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
```

In [2]:
```
pwd
```

Out[2]:
```
'C:\\Users\\Hadi\\Desktop\\data science analystics\\week7 machin learning\\Week-7-MachineLearning\\weather'
```

In [3]:
```
# Create your connection.
cnx = sqlite3.connect('C:\\Users\\Hadi\\Desktop\\data science analystics\\week7 machin learning\\Week-7-MachineLearning\\weather\\database.sqlite')
df = pd.read_sql_query("SELECT * FROM Player_Attributes", cnx)
```

In [4]:
```
df.head()
```

Out[4]:

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

5 rows × 42 columns

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

Out[5]:
```
(183978, 42)
```

In [6]:
```
df.columns
```

Out[6]:
```
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 [7]:
```
features = [
        '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 [8]:
```
target = ['overall_rating']
```

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

In [10]:
```
X = df[features]
```

In [11]:
```
y = df[target]
```

In [12]:
```
X.iloc[2]
```

Out[12]:
```
potential              66.0
crossing               49.0
finishing              44.0
heading_accuracy       71.0
short_passing          61.0
volleys                44.0
dribbling              51.0
curve                  45.0
free_kick_accuracy     39.0
long_passing           64.0
ball_control           49.0
acceleration           60.0
sprint_speed           64.0
agility                59.0
reactions              47.0
balance                65.0
shot_power             55.0
jumping                58.0
stamina                54.0
strength               76.0
long_shots             35.0
aggression             63.0
interceptions          41.0
positioning            45.0
vision                 54.0
penalties              48.0
marking                65.0
standing_tackle        66.0
sliding_tackle         69.0
gk_diving               6.0
gk_handling            11.0
gk_kicking             10.0
gk_positioning          8.0
gk_reflexes             8.0
Name: 2, dtype: float64
```

In [13]:
```
y
```

Out[13]:

	overall_rating
0	67.0
1	67.0
2	62.0
3	61.0
4	61.0
5	74.0
6	74.0
7	73.0
8	73.0
9	73.0
10	73.0
11	74.0
12	73.0
13	71.0
14	71.0
15	71.0
16	70.0
17	70.0
18	70.0
19	70.0
20	70.0
21	70.0
22	69.0
23	69.0
24	69.0
25	69.0
26	69.0
27	69.0
28	69.0
29	68.0
...	...
183933	76.0
183934	75.0
183935	77.0
183936	77.0
183937	63.0
183938	63.0
183939	63.0
183940	63.0
183941	63.0
183942	66.0
183943	66.0
183944	66.0
183945	66.0
183946	66.0
183947	68.0
183948	68.0
183949	68.0
183950	68.0
183951	67.0
183952	67.0
183968	78.0
183969	81.0
183970	81.0
183971	81.0
183972	83.0
183973	83.0
183974	78.0
183975	77.0
183976	78.0
183977	80.0

180354 rows × 1 columns

In [14]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=324)
```

In [15]:
```
regressor = LinearRegression()
regressor.fit(X_train, y_train)
```

Out[15]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,
         normalize=False)
```

In [16]:
```
y_prediction = regressor.predict(X_test)
y_prediction
```

Out[16]:
```
array([[66.51284879],
       [79.77234615],
       [66.57371825],
       ...,
       [69.23780133],
       [64.58351696],
       [73.6881185 ]])
```

In [17]:
```
y_test.describe() # obseved valus or measured values ..
```

Out[17]:

	overall_rating
count	59517.000000
mean	68.635818
std	7.041297
min	33.000000
25%	64.000000
50%	69.000000
75%	73.000000
max	94.000000

In [18]:
```
RMSE = sqrt(mean_squared_error(y_true = y_test, y_pred = y_prediction))
```

In [19]:
```
print(RMSE)
```

```
2.8053030468855208
```

In [21]:
```
regressor = DecisionTreeRegressor(max_depth=20)
regressor.fit(X_train,y_train)# a decision tree regressesor  biulds a model in a top down manner by spliting data set in an attribute so the algorithm choese the attribute which gives  maximam reduction in standard divvation.
```

Out[21]:
```
DecisionTreeRegressor(criterion='mse', max_depth=20, 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=None, splitter='best')
```

In [22]:
```
y_prediction = regressor.predict(X_test)
y_prediction
```

Out[22]:
```
array([63.        , 84.        , 62.38666667, ..., 69.        ,
       62.        , 72.        ])
```

RMSE of 100 for example would be two hight because oure mean is 68.6 and rmse is higher than mean value ? The RMSE captured variance of the predict value from the actual value from our system so it is a measure of how model performs againste operations

In [23]:
```
y_test.describe()
```

Out[23]:

	overall_rating
count	59517.000000
mean	68.635818
std	7.041297
min	33.000000
25%	64.000000
50%	69.000000
75%	73.000000
max	94.000000

In [24]:
```
RMSE = sqrt(mean_squared_error(y_true = y_test, y_pred = y_prediction))
```

In [25]:
```
print(RMSE)
```

```
1.4551544427488428
```