Import Libraries & Dataset

In [3]:
```
import seaborn as sns
import sqlite3
import pandas as pd
conn = sqlite3.connect('C:\\Users\\mallikarjuna.m\\Documents\\Datascience\\Acadgi

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

Out[3]:

	id	player_fifa_api_id	player_api_id	date	overall_rating	potential	preferred_foot	attacking_v
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

In [4]:
```
player_data.columns
```

Out[4]:
```
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')
```

In [5]:
```
req_cols = ['overall_rating', 'crossing', 'finishing', 'heading_accuracy','short_
data = player_data[req_cols]
```

In [6]:
```
data.describe()
```

Out[6]:

	overall_rating	crossing	finishing	heading_accuracy	short_passing	volle
count	183142.000000	183142.000000	183142.000000	183142.000000	183142.000000	181265.0000
mean	68.600015	55.086883	49.921078	57.266023	62.429672	49.4684
std	7.041139	17.242135	19.038705	16.488905	14.194068	18.2566
min	33.000000	1.000000	1.000000	1.000000	3.000000	1.0000
25%	64.000000	45.000000	34.000000	49.000000	57.000000	35.0000
50%	69.000000	59.000000	53.000000	60.000000	65.000000	52.0000
75%	73.000000	68.000000	65.000000	68.000000	72.000000	64.0000
max	94.000000	95.000000	97.000000	98.000000	97.000000	93.0000

8 rows × 34 columns

In [7]:
```
data = player_data.drop(labels = ['id', 'player_fifa_api_id', 'player_api_id', 'd
data.fillna(0, inplace=True)
#data.isnull().values.any()
data.corr()
```

Out[7]:

	overall_rating	crossing	finishing	heading_accuracy	short_passing	volleys
overall_rating	1.000000	0.407858	0.366591	0.380763	0.523361	0.386351
crossing	0.407858	1.000000	0.591967	0.399936	0.800421	0.630650
finishing	0.366591	0.591967	1.000000	0.397826	0.596463	0.825051
heading_accuracy	0.380763	0.399936	0.397826	1.000000	0.577012	0.397802
short_passing	0.523361	0.800421	0.596463	0.577012	1.000000	0.634077
volleys	0.386351	0.630650	0.825051	0.397802	0.634077	1.000000
dribbling	0.409350	0.817845	0.792043	0.430451	0.799437	0.769109
curve	0.387834	0.767877	0.673450	0.333206	0.716161	0.777817
free_kick_accuracy	0.387982	0.718605	0.643965	0.336233	0.704785	0.665599
long_passing	0.493020	0.698429	0.366116	0.396208	0.812058	0.423872
ball_control	0.506064	0.816734	0.729380	0.578213	0.898413	0.734193
acceleration	0.375667	0.622267	0.549116	0.259349	0.549445	0.521969
sprint_speed	0.388299	0.604275	0.530995	0.322614	0.540011	0.504604
agility	0.339832	0.564973	0.516590	0.120136	0.505351	0.616618
reactions	0.818373	0.430127	0.390033	0.358330	0.520335	0.419858
balance	0.281806	0.499524	0.384047	0.127590	0.466458	0.499781
shot_power	0.483631	0.673833	0.736667	0.566895	0.741086	0.731424
jumping	0.365525	0.094655	0.068715	0.294716	0.149338	0.203483
stamina	0.437102	0.590179	0.379412	0.513719	0.645403	0.399289
strength	0.442692	0.009873	0.012439	0.529601	0.182780	0.028764
long_shots	0.427327	0.727289	0.812356	0.432465	0.740322	0.792580
aggression	0.398161	0.358600	0.084772	0.599519	0.491866	0.157824
interceptions	0.303334	0.331699	-0.113877	0.474370	0.450759	-0.017967
positioning	0.412695	0.695439	0.805175	0.438174	0.693312	0.747002
vision	0.461091	0.667140	0.622862	0.346733	0.734937	0.725974
penalties	0.447034	0.592265	0.730417	0.461404	0.634393	0.684004
marking	0.190923	0.258325	-0.251646	0.478005	0.373845	-0.136155
standing_tackle	0.222180	0.308309	-0.196260	0.497810	0.438007	-0.074799
sliding_tackle	0.187744	0.292988	-0.219840	0.444338	0.395919	-0.034479
gk_diving	0.055551	-0.576993	-0.460823	-0.633358	-0.646891	-0.468455
gk_handling	0.041053	-0.566574	-0.445087	-0.613835	-0.640196	-0.460717

	overall_rating	crossing	finishing	heading_accuracy	short_passing	volleys
gk_kicking	0.057562	-0.327121	-0.271216	-0.365397	-0.376692	-0.292742
gk_positioning	0.041757	-0.568712	-0.450705	-0.613889	-0.641546	-0.464414
gk_reflexes	0.040047	-0.573339	-0.453792	-0.618252	-0.644686	-0.466064

34 rows × 34 columns

In [8]:
```
import statsmodels.formula.api as smf
lm = smf.ols(formula = 'overall_rating ~ crossing + finishing + heading_accuracy +
lm.summary()
```

Out[8]:

OLS Regression Results

Dep. Variable:	overall_rating	**R-squared:**	0.847
Model:	OLS	**Adj. R-squared:**	0.847
Method:	Least Squares	**F-statistic:**	3.080e+04
Date:	Mon, 26 Nov 2018	**Prob (F-statistic):**	0.00
Time:	15:36:34	**Log-Likelihood:**	-4.8016e+05
No. Observations:	183978	**AIC:**	9.604e+05
Df Residuals:	183944	**BIC:**	9.607e+05
Df Model:	33		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	3.3250	0.074	44.666	0.000	3.179	3.471
crossing	-0.0126	0.001	-13.086	0.000	-0.015	-0.011
finishing	0.0135	0.001	12.675	0.000	0.011	0.016
heading_accuracy	0.1052	0.001	112.157	0.000	0.103	0.107
short_passing	0.0805	0.002	50.611	0.000	0.077	0.084
volleys	-0.0013	0.001	-1.353	0.176	-0.003	0.001
dribbling	-0.0121	0.001	-8.931	0.000	-0.015	-0.009
curve	0.0116	0.001	12.637	0.000	0.010	0.013
free_kick_accuracy	0.0119	0.001	14.549	0.000	0.010	0.014
long_passing	0.0164	0.001	14.981	0.000	0.014	0.019
ball_control	0.2356	0.002	131.987	0.000	0.232	0.239
acceleration	0.0426	0.002	26.725	0.000	0.039	0.046
sprint_speed	0.0516	0.002	33.917	0.000	0.049	0.055
agility	-0.0079	0.001	-6.951	0.000	-0.010	-0.006
reactions	0.2774	0.001	227.255	0.000	0.275	0.280
balance	0.0101	0.001	11.201	0.000	0.008	0.012
shot_power	0.0234	0.001	22.836	0.000	0.021	0.025
jumping	0.0139	0.001	16.620	0.000	0.012	0.016
stamina	-0.0175	0.001	-18.460	0.000	-0.019	-0.016
strength	0.0725	0.001	76.895	0.000	0.071	0.074
long_shots	-0.0216	0.001	-20.235	0.000	-0.024	-0.019
aggression	0.0098	0.001	12.268	0.000	0.008	0.011

interceptions	0.0177	0.001	20.724	0.000	0.016	0.019
positioning	-0.0215	0.001	-23.506	0.000	-0.023	-0.020
vision	0.0072	0.001	7.413	0.000	0.005	0.009
penalties	0.0168	0.001	19.213	0.000	0.015	0.019
marking	0.0357	0.001	27.455	0.000	0.033	0.038
standing_tackle	0.0132	0.001	8.984	0.000	0.010	0.016
sliding_tackle	-0.0287	0.001	-24.942	0.000	-0.031	-0.026
gk_diving	0.2113	0.002	125.102	0.000	0.208	0.215
gk_handling	0.0600	0.002	26.650	0.000	0.056	0.064
gk_kicking	-0.0293	0.001	-40.786	0.000	-0.031	-0.028
gk_positioning	0.0788	0.002	35.225	0.000	0.074	0.083
gk_reflexes	0.0480	0.002	21.871	0.000	0.044	0.052

Omnibus:	5553.816	**Durbin-Watson:**	0.333
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	12225.076
Skew:	0.176	**Prob(JB):**	0.00
Kurtosis:	4.213	**Cond. No.**	3.06e+03

In [9]:
```
lm2 = smf.ols(formula = 'overall_rating ~ crossing + finishing + heading_accuracy
lm2.summary()
```

Out[9]:

OLS Regression Results

Dep. Variable:	overall_rating	**R-squared:**	0.744
Model:	OLS	**Adj. R-squared:**	0.744
Method:	Least Squares	**F-statistic:**	1.910e+04
Date:	Mon, 26 Nov 2018	**Prob (F-statistic):**	0.00
Time:	15:36:36	**Log-Likelihood:**	-5.2734e+05
No. Observations:	183978	**AIC:**	1.055e+06
Df Residuals:	183949	**BIC:**	1.055e+06
Df Model:	28		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	13.2819	0.088	150.541	0.000	13.109	13.455
crossing	-0.0068	0.001	-5.485	0.000	-0.009	-0.004
finishing	-0.0040	0.001	-2.895	0.004	-0.007	-0.001
heading_accuracy	-0.0322	0.001	-29.281	0.000	-0.034	-0.030
short_passing	0.0150	0.002	7.369	0.000	0.011	0.019
volleys	-0.0129	0.001	-10.662	0.000	-0.015	-0.011
dribbling	-0.0344	0.002	-19.733	0.000	-0.038	-0.031
curve	-0.0077	0.001	-6.550	0.000	-0.010	-0.005
free_kick_accuracy	0.0009	0.001	0.843	0.399	-0.001	0.003
long_passing	0.0971	0.001	70.441	0.000	0.094	0.100
ball_control	0.0850	0.002	37.992	0.000	0.081	0.089
acceleration	0.0334	0.002	16.220	0.000	0.029	0.037
sprint_speed	0.0327	0.002	16.678	0.000	0.029	0.037
agility	0.0029	0.001	1.980	0.048	2.92e-05	0.006
reactions	0.5078	0.001	383.509	0.000	0.505	0.510
balance	-3.106e-05	0.001	-0.027	0.979	-0.002	0.002
shot_power	0.0176	0.001	13.344	0.000	0.015	0.020
jumping	0.0666	0.001	62.749	0.000	0.065	0.069
stamina	-0.0431	0.001	-35.392	0.000	-0.046	-0.041
strength	0.1535	0.001	129.955	0.000	0.151	0.156
long_shots	-0.0140	0.001	-10.205	0.000	-0.017	-0.011
aggression	-0.0204	0.001	-19.906	0.000	-0.022	-0.018

interceptions	0.0183	0.001	17.349	0.000	0.016	0.020
positioning	-0.0285	0.001	-24.681	0.000	-0.031	-0.026
vision	0.0138	0.001	11.025	0.000	0.011	0.016
penalties	0.0308	0.001	27.428	0.000	0.029	0.033
marking	0.0189	0.002	11.282	0.000	0.016	0.022
standing_tackle	-0.0017	0.002	-0.897	0.370	-0.005	0.002
sliding_tackle	-0.0634	0.001	-42.824	0.000	-0.066	-0.061

Omnibus:	12919.221	**Durbin-Watson:**	0.393
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	56697.967
Skew:	0.207	**Prob(JB):**	0.00
Kurtosis:	5.688	**Cond. No.**	2.79e+03

In [10]:
```
from sklearn.model_selection 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, rando
```

In [11]:
```
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()

regressor.fit(x_train, y_train)
```

Out[11]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=False)
```

In [12]:
```
predicted_overall_rating = regressor.predict(x_test)
```

In [13]:
```
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 =  10.962274261905248
Root Mean Squared Error =  3.3109325365982993
```

In [ ]:
```

```