Machine Learning on Kaggle Socccer Dataset (clustering model)

k-means clustering

In [1]:

```
import sqlite3
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.preprocessing import scale
%matplotlib inline
```

Download: Kaggle Socccer Data set (https://www.kaggle.com/hugomathien/soccer)

In [2]:

```
# Create connection and ingest data
cnx = sqlite3.connect('database/database.sqlite')
df = pd.read_sql_query("SELECT pp.player_name, atr.* FROM Player_Attributes atr inne
r join Player pp on pp.player_api_id = atr.player_api_id", cnx)
```

In [3]:

```
# Exploring Data
df.describe(include = "all").transpose()
```

Out[3]:

	count	unique	top	freq	mean	std	min	25%	
player_name	183978	10848	Danilo	108	NaN	NaN	NaN	NaN	
id	183978	NaN	NaN	NaN	91989.5	53110	1	45995.2	91
player_fifa_api_id	183978	NaN	NaN	NaN	165672	53851.1	2	155798	18
player_api_id	183978	NaN	NaN	NaN	135901	136928	2625	34763	7
date	183978	197	2007-02-22 00:00:00	11794	NaN	NaN	NaN	NaN	
overall_rating	183142	NaN	NaN	NaN	68.6	7.04114	33	64	
potential	183142	NaN	NaN	NaN	73.4604	6.59227	39	69	
preferred_foot	183142	2	right	138409	NaN	NaN	NaN	NaN	
attacking_work_rate	180748	8	medium	125070	NaN	NaN	NaN	NaN	
defensive_work_rate	183142	19	medium	130846	NaN	NaN	NaN	NaN	
crossing	183142	NaN	NaN	NaN	55.0869	17.2421	1	45	
finishing	183142	NaN	NaN	NaN	49.9211	19.0387	1	34	
heading_accuracy	183142	NaN	NaN	NaN	57.266	16.4889	1	49	
short_passing	183142	NaN	NaN	NaN	62.4297	14.1941	3	57	
volleys	181265	NaN	NaN	NaN	49.4684	18.2566	1	35	
dribbling	183142	NaN	NaN	NaN	59.1752	17.7447	1	52	
curve	181265	NaN	NaN	NaN	52.9657	18.2558	2	41	
free_kick_accuracy	183142	NaN	NaN	NaN	49.381	17.8317	1	36	
long_passing	183142	NaN	NaN	NaN	57.0699	14.3945	3	49	
ball_control	183142	NaN	NaN	NaN	63.3889	15.1967	5	58	
acceleration	183142	NaN	NaN	NaN	67.6594	12.9833	10	61	
sprint_speed	183142	NaN	NaN	NaN	68.0512	12.5697	12	62	
agility	181265	NaN	NaN	NaN	65.9709	12.9546	11	58	
reactions	183142	NaN	NaN	NaN	66.1037	9.15541	17	61	
balance	181265	NaN	NaN	NaN	65.1895	13.0632	12	58	
shot_power	183142	NaN	NaN	NaN	61.8084	16.1351	2	54	
jumping	181265	NaN	NaN	NaN	66.969	11.0067	14	60	
stamina	183142	NaN	NaN	NaN	67.0385	13.1653	10	61	
strength	183142	NaN	NaN	NaN	67.4245	12.0723	10	60	
long_shots	183142	NaN	NaN	NaN	53.3394	18.367	1	41	
aggression	183142	NaN	NaN	NaN	60.948	16.0895	6	51	
interceptions	183142	NaN	NaN	NaN	52.0093	19.4501	1	34	
positioning	183142	NaN	NaN	NaN	55.7865	18.4483	2	45	
vision	181265	NaN	NaN	NaN	57.8735	15.1441	1	49	

	count	unique	top	freq	mean	std	min	25%
penalties	183142	NaN	NaN	NaN	55.004	15.5465	2	45
marking	183142	NaN	NaN	NaN	46.7722	21.2277	1	25
standing_tackle	183142	NaN	NaN	NaN	50.3513	21.4837	1	29
sliding_tackle	181265	NaN	NaN	NaN	48.0015	21.5988	2	25
gk_diving	183142	NaN	NaN	NaN	14.7044	16.8655	1	7
gk_handling	183142	NaN	NaN	NaN	16.0636	15.8674	1	8
gk_kicking	183142	NaN	NaN	NaN	20.9984	21.453	1	8
gk_positioning	183142	NaN	NaN	NaN	16.1322	16.0992	1	8
gk_reflexes	183142	NaN	NaN	NaN	16.4414	17.1982	1	8

In [4]:

```
# Testing for repeated player_api_id
df.player_api_id.value_counts().head()
```

Out[4]:

```
210278    56
41269     56
42116     55
26472     54
179795    53
Name: player_api_id, dtype: int64
```

In [5]:

```
# Explore for a method to exclude duplicates - column "Date"
df.date.value_counts().head()
```

Out[5]:

```
2007-02-22 00:00:00    11794
2013-09-20 00:00:00     6543
2011-08-30 00:00:00     6525
2015-09-21 00:00:00     6522
2012-08-31 00:00:00     6495
Name: date, dtype: int64
```

In [6]:

```
df = df.loc[df.date=="2015-09-21 00:00:00"]
```

In [7]:

```
# Check for info still duplicated
df.player_api_id.value_counts().head()
```

Out[7]:

```
37254     2
110189    2
163838    2
193866    2
346111    1
Name: player_api_id, dtype: int64
```

In [8]:

```
player_api_id_repeted = [37254, 110189, 163838, 193866]
df.loc[df.player_api_id.isin(player_api_id_repeted)]
```

Out[8]:

	player_name	id	player_fifa_api_id	player_api_id	date	overall_rating	potent
7426	Alex	7427	198033	163838	2015-09-21 00:00:00	73.0	7
7452	Alex	7453	198394	163838	2015-09-21 00:00:00	NaN	N
107760	Mahamadou Samassa	107761	177485	37254	2015-09-21 00:00:00	72.0	7
107787	Mahamadou Samassa	107788	190195	37254	2015-09-21 00:00:00	NaN	N
129776	Nacer Chadli	129777	148581	110189	2015-09-21 00:00:00	79.0	8
129801	Nacer Chadli	129802	200529	110189	2015-09-21 00:00:00	NaN	N
138959	Papy Djilobodji	138960	148337	193866	2015-09-21 00:00:00	78.0	7
138983	Papy Djilobodji	138984	197937	193866	2015-09-21 00:00:00	NaN	N

8 rows × 43 columns

In [9]:

```
# Check for NaN at "Overall rating" column
df.overall_rating.isnull().sum()

# Obs.: It is just these four repeated players with NaN values
```

Out[9]:

```
4
```

In [10]:

```
# Data Cleaning Steps
before_rows = df.shape[0]
df = df.dropna()
after_rows = df.shape[0]
print("It was deleted %d rows" %(before_rows - after_rows))
```

```
It was deleted 4 rows
```

In [11]:

```
# Exploring final dataset
df.describe().transpose()
```

Out[11]:

	count	mean	std	min	25%	50%	
id	6518.0	92627.434489	53436.167338	3.0	46148.75	93393.5	139
player_fifa_api_id	6518.0	182440.111077	42248.975540	2.0	171885.00	191687.5	207
player_api_id	6518.0	186725.952900	166841.362006	2752.0	40445.75	150494.5	276
overall_rating	6518.0	69.975299	6.186119	48.0	66.00	70.0	
potential	6518.0	73.448911	6.051496	51.0	69.00	73.0	
crossing	6518.0	55.005830	18.510649	7.0	45.00	60.0	
finishing	6518.0	49.069040	19.824003	6.0	33.00	53.0	
heading_accuracy	6518.0	57.110003	17.568429	8.0	50.00	60.0	
short_passing	6518.0	63.347346	14.151592	11.0	59.00	66.0	
volleys	6518.0	48.255907	18.697495	5.0	34.00	51.0	
dribbling	6518.0	59.555999	18.625201	6.0	53.00	65.0	
curve	6518.0	52.977447	19.058738	6.0	40.00	56.0	
free_kick_accuracy	6518.0	48.571034	18.513114	4.0	35.00	50.0	
long_passing	6518.0	57.600491	14.788529	10.0	50.00	61.0	
ball_control	6518.0	63.681344	16.064256	10.0	60.00	67.0	
acceleration	6518.0	66.711261	14.006403	13.0	58.00	69.0	
sprint_speed	6518.0	67.154342	13.634979	12.0	60.00	69.0	
agility	6518.0	65.622584	13.958671	16.0	58.00	68.0	
reactions	6518.0	67.059681	8.265205	34.0	62.00	68.0	
balance	6518.0	64.289659	13.844991	12.0	56.00	65.0	
shot_power	6518.0	61.843203	17.201608	10.0	55.00	66.0	
jumping	6518.0	67.616140	11.564535	22.0	60.00	69.0	
stamina	6518.0	66.502301	14.744426	10.0	61.00	69.0	
strength	6518.0	68.425898	11.837629	21.0	61.00	70.0	
long_shots	6518.0	52.958269	19.387968	6.0	40.00	58.0	
aggression	6518.0	61.536207	16.887050	11.0	51.00	65.0	
interceptions	6518.0	51.483431	20.931029	6.0	31.00	57.0	
positioning	6518.0	54.170911	19.798035	3.0	42.00	60.0	
vision	6518.0	56.848880	15.377238	6.0	48.00	59.0	
penalties	6518.0	53.474225	16.204734	9.0	44.00	55.0	
marking	6518.0	47.862535	23.014165	3.0	24.00	54.0	
standing_tackle	6518.0	51.642989	23.093070	6.0	29.00	60.0	
sliding_tackle	6518.0	49.273397	23.145134	5.0	26.00	57.0	
gk_diving	6518.0	16.106321	17.745396	1.0	8.00	11.0	
gk_handling	6518.0	15.736115	16.599638	1.0	8.00	11.0	
gk_kicking	6518.0	15.577785	16.040851	1.0	8.00	11.0	

	count	mean	std	min	25%	50%
gk_positioning	6518.0	15.893372	16.992407	1.0	8.00	11.0
gk_reflexes	6518.0	16.223688	18.214128	1.0	8.00	11.0

In [12]:

```
# Use this command to bring information regarding a player
df.loc[df.player_name=="Cristiano Ronaldo"].transpose()
```

Out[12]:

	33332
player_name	Cristiano Ronaldo
id	33333
player_fifa_api_id	20801
player_api_id	30893
date	2015-09-21 00:00:00
overall_rating	93
potential	93
preferred_foot	right
attacking_work_rate	high
defensive_work_rate	low
crossing	82
finishing	95
heading_accuracy	86
short_passing	81
volleys	87
dribbling	93
curve	88
free_kick_accuracy	77
long_passing	72
ball_control	91
acceleration	91
sprint_speed	93
agility	90
reactions	92
balance	62
shot_power	94
jumping	94
stamina	87
strength	79
long_shots	93
aggression	62
interceptions	29
positioning	93
vision	81
penalties	85
marking	22

	33332
standing_tackle	31
sliding_tackle	23
gk_diving	7
gk_handling	11
gk_kicking	15
gk_positioning	14
gk_reflexes	11

In [13]:

```
# correlation between potencial and overall_rating
df['overall_rating'].corr(df.potential)
```

Out[13]:

```
0.7969820121911912
```

In [14]:

```
# Assign features to a list
cols = ['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 [15]:

```
# check how the features are correlated with the overall rating

correlation = []
for f in cols:
    related = df['overall_rating'].corr(df[f])
    correlation.append(related)
    #print("%s: %f" % (f,related))
```

In [16]:

```
df_corr = pd.DataFrame(
    {'feature': cols,
     'correlation': correlation
    }, columns = [ 'feature', 'correlation' ])
```

In [17]:

```
df_corr.sort_values(by=['correlation'], ascending=False)
```

Out[17]:

	feature	correlation
14	reactions	0.818948
0	potential	0.796982
4	short_passing	0.400583
9	long_passing	0.391820
24	vision	0.390555
10	ball_control	0.357703
16	shot_power	0.334529
7	curve	0.318705
20	long_shots	0.314656
5	volleys	0.295660
1	crossing	0.287956
8	free_kick_accuracy	0.287122
21	aggression	0.283556
6	dribbling	0.271626
25	penalties	0.268231
23	positioning	0.263856
18	stamina	0.250015
3	heading_accuracy	0.247457
2	finishing	0.245785
17	jumping	0.226922
19	strength	0.223814
22	interceptions	0.222483
13	agility	0.196223
12	sprint_speed	0.181173
27	standing_tackle	0.164353
11	acceleration	0.163175
28	sliding_tackle	0.141604
26	marking	0.140271
15	balance	0.082110
30	gk_handling	0.014415
29	gk_diving	0.013075
32	gk_positioning	0.012793
33	gk_reflexes	0.012024
31	gk_kicking	0.010452

Perform k-means clustering

In [18]:

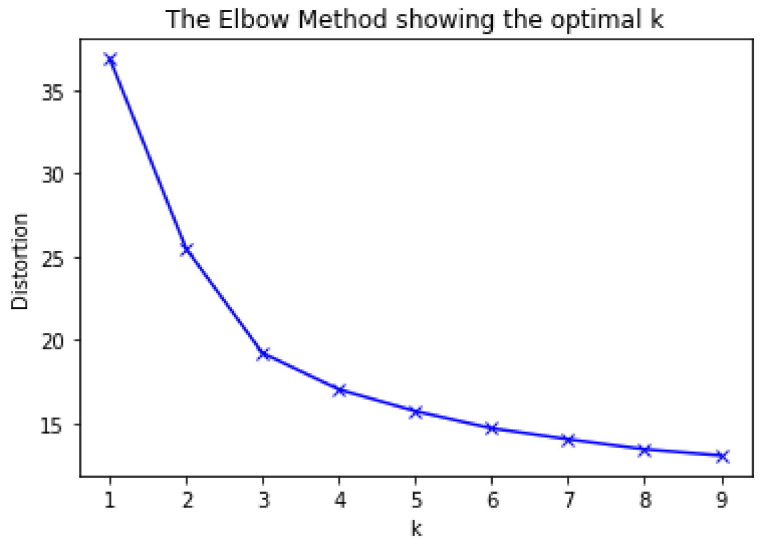
```
# Select Features
select5features = ['reactions', 'crossing', 'marking', 'interceptions', 'gk_handlin
g']

df_select = df[select5features].copy(deep=True)
# In this case, we do not need to scale features
#data = scale(df_select)
```

In [19]:

```
# k means determine k
from scipy.spatial.distance import cdist
distortions = []
K = range(1,10)
for k in K:
    kmeanModel = KMeans(n_clusters=k).fit(df_select)
    kmeanModel.fit(df_select)
    distortions.append(sum(np.min(cdist(df_select, kmeanModel.cluster_centers_, 'euc
lidean'), axis=1)) / df_select.shape[0])
```

In [20]:

```
# Plot the elbow
plt.plot(K, distortions, 'bx-')
plt.xlabel('k')
plt.ylabel('Distortion')
plt.title('The Elbow Method showing the optimal k')
plt.show()
```



In [21]:

```
# Define number of clusters
noOfClusters = 5

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



#Kmean.cluster_centers_
centers = model.cluster_centers_
```

In [22]:

```
centers.shape
```

Out[22]:

```
(5, 5)
```

In [23]:

```
centers
```

Out[23]:

```
array([[68.17803366, 66.7316209 , 37.99734278, 46.54827281, 10.7528786
5],
       [70.05917987, 67.19245107, 67.87138863, 69.86300093, 10.6486486
5],
       [66.60172414, 14.11896552, 13.67931034, 20.28103448, 67.4155172
4],
       [62.52635783, 41.32507987, 67.13658147, 66.61980831, 10.5974440
9],
       [65.81360737, 56.03472714, 22.27356485, 26.87384833, 10.7774627
9]])
```

In [24]:

```
# Understanding the clusters
df_cluster = pd.DataFrame(centers, columns= select5features)
df_cluster.index.name = 'cluster'
df_cluster
```

Out[24]:

cluster	reactions	crossing	marking	interceptions	gk_handling
0	68.178034	66.731621	37.997343	46.548273	10.752879
1	70.059180	67.192451	67.871389	69.863001	10.648649
2	66.601724	14.118966	13.679310	20.281034	67.415517
3	62.526358	41.325080	67.136581	66.619808	10.597444
4	65.813607	56.034727	22.273565	26.873848	10.777463

In [25]:

```
# Number of players in each cluster
pd.value_counts(model.labels_, sort=False)
```

Out[25]:

```
0    1130
4    1412
1    2146
2     580
3    1250
dtype: int64
```

In [26]:

```
model.labels_
```

Out[26]:

```
array([3, 1, 0, ..., 4, 4, 1])
```

In [27]:

```
# Including Prediction column to the initial Dataframe
df['prediction']= model.labels_
```

In [28]:

```
df.head()
```

Out[28]:

	player_name	id	player_fifa_api_id	player_api_id	date	overall_rating	potential	prefe
2	Aaron Appindangoye	3	218353	505942	2015-09-21 00:00:00	62.0	66.0	
12	Aaron Cresswell	13	189615	155782	2015-09-21 00:00:00	73.0	77.0	
40	Aaron Doran	41	186170	162549	2015-09-21 00:00:00	66.0	70.0	
67	Aaron Galindo	68	140161	30572	2015-09-21 00:00:00	69.0	69.0	
88	Aaron Hughes	89	17725	23780	2015-09-21 00:00:00	70.0	70.0	

5 rows × 44 columns

In [29]:

```
# Show features and predicted cluster for some players
player_names = ["Cristiano Ronaldo", "Sergio Ramos", "Gianluigi Buffon", "Miranda",
"Luis Suarez"]

df.loc[df.player_name.isin(player_names)].transpose()
```

Out[29]:

	33332	63847	105984	127009	161329
player_name	Cristiano Ronaldo	Gianluigi Buffon	Luis Suarez	Miranda	Sergio Ramos
id	33333	63848	105985	127010	161330
player_fifa_api_id	20801	1179	176580	168609	155862
player_api_id	30893	30717	40636	19327	30962
date	2015-09-21 00:00:00	2015-09-21 00:00:00	2015-09-21 00:00:00	2015-09-21 00:00:00	2015-09-21 00:00:00
overall_rating	93	84	90	84	87
potential	93	84	90	84	87
preferred_foot	right	right	right	right	right
attacking_work_rate	high	medium	high	medium	high
defensive_work_rate	low	medium	medium	medium	medium
crossing	82	13	77	48	74
finishing	95	15	90	43	60
heading_accuracy	86	13	77	86	86
short_passing	81	37	82	64	76
volleys	87	17	87	51	66
dribbling	93	25	88	49	57
curve	88	20	86	32	73
free_kick_accuracy	77	13	84	39	68
long_passing	72	35	64	65	70
ball_control	91	23	91	64	83
acceleration	91	49	88	74	78
sprint_speed	93	43	78	77	79
agility	90	55	86	67	82
reactions	92	76	91	75	82
balance	62	49	60	58	60
shot_power	94	20	88	70	77
jumping	94	75	69	86	91
stamina	87	39	86	71	81
strength	79	63	76	80	80
long_shots	93	13	85	41	55
aggression	62	38	78	88	83
interceptions	29	20	41	83	88
positioning	93	14	91	43	52
vision	81	50	84	53	63

	33332	63847	105984	127009	161329
penalties	85	22	85	49	68
marking	22	10	30	86	85
standing_tackle	31	11	45	87	89
sliding_tackle	23	11	38	87	90
gk_diving	7	85	27	12	11
gk_handling	11	79	25	6	8
gk_kicking	15	71	31	10	9
gk_positioning	14	89	33	13	7
gk_reflexes	11	83	37	12	11
prediction	0	2	0	3	1

In [30]:

```
# Use this command to select the top 5 players sorted by overall_rating from a speci
fic cluster
prediction_cluster = 4
df.loc[df.prediction==prediction_cluster].sort_values(by=['overall_rating'], ascendi
ng=False).head().transpose()
```

Out[30]:

	102484	183672	131469	58691	160839
player_name	Lionel Messi	Zlatan Ibrahimovic	Neymar	Franck Ribery	Sergio Aguero
id	102485	183673	131470	58692	160840
player_fifa_api_id	158023	41236	190871	156616	153079
player_api_id	30981	35724	19533	30924	37412
date	2015-09-21 00:00:00	2015-09-21 00:00:00	2015-09-21 00:00:00	2015-09-21 00:00:00	2015-09-21 00:00:00
overall_rating	94	89	88	87	87
potential	95	89	93	87	87
preferred_foot	left	right	right	right	right
attacking_work_rate	medium	medium	high	high	high
defensive_work_rate	low	low	medium	medium	medium
crossing	80	76	71	82	70
finishing	93	90	85	78	90
heading_accuracy	71	76	62	41	68
short_passing	88	84	72	87	83
volleys	85	92	83	80	85
dribbling	96	87	94	91	89
curve	89	80	78	83	82
free_kick_accuracy	90	80	78	81	72
long_passing	79	76	72	73	63
ball_control	96	90	91	91	89
acceleration	95	72	91	88	92
sprint_speed	90	74	90	86	86
agility	92	86	92	90	86
reactions	92	85	86	87	87
balance	95	41	84	91	90
shot_power	80	93	77	76	85
jumping	68	72	61	50	76
stamina	76	75	80	64	69
strength	59	93	45	61	71
long_shots	88	88	70	73	81
aggression	48	84	56	52	57
interceptions	22	20	36	36	24
positioning	90	86	87	83	90
vision	90	83	72	88	83

	102484	183672	131469	58691	160839
penalties	74	91	81	80	83
marking	13	15	21	13	13
standing_tackle	23	41	24	25	20
sliding_tackle	21	27	33	26	12
gk_diving	6	13	9	15	13
gk_handling	11	15	9	6	15
gk_kicking	15	10	15	9	6
gk_positioning	14	9	15	7	11
gk_reflexes	8	12	11	10	14
prediction	4	4	4	4	4

In [ ]: