In [1]:
```
import sqlite3
import pandas as pd
from sklearn.tree import DecisionTreeRegressor
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from math import sqrt

%matplotlib notebook
import matplotlib.pyplot as plt

from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.preprocessing import Imputer, StandardScaler
from sklearn.model_selection import train_test_split, GridSearchCV, ShuffleSplit, Rand
from sklearn.pipeline import make_pipeline
```

In [2]:
```
# Create your connection.
cnx = sqlite3.connect('database.sqlite')
```

In [3]:

In [4]:

```
                name
0    sqlite_sequence
1  Player_Attributes
2             Player
3              Match
4             League
5            Country
6               Team
7    Team_Attributes
```

In [5]:

Out[5]:

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

5 rows × 42 columns

In [6]:

In [7]:

Out[7]: (183978, 41)

In [8]:

```
0    67.0
1    67.0
2    62.0
3    61.0
4    61.0
Name: overall_rating, dtype: float64
```

In [9]:

```
#input target function
```

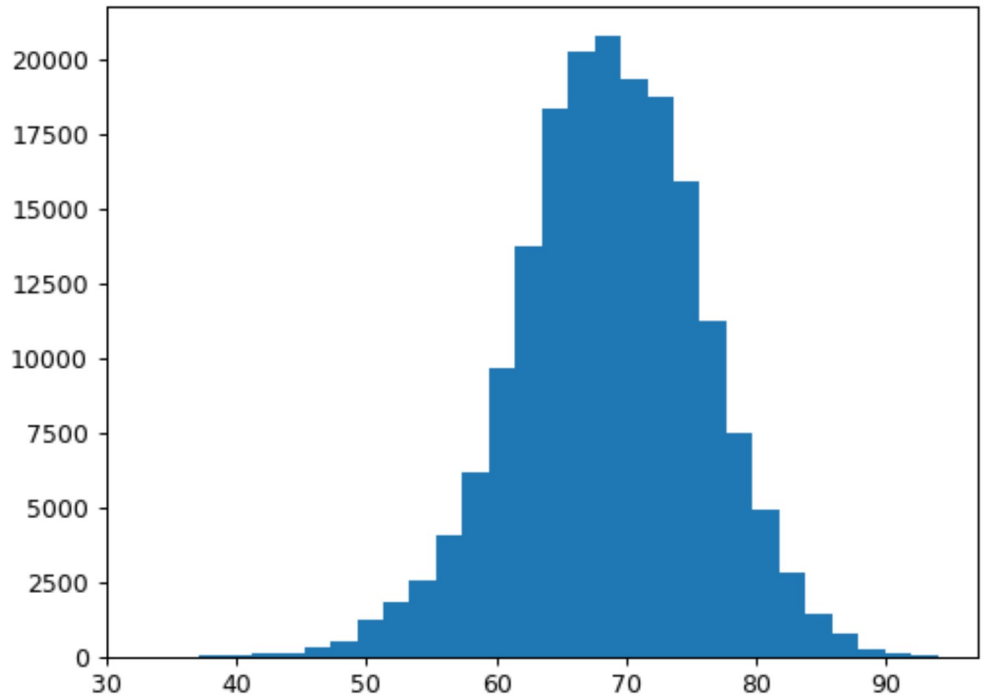
```
836
```

In [10]:

```
count    183142.000000
mean         68.600015
std           7.041139
min          33.000000
25%          64.000000
50%          69.000000
75%          73.000000
max          94.000000
Name: overall_rating, dtype: float64
```

In [11]:

Figure 1




```
C:\Users\HP\Anaconda3\lib\site-packages\numpy\lib\function_base.py:780: RuntimeWar
ning: invalid value encountered in greater_equal
  keep = (tmp_a >= first_edge)
C:\Users\HP\Anaconda3\lib\site-packages\numpy\lib\function_base.py:781: RuntimeWar
ning: invalid value encountered in less_equal
  keep &= (tmp_a <= last_edge)
```

Out[11]:
```
(array([7.0000e+00, 6.0000e+00, 2.0000e+01, 6.5000e+01, 9.4000e+01,
        1.4200e+02, 2.9400e+02, 5.2600e+02, 1.2510e+03, 1.8450e+03,
        2.5780e+03, 4.0870e+03, 6.1890e+03, 9.6500e+03, 1.3745e+04,
        1.8366e+04, 2.0310e+04, 2.0773e+04, 1.9382e+04, 1.8784e+04,
        1.5915e+04, 1.1254e+04, 7.5250e+03, 4.9470e+03, 2.8290e+03,
        1.4590e+03, 7.4800e+02, 2.2800e+02, 8.4000e+01, 3.9000e+01]),
 array([33.        , 35.03333333, 37.06666667, 39.1       , 41.13333333,
        43.16666667, 45.2       , 47.23333333, 49.26666667, 51.3       ,
        53.33333333, 55.36666667, 57.4       , 59.43333333, 61.46666667,
        63.5       , 65.53333333, 67.56666667, 69.6       , 71.63333333,
        73.66666667, 75.7       , 77.73333333, 79.76666667, 81.8       ,
        83.83333333, 85.86666667, 87.9       , 89.93333333, 91.96666667,
        94.        ]),
 <a list of 30 Patch objects>)
```

In [12]:

In [13]:

Out[13]:
```
False
```

In [14]:
```
#data exploration
```

Out[14]:
```
Index(['id', 'player_fifa_api_id', 'player_api_id', 'date', '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 [15]:
```
for col in df.columns:
    unique_cat = len(df[col].unique())
```

```
id--> 183978..int64
player_fifa_api_id--> 11062..int64
player_api_id--> 11060..int64
date--> 197..object
potential--> 57..float64
preferred_foot--> 3..object
attacking_work_rate--> 9..object
defensive_work_rate--> 20..object
crossing--> 96..float64
finishing--> 98..float64
heading_accuracy--> 97..float64
short_passing--> 96..float64
volleys--> 94..float64
dribbling--> 98..float64
curve--> 93..float64
free_kick_accuracy--> 98..float64
long_passing--> 96..float64
ball_control--> 94..float64
acceleration--> 87..float64
sprint_speed--> 86..float64
agility--> 82..float64
reactions--> 79..float64
balance--> 82..float64
shot_power--> 97..float64
jumping--> 80..float64
stamina--> 85..float64
strength--> 83..float64
long_shots--> 97..float64
aggression--> 92..float64
interceptions--> 97..float64
positioning--> 96..float64
vision--> 98..float64
penalties--> 95..float64
marking--> 96..float64
standing_tackle--> 96..float64
sliding_tackle--> 95..float64
gk_diving--> 94..float64
gk_handling--> 91..float64
gk_kicking--> 98..float64
gk_positioning--> 95..float64
gk_reflexes--> 93..float64
```

In [16]:
```
dummy_df = pd.get_dummies(df, columns=['preferred_foot', 'attacking_work_rate', 'defer
```

Out[16]:

	id	player_fifa_api_id	player_api_id	date	potential	crossing	finishing	heading_accuracy	short_passing
0	1	218353	505942	2016-02-18 00:00:00	71.0	49.0	44.0	71.0	61.(
1	2	218353	505942	2015-11-19 00:00:00	71.0	49.0	44.0	71.0	61.(
2	3	218353	505942	2015-09-21 00:00:00	66.0	49.0	44.0	71.0	61.(
3	4	218353	505942	2015-03-20 00:00:00	65.0	48.0	43.0	70.0	60.(
4	5	218353	505942	2007-02-22 00:00:00	65.0	48.0	43.0	70.0	60.(

5 rows × 67 columns

In [17]:

In [19]:
```
#Using Linear Regression

pipe = make_pipeline(StandardScaler(),              #preprocessing(standard scalling)
                     LinearRegression())            #estimator(linear regression)

cv = ShuffleSplit(random_state=0)    #defining type of cross_validation(shuffle splitir

param_grid = {'linearregression__n_jobs': [-1]}      #parameters for model tunning
```

In [21]:

Out[21]:
```
GridSearchCV(cv=ShuffleSplit(n_splits=10, random_state=0, test_size='default',
       train_size=None),
       error_score='raise',
       estimator=Pipeline(memory=None,
     steps=[('standardscaler', StandardScaler(copy=True, with_mean=True, with_std=
True)), ('linearregression', LinearRegression(copy_X=True, fit_intercept=True, n_j
obs=1, normalize=False))]),
       fit_params=None, iid=True, n_jobs=1,
       param_grid={'linearregression__n_jobs': [-1]},
       pre_dispatch='2*n_jobs', refit=True, return_train_score='warn',
       scoring=None, verbose=0)
```

In [22]:

Out[22]:
```
{'linearregression__n_jobs': -1}
```

In [23]:

In [24]:
```
#Using decision tree

pipe = make_pipeline(StandardScaler(),                    #preprocessing
                     DecisionTreeRegressor(criterion='mse', random_state=0))

cv = ShuffleSplit(n_splits=10, random_state=42)         #cross validation

param_grid = {'decisiontreeregressor__max_depth': [3, 5, 7, 9, 13]}
```

In [25]:

Out[25]:
```
GridSearchCV(cv=ShuffleSplit(n_splits=10, random_state=42, test_size='default',
       train_size=None),
       error_score='raise',
       estimator=Pipeline(memory=None,
     steps=[('standardscaler', StandardScaler(copy=True, with_mean=True, with_std=
True)), ('decisiontreeregressor', DecisionTreeRegressor(criterion='mse', max_depth
=None, max_features=None,
           max_leaf_nodes=None, min_impurity_decrease=0.0,
           min_impurity_split=None, min_samples_leaf=1,
           min_samples_split=2, min_weight_fraction_leaf=0.0,
           presort=False, random_state=0, splitter='best'))]),
       fit_params=None, iid=True, n_jobs=1,
       param_grid={'decisiontreeregressor__max_depth': [3, 5, 7, 9, 13]},
       pre_dispatch='2*n_jobs', refit=True, return_train_score='warn',
       scoring=None, verbose=0)
```

In [26]:

Out[26]:
```
{'decisiontreeregressor__max_depth': 13}
```

In [27]:

In [28]:
```
lin_reg = pickle.loads(lin_reg)
```

In [30]:
```
print("""Linear Regressor accuracy is {lin}
DecisionTree Regressor accuracy is {Dec}""".format(lin=lin_reg.score(X_test, y_test),
```

```
Linear Regressor accuracy is 0.8582799243857313
DecisionTree Regressor accuracy is 0.930576264546586
```

By accuracy comparision performed above we can say that Decision Tree regressor gives better result than linear regression model and it can predict the target function with approx 93% accuracy.

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: