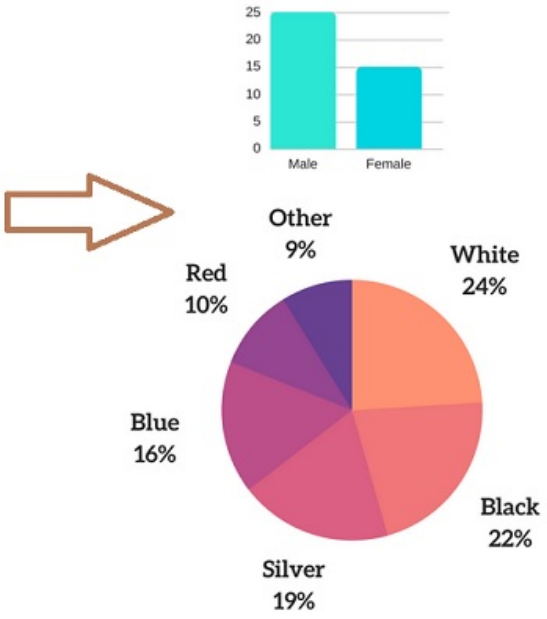
Descriptive statistics

- Descriptive statistics describes, shows, and summarizes the features of a dataset.
- Descriptive statistics are just the representation of the data (sample) available and not based on any theory of probability.
- Charts and graphs are often used to present descriptive statistics.
- It helps Data analysts to understand the data better.

	A	B	C	D
1	Respondent Number	Age	Gender	Favorite Car Color
2	1	22	M	White
3	2	37	F	Silver
4	3	45	F	Black
5	4	62	F	Gray
6	5	28	M	Red
7	6	45	M	Green
8	7	88	F	Brown
9	8	61	M	White
10	9	95	M	Black
11	10	27	M	White
12	11	39	F	Green
13	12	43	M	Brown
14	13	55	F	Black
15	14	59	F	White

RAW DATA

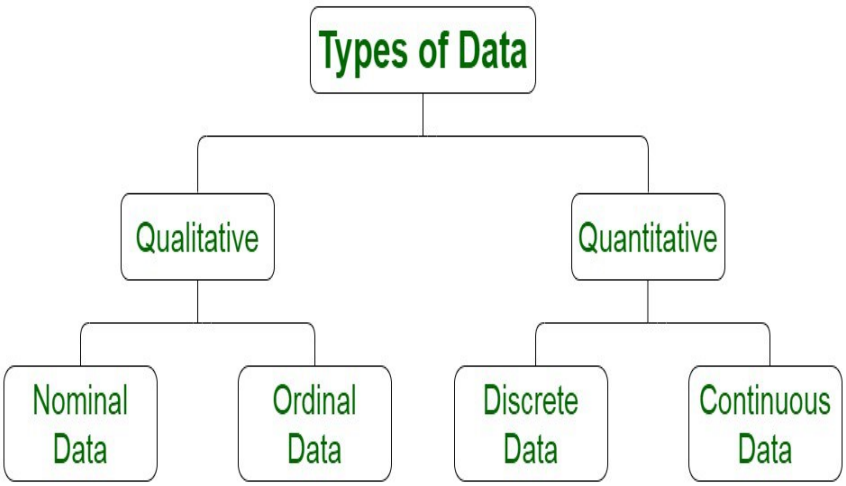


Descriptive Statistics

Data:

- A data set is a collection of related information or Records.
- Data is the most important part of Machine Learning, Artificial Intelligence, and Data Analytics, without data it is impossible to train any model and all modern research and automation will go in worthless.
- Data can be any value, text, sound, or picture i.e., structured or unstructured data (raw data).
- Each row of a data set is called a **record**. Each data set also has multiple **attributes**, each of which gives information on a specific characteristic.

- Attributes can also be termed as feature, variable, dimension or field.



Qualitative Data:

- Qualitative Data can't be measured or counted in the **form of numbers** i.e., sorted by category, not by number. It is also known as **Categorical Data**.
- Qualitative Data may consist of audio, images, symbols, or text.

  Examples: Gender: Male, Female. Product: Good or Bad.
- Qualitative data tells about the perception of people. Which helps market researchers understand the customers' tastes and then design their ideas and strategies accordingly?
- The Qualitative data is further classified into two parts:

  1. Nominal Data

  2. Ordinal Data

1. Nominal Data:

- Nominal Data is used to label variables without any order or quantitative value.
- Numerical task such as Arithmetic operations can't be performed on Nominal data.
- Nominal Data don't have any meaningful order; they are distributed to distinct categories.

Examples:

1. The color of hair can be considered nominal data, as one color can't be compared with another color.

2. Marital status (Single, Married)
3. Nationality (Indian, German, American)
4. Gender (Male, Female)

Ordinal data:

- Ordinal data have natural ordering where a number is present in some kind of order by their position on the scale.

  Examples:

  1. Feedback System on a scale 1 to 10.
  2. Assigning Ranks 1, 2, 3.
  3. Grading in Exam: A, B, C, D.
- Ordinal data is used for observation like customer satisfaction, happiness, etc.
- Since ordering is possible in case of ordinal data, median, and quartiles can be identified Mean can still not be calculated.

Quantitative data

- Quantitative data relates to information about the quantity of an object.
- Quantitative data can be expressed in numerical values, which make it countable and include statistical data analysis.
- Quantitative data also known as a **Numerical data**.
- Quantitative data can be used for statistical manipulation and represented on a wide variety of graphs and charts such as bar graphs, histograms, scatter plots, boxplot, pie charts, etc.

Examples:

- Height or weight of a person: 175cm
- Room Temperature: 29 centigrade
- Marks: 59, 80, 60…
- Time: 2PM, 6AM

The Quantitative data is further classified into two parts:

1. Discrete Data
2. Continuous Data

1. Discrete Data

- Discrete means **distinct or separate**. The discrete data are countable and have finite values, their **subdivision is not possible** i.e., can't be broken into decimal or fraction values.
- The discrete data contain the values that fall under **integers or whole numbers.**
- Represented mainly by a bar graph, scatter plot, etc.

Examples:

- Total no. of students present in a class.
- Numbers of employees in a company.
- Days in a week.

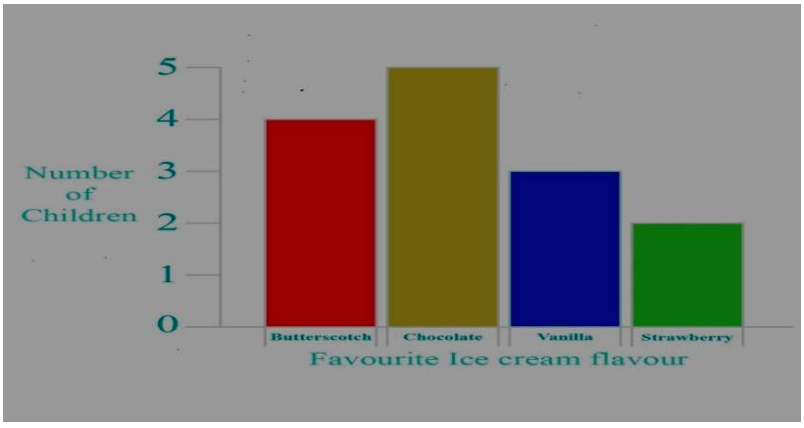


Figure: Bar Plot

2. Continuous data

- Continuous data is data that can take any value within a **range** such as weight, length, temperature, speed, etc.

- Continuous data can be in the form of **decimal or fractional numbers**.
- Continuous data can be measured on an infinite scale; it can take any value between two numbers, no matter how small.
- Represented mainly by a Line plot, Histogram plot**,** etc.

Examples:

- Height of a person
- Speed of a vehicle
- Time-taken to finish the work

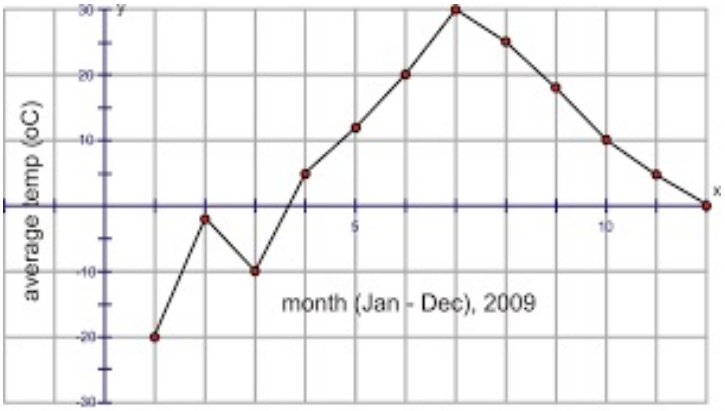


Figure: Line Plot

Interval data

- Interval data is numeric data for which not only the order is known, but the exact difference between values is also known.
- The distance between the two points is equal i.e., equal interval between adjacent values.
- Example: Celsius temperature, IQ Score, Income Ranges.
- The problem with interval values data is that they don't have a "true zero". That means in regards to our example, that there is no such thing as no temperature. With interval data, we can add and subtract, but we cannot multiply, divide or calculate ratios. Because there is no true zero, a lot of descriptive and inferential statistics can't be applied.

Ratio Data:

- Ratio values are also ordered units that have the same difference. Ratio values are the same as interval values, with the difference that they do have an absolute zero. Good examples are height, weight, length etc.

- If there is a true meaning for 0, then we can call it ratio data type. Ratio Data variables can be added, subtracted, multiplied, or divided.
- Ratio variables can be discrete (i.e. expressed in finite, countable units) or continuous (potentially taking on infinite values). Here are some examples of ratio data:
    - Weight in grams (continuous)
    - Number of employees at a company (discrete)
    - Speed in miles per hour (continuous)
    - Age in years (continuous)
    - Income in dollars (continuous)

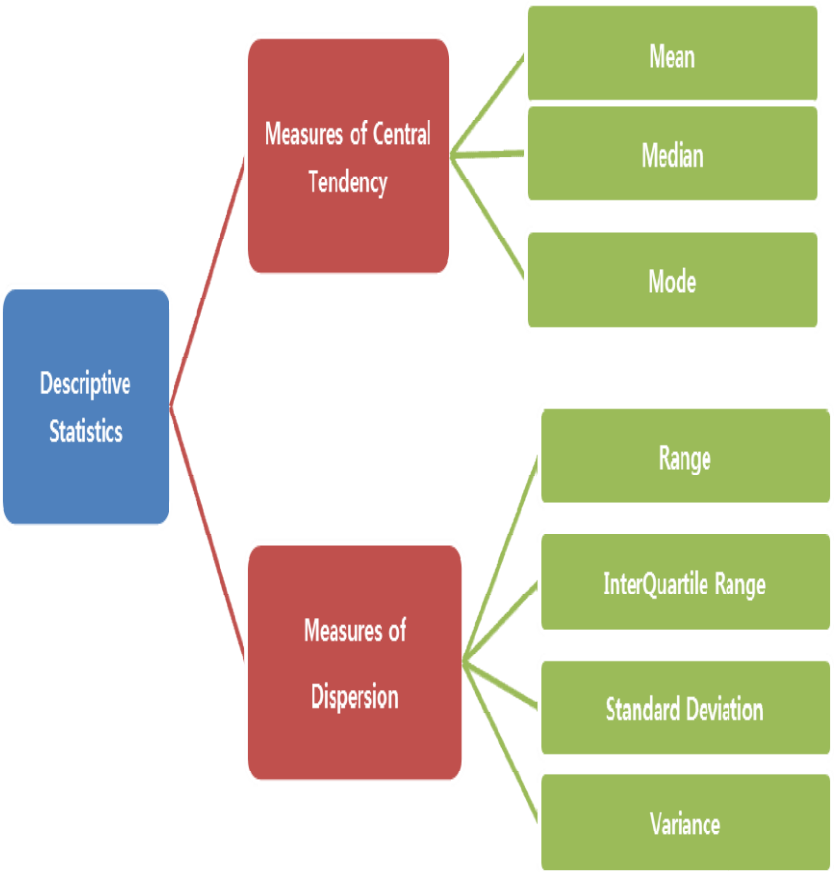
Types of data on the basis of measurement

Scale	True Zero	Equal Intervals	Order	Category	Example
Nominal	No	No	No	Yes	Marital Status, Sex, Gender, Ethnicity
Ordinal	No	No	Yes	Yes	Student Letter Grade, NFL Team Rankings
Interval	No	Yes	Yes	Yes	Temperature in Fahrenheit, SAT Scores, IQ, Year
Ratio	Yes	Yes	Yes	Yes	Age, Height, Weight

NOTE: In general, nominal and ordinal attributes are discrete. On the other hand, interval and ratio attributes are continuous, barring a few exceptions, e.g. 'count' attributes.

Descriptive statistics which can be obtained using interval data include:

1. **Central tendency**: Mode, median, and mean
2. **Dispersion**: Range, Standard deviation and Variance



Central tendency:

➢ Central tendency is defined as "the statistical measure that identifies a single value as representative of an entire distribution.

➢ Central tendency is a one-number summary of the data that typically describes the center of the data. This one-number summary is of three types i.e., measures of central tendency.

- Mean
- Median
- Mode

1. Mean:

Mean is the arithmetic average of a data set. This is found by adding the numbers in a data set and dividing by the number of observations in the data set.

- The Mean

$$\mu_x = \sum_{i=1}^{N} \frac{x_i}{N} = \frac{x_1 + x_2 + \ldots + x_N}{N}$$

Where,

∑-- represents the summation
x -- represents observations
N -- represents the no. of observations.

- It is also known as the **arithmetic mean**, and it is the most common measure of central tendency.
- Arrangement or order of the numbers does not affect calculation for mean.

2. Median:

- The median is the middle number in a data set when the numbers are listed in either ascending or descending order.
- If the total number of observations (n) is an **odd number**, then the formula is given below:

$$Median = \left(\frac{n+1}{2}\right)^{th} observation$$

- If the total number of the observations (n) is an **even number**, then the formula is given below:

$$Median = \frac{\left(\frac{n}{2}\right)^{th} observation + \left(\frac{n}{2}+1\right)^{th} observation}{2}$$

3. Mode:

- The mode is the value that occurs the most often in a data set.
- Mode is the most frequently occurring sample.

Problem 1)

Find the mean, median, mode, and range for the following list of values:
13, 18, 13, 14, 13, 16, 14, 21, 13.

Solution:

Data given : 13, 18, 13, 14, 13, 16, 14, 21, 13

The **mean** is the average.

Mean = {13 + 18 + 13 + 14 + 13 + 16 + 14 + 21 + 13} / {9} = **15**

The **median** is the middle value, so **rewrite** the list in **ascending order** as given below:

13, 13, 13, 13, 14, 14, 16, 18, 21

There are nine numbers in the list, so the middle one will be

{9 + 1} / {2} = {10} / {2} = 5 = 5th number

Hence, the median is **14**.

The **mode** is the number that is repeated more often than any other,

So **13** is the mode.

Difference between Mean and Median	
Mean	**Median**
The average arithmetic of a given set of numbers is called Mean.	The method of separating the higher sample with the lower value, usually from a probability distribution is termed as the median
The application for the mean is for normal distributions	The primary application for the median is skewed distributions.
There are a lot of external factors that limit the use of Mean.	It is much more robust and reliable for measuring the data for uneven data.
Mean can found by calculated by adding all the values and dividing the total by the number of values.	Median can be found by listing all the numbers available in the set in arranging the order and then finding the number in the centre of the distribution.
Mean is considered as an arithmetic average.	Median is considered as a positional average.
It is highly sensitive to outlier data	It is not much sensitive to the outlier data.
It defines the central value of the data set.	It defines the centre of gravity of the midpoint of the data set.

Understanding Data Distribution & measures central tendency:

Let's consider a simple dataset to understand in better way.

Dataset1 = [1,2,2,3,3,3,4,4,4,4,5,5,5,5,5,6,6,6,6,6,6,7,7,7,7,7,7,7,8,8,8,8,8,8,8,8,9,9,9,9,9,9,9,10,10,10,10,10,10,11,11,11,11,11,12,12,12,12,13,13,13,14,14,15].

Example 1:

```
import warnings
warnings.filterwarnings('ignore')
import numpy as np
from scipy import stats
Dataset1 = [1,2,2,3,3,3,4,4,4,4,5,5,5,5,5,6,6,6,6,6,6,7,7,7,7,7,7,7,8,8,8,8,8,8,8,8,9,9,9,9,9,9,9,10,10,10,10,10,10,11,11,11,11,11,12,12,12,12,13,13,13,14,14,15]
```

x1=np.mean(Dataset1)
print(" The Average of Dataset is:",x1)
x2=np.median(Dataset1)
print(" The middle value of Dataset is:",x2)
x3 = stats.mode(Dataset1)
print(" Most frequently used value of Dataset is ",x3)

Let's understand with a plot:

import numpy as np

import matplotlib.pyplot as plt

Dataset1=[1,2,2,3,3,3,4,4,4,4,5,5,5,5,5,6,6,6,6,6,6,7,7,7,7,7,7,7,8,8,8,8,8,8,8,8,9,9,9,9,9,9,9,10,

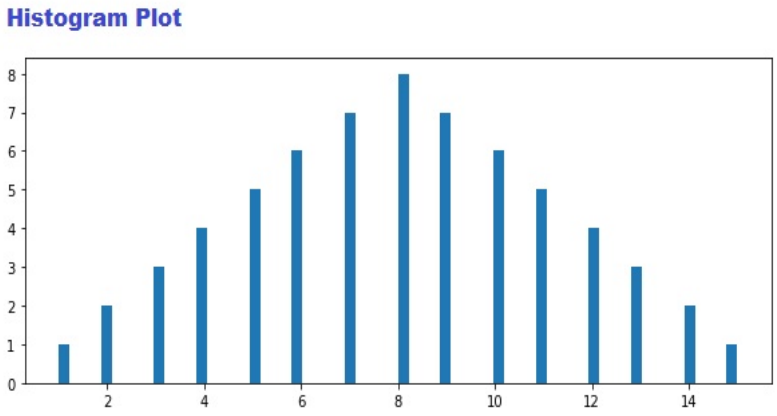
10,10,10,10,10,11,11,11,11,11,12,12,12,12,13,13,13,14,14,15]

y=Dataset1

plt.figure(figsize=(10, 4))

plt1 = plt.hist(Dataset1,bins=64)

plt.show()



Observation 1

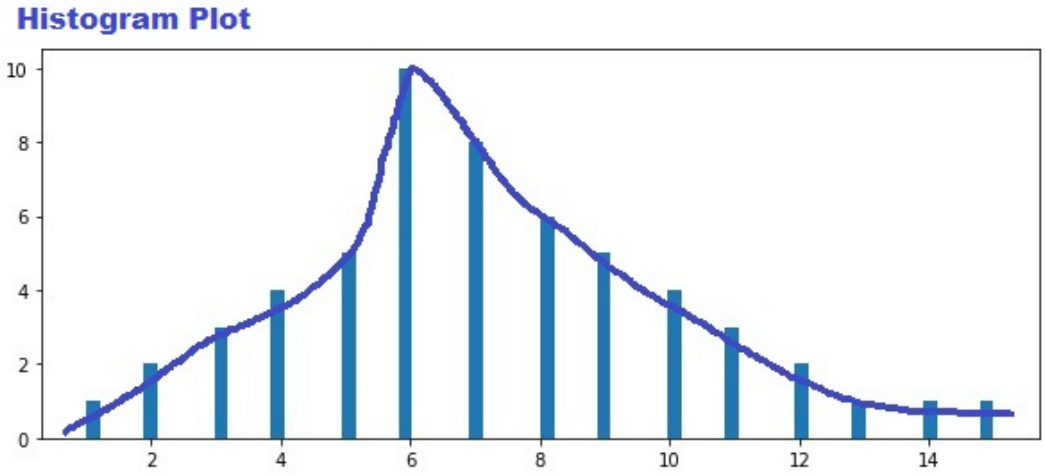
- If mean, mode and median are exactly the same then the data is distributed **symmetrically** i.e., **normal distribution**.
- In a normal distribution, data is symmetrically distributed with no skew. Most values cluster around a central region.

Example 2:

```
import warnings
warnings.filterwarnings('ignore')
import numpy as np
from scipy import stats
Dataset1 =
[1,2,2,3,3,3,4,4,4,4,5,5,5,5,5,6,6,6,6,6,6,6,6,6,6,7,7,7,7,7,7,7,7,8,8,8,8,8,8,9,9,9,9,9,10,10,10,10,
11,11,11,12,12,13,14,15]
x1=np.mean(Dataset1)
print(" The Average of Dataset is:",x1)
x2=np.median(Dataset1)
print(" The middle value of Dataset is:",x2)
x3 = stats.mode(Dataset1)
print(" Most frequently used value of Dataset is ",x3)
```

Understanding with Plot

```
import numpy as np
import matplotlib.pyplot as plt
Dataset1 =
[1,2,2,3,3,3,4,4,4,4,5,5,5,5,5,6,6,6,6,6,6,6,6,6,6,7,7,7,7,7,7,7,7,8,8,8,8,8,8,9,9,9,9,9,10,10,10,10,
11,11,11,12,12,13,14,15]
y=Dataset1
plt.figure(figsize=(10, 4))
plt1 = plt.hist(Dataset1,bins=64)
plt.show()
```



Observation 2

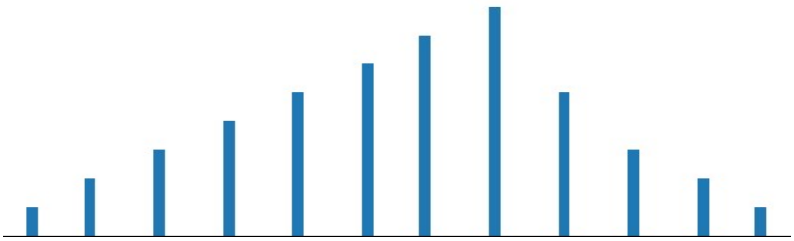
- If **mode < median < mean** then positively skewed distribution,
- In a positively skewed distribution, there's a cluster of lower scores and a spread out tail on the right.
- It is Observed in histogram, distribution is skewed to the right, and the central tendency of dataset is on the lower end of possible scores.

Example 3

```
import warnings
warnings.filterwarnings('ignore')
import numpy as np
from scipy import stats
Dataset1 =
[1,2,2,3,3,3,4,4,4,4,5,5,5,5,5,6,6,6,6,6,6,7,7,7,7,7,7,7,8,8,8,8,8,8,8,8,8,9,9,9,9,9,10,10,10,11,11,12
]
x1=np.mean(Dataset1)
print(" The Average of Dataset is:",x1)
x2=np.median(Dataset1)
print(" The middle value of Dataset is:",x2)
x3 = stats.mode(Dataset1)
print(" Most frequently used value of Dataset is ",x3)
```

Plotting:

```
import numpy as np
import matplotlib.pyplot as plt
Dataset1 =
[1,2,2,3,3,3,4,4,4,4,5,5,5,5,5,6,6,6,6,6,6,7,7,7,7,7,7,7,8,8,8,8,8,8,8,8,9,9,9,9,9,10,10,10,11,11,12
]
y=Dataset1
plt.figure(figsize=(10, 4))
plt1 = plt.hist(Dataset1,bins=64)
plt.show()
```



Observation 3

- In a negatively skewed distribution, **mean < median < mode**.
- In a negatively skewed distribution, there's a cluster of higher scores and a spread out tail on the left.
- In this histogram, distribution is skewed to the left, and the central tendency of dataset is towards the higher end of possible scores.

When should you use the mean, median or mode?

- For **normally distributed data**, all **three measures** of central tendency will give you the

same answer so they can all be used but **mean** is the most widely used for normal distributed data.

- In **skewed distributions**, the **median** is the best measure because it is unaffected by extreme outliers or non-symmetric distributions of scores. The mean and mode can vary in skewed distributions.
- The **mode** can be of used for any level measurement, but it's most meaningful for nominal and ordinal levels.
- The **median** can only be used on data that can be ordered – that is, from ordinal, interval and ratio levels of measurement.
- The **mean** can only be used on interval and ratio levels of measurement because it requires equal spacing between adjacent values or scores in the scale.

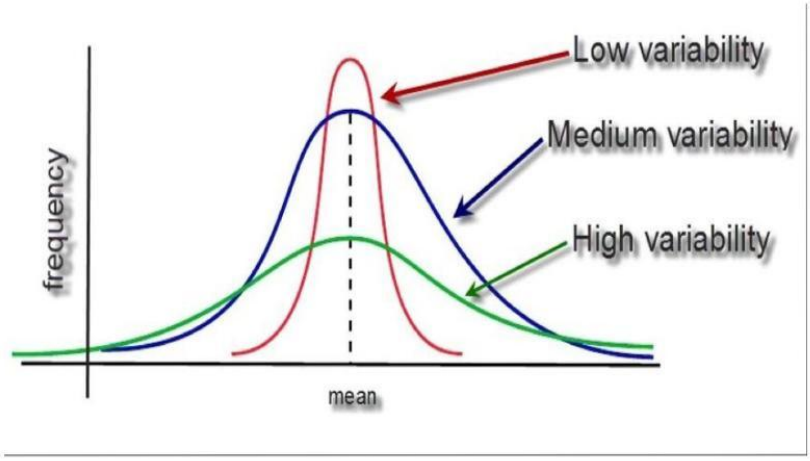
Levels of measurement	Examples	Measure of central tendency
Nominal	•Gender: Male , Female •Nationality: Indian	•Mode
Ordinal	•Assigning Ranks •IQ Score	•Mode •Median
Interval and ratio	•Reaction time •Test score •Temperature	•Mode •Median •Mean

Measures of Dispersion (Variability)

Data variability also known as spread or dispersion, refers to how spread out a set of data is. Variability gives users a way to describe how much data sets vary and allows users to use statistics to compare their data to other sets of data.

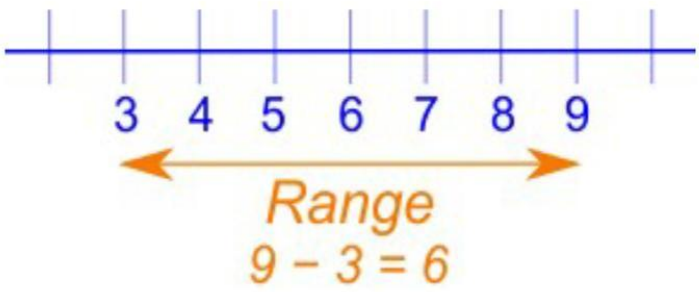
The four main ways to describe variability in a data set are:

- Range
- Variance
- Standard deviation.
- Interquartile range



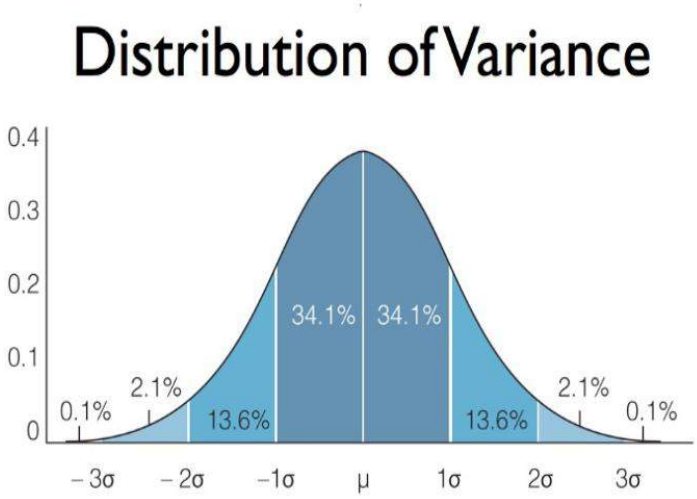
1. Range:

- In statistics, the range is the spread of your data from the lowest to the highest value in the distribution.
- It is a commonly used measure of **variability**.
- The range is calculated by subtracting the lowest value from the highest value.
- Large range means high variability, a small range means low variability in a distribution.
- For example, if the given data set is {2, 5, 8, 10, 3}, then the range will be 10 – 2 = 8.



2. Variance:

- Variance is a measure of how data points differ from the **mean**.
- A small variance indicates that the data points tend to be very close to the mean.
- A high variance indicates that the data points are very spread out from the mean.
- Therefore, it is called a **measure of spread of data from mean**.
- Variance is the average of the squared distances from each point to the mean.



- Variance is the sum of squares of differences between all numbers and mean (μ). The mathematical formula for variance is as follows

$$\text{Variance} = \sigma^2 = \frac{\Sigma(x_i - \mu)^2}{n}$$

where: *xi = Each Sample*
μ = Mean
n= total number of samples

Let's say we have values: 5, 7, 9, and 3.

1. Calculate the mean

$$\bar{x} = \frac{\sum x}{n} \Rightarrow \bar{x} = \frac{x1+x2+x3.....xn}{n}$$

$$\textbf{Mean} = \frac{5+7+9+3}{4}$$

$$= 6$$

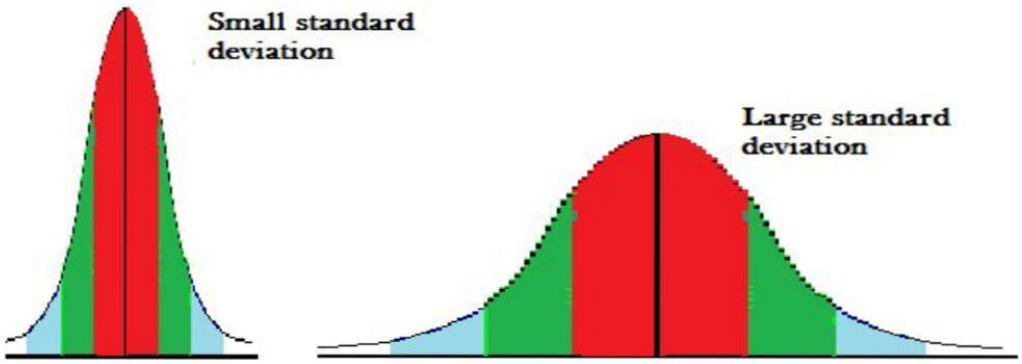
2. Subtract mean from all observation to find the distance of all observation from mean.

$$\textbf{Variance} = \frac{(5-6)^2 + (7-6)^2 + (9-6)^2 + (3-6)^2}{4}$$

$$= \frac{1+1+9+9}{4} \Rightarrow 5$$

3. Standard deviation

- Standard deviation is a squared root of the variance to get original values.
- Low standard deviation means data are clustered around the mean, and high standard deviation indicates data are more spread out.



Population SD:

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2}$$

Sample SD:

$$s = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \bar{x})^2}$$

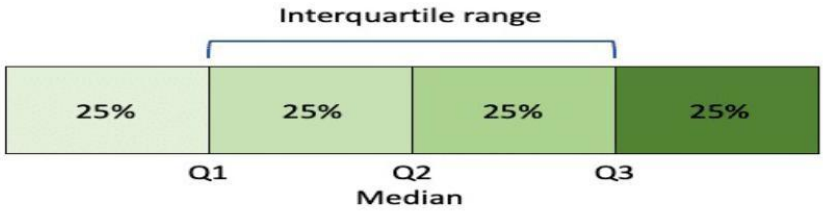
Finding measures of Variability on simple Data

```
# Importing the NumPy module
import numpy as np
# Taking a list of elements
list = [2, 4, 4, 4, 5, 5, 7,25]
X= list
# Calculating variance using var()
x1=np.var(list)
print("varaince of list is :",x1)
range=np.max(X)- np.min(X)
print("range is:",range)
SD =np.std(X)
print("Standard devaition is", SD)
```

```
varaince of list is : 48.0
range is: 23
Standard devaition is 6.928203230275509
```

4. Interquartile Range:

- Interquartile Range, (IQR) is a property that is used to measure variability.
- The IQR divides a dataset into quartiles. These quartiles store data after the data gets sorted in ascending order and split into 4 equal parts. The first, second, and third quartiles are called Q1, Q2, and Q3. These quartiles are the values that separate the 4 equal parts.



The following is the percentile distribution of the data amongst Q1, Q2, and Q3-

- 25^{th} percentile of the data is represented in Q1
- 50^{th} percentile of the data is represented in Q2
- 75^{th} percentile of data is represented in Q3

The measurement of IQR gives an insight into the width of distribution as most of the points of the dataset are contained in this range.

- In a dataset that contains even or odd elements of data points, then-
- Q2 is the median
- Q1 is the median of *x* smallest points of data i.e., lower half of data
- Q3 is the median of *x* highest points of data i.e., Upper half of data

Problem) Find the median, lower quartile, upper quartile, interquartile range and range of the following numbers.
12, 5, 22, 30, 7, 36, 14, 42, 15, 53, 25, 65

Solution:

- Find the median, lower quartile, upper quartile, interquartile range and range of the following numbers.
  12, 5, 22, 30, 7, 36, 14, 42, 15, 53, 25, 65
- **Solution:**
  First, arrange the data in ascending order:

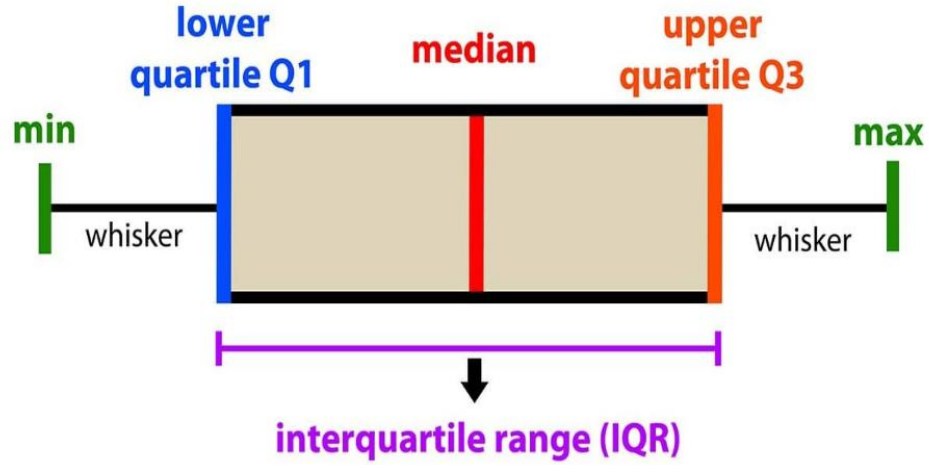
5, 7, 12, 14, 15, 22, 25, 30, 36, 42, 53, 65

lower quartile or first quartile — median or second quartile — upper quartile or third quartile

- **Lower quartile or first quartile(Q1) =** $\frac{12+14}{2} = 13$
- **Median or second quartile(Q2) =** $\frac{22+25}{2} = 23.5$
- **Upper quartile or third quartile(Q3) =** $\frac{36+42}{2} = 39$
- **Interquartile range** = Upper quartile − lower quartile
  = 39 − 13 = 26
- **Range** = largest value − smallest value
  = 65 − 5 = 60

BOX PLOT:

- The box and whisker plot, sometimes simply called the box plot, is a type of graph that help visualize the five-number summary.
- Box plots are a standardized way of displaying the distribution of data based on a five number summary ("minimum", first quartile (Q1), median, third quartile (Q3), and "maximum").
- It doesn't show the distribution in as much detail as histogram does, but it's especially useful for indicating whether a distribution is skewed and whether there are potential unusual observations (outliers) in the data set.
- A box plot is ideal for comparing distributions because the center, spread and overall range are immediately apparent.

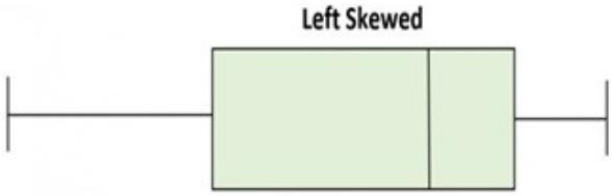


In a box and whisker plot:

- The left and right sides of the box are the lower and upper quartiles. The box covers the interquartile interval, where 50% of the data is found.
- The vertical line that split the box in two is the median. Sometimes, the mean is also indicated by a dot or a cross on the box plot.
- The whiskers are the two lines outside the box, that go from the minimum to the lower quartile (the start of the box) and then from the upper quartile (the end of the box) to the maximum.
- Box plot can be displayed in both vertical & horizontal pattern.
- The box plot shape will show if a data set is normally distributed or skewed.
- When the median is in the middle of the box, and the whiskers are about the same on both sides of the box, then the distribution is symmetric.

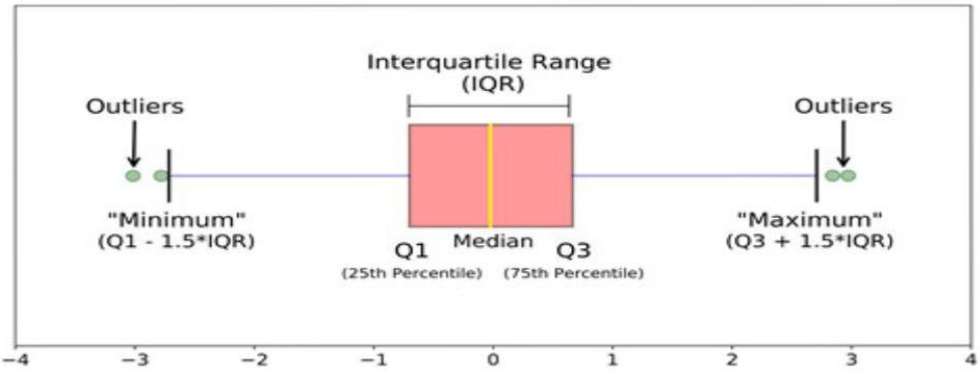
- When the median is closer to the bottom of the box, and if the whisker is shorter on the lower end of the box, and longer to the upper side then the distribution is positively skewed (skewed right).

- When the median is closer to the top of the box, and if the whisker is shorter on the upper end of the box, then the distribution is negatively skewed (skewed left).



Outliers:

- Outliers are those data points that are significantly different from the rest of the dataset.
- They are often abnormal observations that skew the data distribution, and arise due to incorrect data entry or false observations.
- The outlier formula — also known as the 1.5 IQR rule — is a Thumb rule used for identifying outliers.
- The outlier formula designates outliers based on an upper and lower boundary .i.e.,
- Anything above Q3 + 1.5 x IQR is an outlier
- Anything below Q1 - 1.5 x IQR is an outlier



Finding Outliers Using Boxplot:

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
speed = [60,66,87,88,90,66,103,67,94,78,77,85,66,66,66,65,200,500,800,]
y=speed
plt.figure(figsize=(10, 4))
plt1 = sns.boxplot(speed)
plt.show()
```

Outlier Detection Methods:

1. Standard Deviation Method
2. Z-Score Method
3. IQR Method

Understanding outliers detection with simple Data

```
data1 = [11,12,24,11,14,12,15,11,12,13,12,13,14,102,12,11,13,14,107,15,11,12,14,108,11,14,16,60,140]
len(data1)
```

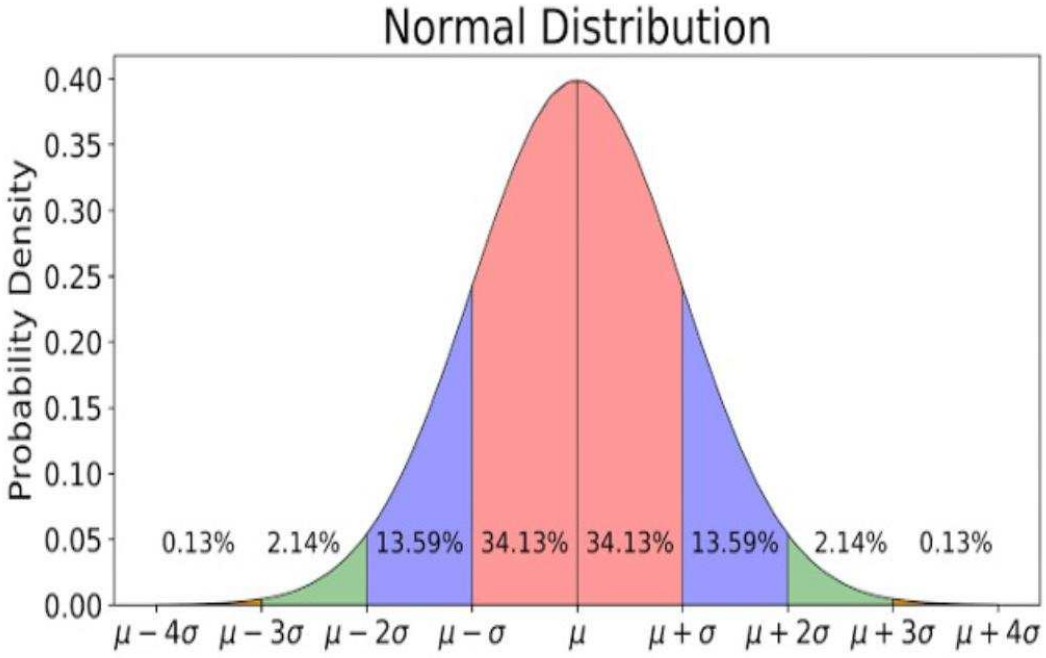
29

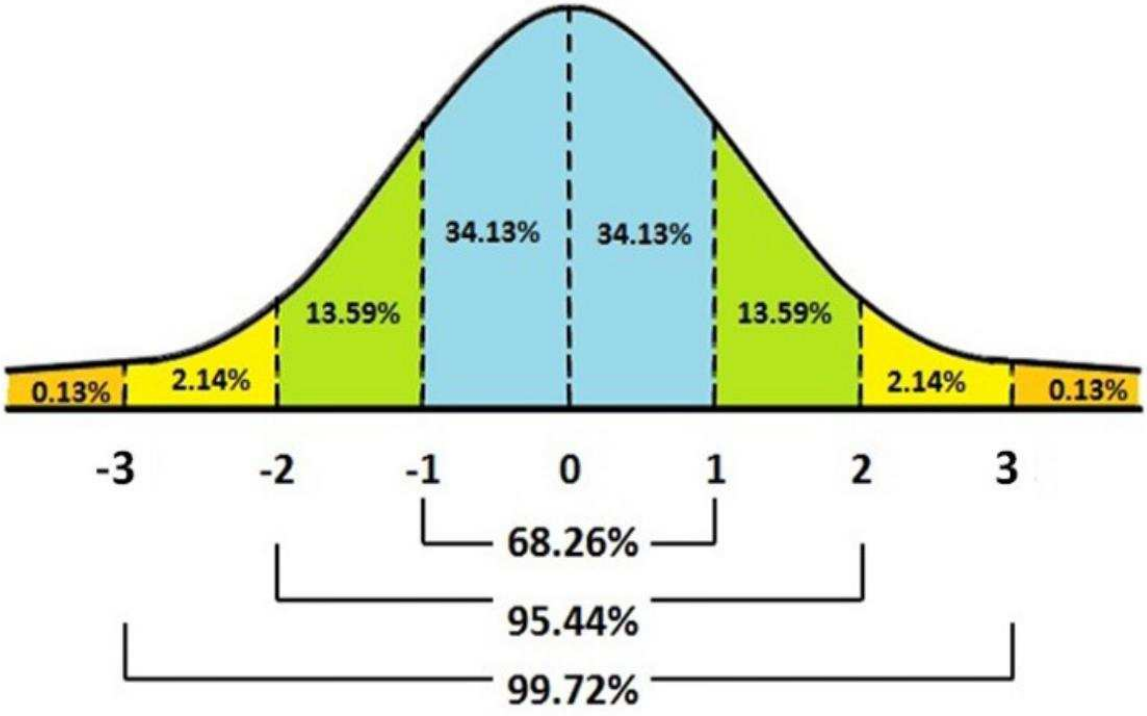
+ Code + Markdown

```
X1=np.mean(data1)
print("The mean value is",X1)
SD =np.std(data1)
print("Standard devaition is", SD)
```

```
The mean value is 28.75862068965517
Standard devaition is 35.76325910128688
```

1. Standard Deviation Method





Outlier detection using Normal Standard Deviation method

```
import numpy as np
lower_limit = np.mean(data) - 1* np.std(data1)
upper_limit = np.mean(data) + 1* np.std(data1)
print(lower_limit, upper_limit)
```

```
-12.281777619805396 59.24474058276836
```

```
outliers = []
def detect_outliers(data1):
  for i in data1:
    if (i<lower_limit or i>upper_limit):
      outliers.append(i)
  return outliers

outliers = detect_outliers(data1)
outliers
```

```
import numpy as np
lower_limit = np.mean(data1) - 2* np.std(data1)
upper_limit = np.mean(data1) + 2* np.std(data1)
print(lower_limit, upper_limit)
```

```
-42.76789751291858    100.28513889222893
```

```
outliers = []
def detect_outliers(data1):
  for i in data1:
    if (i<lower_limit or i>upper_limit):
      outliers.append(i)
  return outliers

outliers = detect_outliers(data1)
outliers
```

```
[102, 107, 108, 140]
```

```
import numpy as np
lower_limit = np.mean(data1) - 3* np.std(data1)
upper_limit = np.mean(data1) + 3* np.std(data1)
print(lower_limit, upper_limit)
```

```
-78.53115661420546      136.0483979935158
```

```
outliers = []
def detect_outliers(data):
  for i in data:
    if (i<lower_limit or i>upper_limit):
      outliers.append(i)
  return outliers

outliers = detect_outliers(data1)
outliers
```

```
[140]
```

2. Z Score Method:

- Z score is also called standard score.
- Z score tells how many standard deviations away a data point is from the mean. i.e.,

Z score detects outliers based threshold value

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

```
outliers = []
def detect_outliers(data1):
  threshold = 2.0
  mean = np.mean(data1)
  std = np.std(data1)

  for x in data1:
    z_score = np.abs((x - mean)/std)
    if np.ceil(z_score) > threshold:
      outliers.append(x)
  return outliers
```

+ Code + Markdown

```
outliers_pts = detect_outliers(data1)
outliers_pts
```

```
[102, 107, 108, 140]
```

3. IQR (Inter Quartile Range)

IQR (Inter Quartile Range) Inter Quartile Range approach to finding the outliers is the most commonly used and most trusted approach used in the research field.
IQR = Quartile3 – Quartile1

To define the outlier base value is defined above and below datasets normal range namely Upper and Lower bounds, define the upper and the lower bound (1.5*IQR value is considered) :

upper = Q3 +1.5*IQR lower = Q1 – 1.5*IQR

In the above formula as according to statistics, the 0.5 scale-up of IQR (new_IQR = IQR + 0.5*IQR) is taken, to consider all the data between 2.7 standard deviations in the Gaussian Distribution.

Example1:

Consider following dataset
2, 19, 22, 27, 29, 30, 32, 35, 52, 59

Median(Q2)=(29+30)/2 = 29.5

Median of Lower half of data(Q1) 2, 9, 2, 27, 29

Q1=22

Median of Upper half of data(Q3)
30, 32, 35, 52, 59

Q3=35

Inter-quartile range, IQR.

Inter-quartile range is the difference between Q3 and Q1.

IQR = Q3 - Q1 = 35 - 22 = 13

Find the upper boundary

Upper boundary = Q3 + 1.5 IQR = 35 + (1.5)(13) = 54.5

Find the lower boundary

Lower boundary = Q1 - 1.5 IQR = 22 - (1.5)(13) = 2.5
Identify the outliers

The outliers are any data points that lie above the upper boundary or below the lower boundary. In this case, the outliers are 2 and 59.

Example2: The runs scored by a cricket team in a league of 12 matches – 100,120,110,150,110,140,130,170,120,220,140,110. Draw box plot

STEP 1) To draw a box plot for the given data first we need to arrange the data in ascending order Ascending Order -

100,110,110,110,120,120,130,140,140,150,170,220

STEP 2) Find the minimum, first quartile, median, third quartile and IQR

A) Median(Q2) 100,110,110,110,120,120,130,140,140,150,170,220

Median (Q2) = (120+130)/2 = 125 ; Since there were even values

B) Find the First Quartile we take the first six values and find their median. 100,110,110,110,120,120

Q1 = (110+110)/2 = 110

C) For the Third Quartile, we take the next six and find their median. 130,140,140,150,170,220 Q3 = (140+150)/2 = 145

Note: If the total number of values is odd then we exclude the Median while calculating Q1 and Q3. Here since there were two central values we included them.

D) Calculate the Inter Quartile Range. IQR = Q3-Q1 = 145-110 = 35

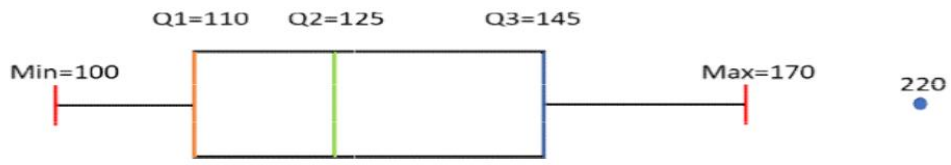
STEP 3: Calculate the Upper and Lower Limits to find the minimum and maximum values and also the outliers if any.

Lower Limit = Q1-1.5*IQR = 110-1.5*35 = 57.5 Upper Limit = Q3+1.5*IQR = 145+1.5*35 = 197.5

So the minimum and maximum between the range [57.5,197.5] for our given data are –
Minimum = 100
Maximum = 170

The outliers which are outside this range are –
Outliers = 220

we have all the information, so we can draw the box plot which is as below-



Example3:

Find Q1, Q2, and Q3 for the following data set. Identify any outliers, and draw a box-and- whisker plot.

{5,40,42,46,48,49,50,50,52,53,55,56,58,75,102}

STEP 1: Data set has 15 values, arranged in increasing order. 5,40,42,46,48,49,50, 50, 52, 53, 55, 56, 58, 75, 102

STEP 2: Find the minimum, first quartile, median, third quartile and IQR

A) Median(Q2)

5,40,42,46,48,49,50, 50, 52, 53,55, 56, 58, 75, 102

Median (Q2) = 50;

B) Find the First Quartile we take the first six values and find their median. 5,40,42,46,48,49,50

Q1 = 46

C) For the Third Quartile, we take the next six and find their median. 52,53,55,56,58,75,102

Q3 = 56

D) Calculate the Inter Quartile Range. IQR = Q3-Q1 = 56-46 = 10

STEP 3: Calculate the Upper and Lower Limits to find the minimum and maximum values and also the outliers if any.

Lower Limit = Q1-1.5*IQR = 46−15=31

Upper Limit = Q3+1.5*IQR = 56+15=71

So the minimum and maximum between the range [31,71] for our given data are –

Minimum = 31

Maximum = 71

The outliers which are outside this range are –

Outliers = 5, 75, 102

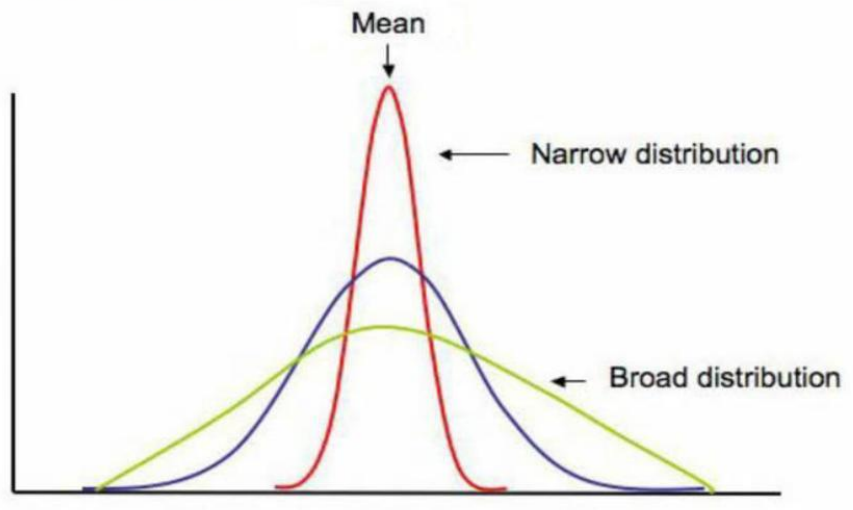
40 and 58 are shown as the ends of the whiskers, with the outliers plotted separately.

Mean absolute deviation

The mean absolute deviation of a dataset is the average distance between each data point and the mean. It gives us an idea about the variability in a dataset.

The measure of spread represents the amount of dispersion in a data-set. i.e how spread-out are the values of data-set around the central value (example- mean/mode/median). It tells how far away the data points tend to fall from the central value.

- The lower value of the measure of spread reflects that the data points are close to the central value. In this case, the values in a data-set are more consistent.
- Further, the distance of the data points from the central-value, the greater is the spread. whereas here, the values are not much consistent.



Using the above diagram, we can infer that the narrow distribution represents a lower spread, and the broad distribution represents a higher spread.

Formula for mean absolute deviation

$$\text{MAD} = \frac{\sum |x_i - \bar{x}|}{n}$$

- Here || gives the absolute value that means all negative deviation (distance) made positive.

Example 1) Find the mean absolute deviation of the following data set: 10,15,15,17,18,21

Solution: Mean (X_i) = (10+15+15+17+18+21)/6 =16

Data points	Distance from mean($\|X_i$-X|)
10	|10-16|=6
15	|15-16|=1
15	|15-16|=1
17	|17-16|=1
18	|18-16|=2
21	|21-16|=5

average of all the absolute values=(6+1+1+1+2+5)/6=16/6=2.67

Example 2) Find the mean absolute deviation of the following data set: 26, 46, 56, 45, 19, 22, 24.

Solution:

Given set of data is:

26, 46, 56, 45, 19, 22, 24

Mean = (26 + 46 + 56 + 45 + 19 + 22 + 24)/7 = 238/7 = 34

Data points	Distance from mean(|Xi-X|
26	|26-34|=8
46	|46-34|=12
56	|56-34|=22
45	|45-34|=11
19	|19-34|=15
22	|22-34|=12
24	|24-34|=10

Average of all the absolute values:

(8+12+22+11+15+12+10)/7=90/7=12.857

Therefore, the mean absolute deviation of the given data set is 12.857.

Covariance:

Covariance signifies the direction of the linear relationship between the two variables. By direction we mean if the variables are directly proportional or inversely proportional to each other. (Increasing the value of one variable might have a positive or a negative impact on the value of the other variable).

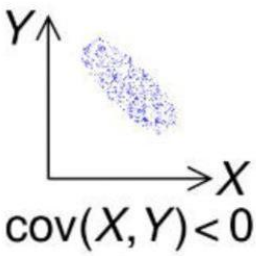
The value of covariance between 2 variables is achieved by taking the summation of the product of the differences from the means of the variables as follows:

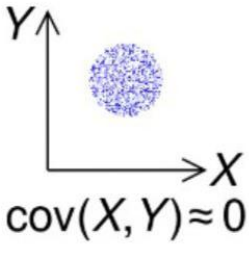
Where,

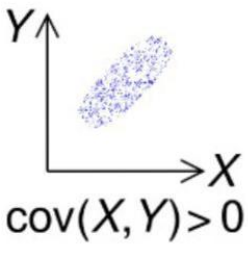
$$\mathrm{COV}_{x,y} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{n-1}$$

x = the independent variable
y = the dependent variable
n = number of data points in the sample
$\bar{x}$ = the mean of the independent variable x
$\bar{y}$ = the mean of the dependent variable y

The upper and lower limits for the covariance depend on the variances of the variables involved. These variances, in turn, can vary with the scaling of the variables. Even a change in the units of measurement can change the covariance. Thus, covariance is only useful to find the direction of the relationship between two variables and not the magnitude. Below are the plots which help us understand how the covariance between two variables would look in different directions.







Example:

X	Y
10	40
12	48
14	56
8	32

Step 1: Calculate Mean of X and Y

Mean of X : (10+12+14+8) / 4 = 11

Mean of Y : (40+48+56+32) / 4 = 44

Step 2: Substitute the values in the formula

X	Y	$x_i - \bar{x}$	$y_i - \bar{y}$
10	40	10 – 11 = -1	40 – 44 = – 4
12	48	12 – 11 = 1	48 – 44 = 4
14	56	14 – 11 = 3	56 – 44 = 12
8	32	8 – 11 = -3	32 – 44 = -12

Substitute the above values in the formula

Cov(x,y) = (-1) (-4) +(1)(4)+(3)(12)+(-3)(-12) / 4-1

Cov(x,y) = 80/3 = 26.67

Hence, Co-variance for the above data is 26.67.

```
import numpy as np
height = [5.1,5.2,5.3,5.4,5.5]
weight=[60.3,59.2,63.6,88.4,68.7]
print("covariance",np.cov(height,weight))
```

```
covariance [[2.50000e-02 1.15000e+00]
 [1.15000e+00 1.43183e+02]]
```

```
print("covariance",np.cov(height,weight)[0,1])
```

```
covariance 1.1500000000000026
```

Correlation:

Correlation analysis is a method of statistical evaluation used to study the strength of a relationship between two numerically measured continuous variables. It not only shows the kind of relation (in terms of direction) but also how strong the relationship is. Thus, we can say the correlation values have standardized notions, whereas the covariance values are not standardized and cannot be used to compare how strong or weak the relationship is because the magnitude has no direct significance. It can assume values from -1 to +1.

To determine whether the covariance of the two variables is large or small, we need to assess it relative to the standard deviations of the two variables.

To do so we have to normalize the covariance by dividing it with the product of the standard deviations of the two variables, thus providing a correlation between the two variables.

The main result of a correlation is called the correlation coefficient.

$$Correlation = \frac{Cov\ (x, y)}{\sigma x * \sigma y}$$

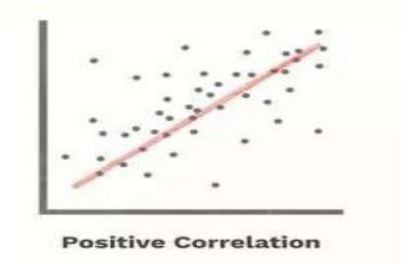
where:

- cov is the covariance
- σ_X is the standard deviation of X
- σ_Y is the standard deviation of Y

Types of correlation

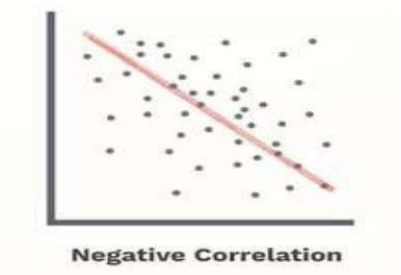
Positive correlation

If with increase in random variable A, random variable B increases too, or vice versa.



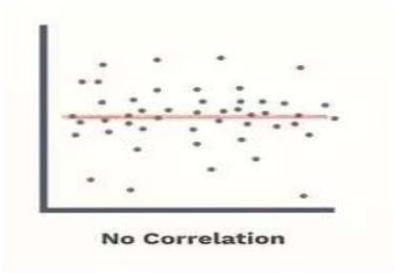
Negative correlation

If increase in random variable A leads to adecrease in B, or vice versa.



No correlation –

When both the variables are completely unrelated and change in one leads to no changein other.



Example

X	Y
10	40
12	48
14	56
8	32

Step 1: Calculate Mean of X and Y

Mean of X : 10+12+14+8 / 4 = 11

Mean of Y = 40+48+56+32/4 = 44

Step 2: Substitute the values in the formula

xi –$\bar{x}$	yi – $\bar{y}$
10 – 11 = -1	40 – 44 = – 4
12 – 11 = 1	48 – 44 = 4
14 – 11 = 3	56 – 44 = 12
8 – 11 = -3	32 – 44 = -12

Step 3: Now substitute the obtained answer in Correlation formula

Substitute the above values in the formula

Cov(x,y) = (-1) (-4) +(1)(4)+(3)(12)+(-3)(-12) / 4-1

Cov(x,y) = 80/3 = 26.67

Hence, Co-variance for the above data is 26.67.

$$Correlation = \frac{Cov\ (x, y)}{\sigma x * \sigma y}$$

- Before substitution we have to find standarddeviation of x and y
- Let's take the data for X as mentioned in the tablethat is 10,12,14,8
- To find standard deviation

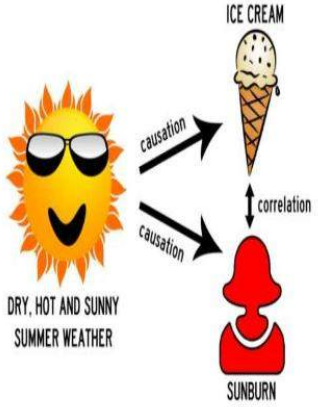
$$s = \sqrt{\frac{\Sigma\ (X - \bar{x})^2}{n - 1}}$$

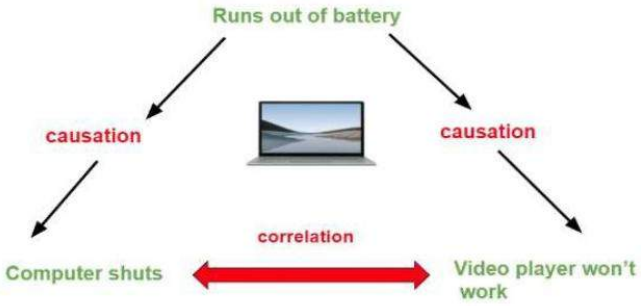
Causation

Causation means that changes in one variable bringsabout changes in the other; there is a cause-and- effect relationship between variables. The two variables are correlated with each other andthere is also a causal link between them.

Correlation and Causation can exist at the same time also, so definitely correlation doesn't imply causation.Below example is to show this difference more clearly

No battery in computer causes computer to shut and also causes video player to stop ,shows causality of battery over laptop and video player. The moment computer shuts, video player also shuts shows both are correlated. More specifically positively correlated.





Step 1: Find the mean of x that is $\bar{x}$
10+14+12+8 /4 = 11

Step 2: Find each number deviation: Subtract each scorewith mean to get mean deviation
10 – 11 = -1
12 – 11 = 1
14 – 11 = 3
8 – 11 = -3

Step 3: Square the mean deviation obtained

Mean Deviation Value	Squared Mean Deviation Value
-1	1
1	1
3	9
-3	9

Step 4: Sum the squares
1+1+9+9 = 20

Step5: Find the variance
Divide the sum of squares with n-1 that is 4-1 = 3
20 /3 = 6.6

Step 6: Find the square root
Sqrt of 6.6 = 2.581

Therefore, Standard Deviation of x = 2.581
Find for Y using same method

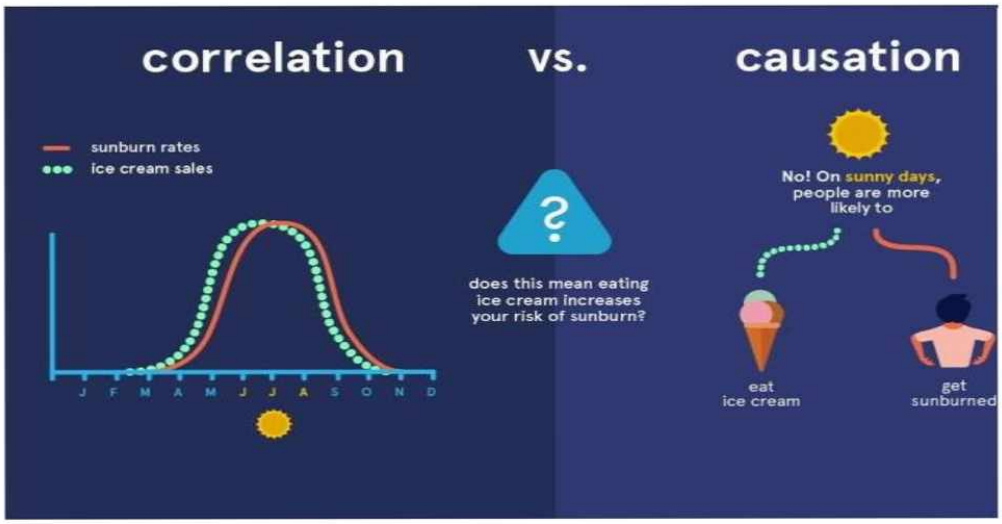
The Standard Deviation of y = 10.29
Correlation = 26.67 /(2.581 * 10.29)
Correlation = 1.0041

```
height = [5.1,5.2,5.3,5.4,5.5]
weight=[60.3,59.2,63.6,88.4,68.7]
print("correlation",np.corrcoef(height,weight))
```

```
correlation [[1.         0.60782997]
 [0.60782997 1.        ]]
```

```
print("correlation",np.corrcoef(height,weight)[0,1])
```

```
correlation 0.6078299663324911
```


correlation
vs.
causation
sunburn rates
ice cream sales
J F M A M J J A S O N D
?
does this mean eating ice cream increases your risk of sunburn?
No! On sunny days, people are more likely to
eat ice cream
get sunburned

Exploratory Data Analysis (EDA):

It refers to the critical process of performing initial investigations on data so as to discover patterns, to spot anomalies, to test hypothesis and to check assumptions with the help of summary statistics and graphical representations.

Understanding EDA with a Dataset

Domain: Automobile Industry

The Automotive (Automobile) industry is **the industry of automobiles**. It is the industry that designs, develops, manufactures, markets, and sells motor vehicles, and is one of the earth's most important and largest economic sectors by revenue.

Toyota Dataset:

- Toyota Data set contains used cars information about Price, Horse Power,Number of KM travalled,Fuel type,Age,Aotomatic,number of doors,MetalColor,CC & Weight.

Summary of DataFrame

```
import pandas as pd
import numpy as np
stoyota=pd.read_csv('../input/toyatacars/Toyota.csv')
stoyota.head()
stoyota.tail()
```

(i) Checking format of each column

info( ) – returns concise summary of dataframe

```
[23] stoyota.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 1436 entries, 0 to 1435
Data columns (total 10 columns):
Price          1436 non-null int64
Age            1336 non-null float64
KM             1436 non-null object
FuelType       1336 non-null object
HP             1436 non-null object
MetColor       1286 non-null float64
Automatic      1436 non-null int64
CC             1436 non-null int64
Doors          1436 non-null object
Weight         1436 non-null int64
dtypes: float64(2), int64(4), object(4)
memory usage: 123.4+ KB
```

By using info( ), we can see that 'KM' has been read as object instead of integer 'HP' has been read as object instead of integer. 'MetColor' and 'Automatic' have been read as float64 and int64 respectively since it has values 0/1.Ideally, 'Doors' should've been read as int64 since it has values 2, 3, 4, 5. But it has been read as Object.

(ii) Finding unique elements of columns

- ***unique( )*** – returns unique elements of a column

```
[19] stoyota['HP'].unique()

     array(['90', '????', '192', '110', '97', '71', '116', '98', '69', '86',
            '72', '107', '73'], dtype=object)
```

```
[28] import numpy as np
     np.unique(stoyota['KM'])

     array(['1', '10000', '100123', ..., '99865', '99971', '??'], dtype=object)
```

```
[29] stoyota['Doors'].unique()

     array(['three', '3', '5', '4', 'four', 'five', '2'], dtype=object)
```

```
[20] stoyota['Automatic'].unique()

     array([0, 1])
```

```
[18] stoyota['MetColor'].unique()

     array([ 1., nan,  0.])
```

Values 0. , 1. and 0 , 1 made these attributes treated as ‘float’ and ‘int’, but these are categories.

Importing data with other forms of missing values .We need to know how missing values are represented in the dataset in order to makereasonable decisions.

Missing values exists in the form of ‘nan’, ‘??’ and ‘????’ Python, by default replace blank values with ‘nan’. Now, import the data considering other forms of missing values in a dataframe.

Observation :

- ‘KM’ has been read as object instead of float64
- ‘HP’ has been read as object instead of float64
- ‘MetColor’ and ‘Automatic’ have been read as float64 and int64 respectively since it has values 0/1
- Ideally, ‘Doors’ should’ve been read as int64 since it has values 2, 3, 4, 5. But it has been read as Object.
- Missing values present in few variables

(iii) Converting Variable's data types

Converting Variable's data types

- Converting 'MetColor' , 'Automatic' to object data type:
- ***astype( )*** - explicitly converts from one to another data type
- Syntax: dataframe.***astype(****dtype****)***

```
[ ]  stoyota['MetColor']=stoyota['MetColor'].astype('object')
     stoyota.MetColor.dtype

     dtype('O')
```

```
[ ]  stoyota['Automatic']=stoyota['Automatic'].astype('object')
     stoyota.Automatic.dtype

     dtype('O')
```

Recheck the data types

Summary – before converting variable's data type

```
[14] stoyota.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 1436 entries, 0 to 1435
Data columns (total 10 columns):
Price        1436 non-null int64
Age          1336 non-null float64
KM           1421 non-null float64
FuelType     1336 non-null object
HP           1430 non-null float64
MetColor     1286 non-null float64
Automatic    1436 non-null int64
CC           1436 non-null int64
Doors        1436 non-null object
Weight       1436 non-null int64
dtypes: float64(4), int64(4), object(2)
memory usage: 123.4+ KB
```

Summary – after converting variable's data type

```
[37] stoyota.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 1436 entries, 0 to 1435
Data columns (total 10 columns):
Price        1436 non-null int64
Age          1336 non-null float64
KM           1421 non-null float64
FuelType     1336 non-null object
HP           1430 non-null float64
MetColor     1286 non-null object
Automatic    1436 non-null object
CC           1436 non-null int64
Doors        1436 non-null object
Weight       1436 non-null int64
dtypes: float64(3), int64(3), object(4)
memory usage: 123.4+ KB
```

(iv) Cleaning 'Doors' Column

- Checking unique values of variable 'Doors':

```
print(np.unique(stoyota['Doors']))

['2' '3' '4' '5' 'five' 'four' 'three']
```

```
[41] stoyota['Doors'].replace('three',3,inplace=True)
     stoyota['Doors'].replace('four',4,inplace=True)
     stoyota['Doors'].replace('five',5,inplace=True)
```

(v) Converting 'Doors' datatype

- Converting 'Doors' into int64

```
[45] stoyota['Doors'] = stoyota['Doors'].astype('int64')
```

- `replace()` is used to replace a value with the desired value
- Syntax: `DataFrame.replace([to_replace, value, …])`

```
6] stoyota.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 1436 entries, 0 to 1435
Data columns (total 10 columns):
Price          1436 non-null int64
Age            1336 non-null float64
KM             1421 non-null float64
FuelType       1336 non-null object
HP             1430 non-null float64
MetColor       1286 non-null object
Automatic      1436 non-null object
CC             1436 non-null int64
Doors          1436 non-null int64
Weight         1436 non-null int64
dtypes: float64(3), int64(4), object(3)
memory usage: 123.4+ KB
```

Identifying missing values

- Check the count of missing values present in each column
- *Dataframe.**isnull().sum()***

```
[165] stoyota.isnull().sum()

Price          0
Age          100
KM            15
FuelType     100
HP             6
MetColor     150
Automatic      0
CC             0
Doors          0
Weight         0
dtype: int64
```

```
[169] stoyota.isna().sum()

Price          0
Age          100
KM            15
FuelType     100
HP             6
MetColor     150
Automatic      0
CC             0
Doors          0
Weight         0
dtype: int64
```

Handling missing values

Two ways:

- Deleting missing values
- Imputing/filling the missing values

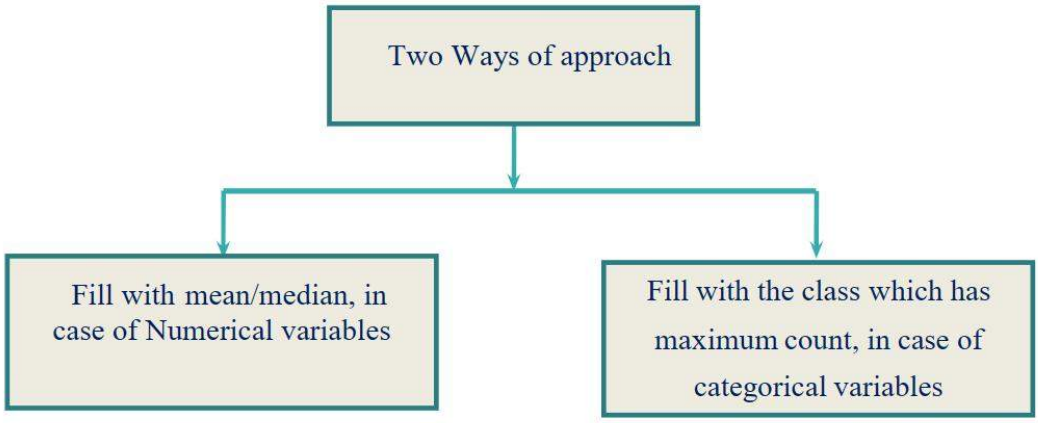
(i) Handling missing values - by deleting

- ***dropna():*** Method used to drop rows/cols containing missing values

```
[85] stoyota.dropna(inplace=True)
     stoyota.isnull().sum()

Price          0
Age            0
KM             0
FuelType       0
HP             0
MetColor       0
Automatic      0
CC             0
Doors          0
Weight         0
dtype: int64
```

```
[83] stoyota.shape

(1096, 10)
```

(ii) Handling missing values- by imputing




```
[48] stoyota.isnull().sum()

Price          0
Age          100
KM            15
FuelType     100
HP             6
MetColor     150
Automatic      0
CC             0
Doors          0
Weight         0
dtype: int64
```

- *DataFrame.**describe( ):*** Returns the statistical information about the data frame

```
[ ] stoyota.describe()
```

	Price	Age	KM	HP	CC	Doors	Weight
count	1436.000000	1336.000000	1421.000000	1430.000000	1436.000000	1436.000000	1436.00000
mean	10730.824513	55.672156	68647.239972	101.478322	1566.827994	4.033426	1072.45961
std	3626.964585	18.589804	37333.023589	14.768255	187.182436	0.952677	52.64112
min	4350.000000	1.000000	1.000000	69.000000	1300.000000	2.000000	1000.00000
25%	8450.000000	43.000000	43210.000000	90.000000	1400.000000	3.000000	1040.00000
50%	9900.000000	60.000000	63634.000000	110.000000	1600.000000	4.000000	1070.00000
75%	11950.000000	70.000000	87000.000000	110.000000	1600.000000	5.000000	1085.00000
max	32500.000000	80.000000	243000.000000	192.000000	2000.000000	5.000000	1615.00000

Imputing missing values for 'Age' variable

```
[58] stoyota['Age'].mean()

     55.67215568862275
```

- fillna() is used to fill NA/NaN values using the specified value
- Syntax: *DataFrame.**fillna()***

```
[ ] stoyota['Age'].fillna(stoyota['Age'].mean(),inplace=True)
```

Before imputing Null values

```
[ ] print(missing)

      Price   Age       KM FuelType  ...
2     13950  24.0  41711.0   Diesel  ...
6     16900  27.0      NaN   Diesel  ...
7     18600  30.0  75889.0      NaN  ...
9     12950  23.0  71138.0   Diesel  ...
15    22000  28.0  18739.0   Petrol  ...
...     ...   ...      ...      ...  ...
1428   8450  72.0      NaN   Petrol  ...
1431   7500   NaN  20544.0   Petrol  ...
1432  10845  72.0      NaN   Petrol  ...
1433   8500   NaN  17016.0   Petrol  ...
1434   7250  70.0      NaN      NaN  ...

[340 rows x 10 columns]
```

After imputing Null values

```
[ ] stoyota['Age'].tail(10)

1426    78.000000
1427    55.672156
1428    72.000000
1429    78.000000
1430    80.000000
1431    55.672156
1432    72.000000
1433    55.672156
1434    70.000000
1435    76.000000
Name: Age, dtype: float64
```

Imputing missing values for 'KM'

```
[61] stoyota['KM'].median()

     63634.0
```

```
[ ] stoyota['KM'].fillna(stoyota['KM'].median(),inplace=True)
```

```
[ ]  stoyota['KM'].head(10)

0    46986.0
1    72937.0
2    41711.0
3    48000.0
4    38500.0
5    61000.0
6    63634.0
7    75889.0
8    19700.0
9    71138.0
Name: KM, dtype: float64
```

Imputing missing values for 'HP'

```
[107] stoyota['HP'].mean()

101.47832167832168
```

```
[108] stoyota['HP'].fillna(stoyota['HP'].mean(),inplace=True)
```

```
[99] missing.loc[1:50]
```

	Price	Age	KM	FuelType	HP
2	13950	24.0	41711.0	Diesel	90.0
6	16900	27.0	NaN	Diesel	NaN
7	18600	30.0	75889.0	NaN	90.0
9	12950	23.0	71138.0	Diesel	NaN
15	22000	28.0	18739.0	Petrol	NaN
21	16950	29.0	43905.0	NaN	110.0
26	17495	27.0	34545.0	NaN	110.0

```
[109] stoyota['HP'].loc[5:20]

5        90.000000
6       101.478322
7        90.000000
8       192.000000
9       101.478322
10      192.000000
11      192.000000
12      192.000000
13      192.000000
14      192.000000
15      101.478322
16      192.000000
17      110.000000
18      110.000000
19      110.000000
20      110.000000
Name: HP, dtype: float64
```

Imputing missing values for 'FuelType'

- Most frequently occurred category is to be found
- Syntax: Series.***value_counts()***
- Returns a Series containing counts of unique values in descending order
- First element will be the most frequently-occurring element
- Excludes 'nan' values by default

```
[ ] stoyota['FuelType'].value_counts()

Petrol    1277
Diesel     144
CNG         15
Name: FuelType, dtype: int64
```

```
[ ] stoyota['FuelType'].fillna(stoyota['FuelType'].value_counts().index[0],inplace=True)
```

Imputing missing values for 'MetColor'

```
[110] stoyota['MetColor'].mode()

0    1
dtype: object
```

```
[111] stoyota['MetColor'].fillna(stoyota['MetColor'].mode().index[0],inplace=True)
```

Rechecking missing values

Before imputing Null values

```
[48] stoyota.isnull().sum()

Price          0
Age          100
KM            15
FuelType     100
HP             6
MetColor     150
Automatic      0
CC             0
Doors          0
Weight         0
dtype: int64
```

After imputing Null values

```
[ ] stoyota.isnull().sum()

Price        0
Age          0
KM           0
FuelType     0
HP           0
MetColor     0
Automatic    0
CC           0
Doors        0
Weight       0
dtype: int64
```

Correlation

```
DataFrame.corr(self, method='pearson')
```

It is used to compute pair wise correlation of columns excluding NA/null values
Excluding the categorical variables to find the Pearson's correlation

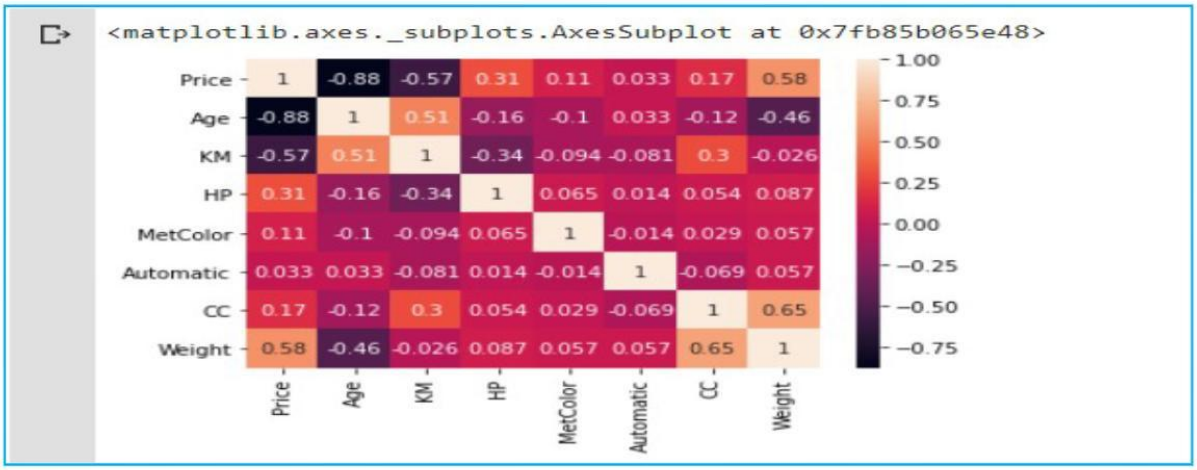
A strong negative correlationbetween Age and Price.

A Fair positive correlation between Weight and Price& KM and Age.

A Fair negative correlation between KM and Price.

Correlation – using heatmap

Import seaborn as sns
corr_matrix = stoyota.corr()
sns.heatmap(corr_matrix, annot=True)



EDA for Categorical Variables

What is the problem with Categorical data?

Categorical data are variables that contain label values rather than numeric values.

- The number of possible values is often limited to a fixed set.
- Categorical variables are often called '*nominal*'.

Many machine learning algorithms cannot operate on label data directly. They require all input variables and output variables to be numeric. This means that categorical data must be converted to a numerical form.

Converting categorical variables into numeric

- Label Encoding
- One-Hot Encoding

Label Encoding

Label encoding is simply converting each value in a column to a number

One method is converting a column to a category, and then uses those categoryvalues for your label encoding.

```
[ ]  df1['FuelType'] = df1['FuelType'].astype('category')
```

Then assign the encoded variable to a new column using the '*cat.codes'* accessor

One – Hot Encoding

A new binary variable is added for each of the categorical value. Pandas supports this feature using '*get_dummies()*'

Syntax:

```
pandas.get_dummies(data, prefix=None, prefix_sep='_', dummy_na=False, columns=None,
sparse=False, drop_first=False, dtype=None) ¶
```

```
df1["FuelType_cat"] = df1["FuelType"].cat.codes
print(df1['FuelType_cat'].value_counts())
print(df1['FuelType'].value_counts())
```

```
2    968
1    116
0     12
Name: FuelType_cat, dtype: int64
Petrol    968
Diesel    116
CNG        12
Name: FuelType, dtype: int64
```

```
[ ] new_carsdf = pd.get_dummies(stoyota)
    new_carsdf.shape
```

```
(1436, 13)
```

```
new_carsdf.columns
```

```
Index(['Price', 'Age', 'KM', 'HP', 'MetColor', 'CC', 'Doors', 'Weight',
       'FuelType_CNG', 'FuelType_Diesel', 'FuelType_Petrol', 'Automatic_0',
       'Automatic_1'],
      dtype='object')
```

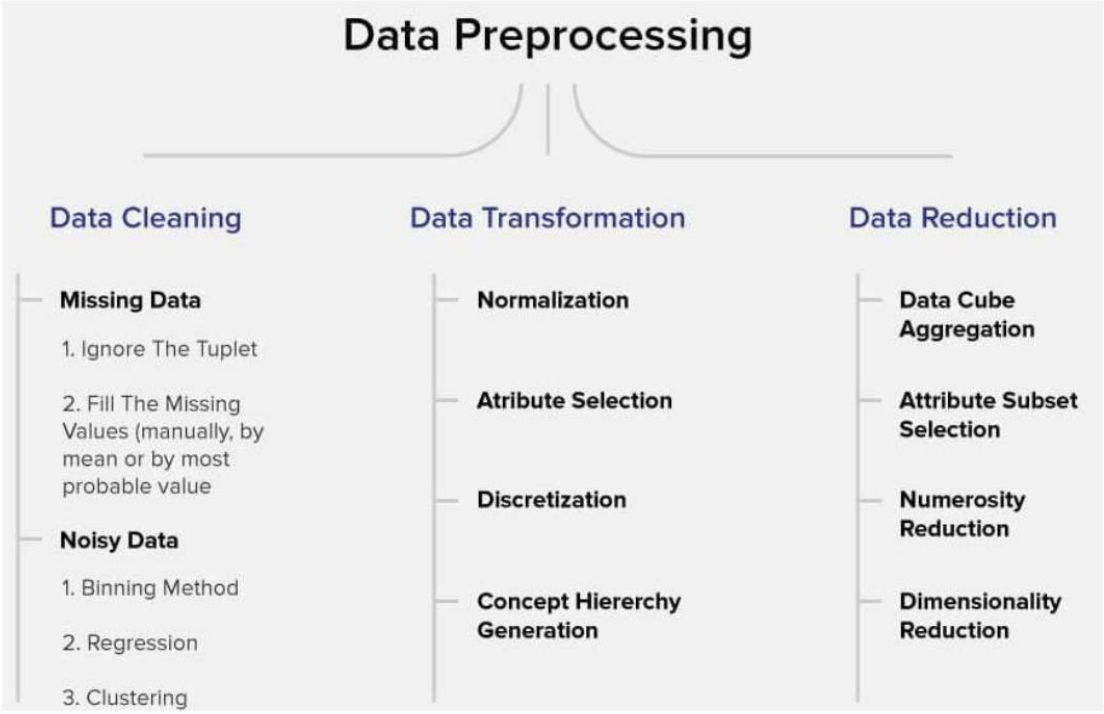
```
[ ] new_carsdf.iloc[1:6,7:]
```

	Weight	FuelType_CNG	FuelType_Diesel	FuelType_Petrol	Automatic_0	Automatic_1
1	1165	0	1	0	1	0
2	1165	0	1	0	1	0
3	1165	0	1	0	1	0
4	1170	0	1	0	1	0
5	1170	0	1	0	1	0

```
new_carsdf.dtypes
```

```
Price              int64
Age              float64
KM               float64
HP               float64
MetColor         float64
CC                 int64
Doors              int64
Weight             int64
FuelType_CNG       uint8
FuelType_Diesel    uint8
FuelType_Petrol    uint8
Automatic_0        uint8
Automatic_1        uint8
dtype: object
```

Data Preprocessing



Data preprocessing is a technique which is used to transform the raw data in a useful and efficient format.

Steps Involved in Data Pre-processing:

1. Data Cleaning:

The data can have many irrelevant and missing parts. To handle this part, data cleaning is done. It involves handling of missing data, noisy data etc.

(a) Missing Data:
This situation arises when some data is missing in the data. It can be handled in various ways.Some of them are:

- **Ignore the tuples:**

  This approach is suitable only when the dataset we have is quite large and multiple values are missing within a tuple.

- **Fill the Missing values:**

  There are various ways to do this task. You can choose to fill the missing values manually, by attribute mean or the most probable value.

(b) Noisy Data:

Noisy data is a meaningless data that can't be interpreted by machines. It can be generated due to faulty data collection, data entry errors etc. It canbe handled in following ways:

- **Binning Method:**

  This method works on sorted data in order to smooth it. The whole data is divided into segments of equal size and then various methods are performed to complete the task. Each segmented is handled separately. One can replace all data in a segment by its mean or boundary values.

- **Regression:**

  Here data can be made smooth by fitting it to a regression function. The regression used may be linear (having one independent variable) or multiple (having multiple independent variables).

- **Clustering:**

  This approach groups the similar data in a cluster. The outliers may be undetected or it will fall outside the clusters.

2. Data Transformation:

This step is taken in order to transform the data in appropriate forms. This involves following ways:

(a) Normalization:

It is done in order to scale the data values in a specified range (-1.0 to1.0 or 0.0 to 1.0)

(b) Attribute Selection:

In this strategy, new attributes are constructed from the given set of attributes to help the mining process.

(c) Discretization:

This is done to replace the raw values of numeric attribute by interval levels or conceptual levels.

(d) Concept Hierarchy Generation:

Here attributes are converted from lower level to higher level in hierarchy. Example-The attribute "city" can be converted to country".

3. Data Reduction

The size of the dataset in a data warehouse can be too large to be handled by data analysis and data mining algorithms.

One possible solution is to obtain a reduced representation of thedataset that is much smaller in volume but produces the same quality of analytical results. Here is a walkthrough of various Data Reduction strategies.

(a) Data cube aggregation

It is a way of data reduction, in which the gathered data is expressed in a summary form.

(b) Numerosity reduction

The data can be represented as a model or equation like a regressionmodel. This would save the burden of storing huge datasets insteadof a model.

(c) Attribute subset selection

It is very important to be specific in the selection of attributes.

Otherwise, it might lead to high dimensional data, which are difficult to train due to underfitting / overfitting problems. Only attributes that add more value towards model training should be considered, and the rest all can be discarded.

(d) Dimensionality reduction

Dimensionality reduction techniques are used to perform feature extraction. The dimensionality of a dataset refers to the attributes orindividual features of the data. This technique aims to reduce the number of redundant features we consider in machine learning algorithms. Dimensionality reduction can be done using techniqueslike Principal Component Analysis etc

Data visualization

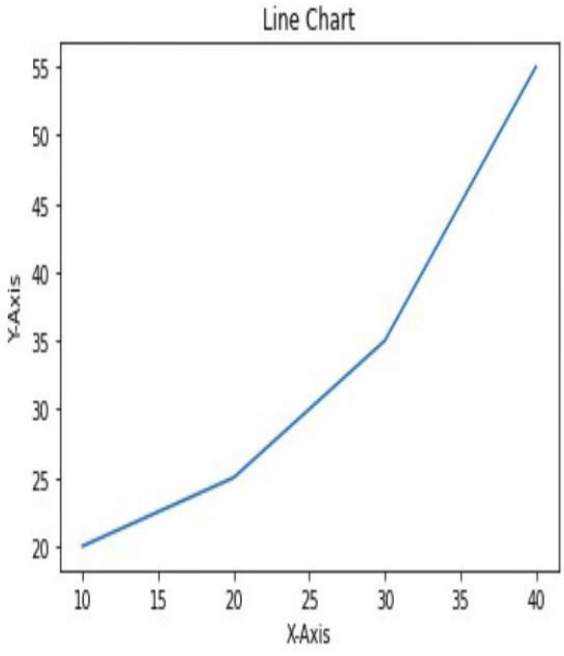
- Data visualization is the graphical representation of information and data. By using visual elements likecharts, graphs, and maps, data visualization tools provide an accessible way to see and understand trends, outliers, and patterns in data.

- Matplotlib has a module called pyplot which aids in plotting figure.
- Importing required libraries and dataset to plot using Pandas pd.read_csv()
- plt.plot()for plotting line chart similarly in place of plot other functions are used for plotting.
- plot.xlabel , plt.ylabel for labeling x and y-axis respectively.
- plt.title() for setting the title of the plot.
- plot.show() for displaying the plot.

(a) Line Graph/Chart

Line chart is one of the basic plots and can be created using the plot() function. It is used to represent a relationship between two data X and Y on a different axis.

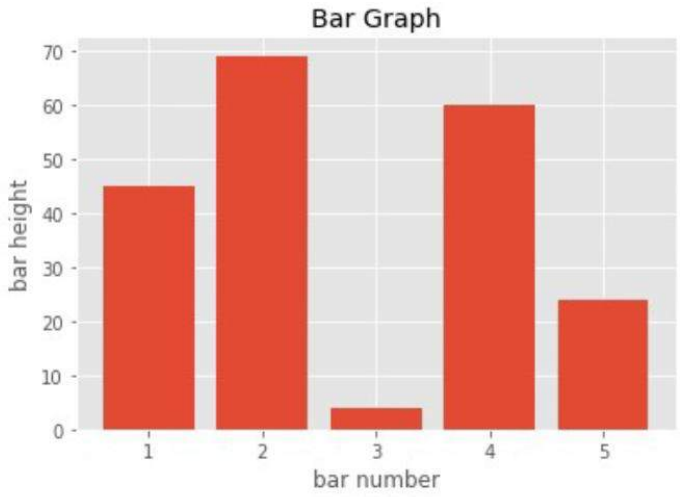
```
import matplotlib.pyplot as plt


# initializing the data
x = [10, 20, 30, 40]
y = [20, 25, 35, 55]

# plotting the data
plt.plot(x, y)

# Adding title to the plot
plt.title("Line Chart")

# Adding label on the y-axis
plt.ylabel('Y-Axis')

# Adding label on the x-axis
plt.xlabel('X-Axis')

plt.show()
```



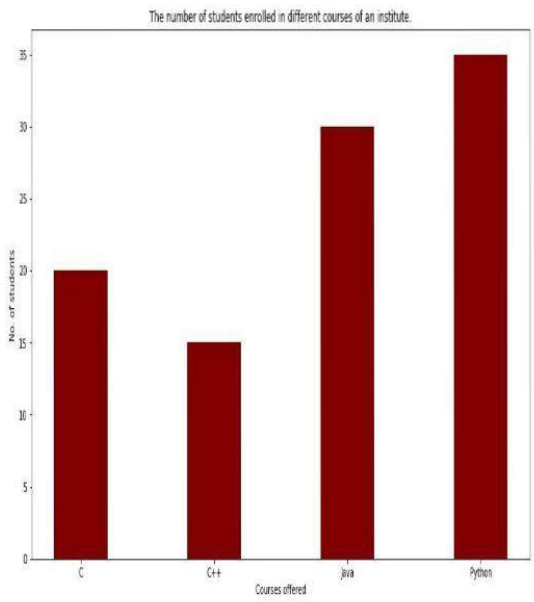
(b) Bar Graph

- Bar graphs used to show the changes over time and to compare attributes.
- Bar graph will have an x-axis (horizontal) and a y-axis (vertical).
- A bar plot or bar chart is a graph that represents the category of data with rectangular bars with lengths and heights that is proportional to the values which they represent.

```
#import matplotlib
from matplotlib import pyplot as plt
#Bar Graph
plt.bar([1,2,3,4,5],[45,69,4,60,24])
plt.xlabel('bar number')
plt.ylabel('bar height')
plt.title("Bar Graph")
plt.show()
```



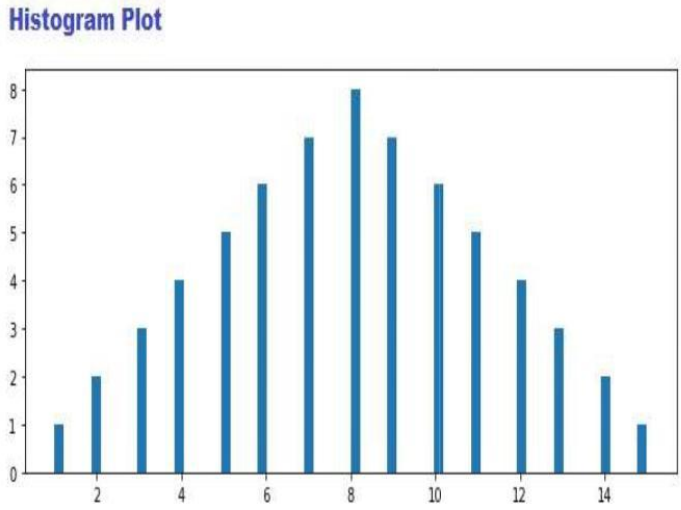
```python
import numpy as np
import matplotlib.pyplot as plt


# creating the dataset
data = {'C':20, 'C++':15, 'Java':30,
        'Python':35}
courses = list(data.keys())
values = list(data.values())

fig = plt.figure(figsize = (10, 5))

# creating the bar plot
plt.bar(courses, values, color ='maroon',
        width = 0.4)

plt.xlabel("Courses offered")
plt.ylabel("No. of students enrolled")
plt.title("Students enrolled in different courses")
plt.show()
```



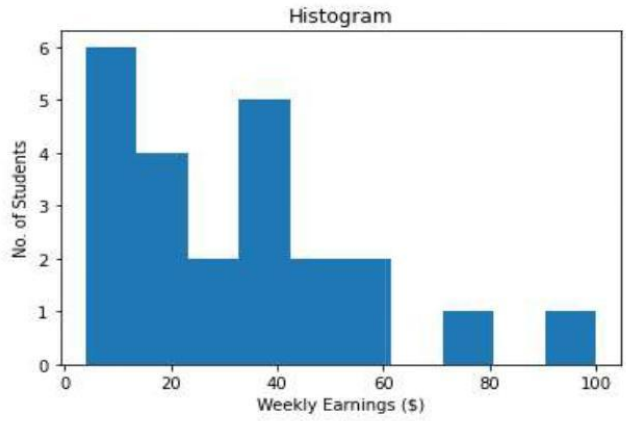
(c) Histogram

- A histogram takes in a series of data and divides the data into a number of bins. It then plots the frequency data points in each bin (i.e. the interval of points). It is useful in understanding the count of data ranges.
- Histogram graph looks like Bar Graph but this is continuous type of chart. In a histogram, each bar groups numbers into ranges.
- Taller bars show that more data falls in that range.
- Bins are used to create data with n number of bins

```
import numpy as np
import matplotlib.pyplot as plt
Dataset1 =
    [1,2,2,3,3,3,4,4,4,4,5,5,5,5,5,6,6,
    6,6,6,6,7,7,7,7,7,7,7,8,8,8,8,8,8,8
    ,8,9,9,9,9,9,9,9,10,10,10,10,10,1
    0,11,11,11,11,11,12,12,12,12,13,
    13,13,14,14,15]
y=Dataset1
plt.figure(figsize=(10, 4))
plt1 = plt.hist(Dataset1,bins=64)
plt.show()
```




```
from matplotlib import pyplot as plt
x = [21,22,23,4,5,6,77,8,9,10,31,32,33,34,60,55,35,36,37,18,49,50,100]
num_bins = 10
plt.hist(x, num_bins)
plt.xlabel("Weekly Earnings ($)")
plt.ylabel("No. of Students")
plt.title("Histogram")
plt.show()
```

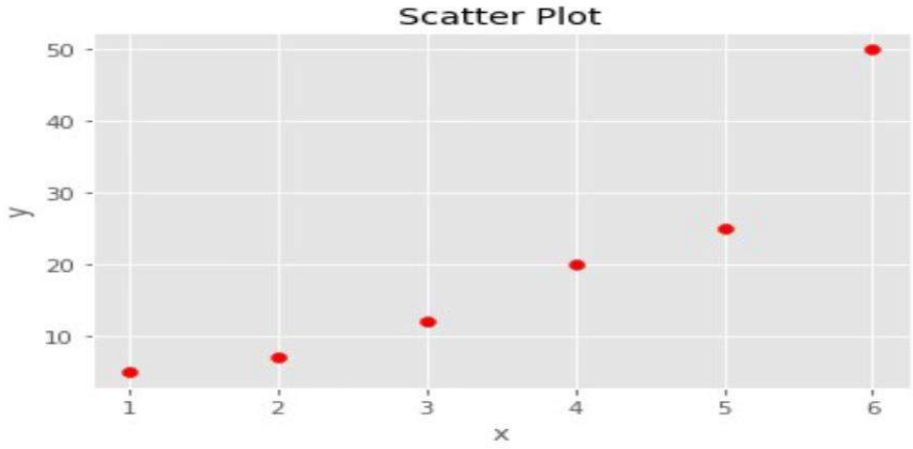


(d) Scatter Plot

- A scatter plot uses dots to represent values for two different numeric variables. The position of each dot onthe horizontal and vertical axis indicates values for an individual data point. Scatter plots are used to observe relationships between variables.

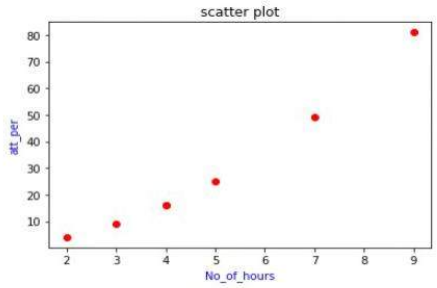
Code:

```
from matplotlib import pyplot as plt
x= [1,2,3,4,5,6]
y= [5,7,12,20,25,50]
plt.scatter(x,y, color = 'r') plt.xlabel('x')plt.ylabel('y') plt.title('Scatter Plot')plt.show()
```




```
import numpy as np
import matplotlib.pyplot as plt
x=np.array([2,3,4,5,7,9,4])
print(x)
y=x**2
print(y)
plt.scatter(x,y,c='r')
plt.xlabel('No_of_hours',c='blue')
plt.ylabel('att_per',c='blue')
plt.title('scatter plot')
plt.figure(figsize=(5,5))
plt.show()
```

```
[2 3 4 5 7 9 4]
[ 4  9 16 25 49 81 16]
```

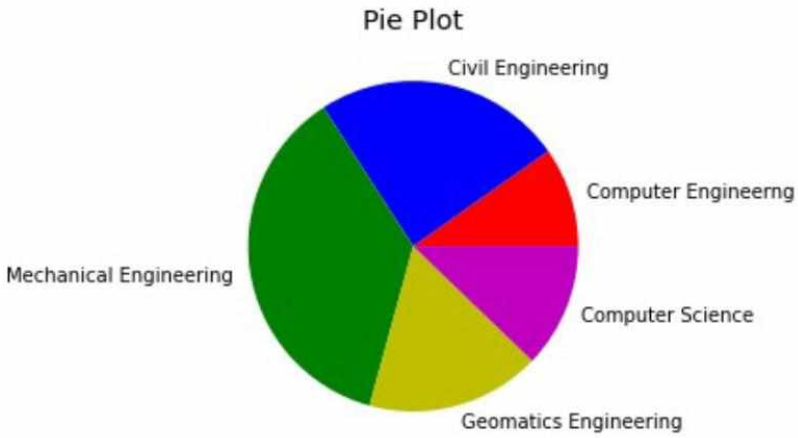



```
<Figure size 360x360 with 0 Axes>
```

(e) Pie Chart

- A pie chart is a circular statistical graph, which is divided into slices to illustrate numerical proportion. In a pie chart, the arc length of each slice is proportional to the quantity it represents.

```
from matplotlib import pyplot as plt
students = [400, 1000, 1500, 700, 500]
interests = ['Computer Engineering','Civil Engineering','Mechanical Engineering','Geomatics Engineering','Computer Science']
col= ['r','b','g','y','m']
plt.pie(students,labels =interests, colors= col)
plt.title('Pie Plot')
plt.show()
```


Pie Plot
Civil Engineering
Computer Engineerng
Computer Science
Geomatics Engineering
Mechanical Engineering

```
import numpy as np
import matplotlib.pyplot as plt
x=np.array([2,3,4,5,7,9,4])
print(x)
y=x**2
print(y)
plt.subplot(231)
plt.scatter(x,y,c='r')
plt.subplot(232)
plt.bar(x,y)
plt.subplot(233)
plt.pie(x)
plt.subplot(234)
plt.boxplot(y)
plt.subplot(235)
plt.plot(x,x/y)
plt.subplot(236)
plt.hist(y)
plt.show()
```

```
[2 3 4 5 7 9 4]
[ 4  9 16 25 49 81 16]
```