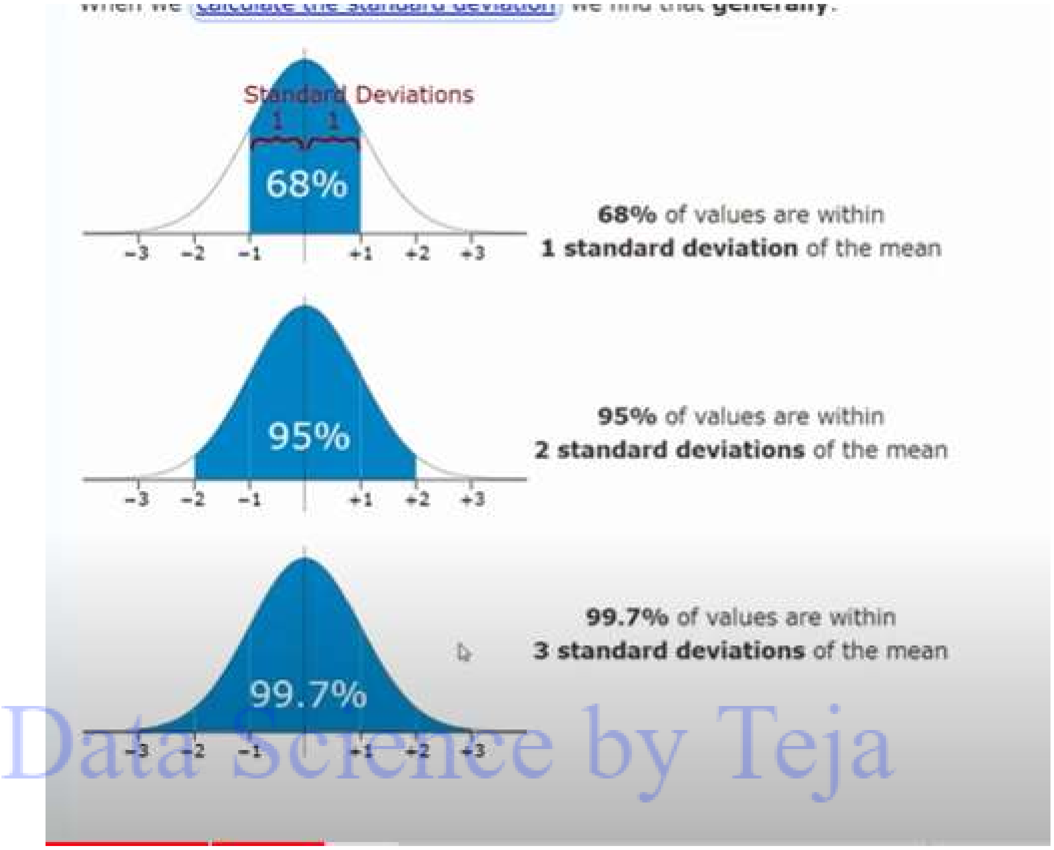
Outliers


When we calculate the standard deviation we find that generally:
Standard Deviations
1
1
68%
-3
-2
-1
+1
+2
+3
68% of values are within
1 standard deviation of the mean
95%
-3
-2
-1
+1
+2
+3
95% of values are within
2 standard deviations of the mean
99.7%
-3
-2
-1
+1
+2
+3
99.7% of values are within
3 standard deviations of the mean

Outlier

An outliers is the datapoint in the dataset but they are distant from all other observation and they are significantly different from the remaining data in the dataset

what are impact of the Outliers ?

- It causes various prombelm in the statitical analysis
- It may causes a significant impact on the mean and the standard deviation

Reasons for the outliers ?

- Data Entry Errors
- Measurement Errors
- Equipment Errors

Types of Outliers ?

Univariant outliers

- identifying the outliers for single variable

Bi-Variant Outliers

- For Some outliers we will find by analying the twi variables

Importing libiraries

In [1]:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

Load Dataset

In [2]:

```
df=pd.read_csv("SOCR-HeightWeight.csv")
df.head(2)
```

Out[2]:

	Index	Height(Inches)	Weight(Pounds)
0	1	65.78331	112.9925
1	2	71.51521	136.4873

In [3]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 25000 entries, 0 to 24999
Data columns (total 3 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   Index           25000 non-null  int64  
 1   Height(Inches)  25000 non-null  float64
 2   Weight(Pounds)  25000 non-null  float64
dtypes: float64(2), int64(1)
memory usage: 586.1 KB
```

In [4]:

```
df.rename(columns={'Height(Inches)':'height','Weight(Pounds)':'weight'},inpla
```

Various way to find the outliers

Z-Score

- |Z Score| >2

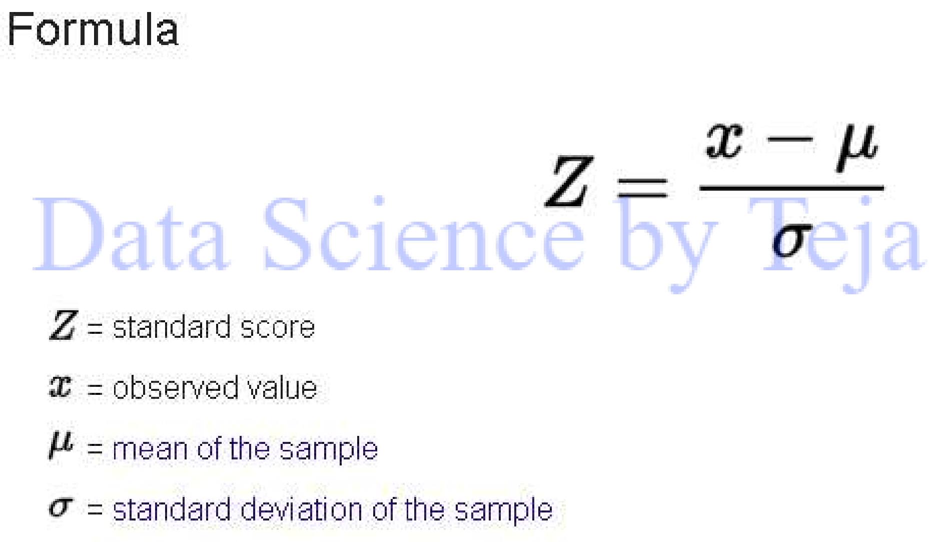
IQR

- Value < Q1 - 1.5[IQR]
- Value > Q3 + 1.5[IQR]

Visualization

- BoxPlot [Uni-var]
- HistoGram [Uni-var]
- ScatterPlot [Bi-var]

Z Score



Formula

$$Z = \frac{x - \mu}{\sigma}$$

Z = standard score

x = observed value

μ = mean of the sample

σ = standard deviation of the sample

```
In [5]: df["Z_score"]=(df.height-df.height.mean())/df.height.std()
        df.head()
```

Out[5]:

	Index	height	weight	Z_score
0	1	65.78331	112.9925	-1.162028
1	2	71.51521	136.4873	1.852099
2	3	69.39874	153.0269	0.739150
3	4	68.21660	142.3354	0.117521
4	5	67.78781	144.2971	-0.107959

In [6]:

```
df[df["Z_score"]>3]
```

Out[6]:

	Index	height	weight	Z_score
138	139	73.90107	151.3913	3.106706
174	175	73.83364	139.2983	3.071248
1162	1163	74.24899	150.2167	3.289660
1383	1384	74.19488	129.0597	3.261206
1893	1894	75.15280	146.9701	3.764929
2395	2396	73.99549	142.9016	3.156357
2481	2482	75.11519	153.9562	3.745152
4191	4192	74.03777	139.5953	3.178590
4508	4509	74.28376	147.7877	3.307944
6627	6628	73.72628	142.8110	3.014792
7269	7270	73.81695	140.0915	3.062471
7839	7840	73.85521	136.0667	3.082590
8472	8473	73.95409	145.2695	3.134586
8828	8829	74.27270	144.6600	3.302128
9225	9226	73.75335	153.1022	3.029027
9492	9493	74.05895	133.8172	3.189727
10330	10331	74.36328	164.6643	3.349759
10635	10636	73.88574	135.9816	3.098644
11173	11174	74.16797	142.7732	3.247055
13681	13682	74.74047	155.5462	3.548105
14063	14064	74.04804	149.6303	3.183990
15209	15210	74.59993	147.0372	3.474202
15966	15967	74.25069	150.0567	3.290554
16145	16146	74.47517	130.9092	3.408597
16385	16386	73.88318	134.2179	3.097298
16752	16753	74.84890	122.1664	3.605123
17079	17080	74.29570	170.5479	3.314222
19005	19006	74.01942	124.2312	3.168940
21949	21950	74.42744	141.7416	3.383498
22471	22472	74.51784	146.9867	3.431035
22769	22770	74.19842	141.6148	3.263068
23039	23040	73.95494	154.3987	3.135033
24801	24802	74.53177	148.9104	3.438360

In [7]:
```
df[df["Z_score"]<-3]
```

Out[7]:

	Index	height	weight	Z_score
412	413	62.01666	109.08480	-3.142725
2651	2652	60.61265	88.04646	-3.881025
3696	3697	61.89340	95.74545	-3.207542
5641	5642	60.86340	106.19390	-3.749168
6405	6406	62.23548	94.80998	-3.027658
6481	6482	61.59011	99.81074	-3.367027
6941	6942	61.40550	119.26520	-3.464104
9876	9877	61.30021	120.88190	-3.519471
10240	10241	61.93152	85.29040	-3.187496
12031	12032	60.86977	108.86330	-3.745819
13971	13972	60.27836	110.11380	-4.056812
14106	14107	61.90725	78.56785	-3.200258
19198	19199	61.82700	100.93910	-3.242458
19750	19751	62.05222	120.43650	-3.124026
20608	20609	60.80620	113.91450	-3.779247
22507	22508	61.57720	96.81420	-3.373816
22945	22946	61.92639	78.01476	-3.190194
24244	24245	62.26498	104.13480	-3.012146

This are the outliers

In [8]:

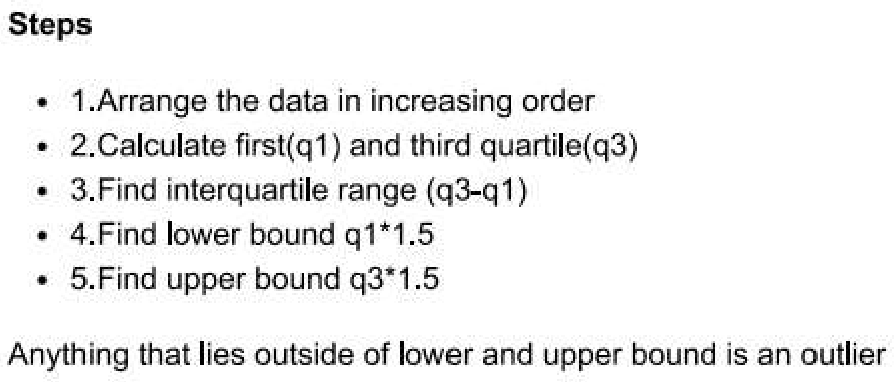
```
df[(df["Z_score"]>3)|(df["Z_score"]<-3)]
```

Out[8]:

	Index	height	weight	Z_score
138	139	73.90107	151.39130	3.106706
174	175	73.83364	139.29830	3.071248
412	413	62.01666	109.08480	-3.142725
1162	1163	74.24899	150.21670	3.289660
1383	1384	74.19488	129.05970	3.261206
1893	1894	75.15280	146.97010	3.764929
2395	2396	73.99549	142.90160	3.156357
2481	2482	75.11519	153.95620	3.745152
2651	2652	60.61265	88.04646	-3.881025
3696	3697	61.89340	95.74545	-3.207542
4191	4192	74.03777	139.59530	3.178590
4508	4509	74.28376	147.78770	3.307944
5641	5642	60.86340	106.19390	-3.749168
6405	6406	62.23548	94.80998	-3.027658
6481	6482	61.59011	99.81074	-3.367027
6627	6628	73.72628	142.81100	3.014792
6941	6942	61.40550	119.26520	-3.464104
7269	7270	73.81695	140.09150	3.062471
7839	7840	73.85521	136.06670	3.082590
8472	8473	73.95409	145.26950	3.134586
8828	8829	74.27270	144.66000	3.302128
9225	9226	73.75335	153.10220	3.029027
9492	9493	74.05895	133.81720	3.189727
9876	9877	61.30021	120.88190	-3.519471
10240	10241	61.93152	85.29040	-3.187496
10330	10331	74.36328	164.66430	3.349759
10635	10636	73.88574	135.98160	3.098644
11173	11174	74.16797	142.77320	3.247055
12031	12032	60.86977	108.86330	-3.745819
13681	13682	74.74047	155.54620	3.548105
13971	13972	60.27836	110.11380	-4.056812
14063	14064	74.04804	149.63030	3.183990
14106	14107	61.90725	78.56785	-3.200258

	Index	height	weight	Z_score
15209	15210	74.59993	147.03720	3.474202
15966	15967	74.25069	150.05670	3.290554
16145	16146	74.47517	130.90920	3.408597
16385	16386	73.88318	134.21790	3.097298
16752	16753	74.84890	122.16640	3.605123
17079	17080	74.29570	170.54790	3.314222
19005	19006	74.01942	124.23120	3.168940
19198	19199	61.82700	100.93910	-3.242458
19750	19751	62.05222	120.43650	-3.124026
20608	20609	60.80620	113.91450	-3.779247
21949	21950	74.42744	141.74160	3.383498
22471	22472	74.51784	146.98670	3.431035
22507	22508	61.57720	96.81420	-3.373816
22769	22770	74.19842	141.61480	3.263068
22945	22946	61.92639	78.01476	-3.190194
23039	23040	73.95494	154.39870	3.135033
24244	24245	62.26498	104.13480	-3.012146
24801	24802	74.53177	148.91040	3.438360

IQR



Steps

- 1.Arrange the data in increasing order
- 2.Calculate first(q1) and third quartile(q3)
- 3.Find interquartile range (q3-q1)
- 4.Find lower bound q1*1.5
- 5.Find upper bound q3*1.5

Anything that lies outside of lower and upper bound is an outlier

In [9]:

```
q1 =np.percentile(sorted(df["height"]),25)
q3=np.percentile(sorted(df["height"]),75)

print("The Q1 is :",q1)
print("The Q3 is :",q3)
```

```
The Q1 is : 66.7043975
The Q3 is : 69.2729575
```

In [10]:

```
iqr=q3-q1
iqr
```

Out[10]:

```
2.568560000000005
```

In [11]:

```
## Finding the lower boundaries and upper boundaries

lower_boundary= q1-(1.5*(iqr))
upper_boundary= q3+(1.5*(iqr))

print("The upper boundary is :",upper_boundary)
print("The lower boundary is :",lower_boundary)
```

```
The upper boundary is : 73.1257975
The lower boundary is : 62.85155749999999
```

This are the outliers

In [12]:

```
df[(df["height"]>upper_boundary)|(df["height"]<lower_boundary)]
```

Out[12]:

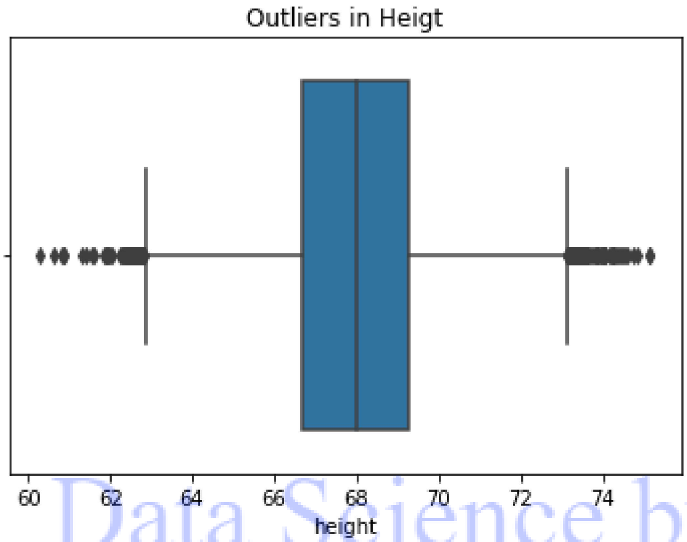
	Index	height	weight	Z_score
138	139	73.90107	151.3913	3.106706
174	175	73.83364	139.2983	3.071248
269	270	73.26872	130.2636	2.774184
412	413	62.01666	109.0848	-3.142725
1133	1134	62.75039	114.4900	-2.756892
...	...	...	...	...
23896	23897	73.38057	154.3189	2.833000
24078	24079	73.22107	136.7360	2.749127
24244	24245	62.26498	104.1348	-3.012146
24475	24476	62.68591	118.6002	-2.790799
24801	24802	74.53177	148.9104	3.438360

167 rows × 4 columns

Visualization

```
In [13]: sns.boxplot(df.height)
         plt.title("Outliers in Heigt")
         plt.show()
```

```
C:\Users\tswar\anaconda3\TSWARUP\lib\site-packages\seaborn\_decorators.py:3
6: FutureWarning: Pass the following variable as a keyword arg: x. From vers
ion 0.12, the only valid positional argument will be `data`, and passing oth
er arguments without an explicit keyword will result in an error or misinter
pretation.
  warnings.warn(
```



Methods to remove Outliers(3'R' Techinque)

- 1 . Remove (Trimming : Remove the outliers from our dataset)
- 2 . Rectify or Replace [Data Entry Error] Ask and Conform from the DataEngineering team
- 3 . Retain (Consider for analysis) Treat them Seperately
- 4 . Censoring (Capping the variable distribution at a max or min value)

Censoring :

- Top and Bottom coding
- Winsorization
- Capping

Note : Outliers should be detected and removed from the only training dataset not for testing dataset

In [17]:

```
height_trim= df[(df["height"]>lower_boundary)&(df["height"]<upper_boundary)]
height_trim
```

Out[17]:

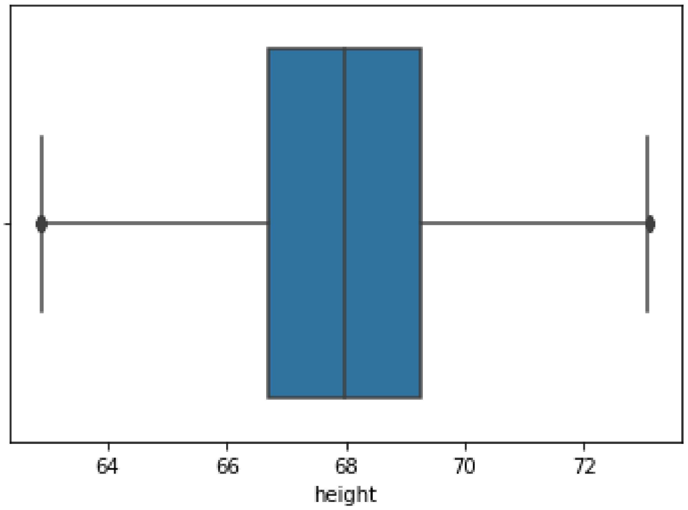
	Index	height	weight	Z_score
0	1	65.78331	112.9925	-1.162028
1	2	71.51521	136.4873	1.852099
2	3	69.39874	153.0269	0.739150
3	4	68.21660	142.3354	0.117521
4	5	67.78781	144.2971	-0.107959
...	...	...	...	...
24995	24996	69.50215	118.0312	0.793529
24996	24997	64.54826	120.1932	-1.811480
24997	24998	64.69855	118.2655	-1.732450
24998	24999	67.52918	132.2682	-0.243960
24999	25000	68.87761	124.8742	0.465113

24833 rows × 4 columns

In [20]:

```
sns.boxplot(height_trim["height"])
plt.show()
```

```
C:\Users\tswar\anaconda3\TSWARUP\lib\site-packages\seaborn\_decorators.py:3
6: FutureWarning: Pass the following variable as a keyword arg: x. From vers
ion 0.12, the only valid positional argument will be `data`, and passing oth
er arguments without an explicit keyword will result in an error or misinter
pretation.
  warnings.warn(
```



1.R[Replace]

In [23]:

```
from feature_engine.outliers import Winsorizer
win = Winsorizer(capping_method='iqr',tail='both',fold=1.5,variables=['height
height_1= win.fit_transform(df[["height"]])
```

We have to what is min and max cap know ?

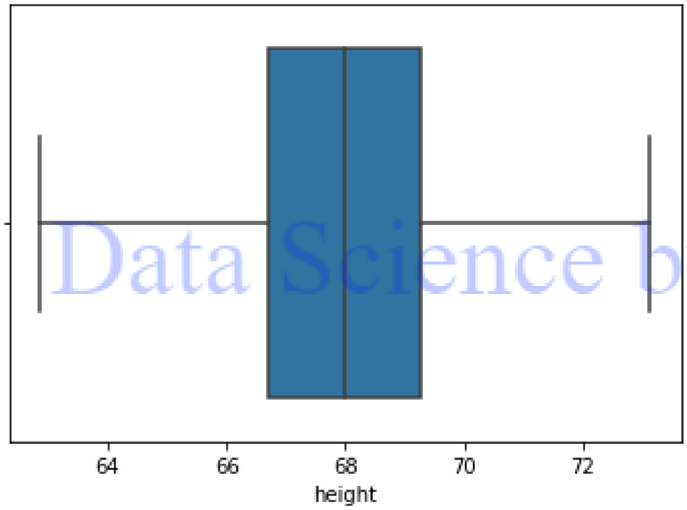
In [24]:

```
print("The minimum cap is :",win.left_tail_caps_)
print("The maximum Cap is  :",win.right_tail_caps_)
```

```
The minimum cap is : {'height': 62.85155749999999}
The maximum Cap is  : {'height': 73.1257975}
```

In [26]:

```
sns.boxplot(height_1.height)
plt.show()
```



2.R Replacing the winsorzing [Min and Max values automatically taken based on the gaussian distribution]

In [29]:

```
from feature_engine.outliers import Winsorizer
win = Winsorizer(capping_method='gaussian',tail='both',fold=1.5,variables=['h
height_2= win.fit_transform(df[["height"]])
```

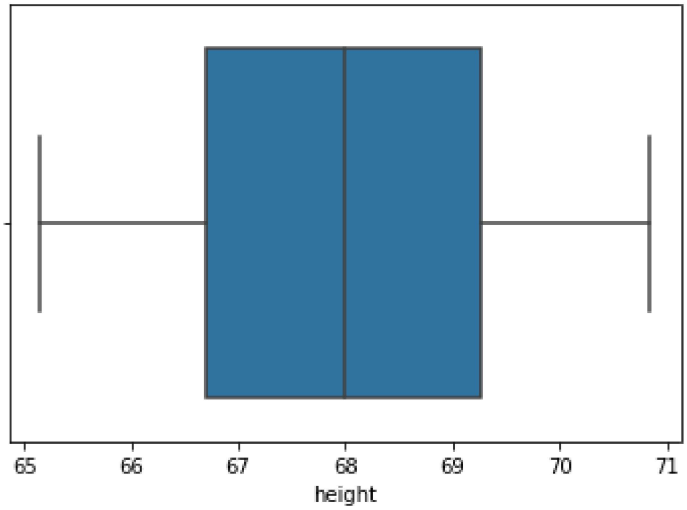
In [28]:

```
print("The minimum cap is :",win.left_tail_caps_)
print("The maximum Cap is  :",win.right_tail_caps_)
```

```
The minimum cap is : {'height': 65.14059543999139}
The maximum Cap is  : {'height': 70.84563175360819}
```

In [30]:
```
sns.boxplot(height_2.height)
plt.show()
```

```
C:\Users\tswar\anaconda3\TSWARUP\lib\site-packages\seaborn\_decorators.py:3
6: FutureWarning: Pass the following variable as a keyword arg: x. From vers
ion 0.12, the only valid positional argument will be `data`, and passing oth
er arguments without an explicit keyword will result in an error or misinter
pretation.
  warnings.warn(
```



3.R Replace Arbitrary Outlier Capper (Min And Max values by replace determine by the user)

In [39]:
```
from feature_engine.outliers import ArbitraryOutlierCapper
capper = ArbitraryOutlierCapper(max_capping_dict={'height':70.8},
                                min_capping_dict={'height':60.8})
height_3= capper.fit_transform(df[["height"]])
```

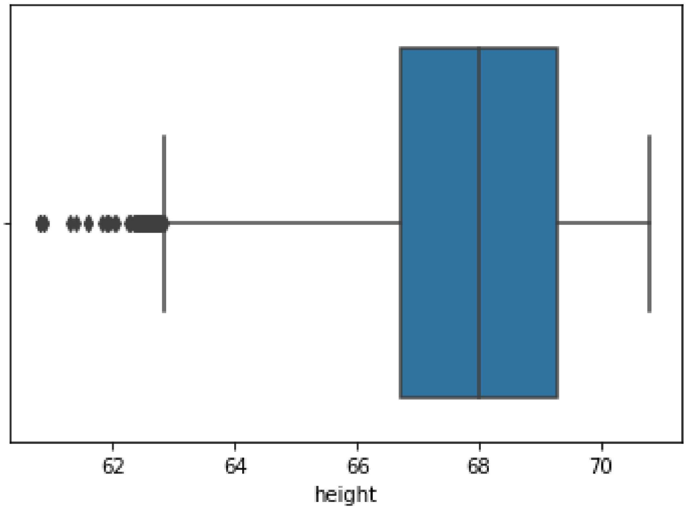
In [40]:
```
print("The minimum cap is :",capper.left_tail_caps_)
print("The maximum Cap is  :",capper.right_tail_caps_)
```

```
The minimum cap is : {'height': 60.8}
The maximum Cap is  : {'height': 70.8}
```

In [41]:

```
sns.boxplot(height_3.height)
plt.show()
```

```
C:\Users\tswar\anaconda3\TSWARUP\lib\site-packages\seaborn\_decorators.py:3
6: FutureWarning: Pass the following variable as a keyword arg: x. From vers
ion 0.12, the only valid positional argument will be `data`, and passing oth
er arguments without an explicit keyword will result in an error or misinter
pretation.
  warnings.warn(
```



In [ ]:

```
```