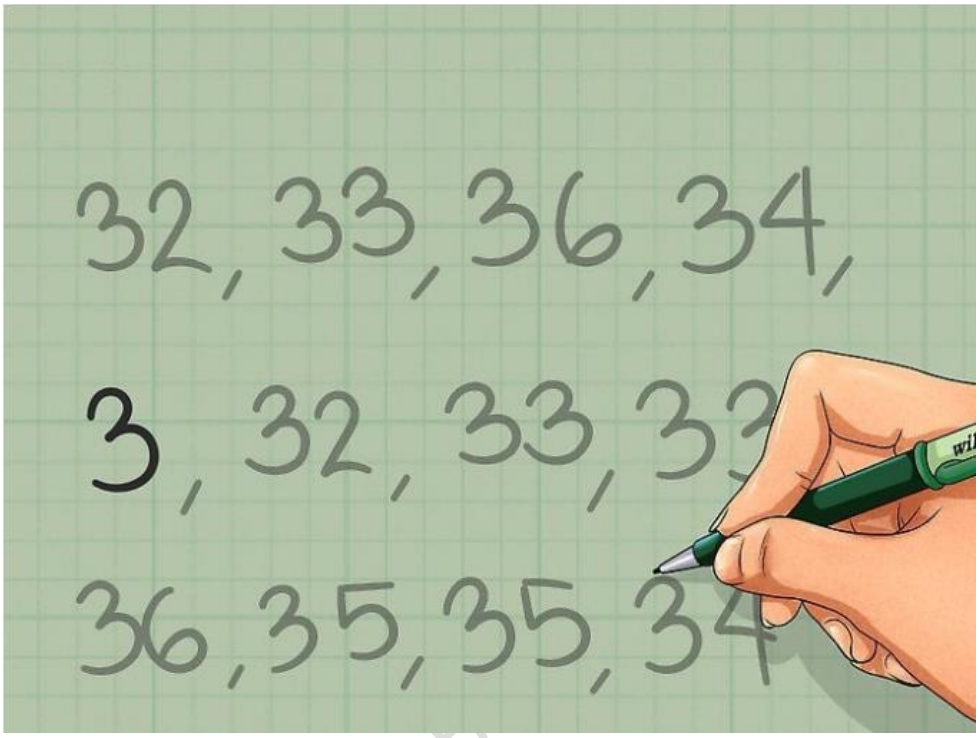
Outliers

*In statistics, an **outlier** is an observation point that is distant from other observations.*

The above definition suggests that outlier is something, which is separate/different from the crowd.



Do you see anything different in the above image? All the numbers in the 30's range except number 3. That's our outlier, because it is nowhere near to the other numbers.

How did an outlier introduce to the population?

The Data Science project starts with data collection and that's when outliers first introduced to the population (in some cases). Though, you will not know about the outliers at all in the collection phase. The outliers can be a result of a mistake during data collection or it can be just an indication of variance in your data.

Let's have a look at some examples. Suppose you have been asked to observe the performance of Indian cricket team i.e. Run made by each player and collect the data.

Players	Scores
Player1	500
Player2	350
Player3	10
Player4	300
Player5	450

As you can see from the above collected data that all other players scored 300+ except Player3 who scored 10. This figure can be just a typing **mistake,** or it is showing the **variance** in your data and indicating that Player3 is performing very bad so, needs improvements.

Now that we know outliers can either be a mistake or just variance, how would you decide if they are important or not. Well, it is pretty simple if they are the result of a mistake, then we can ignore them, but if it is just a variance in the data, we would need think a bit further. Before we try to understand whether to ignore the outliers or not, we need to know the ways to identify them.

Finding Outliers

In Statistics, a few methods are available to detect the outliers.

We will use the Boston House Pricing Dataset, which is included in the sklearn dataset API. We will load the dataset and separate out the features and targets.

Jupyter Notebook

```
import matplotlib.pyplot as plt
import numpy as np
from sklearn import datasets, linear_model, metrics
from sklearn.metrics import mean_squared_error
import pandas as pd
```

```
# load the boston dataset
boston = datasets.load_boston(return_X_y=False)

columns = boston.feature_names

boston_dataset = pd.DataFrame(boston.data)
boston_dataset.columns = columns
boston_dataset.head()
```

Output:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90	4.98
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90	9.14
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	4.03
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63	2.94
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	5.33

There are two types of analysis we will follow to find the outliers- Univariate (one variable outlier analysis) and Multivariate(two or more variable outlier analysis)

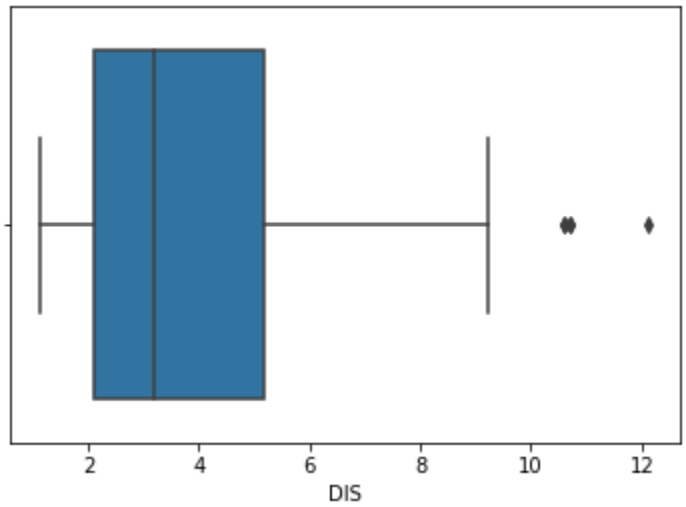
Box and Whisker Plot:

In descriptive statistics, a **box plot** is a method for graphically depicting groups of numerical data through their quartiles. Box plots may also have **lines extending vertically** from the boxes (whiskers) **indicating variability** outside the upper and lower quartiles, hence the terms box-and-whisker plot and box-and-whisker diagram. **Outliers** may be **plotted** as **individual** points.

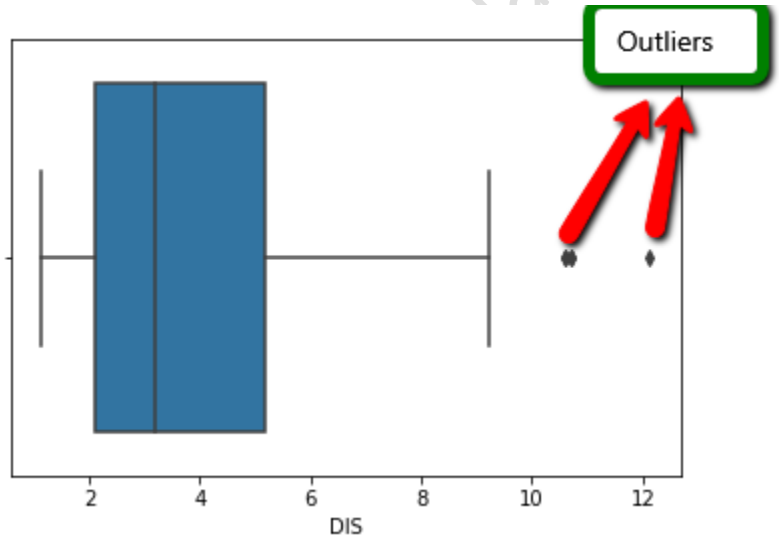
Let's try for a feature:

```
import seaborn as sns
sns.boxplot(x=boston_dataset['DIS'])
```

Output:



Interpretation:



Above plot shows three points between 10 to 12, these are outliers as there are not included in the box of other observation i.e. nowhere near the quartiles.

Here we analyzed Uni-variate outlier i.e. we used DIS column only to check the outlier.

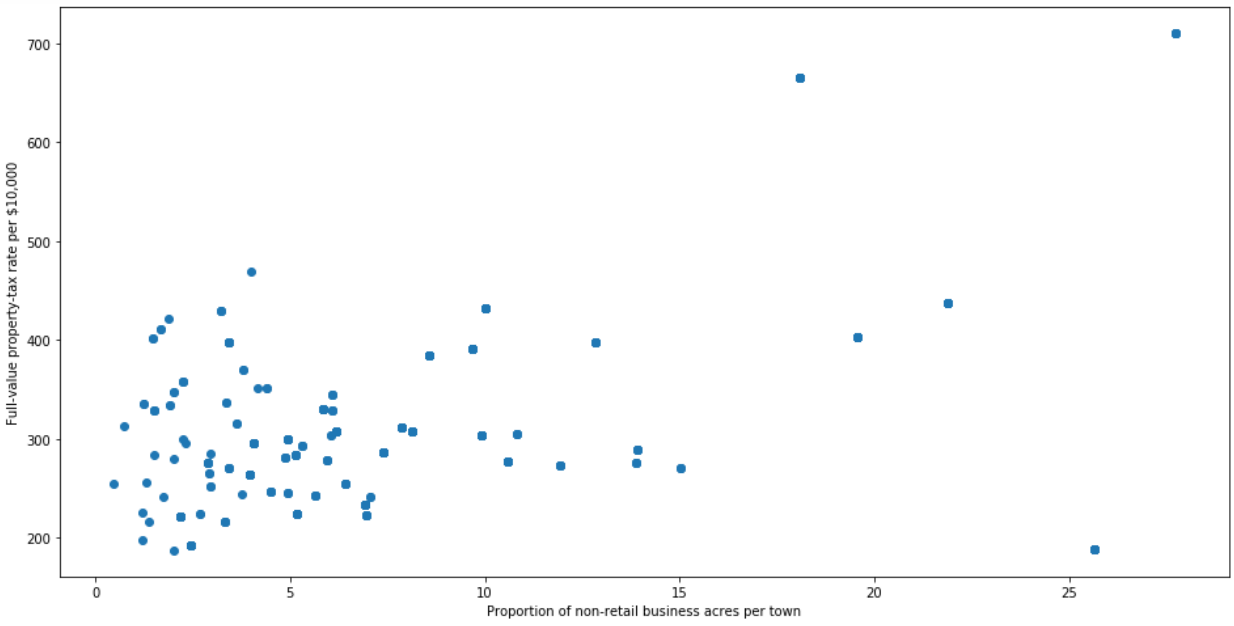
Scatter Plot:

A **scatter plot** is a type of plot or mathematical diagram using Cartesian coordinates to display values for typically two variables for a set of data. The data are displayed as a **collection of points**, each having the value of **one variable** determining the position on the **horizontal** axis and the value of the **other variable** determining the position on the **vertical** axis.

As the definition suggests, the scatter plot is the collection of points that shows values for two variables. We can try and draw scatter plot for two variables from our housing dataset.

```
fig, ax = plt.subplots(figsize=(16,8))
ax.scatter(boston_dataset['INDUS'], boston_dataset['TAX'])
ax.set_xlabel('Proportion of non-retail business acres per town')
ax.set_ylabel('Full-value property-tax rate per $10,000')
plt.show()
```

Output:



Discover outliers with mathematical function

Z-Score

$$z = \frac{X - \overline{X}}{S}$$

where z is the standard score,
S = the standard deviation of a sample,
X = each value in the data set,
$\overline{X}$ = mean of all values in the data set.

Note: This formula is for sample data.

The **Z-score** is the signed number of standard deviations by which the value of an observation or data point is above the mean value of what is being observed or measured.

The intuition behind Z-score is to describe any data point by finding their relationship with the Standard Deviation and Mean of the group of data points. Z-score is finding the distribution of data where mean is 0 and standard deviation is 1 i.e. normal distribution.

You must be wondering that; how does this help in identifying the outliers? Well, while calculating the Z-score we re-scale and center the data and look for data points which are too far from zero. These data points which are way too far from zero will be treated as the outliers. In most of the cases a threshold of 3 or -3 is used i.e. if the Z-score value is greater than or less than 3 or -3 respectively, that data point will be identified as outliers.

We will use Z-score function defined in scipy library to detect the outliers.

Looking the code and the output above, it is difficult to say which data point is an outlier. Let's try and define a threshold to identify an outlier.

```
from scipy import stats
import numpy as np
z = np.abs(stats.zscore(boston_dataset['DIS']))
print(z)
```

```
6.74813737e-01 6.74813737e-01 9.88081999e-01 1.35273767e+00
1.35273767e+00 1.26607787e+00 1.26607787e+00 1.04118073e+00
6.82657328e-01 8.65151540e-01 4.83525195e-01 4.83525195e-01
4.83525195e-01 1.54155429e+00 1.17504468e+00 9.20817266e-01
1.26940546e+00 2.00537196e+00 2.00537196e+00 2.25337204e+00
2.16243392e+00 2.37544683e+00 2.37544683e+00 3.28729991e+00
3.28729991e+00 3.96051769e+00 3.22806892e+00 3.22806892e+00
7.95217788e-01 6.13151408e-01 5.09758620e-01 6.11297468e-01
6.06971609e-01 7.12836316e-01 8.04059654e-01 8.98610575e-01
8.98610575e-01 1.03718068e+00 9.71056832e-01 1.08595355e+00
1.17061680e+00 1.15911286e+00 1.23246232e+00 1.24829211e+00
1.26706919e+00 1.24586773e+00 1.26355146e+00 1.17831778e+00
1.16472222e+00 1.15849488e+00 1.14513701e+00 1.14513701e+00
```

```
threshold = 3
print(np.where(z > 3))
```

Output:

```
(array([351, 352, 353, 354, 355], dtype=int64),)
```

Note:

The observations 351, 352, 353, 354 and 355 (column – 'DIS') has outliers.

To print z-score value:

```
print(z[351])
```

```
3.2872999127421076
```

IQR score

Box plot use the IQR method to display data and outliers(shape of the data) but in order to be get a list of identified outliers, we will need to use the mathematical formula and retrieve the outlier data

The **interquartile range** (**IQR**), also called the **midspread** or **middle 50%**, or technically **H-spread**, is a measure of statistical dispersion, being equal to the difference between 75th and 25th percentiles, or between upper and lower quartiles, IQR = Q3 − Q1.

IQR is somewhat similar to Z-score in terms of finding the distribution of data and then keeping some threshold to identify the outlier.

How to find outliers for a column/feature/dimension:

```
Q1 = boston_dataset['DIS'].quantile(0.25)
Q3 = boston_dataset['DIS'].quantile(0.75)
IQR = Q3 - Q1
print(IQR, Q1, Q3)
```

Output:

```
3.0882500000000004 2.100175 5.1884250000000005
```

The below prints "Number of Outliers" present in the column "DIS":

```
((boston_dataset['DIS'] < (Q1 - 1.5 * IQR)) |
(boston_dataset['DIS'] > (Q3 + 1.5 * IQR))).sum()
```

Output:

5

There are 5 outliers.

Printing the observations/rows that has the outliers on "DIS":

```
Q1 = boston_dataset['DIS'].quantile(0.25)
Q3 = boston_dataset['DIS'].quantile(0.75)
IQR = Q3 - Q1

boston_dataset_new = boston_dataset[((boston_dataset['DIS'] < (Q1
- 1.5 * IQR)) | (boston_dataset['DIS'] > (Q3 + 1.5 * IQR)))]
boston_dataset_new
```

Output:

:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT
351	0.07950	60.0	1.69	0.0	0.411	6.579	35.9	10.7103	4.0	411.0	18.3	370.78	5.49
352	0.07244	60.0	1.69	0.0	0.411	5.884	18.5	10.7103	4.0	411.0	18.3	392.33	7.79
353	0.01709	90.0	2.02	0.0	0.410	6.728	36.1	12.1265	5.0	187.0	17.0	384.46	4.50
354	0.04301	80.0	1.91	0.0	0.413	5.663	21.9	10.5857	4.0	334.0	22.0	382.80	8.05
355	0.10659	80.0	1.91	0.0	0.413	5.936	19.5	10.5857	4.0	334.0	22.0	376.04	5.57

Display the remaining rows:

```
Q1 = boston_dataset['DIS'].quantile(0.25)
Q3 = boston_dataset['DIS'].quantile(0.75)
IQR = Q3 - Q1

boston_dataset_new = boston_dataset[~((boston_dataset['DIS'] <
(Q1 - 1.5 * IQR)) | (boston_dataset['DIS'] > (Q3 + 1.5 * IQR)))]
boston_dataset_new
```

IQR value for each feature:

Let's find out we can box plot uses IQR and how we can use it to find the list of outliers as we did use Z-score calculation. First, we will calculate IQR:

```
Q1 = boston_dataset.quantile(0.25)
Q3 = boston_dataset.quantile(0.75)
IQR = Q3 - Q1
print(IQR)
```

Output:

```
CRIM        3.595038
ZN         12.500000
INDUS      12.910000
CHAS        0.000000
NOX         0.175000
RM          0.738000
AGE        49.050000
DIS         3.088250
RAD        20.000000
TAX       387.000000
PTRATIO     2.800000
B          20.847500
LSTAT      10.005000
dtype: float64
```

Z-Score

we saw how one can detect the outlier using Z-score but now we want to remove or filter the outliers and get the clean data. This can be done with just one-line code as we have already calculated the Z-score.

Removing observations based on z-score value:

```
z = np.abs(stats.zscore(boston_dataset))
```

```
boston_dataset_o = boston_dataset[(z < 3).all(axis=1)]
```

```
boston_dataset_o.shape
```

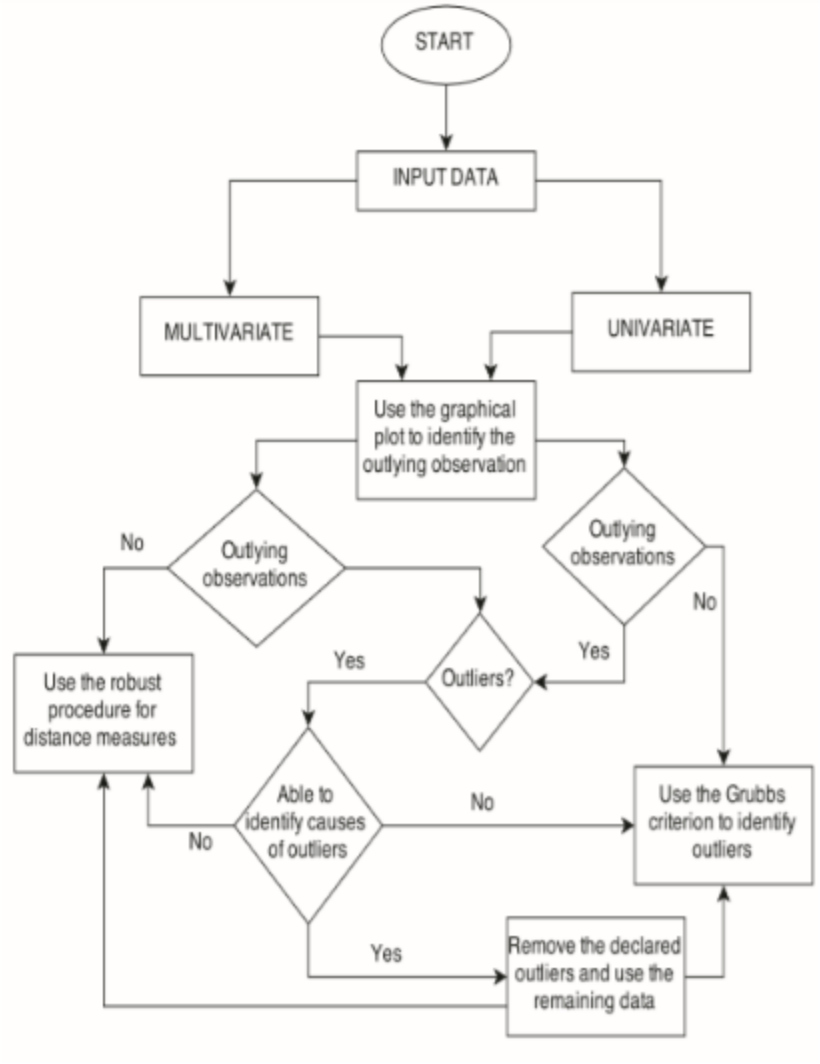
```
(415, 13)
```

```
boston_dataset.shape
```

```
(506, 13)
```


Reference:
https://towardsdatascience.com/ways-to-detect-and-remove-the-outliers-404d16608dba

https://medium.com/@mehulved1503/effective-outlier-detection-techniques-in-machine-learning-ef609b6ade72



General Guiding Principle to Outlier Detection Approach Selection

Dealing with outliers:

Below are some techniques:

- Use standardization techniques
- If you are using data for classification use Decision Trees (CART, Random Forest, XgBoost, etc.,). As decision trees are split by purity so they don't get affected by outliers like median.
- Remove the outliers and apply machine learning