Estimate Rating of a Player

Data Import

In [2]:

```
import seaborn as sns
import sqlite3
import pandas as pd
conn = sqlite3.connect('D:/class/M.Tech 2nd/DL/lab/projects/sports/database.sqlite')

player_data = pd.read_sql_query("SELECT * FROM Player_Attributes", conn)
player_data.head()
```

Out[2]:

	id	player_fifa_api_id	player_api_id	date	overall_rating	potential	preferred_foot	attacki
0	1	218353	505942	2016-02-18 00:00:00	67.0	71.0	right	
1	2	218353	505942	2015-11-19 00:00:00	67.0	71.0	right	
2	3	218353	505942	2015-09-21 00:00:00	62.0	66.0	right	
3	4	218353	505942	2015-03-20 00:00:00	61.0	65.0	right	
4	5	218353	505942	2007-02-22 00:00:00	61.0	65.0	right	

5 rows × 42 columns

Data Manipulation

In [3]:

```
player_data.columns
```

Out[3]:

```
Index(['id', 'player_fifa_api_id', 'player_api_id', 'date', 'overall_ratin
g',
       'potential', 'preferred_foot', 'attacking_work_rate',
       'defensive_work_rate', 'crossing', 'finishing', 'heading_accuracy',
       'short_passing', 'volleys', 'dribbling', 'curve', 'free_kick_accurac
y',
       'long_passing', 'ball_control', 'acceleration', 'sprint_speed',
       'agility', 'reactions', 'balance', 'shot_power', 'jumping', 'stamin
a',
       'strength', 'long_shots', 'aggression', 'interceptions', 'positionin
g',
       'vision', 'penalties', 'marking', 'standing_tackle', 'sliding_tackl
e',
       'gk_diving', 'gk_handling', 'gk_kicking', 'gk_positioning',
       'gk_reflexes'],
      dtype='object')
```

In [4]:

```
req_cols = ['overall_rating', 'crossing', 'finishing', 'heading_accuracy','short_passing',
data = player_data[req_cols]
```

In [4]:

```
data.describe()
```

Out[4]:

	overall_rating	crossing	finishing	heading_accuracy	short_passing	
count	183142.000000	183142.000000	183142.000000	183142.000000	183142.000000	181265.
mean	68.600015	55.086883	49.921078	57.266023	62.429672	49.
std	7.041139	17.242135	19.038705	16.488905	14.194068	18.
min	33.000000	1.000000	1.000000	1.000000	3.000000	1.
25%	64.000000	45.000000	34.000000	49.000000	57.000000	35.
50%	69.000000	59.000000	53.000000	60.000000	65.000000	52.
75%	73.000000	68.000000	65.000000	68.000000	72.000000	64.
max	94.000000	95.000000	97.000000	98.000000	97.000000	93.

In [5]:

```
player_data.columns
```

Out[5]:

```
Index(['id', 'player_fifa_api_id', 'player_api_id', 'date', 'overall_ratin
g',
       'potential', 'preferred_foot', 'attacking_work_rate',
       'defensive_work_rate', 'crossing', 'finishing', 'heading_accuracy',
       'short_passing', 'volleys', 'dribbling', 'curve', 'free_kick_accurac
y',
       'long_passing', 'ball_control', 'acceleration', 'sprint_speed',
       'agility', 'reactions', 'balance', 'shot_power', 'jumping', 'stamin
a',
       'strength', 'long_shots', 'aggression', 'interceptions', 'positionin
g',
       'vision', 'penalties', 'marking', 'standing_tackle', 'sliding_tackl
e',
       'gk_diving', 'gk_handling', 'gk_kicking', 'gk_positioning',
       'gk_reflexes'],
      dtype='object')
```

Feature Selection

In [6]:

```
data = player_data.drop(labels = ['id', 'player_fifa_api_id', 'player_api_id', 'date', 'pot
data.fillna(0, inplace=True)
#data.isnull().values.any()
data.corr()
```

Out[6]:

	overall_rating	crossing	finishing	heading_accuracy	short_passing	vol
overall_rating	1.000000	0.407858	0.366591	0.380763	0.523361	0.386
crossing	0.407858	1.000000	0.591967	0.399936	0.800421	0.630
finishing	0.366591	0.591967	1.000000	0.397826	0.596463	0.825
heading_accuracy	0.380763	0.399936	0.397826	1.000000	0.577012	0.397
short_passing	0.523361	0.800421	0.596463	0.577012	1.000000	0.634
volleys	0.386351	0.630650	0.825051	0.397802	0.634077	1.000
dribbling	0.409350	0.817845	0.792043	0.430451	0.799437	0.769
curve	0.387834	0.767877	0.673450	0.333206	0.716161	0.777
free_kick_accuracy	0.387982	0.718605	0.643965	0.336233	0.704785	0.665
long_passing	0.493020	0.698429	0.366116	0.396208	0.812058	0.423
ball_control	0.506064	0.816734	0.729380	0.578213	0.898413	0.734
acceleration	0.375667	0.622267	0.549116	0.259349	0.549445	0.521
sprint_speed	0.388299	0.604275	0.530995	0.322614	0.540011	0.504
agility	0.339832	0.564973	0.516590	0.120136	0.505351	0.616
reactions	0.818373	0.430127	0.390033	0.358330	0.520335	0.419
balance	0.281806	0.499524	0.384047	0.127590	0.466458	0.499
shot_power	0.483631	0.673833	0.736667	0.566895	0.741086	0.731
jumping	0.365525	0.094655	0.068715	0.294716	0.149338	0.203
stamina	0.437102	0.590179	0.379412	0.513719	0.645403	0.399
strength	0.442692	0.009873	0.012439	0.529601	0.182780	0.028
long_shots	0.427327	0.727289	0.812356	0.432465	0.740322	0.792
aggression	0.398161	0.358600	0.084772	0.599519	0.491866	0.157
interceptions	0.303334	0.331699	-0.113877	0.474370	0.450759	-0.017
positioning	0.412695	0.695439	0.805175	0.438174	0.693312	0.747
vision	0.461091	0.667140	0.622862	0.346733	0.734937	0.725
penalties	0.447034	0.592265	0.730417	0.461404	0.634393	0.684
marking	0.190923	0.258325	-0.251646	0.478005	0.373845	-0.136
standing_tackle	0.222180	0.308309	-0.196260	0.497810	0.438007	-0.074
sliding_tackle	0.187744	0.292988	-0.219840	0.444338	0.395919	-0.034
gk_diving	0.055551	-0.576993	-0.460823	-0.633358	-0.646891	-0.468
gk_handling	0.041053	-0.566574	-0.445087	-0.613835	-0.640196	-0.460
gk_kicking	0.057562	-0.327121	-0.271216	-0.365397	-0.376692	-0.292

	overall_rating	crossing	finishing	heading_accuracy	short_passing	vol
gk_positioning	0.041757	-0.568712	-0.450705	-0.613889	-0.641546	-0.46
gk_reflexes	0.040047	-0.573339	-0.453792	-0.618252	-0.644686	-0.46

34 rows × 34 columns

Model Creation

In [7]:

```
from sklearn.cross_validation import train_test_split
feature_cols = ['crossing', 'finishing', 'heading_accuracy', 'short_passing',
       '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']

x = data[feature_cols]
y = data.overall_rating

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.25, random_state =
```

```
E:\soft\anaconda\lib\site-packages\sklearn\cross_validation.py:41: Deprecati
onWarning: This module was deprecated in version 0.18 in favor of the model_
selection module into which all the refactored classes and functions are mov
ed. Also note that the interface of the new CV iterators are different from
that of this module. This module will be removed in 0.20.
  "This module will be removed in 0.20.", DeprecationWarning)
```

In [8]:

```
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()

regressor.fit(x_train, y_train)
```

Out[8]:

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=False)
```

Prediction And Validation

In [9]:

```
predicted_overall_rating = regressor.predict(x_test)
```

In [10]:

```
from sklearn.metrics import mean_squared_error
import numpy as np
msr = mean_squared_error(y_test, predicted_overall_rating)
rmsr = np.sqrt(msr)
print('Mean Squared Error = ', msr)
print('Root Mean Squared Error = ', rmsr)
```

```
Mean Squared Error =  10.962274261905245
Root Mean Squared Error =  3.310932536598299
```