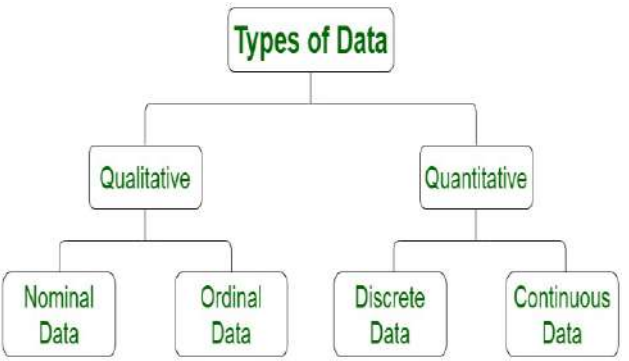
UNIT II Descriptive statistics:

- Descriptive statistics describes, shows, and summarize 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.

What is Data?

- A data set is a collection of related information or Records.
- Data is the most important part of Machine Learning, Artificial Intelligence, Data Analytics, Without data it is impossible to train any model and all modern research and automation will go in vain.
- 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. 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

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 makes it countable and includes statistical data analysis.
- Also known as 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.

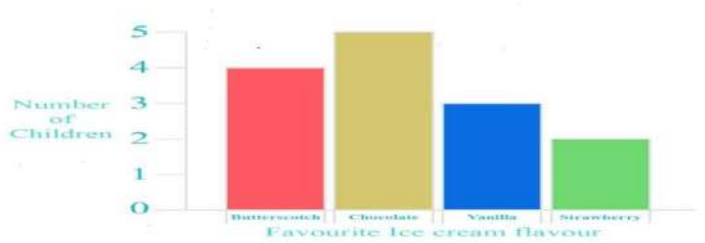


Figure: Bar Plot

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

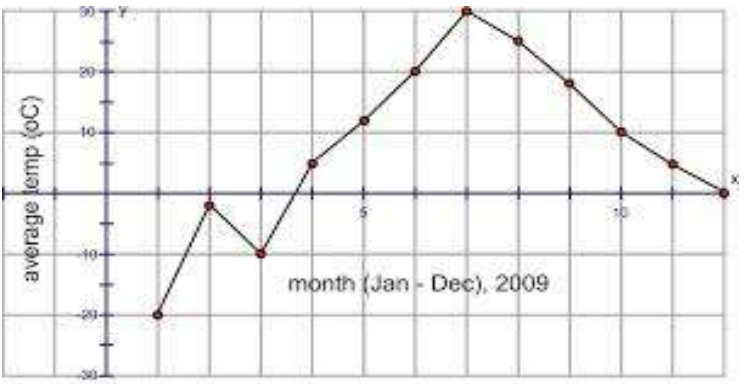


Figure: Line Plot

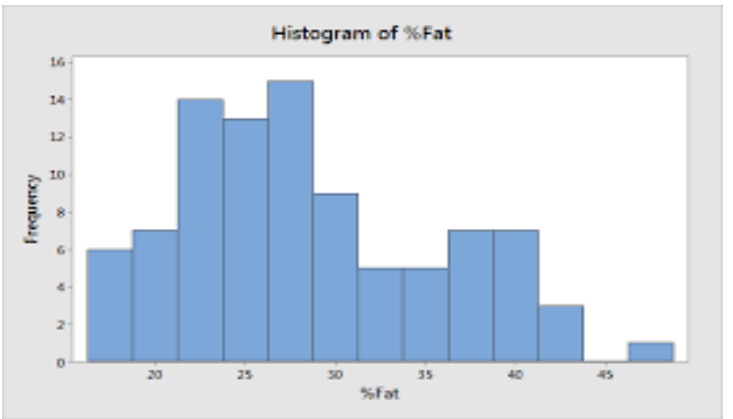
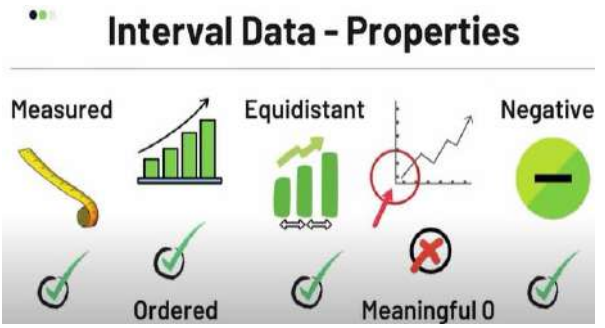


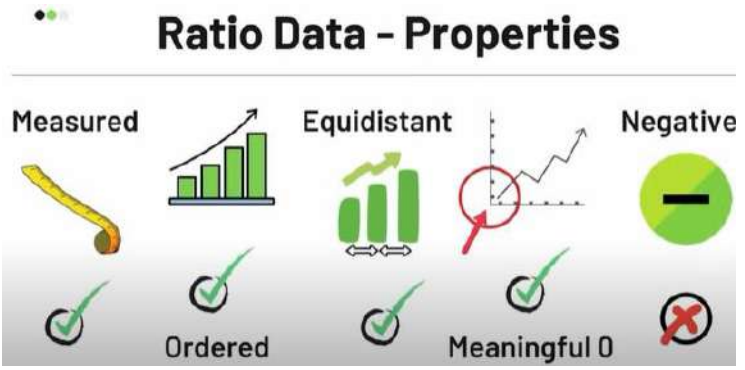
Figure: Histogram

- **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.
- Addition & Subtraction operations can be performed on Interval data.
- interval data do not 'true zero' value.
- Descriptive statistics which can be obtained using interval data include:
    1.Central tendency: Mode, median, and mean
    2. Dispersion: Range, Standard deviation and Variance



Ratio Data:

- Ratio data represents numeric data for which exact value can be measured.
    Example: Distance Travelled, Age of a Person.
- Absolute zero is available for ratio data.
    Example: its possible to say 0km distance travelled
- Ratio Data variables can be added, subtracted, multiplied, or divided.
- Descriptive statistics which can be obtained using Ratio data include:
    1.Central tendency: Mode, median, and mean
    2.Disperson: Range, Standard deviation and variance

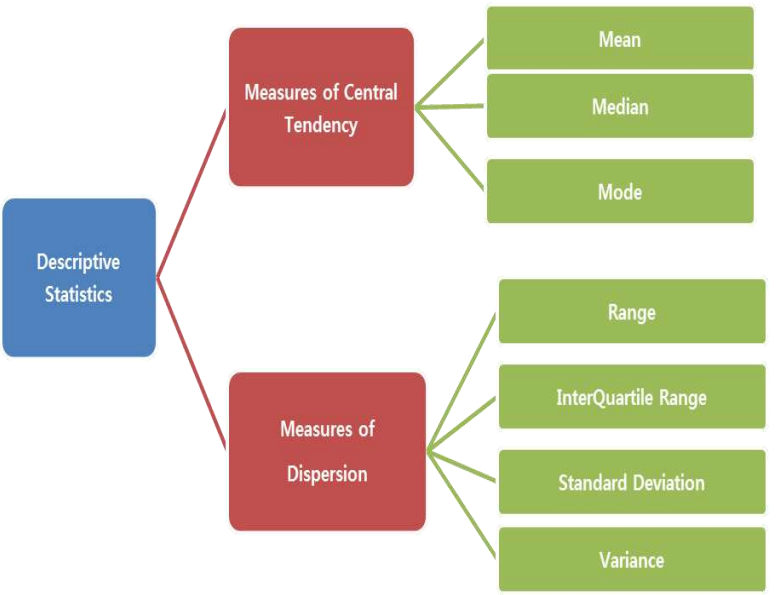


NOTE: In general, nominal and ordinal attributes are discrete. On the other hand, interval and ratio attributes are continuous, barring a few exceptions,e.g. 'count' attributes.

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

Central Tendency:



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

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

$$\overline{x} = \frac{\sum x}{N}$$

where,

$\sum$ 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 our calculation for mean.

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:**

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

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

(Note that the mean is not a value from the original list. This is a common result. You should not assume that your mean will be one of your original numbers.)

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
Covariance:

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

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

import warnings

warnings.filterwarnings('ignore')

import numpy as np

from scipy import stats

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

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)

Lets understand with a plot:

import numpy as np

import matplotlib.pyplot as plt

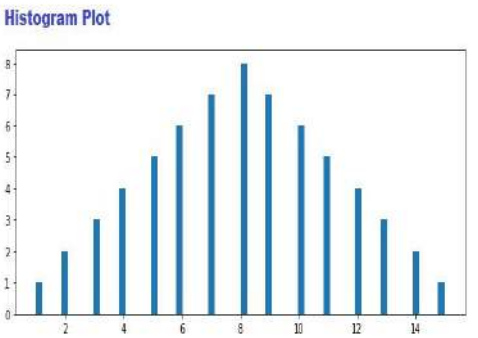
Dataset1=[1,2,2,3,3,3,4,4,4,4,5,5,5,5,5,6,6,6,6,6,6,7,7,7,7,7,7,7,8,8,8,8,8,8,8,8,8,9,9,9,9,9,9,9,10,10,10,10,10,10,11,11,11,11,11,12,12,12,12,13,13,13,14,14,15]

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

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

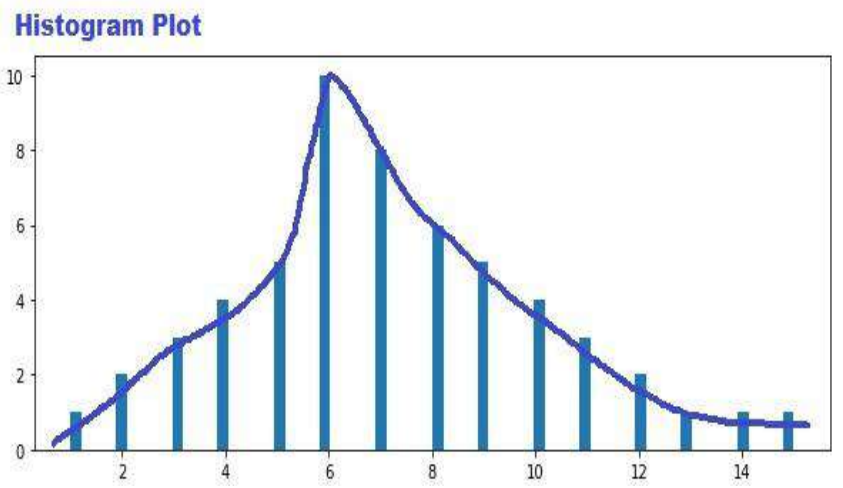
```
import numpy as np

import matplotlib.pyplot as plt

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

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

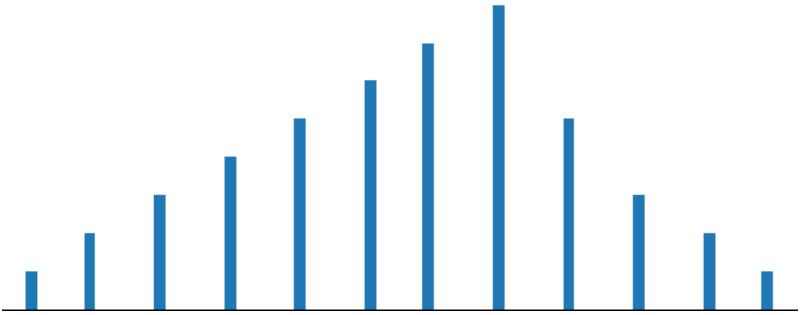
warnings.filterwarnings('ignore')

import numpy as np

```
from scipy import stats

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

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 = [1,2,2,3,3,3,4,4,4,4,5,5,5,5,5,6,6,6,6,6,6,7,7,7,7,7,7,7,8,8,8,8,8,8,8,8,8,8,9,9,9,9,9,10,10,10,11,11,12]

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.**

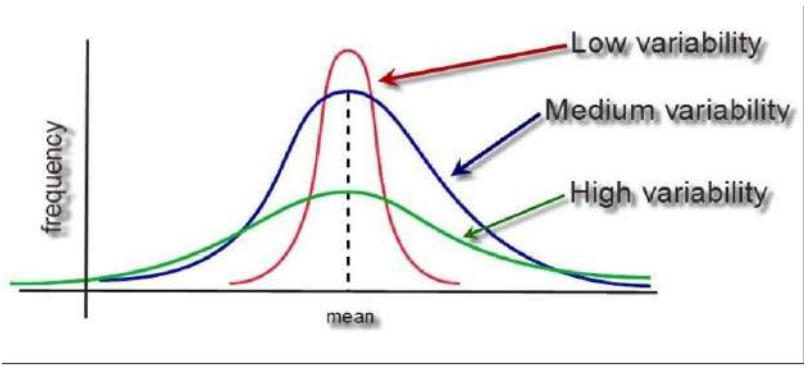
When should you use the mean, median or mode?

- 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 used for any level of 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

Variability:

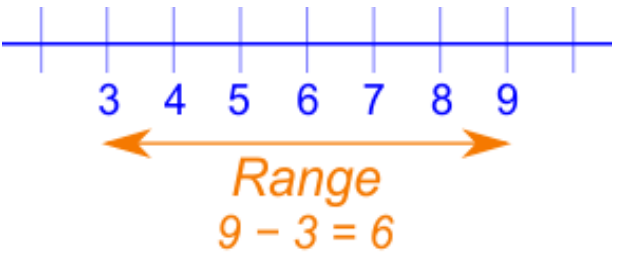
- Variability (also called spread or dispersion) refers to how a data of each attribute is spread out in a dataset.
- The four main ways to describe variability in a data set are:
  - range
  - Variance
  - Standard deviation.
  - Interquartile range



Range:

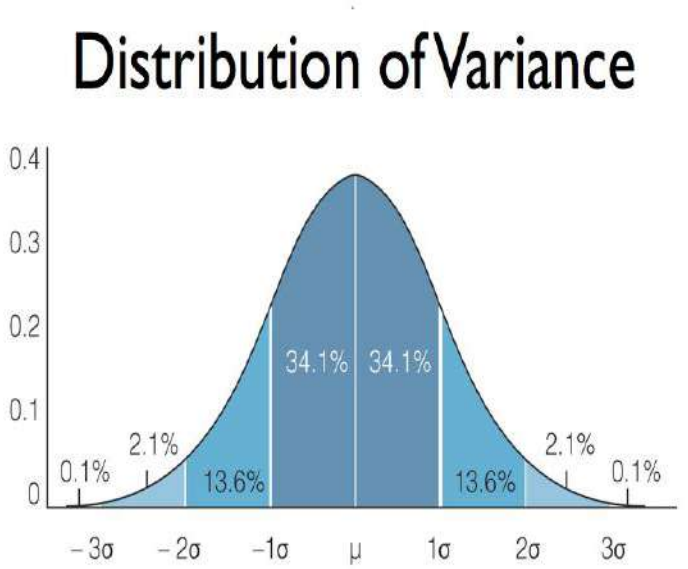
- In statistics, the range is the spread of your data from the lowest to the highest value in the distribution.

- 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.



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 means(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

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

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

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

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

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

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

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

1. Calculate the mean

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

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

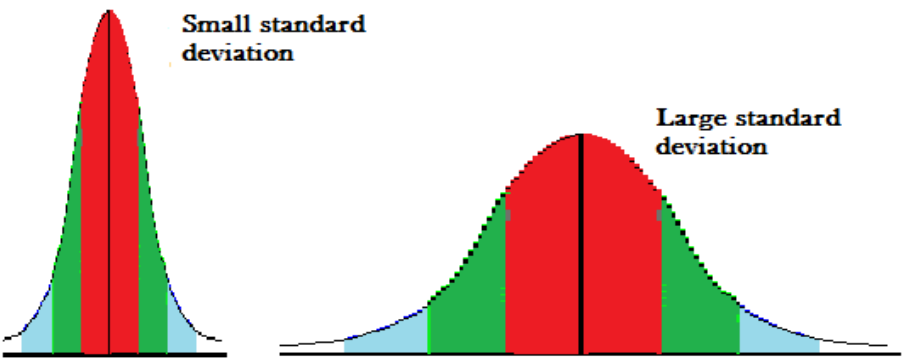
$$= 6$$

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

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

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

- **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:

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

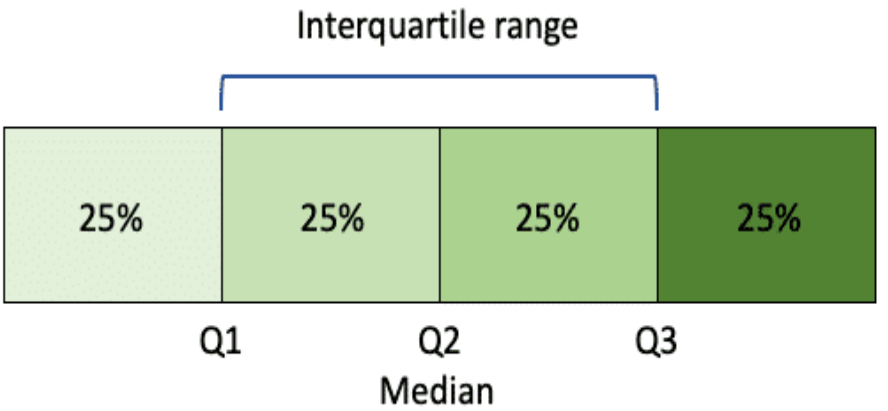
Finding measures of Variability on simple Data

```
# 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
```

Interquartile Range:

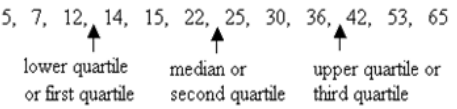
- Interquartile Range, (IQR) is a property that is used to measure variability.
- The IQR divides a dataset into quartiles. These quartiles store data after the data gets sorted in ascending order and split into 4 equal parts. The first, second, and third quartiles are called Q1, Q2, and Q3. These quartiles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

P)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

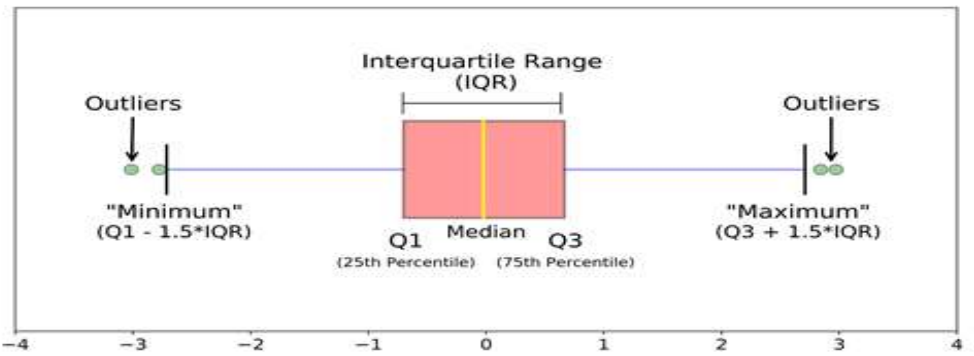
- **Solution:**
  - Find the median, lower quartile, upper quartile, interquartile range and range of the following numbers.
    12, 5, 22, 30, 7, 36, 14, 42, 15, 53, 25, 65
  - **Solution:**
    First, arrange the data in ascending order:



  - **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

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)
```

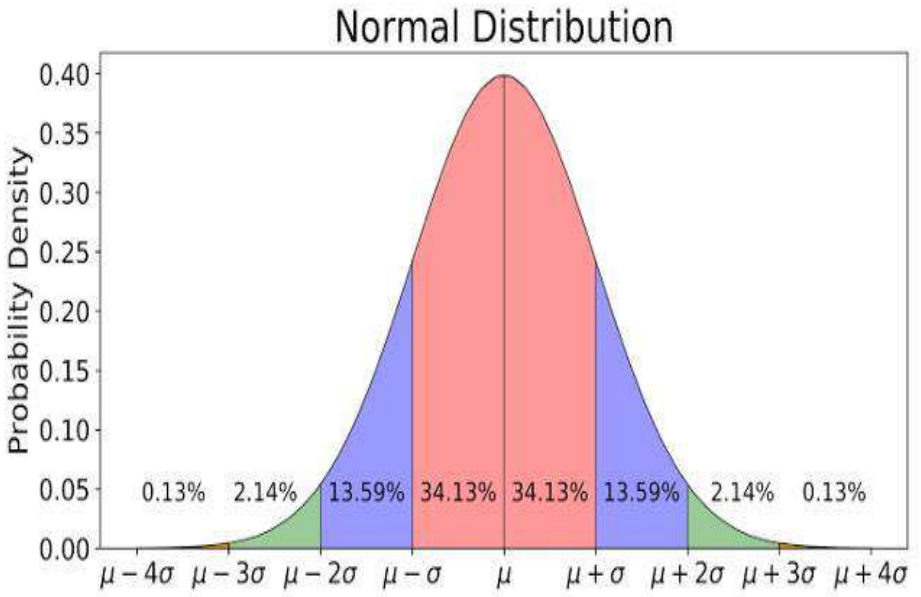
```
29
```

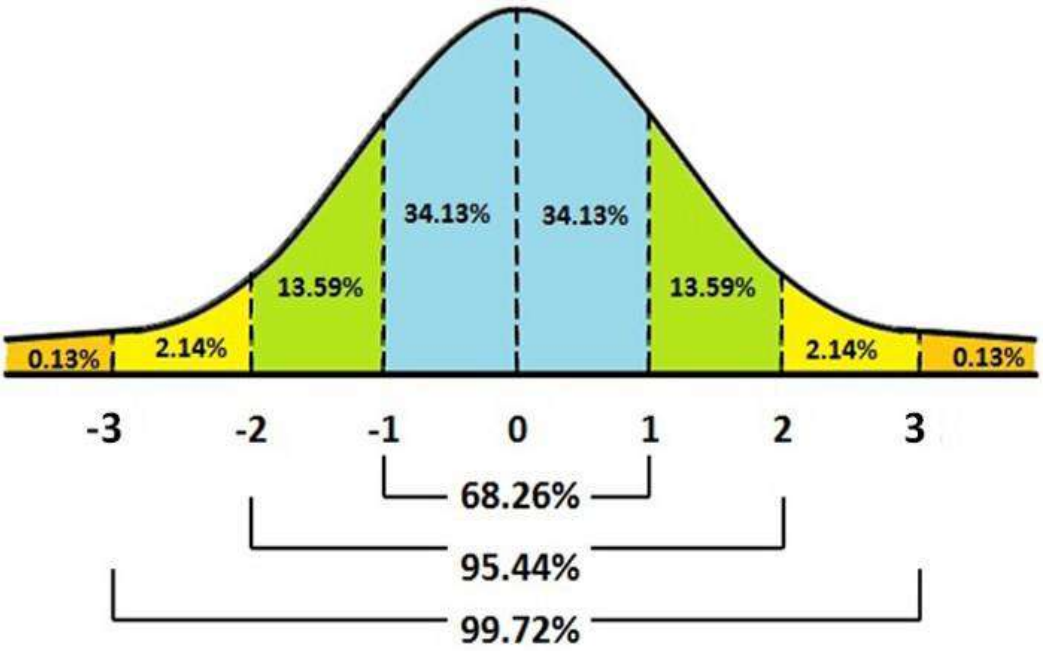
+ Code + Markdown

```
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
```

```
[102, 107, 108, 60, 140]
```

```
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]
```

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]
```

IQR Method

```
q1,q3 = np.percentile(data1,[25,75])
print('Quartile 1: ',q1)
print('Quartile 2: ',q3)
iqr = q3 - q1
print('Inter Quartile Range: ',iqr)
```

```
Quartile 1:  12.0
Quartile 2:  15.0
Inter Quartile Range:  3.0
```

```
lower_bound = q1 - (iqr*1.5)
upper_bound = q3 + (iqr*1.5)
print(lower_bound)
print(upper_bound)
```

```
7.5
19.5
```

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

+ Code + Markdown

```
outliers = detect_outliers(data1)
outliers
```

```
[24, 102, 107, 108, 60, 140]
```

Which one to Choose: IQR or STD ?

Which one to Choose out of IOR & Standard deviation

```
Dataset= [1, 4, 8, 11, 13, 17, 19, 19, 20, 23, 24, 24, 25, 28, 29, 31, 32]
```

```
q1,q3 = np.percentile(Dataset,[25,75])
print('Quartile 1: ',q1)
print('Quartile 2: ',q3)
iqr = q3 - q1
print('Inter Quartile Range: ',iqr)
```

```
Quartile 1:  13.0
Quartile 2:  25.0
Inter Quartile Range:  12.0
```

```
SD =np.std(Dataset)
print("Standard devaition is", SD)
```

```
Standard devaition is 8.975553110155273
```

```
Dataset= [1, 4, 8, 11, 13, 17, 19, 19, 20, 23, 24, 24, 25, 28, 29, 31, 32,300]
q1,q3 = np.percentile(Dataset,[25,75])
print('Quartile 1: ',q1)
print('Quartile 2: ',q3)
iqr = q3 - q1
print('Inter Quartile Range: ',iqr)
```

```
Quartile 1:  14.0
Quartile 2:  27.25
Inter Quartile Range:  13.25
```

+ Code + Markdown

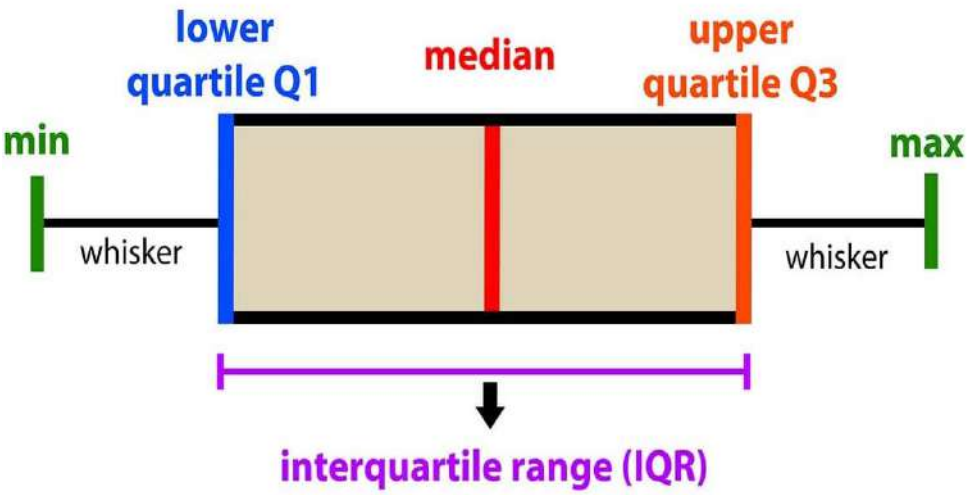
```
SD =np.std(Dataset)
print("Standard devaition is", SD)
```

```
Standard devaition is 64.88784245731577
```

It evident from the above example standard deviation method is more prone to outliers,IQR method is robust to outliers.

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.

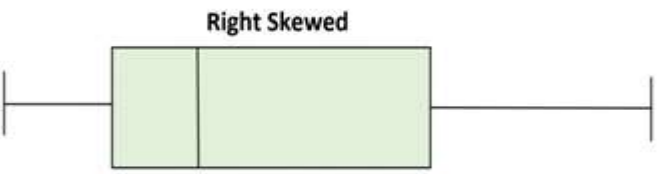


- **In a box and whisker plot:**
- The left and right sides of the box are the lower and upper quartiles. The box covers the interquartile interval, where 50% of the data is found.
- The vertical line that split the box in two is the median. Sometimes, the mean is also indicated by a dot or a cross on 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

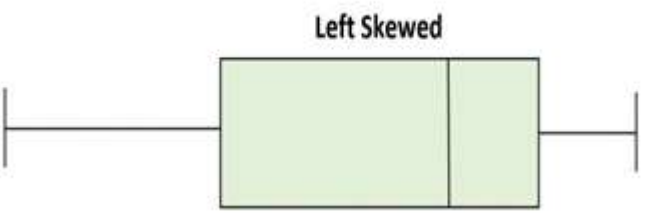
the upper quartile (the end of the box) to the maximum.

- 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).



-

Covariance:

Covariance measures how two variables move with respect to each other .The covariance between two random variables X and Y measure the degree to which X and Y are (linearly) related, which means how X varies with Y and vice versa.The values of covariance can be any number between the two opposite infinities.

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

Where,

x = the independent variable

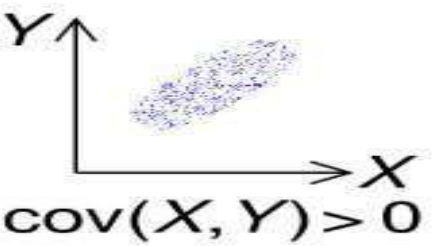
y = the dependent variable

n = number of data points in the sample

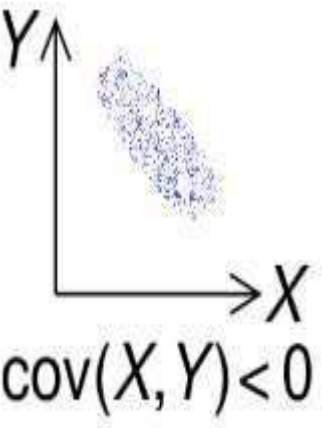
$\bar{x}$ = the mean of the independent variable x

$\bar{y}$ = the mean of the dependent variable y

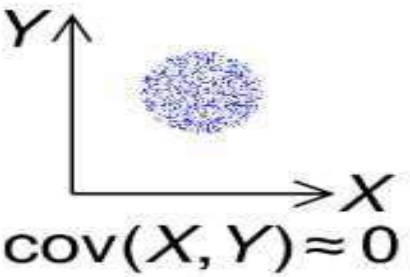
A positive number signifies positive covariance and denotes that there is a direct relationship



A negative covariance which denotes an inverserelationship between the two variables.



When two variables are independent, the covariance will be zero.



X	Y
10	**40**
12	**48**
14	**56**
8	**32**

Example

Step 1: Calculate Mean of X and Y

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

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

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

Substitute the above values in the formula

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

Cov(x,y) = 8/4 = **2**

Hence, Co-variance for the above data is 2

```
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 thetwo variables.

To do so we have to normalize the covariance by dividing it with theproduct 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.

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

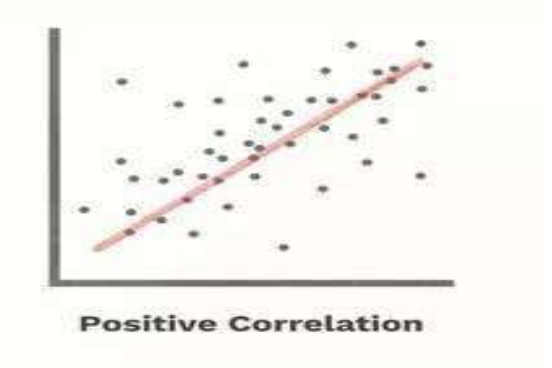
where:

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

Types of correlation

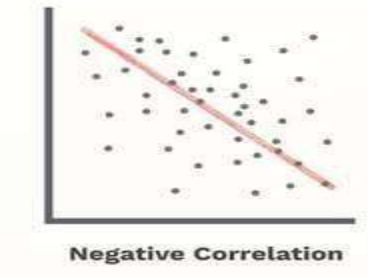
Positive correlation

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



Negative correlation

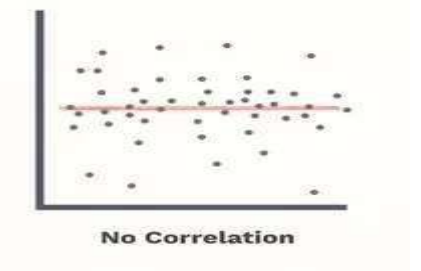
If increase in random variable A leads to adecrease in B, or vice



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

$$Cov(x,y) = \frac{\Sigma (x_i - \bar{x}) * (y_i - \bar{y})}{N}$$

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

Hence, Co-variance for the above data is 2

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

Cov(x,y) = 8/4 = **2**

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

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

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

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

-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 = 10.29Correlation = 2 /(**2.581** x10.29) Correlation = 0.075

```
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
```

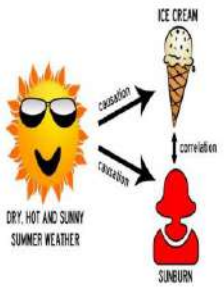
Key	Covariance	Correlation
Meaning	Covariance is a measure indicating the extent to which two random variables change in tandem.	Correlation is a statistical measure that indicates how strongly two variables are related.
What is it?	Measure of correlation	Scaled version of covariance
Values	Lie between -∞ and +∞	Lie between -1 and +1
Change in scale	Affects covariance	Does not affects correlation
Unit free measure	No	Yes

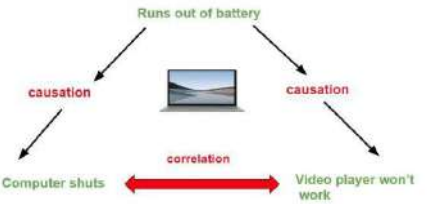
Causation

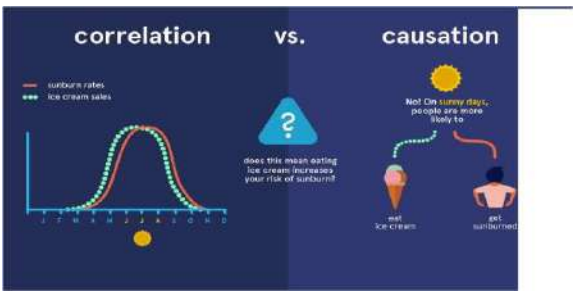
Causation means that changes in one variable bringsabout changes in the other; there is a cause-and- effect relationship between variables.The two variables are correlated with each other andthere is also a causal link between them.

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.







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.

Concise Summary of DataFrame

- ***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
```

Checking format of each column

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.

Missing values present in few variables,Let's encounter the reason !

Finding Unique elements of columns

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

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])
```

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 make reasonable decisions.

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

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

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.

```
[ ]  import pandas as pd
     toyota=pd.read_csv('Toyota.csv',
                        index_col=0,
                        na_values=["??","????"])

     stoyota = toyota.copy()
```

Concise summary of DataFrame

Summary – before replacing special

```
[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
```

Summary – after replacing special

```
[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
```

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)***

Recheck the data types

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

    dtype('O')
```

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

    dtype('O')
```

Summary – before converting

```
[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

```
[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
```

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)
```

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

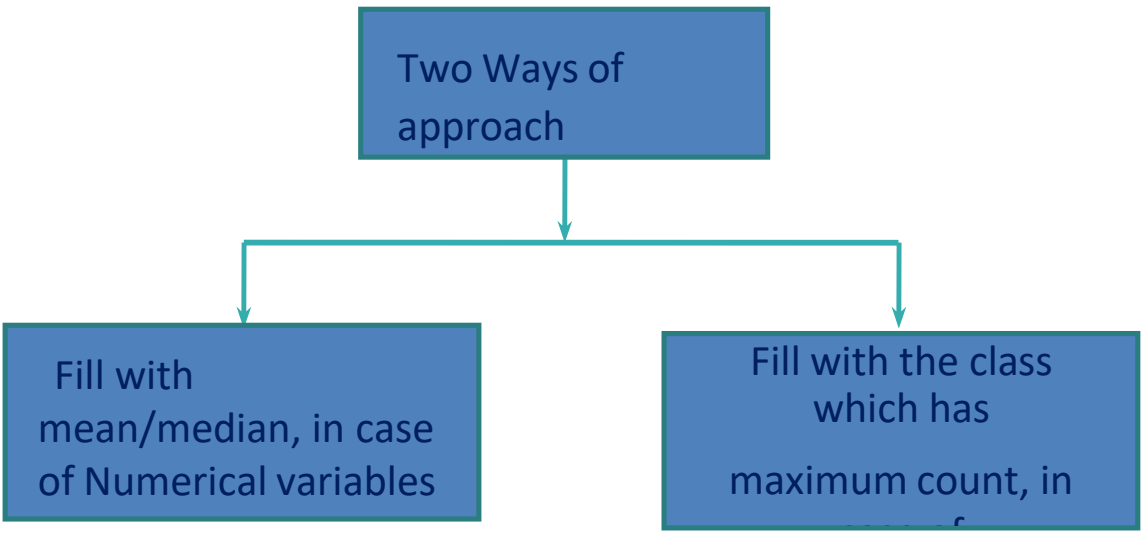
- Two ways:
- Deleting missing values
- Imputing/filling the missing values

Handling missing values- by removing

dropna()

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

Price          0
Age            0
KM             0
FuelType       0
HP             0
MetColor       0
Automatic      0
```

Method used to droprows/cols containingmissing values



Handling missing values- by imputing

- Different approaches to fill the missing values

Handling missing values- by imputing
Handling missing values- by imputing

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

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

Price          0
Age          100
KM            15
FuelType     100
HP             6
MetColor     150
Automatic      0
CC             0
```

Imputing missing values for ‘Age’

```
[ ] 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

Find the mean value of the ‘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(),inpla
```

Imputing missing values for ‘Age’

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]
```

```
[ ] 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(),inplac
```

```
[ ] 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().
```

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
After imputing Null values

Correlation

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

```
[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
```

```
[ ] 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
```

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

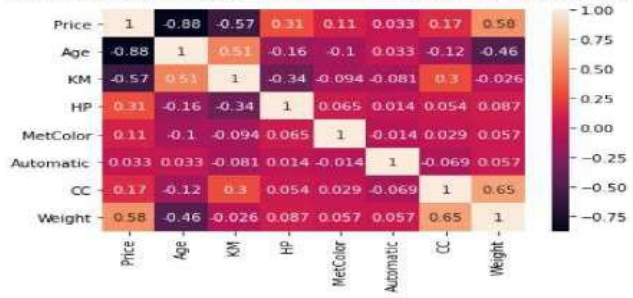
A
be
Co

```
toyotadf.corr()
```

	Price	Age	KM	HP	MetColor	Automatic	
Price	1.000000	-0.878407	-0.574720	0.309902	0.112041	0.033081	0.
Age	-0.878407	1.000000	0.512735	-0.157904	-0.099659	0.032573	-0.
KM	-0.574720	0.512735	1.000000	-0.335285	-0.093825	-0.081248	0.
HP	0.309902	-0.157904	-0.335285	1.000000	0.064749	0.013755	0.

```
[35] import seaborn as sns
     corr_matrix = toyotadf.corr()
     sns.heatmap(corr_matrix, annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fb85b065e48>
```



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 labeldata directly.They require all input variables and output variables to benumeric.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 trick is convert a column to a category, then use 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 thecategorical 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) ¶
```

```
_carsdf = pd.get_dummies(stoyota)
_carsdf.shape
```

```
6, 13)
```

```
_carsdf.columns
```

```
x(['Price', 'Age', 'KM', 'HP', 'MetC
    'FuelType_CNG', 'FuelType_Diesel',
    'Automatic_1'],
   dtype='object')
```

```
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
```

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

Weight	FuelType_CNG	FuelType_Diesel	FuelType_Petrol	Automatic_
1165	0	1	0	
1165	0	1	0	
1165	0	1	0	
1170	0	1	0	
1170	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
```

Now all columns are of numeric type only
Using Scikit-Learn

- To do a label encoding, we need to instantiate

```
from sklearn.preprocessing import LabelEncoder

lbe = LabelEncoder()
df4['FuelType_lbe'] = lbe.fit_transform(df4['FuelType'])
print(df4.columns)
print(df4['FuelType_lbe'].unique())
```

```
Index(['Price', 'Age', 'KM', 'FuelType', 'HP', 'MetColor', 'Automatic', 'CC',
       'Doors', 'Weight', 'FuelType_lbe'],
      dtype='object')
[1 2 0]
```

```
CNG     0
Diesel  1
Petrol  2
```

aLabelEncoder object and fit_transform the data

Drop this column afterwards…….
Alternately…

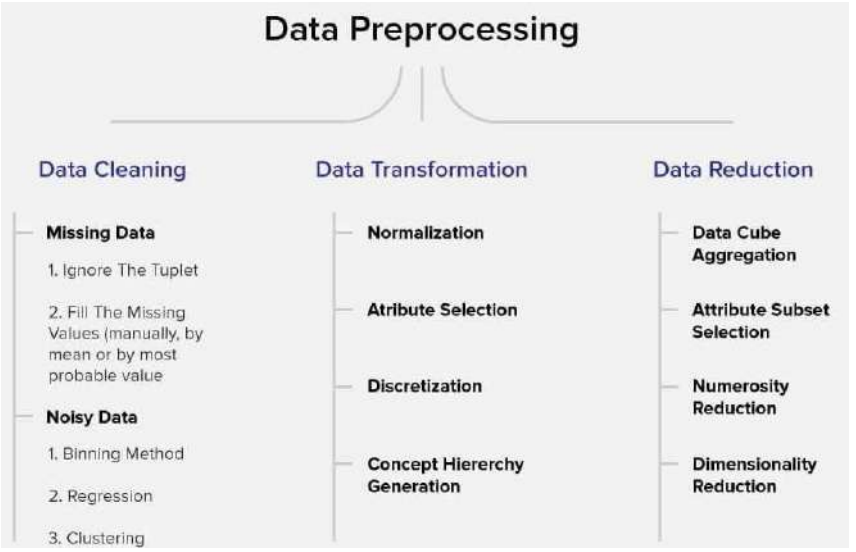
```
[38] df3 = stoyota.copy()
```

```
from sklearn import preprocessing
le = preprocessing.LabelEncoder()
df3['FuelType'] = le.fit_transform(df3['FuelType'])
df3['Automatic'] = le.fit_transform(df3['Automatic'])
print(df3['FuelType'].unique())
print(df3['Automatic'].unique())
df3.shape
```

```
[1 2 0]
[0 1]
(1436, 10)
```

```
[40] df3.dtypes
```

```
Price          int64
Age          float64
KM           float64
FuelType       int64
HP           float64
MetColor     float64
Automatic      int64
CC             int64
Doors          int64
Weight         int64
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 Preprocessing:**
- **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 andmultiple 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 methodsare performed to complete the task. Each

segmented is handled separately. One can replace all data in a segment by its mean or boundary values can be used to complete the task.

Regression:

Here data can be made smooth by fitting it to a regression function.Theregression 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.

Data Transformation:

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

Normalization:

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

Attribute Selection:

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

Discretization:

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

Concept Hierarchy Generation:

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

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 the dataset that is much smaller in volume but produces the same quality of analytical results. Here is a walkthrough of various Data Reduction strategies.

Data cube aggregation

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

- **Numerosity reduction**

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

- **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 difficultto 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.

- **Dimensionality reduction**

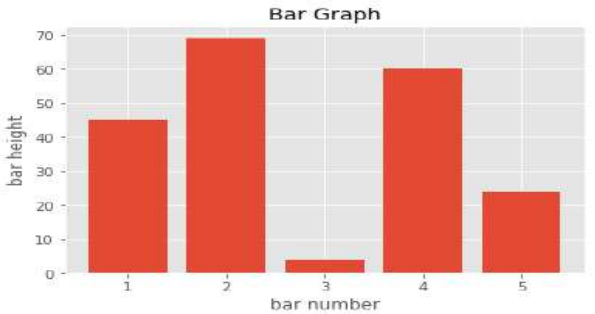
Dimensionality reduction techniques are used to perform feature extraction. The dimensionality of a dataset refers to the attributes orindividual features of the data. This technique aims to reduce the number of redundant features we consider in machine learning algorithms. Dimensionality reduction can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

Bar Graph

Bar graph are the graphs that can be used to show something that changes over time and to compare items. They have an x-axis (horizontal) and a y-axis (vertical).

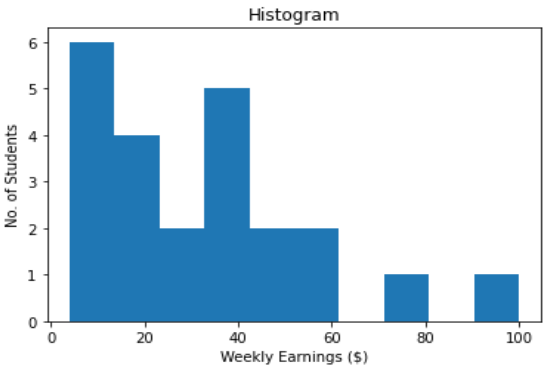



```
#import matplotlib
from matplotlib import pyplot as plt
#Bar Graph
plt.bar([1,2,3,4,5],[45,69,4,60,24])
plt.xlabel('bar number')
plt.ylabel('bar height')
plt.title("Bar Graph")
plt.show()
```

Histogram

- A Histogram is a graphical display of data using bars. This 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.

```
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()
```



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.

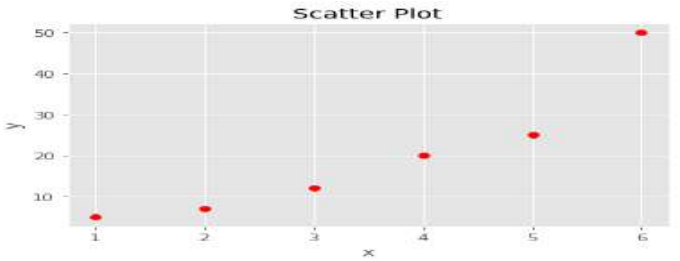
Code:

from matplotlib import pyplot as plt

x= [1,2,3,4,5,6]

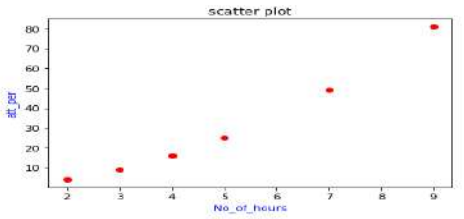
y= [5,7,12,20,25,50]

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]
```

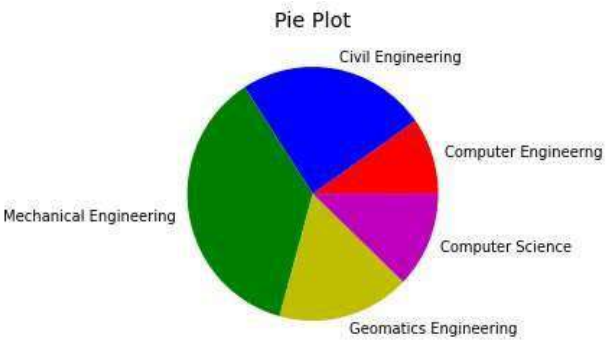



```
<Figure size 360x360 with 0 Axes>
```

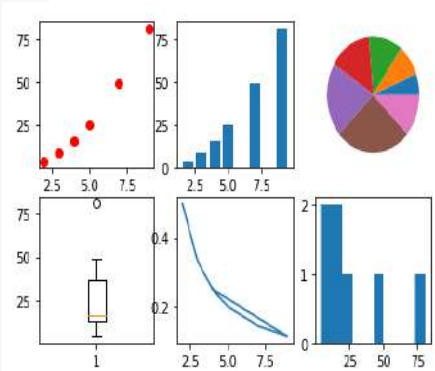
Pie Chart

- A pie chart is a circular statistical graphic, 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()
```



```
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]
```