782	2014-05-02 00:00:00	70.0	79.0	left	medium	medium	77.0	50.0	51.0	67.0	28.0	66.0	67.0	68.
18	19	189615	155782	2014-04-04 00:00:00	70.0	79.0	left	medium	medium	77.0	50.0	51.0	67.0	28.0	66.0	67.0	68.
19	20	189615	155782	2014-03-14 00:00:00	70.0	79.0	left	medium	medium	77.0	50.0	51.0	67.0	28.0	66.0	66.0	68.
20	21	189615	155782	2013-12-13 00:00:00	70.0	79.0	left	medium	medium	77.0	50.0	51.0	67.0	28.0	66.0	66.0	68.
21	22	189615	155782	2013-11-08 00:00:00	70.0	79.0	left	medium	medium	77.0	50.0	51.0	67.0	28.0	66.0	66.0	68.
22	23	189615	155782	2013-10-04 00:00:00	69.0	79.0	left	medium	medium	77.0	50.0	50.0	64.0	28.0	66.0	66.0	68.
23	24	189615	155782	2013-09-20 00:00:00	69.0	79.0	left	medium	medium	77.0	50.0	50.0	64.0	28.0	66.0	66.0	68.
24	25	189615	155782	2013-05-03 00:00:00	69.0	80.0	left	medium	medium	77.0	50.0	50.0	64.0	28.0	65.0	65.0	68.
25	26	189615	155782	2013-03-22 00:00:00	69.0	80.0	left	medium	medium	77.0	50.0	50.0	64.0	28.0	65.0	65.0	68.
26	27	189615	155782	2013-03-15 00:00:00	69.0	80.0	left	medium	medium	77.0	50.0	50.0	64.0	28.0	65.0	65.0	68.
27	28	189615	155782	2013-02-22 00:00:00	69.0	80.0	left	medium	medium	77.0	50.0	50.0	64.0	28.0	65.0	65.0	68.
28	29	189615	155782	2013-02-15 00:00:00	69.0	80.0	left	medium	medium	77.0	50.0	50.0	64.0	28.0	65.0	65.0	68.
29	30	189615	155782	2012-08-31 00:00:00	68.0	80.0	left	medium	medium	77.0	50.0	50.0	64.0	28.0	64.0	63.0	68.
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
183948	183949	164680	111182	2012-02-22 00:00:00	68.0	76.0	left	medium	medium	73.0	48.0	62.0	66.0	60.0	67.0	67.0	64.

	id	player_fifa_api_id	player_api_id	date	overall_rating	potential	preferred_foot	attacking_work_rate	defensive_work_rate	crossing	finishing	heading_accuracy	short_passing	volleys	dribbling	curve	free_kick_accuracy
183949	183950	164680	111182	2011-08-30 00:00:00	68.0	76.0	left	medium	medium	73.0	48.0	62.0	66.0	60.0	67.0	67.0	64.
183950	183951	164680	111182	2011-02-22 00:00:00	68.0	79.0	left	medium	medium	71.0	48.0	63.0	67.0	52.0	65.0	50.0	52.
183951	183952	164680	111182	2010-08-30 00:00:00	67.0	77.0	left	medium	medium	68.0	48.0	63.0	57.0	48.0	49.0	50.0	52.
183952	183953	164680	111182	2007-02-22 00:00:00	67.0	77.0	left	medium	medium	68.0	48.0	63.0	57.0	48.0	49.0	50.0	52.
183953	183954	111191	36491	2011-02-22 00:00:00	68.0	73.0	left	None	_0	64.0	38.0	71.0	66.0	57.0	65.0	60.0	44.
183954	183955	111191	36491	2010-08-30 00:00:00	68.0	73.0	left	None	_0	64.0	38.0	71.0	67.0	57.0	67.0	60.0	44.
183955	183956	111191	36491	2009-08-30 00:00:00	68.0	74.0	left	None	_0	64.0	38.0	71.0	67.0	57.0	68.0	60.0	44.
183956	183957	111191	36491	2009-02-22 00:00:00	68.0	74.0	left	None	_0	64.0	38.0	71.0	67.0	57.0	68.0	60.0	44.
183957	183958	111191	36491	2008-08-30 00:00:00	67.0	72.0	left	None	_0	62.0	58.0	51.0	67.0	57.0	66.0	60.0	64.
183958	183959	111191	36491	2007-08-30 00:00:00	67.0	72.0	left	None	_0	62.0	51.0	42.0	67.0	57.0	67.0	60.0	64.
183959	183960	111191	36491	2007-02-22 00:00:00	67.0	72.0	left	None	_0	62.0	51.0	42.0	67.0	57.0	67.0	60.0	64.
183960	183961	47058	35506	2011-02-22 00:00:00	67.0	78.0	right	None	_0	48.0	43.0	79.0	59.0	59.0	36.0	29.0	37.
183961	183962	47058	35506	2010-08-30 00:00:00	67.0	78.0	right	None	_0	48.0	43.0	79.0	59.0	59.0	36.0	29.0	37.
183962	183963	47058	35506	2010-02-22 00:00:00	67.0	78.0	right	None	_0	48.0	43.0	79.0	59.0	59.0	36.0	29.0	37.
183963	183964	47058	35506	2009-08-30 00:00:00	70.0	78.0	right	None	_0	48.0	43.0	79.0	59.0	59.0	36.0	29.0	37.
183964	183965	47058	35506	2009-02-22	70.0	78.0	right	None	_0	48.0	43.0	79.0	59.0	59.0	36.0	29.0	37.

	id	player_fifa_api_id	player_api_id	date	overall_rating	potential	preferred_foot	attacking_work_rate	defensive_work_rate	crossing	finishing	heading_accuracy	short_passing	volleys	dribbling	curve	free_kick_accuracy
				00:00:00													
183965	183966	47058	35506	2008-08-30 00:00:00	72.0	78.0	right	None	_0	48.0	43.0	79.0	59.0	59.0	36.0	29.0	37.
183966	183967	47058	35506	2007-08-30 00:00:00	75.0	78.0	right	None	_0	48.0	43.0	79.0	59.0	59.0	36.0	29.0	37.
183967	183968	47058	35506	2007-02-22 00:00:00	78.0	79.0	right	None	_0	38.0	43.0	79.0	58.0	59.0	36.0	29.0	72.
183968	183969	102359	39902	2012-08-31 00:00:00	78.0	78.0	right	medium	low	77.0	70.0	59.0	80.0	77.0	78.0	84.0	78.
183969	183970	102359	39902	2012-02-22 00:00:00	81.0	81.0	right	medium	low	80.0	72.0	59.0	86.0	77.0	80.0	86.0	78.
183970	183971	102359	39902	2011-08-30 00:00:00	81.0	81.0	right	medium	low	80.0	72.0	59.0	86.0	77.0	80.0	86.0	78.
183971	183972	102359	39902	2011-02-22 00:00:00	81.0	85.0	right	medium	low	81.0	78.0	59.0	85.0	77.0	80.0	88.0	78.
183972	183973	102359	39902	2010-08-30 00:00:00	83.0	85.0	right	medium	low	84.0	78.0	59.0	87.0	77.0	84.0	86.0	78.
183973	183974	102359	39902	2009-08-30 00:00:00	83.0	85.0	right	medium	low	84.0	77.0	59.0	89.0	77.0	84.0	86.0	78.
183974	183975	102359	39902	2009-02-22 00:00:00	78.0	80.0	right	medium	low	74.0	76.0	53.0	84.0	77.0	85.0	86.0	74.
183975	183976	102359	39902	2008-08-30 00:00:00	77.0	80.0	right	medium	low	74.0	71.0	53.0	84.0	77.0	85.0	86.0	74.
183976	183977	102359	39902	2007-08-30 00:00:00	78.0	81.0	right	medium	low	74.0	64.0	57.0	86.0	77.0	87.0	86.0	73.
183977	183978	102359	39902	2007-02-22 00:00:00	80.0	81.0	right	medium	low	74.0	68.0	57.0	88.0	77.0	87.0	86.0	53.

183978 rows × 42 columns

Exploring Data

We will start our data exploration by generating simple statistics of the data.

In [5]:
```
df.columns
```

Out[5]:
```
Index(['id', 'player_fifa_api_id', 'player_api_id', 'date', 'overall_rating',
       'potential', 'preferred_foot', 'attacking_work_rate',
       'defensive_work_rate', 'crossing', 'finishing', 'heading_accuracy',
       'short_passing', 'volleys', 'dribbling', 'curve', 'free_kick_accuracy',
       'long_passing', 'ball_control', 'acceleration', 'sprint_speed',
       'agility', 'reactions', 'balance', 'shot_power', 'jumping', 'stamina',
       'strength', 'long_shots', 'aggression', 'interceptions', 'positioning',
       'vision', 'penalties', 'marking', 'standing_tackle', 'sliding_tackle',
       'gk_diving', 'gk_handling', 'gk_kicking', 'gk_positioning',
       'gk_reflexes'],
      dtype='object')
```

- Next will display simple statistics of our dataset. You need to run each cell to make sure you see the outputs.

In [6]:
```
df.describe().transpose()
```

Out[6]:

	count	mean	std	min	25%	50%	75%	max
id	183978.0	91989.500000	53110.018250	1.0	45995.25	91989.5	137983.75	183978.0
player_fifa_api_id	183978.0	165671.524291	53851.094769	2.0	155798.00	183488.0	199848.00	234141.0
player_api_id	183978.0	135900.617324	136927.840510	2625.0	34763.00	77741.0	191080.00	750584.0
overall_rating	183142.0	68.600015	7.041139	33.0	64.00	69.0	73.00	94.0
potential	183142.0	73.460353	6.592271	39.0	69.00	74.0	78.00	97.0
crossing	183142.0	55.086883	17.242135	1.0	45.00	59.0	68.00	95.0
finishing	183142.0	49.921078	19.038705	1.0	34.00	53.0	65.00	97.0
heading_accuracy	183142.0	57.266023	16.488905	1.0	49.00	60.0	68.00	98.0
short_passing	183142.0	62.429672	14.194068	3.0	57.00	65.0	72.00	97.0
volleys	181265.0	49.468436	18.256618	1.0	35.00	52.0	64.00	93.0
dribbling	183142.0	59.175154	17.744688	1.0	52.00	64.0	72.00	97.0
curve	181265.0	52.965675	18.255788	2.0	41.00	56.0	67.00	94.0
free_kick_accuracy	183142.0	49.380950	17.831746	1.0	36.00	50.0	63.00	97.0
long_passing	183142.0	57.069880	14.394464	3.0	49.00	59.0	67.00	97.0
ball_control	183142.0	63.388879	15.196671	5.0	58.00	67.0	73.00	97.0
acceleration	183142.0	67.659357	12.983326	10.0	61.00	69.0	77.00	97.0
sprint_speed	183142.0	68.051244	12.569721	12.0	62.00	69.0	77.00	97.0
agility	181265.0	65.970910	12.954585	11.0	58.00	68.0	75.00	96.0
reactions	183142.0	66.103706	9.155408	17.0	61.00	67.0	72.00	96.0
balance	181265.0	65.189496	13.063188	12.0	58.00	67.0	74.00	96.0
shot_power	183142.0	61.808427	16.135143	2.0	54.00	65.0	73.00	97.0
jumping	181265.0	66.969045	11.006734	14.0	60.00	68.0	74.00	96.0
stamina	183142.0	67.038544	13.165262	10.0	61.00	69.0	76.00	96.0
strength	183142.0	67.424529	12.072280	10.0	60.00	69.0	76.00	96.0
long_shots	183142.0	53.339431	18.367025	1.0	41.00	58.0	67.00	96.0
aggression	183142.0	60.948046	16.089521	6.0	51.00	64.0	73.00	97.0
interceptions	183142.0	52.009271	19.450133	1.0	34.00	57.0	68.00	96.0
positioning	183142.0	55.786504	18.448292	2.0	45.00	60.0	69.00	96.0
vision	181265.0	57.873550	15.144086	1.0	49.00	60.0	69.00	97.0
penalties	183142.0	55.003986	15.546519	2.0	45.00	57.0	67.00	96.0
marking	183142.0	46.772242	21.227667	1.0	25.00	50.0	66.00	96.0
standing_tackle	183142.0	50.351257	21.483706	1.0	29.00	56.0	69.00	95.0
sliding_tackle	181265.0	48.001462	21.598778	2.0	25.00	53.0	67.00	95.0
gk_diving	183142.0	14.704393	16.865467	1.0	7.00	10.0	13.00	94.0

	count	mean	std	min	25%	50%	75%	max
gk_handling	183142.0	16.063612	15.867382	1.0	8.00	11.0	15.00	93.0
gk_kicking	183142.0	20.998362	21.452980	1.0	8.00	12.0	15.00	97.0
gk_positioning	183142.0	16.132154	16.099175	1.0	8.00	11.0	15.00	96.0
gk_reflexes	183142.0	16.441439	17.198155	1.0	8.00	11.0	15.00	96.0

Data Cleaning: Handling Missing Data.

Real data is never clean. We need to make sure we clean the data by converting or getting rid of null or missing values. The next code cell will show you if any of the 183978 rows have null value in one of the 42 columns.

In [7]:
```
df.isnull().any().any(), df.shape
```

Out[7]:
```
(True, (183978, 42))
```

- Now let's try to find how many data points in each column are null.

In [8]:
```
df.isnull().sum(axis=0)
```

Out[8]:
```
id                        0
player_fifa_api_id        0
player_api_id             0
date                      0
overall_rating          836
potential               836
preferred_foot          836
attacking_work_rate    3230
defensive_work_rate     836
crossing                836
finishing               836
heading_accuracy        836
short_passing           836
volleys                2713
dribbling               836
curve                  2713
free_kick_accuracy      836
long_passing            836
ball_control            836
acceleration            836
sprint_speed            836
agility                2713
reactions               836
balance                2713
shot_power              836
jumping                2713
stamina                 836
strength                836
long_shots              836
aggression              836
interceptions           836
positioning             836
vision                 2713
penalties               836
marking                 836
standing_tackle         836
sliding_tackle         2713
gk_diving               836
gk_handling             836
gk_kicking              836
gk_positioning          836
gk_reflexes             836
dtype: int64
```

Fixing Null Values by Deleting Them

- In our next two lines, we will drop the null values by going through each row.

In [9]:
```
rows = df.shape[0]
df=df.dropna()
```

- Now if we check the null values and number of rows, we will see that there are no null values and number of rows decreased accordingly.

In [10]:
```
print(rows)
df.isnull().any().any(), df.shape
```

Out[10]:
```
183978
(False, (180354, 42))
```

- To find exactly how many lines we removed, we need to subtract the current number of rows in our data frame from the original number of rows.

In [11]:
```
rows - df.shape[0]
```

Out[11]:
```
3624
```

Predicting: 'overall_rating' of a player

In [12]:
```
df.isnull().sum()
```

Out[12]:
```
id                    0
player_fifa_api_id    0
player_api_id         0
date                  0
overall_rating        0
potential             0
preferred_foot        0
attacking_work_rate   0
defensive_work_rate   0
crossing              0
finishing             0
heading_accuracy      0
short_passing         0
volleys               0
dribbling             0
curve                 0
free_kick_accuracy    0
long_passing          0
ball_control          0
acceleration          0
sprint_speed          0
agility               0
reactions             0
balance               0
shot_power            0
jumping               0
stamina               0
strength              0
long_shots            0
aggression            0
interceptions         0
positioning           0
vision                0
penalties             0
marking               0
standing_tackle       0
sliding_tackle        0
gk_diving             0
gk_handling           0
gk_kicking            0
gk_positioning        0
gk_reflexes           0
dtype: int64
```

- Now that our data cleaning step is reasonably complete and we can trust and understand the data more, we will start diving into the dataset further.

Let's take a look at top few rows.

We will use the head function for data frames for this task. This gives us every column in every row.

In [13]:
```
df.head(10)
```

Out[13]:

	id	player_fifa_api_id	player_api_id	date	overall_rating	potential	preferred_foot	attacking_work_rate	defensive_work_rate	crossing	finishing	heading_accuracy	short_passing	volleys	dribbling	curve	free_kick_accuracy	long_pa
0	1	218353	505942	2016-02-18 00:00:00	67.0	71.0	right	medium	medium	49.0	44.0	71.0	61.0	44.0	51.0	45.0	39.0	
1	2	218353	505942	2015-11-19 00:00:00	67.0	71.0	right	medium	medium	49.0	44.0	71.0	61.0	44.0	51.0	45.0	39.0	
2	3	218353	505942	2015-09-21 00:00:00	62.0	66.0	right	medium	medium	49.0	44.0	71.0	61.0	44.0	51.0	45.0	39.0	
3	4	218353	505942	2015-03-20 00:00:00	61.0	65.0	right	medium	medium	48.0	43.0	70.0	60.0	43.0	50.0	44.0	38.0	
4	5	218353	505942	2007-02-22 00:00:00	61.0	65.0	right	medium	medium	48.0	43.0	70.0	60.0	43.0	50.0	44.0	38.0	
5	6	189615	155782	2016-04-21 00:00:00	74.0	76.0	left	high	medium	80.0	53.0	58.0	71.0	40.0	73.0	70.0	69.0	
6	7	189615	155782	2016-04-07 00:00:00	74.0	76.0	left	high	medium	80.0	53.0	58.0	71.0	32.0	73.0	70.0	69.0	
7	8	189615	155782	2016-01-07 00:00:00	73.0	75.0	left	high	medium	79.0	52.0	57.0	70.0	29.0	71.0	68.0	69.0	
8	9	189615	155782	2015-12-24 00:00:00	73.0	75.0	left	high	medium	79.0	51.0	57.0	70.0	29.0	71.0	68.0	69.0	
9	10	189615	155782	2015-12-17 00:00:00	73.0	75.0	left	high	medium	79.0	51.0	57.0	70.0	29.0	71.0	68.0	69.0	

- Most of the time, we are only interested in plotting some columns. In that case, we can use the pandas column selection option as follows. Please ignore the first column in the output of the one line code below. It is the unique identifier that acts as an index for the data.

Note: From this point on, we will start referring to the columns as "features" in our description.

In [14]:
```
df[:10][['penalties', 'overall_rating']]
```

Out[14]:

	penalties	overall_rating
0	48.0	67.0
1	48.0	67.0
2	48.0	62.0
3	47.0	61.0
4	47.0	61.0
5	59.0	74.0
6	59.0	74.0
7	59.0	73.0
8	59.0	73.0
9	59.0	73.0

Feature Correlation Analysis

Next, we will check if 'penalties' is correlated to 'overall_rating'. We are using a similar selection operation, this time for all the rows and within the correlation function.

Are these correlated (using Pearson's correlation coefficient) ?

In [15]:
```
df['overall_rating'].corr(df['penalties'])
```

Out[15]:
```
0.39271510791118824
```

Observations:

We see that Pearson's Correlation Coefficient for these two columns is 0.39.

Pearson goes from -1 to +1. A value of 0 would have told there is no correlation, so we shouldn't bother looking at that attribute. A value of 0. shows some correlation, although it could be stronger.

At least, we have these attributes which are slightly correlated.

Next, we will create a list of features that we would like to iterate the same operation on.

Create a list of potential Features that you want to measure correlation with

In [16]:
```
potentialFeatures  = ['acceleration', 'curve', 'free_kick_accuracy', 'ball_control', 'shot_power', 'stamina']
```

- The for loop below prints out the correlation coefficient of "overall_rating" of a player with each feature we added to the list as potential.

In [17]:
```
for fc in potentialFeatures:
    related = df['overall_rating'].corr(df[fc])
    print("%s: %f" % (fc,related))
```

```
acceleration: 0.243998
curve: 0.357566
free_kick_accuracy: 0.349800
ball_control: 0.443991
shot_power: 0.428053
stamina: 0.325606
```

Which features have the highest correlation with overall_rating?

- Looking at the values printed by the previous cell, we notice that two features "ball_control" (0.44) and "shot_power" (0.43) seem to have higher correlation with "overall_rating".

Data Visualization:

- Next we will start plotting the correlation coefficients of each feature with "overall_rating". We start by selecting the columns and creating a list with correlation coefficients, called "correlations".

In [18]:
```
correlation_matrix  = df.corr()
correlation_matrix["overall_rating"].sort_values(ascending=False)
```

Out[18]:
```
overall_rating        1.000000
reactions             0.771856
potential             0.765435
short_passing         0.458243
ball_control          0.443991
long_passing          0.434525
vision                0.431493
shot_power            0.428053
penalties             0.392715
long_shots            0.392668
positioning           0.368978
volleys               0.361739
curve                 0.357566
crossing              0.357320
dribbling             0.354191
free_kick_accuracy    0.349800
finishing             0.330079
stamina               0.325606
aggression            0.322782
strength              0.315684
heading_accuracy      0.313324
jumping               0.258978
sprint_speed          0.253048
interceptions         0.249094
acceleration          0.243998
agility               0.239963
standing_tackle       0.163986
balance               0.160211
marking               0.132185
sliding_tackle        0.128054
gk_kicking            0.028799
gk_diving             0.027675
gk_positioning        0.008029
gk_reflexes           0.007804
gk_handling           0.006717
id                   -0.003738
player_fifa_api_id   -0.278703
player_api_id        -0.328315
Name: overall_rating, dtype: float64
```

In [19]:
```
columns = ['potential',  'crossing', 'finishing', 'heading_accuracy',
       'short_passing', 'volleys', 'dribbling', 'curve', 'free_kick_accuracy',
       'long_passing', 'ball_control', 'acceleration', 'sprint_speed',
       'agility', 'reactions', 'balance', 'shot_power', 'jumping', 'stamina',
       'strength', 'long_shots', 'aggression', 'interceptions', 'positioning',
       'vision', 'penalties', 'marking', 'standing_tackle', 'sliding_tackle',
       'gk_diving', 'gk_handling', 'gk_kicking', 'gk_positioning',
       'gk_reflexes']
```

In [20]:
```
correlations = [ df['overall_rating'].corr(df[f]) for f in columns ]
```

Check length of Columns and Correlations

In [21]:
```
len(columns),len(correlations)
```

Out[21]:
```
(34, 34)
```

Graph or Ploting

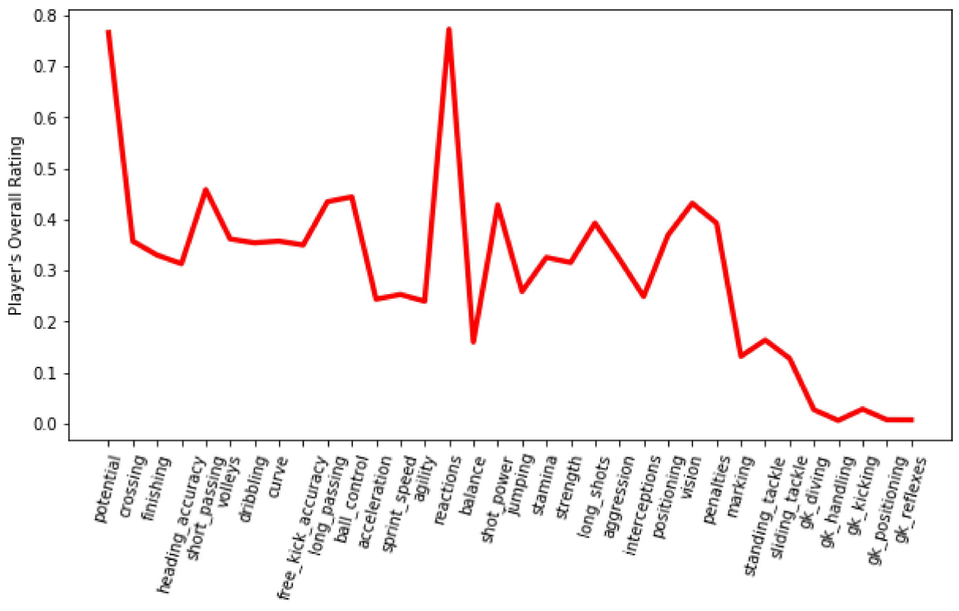
We make sure that the number of selected features and the correlations calculated are the same, e.g., both 34 in this case. Next couple of cells show some lines of code that use pandas plotting functions to create graph of these correlation vealues and column names.

In [22]:
```
def plot_dataframe(df, y_label):
    color='red'
    fig = plt.gcf()
    fig.set_size_inches(10, 5)
    plt.ylabel(y_label)

    ax = df.correlation.plot(linewidth=3.3, color=color)
    ax.set_xticks(df.index)
    ax.set_xticklabels(df.attributes, rotation=75);
    plt.show()
```

In [23]:
```
df1 = pd.DataFrame({'attributes': columns, 'correlation': correlations})
```

In [24]:
```
plot_dataframe(df1, 'Player\'s Overall Rating')
plt.xkcd()
```



Out[24]:
```
<matplotlib.rc_context at 0x78b786399160>
```

Clustering Players into Similar Groups

Until now, we used basic statistics and correlation coefficients to start forming an opinion, but can we do better? What if we took some features and start looking at each player using those features? Can we group similar players based on these features? Let's see how we can do this.

Generally, someone with domain knowledge needs to define which features. We could have also selected some of the features with highest correlation with overall_rating. However, it does not guarantee best outcome always as we are not sure if the top five features are independent. For example, if 4 of the 5 features depend on the remaining 1 feature, taking all 5 does not give new information.

Select Features on Which to Group Players

In [25]:
```
selectfivefeatures = ['gk_kicking', 'potential', 'marking', 'interceptions', 'standing_tackle']
selectfivefeatures
```

Out[25]:
```
['gk_kicking', 'potential', 'marking', 'interceptions', 'standing_tackle']
```

In [26]:
```
df_select = df[selectfivefeatures].copy(deep=True)
```

In [27]:
```
df_select.head()
```

Out[27]:

	gk_kicking	potential	marking	interceptions	standing_tackle
0	10.0	71.0	65.0	70.0	69.0
1	10.0	71.0	65.0	70.0	69.0
2	10.0	66.0	65.0	41.0	66.0
3	9.0	65.0	62.0	40.0	63.0
4	9.0	65.0	62.0	40.0	63.0

Perform KMeans Clustering

Now we will use a machine learning method called KMeans to cluster the values (i.e., player features on gk_kicking, potential, marking, interceptions, and standing_tackle). We will ask for four clusters.

In [28]:
```
# Perform scaling on the dataframe containing the features

data = scale(df_select)

# Define number of clusters
noOfClusters = 4

# Train a model
model = KMeans(init='k-means++', n_clusters=noOfClusters, n_init=20).fit(data)
```

In [29]:
```
print(90*'*')
print("\nCount of players in each cluster")
print(90*'*')

pd.value_counts(model.labels_, sort=False)
```

Out[29]:
```
******************************************************************************************

Count of players in each cluster
******************************************************************************************
0    50477
1    55903
2    23777
3    50197
dtype: int64
```

In [30]:
```
def pd_centers(featuresUsed, centers):
    from itertools import cycle, islice
```

```
    from pandas.tools.plotting import parallel_coordinates
    import matplotlib.pyplot as plt
    import pandas as pd
    import numpy as np

    colNames = list(featuresUsed)
    colNames.append('prediction')
    # Zip with a column called 'prediction' (index)
    Z = [np.append(A, index) for index, A in enumerate(centers)]

    # Convert to pandas for plotting
    P = pd.DataFrame(Z, columns=colNames)
    P['prediction'] = P['prediction'].astype(int)
    return P

def parallel_plot(data):
    from itertools import cycle, islice
    from pandas.tools.plotting import parallel_coordinates
    import matplotlib.pyplot as plt

    my_colors = list(islice(cycle(['b', 'r', 'g', 'y', 'k']), None, len(data)))
    plt.figure(figsize=(15,8)).gca().axes.set_ylim([-2.5,+2.5])
    parallel_coordinates(data, 'prediction', color = my_colors, marker='o')
```

In [31]:
```
Pan = pd_centers(featuresUsed=selectfivefeatures, centers=model.cluster_centers_)
Pan
```

Out[31]:

	gk_kicking	potential	marking	interceptions	standing_tackle	prediction
0	-0.040483	0.704384	1.027772	0.982440	1.030219	0
1	-0.477095	0.105682	-0.947616	-0.975181	-0.914186	1
2	1.920631	0.037763	-1.111727	-0.653150	-1.200879	2
3	-0.337495	-0.843610	0.548359	0.407487	0.550891	3

Visualization of Clusters

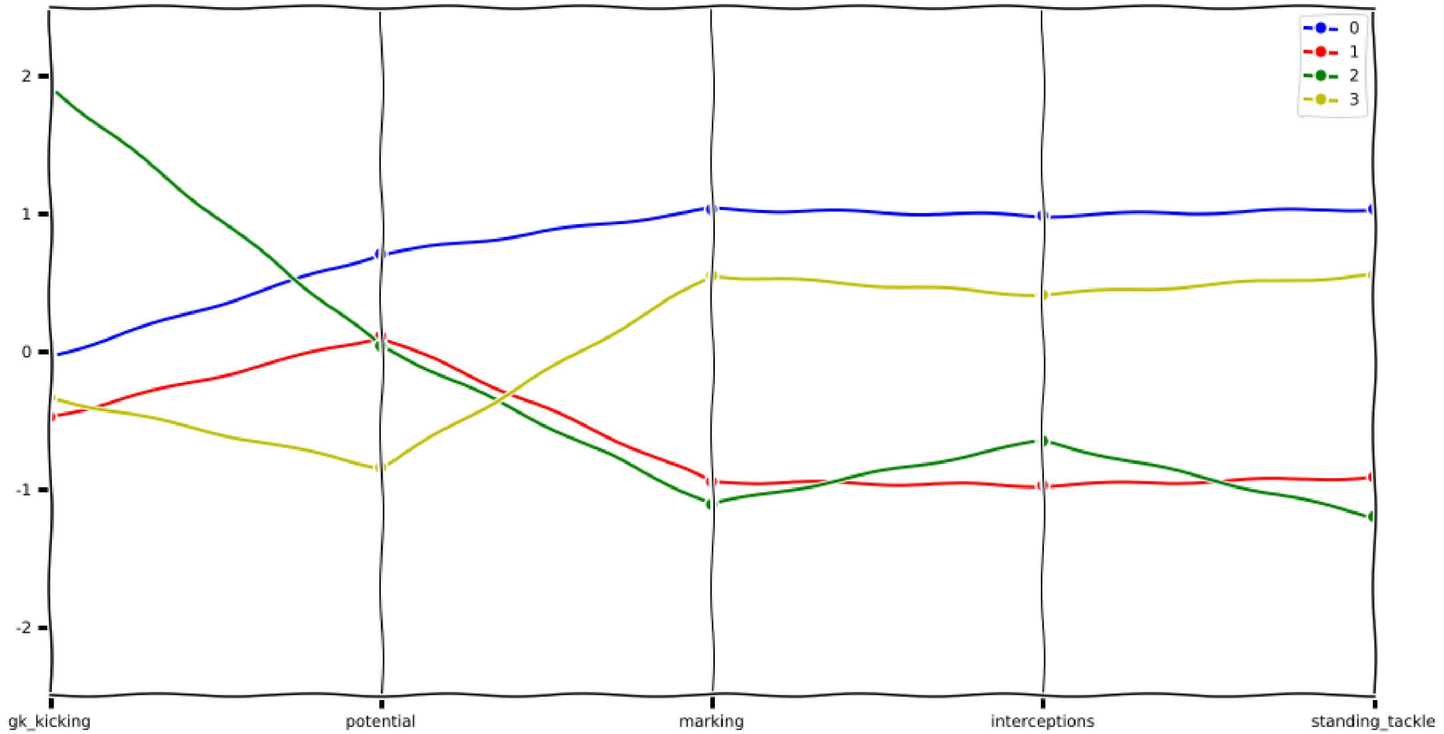
We now have 4 clusters based on the features we selected, we can treat them as profiles for similar groups of players. We can visualize these profiles by plotting the centers for each cluster, i.e., the average values for each feature within the cluster.

In [32]:
```
%matplotlib inline
parallel_plot(Pan)
plt.xkcd()
```

```
/opt/conda/lib/python3.6/site-packages/ipykernel_launcher.py:25: FutureWarning: 'pandas.tools.plotting.parallel_coordinates' is deprecated, import 'pandas.plotting.parallel_coordinates' instead.
```

Out[32]:
```
<matplotlib.rc_context at 0x78b7837b2668>
```

```
/opt/conda/lib/python3.6/site-packages/matplotlib/font_manager.py:1331: UserWarning: findfont: Font family ['xkcd', 'Humor Sans', 'Comic Sans MS'] not found. Falling back to DejaVu Sans
  (prop.get_family(), self.defaultFamily[fontext]))
```


0
1
2
3
2
1
0
-1
-2
gk_kicking
potential
marking
interceptions
standing_tackle