# DS PRACTICAL 4

 4. Handling Outliers In A Dataset

##  4.1 Outliers Percentile



import pandas as pd

df = pd.read\_csv("heights.csv") df.head()

|  |  |
| --- | --- |
| **name** | **height** |
| **0** mohan | 5.9 |
| **1** maria | 5.2 |
| **2** sakib | 5.1 |
| **3** tao | 5.5 |
| **4** virat | 4.9 |



###  Detect outliers using percentile

max\_thresold = df['height'].quantile(0.95) max\_thresold

 np.float64(9.689999999999998)

df[df['height']>max\_thresold]

##### name height

**9** imran 14.5

min\_thresold = df['height'].quantile(0.05) min\_thresold

 np.float64(3.6050000000000004)

df[df['height']<min\_thresold]

##### name height

**12** yoseph 1.2



###  Remove outliers

df[(df['height']<max\_thresold) & (df['height']>min\_thresold)]

**name height**

1. mohan 5.9
2. maria 5.2
3. sakib 5.1
4. tao 5.5
5. virat 4.9
6. khusbu 5.4
7. dmitry 6.2
8. selena 6.5
9. john 7.1
10. jose 6.1
11. deepika 5.6

**13** binod 5.5

 Banglore Property Prices Dataset

df = pd.read\_csv("bhp.csv") df.head()



|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **location** | **size** | **total\_sqft** | **bath** | **price** | **bhk** | **price\_per\_sqft** |
| **0** Electronic City Phase II | 2 BHK | 1056.0 | 2.0 | 39.07 | 2 | 3699 |
| **1** Chikka Tirupathi | 4 Bedroom | 2600.0 | 5.0 | 120.00 | 4 | 4615 |
| **2** Uttarahalli | 3 BHK | 1440.0 | 2.0 | 62.00 | 3 | 4305 |
| **3** Lingadheeranahalli | 3 BHK | 1521.0 | 3.0 | 95.00 | 3 | 6245 |
| **4** Kothanur | 2 BHK | 1200.0 | 2.0 | 51.00 | 2 | 4250 |

df.shape

 (13200, 7)



df.describe()

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **total\_sqft** | **bath** | **price** | **bhk** | **price\_per\_sqft** |
| **count** | 13200.000000 | 13200.000000 | 13200.000000 | 13200.000000 | 1.320000e+04 |
| **mean** | 1555.302783 | 2.691136 | 112.276178 | 2.800833 | 7.920337e+03 |
| **std** | 1237.323445 | 1.338915 | 149.175995 | 1.292843 | 1.067272e+05 |
| **min** | 1.000000 | 1.000000 | 8.000000 | 1.000000 | 2.670000e+02 |
| **25%** | 1100.000000 | 2.000000 | 50.000000 | 2.000000 | 4.267000e+03 |
| **50%** | 1275.000000 | 2.000000 | 71.850000 | 3.000000 | 5.438000e+03 |
| **75%** | 1672.000000 | 3.000000 | 120.000000 | 3.000000 | 7.317000e+03 |
| **max** | 52272.000000 | 40.000000 | 3600.000000 | 43.000000 | 1.200000e+07 |

###  Samples that are above 99.90% percentile and below 1% percentile rank

min\_thresold, max\_thresold = df.price\_per\_sqft.quantile([0.001, 0.999]) min\_thresold, max\_thresold

(1366.184, 50959.36200000098)

df[df.price\_per\_sqft < min\_thresold]



|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | **location** | **size** | **total\_sqft** | **bath** | **price** | **bhk** | **price\_per\_sqft** |
| **665** | Yelahanka | 3 BHK | 35000.0 | 3.0 | 130.0 | 3 | 371 |
| **798** | other | 4 Bedroom | 10961.0 | 4.0 | 80.0 | 4 | 729 |
| **1867** | other | 3 Bedroom | 52272.0 | 2.0 | 140.0 | 3 | 267 |
| **2392** | other | 4 Bedroom | 2000.0 | 3.0 | 25.0 | 4 | 1250 |
| **3934** | other | 1 BHK | 1500.0 | 1.0 | 19.5 | 1 | 1300 |
| **5343** | other | 9 BHK | 42000.0 | 8.0 | 175.0 | 9 | 416 |
| **5417** | Ulsoor | 4 BHK | 36000.0 | 4.0 | 450.0 | 4 | 1250 |
| **5597** | JP Nagar | 2 BHK | 1100.0 | 1.0 | 15.0 | 2 | 1363 |
| **7166** | Yelahanka | 1 Bedroom | 26136.0 | 1.0 | 150.0 | 1 | 573 |
| **7862** | JP Nagar | 3 BHK | 20000.0 | 3.0 | 175.0 | 3 | 875 |
| **8300** | Kengeri | 1 BHK | 1200.0 | 1.0 | 14.0 | 1 | 1166 |
| **9144** | other | 4 Bedroom | 10961.0 | 4.0 | 80.0 | 4 | 729 |
| **11635** | Begur | 3 BHK | 2400.0 | 3.0 | 12.0 | 3 | 500 |
| **12355** | other | 4 BHK | 16335.0 | 4.0 | 149.0 | 4 | 912 |



df[df.price\_per\_sqft > max\_thresold]

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | **location** | **size** | **total\_sqft** | **bath** | **price** | **bhk** | **price\_per\_sqft** |
| **345** | other | 3 Bedroom | 11.0 | 3.0 | 74.0 | 3 | 672727 |
| **1005** | other | 1 BHK | 15.0 | 1.0 | 30.0 | 1 | 200000 |
| **1106** | other | 5 Bedroom | 24.0 | 2.0 | 150.0 | 5 | 625000 |
| **4044** | Sarjapur Road | 4 Bedroom | 1.0 | 4.0 | 120.0 | 4 | 12000000 |
| **4924** | other | 7 BHK | 5.0 | 7.0 | 115.0 | 7 | 2300000 |
| **5911** | Mysore Road | 1 Bedroom | 45.0 | 1.0 | 23.0 | 1 | 51111 |
| **6356** | Bommenahalli | 4 Bedroom | 2940.0 | 3.0 | 2250.0 | 4 | 76530 |
| **7012** | other | 1 BHK | 650.0 | 1.0 | 500.0 | 1 | 76923 |
| **7575** | other | 1 BHK | 425.0 | 1.0 | 750.0 | 1 | 176470 |
| **7799** | other | 4 BHK | 2000.0 | 3.0 | 1063.0 | 4 | 53150 |
| **8307** | Bannerghatta Road | 5 BHK | 2500.0 | 4.0 | 1400.0 | 5 | 56000 |
| **9436** | Indira Nagar | 4 Bedroom | 2400.0 | 5.0 | 1250.0 | 4 | 52083 |
| **11447** | Whitefield | 4 Bedroom | 60.0 | 4.0 | 218.0 | 4 | 363333 |
| **12328** | other | 4 Bedroom | 4350.0 | 8.0 | 2600.0 | 4 | 59770 |



###  Remove Outliers

df2 = df[(df.price\_per\_sqft<max\_thresold) & (df.price\_per\_sqft>min\_thresold)] df2.shape

 (13172, 7)



df2.describe()

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **total\_sqft** | **bath** | **price** | **bhk** | **price\_per\_sqft** |
| **count** | 13172.000000 | 13172.000000 | 13172.000000 | 13172.000000 | 13172.000000 |
| **mean** | 1537.861049 | 2.690100 | 111.591865 | 2.799651 | 6663.653735 |
| **std** | 967.123711 | 1.337026 | 145.372047 | 1.291130 | 4141.020700 |
| **min** | 250.000000 | 1.000000 | 8.000000 | 1.000000 | 1379.000000 |
| **25%** | 1100.000000 | 2.000000 | 50.000000 | 2.000000 | 4271.000000 |
| **50%** | 1274.500000 | 2.000000 | 71.550000 | 3.000000 | 5438.000000 |
| **75%** | 1670.000000 | 3.000000 | 120.000000 | 3.000000 | 7311.000000 |
| **max** | 30400.000000 | 40.000000 | 3600.000000 | 43.000000 | 50349.000000 |

#  Exercise

Q) Use air bnb new york city data set and remove outliers using percentile based on price per night for a given apartment/home. You can use suitable upper and lower limits on percentile

 based on your intuition. Your goal is to come up with new pandas dataframe that doesn't

have the outliers present in it.

import pandas as pd

df = pd.read\_csv("AB\_NYC\_2019.csv") df.head()



**2** 3647

Castle

THE VILLAGE

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **id name host\_id** | **host\_name** | **neighbourhood\_group** | **neighbourhood** | **la** |
| Clean & quiet  **0** 2539 apt home by the 2787  park | John | Brooklyn | Kensington | 40 |
| **1** 2595 Skylit Midtown 2845 | Jennifer | Manhattan | Midtown | 40 |

OF HARLEM. NEW

YORK !

4632 Elisabeth Manhattan Harlem 40

**3** 3831

**4** 5022

Cozy Entire

Floor of Brownstone

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 4869 | LisaRoxanne | Brooklyn | Clinton Hill | 40 |
| 7192 | Laura | Manhattan | East Harlem | 40 |

Entire Apt: Spacious Studio/Loft by central park





df.price.describe()

|  |  |
| --- | --- |
| count | 48895.000000 |
| mean | 152.720687 |
| std | 240.154170 |
| min | 0.000000 |
| 25% | 69.000000 |
| 50% | 106.000000 |
| 75% | 175.000000 |
| max | 10000.000000 |
| Name: | price, dtype: float64 |

min\_thresold, max\_thresold = df.price.quantile([0.01,0.999]) min\_thresold, max\_thresold

(30.0, 3000.0)

df[df.price<min\_thresold]

##### id name host\_id host\_name neighbourhood\_group neighbourho

**957** 375249

Enjoy Staten

Island Hospitality

Central,

1887999

Rimma & Staten Island Granitevi Jim

**2675** 1428154

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | | Semi-Private  Room |  | | |
| Large furnished 2 |
| **2860** | 1620248 | bedrooms- -  30 days Minimum | 2196224 | Sally | Manhattan |
| **3020** | 1767037 | Small Cozy Room Wifi & | 9284163 | Antonio | Queens |
|  |  | AC near JFK |  |  |  |
| **3918** | 2431607 | Bright, Airy Room Share | 4973668 | Gloria | Brooklyn |
|  |  | for 2 |  |  |  |
| **...** | ... | ... | ... | ... | ... |
|  |  | Cable and |  |  |  |
| **48486** | 36280646 | wfi, L/G included. | 272872092 | Chris | Queens |
|  |  | Cozy |  |  |  |
| **48647** | 36354776 | bedroom in  diverse | 273393150 | Liza | Queens |
|  |  | neighborhood |  |  |  |
|  |  | near JFK |  |  |  |
| **48832** | 36450814 | FLATBUSH HANG OUT | 267223765 | Jarmel | Brooklyn |
|  |  | AND GO |  |  |  |
|  |  | The place |  |  |  |
| **48867** | 36473044 | you were dreaming for. | 261338177 | Diana | Brooklyn |
|  |  | (only for |  |  |  |
|  |  | guys) |  |  |  |
| **48868** | 36473253 | Heaven for you(only for | 261338177 | Diana | Brooklyn |
|  |  | guy) |  |  |  |

Peaceful

5912572 Tangier Brooklyn Flatbu

404 rows × 16 columns

East Villa

Woodhav

Bedfor Stuyvesa

Forest Hi

Richmond H

Flatbu

Gravese

Gravese

df2 = df[(df.price>min\_thresold)&(df.price<max\_thresold)] df2.shape

(48183, 16)

df2.sample(5)



bedroom

|  |  |  |  |
| --- | --- | --- | --- |
| **id name host\_id** | **host\_name** | **neighbourhood\_group** | **neighbourh** |
| One room in a  **24530** 19729892 beautiful two 4452444  appartment  Large Upper | Jūrate | Brooklyn | Williamsb  Upper E |

|  |  |  |
| --- | --- | --- |
| **17785** 13952384 East Side 14945903  Alcove Studio | Nicole | Manhattan S |
| **37027** 29439494 VERREZZANO 221760432  LUXURY  **24132** 19439956 APARTMENT 136300414 | Daniel  Gonzalo | Staten Island Conc  Queens East Elmh |

HOUSE

**1128** 478832

5 MIN TO LGA

20 TO JFK

Gorgeous 2 bdrm in Carroll

Gardens

2371814 Jennifer Brooklyn Carroll Gard



df2.price.describe()

|  |  |
| --- | --- |
| count | 48183.000000 |
| mean | 148.772036 |
| std | 153.594795 |
| min | 31.000000 |
| 25% | 70.000000 |
| 50% | 110.000000 |
| 75% | 179.000000 |
| max | 2999.000000 |
| Name: | price, dtype: float64 |

## 4.2 Outlier detection and removal using z-score and standard



## deviation in python pandas

pip install matplotlib

 Requirement already satisfied: matplotlib in c:\users\harsh\appdata\local\programs\py Requirement already satisfied: contourpy>=1.0.1 in c:\users\harsh\appdata\local\progr Requirement already satisfied: cycler>=0.10 in c:\users\harsh\appdata\local\programs\ Requirement already satisfied: fonttools>=4.22.0 in c:\users\harsh\appdata\local\prog Requirement already satisfied: kiwisolver>=1.3.1 in c:\users\harsh\appdata\local\prog Requirement already satisfied: numpy>=1.23 in c:\users\harsh\appdata\local\programs\p Requirement already satisfied: packaging>=20.0 in c:\users\harsh\appdata\local\progra

Requirement already satisfied: pillow>=8 in c:\users\harsh\appdata\local\programs\pyt Requirement already satisfied: pyparsing>=2.3.1 in c:\users\harsh\appdata\local\progr Requirement already satisfied: python-dateutil>=2.7 in c:\users\harsh\appdata\local\p Requirement already satisfied: six>=1.5 in c:\users\harsh\appdata\local\programs\pyth Note: you may need to restart the kernel to use updated packages.

import pandas as pd import matplotlib

from matplotlib import pyplot as plt

%matplotlib inline

matplotlib.rcParams['figure.figsize'] = (10,6)

#### We are going to use heights dataset from kaggle.com. Dataset has heights and

 weights both but I have removed weights to make it simple

[https://www.kaggle.com/mustafaali96/weight-height](https://www.google.com/url?q=https%3A%2F%2Fwww.kaggle.com%2Fmustafaali96%2Fweight-height)

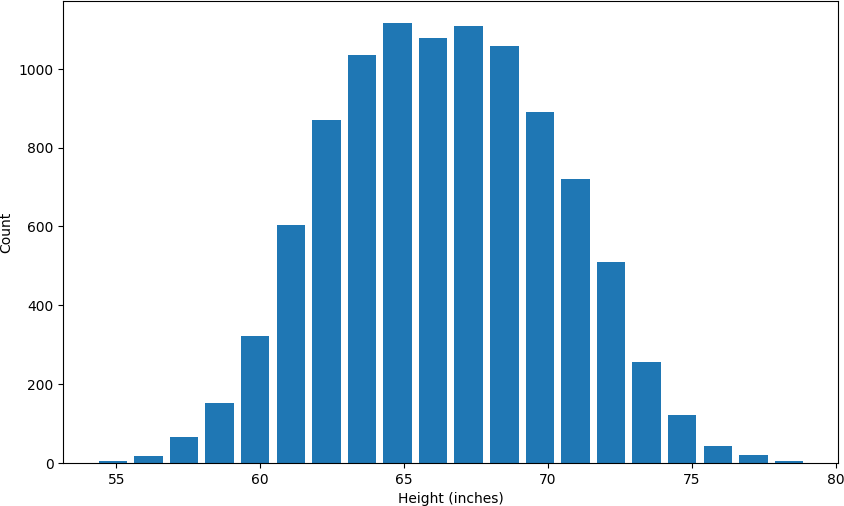


df = pd.read\_csv("heights (2).csv") df.sample(5)

|  |  |  |
| --- | --- | --- |
|  | **gender** | **height** |
| **2002** | Male | 70.214947 |
| **4472** | Male | 70.949770 |
| **9292** | Female | 62.234939 |
| **2666** | Male | 71.154717 |
| **615** | Male | 70.413869 |

plt.hist(df.height, bins=20, rwidth=0.8) plt.xlabel('Height (inches)')

plt.ylabel('Count') plt.show()



###  Plot bell curve along with histogram for our dataset

pip install scipy

 Requirement already satisfied: scipy in c:\users\harsh\appdata\local\programs\python\ Requirement already satisfied: numpy<2.3,>=1.23.5 in c:\users\harsh\appdata\local\pro Note: you may need to restart the kernel to use updated packages.

from scipy.stats import norm import numpy as np

df = pd.read\_csv("heights (2).csv")

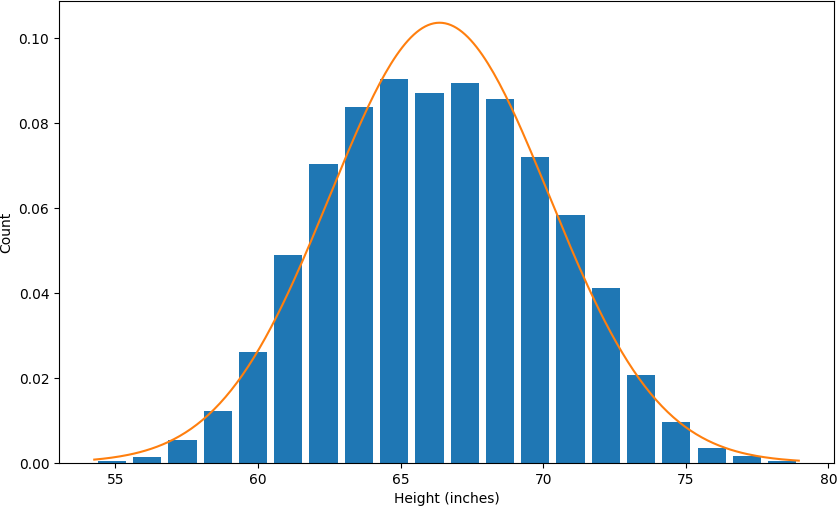
plt.hist(df.height, bins=20, rwidth=0.8, density=True) plt.xlabel('Height (inches)')

plt.ylabel('Count')

rng = np.arange(df.height.min(), df.height.max(), 0.1)

plt.plot(rng, norm.pdf(rng,df.height.mean(),df.height.std()))

 [<matplotlib.lines.Line2D at 0x1ae4288acf0>]



df.height.mean()

 np.float64(66.367559754866)

df.height.std()

np.float64(3.847528120795573)

#### Here the mean is 66.37 and standard deviation is 3.84.



###  (1) Outlier detection and removal using 3 standard deviation

One of the ways we can remove outliers is remove any data points that are beyond 3

 standard deviation from mean. Which means we can come up with following upper and lower bounds

upper\_limit = df.height.mean() + 3\*df.height.std() upper\_limit

 np.float64(77.91014411725271)

lower\_limit = df.height.mean() -3\*df.height.std() lower\_limit

np.float64(54.824975392479274)



df[(df.height>upper\_limit) | (df.height<lower\_limit)]

|  |  |  |
| --- | --- | --- |
|  | **gender** | **height** |
| **994** | Male | 78.095867 |
| **1317** | Male | 78.462053 |
| **2014** | Male | 78.998742 |
| **3285** | Male | 78.528210 |
| **3757** | Male | 78.621374 |
| **6624** | Female | 54.616858 |
| **9285** | Female | 54.263133 |

Above the heights on higher end is 78 inch which is around 6 ft 6 inch. Now that is quite unusual height. There are people who have this height but it is very uncommon and it is ok if you remove those data points. Similarly on lower end it is 54 inch which is around 4 ft 6 inch. While this is also a legitimate height you don't find many people having this height so it is safe to consider both of these cases as outliers

####  Now remove these outliers and generate new dataframe

df\_no\_outlier\_std\_dev = df[(df.height<upper\_limit) & (df.height>lower\_limit)] df\_no\_outlier\_std\_dev.head()



|  |  |
| --- | --- |
| **gender** | **height** |
| **0** Male | 73.847017 |
| **1** Male | 68.781904 |
| **2** Male | 74.110105 |
| **3** Male | 71.730978 |
| **4** Male | 69.881796 |

df\_no\_outlier\_std\_dev.shape

 (9993, 2)

df.shape

(10000, 2)

Above shows original dataframe data 10000 data points. Out of that we removed 7 outliers (i.e. 10000-9993)



###  (2) Outlier detection and removal using Z Score

Z score is a way to achieve same thing that we did above in part (1)

Z score indicates how many standard deviation away a data point is. For example in our case mean is 66.37 and standard deviation is 3.84.

If a value of a data point is 77.91 then Z score for that is 3 because it is 3 standard deviation away (77.91 = 66.37 + 3 \* 3.84)

##  Calculate the Z Score

zscore.png

df['zscore'] = ( df.height - df.height.mean() ) / df.height.std() df.head(5)



|  |  |  |
| --- | --- | --- |
| **gender** | **height** | **zscore** |
| **0** Male | 73.847017 | 1.943964 |
| **1** Male | 68.781904 | 0.627505 |
| **2** Male | 74.110105 | 2.012343 |
| **3** Male | 71.730978 | 1.393991 |
| **4** Male | 69.881796 | 0.913375 |

Above for first record with height 73.84, z score is 1.94. This means 73.84 is 1.94 standard

 deviation away from mean

(73.84-66.37)/3.84

 1.9453124999999998

Get data points that has z score higher than 3 or lower than -3. Another way of saying same thing is get data points that are more than 3 standard deviation away



df[df['zscore']>3]

|  |  |  |  |
| --- | --- | --- | --- |
|  | **gender** | **height** | **zscore** |
| **994** | Male | 78.095867 | 3.048271 |
| **1317** | Male | 78.462053 | 3.143445 |
| **2014** | Male | 78.998742 | 3.282934 |
| **3285** | Male | 78.528210 | 3.160640 |
| **3757** | Male | 78.621374 | 3.184854 |

df[df['zscore']<-3]

|  |  |  |
| --- | --- | --- |
| **gender** | **height** | **zscore** |
| **6624** Female | 54.616858 | -3.054091 |
| **9285** Female | 54.263133 | -3.146027 |

 # Here is the list of all outliers

df[(df.zscore<-3) | (df.zscore>3)]



|  |  |  |  |
| --- | --- | --- | --- |
|  | **gender** | **height** | **zscore** |
| **994** | Male | 78.095867 | 3.048271 |
| **1317** | Male | 78.462053 | 3.143445 |
| **2014** | Male | 78.998742 | 3.282934 |
| **3285** | Male | 78.528210 | 3.160640 |
| **3757** | Male | 78.621374 | 3.184854 |
| **6624** | Female | 54.616858 | -3.054091 |
| **9285** | Female | 54.263133 | -3.146027 |



###  Remove the outliers and produce new dataframe



df\_no\_outliers = df[(df.zscore>-3) & (df.zscore<3)] df\_no\_outliers.head()

|  |  |  |
| --- | --- | --- |
| **gender** | **height** | **zscore** |
| **0** Male | 73.847017 | 1.943964 |
| **1** Male | 68.781904 | 0.627505 |
| **2** Male | 74.110105 | 2.012343 |
| **3** Male | 71.730978 | 1.393991 |
| **4** Male | 69.881796 | 0.913375 |

df\_no\_outliers.shape

 (9993, 3)

df.shape

 (10000, 3)

#  Exercise

Q) You are given bhp.csv which contains property prices in the city of

 banglore, India. You need to examine price\_per\_sqft column and do

following,

1. Remove outliers using percentile technique first. Use [0.001, 0.999] for lower and upper bound percentiles
2. After removing outliers in step 1, you get a new dataframe.
3. On step(2) dataframe, use 4 standard deviation to remove outliers
4. Plot histogram for new dataframe that is generated after step (3). Also plot bell curve on same histogram
5. On step(2) dataframe, use zscore of 4 to remove outliers. This is quite similar to step (3) and you will get exact same result

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| import pandas as pd |  |  | | | |
| import matplotlib  from matplotlib import pyplot as plt  %matplotlib inline  matplotlib.rcParams['figure.figsize'] = | (12,8) |
|  |  |  |  |  |  |
| df = pd.read\_csv("bhp.csv") df.head() |  |  |  |  |  |
| **location size** | **total\_sqft** | **bath** | **price** | **bhk** | **price\_per\_sqft** |
| **0** Electronic City Phase II 2 BHK | 1056.0 | 2.0 | 39.07 | 2 | 3699 |
| **1** Chikka Tirupathi 4 Bedroom | 2600.0 | 5.0 | 120.00 | 4 | 4615 |
| **2** Uttarahalli 3 BHK | 1440.0 | 2.0 | 62.00 | 3 | 4305 |
| **3** Lingadheeranahalli 3 BHK | 1521.0 | 3.0 | 95.00 | 3 | 6245 |
| **4** Kothanur 2 BHK | 1200.0 | 2.0 | 51.00 | 2 | 4250 |

df.price\_per\_sqft.describe()

 count 1.320000e+04 mean 7.920337e+03

std 1.067272e+05

min 2.670000e+02

25% 4.267000e+03

50% 5.438000e+03

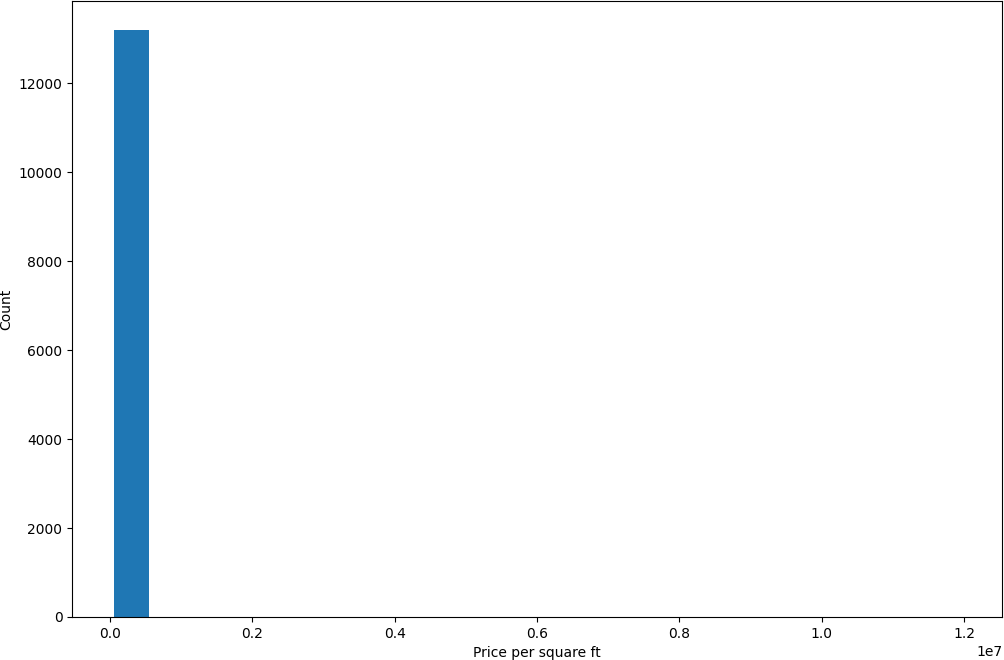
75% 7.317000e+03

max 1.200000e+07

Name: price\_per\_sqft, dtype: float64

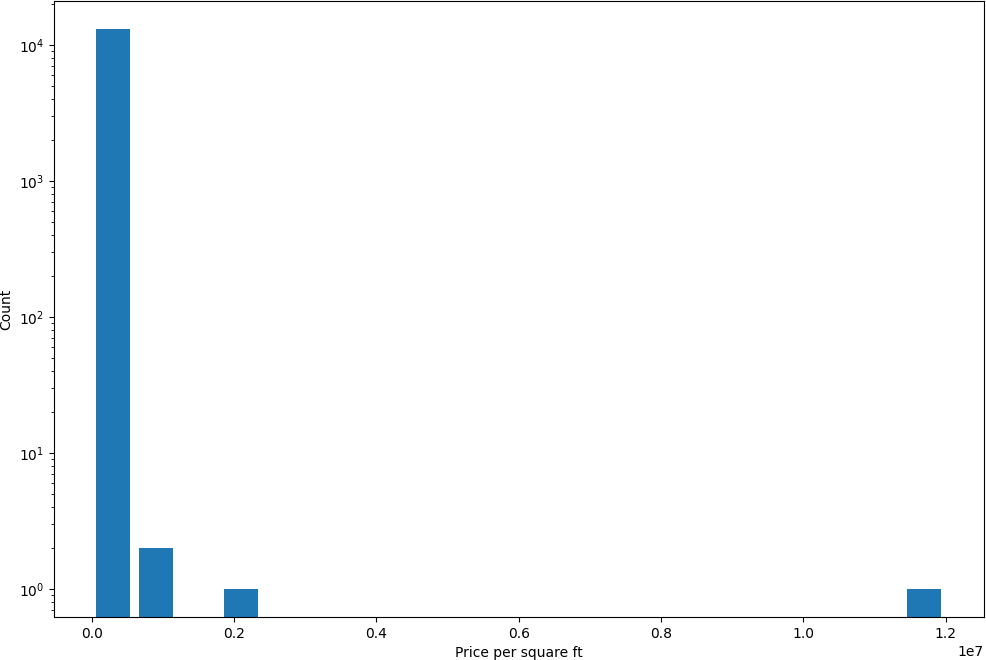
plt.hist(df.price\_per\_sqft, bins=20, rwidth=0.8) plt.xlabel('Price per square ft')

plt.ylabel('Count') plt.show()



plt.hist(df.price\_per\_sqft, bins=20, rwidth=0.8) plt.xlabel('Price per square ft')

plt.ylabel('Count') plt.yscale('log') plt.show()



####  (1) Treat outliers using percentile first

lower\_limit, upper\_limit = df.price\_per\_sqft.quantile([0.001, 0.999]) lower\_limit, upper\_limit

(1366.184, 50959.36200000098)

outliers = df[(df.price\_per\_sqft>upper\_limit) | (df.price\_per\_sqft<lower\_limit)] outliers.sample(10)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | **location** | **size** | **total\_sqft** | **bath** | **price** | **bhk** | **price\_per\_sqft** |
| **2392** | other | 4 Bedroom | 2000.0 | 3.0 | 25.0 | 4 | 1250 |
| **12328** | other | 4 Bedroom | 4350.0 | 8.0 | 2600.0 | 4 | 59770 |
| **9144** | other | 4 Bedroom | 10961.0 | 4.0 | 80.0 | 4 | 729 |
| **11635** | Begur | 3 BHK | 2400.0 | 3.0 | 12.0 | 3 | 500 |
| **12355** | other | 4 BHK | 16335.0 | 4.0 | 149.0 | 4 | 912 |
| **7166** | Yelahanka | 1 Bedroom | 26136.0 | 1.0 | 150.0 | 1 | 573 |
| **7862** | JP Nagar | 3 BHK | 20000.0 | 3.0 | 175.0 | 3 | 875 |
| **7012** | other | 1 BHK | 650.0 | 1.0 | 500.0 | 1 | 76923 |
| **1867** | other | 3 Bedroom | 52272.0 | 2.0 | 140.0 | 3 | 267 |
| **7799** | other | 4 BHK | 2000.0 | 3.0 | 1063.0 | 4 | 53150 |

df2 = df[(df.price\_per\_sqft<upper\_limit) & (df.price\_per\_sqft>lower\_limit)] df2.shape

 (13172, 7)

df.shape

 (13200, 7)

df.shape[0] - df2.shape[0]

28

We removed total 28 outliers



####  (2) Now remove outliers using 4 standard deviation

max\_limit = df2.price\_per\_sqft.mean() + 4\*df2.price\_per\_sqft.std() min\_limit = df2.price\_per\_sqft.mean() - 4\*df2.price\_per\_sqft.std() max\_limit, min\_limit

(np.float64(23227.73653589432), np.float64(-9900.429065502582))

df2[(df2.price\_per\_sqft>max\_limit) | (df2.price\_per\_sqft<min\_limit)].sample(10)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | **location** | **size** | **total\_sqft** | **bath** | **price** | **bhk** | **price\_per\_sqft** |
| **12900** | HAL 2nd Stage | 5 Bedroom | 2040.0 | 4.0 | 500.0 | 5 | 24509 |
| **10000** | other | 6 Bedroom | 1200.0 | 5.0 | 280.0 | 6 | 23333 |
| **45** | HSR Layout | 8 Bedroom | 600.0 | 9.0 | 200.0 | 8 | 33333 |
| **3500** | Kundalahalli | 1 BHK | 2400.0 | 1.0 | 650.0 | 1 | 27083 |
| **3675** | Kasturi Nagar | 5 Bedroom | 1650.0 | 5.0 | 450.0 | 5 | 27272 |
| **1281** | Chamrajpet | 9 Bedroom | 4050.0 | 7.0 | 1200.0 | 9 | 29629 |
| **9873** | other | 3 Bedroom | 2400.0 | 6.0 | 775.0 | 3 | 32291 |
| **8157** | other | 4 BHK | 2230.0 | 4.0 | 792.0 | 4 | 35515 |
| **12393** | Electronic City Phase II | 1 BHK | 1200.0 | 1.0 | 295.0 | 1 | 24583 |
| **10536** | other | 4 Bedroom | 2400.0 | 4.0 | 595.0 | 4 | 24791 |

df3 = df2[(df2.price\_per\_sqft>min\_limit) & (df2.price\_per\_sqft<max\_limit)] df3.shape

 (13047, 7)

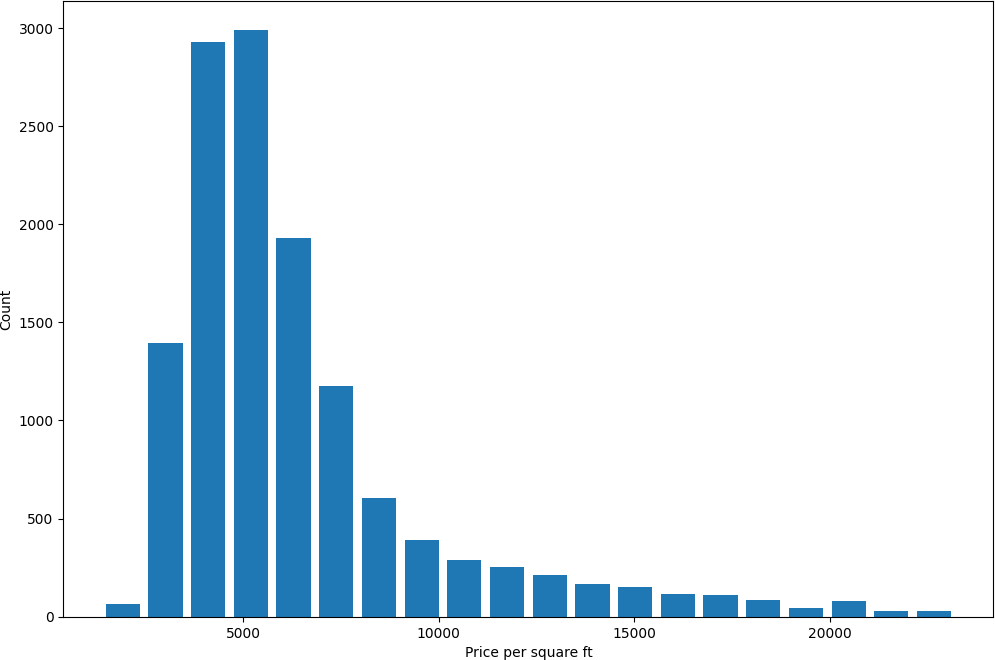
df2.shape[0]-df3.shape[0]

 125

####  In this step we removed total 125 outliers

plt.hist(df3.price\_per\_sqft, bins=20, rwidth=0.8) plt.xlabel('Price per square ft')

plt.ylabel('Count') plt.show()

from scipy.stats import norm import numpy as np

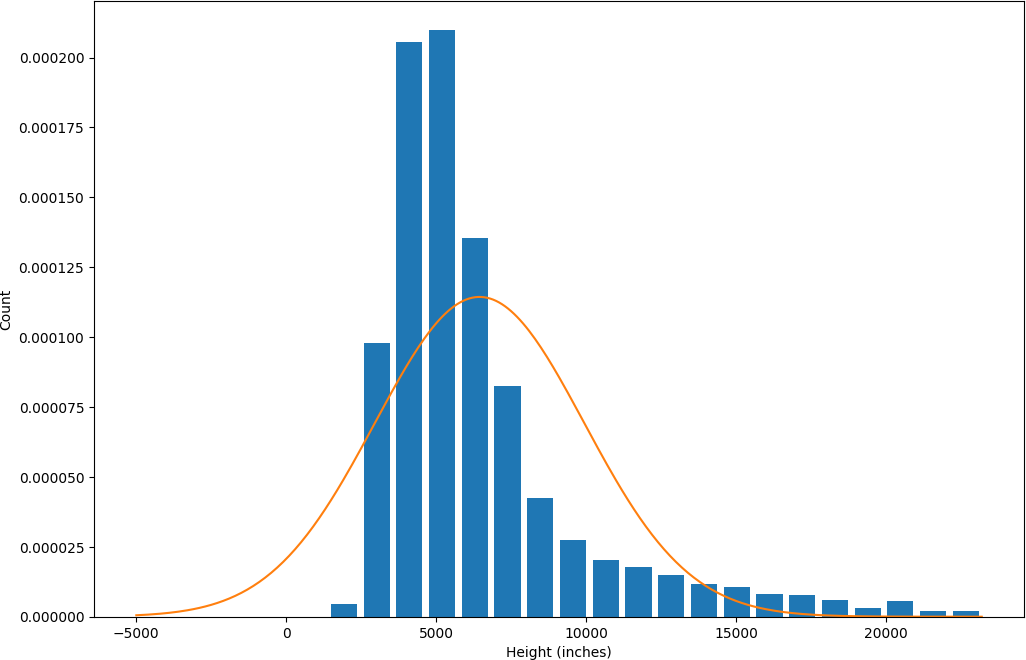
plt.hist(df3.price\_per\_sqft, bins=20, rwidth=0.8, density=True) plt.xlabel('Height (inches)')

plt.ylabel('Count')

rng = np.arange(-5000, df3.price\_per\_sqft.max(), 100)

plt.plot(rng, norm.pdf(rng,df3.price\_per\_sqft.mean(),df3.price\_per\_sqft.std()))

[<matplotlib.lines.Line2D at 0x1ae443cda60>]





####  (3) Now remove outliers using z score. Use z score of 4 as your thresold

df2['zscore'] = (df2.price\_per\_sqft-df2.price\_per\_sqft.mean())/df2.price\_per\_sqft.std() df2.sample(10)

 C:\Users\harsh\AppData\Local\Temp\ipykernel\_18888\722868599.py:1: SettingWithCopyWarn A value is trying to be set on a copy of a slice from a DataFrame.

Try using .loc[row\_indexer,col\_indexer] = value instead

See the caveats in the documentation: [https://pandas.pydata.org/pandas-docs/stable/u](https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)s df2['zscore'] = (df2.price\_per\_sqft-df2.price\_per\_sqft.mean())/df2.price\_per\_sqft.s

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **location** | **size** | **total\_sqft** | **bath** | **price** | **bhk** | **price\_per\_sqft** | **zscore** |
| **11060** | Talaghattapura | 2 BHK | 1062.0 | 2.0 | 42.48 | 2 | 4000 | -0.643236 |
| **5011** | Marathahalli | 3 BHK | 1730.0 | 3.0 | 110.00 | 3 | 6358 | -0.073811 |
| **4963** | NRI Layout | 2 BHK | 1060.0 | 2.0 | 35.00 | 2 | 3301 | -0.812035 |
| **5024** | BTM 2nd  Stage | 2 BHK | 1280.0 | 2.0 | 80.00 | 2 | 6250 | -0.099892 |
| **3987** | Gottigere | 3 BHK | 1304.0 | 3.0 | 80.00 | 3 | 6134 | -0.127904 |
| **12990** | Whitefield | 3 BHK | 1404.0 | 2.0 | 59.00 | 3 | 4202 | -0.594456 |
| **6955** | Vijayanagar | 3 BHK | 2047.0 | 3.0 | 136.00 | 3 | 6643 | -0.004988 |
| **4672** | 7th Phase JP  Nagar | 2 BHK | 1130.0 | 2.0 | 73.00 | 2 | 6460 | -0.049180 |

5

outliers\_z = df2[(df2.zscore < -4) | (df2.zscore>4)] outliers\_z.shape

 (125, 8)



outliers\_z.sample(5)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **location** | **size** | **total\_sqft** | **bath** | **price** | **bhk** | **price\_per\_sqft** | **zscore** |
| **3401** | Indira Nagar | 6 Bedroom | 2480.0 | 4.0 | 750.0 | 6 | 30241 | 5.693607 |
| **3816** | Domlur | 6 BHK | 2400.0 | 4.0 | 600.0 | 6 | 25000 | 4.427977 |
| **6597** | other | 2 BHK | 1030.0 | 2.0 | 300.0 | 2 | 29126 | 5.424350 |
| **3340** | other | 19 BHK | 2000.0 | 16.0 | 490.0 | 19 | 24500 | 4.307234 |
| **7262** | other | 4 Bedroom | 1200.0 | 5.0 | 325.0 | 4 | 27083 | 4.930994 |

df4 = df2[(df2.zscore>-4)&(df2.zscore<4)] df4.shape

 (13047, 8)

df2.shape[0] - df4.shape[0]

125

In this step also we removed 125 outliers. The result would be exactly same as 4 standard deviation

##  4.3 Outlier Detection and Removal Using IQR

import pandas as pd

df = pd.read\_csv("heights (3).csv") df

##### name height

1. mohan 1.2
2. maria 2.3
3. sakib 4.9
4. tao 5.1
5. virat 5.2
6. khusbu 5.4
7. dmitry 5.5
8. selena 5.5
9. john 5.6
10. imran 5.6
11. jose 5.8
12. deepika 5.9
13. yoseph 6.0
14. binod 6.1
15. gulshan 6.2
16. johnson 6.5
17. donald 7.1
18. aamir 14.5

**18** ken 23.2

**19** Liu 40.2

df.describe()



|  |  |
| --- | --- |
|  | **height** |
| **count** | 20.000000 |
| **mean** | 8.390000 |
| **std** | 8.782812 |
| **min** | 1.200000 |
| **25%** | 5.350000 |
| **50%** | 5.700000 |
| **75%** | 6.275000 |
| **max** | 40.200000 |

###  # Detect outliers using IQR

Q1 = df.height.quantile(0.25) Q3 = df.height.quantile(0.75) Q1, Q3

 (np.float64(5.3500000000000005), np.float64(6.275))

IQR = Q3 - Q1 IQR

 np.float64(0.9249999999999998)

lower\_limit = Q1 - 1.5\*IQR upper\_limit = Q3 + 1.5\*IQR lower\_limit, upper\_limit

 (np.float64(3.962500000000001), np.float64(7.6625))

###  # Here are the outliers

df[(df.height<lower\_limit)|(df.height>upper\_limit)]

##### name height

1. mohan 1.2
2. maria 2.3

**17** aamir 14.5

**18** ken 23.2

**19** Liu 40.2



###  Remove outliers

df\_no\_outlier = df[(df.height>lower\_limit)&(df.height<upper\_limit)] df\_no\_outlier

##### name height

1. sakib 4.9
2. tao 5.1
3. virat 5.2
4. khusbu 5.4
5. dmitry 5.5
6. selena 5.5
7. john 5.6
8. imran 5.6
9. jose 5.8
10. deepika 5.9
11. yoseph 6.0
12. binod 6.1
13. gulshan 6.2
14. johnson 6.5
15. donald 7.1

#  Exercise

You are given height\_weight.csv file which contains heights and weights of 1000 people.

 Dataset is taken from here, [https://www.kaggle.com/mustafaali96/weight-height](https://www.google.com/url?q=https%3A%2F%2Fwww.kaggle.com%2Fmustafaali96%2Fweight-height)

You need to do this,

1. Load this csv in pandas dataframe and first plot histograms for height and weight parameters
2. Using IQR detect weight outliers and print them
3. Using IQR, detect height outliers and print them



import pandas as pd import matplotlib

from matplotlib import pyplot as plt

%matplotlib inline

matplotlib.rcParams['figure.figsize'] = (8,4)

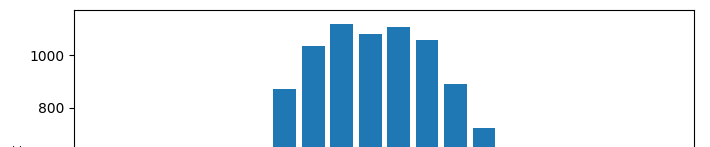
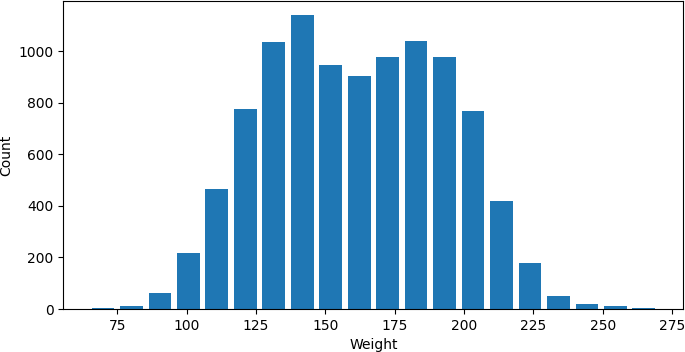
df = pd.read\_csv("height\_weight.csv") df.head(5)

|  |  |  |
| --- | --- | --- |
| **gender** | **height** | **weight** |
| **0** Male | 73.847017 | 241.893563 |
| **1** Male | 68.781904 | 162.310473 |
| **2** Male | 74.110105 | 212.740856 |
| **3** Male | 71.730978 | 220.042470 |
| **4** Male | 69.881796 | 206.349801 |

 # Histgram for weights

plt.hist(df.weight, bins=20, rwidth=0.8) plt.xlabel('Weight')

plt.ylabel('Count') plt.show()



 # Histgram for heights

plt.hist(df.height, bins=20, rwidth=0.8) plt.xlabel('Height')

plt.ylabel('Count') plt.show()