**Predicting Son's height from Father's height using linear regression**[**¶**](#gjdgxs)

we have to predict Son's height from father's height.

**Now import all the necessary libraries**

In [1]:

**import** **pandas** **as** **pd** *# for loading dataset*  
**import** **matplotlib.pyplot** **as** **plt** *# for graphical representation of data*

Let's start with the first step of machine learning development life cycle.

**1) Acquire the data**[**¶**](#30j0zll)

In [2]:

df=pd.read\_csv("father\_son.csv")  
*# In this, we are storing our csv file into dataframe df*  
df *# show dataframe.We can see our dataset below.*

Out[2]:

|  |  |  |
| --- | --- | --- |
|  | **Father** | **Son** |
| **0** | 65.0 | 59.8 |
| **1** | 63.3 | 63.2 |
| **2** | 65.0 | 63.3 |
| **3** | 65.8 | 62.8 |
| **4** | 61.1 | 64.3 |
| **5** | 63.0 | 64.2 |
| **6** | 65.4 | 64.1 |
| **7** | 64.7 | 64.0 |
| **8** | 66.1 | 64.6 |
| **9** | 67.0 | 64.0 |
| **10** | 59.0 | 65.2 |
| **11** | 62.9 | 65.4 |
| **12** | 63.7 | 65.7 |
| **13** | 64.1 | 65.4 |
| **14** | 64.7 | 65.3 |
| **15** | 65.2 | 64.8 |
| **16** | 66.4 | 65.0 |
| **17** | 65.6 | 65.5 |
| **18** | 67.4 | 65.1 |
| **19** | 66.8 | 65.5 |
| **20** | 67.8 | 65.1 |
| **21** | 69.5 | 65.5 |
| **22** | 62.5 | 66.6 |
| **23** | 63.8 | 66.4 |
| **24** | 64.5 | 66.1 |
| **25** | 65.0 | 66.0 |
| **26** | 64.7 | 66.0 |
| **27** | 65.7 | 66.5 |
| **28** | 65.5 | 65.7 |
| **29** | 65.6 | 66.0 |
| **...** | ... | ... |
| **1048** | 70.7 | 72.1 |
| **1049** | 72.4 | 72.3 |
| **1050** | 72.4 | 72.1 |
| **1051** | 67.2 | 72.8 |
| **1052** | 67.8 | 72.8 |
| **1053** | 68.9 | 73.5 |
| **1054** | 70.4 | 73.7 |
| **1055** | 71.2 | 72.8 |
| **1056** | 71.4 | 73.4 |
| **1057** | 71.7 | 73.0 |
| **1058** | 72.6 | 73.2 |
| **1059** | 67.6 | 74.5 |
| **1060** | 68.6 | 73.8 |
| **1061** | 71.0 | 73.8 |
| **1062** | 72.0 | 73.8 |
| **1063** | 75.2 | 73.8 |
| **1064** | 73.1 | 75.6 |
| **1065** | 69.9 | 77.2 |
| **1066** | 65.5 | 60.1 |
| **1067** | 72.6 | 76.8 |
| **1068** | 72.2 | 66.7 |
| **1069** | 63.2 | 58.8 |
| **1070** | 73.3 | 67.9 |
| **1071** | 65.8 | 61.0 |
| **1072** | 67.7 | 59.8 |
| **1073** | 67.0 | 70.8 |
| **1074** | 71.3 | 68.3 |
| **1075** | 71.8 | 69.3 |
| **1076** | 70.7 | 69.3 |
| **1077** | 70.3 | 67.0 |

1078 rows × 2 columns

In [3]:

df.describe()

Out[3]:

|  |  |  |
| --- | --- | --- |
|  | **Father** | **Son** |
| **count** | 1078.000000 | 1078.000000 |
| **mean** | 67.686827 | 68.684230 |
| **std** | 2.745827 | 2.816194 |
| **min** | 59.000000 | 58.500000 |
| **25%** | 65.800000 | 66.900000 |
| **50%** | 67.800000 | 68.600000 |
| **75%** | 69.600000 | 70.500000 |
| **max** | 75.400000 | 78.400000 |

**Train Test Split**[**¶**](#1fob9te)

Now let's split the data into a training set(80%) and a testing set(20%). We will train out model on the training set and then use the test set to evaluate the model.

In [4]:

**from** **sklearn.model\_selection** **import** train\_test\_split

In [5]:

X = df.drop("Son", axis=1)  
y = df['Son']

In [6]:

X\_train,X\_test,y\_train,y\_test= train\_test\_split(X,y, test\_size = 0.2,\  
 random\_state=112)

In [7]:

X\_train

Out[7]:

|  |  |
| --- | --- |
|  | **Father** |
| **136** | 65.7 |
| **1026** | 67.6 |
| **988** | 68.8 |
| **218** | 65.6 |
| **250** | 68.3 |
| **279** | 68.1 |
| **1077** | 70.3 |
| **666** | 68.2 |
| **460** | 65.7 |
| **185** | 70.7 |
| **318** | 72.4 |
| **854** | 73.3 |
| **669** | 63.3 |
| **397** | 69.4 |
| **770** | 69.3 |
| **950** | 68.0 |
| **175** | 72.4 |
| **813** | 68.7 |
| **521** | 67.8 |
| **712** | 68.4 |
| **642** | 71.4 |
| **497** | 69.1 |
| **415** | 72.1 |
| **878** | 65.2 |
| **447** | 66.2 |
| **174** | 71.2 |
| **536** | 63.0 |
| **827** | 68.4 |
| **944** | 66.6 |
| **429** | 65.2 |
| **...** | ... |
| **462** | 67.3 |
| **526** | 69.0 |
| **610** | 67.8 |
| **116** | 67.7 |
| **296** | 65.5 |
| **96** | 69.1 |
| **237** | 62.2 |
| **829** | 68.6 |
| **560** | 70.1 |
| **458** | 66.1 |
| **586** | 72.3 |
| **410** | 72.4 |
| **125** | 68.6 |
| **515** | 66.9 |
| **1017** | 71.5 |
| **997** | 66.0 |
| **123** | 68.6 |
| **51** | 65.3 |
| **176** | 73.0 |
| **797** | 69.9 |
| **134** | 64.9 |
| **872** | 70.6 |
| **766** | 68.3 |
| **304** | 68.2 |
| **916** | 66.5 |
| **684** | 69.7 |
| **1012** | 69.9 |
| **321** | 63.9 |
| **232** | 65.7 |
| **939** | 64.7 |

862 rows × 1 columns

**2) Train the Model**[**¶**](#3znysh7)

In [8]:

**from** **sklearn.linear\_model** **import** LinearRegression

In [9]:

my\_model = LinearRegression()   
result = my\_model.fit(X\_train, y\_train)

**3) Test the Model**[**¶**](#2et92p0)

In [10]:

predictions = result.predict(X\_test)   
predictions

Out[10]:

array([66.32689507, 70.0337033 , 66.87056028, 67.71077014, 71.31873015,  
 67.31537727, 68.05673891, 70.23139974, 65.38783699, 67.16710494,  
 68.94637289, 67.95789069, 68.05673891, 65.28898877, 65.04186822,  
 67.36480138, 68.60040412, 68.40270768, 68.89694878, 68.35328357,  
 69.78658275, 70.28082385, 68.20501124, 68.05673891, 68.30385946,  
 68.60040412, 67.80961836, 67.11768083, 70.08312741, 67.66134603,  
 66.03035041, 68.15558713, 68.89694878, 67.5130737 , 66.22804685,  
 67.80961836, 69.04522111, 68.20501124, 70.37967207, 69.73715864,  
 68.40270768, 69.58888631, 69.44061398, 68.64982823, 69.49003809,  
 68.35328357, 67.31537727, 69.14406933, 69.88543097, 67.71077014,  
 68.84752467, 69.14406933, 68.64982823, 68.30385946, 70.57736851,  
 68.69925234, 67.5130737 , 69.09464522, 69.68773453, 68.25443535,  
 69.09464522, 69.88543097, 68.94637289, 70.97276139, 70.5279444 ,  
 69.19349344, 67.56249781, 72.45548468, 67.61192192, 67.95789069,  
 67.85904247, 66.87056028, 70.92333728, 67.61192192, 71.12103371,  
 67.80961836, 70.97276139, 69.83600686, 70.97276139, 66.4751674 ,  
 67.5130737 , 68.89694878, 68.64982823, 69.24291754, 66.22804685,  
 67.26595316, 69.09464522, 66.62343973, 68.60040412, 67.76019425,  
 66.9694085 , 70.97276139, 67.71077014, 70.42909618, 68.74867645,  
 66.9694085 , 67.31537727, 67.61192192, 69.24291754, 68.94637289,  
 69.63831042, 68.05673891, 69.44061398, 68.45213179, 68.20501124,  
 68.10616302, 69.19349344, 68.35328357, 67.80961836, 68.69925234,  
 66.87056028, 68.0073148 , 68.84752467, 70.13255152, 68.05673891,  
 70.08312741, 65.88207808, 69.39118987, 68.20501124, 68.15558713,  
 68.74867645, 68.5015559 , 67.76019425, 66.77171206, 67.21652905,  
 69.5394622 , 69.29234165, 67.90846658, 71.17045782, 67.95789069,  
 68.995797 , 68.55098001, 67.71077014, 70.62679262, 68.5015559 ,  
 70.67621673, 70.18197563, 68.40270768, 69.73715864, 68.74867645,  
 67.5130737 , 68.64982823, 69.63831042, 67.56249781, 69.98427919,  
 67.5130737 , 68.69925234, 68.5015559 , 68.05673891, 70.72564084,  
 69.5394622 , 66.87056028, 69.14406933, 69.83600686, 71.0716096 ,  
 67.71077014, 68.60040412, 67.71077014, 68.79810056, 68.0073148 ,  
 67.66134603, 68.20501124, 68.20501124, 66.77171206, 71.17045782,  
 67.16710494, 71.0716096 , 69.04522111, 70.42909618, 67.76019425,  
 67.06825672, 69.34176576, 67.61192192, 68.74867645, 69.73715864,  
 70.47852029, 67.01883261, 68.25443535, 69.88543097, 68.25443535,  
 68.15558713, 68.69925234, 66.9694085 , 67.06825672, 67.46364959,  
 70.5279444 , 67.21652905, 66.32689507, 68.60040412, 69.83600686,  
 70.18197563, 71.02218549, 66.62343973, 65.73380575, 70.33024796,  
 68.20501124, 67.85904247, 70.97276139, 68.10616302, 67.95789069,  
 70.33024796, 69.83600686, 70.23139974, 68.74867645, 69.78658275,  
 67.71077014, 66.03035041, 70.13255152, 69.49003809, 67.90846658,  
 68.45213179, 67.5130737 , 66.67286384, 68.60040412, 69.78658275,  
 67.46364959])

**Plot the Line**

In [11]:

plt.scatter(X\_train, y\_train, color ='c')   
plt.plot(X\_test, predictions, color ='k')   
plt.show()

**Measure performance by using r^2 of regression.**

In [12]:

**from** **sklearn.metrics** **import** r2\_score

In [13]:

r2\_score(y\_test,predictions)

Out[13]:

0.321438318349059

**4) Deploy the Model**[**¶**](#tyjcwt)

In [20]:

pred\_new=result.predict([[63]])   
pred\_new

Out[20]:

array([66.32689507])

In [24]:

pred\_new=result.predict([[55]])   
pred\_new

Out[24]:

array([62.37296629])

In [25]:

pred\_new=result.predict([[71.3]])   
pred\_new

Out[25]:

array([70.42909618])