Machine Learning-

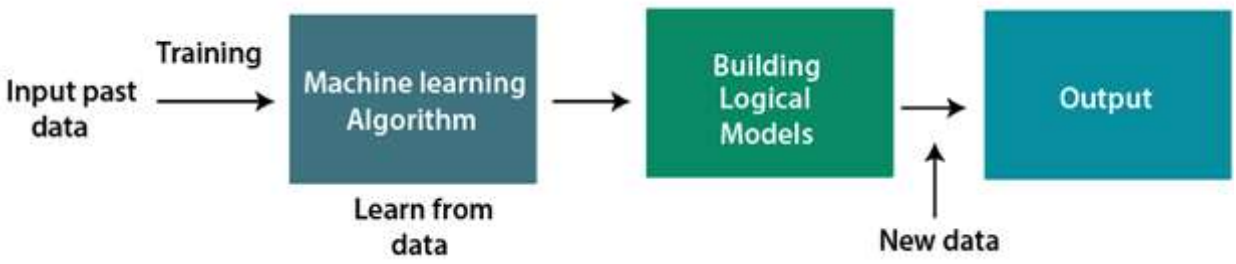
What Is Machine Learning-

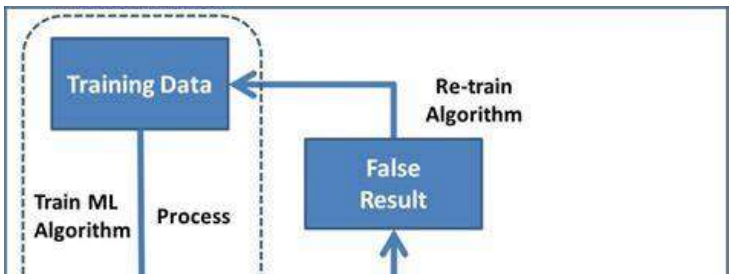
- In the real world, we are surrounded by humans who can learn everything from their experiences with their learning capability, and we have computers or machines which work on our instructions. But can a machine also learn from experiences or past data like a human does? So here comes the role of **Machine Learning.**
- Machine Learning is said as a subset of **artificial intelligence** that is mainly concerned with the development of algorithms which allow a computer to learn from the data and past experiences on their own. The term machine learning was first introduced by **Arthur Samuel** in **1959**. We can define it in a summarized way as:

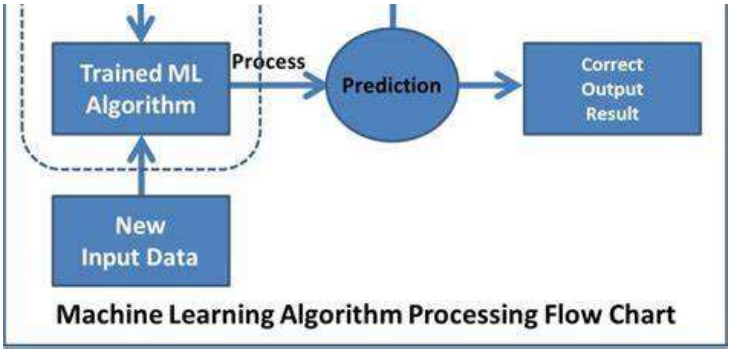
  ''Machine learning enables a machine to automatically learn from data, improve performance from experiences, and predict things without being explicitly programmed.''
- With the help of sample historical data, which is known as **training data**, machine learning algorithms build a **mathematical model** that helps in making predictions or decisions without being explicitly programmed. Machine learning brings computer science and statistics together for creating predictive models. Machine learning constructs or uses the algorithms that learn from historical data. The more we will provide the information, the higher will be performance.

How Does Machine Learning Works..??

- A Machine Learning system **learns from historical data, builds the prediction models, and whenever it receives new data, predicts the output for it**. The accuracy of predicted output depends upon the amount of data, as the huge amount of data helps to build a better model which predicts the output more accurately.
- Suppose we have a complex problem, where we need to perform some predictions, so instead of writing a code for it, we just need to feed the data to generic algorithms, and with the help of these algorithms, machine builds the logic as per the data and predict the output. Machine learning has changed our way of thinking about the problem. The below block diagram explains the working of Machine Learning algorithm:-







Features Of Machine Learning-

1. Machine learning uses data to detect various patterns in a given dataset.
2. It can learn from past data and improve automatically.
3. It is a data-driven technology.
4. Machine learning is much similar to data mining as it also deals with the huge amount of the data.

Need Of Machine Leaning-

- The need for machine learning is increasing day by day. The reason behind the need for machine learning is that it is capable of doing tasks that are too complex for a person to implement directly. As a human, we have some limitations as we cannot access the huge amount of data manually, so for this, we need some computer systems and here comes the machine learning to make things easy for us.
- We can train machine learning algorithms by providing them the huge amount of data and let them explore the data, construct the models, and predict the required output automatically. The performance of the machine learning algorithm depends on the amount of data, and it can be determined by the cost function. With the help of machine learning, we can save both time and money.
- The importance of machine learning can be easily understood by its uses cases, Currently, machine learning is used in **self-driving cars**, **cyber fraud detection**, **face recognition**, and **friend suggestion by Facebook**, etc. *Various top companies such as Netflix and Amazon have build machine learning models that are using a vast amount of data to analyze the user interest and recommend product accordingly.*
- **Following are some key points which show the importance of Machine Learning:**
  - Rapid increment in the production of data.
  - Solving complex problems, which are difficult for a human.
  - Decision making in various sector including finance.
  - Finding hidden patterns and extracting useful information from data.

Application Of Machine Learning-

- Machine learning is a buzzword for today's technology, and it is growing very rapidly day by day. We are using machine learning in our daily life even without knowing it such as Google

Maps, Google assistant, Alexa, etc. Below are some most trending real-world applications of Machine Learning:

1.Image Recognition-

- Image recognition is one of the most common applications of machine learning. It is used to identify objects, persons, places, digital images, etc. The popular use case of image recognition and face detection is, **Automatic friend tagging suggestion**:
- Facebook provides us a feature of auto friend tagging suggestion. Whenever we upload a photo with our Facebook friends, then we automatically get a tagging suggestion with name, and the technology behind this is machine learning's **face detection** and **recognition algorithm**.
- It is based on the Facebook project named "**Deep Face**," which is responsible for face recognition and person identification in the picture.

2.Speech Recognition-

- While using Google, we get an option of "**Search by voice**," it comes under speech recognition, and it's a popular application of machine learning.
- Speech recognition is a process of converting voice instructions into text, and it is also known as "**Speech to text**", or "**Computer speech recognition**." At present, machine learning algorithms are widely used by various applications of speech recognition. **Google assistant**, **Siri**, **Cortana**, and **Alexa** are using speech recognition technology to follow the voice instructions.

3.Traffic Prediction-

- If we want to visit a new place, we take help of Google Maps, which shows us the correct path with the shortest route and predicts the traffic conditions.
- It predicts the traffic conditions such as whether traffic is cleared, slow-moving, or heavily congested with the help of two ways:
  - **Real Time location** of the vehicle form Google Map app and sensors.
  - **Average time has taken** on past days at the same time.
- Everyone who is using Google Map is helping this app to make it better. It takes information from the user and sends back to its database to improve the performance.

4.Product Recommendations-

- Machine learning is widely used by various e-commerce and entertainment companies such as **Amazon**, **Netflix**, etc., for product recommendation to the user. Whenever we search for some product on Amazon, then we started getting an advertisement for the same product while internet surfing on the same browser and this is because of machine learning.
- Google understands the user interest using various machine learning algorithms and suggests the product as per customer interest.
- As similar, when we use Netflix, we find some recommendations for entertainment series, movies, etc., and this is also done with the help of machine learning.

5.Self Driving Cars-

- One of the most exciting applications of machine learning is self-driving cars. Machine learning plays a significant role in self-driving cars. Tesla, the most popular car manufacturing company is working on self-driving car. It is using unsupervised learning method to train the car models to detect people and objects while driving.

6.Virtual Personal Assistant-

- We have various virtual personal assistants such as **Google assistant**, **Alexa**, **Cortana**, **Siri**. As

the name suggests, they help us in finding the information using our voice instruction. These assistants can help us in various ways just by our voice instructions such as Play music, call someone, Open an email, Scheduling an appointment, etc.

- These virtual assistants use machine learning algorithms as an important part.
- These assistant record our voice instructions, send it over the server on a cloud, and decode it using ML algorithms and act accordingly.

7.Online Fraud Detection-

- Machine learning is making our online transaction safe and secure by detecting fraud transaction. Whenever we perform some online transaction, there may be various ways that a fraudulent transaction can take place such as **fake accounts**, **fake ids**, and **steal money** in the middle of a transaction. So to detect this, **Feed Forward Neural network** helps us by checking whether it is a genuine transaction or a fraud transaction.
- For each genuine transaction, the output is converted into some hash values, and these values become the input for the next round. For each genuine transaction, there is a specific pattern which gets change for the fraud transaction hence, it detects it and makes our online transactions more secure.

8.Stock Market Trading-

- Machine learning is widely used in stock market trading. In the stock market, there is always a risk of up and downs in shares, so for this machine learning's **long short term memory neural network** is used for the prediction of stock market trends.

9.Medical Diagnosis-

- In medical science, machine learning is used for diseases diagnoses. With this, medical technology is growing very fast and able to build 3D models that can predict the exact position of lesions in the brain.
- It helps in finding brain tumors and other brain-related diseases easily.

10.Automatic Language Translator-

- Nowadays, if we visit a new place and we are not aware of the language then it is not a problem at all, as for this also machine learning helps us by converting the text into our known languages. Google's GNMT (Google Neural Machine Translation) provide this feature, which is a Neural Machine Learning that translates the text into our familiar language, and it called as automatic translation.
- The technology behind the automatic translation is a sequence to sequence learning algorithm, which is used with image recognition and translates the text from one language to another language.

Classification Of Machine Learning-

- At a broad level, machine learning can be classified into three types:
  1. **Supervised learning.**
  2. **Unsupervised learning.**
  3. **Reinforcement learning.**

Supervised Learning-

- Supervised learning is a type of *machine learning method in which we provide* ***sample labeled***

data *to the machine learning system in order to train it, and on that basis, it predicts the output.*

- The system creates a model using labeled data to understand the datasets and learn about each data, once the training and processing are done then we test the model by providing a sample data to check whether it is predicting the exact output or not.
- The goal of supervised learning is to map input data with the output data. The supervised learning is based on supervision, and it is the same as when a student learns things in the supervision of the teacher. The example of supervised learning is **spam filtering**
- **Supervised learning can be grouped further in two categories of algorithms:**
  - Classification-
  - Regression-
- **Popular Supervised Algorithms-**
  - Linear Regression-
  - Support Vector Machine-
  - Random Forest-
- **Common Use cases Of Supervised Algorithns**

1)Banking-

i) Used to predict the credit worthiness of credit card holder

2)Healthcare Center-

i)Used to Predict patient readmission rates

3)Retail Sector-

i)Used to analyze the product that a customer by together.

Unsupervised Learning-

- *Unsupervised learning is a learning method in which a machine learns without any supervision.*
- The training is provided to the machine with the *set of data that has not been labeled, classified, or categorized, and the algorithm needs to act on that data without any supervision.* The goal of unsupervised learning is to restructure the input data into new features or a group of objects with similar patterns.
- *In unsupervised learning, we don't have a predetermined result.* The machine tries to find useful insights from the huge amount of data. It can be further classifieds into two categories of algorithms:
  - **Clustering**
  - **Association**
- **Popular Un-Supervised Algorithms-**
  - Apriori Algorithm.
  - K-means Algorithm.
  - Hierarchical Clustering.
- **Common Use cases Of Un-Supervised Algorithms**

1)Banking-

i) Segment Customer by behavioral characteristics by surveying prospects and customers to develop multiple segments using clustering.

2)Healthcare Center-

i)Categorize the MRI data by normal or abnormal images.

3)Retail Sector-

3)Retail Sector-

i)used to recommend the product to customer based on their past purchases.

- [ ] **Reinforcement Learning-**

- Reinforcement learning is a feedback-based learning method, in which a learning agent gets a reward for each right action and gets a penalty for each wrong action. The agent learns automatically with these feedbacks and improves its performance. In reinforcement learning, the agent interacts with the environment and explores it. The goal of an agent is to get the most reward points, and hence, it improves its performance.
- The robotic dog, which automatically learns the movement of his arms, is an example of Reinforcement learning.
- **Common Use cases Of Reinforcement Algorithms-**

1)Banking-

i) Used to create next best offer model for a call center.

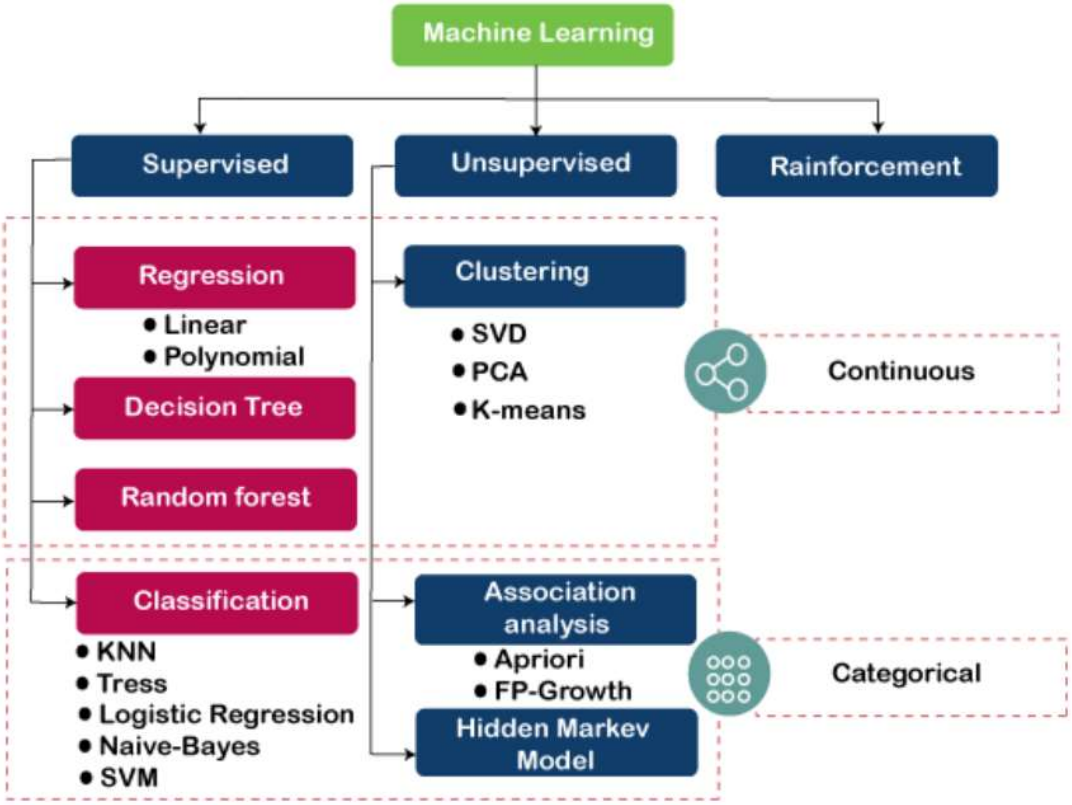
2)Healthcare Center-

i)Used to allocate scars medial resources to handle different types of ER cases.

3)Retail Sector-

i)used to reduce excess stock by dynamic pricing.

Machine Learning algorithm and it Types



- [ ] **AL,ML and Deep Learning-**

1) Artificial Intelligence-

- Artificial intelligence is a technique which allows machines to mimic human behaviour.

2) Machine Learning -

- Machine learning is a subset of AI which use statistical methods to allows machines to improve

Machine learning is a subset of AI which use statistical methods to allows machines to improve with experience.

3)Deep Learning -

- Deep learning is a particular kind of machine learning that is inspired by the functionality of our brain cells called neurons which led to the concept of artificial neural network.

- [ ] **Mathematics For Data Science-**

- Mathematics is the core of designing ML algorithms that can automatically learn from data and make predictions. Therefore, it is very important to understand the Maths before going into the deep understanding of ML algorithms.
- There is always a question in enthusiast learners that what is the need of mathematics in machine learning? As computers can solve mathematics problems faster than humans. ***So, the answer is, learning mathematics in machine learning is not about solving a maths problem, rather understanding the application of maths in ML algorithms and their working***.
- Other below points explain the significance of maths in ML:
  - Mathematics defines the concept behind the ML algorithms & helps in choosing the right algorithm by considering *accuracy, training time, the complexity of the model, number of features.*
  - Computers understand data differently than humans, such as an image is seen as a 2D-3D matrix by a computer for which mathematics is required.
  - With Maths, we can correctly determine the interval & uncertainty.
  - It helps in selecting correct parameter values and validation methods.
  - Understanding the Bias-Variance trade-off helps us identify underfitting and overfitting issues that are the main issues in ML models.

1) Median-

- A median is a number that is separated by the higher half of a data sample, a population or a probability distribution, from the lower half.
- If we have outliers in our dataset it is best choice to use Median because mean can be affected by outliers.
- Median is nothing but middle value from our dataset when datapoints are arranged with ascending order.
- Outliers- Outliers is nothing but datapoints which are very different from rest of the datapoints.
- Median allows you to come with right guess while filling null values.
- When number of datapoints are even,then take mean of middle two values.
- Use cases of Median-

  1) In simple Descriptive Statistics.

  2) Handling Missing values from dataset.

2) Mean-

- Mean,is nothing but the average value of the given numbers or data.
- If our data is balanced and does not contains any outliers then we can use mean for filling missing values from dataset.
- To calculate the mean, we need to add the total values given in a datasheet and then divide the

sum by the total number of values.

- Use cases of Mean-

  1) In simple descriptive statistics.

  2) Handling missing values from dataset.

3) Mode-

- A mode is defined as the **value that has a higher frequency in a given set of values**. It is the value that appears the most number of times. Example: In the given set of data: 2, 4, 5, 5, 6, 7, the mode of the data set is 5 since it has appeared in the set twice.
- Use cases of Mode-

  1) In simple descriptive statistics.

  2) Fill missing values from categorical features.

4) Percentile-

- Percentiles are used to understand and interpret data. The n th percentile of a set of data is the **value at which n percent of the data is below it**. In everyday life, percentiles are used to understand values such as test scores, health indicators, and other measurements.
- For example-

Name>>	Naina	Abhimanyu	Sudarshan	Rohan	Roshan	Dipak	Snehal	Rishi	Neha
Salary>>	4500	8500	7000	5500	6500	9500	10000	4000	5000
0 th percentile				50 th Percentile				100 th Percentile	

- 50 th percentile >> It means 50% datapoints are at left side and right side of this point.
- 25 th percentile >> It means 25% datapoints are at left side and 75 % datapoints are at right side of this point.
- 75 th percentile >> It means 75% datapoints are at left side and 25% datapoints are at right side of this point.
- The range between 25 th to 75 th percentile is called as Interquartile range.

5) Variance-

- The variance is one of the measures of dispersion which is a measure of how much the values in the data set may differ from the average of the values.
- t is an average of the squares of the deviations from the average.
- To calculate the Variance, take each difference of values from mean, square it, and then average the result.

6) Standard Deviation-

- Standard deviation in data science is the measure of the dispersion/variation of a set of data from its mean.
- It measures the absolute variability of a distribution; the higher the dispersion is, the greater is the standard deviation will be and greater will be the magnitude of the deviation of the value from their mean.
- SD is applicable in comparing sets of data that may have the same mean but a different range.
- Standard Deviation is just the square root of Variance.

Hypothesis Testing-

- A statistical hypothesis is an assumption made about a population parameter. This assumption may or may not be right.

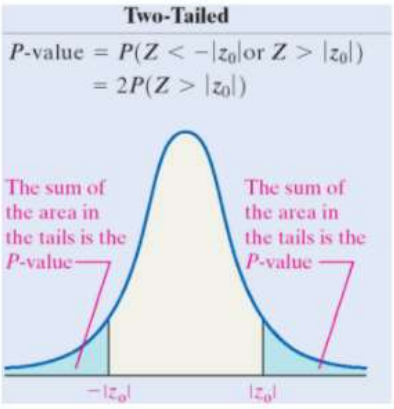
- Hypothesis testing is a formal procedure used by statisticians to approve or disapprove a statistical hypothesis.
- ***"A fact is a simple statement that everyone believes. It is innocent, unless found guilty. A hypothesis is a novel suggestion that no one wants to believe. It is guilty, until found effective."***
- Hypothesis testing is nothing but eliminate factor of randomness from the claim that you are testing.
- We have to evaluate two mutual exclusive statements on population using sample data.
- Steps in Hypothesis Testing-
  1) Make Initial Assumption(H0=Null Hypothesis)
  2) Collect Data(Collecting Evidences)
  3)Gather evidences to reject or accept null hypothesis.
- If evidences is against null hypothesis then we have to reject null hypothesis. And that means we are accepting Alternate hypothesis(H1).
- There are basically five Method to conduct Hypothesis testing,

1) P Value Test-

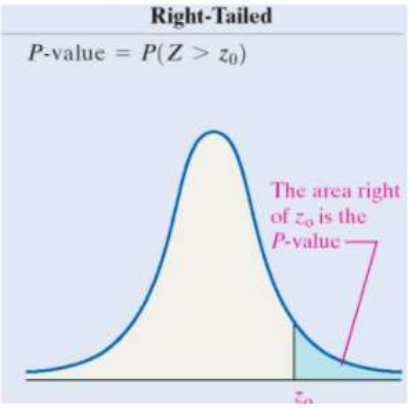
- The p-value is the probability of getting the observed value of the test statistic, or a value with even greater evidence against H_0 if the null hypothesis is actually true.
- The p-value is the level of marginal significance within a statistical hypothesis test that represents the probability of a particular event occurring. The p-value is used as an alternate method to reject points to provide the smallest level of significance at which the null hypothesis would be rejected. A smaller p-value means that there is stronger evidence to prove an alternative hypothesis.
- P-value is the area or the region or the size of test statistical value.
- **Example-**Suppose an investor claims that their investment portfolio's performance is nearly identical to that of the Standard & Poor's (S&P) 900 Index. In order to identify this, the investor performed a two-tailed test.

Assume that the null hypothesis is true.
The P-Value is the probability of observing a sample mean that is as or more extreme than the observed.
How to compute the P-Value for each type of test:
Step 1: Compute the test statistic $z_0 = \dfrac{\bar{x} - \mu_0}{\sigma / \sqrt{n}}$

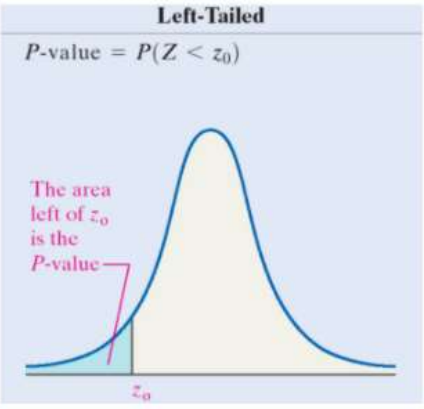
Two-tail



Right Tail



Left Tail



- The null hypothesis states that the portfolio's returns are nearly equal to the S&P 900's returns over a specified period whereas the alternative hypothesis tells that the portfolio's returns and the S&P 900's returns are not equivalent. If the investor conducts a one-tailed test, the alternative hypothesis will tell that a portfolio's returns are either less than or greater than the S&P 900's returns.
- One normally uses the p-value is 0.05. If the investor comes to the conclusion that the p-value is less than 0.05, there is strong evidence against the null hypothesis. Henceforth, the investor will reject the null hypothesis and accept the alternative hypothesis.
- On the other hand, if the p-value is greater than 0.05, that indicates that there is weak evidence against the supposition, so the investor will fail to reject the null hypothesis.
- If the investor determines that the p-value is 0.001, there is strong evidence against the null hypothesis, and the portfolio's returns and the S&P 900's returns may not be nearly the same.
- In the above example, p-value helps the investor in determining the risk.

2) T test-

- When we have a continuous type of feature for evaluate significant value then we can use T test over there.

3)One Sample Proportion Test-

- When we have one categorical feature from dataset to evaluate significant value,then we can use One Sample Proportion Test for hypothesis Testing.

4) CHI Square Test-

- When we have two categorical features to evaluate significant value,then we can use CHI square test for Hypothesis Testing.

5) ANOVA Test-

- If we have two features and among this one is numeric and another is categorical with more than two categories in it then we have to use ANOVA Test.

Pandas-

- Pandas is defined as an open-source library that provides high-performance data manipulation in Python. The name of Pandas is derived from the word **Panel Data**, which means **an Econometrics from Multidimensional data**. It is used for data analysis in Python and developed by **Wes McKinney** in **2008**.
- Data analysis requires lots of processing, such as **restructuring, cleaning** or **merging**, etc. There are different tools are available for fast data processing, such as **Numpy, Scipy, Cython**, and **Panda**. But we prefer Pandas because working with Pandas is fast, simple and more expressive than other tools.
- Pandas is built on top of the **Numpy** package, means **Numpy** is required for operating the Pandas.
- Before Pandas, Python was capable for data preparation, but it only provided limited support for data analysis. So, Pandas came into the picture and enhanced the capabilities of data analysis. It can perform five significant steps required for processing and analysis of data irrespective of the origin of the data, i.e., **load, manipulate, prepare, model, and analyze**.

Key Features Of Pandas-

- It has a fast and efficient DataFrame object with the default and customized indexing.

- It has a fast and efficient DataFrame object with the default and customized indexing.
- Used for reshaping and pivoting of the data sets.
- Group by data for aggregations and transformations.
- It is used for data alignment and integration of the missing data.
- Provide the functionality of Time Series.
- Process a variety of data sets in different formats like matrix data, tabular heterogeneous, time series.
- Handle multiple operations of the data sets such as subsetting, slicing, filtering, groupBy, re-ordering, and re-shaping.
- It integrates with the other libraries such as SciPy, and scikit-learn.
- Provides fast performance, and If you want to speed it, even more, you can use the **Cython**.

- [ ] **Benefits Of Pandas-**

- The benefits of pandas over using other language are as follows:
  - **Data Representation:** It represents the data in a form that is suited for data analysis through its DataFrame and Series.
  - **Clear code:** The clear API of the Pandas allows you to focus on the core part of the code. So, it provides clear and concise code for the user.

- [ ] **Read Data In The Form Of DataFrame-**

- DataFrames store data in the form of rectangular grids by which the data can be over viewed easily. Each row of the rectangular grid contains values of an instance, and each column of the grid is a vector which holds data for a specific variable.
- This means that rows of a DataFrame do not need to contain, values of same data type, they can be numeric, character, logical, etc. DataFrames for Python come with the Pandas library, and they are defined as two-dimensional labeled data structures with potentially different types of columns.
- Pandas DataFrame is a widely used data structure which works with a two-dimensional array with labeled axes (rows and columns). DataFrame is defined as a standard way to store data that has two different indexes, i.e., **row index** and **column index**. It consists of the following properties:
  - The columns can be heterogeneous types like int, bool, and so on.
  - It can be seen as a dictionary of Series structure where both the rows and columns are indexed. It is denoted as "columns" in case of columns and "index" in case of rows.

1.Read Data by Using CSV-

- Syntax-
  pandas.read_csv('<csv file path>')

```
In [11]: df = pd.read_csv('C:\\Users\Abhimanyu Devadhe\Desktop\\csv.csv')
```

```
In [12]: df
```

Out[12]:

	Emp Id	Emp Name	Emp Salary	Emp Department

0	1	Jon	85000	HR
1	2	Mikel	95200	Finance
2	3	David	98563	Development
3	4	Kevin	45000	Testing

- We converted CSV.csv file into DataFrame.
- Until we have seen 2 methods for read csv file.

1. Using csv module,
2. Using pandas.

- To find current working Directory in Jupyter Notebook we use command - *PWD*

---

2. Read Data by using Excel-

- Syntax-

pandas.read_excel('path of excel file',sheet_name='<sheet name from Excel')

```
In [31]: df1=pd.read_excel('C:\\Users\Abhimanyu Devadhe\Desktop\\excel.xlsx',sheet_name='Sheet2')
```

```
In [32]: df1
```

Out[32]:

	Id	Name	Pass	DOJ
0	1	Jon	12345	2012-12-12
1	2	Kevin	12345	2012-11-12
2	3	Martin	12345	2018-12-11

- We have successfully import excel sheet in the form of DataFrame.
- We can mention sheet name from Excel file,otherwise it will read first sheet by default.

3.Read Data From Dictionary-

- Syntax-

pandas.DataFrame(<dict_name>)

```
In [38]: emp_records={'Id':[1,2,3,4],'Name':['A','B','C','D'],'Location':['Pune','Dhule','Nashik','Jalna']}
```

```
In [40]: df3=pd.DataFrame(emp_records)
```

```
In [41]: df3
```

Out[41]:

	Id	Name	Location
0	1	A	Pune
1	2	B	Dhule
2	3	C	Nashik
3	4	D	Jalna

- We have successfully read data from Dictionary.

- [ ] **Creating Data Frame by Using Tuple List-**

- Syntax-

pandas.DataFrame(<List Variable name>,Columns=['<col 1>','<col 2>','<col 3>']

```
In [46]: emp_list=[('01/12/2022','ML Engg','Jon'),
                   ('02/11/2021','DA','Thomas'),
                   ('04/10/2021','DS','Jason')]
```

```
In [47]: df5=pd.DataFrame(emp_list,columns=['DOJ','Designation','Name'])
```

```
In [48]: df5
```

Out[48]:

	DOJ	Designation	Name
0	01/12/2022	ML Engg	Jon
1	02/11/2021	DA	Thomas
2	04/10/2021	DS	Jason

- We can give columns name to DataFrame as above.
- We can also covert above data in csv,excel,Dict,jason,pickle,sql, html etc-

```
In [49]: df5.to_csv('csv1.csv')
```

```
In [53]: df5.to_excel('excel2.xlsx')
```

```
In [54]: df5.to_json('jason.jason')
```

```
In [56]: df5.to_dict('dict')
```

```
Out[56]: {'DOJ': {0: '01/12/2022', 1: '02/11/2021', 2: '04/10/2021'},
          'Designation': {0: 'ML Engg', 1: 'DA', 2: 'DS'},
          'Name': {0: 'Jon', 1: 'Thomas', 2: 'Jason'}}
```

- Likewise we can convert data in any file format for backup/storage purpose.

- [ ] **Read Data from List of Dictionary-**

- Syntax-

pandas.DataFrame(<List Of Dict_Name>)

```
In [85]: p=[{'Name':'Abhimanyu','Designation':'ML Engineer','Salary':85000},
            {'Name':'Sudarshan','Designation':'DA','Salary':68500},
            {'Name':'Jon','Desiganation':'DS','Salary':98326}]
```

```
In [86]: df6=pd.DataFrame(p)
```

```
In [87]: df6
```

Out[87]:

	Name	Designation	Salary	Desiganation
0	Abhimanyu	ML Engineer	85000	NaN
1	Sudarshan	DA	68500	NaN
2	Jon	NaN	98326	DS

- [ ] **Read Data using Web Scraping-**

- Syntax-
  pandas.read_html('<html link>')

```
In [91]: df7=pd.read_html('https://en.wikipedia.org/wiki/Virat_Kohli')
```

```
In [99]: df7[1]
```

Out[99]:

	Competition	Test	ODI	T20I	FC
0	Matches	99	250	97	131
1	Runs scored	7962	12285	3296	10211
2	Batting average	50.39	58.78	52.50	51.50
3	100s/50s	27/28	43/64	0/30	34/36
4	Top score	254*	183	94*	254*
5	Balls bowled	175	641	146	643
6	Wickets	0	4	4	3
7	Bowling average	–	166.25	49.50	112.66

- We can easily read data from html link with web scraping.
- *Data which is extract from web will be always in the form of List.*

- [ ] **Read Data From Jason-**

- Syntax-
  pandas.read_jason('jason api link')

```
In [100]: df8=pd.read_json('https://www.7timer.info/bin/astro.php?lon=113.2&lat=23.1&ac=0&ur
```

```
In [101]: df8
```

Out[101]:

	product	init	dataseries
0	astro	2022021906	{'timepoint': 3, 'cloudcover': 9, 'seeing': 7,...
1	astro	2022021906	{'timepoint': 6, 'cloudcover': 9, 'seeing': 7,...
2	astro	2022021906	{'timepoint': 9, 'cloudcover': 9, 'seeing': 7,...
3	astro	2022021906	{'timepoint': 12, 'cloudcover': 9, 'seeing': 7...
4	astro	2022021906	{'timepoint': 15, 'cloudcover': 9, 'seeing': 7...

- Likewise we can read Jason in Pandas.

- [ ] **Operations On DataFrame-**

- We can perform many Operations on DataFrame like Fetch columns, Add single column,Add

multiple column,Encode columns,Delete columns etc.

1.Fetch Columns and Rows from DataFrame-

- Syantax-

DF.shape

```
In [3]: df
```

Out[3]:

	Emp Id	Emp Name	Emp Salary	Emp Department	EDOJ	Eloc	Age	Gender
0	1	Jon	85000	HR	12-12-2008	Pune,MH	35 year	Male
1	2	Mikel	95200	Finance	11-08-2011	Nashik,MH	25 Year	Male
2	3	David	98563	Development	08-05-2020	Varanasi,UP	22 Year	Male
3	4	Laila	45000	Testing	27-09-2015	Delhi,Delhi	27 Year	Female
4	5	Jason	69879	Development	28-09-2015	Jaipur,RJ	29 Year	Male
5	6	Priety	45979	Development	29-09-2015	Kolakata,WB	31 Year	Female
6	7	Naim	68796	Testing	30-09-2015	Nagpur,MH	30 Year	Male

```
In [4]: df.shape
Out[4]: (7, 8)
```

- In above we can observe that 7 rows and 8 columns are present DataFrame.

2.Fetch Specific column from DataFrame-

- Syntax-

DF['<Column_Name> ']

```
In [5]: df['Eloc']
Out[5]: 0        Pune,MH
        1      Nashik,MH
        2    Varanasi,UP
        3    Delhi,Delhi
        4      Jaipur,RJ
        5    Kolakata,WB
        6      Nagpur,MH
        Name: Eloc, dtype: object
```

- Data type will be series data i.e 1 Dimensional array.

3.List of Columns From DataFrame-

- Syntax-

DF.Columns

```
In [7]: df.columns
Out[7]: Index(['Emp Id', 'Emp Name', 'Emp Salary', 'Emp Department', 'EDOJ', 'Eloc',
               'Age', 'Gender'],
              dtype='object')
```

- Will give list of columns available in DataFrame.

4.Fetch Multiple Columns from DataFrame-

- Syntax-

DF[[' Col_1 ', ' Col_2 ', ' Col_3 ']]

```
In [12]: df[['Emp Id','Emp Salary','EDOJ','Emp Department']]
```

Out[12]:

	Emp Id	Emp Salary	EDOJ	Emp Department
0	1	85000	12-12-2008	HR
1	2	95200	11-08-2011	Finance
2	3	98563	08-05-2020	Development
3	4	45000	27-09-2015	Testing

- With the help of above command we can manipulate data.

5.Add new Column in DataFrame-

- Syntax-

DF['<Column_Name ']= Value

- Value will allocate to whole rows from this newly created column.

```
In [14]: import datetime
```

```
In [15]: df['Last_Updated']=datetime.datetime.now()
```

```
In [16]: df
```

Out[16]:

	Emp Id	Emp Name	Emp Salary	Emp Department	EDOJ	Eloc	Age	Gender	Last_Updated
0	1	Jon	85000	HR	12-12-2008	Pune,MH	35 year	Male	2022-02-21 15:32:08.704515
1	2	Mikel	95200	Finance	11-08-2011	Nashik,MH	25 Year	Male	2022-02-21 15:32:08.704515
2	3	David	98563	Development	08-05-2020	Varanasi,UP	22 Year	Male	2022-02-21 15:32:08.704515

- New created column always located at the end of dataframe.

6.Create new Dataframe from using selected columns of existing Dataframe-

- Syntax-

DF1=DF[['Emp Id', 'Emp Name ', ' Emp Salary' , 'EDOJ']]

```
In [18]: df1=df[['Emp Id','Emp Name','Emp Salary','EDOJ']]
```

```
In [19]: df1
```

Out[19]:

	Emp Id	Emp Name	Emp Salary	EDOJ
0	1	Jon	85000	12-12-2008
1	2	Mikel	95200	11-08-2011
2	3	David	98563	08-05-2020

- columns name should be give in list of list,otherwise it will show exception i.e Key Error.

7.Clean data using Lambda Function-

- Syntax-

df1['New_Column']=df1['Old_Column'].apply (lambda_Function)

```
In [26]: df1['Eloc_New']=df1['Eloc'].apply(lambda x:x.split (',')[1])
```

```
<ipython-input-26-e7d8e96e1794>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/s
rsus-a-copy
  df1['Eloc_New']=df1['Eloc'].apply(lambda x:x.split (',')[1])
```

```
In [27]: df1
```

Out[27]:

	Emp Id	Emp Name	Emp Salary	EDOJ	Eloc	Eloc_New
0	1	Jon	85000	12-12-2008	Pune,MH	MH
1	2	Mikel	95200	11-08-2011	Nashik,MH	MH
2	3	David	98563	08-05-2020	Varanasi,UP	UP

- You have to always mention New_Column_Name and on which column you have to perform operation as syntax.

8.Operation on specific Column-

- If you to want to give salary hike to all your employees ,
- Syntax-

DF['New_Column_Name']=DF['Old_Column_Name'].apply(lambda_Function)

```
In [35]: df1['Updated_Salary']=df1['Emp Salary'].apply(lambda x:x*1.3)
```

```
<ipython-input-35-2032ca1e3002>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/use
rsus-a-copy
  df1['Updated_Salary']=df1['Emp Salary'].apply(lambda x:x*1.3)
```

```
In [37]: df1
```

Out[37]:

	Emp Id	Emp Name	Emp Salary	EDOJ	Eloc	Eloc_New	Updated_Salary
0	1	Jon	85000	12-12-2008	Pune,MH	MH	110500.0
1	2	Mikel	95200	11-08-2011	Nashik,MH	MH	123760.0
2	3	David	98563	08-05-2020	Varanasi,UP	UP	128131.9

9.Encoding Columns-

- We can Encode columns with various techniques.

A)Using Lambda Function-

- Syntax

df1['New_Column_Name']=df1['Old_Column'].apply(Lambda_Function)

```
In [45]: df1['Gender_Encode_Lambda']=df1['Gender'].apply(lambda x:1 if x=='Male' else 0 )
```

```
<ipython-input-45-3bbc8d100bdc>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/us
rsus-a-copy
  df1['Gender_Encode_Lambda']=df1['Gender'].apply(lambda x:1 if x=='Male' else 0 )
```

```
In [46]: df1
```

Out[46]:

	Emp Id	Emp Name	Emp Salary	EDOJ	Eloc	Gender	Gender_Encode_Lambda
0	1	Jon	85000	12-12-2008	Pune,MH	Male	1
1	2	Mikel	95200	11-08-2011	Nashik,MH	Male	1
2	3	David	98563	08-05-2020	Varanasi,UP	Male	1
3	4	Laila	45000	27-09-2015	Delhi,Delhi	Female	0

- Lambda function is probably used when we want apply normally 2 conditions.

B)Using Map function-

- Syntax-

df1['New_Column_Name']=df1['Old_Column'].map(Pass condition in Dictionary format)

```
In [49]: df1['Encode_Gender_Map']=df1['Gender'].map({'Male':1,'Female':0})
```

```
<ipython-input-49-745eb908d82d>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user
rsus-a-copy
  df1['Encode_Gender_Map']=df1['Gender'].map({'Male':1,'Female':0})
```

```
In [50]: df1
```

Out[50]:

	Emp Id	Emp Name	Emp Salary	EDOJ	Eloc	Gender	Encode_Gender_Map
0	1	Jon	85000	12-12-2008	Pune,MH	Male	1
1	2	Mikel	95200	11-08-2011	Nashik,MH	Male	1
2	3	David	98563	08-05-2020	Varanasi,UP	Male	1
3	4	Laila	45000	27-09-2015	Delhi,Delhi	Female	0

- Map function is probably used when we want apply normally 4 to 5 conditions.

C)Using Normal Function-

- Normal function is probably used when we want apply multiple conditions.

In [61]:
```
def encode(x):
    if x=='Male':
        return 1
    else:
        return 0
```

In [62]:
```
df1['Encode_Gender_NorFun']=df1['Gender'].apply(encode)
```

```
<ipython-input-62-e78db381dfa1>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/use
rsus-a-copy
  df1['Encode_Gender_NorFun']=df1['Gender'].apply(encode)
```

In [63]:
```
df1
```

Out[63]:

	Emp Id	Emp Name	Emp Salary	EDOJ	Eloc	Gender	Encode_Gender_NorFun
0	1	Jon	85000	12-12-2008	Pune,MH	Male	1
1	2	Mikel	95200	11-08-2011	Nashik,MH	Male	1
2	3	David	98563	08-05-2020	Varanasi,UP	Male	1
3	4	Laila	45000	27-09-2015	Delhi,Delhi	Female	0

10.Delete Column from DataFrame-

- Syntax-

df.drop('Column_Name', axis=1,inplace=True)

In [66]:
```
df1.drop('Encode_Gender_NorFun',axis=1,inplace=True)
```

```
C:\Users\Abhimanyu Devadhe\anaconda3\lib\site-packages\pandas\core\frame.p
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-doc
rsus-a-copy
  return super().drop(
```

In [67]:
```
df1
```

Out[67]:

	Emp Id	Emp Name	Emp Salary	EDOJ	Eloc	Gender
0	1	Jon	85000	12-12-2008	Pune,MH	Male
1	2	Mikel	95200	11-08-2011	Nashik,MH	Male
2	3	David	98563	08-05-2020	Varanasi,UP	Male
3	4	Laila	45000	27-09-2015	Delhi,Delhi	Female

- Axis- Axis have 2 arguments,

  i)0 =Interpreter will search given column name in rows of DataFrame.

  ii)1 =Interpreter will search given column name in Column of DataFrame.
- Inplace- Inplace also have 2 Arguments.Inplace method is use for to commit changes in DataFrame.

i) True= Will commit changes on Dataframe.

ii) False= Changes will appear but wont commit.

11.Delete Multiple Columns From DataFrame-

- Syntax-

DF.drop(columns=['Col_1',' Col_2', 'Col_3'],axis=1,inplace=True)

```
In [70]: df1.drop( columns=[ 'EDOJ','Eloc', 'Gender'],axis=1,inplace=True)
```

```
C:\Users\Abhimanyu Devadhe\anaconda3\lib\site-packages\pandas\core\frame.py:4308:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable
rsus-a-copy
  return super().drop(
```

```
In [71]: df1
```

Out[71]:

	Emp Id	Emp Name	Emp Salary
0	1	Jon	85000
1	2	Mikel	95200
2	3	David	98563

- Required proper syntax,if you mistakenly give space between column name it will thow error.

12.Split DOJ as Date,Month and Year-

```
In [50]: df['Date']=df['EDOJ'].apply(lambda x:x.split('-')[0])
```

```
In [51]: df['Month']=df['EDOJ'].apply(lambda x:x.split('-')[1])
```

```
In [52]: df['Year']=df['EDOJ'].apply(lambda x:x.split('-')[2])
```

```
In [53]: df
```

Out[53]:

	Emp Id	Emp Name	Emp Salary	Emp Department	EDOJ	Eloc	Age	Gender	Last_Updated	Date	Month	Year
0	1	Jon	85000	HR	12-12-2008	Pune,MH	35 year	Male	2022-02-21 21:35:18.972650	12	12	2008
1	2	Mikel	95200	Finance	11-08-2011	Nashik,MH	25 Year	Male	2022-02-21 21:35:18.972650	11	08	2011
2	3	David	98563	Development	08-05-2020	Varanasi,UP	22 Year	Male	2022-02-21 21:35:18.972650	08	05	2020
3	4	Laila	45000	Testing	27-09-2015	Delhi,Delhi	27 Year	Female	2022-02-21 21:35:18.972650	27	09	2015
4	5	Jason	69879	Development	28-09-2015	Jaipur,RJ	29 Year	Male	2022-02-21 21:35:18.972650	28	09	2015

- Above scenario is frequently used for data manipulation/Cleaning.

☐ Data Definition Operations-

- We can perform Drop,Rename,Replace, Group by and Concatenation operations on columns.

1.Find Null Values from columns-

- We can find null values from each columns.
- Syntax-

df.isnull( )

```
In [5]: #Find Null values from Date Frame-
```

```
In [7]: df.isnull()
```

Out[7]:

	show_id	type	title	director	cast	country	date_added	release_year	rating	duration	listed_in	description
0	False	False	False	False	True	False	False	False	False	False	False	False
1	False	False	False	True	False	False	False	False	False	False	False	False
2	False	False	False	False	False	True	False	False	False	False	False	False

- From above syntax we get output in True or False.
- We can also find null values count from each columns-
- Syntax-

df.isnull( ).sum( )

```
In [8]: df.isnull().sum()

Out[8]: show_id          0
        type             0
        title            0
        director      2634
        cast           825
        country        831
        date_added      10
        release_year     0
        rating           4
```

- We get each columns null value count.

---

2. To find Unique values from specific columns-

- We can find unique values from any columns to filtrate data accordingly.
- Syntax-

df.['<Column_Name>'].Unique( )

```
In [13]: df['date_added'].unique()

Out[13]: array(['September 25, 2021', 'September 24, 2021', 'September 23, 2021',
                ..., 'December 6, 2018', 'March 9, 2016', 'January 11, 2020'],
               dtype=object)
```

- If we want find unique values in target feature then we can use above operation.

---

3. To find Data Type of Each column-

- We can find datatype of each columns ,
- Syntax-

df.dtypes

```
In [17]: df.dtypes

Out[17]: show_id         object
         type            object
         title           object
         director        object
         cast            object
         country         object
         date_added      object
```

- In pandas string is denoted as Object as per above output.

4.Change Data type of Existing Column-

- Syntax-

df ['<Column_Name>'] = df ['<Column_Name>'].astype('<Data_Type')

```
In [20]: df['release_year']=df['release_year'].astype('object')
```

```
In [21]: df.dtypes
```

```
Out[21]: show_id         object
         type            object
         title           object
         director        object
         cast            object
         country         object
         date_added      object
         release_year    object
         rating          object
```

- We can change existing column data type as above.

5.Delete Rows which have null Values-

- Syntax-

df.dropna(inplace=True)

```
In [26]: df.dropna(inplace=True)
```

```
In [30]: df.shape
```

```
Out[30]: (5332, 12)
```

- All rows are deleted which have null values from Data Frame.

6.Drop Null values from Rows or Columns with conditions-

- Syntax-

df.dropna(axis=0/1,how=any/all,subset[' Col_1' , 'Col_2'])

```
In [44]: df1.dropna(axis=0,how='any',subset=['EDOJ'])
```

Out[44]:

	Emp Id	Emp Name	Emp Salary	Emp Department	EDOJ	Eloc	Age	Gender
0	1	Jon	85000	HR	12-12-2008	Pune,MH	35 year	Male
1	2	Null	95200	Finance	11-08-2011	Nashik,MH	25 Year	Male
2	3	David	98563	NaN	08-05-2020	Varanasi,UP	NaN	Male
3	4	Laila	45000	Testing	27-09-2015	Delhi,Delhi	27 Year	Female
5	6	Priety	45979	Development	29-09-2015	Kolakata,WB	31 Year	Female

- 4th no record which contains null value has dropped from Data Frame.

7.Replace None Values from Dataframe-

- We can replace none values with user defined values from DataFrame-

df.replace(np.nan,<User_Defined_Value>)

df.replace(np.nan,<User_Defined_Value>)

```
In [51]: import numpy as np
```

```
In [53]: df1.replace(np.nan,'Python')
```

Out[53]:

	Emp Id	Emp Name	Emp Salary	Emp Department	EDOJ	Eloc	Age	Gender
0	1	Jon	85000	HR	12-12-2008	Pune,MH	35 year	Male
1	2	Null	95200	Finance	11-08-2011	Nashik,MH	25 Year	Male
2	3	David	98563	Python	08-05-2020	Varanasi,UP	Python	Male
3	4	Laila	45000	Testing	27-09-2015	Delhi,Delhi	27 Year	Female
4	5	Jason	69879	Development	Python	Python	29 Year	Male

- Will replace all NaN values with user defined value.

8.We can use Replace encoding of Column-

- Syntax-

df1.replace(['<Col_Value>' , '<Col_Value>'], ['<User_Defined_Value>', '<User_Defined_Value>'])

```
In [71]: df1.replace(['Male','Female'],[1,2])
```

Out[71]:

	Emp Id	Emp Name	Emp Salary	Emp Department	EDOJ	Eloc	Age	Gender
0	1	Jon	85000	HR	12-12-2008	Pune,MH	35 year	1
1	2	Null	95200	Finance	11-08-2011	Nashik,MH	25 Year	1
2	3	David	98563	Python	08-05-2020	Varanasi,UP	Python	1
3	4	Laila	45000	Testing	27-09-2015	Delhi,Delhi	27 Year	2
4	5	Jason	69879	Development	Python	Python	29 Year	1
5	6	Prity	45979	Development	29-09-2015	Kolakata,WB	31 Year	2

- We can also use replace for Encoding Of columns-

9.Use Group By clause-

- syntax-

df.groupby('<Column_Name>')

```
In [79]: df1.groupby('Emp Department')
```

```
Out[79]: <pandas.core.groupby.generic.DataFrameGroupBy object at 0x000002728F8DA280>
```

- If you use above command will give you a Object,so we have to store object in variable.

```
In [80]: obj=df1.groupby('Emp Department')
```

- Now we have to fetch group from 'Emp Department',

Syntax= obj.get_group('<Department_Name>')

```
In [82]: obj.get_group('Development')
```

Out[82]:

Emp Id	Emp Name	Emp Salary	Emp Department	EDOJ	Eloc	Age	Gender	Gender Encode

4	5	Jason	69879	Development	Python	Python	29 Year	1	None
5	6	Priety	45979	Development	29-09-2015	Kolakata,WB	31 Year	2	None

- We can also find each department's size,

Syntax-

obj.size( )

```
In [83]: obj.size()
```

```
Out[83]: Emp Department
         Development    2
         Finance        1
         HR             1
         Python         1
         Testing        2
         dtype: int64
```

10.Rename Existing column-

- Syntax-

df.rename({ 'Old_Col_Name':'New_Col_Name' },axis=1,inplace=True)

```
In [35]: df.rename({'release_year':'Release_Year','date_added':'Date_Added'},axis=1,inplace=True)
```

```
In [36]: df
```

Out[36]:

	show_id	type	title	director	cast	country	Date_Added	Release_Year	rating	duration
7	s8	Movie	Sankofa	Haile Gerima	Kofi Ghanaba, Oyafunmike Ogunlano, Alexandra D...	United States, Ghana, Burkina Faso, United Kin...	September 24, 2021	1993	TV-MA	125 min
					Mel Giedrovc.					

- We can also change multiple column names at once with passing in dictionary.
- We have also another method to rename columns ,syntax is-

df.columns=['New_Col1_Name','New_Col2_Name','New_Col3_Name']

```
In [53]: df7.columns=['EMP_Name','EMP_Age','EMP_City']
```

```
In [54]: df7
```

Out[54]:

	EMP_Name	EMP_Age	EMP_City
0	Jon	22	Pune
1	Kevin	25	USA
2	Abhi	32	Ahmednagar

11.Concatention Of Dataframe-

- We have 3 DataFrame and we have to concatenate these and make new DataFrame,

- Syntax-

df7=pd.concat([df4,df5,df6])

```
In [48]: df7=pd.concat([ df4,df5,df6])
```

```
In [49]: df7
```

Out[49]:

	Name	Age	City
0	Jon	22	Pune
1	Kevin	25	USA
2	Abhi	32	Ahmednagar
3	Rutul	23	Nashik
0	Jason	23	Ahmednagar
1	Nikhil	26	Pune

- We can concatenate multiple DataFrame at once.
- If columns are not same in these dataframe it will show null values in that additional columns.
- We can ignore index,

```
In [50]: df7=pd.concat([ df4,df5,df6],ignore_index=True)
```

```
In [51]: df7
```

Out[51]:

	Name	Age	City
0	Jon	22	Pune
1	Kevin	25	USA
2	Abhi	32	Ahmednagar
3	Rutul	23	Nashik
4	Jason	23	Ahmednagar
5	Nikhil	26	Pune

- With the help of Concatenation method we can add column and values to Dataframe.

```
In [56]: New_Col=pd.Series([100,200,300])
```

```
In [60]: df7=pd.concat([df7,New_Col],axis=1)
```

```
In [61]: df7
```

Out[61]:

	EMP_Name	EMP_Age	EMP_City	0
0	Jon	22	Pune	100.0
1	Kevin	25	USA	200.0
2	Abhi	32	Ahmednagar	300.0
3	Rutul	23	Nashik	NaN

- [ ] **Aggregate Functions-**

- We can perform all aggregate functions like Min,Max,Mean,Standard Deviation etc.

1.MAX function-

- Finding Maximum temperature from Max_Temp column,

Syntax-

df ['Max_Temp']== df ['Max_Temp'].max( )

```
In [12]: df['MaxTemp']==df['MaxTemp'].max()
```

```
Out[12]: 0      False
         1      False
         2      False
         3      False
         4      False
                ...
         361    False
```

- We got values in True and False ,if we we have to display in DataFrame format then add auery in "df[ ]".

```
In [13]: df[df['MaxTemp']==df['MaxTemp'].max()]
```

Out[13]:

	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir	WindGustSpeed	WindDir9am	WindDir3pm	WindS
71	20.9	35.8	0.0	9.4	13.3	SSE	57.0	NNW	NW	

1 rows × 22 columns

- If we want only Evaporation at maximum temperature from MaxTemp column.

```
In [32]: df[df['MaxTemp']==df['MaxTemp'].max()]
```

Out[32]:

	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir	WindGustSpeed	WindDir9am	W
71	20.9	35.8	0.0	9.4	13.3	SSE	57.0	NNW	

1 rows × 22 columns

```
In [33]: df['Evaporation'][df['MaxTemp']==df['MaxTemp'].max()]
```

```
Out[33]: 71    9.4
         Name: Evaporation, dtype: float64
```

2.MIN function-

- Finding Minimum temperature from Min_Temp column,

Syntax-

df ['Min_Temp']== df ['Min_Temp'].min( )

```
In [15]: df[df['MinTemp']==df['MinTemp'].min()]
```

Out[15]:

	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir	WindGustSpeed	WindDir9am	WindDi
0	20.9	24.3	0.0	3.4	6.3	NW	30.0	SW	
1	20.9	26.9	3.6	4.4	9.7	ENE	39.0	E	
2	20.9	23.4	3.6	5.8	3.3	NW	85.0	N	
3	20.9	15.5	39.8	7.2	9.1	NW	54.0	WNW	

4	20.9	16.1	2.8	5.6	10.6	SSE	50.0	SSE

- Min temp is 20.9*c so we got all rows which have 20.9*c temp.

3.Mean Function()-

- We can perform mean operation on any columns from DataFrame.

```
In [17]: df['MinTemp'].mean()
Out[17]: 20.899999999999864
```

4.Standard Deviation of Columns-

- We can perform Standard Deviation operation on any columns from DataFrame.

```
In [19]: df['MaxTemp'].std()
Out[19]: 6.690515669598577
```

Comparison Operator-

- We can use comparison operator to fetch specific data from Dataframe.

Ex.1-Fetch specific row which have 15* C from MaxTemp column.

```
In [45]: df[df['MaxTemp']==15]
```

Out[45]:

	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir	WindGustSpeed	WindDir9am	WindDir3pm	W
223	20.9	15.0	4.8	0.2	0.5	NW	46.0	NNW	NW	
356	20.9	15.0	0.8	4.8	11.7	S	70.0	S	S	

Ex.2-Fetch rows from MaxTemp column which have temp above 35*C.

```
In [50]: df[df['MaxTemp']>35]
```

Out[50]:

	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir	WindGustSpeed	WindDir9am	WindDir3
71	20.9	35.8	0.0	9.4	13.3	SSE	57.0	NNW	
72	20.9	35.7	0.0	13.8	6.9	SW	50.0	E	W
135	20.9	35.2	0.0	6.4	11.2	SE	48.0	SE	E

3 rows × 22 columns

Ex.3-Fetch rows from MaxTemp column which have temp below 8*C.

```
In [55]: df[df['MaxTemp']<8]
```

Out[55]:

	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir	WindGustSpeed	WindDir9am	WindDir3pm	W
283	20.9	7.6	0.4	2.4	4.7	NW	50.0	NW	NW	

1 rows × 22 columns

Ex.4-We can give also one or more conditions to filter data using comparison operator,so we have to
fetch rows which have temp between range 30 to 31*C from MaxTemp column.

```
In [63]: df[(df['MaxTemp']>30) & (df['MaxTemp']<31)]
```

Out[63]:

	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir	WindGustSpeed	WindDir9am	WindDir3pm	Wi
13	20.9	30.9	0.0	6.2	12.4	NW	44.0	WNW	W	
68	20.9	30.3	0.0	10.0	8.1	E	46.0	E	N	
137	20.9	30.2	0.0	7.8	11.2	NE	33.0	ESE	NNW	
361	20.9	30.7	0.0	7.6	12.1	NNW	76.0	SSE	NW	
365	20.9	30.2	0.0	6.0	12.6	NW	78.0	NW	WNW	

5 rows × 22 columns

☐ Indexing-

- We can perform indexing operation on columns-

Ex.1-Set specific column at first index

```
In [67]: df.set_index('RainTomorrow',inplace=True)
```

```
In [68]: df
```

Out[68]:

RainTomorrow	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir	WindGustSpeed	WindDir9am	WindDir3pm	\
Yes	20.9	24.3	0.0	3.4	6.3	NW	30.0	SW	NW	
Yes	20.9	26.9	3.6	4.4	9.7	ENE	39.0	E	W	
Yes	20.9	23.4	3.6	5.8	3.3	NW	85.0	N	NNE	
Yes	20.9	15.5	39.8	7.2	9.1	NW	54.0	WNW	W	
No	20.9	16.1	2.8	5.6	10.6	SSE	50.0	SSE	ESE	

Ex.2-We can Reset/rollback index-

```
In [69]: df.reset_index(inplace=True)
```

```
In [70]: df
```

Out[70]:

	RainTomorrow	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir	WindGustSpeed	WindDir9am	WindDir3pm	..
0	Yes	20.9	24.3	0.0	3.4	6.3	NW	30.0	SW	NW	..
1	Yes	20.9	26.9	3.6	4.4	9.7	ENE	39.0	E	W	..
2	Yes	20.9	23.4	3.6	5.8	3.3	NW	85.0	N	NNE	..
3	Yes	20.9	15.5	39.8	7.2	9.1	NW	54.0	WNW	W	..
4	No	20.9	16.1	2.8	5.6	10.6	SSE	50.0	SSE	ESE	..

☐ Loc and iLoc function-

- The Loc and iLoc functions in Pandas are used to slice a dataset and filter row wise data.

Ex.1-We can fetch only RainTomorrow= ''Yes" from table.

```
In [93]: df.loc['Yes']
```

Out[93]:

RainTomorrow	level_0	index	WindGustDir	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustSpeed	WindDir9am	...
Yes	0	0	NW	20.9	24.3	0.0	3.4	6.3	30.0	SW	...
Yes	1	1	ENE	20.9	26.9	3.6	4.4	9.7	39.0	E	...
Yes	2	2	NW	20.9	23.4	3.6	5.8	3.3	85.0	N	...
Yes	3	3	NW	20.9	15.5	39.8	7.2	9.1	54.0	WNW	...
Yes	8	8	S	20.9	19.5	0.0	4.0	4.1	48.0	E	...

Ex.2-We can slice data as we want,in below case i have to fetch only 3 rows and column from Emp Name to Eloc.

```
In [102]: df5.loc[0:2,'Emp Name':'Eloc']
```

Out[102]:

	Emp Name	Emp Salary	Emp Department	EDOJ	Eloc
0	Jon	85000	HR	12-12-2008	Pune,MH
1	Null	95200	Finance	11-08-2011	Nashik,MH
2	David	98563	NaN	08-05-2020	Varanasi,UP

- in Loc argument firstly we have to specify and rows and then *Column name* as argument.
- In Loc function () range that we have given is included type.i.e upper limit is included.

Ex.3-We have to fetch 0 to 2 index rows and Emp id to EDOJ columns.

```
In [104]: df5.iloc[0:3,0:5]
```

Out[104]:

	Emp Id	Emp Name	Emp Salary	Emp Department	EDOJ
0	1	Jon	85000	HR	12-12-2008
1	2	Null	95200	Finance	11-08-2011
2	3	David	98563	NaN	08-05-2020

- In iLoc we have to give column index for fetch and upper limit is excluded in iLoc function.

```
In [105]: df5.loc[0:3]
```

Out[105]:

	Emp Id	Emp Name	Emp Salary	Emp Department	EDOJ	Eloc	Age	Gender
0	1	Jon	85000	HR	12-12-2008	Pune,MH	35 year	Male
1	2	Null	95200	Finance	11-08-2011	Nashik,MH	25 Year	Male
2	3	David	98563	NaN	08-05-2020	Varanasi,UP	NaN	Male
3	4	Laila	45000	Testing	27-09-2015	Delhi,Delhi	27 Year	Female

```
In [106]: df5.iloc[0:3]
```

Out[106]:

	Emp Id	Emp Name	Emp Salary	Emp Department	EDOJ	Eloc	Age	Gender
0	1	Jon	85000	HR	12-12-2008	Pune,MH	35 year	Male
1	2	Null	95200	Finance	11-08-2011	Nashik,MH	25 Year	Male
2	3	David	98563	NaN	08-05-2020	Varanasi,UP	NaN	Male

- We can find difference between Loc and iLoc from above example.

Data Normalization-

- Normalization is a pre-processing stage of any type of problem statement. In particular, normalization takes an important role in the field of soft computing, cloud computing, etc. for manipulation of data, scaling down, or scaling up the range of data before it becomes used for further stages. There are so many normalization techniques there, namely Min-Max normalization, Z-score normalization, and Decimal scaling normalization.
- *The goal of Data normalization is to transform features to be on a similar scale. This improves the performance and training stability of the model.*
- Data transformation operations, such as normalization and aggregation, are additional data preprocessing procedures that would contribute toward the success of the data extract process.
- *Data normalization consists of remodeling numeric columns to a standard scale.* Data normalization is generally considered the development of clean data. Diving deeper, however, the meaning or goal of data normalization is twofold:
  - Data normalization is the organization of data to appear similar across all records and fields.
  - It increases the cohesion of entry types, leading to cleansing, lead generation, segmentation, and higher quality data.

1.Importance Of Data Normalization-

- *Data Normalization disposes of various anomalies that can make an examination of the information more complicated.* A portion of those irregularities can manifest from erasing information, embedding more data, or refreshing existing data. Once those mistakes are worked out and eliminated from the framework, further advantages can be acquired through different jobs in the data and data examination.
- It is for the most part through data normalization that the data inside a data set can be designed so that it can be visualized and examined.

2.Data Normalization Advantages-

- We can have more clustered indexes.
- Index searching is often faster.
- Data modification commands are faster.
- Fewer null values and less redundant data, making your data more compact.
- Data modification anomalies are reduced.
- Normalization is conceptually cleaner and easier to maintain and change as your needs change.
- Searching, sorting, and creating indexes is faster, since tables are narrower, and more rows fit on a data page.

3.Normalization Techniques at a Glance

Four common normalization techniques may be useful:

1. scaling to a range
2. clipping
3. log scaling
4. z-score

- The following charts show the effect of each normalization technique on the distribution of the raw feature (price) on the left. The charts are based on the data set from 1985 Ward's Automotive Yearbook that is part of the UCI Machine Learning Repository under Automobile

Data Set.

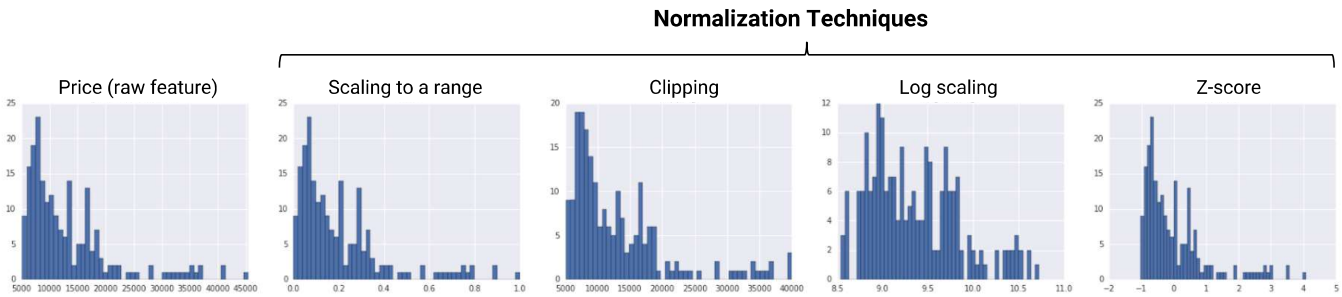


Figure - Summary of normalization techniques.

1.Scaling to a range/Min-Max Normalization

- **Scaling** means converting values of features from their natural range into floating-point feature values (for example, 100 to 900) into a standard range—usually 0 and 1 (or sometimes -1 to +1). Use the following simple formula to scale to a range:

$$x' = (x - xmin)/(xmax - xmin)$$

Scaling to a range is a good choice when both of the following conditions are met:

1. You know the approximate upper and lower bounds on your data with few or no outliers.
2. Your data is approximately uniformly distributed across that range.

- A good example is age. Most age values falls between 0 and 90, and every part of the range has a substantial number of people.
- *In contrast, you would not use scaling on income, because only a few people have very high incomes.* The upper bound of the linear scale for income would be very high, and most people would be squeezed into a small part of the scale.
- Example-

```
In [31]: df1=df[['ApplicantIncome','CoapplicantIncome','LoanAmount','Loan_Amount_Term']]
```

```
In [32]: df1
```

Out[32]:

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term
0	4950	0.0	125	360
1	2882	1843.0	123	480
2	3000	3416.0	56	180

- Import class MinMaxscaler from sklearn library.

```
In [37]: from sklearn.preprocessing import MinMaxScaler
         import pandas as pd
```

```
In [38]: obj=MinMaxScaler()
```

```
In [ ]: df1[['ApplicantIncome','CoapplicantIncome','LoanAmount','Loan_Amount_Term']]=obj.fit_transform
        (df1[['ApplicantIncome','CoapplicantIncome','LoanAmount','Loan_Amount_Term']])
```

```
In [35]: df1
```

Out[35]:

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term
0	0.059369	0.000000	0.196277	0.729730
1	0.033791	0.054467	0.192893	1.000000
2	0.035250	0.100955	0.079526	0.324324

2.Feature Clipping

- *If your data set contains extreme outliers, you might try feature clipping, which caps all feature values above (or below) a certain value to fixed value.* For example, you could clip all temperature values above 40 to be exactly 40.
- You may apply feature clipping before or after other normalizations.

Formula: Set min/max values to avoid outliers.

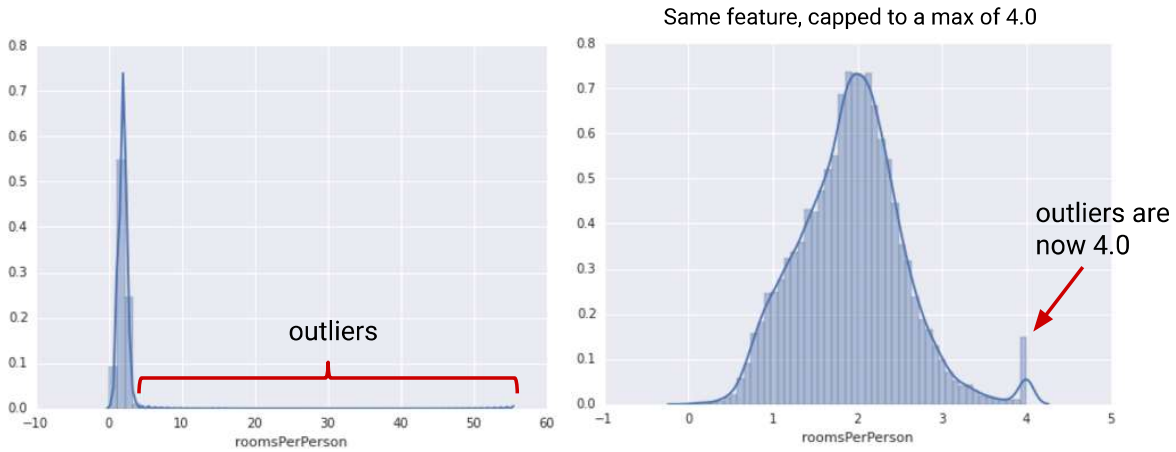


Figure- Comparing a raw distribution and its clipped version.

- Another simple clipping strategy is to clip by z-score to +-Nσ (for example, limit to +-3σ). Note that σ is the standard deviation.

3.Log Scaling

- Log scaling computes the log of your values to compress a wide range to a narrow range.

$$x' = \log(x)$$

- *Log scaling is helpful when a handful of your values have many points, while most other values have few points. This data distribution is known as the* ***power law distribution.*** *Movie ratings are a good example.* In the chart below, most movies have very few ratings (the data in the tail), while a few have lots of ratings (the data in the head). Log scaling changes the distribution, helping to improve linear model performance.

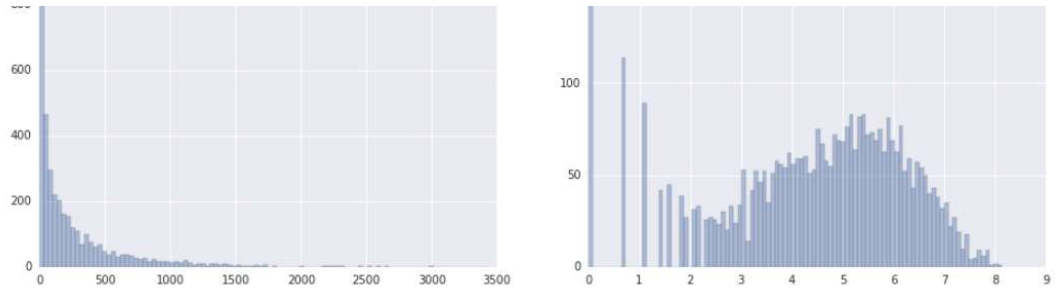


Figure- Comparing a raw distribution to its log.

4.Z-Score

- *Z-score is a variation of scaling that represents the number of standard deviations away from the mean.* You would use z-score to ensure your feature distributions have mean = 0 and std = 1. *It's useful when there are a few outliers, but not so extreme that you need clipping.*
- The formula for calculating the z-score of a point, *x*, is as follows:

$$x' = (x - \mu)/\sigma$$

- **Note:** μ is the mean and σ is the standard deviation.

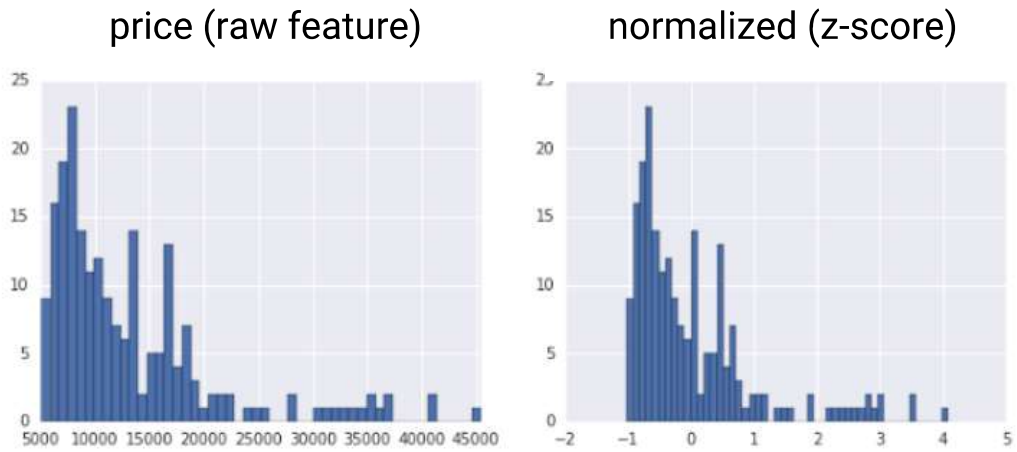


Figure- Comparing a raw distribution to its z-score distribution.

- Notice that z-score squeezes raw values that have a range of ~40000 down into a range from roughly -1 to +4.
- Suppose you're not sure whether the outliers truly are extreme. In this case, start with z-score unless you have feature values that you don't want the model to learn; for example, the values are the result of measurement error or a quirk.
- Example-

```
In [121]: df
```

Out[121]:

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area
0	-0.037697	-0.599846	-0.219389	0.272293	0.408868	Urban
1	-0.428032	0.103374	-0.246028	2.097087	0.408868	Semiurban
2	-0.405760	0.703571	-1.138429	-2.464897	0.408868	Semiurban

3	0.859432	-0.599846	-0.392541	0.272293	0.408868	Urban
4	-0.531656	0.322390	-0.072875	0.272293	0.408868	Urban

- import scale class from sklearn

```
In [122]: from sklearn.preprocessing import scale
```

```
In [123]: df[col]=scale(df[col])
```

```
In [124]: df
```

Out[124]:

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area
0	-0.037697	-0.599846	-0.219389	0.272293	0.408868	Urban
1	-0.428032	0.103374	-0.246028	2.097087	0.408868	Semiurban
2	-0.405760	0.703571	-1.138429	-2.464897	0.408868	Semiurban
3	0.859432	-0.599846	-0.392541	0.272293	0.408868	Urban
4	-0.531656	0.322390	-0.072875	0.272293	0.408868	Urban

Summary -

Normalization Technique	Formula	When to Use
Linear Scaling	$x' = (x - x_{min})/(x_{max} - x_{min})$	When the feature is more-or-less uniformly distributed across a fixed range.
Clipping	if x > max, then x' = max. if x < min, then x' = min	When the feature contains some extreme outliers.
Log Scaling	x' = log(x)	When the feature conforms to the power law.
Z-score	x' = (x - μ) / σ	When the feature distribution does not contain extreme outliers.

Merge Of Columns-

- Merging two columns in Pandas can be a tedious task if you don't know the Pandas merging concept. You can easily merge two different data frames easily. But on two or more columns on the same data frame is of a different concept

Ex.1-Inner join

```
In [30]: pd.merge(P_Details,J_Details,on='Emp_Id')
```

Out[30]:

	Name	Age	City	Emp_Id	Dept	Salary	Client
0	Jon	22	Pune	1	HR	58697	HDFC
1	Kevin	25	USA	2	Dev	98000	SBI
2	Abhi	32	Ahmednagar	3	Testing	69876	CITI
3	Rutul	23	Nashik	4	ACC	79868	ICICI

- By-default merge function joins two tables with Inner Join.

Ex.2-Right merge with Matching records-

```
In [36]: pd.merge(P_Details,J_Details,on='Emp_Id',how='right')
```

Out[36]:

	Name	Age	City	Emp_Id	UID	Dept	Salary	Client
0	Jon	22	Pune	1	595489	HR	58697	HDFC
1	Kevin	25	USA	2	154415	Dev	98000	SBI
2	Abhi	32	Ahmednagar	3	498494	Testing	69876	CITI
3	Rutul	23	Nashik	4	489491	ACC	79868	ICICI

Ex.3-Full outer Join-

```
In [44]: pd.merge(P_Details,J_Details,on='Emp_Id',how='outer')
```

Out[44]:

	Name	Age	City	Emp_Id	UID	Dept	Salary	Client
0	Jon	22	Pune	1	595489	HR	58697.0	HDFC
1	Kevin	25	USA	2	154415	Dev	98000.0	SBI
2	Abhi	32	Ahmednagar	3	498494	Testing	69876.0	CITI
3	Rutul	23	Nashik	4	489491	ACC	79868.0	ICICI
4	Neha	26	US	5	48948	NaN	NaN	NaN

- We got all records from both of DataFrame and we can also specify from which DataFrame data has reflected.

```
In [45]: pd.merge(P_Details,J_Details,on='Emp_Id',how='outer',indicator=True)
```

Out[45]:

	Name	Age	City	Emp_Id	UID	Dept	Salary	Client	_merge
0	Jon	22	Pune	1	595489	HR	58697.0	HDFC	both
1	Kevin	25	USA	2	154415	Dev	98000.0	SBI	both
2	Abhi	32	Ahmednagar	3	498494	Testing	69876.0	CITI	both
3	Rutul	23	Nashik	4	489491	ACC	79868.0	ICICI	both
4	Neha	26	US	5	48948	NaN	NaN	NaN	left_only

Seaborn-

- In the world of Analytics, the best way to get insights is by visualizing the data. Data can be visualized by representing it as plots which is easy to understand, explore and grasp. Such data helps in drawing the attention of key elements.
- To analyse a set of data using Python, we make use of Matplotlib, a widely implemented 2D plotting library. Likewise, Seaborn is a visualization library in Python. It is built on top of Matplotlib.

Important Features Of Seaborn-

- Seaborn is built on top of Python's core visualization library Matplotlib. It is meant to serve as a complement, and not a replacement. However, Seaborn comes with some very important features. Let us see a few of them here. The features help in

features. Let us see a few of them here. The features help in –

1. Built in themes for styling matplotlib graphics.
2. Visualizing univariate and bivariate data.
3. Fitting in and visualizing linear regression models.
4. Plotting statistical time series data.
5. Seaborn works well with NumPy and Pandas data structures.
6. It comes with built in themes for styling Matplotlib graphics.

- In most cases, you will still use Matplotlib for simple plotting. The knowledge of Matplotlib is recommended to tweak Seaborn's default plots.

☐ **Importing Datasets-**

- Seaborn comes with a few important datasets in the library. When Seaborn is installed, the datasets download automatically.
- You can use any of these datasets for your learning. With the help of the following function you can load the required dataset.

sns.load_dataset( )

- To view all the available data sets in the Seaborn library, you can use the following command with the get_dataset_names() function as shown below –

sns.get_dataset_names ( )

☐ **Plotting Univariate Distribution-**

- Distribution of data is the foremost thing that we need to understand while analysing the data. Here, we will see how seaborn helps us in understanding the univariate distribution of the data.
- Function distplot() provides the most convenient way to take a quick look at univariate distribution. This function will plot a histogram that fits the *kernel density estimation* of the data.
- Syntax- seaborn.distplot()
- Parameters-

Sr.No.	Parameter & Description
1	**data** Series, 1d array or a list
2	**bins** Specification of hist bins
3	**hist** bool
4	**kde** bool

- Ex-

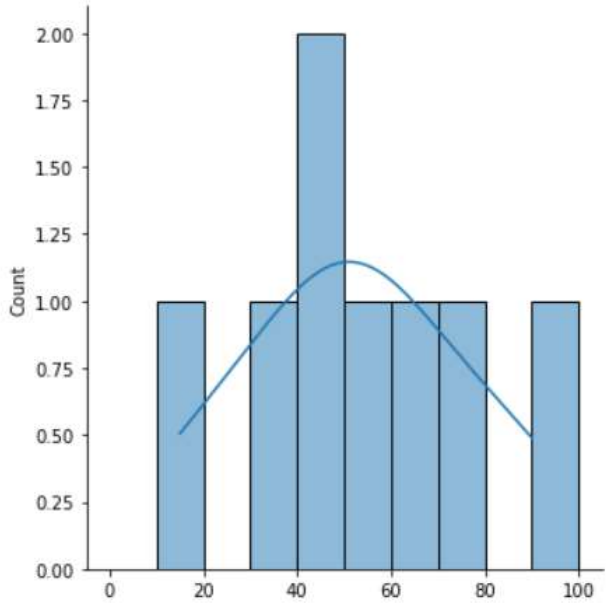
Ex-

```
In [43]: df2=[90,75,60,45,30,15,45.96,56]
```

```
In [44]: bins=[]
         for i in range(0,101,10):
             bins.append(i)
```

```
In [45]: sns.displot(df2,bins=bins,kde=True)
```

```
Out[45]: <seaborn.axisgrid.FacetGrid at 0x1fd15484310>
```



1.Histogram-

- Histograms represent the data distribution by forming bins along the range of the data and then drawing bars to show the number of observations that fall in each bin.
- Seaborn comes with some datasets and we have used few datasets in our previous chapters. We have learnt how to load the dataset and how to lookup the list of available datasets.
- Ex-

```
In [50]: import seaborn as sns
```

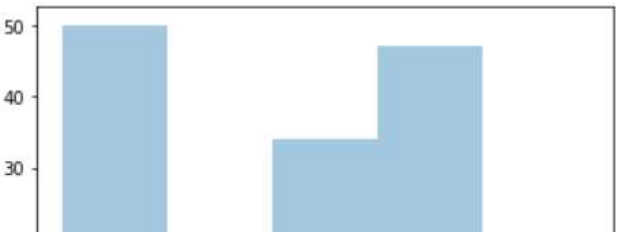
```
In [51]: from matplotlib import pyplot as plt
```

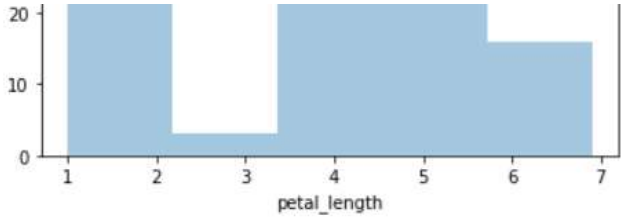
```
In [52]: df=sns.load_dataset('iris')
```

```
In [56]: sns.distplot(df['petal_length'],kde=False)
```

```
C:\Users\Abhimanyu Devadhe\anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureW
function and will be removed in a future version. Please adapt your code to use either `displ
similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

```
Out[56]: <AxesSubplot:xlabel='petal_length'>
```





- Here, kde flag is set to False. As a result, the representation of the kernel estimation plot will be removed and only histogram is plotted.

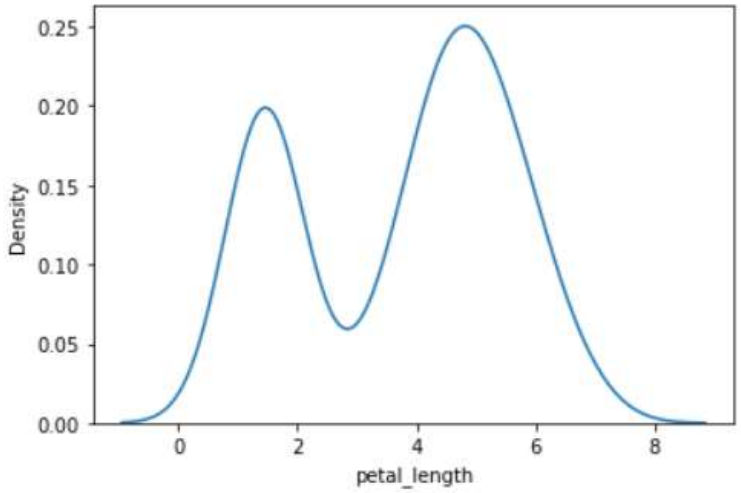
2.Kernel Density Estimates-

- Kernel Density Estimation (KDE) is a way to estimate the probability density function of a continuous random variable. It is used for non-parametric analysis.
- Setting the hist flag to False in distplot will yield the kernel density estimation plot.

```
In [81]: sns.distplot(df['petal_length'],hist=False)
```

```
C:\Users\Abhimanyu Devadhe\anaconda3\lib\site-packages\seaborn\distributions.py:255
function and will be removed in a future version. Please adapt your code to use eit
similar flexibility) or `kdeplot` (an axes-level function for kernel density plots)
  warnings.warn(msg, FutureWarning)
```

```
Out[81]: <AxesSubplot:xlabel='petal_length', ylabel='Density'>
```

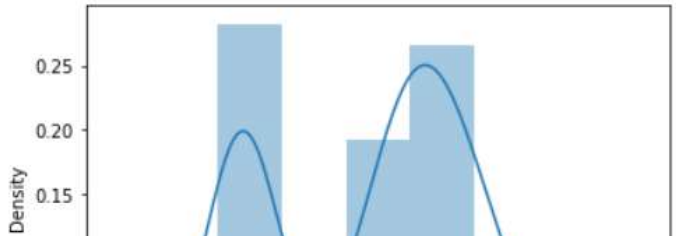


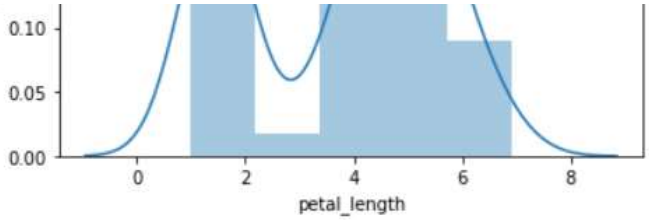
3.Fitting Parametric Distribution-

- distplot() is used to visualize the parametric distribution of a dataset.

```
In [82]: sns.distplot(df['petal_length'])
         plt.show()
```

```
C:\Users\Abhimanyu Devadhe\anaconda3\lib\site-packages\seaborn\distributions.py:2557: Futur
function and will be removed in a future version. Please adapt your code to use either `dis
similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```





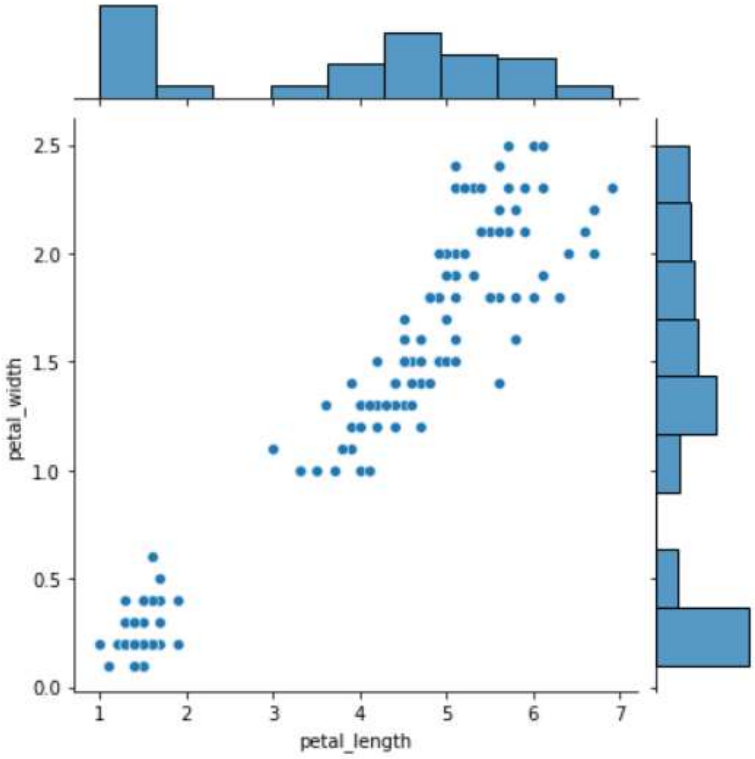
- [ ] **Plotting Bivariate Distribution-**

- Bivariate Distribution is used to determine the relation between two variables. This mainly deals with relationship between two variables and how one variable is behaving with respect to the other.
- The best way to analyze Bivariate Distribution in seaborn is by using the jointplot() function.
- Jointplot creates a multi-panel figure that projects the bivariate relationship between two variables and also the univariate distribution of each variable on separate axes.

1.Scatter Plot-

- Scatter plot is the most convenient way to visualize the distribution where each observation is represented in two-dimensional plot via x and y axis.

```
In [85]: sns.jointplot(x='petal_length',y='petal_width',data=df)
Out[85]: <seaborn.axisgrid.JointGrid at 0x1fd169d76d0>
```



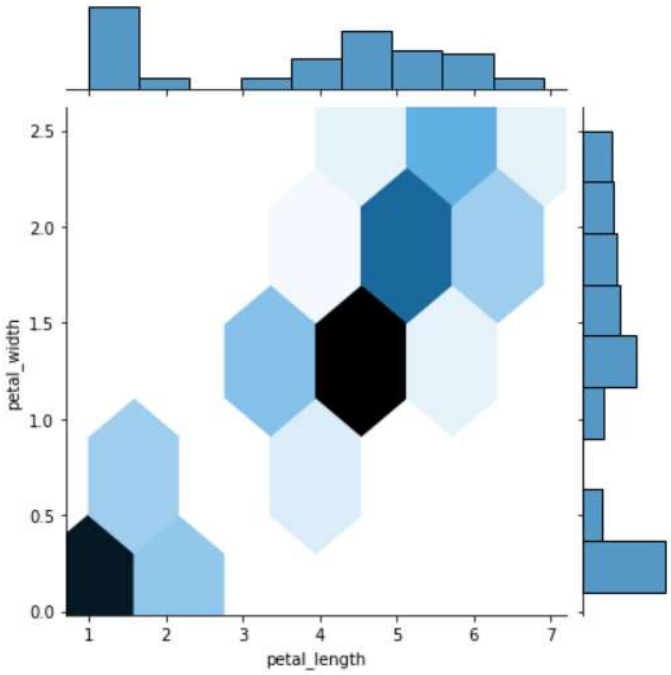
- The above figure shows the relationship between the petal_length and petal_width in the Iris data. A trend in the plot says that positive correlation exists between the variables under study.
- *Joint plot is used for checking Distribution.*

2.Hexbin Plot-

- The above figure shows the relationship between the petal_length and petal_width in the Iris data. A trend in the plot says that positive correlation exists between the variables under study.

- An addition parameter called ‘kind’ and value ‘hex’ plots the hexbin plot.

```
In [86]: sns.jointplot(x='petal_length',y='petal_width',data=df,kind='hex')
         plt.show()
```



- [ ] **Visualizing Pairwise Relationship-**

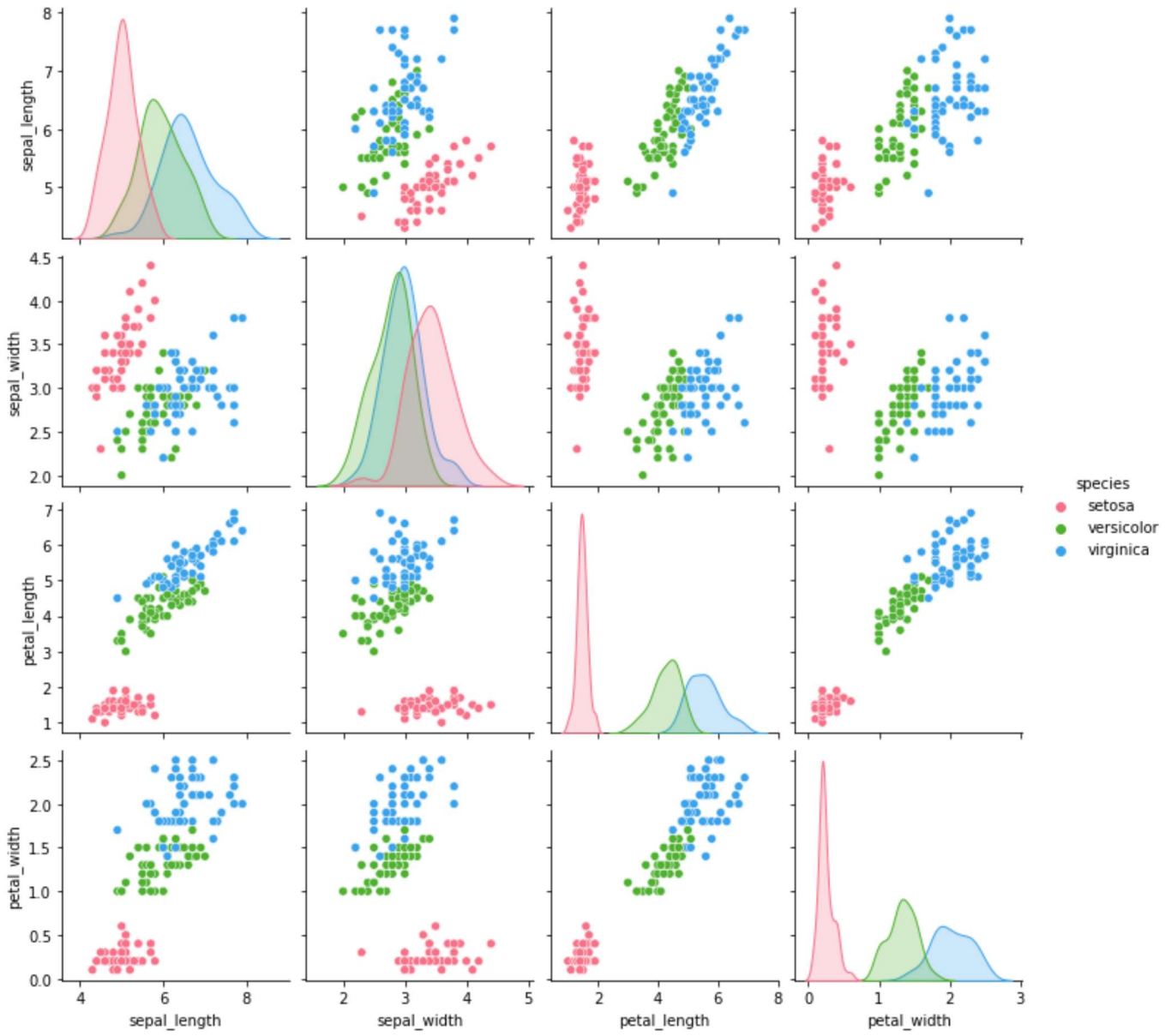
- Datasets under real-time study contain many variables. In such cases, the relation between each and every variable should be analyzed. Plotting Bivariate Distribution for (n,2) combinations will be a very complex and time taking process.
- To plot multiple pairwise bivariate distributions in a dataset, you can use the pairplot() function. This shows the relationship for (n,2) combination of variable in a DataFrame as a matrix of plots and the diagonal plots are the univariate plots.
- Parameters-

Sr.No.	Parameter & Description
1	**data** Dataframe
2	**hue** Variable in data to map plot aspects to different colors.
3	**palette** Set of colors for mapping the hue variable
4	**kind** Kind of plot for the non-identity relationships. {‘scatter’, ‘reg’}

5	**diag_kind** Kind of plot for the diagonal subplots. {'hist', 'kde'}

- Except data, all other parameters are optional. There are few other parameters which pairplot can accept. The above mentioned are often used params.
- Ex-

```
In [91]: sns.pairplot(df,hue='species',diag_kind="kde",kind='scatter',palette='husl')
         plt.show()
```



- We can observe the variations in each plot. The plots are in matrix format where the row name represents x axis and column name represents the y axis.

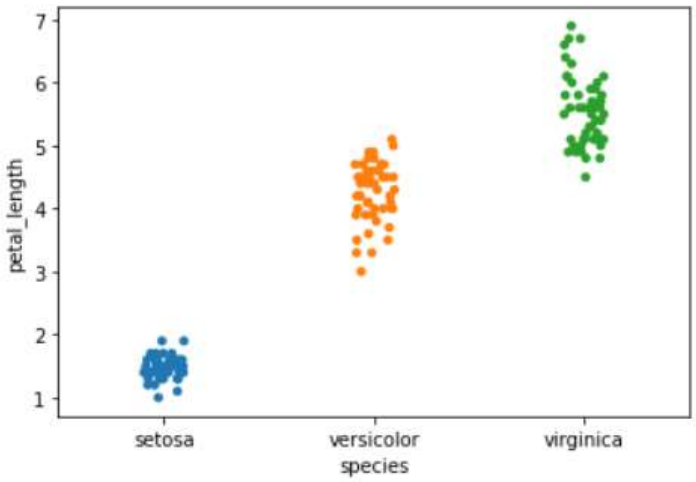
- [ ] **Plotting Categorical Data-**

- Hexabin plots and kde plots which are used to analyze the continuous variables under study. These plots are not suitable when the variable under study is categorical.
- When one or both the variables under study are categorical, we use plots like striplot(), swarmplot(), etc,. Seaborn provides interface to do so.

1.Stripplot( )-

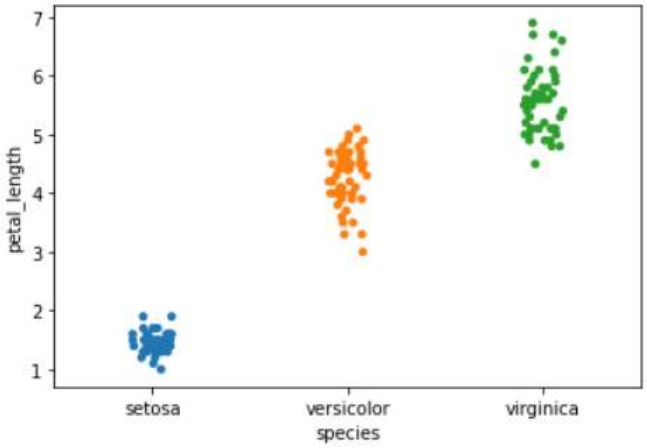
- stripplot() is used when one of the variable under study is categorical. It represents the data in sorted order along any one of the axis.

```
In [94]: sns.stripplot(x='species',y='petal_length',data=df)
         plt.show()
```



- In the above plot, we can clearly see the difference of petal_length in each species. But, the major problem with the above scatter plot is that the points on the scatter plot are overlapped. We use the 'Jitter' parameter to handle this kind of scenario.
- *Jitter adds some random noise to the data. This parameter will adjust the positions along the categorical axis.*

```
In [98]: sns.stripplot(x='species',y='petal_length',data=df,jitter=True)
         plt.show()
```



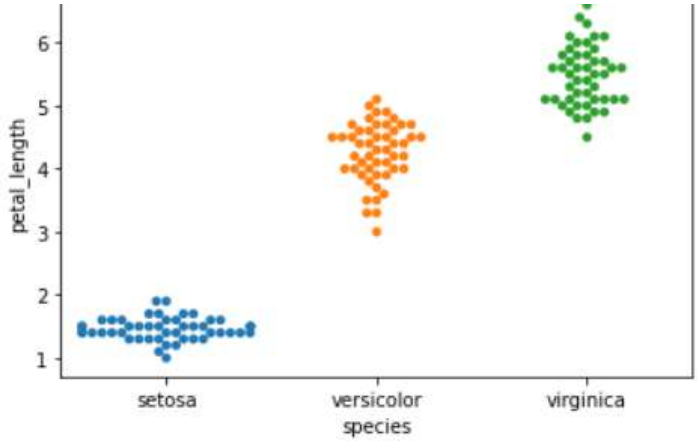
2.swarmplot( )-

- Another option which can be used as an alternate to 'Jitter' is function swarmplot( ). This function positions each point of scatter plot on the categorical axis and thereby avoids overlapping points –

```
In [100]: sns.swarmplot(x='species',y='petal_length',data=df)
```

```
C:\Users\Abhimanyu Devadhe\anaconda3\lib\site-packages\seaborn\categorical.py:1296:
placed; you may want to decrease the size of the markers or use stripplot.
  warnings.warn(msg, UserWarning)
```

```
Out[100]: <AxesSubplot:xlabel='species', ylabel='petal_length'>
```



Distribution Of Observations-

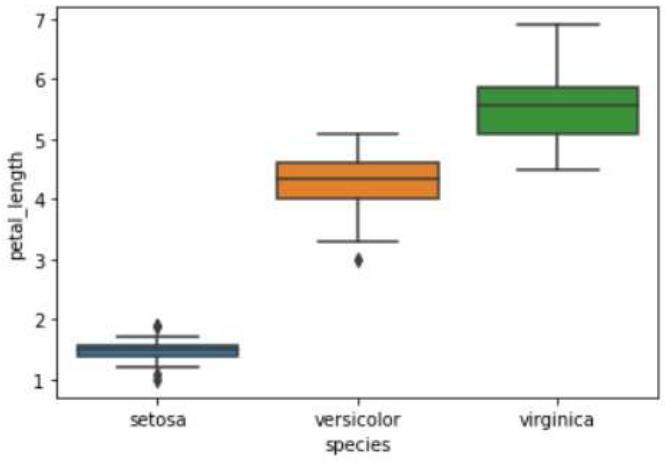
- In categorical scatter plots which we dealt in the previous chapter, the approach becomes limited in the information it can provide about the distribution of values within each category. Now, going further, let us see what can facilitate us with performing comparison with in categories.

1. Box Plots-

- Boxplot is a convenient way to visualize the distribution of data through their quartiles.
- *Box plot is mainly used for to identify outliers.*
- Box plots usually have vertical lines extending from the boxes which are termed as whiskers. These whiskers indicate variability outside the upper and lower quartiles, hence Box Plots are also termed as box-and-whisker plot and box-and-whisker diagram. Any Outliers in the data are plotted as individual points.

```
In [103]: sns.boxplot(x='species',y='petal_length',data=df)
```

```
Out[103]: <AxesSubplot:xlabel='species', ylabel='petal_length'>
```



- The dots on the plot indicates the outlier.

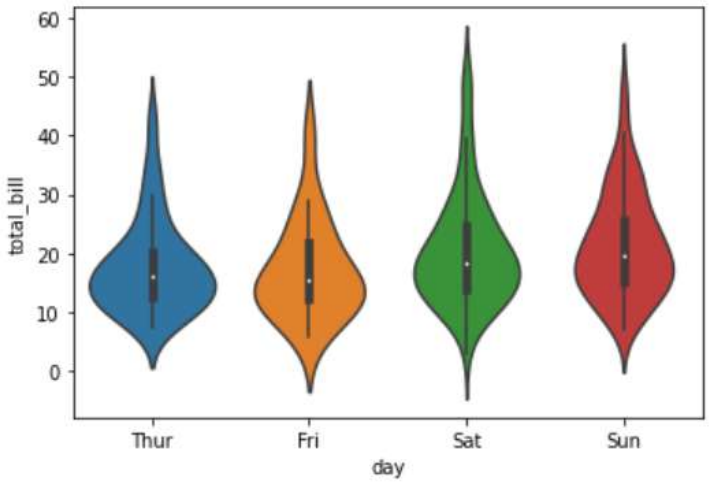
2.Violin Plots-

- Violin Plots are a *combination of the box plot with the kernel density estimates*. So, these plots are easier to analyze and understand the distribution of the data.
- Let us use tips dataset called to learn more into violin plots. This dataset contains the

information related to the tips given by the customers in a restaurant.

```
In [8]: sns.violinplot(x='day',y='total_bill',data=df)
```

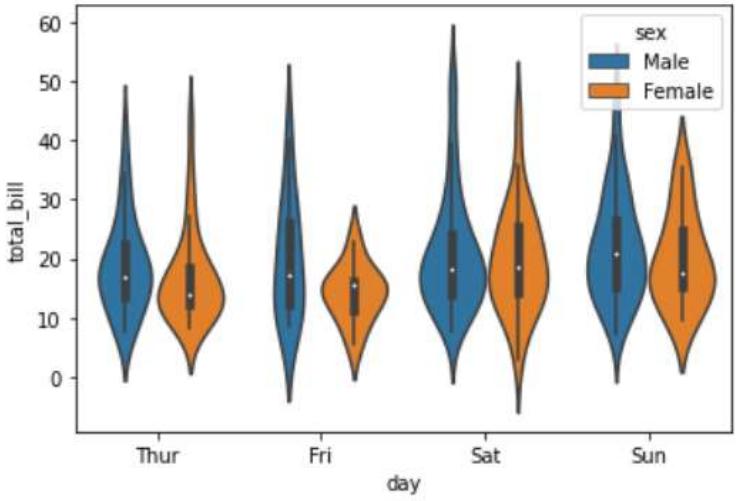
```
Out[8]: <AxesSubplot:xlabel='day', ylabel='total_bill'>
```



- The quartile and whisker values from the boxplot are shown inside the violin. As the violin plot uses KDE, the wider portion of violin indicates the higher density and narrow region represents relatively lower density. The Inter-Quartile range in boxplot and higher density portion in kde fall in the same region of each category of violin plot.
- The above plot shows the distribution of total_bill on four days of the week. But, in addition to that, *if we want to see how the distribution behaves with respect to sex, lets explore it in below example.*

```
In [10]: sns.violinplot(x='day',y='total_bill',hue='sex',data=df)
```

```
Out[10]: <AxesSubplot:xlabel='day', ylabel='total_bill'>
```



- Now we can clearly see the spending behavior between male and female. We can easily say that, men make more bill than women by looking at the plot.
- And, if the hue variable has only two classes, we can beautify the plot by splitting each violin into two instead of two violins on a given day. Either parts of the violin refer to each class in the hue variable.

- [ ] **Statistical Estimation-**

- In most of the situations, we deal with estimations of the whole distribution of the data. But

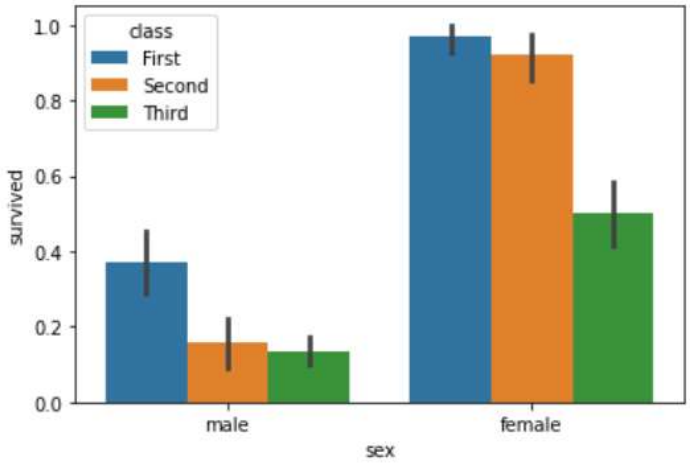
- In most of the situations, we deal with estimations of the whole distribution of the data. But when it comes to central tendency estimation, we need a specific way to summarize the distribution. *Mean and median are the very often used techniques to estimate the central tendency of the distribution.*
- In all the plots that we learnt in the above section, we made the visualization of the whole distribution. Now, let us discuss regarding the plots with which we can estimate the central tendency of the distribution.

1. Bar Plots-

- In all the plots that we learnt in the above section, we made the visualization of the whole distribution. Now, let us discuss regarding the plots with which we can estimate the central tendency of the distribution.
- Bar plot represents the estimate of central tendency. Let us use the 'titanic' dataset to learn bar plots.

```
In [13]: sns.barplot(x='sex',y='survived',hue='class',data=df3)
```

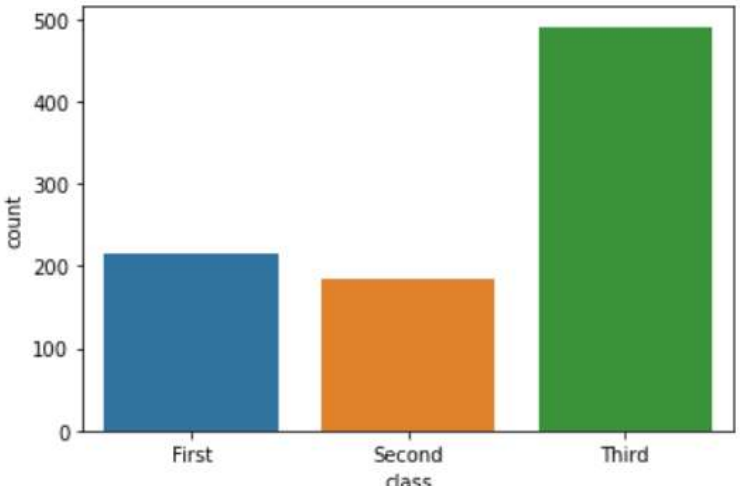
```
Out[13]: <AxesSubplot:xlabel='sex', ylabel='survived'>
```



- In the above example, we can see that the average number of survivals of male and female in each class. From the plot we can understand that more number of females survived than males. In both males and females more number of survivals are from first class.
- A special case in barplot is to show the no of *observations in each category rather than computing a statistic for a second variable. For this, we use countplot().*

```
In [14]: sns.countplot(x='class',data=df3)
```

```
Out[14]: <AxesSubplot:xlabel='class', ylabel='count'>
```



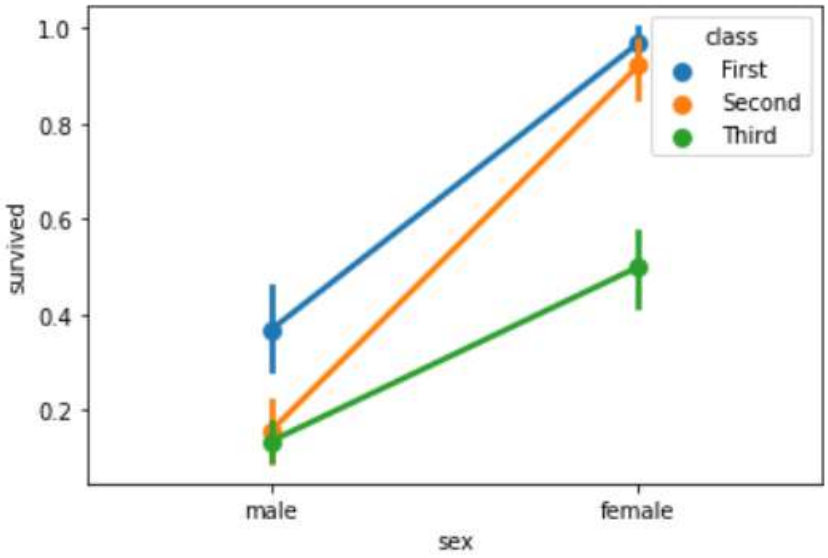
- Plot says that, the number of passengers in the third class are higher than first and second class.

2.Point Plots-

- Point plots serve same as bar plots but in a different style. Rather than the full bar, the value of the estimate is represented by the point at a certain height on the other axis.

```
In [16]: sns.pointplot(x='sex',y='survived',hue='class',data=df3)
```

```
Out[16]: <AxesSubplot:xlabel='sex', ylabel='survived'>
```

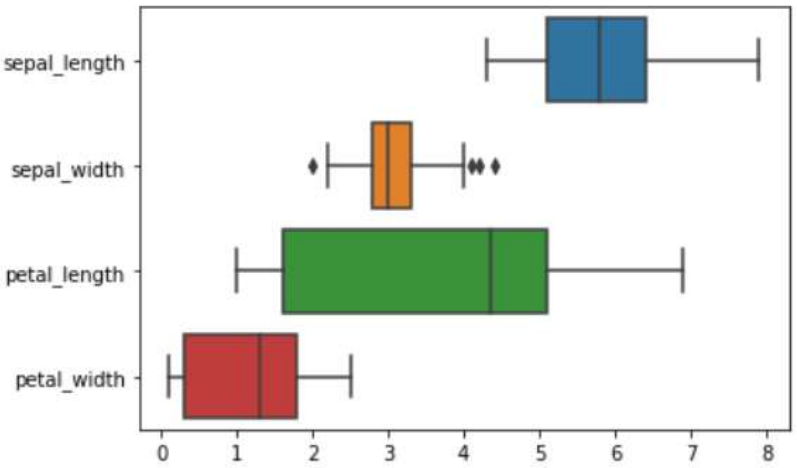


- [ ] **Plotting Wide Form Data-**

- It is always preferable to use ‘long-from’ or ‘tidy’ datasets. But at times when we are left with no option rather than to use a ‘wide-form’ dataset, same functions can also be applied to “wide-form” data in a variety of formats, including Pandas Data Frames or two-dimensional NumPy arrays. These objects should be passed directly to the data parameter the x and y variables must be specified as strings.

```
In [19]: sns.boxplot(data=df1,orient='h')
```

```
Out[19]: <AxesSubplot:>
```



Multi-Panel Categorical Data-

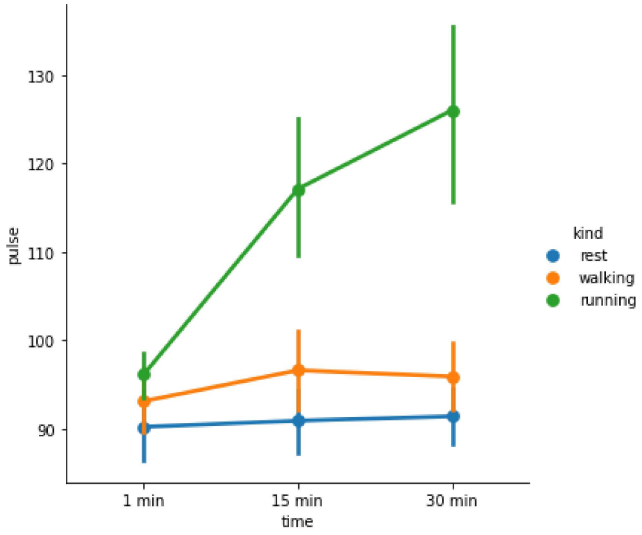
- Categorical data can we visualized using two plots, you can either use the functions pointplot(), or the higher-level function factorplot().

1.FactorPlot()-

- Factorplot draws a categorical plot on a FacetGrid. Using 'kind' parameter we can choose the plot like boxplot, violinplot, barplot and stripplot. FacetGrid uses pointplot by default.

```
In [23]: sns.factorplot(x='time',y='pulse',hue='kind',data=df4)
```

```
C:\Users\Abhimanyu Devadhe\anaconda3\lib\site-packages\seaborn\categorical.py:3714: UserWar
```



Linear Relationships-

- Most of the times, we use datasets that contain multiple quantitative variables, and the goal of an analysis is often to relate those variables to each other. This can be done through the regression lines.
- While building the regression models, we often check for multicollinearity, where we had to see the correlation between all the combinations of continuous variables and will take necessary action to remove multicollinearity if exists. In such cases, the following techniques helps.

Functions to Draw Linear Regression Models-

- There are two main functions in Seaborn to visualize a linear relationship determined through regression. These functions are regplot() and lmplot().

regplot vs lmplot

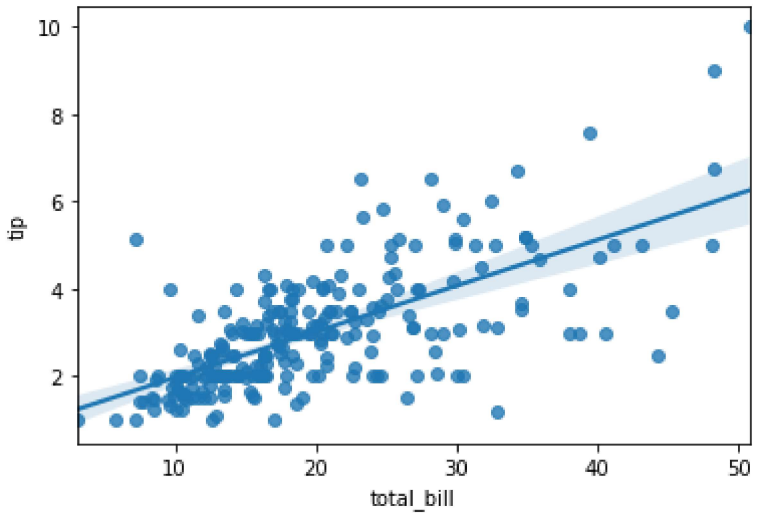
regplot	lmplot
accepts the x and y variables in a variety of formats including simple numpy arrays, pandas Series objects, or as references to variables in a pandas DataFrame	has data as a required parameter and the x and y variables must be specified as strings. This data format is called "long-form" data

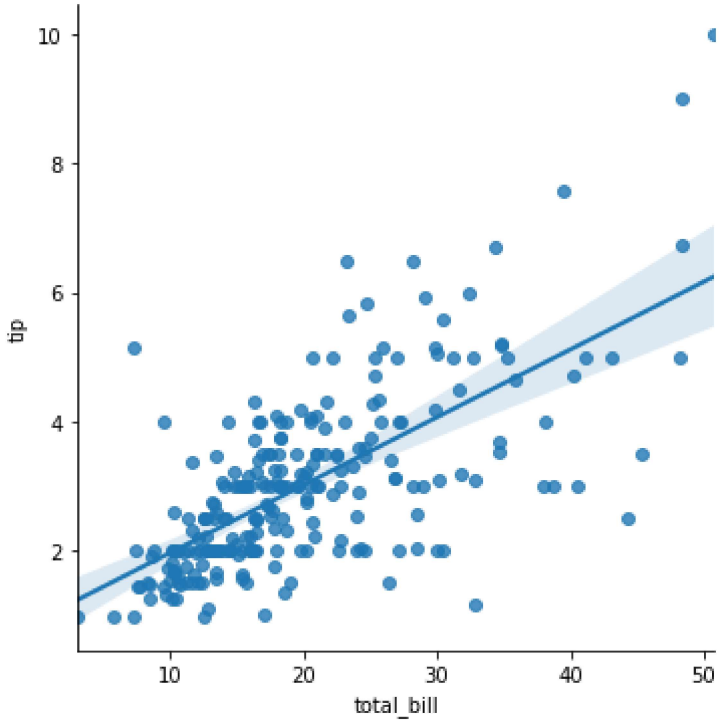
Ex 1-Plotting the regplot and then lmplot with the same data in this example

Ex.1-Plotting the regplot and then lmplot with the same data in this example

```
In [28]: sns.regplot(x='total_bill',y='tip',data=df)
         sns.lmplot(x = "total_bill", y = "tip", data = df)
```

```
Out[28]: <seaborn.axisgrid.FacetGrid at 0x19513c46c40>
```





- You can see the difference in the size between two plots.

Matplotlib-

- Matplotlib is one of the most popular Python packages used for data visualization. It is a cross-platform library for making 2D plots from data in arrays. Matplotlib is written in Python and makes use of NumPy, the numerical mathematics extension of Python. It provides an object-oriented API that helps in embedding plots in applications using Python GUI toolkits such as PyQt, WxPythonotTkinter. It can be used in Python and IPython shells, Jupyter notebook and web application servers also

- Matplotlib has a procedural interface named the Pylab, which is designed to resemble MATLAB, a proprietary programming language developed by MathWorks. Matplotlib along with NumPy can be considered as the open source equivalent of MATLAB.
- Matplotlib was originally written by John D. Hunter in 2003. The current stable version is 2.2.0 released in January 2018.

1. Bar Plot-

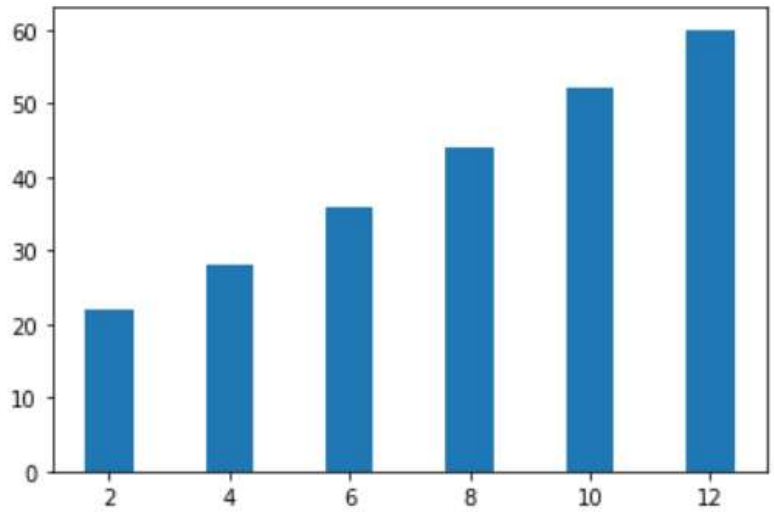
- A bar chart or bar graph is a chart or graph that presents categorical data with rectangular bars with heights or lengths proportional to the values that they represent. The bars can be plotted vertically or horizontally.
- A bar graph shows comparisons among discrete categories. One axis of the chart shows the specific categories being compared, and the other axis represents a measured value.

Ex.1-

```
In [2]: from matplotlib import pyplot as plt
```

```
In [3]: plt.bar([2,4,6,8,10,12],[22,28,36,44,52,60])
```

```
Out[3]: <BarContainer object of 6 artists>
```

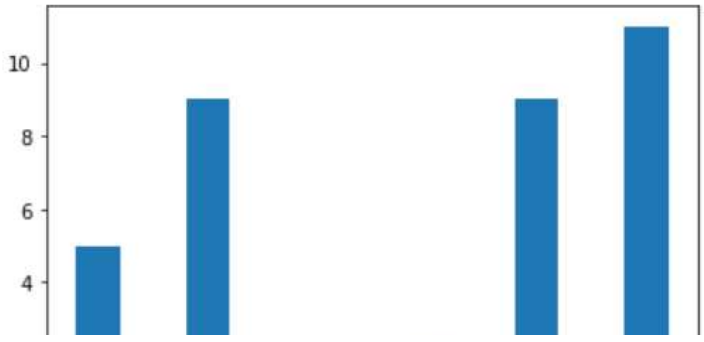


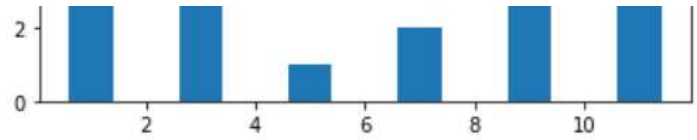
Ex.2-We can pass list as x and y axis.

```
In [9]: bjp=[1,3,5,7,9,11]
        congress=[5,9,1,2,9,11]
```

```
In [10]: plt.bar(bjp,congress)
```

```
Out[10]: <BarContainer object of 6 artists>
```

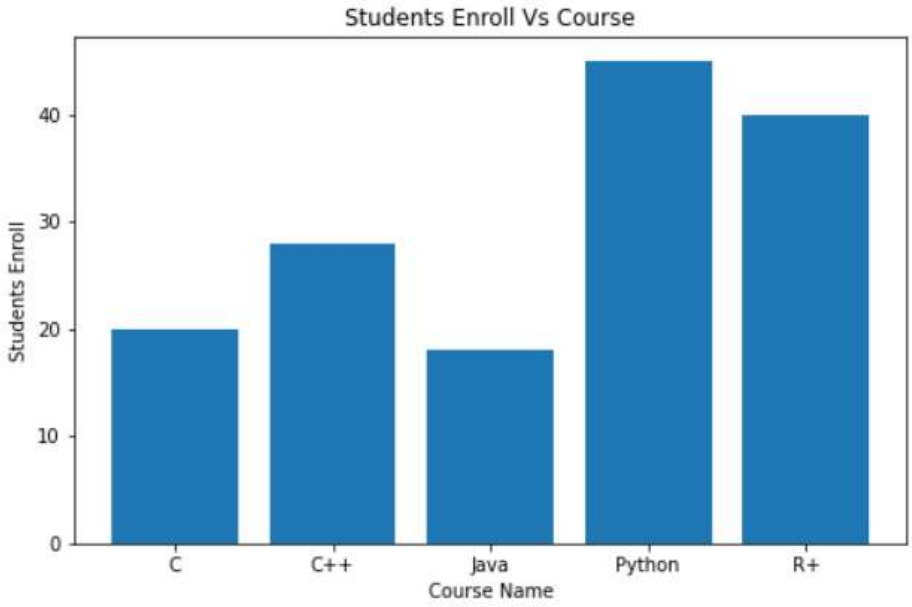




Ex.3-We can give lables to plot and axis.

In [19]:
```
data={'C':20,'C++':28,'Java':18,'Python':45,'R+':40}
course=list(data.keys())
students=list(data.values())

fig=plt.figure(figsize=(8,5))
#Creating Bar Graph

plt.bar(course,students)
plt.xlabel('Course Name')
plt.ylabel('Students Enroll')
plt.title('Students Enroll Vs Course')
```

Out[19]:
```
Text(0.5, 1.0, 'Students Enroll Vs Course')
```



2.Histogram-

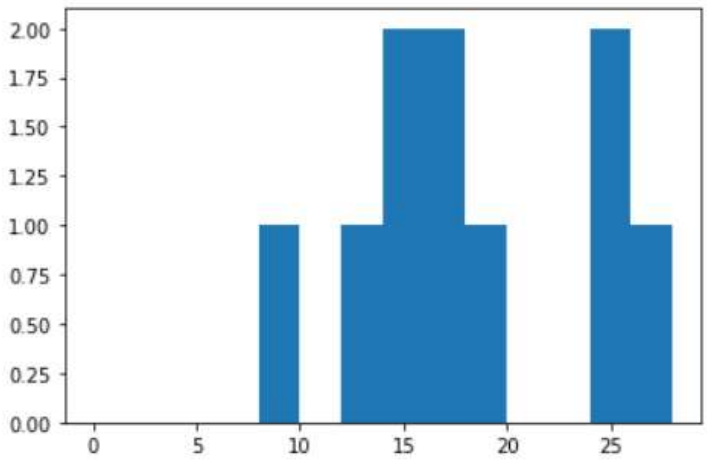
- A histogram is an accurate representation of the distribution of numerical data. It is an estimate of the probability distribution of a continuous variable. It is a kind of bar graph.
- To construct a histogram, follow these steps –
  - **Bin** the range of values.
  - Divide the entire range of values into a series of intervals.
  - Count how many values fall into each interval.
- The bins are usually specified as consecutive, non-overlapping intervals of a variable.
- The matplotlib.pyplot.hist() function plots a histogram. It computes and draws the histogram of x.
- Parameters-

x	array or sequence of arrays

bins	integer or sequence or ‘auto’, optional
optional parameters	
range	The lower and upper range of the bins.
density	If True, the first element of the return tuple will be the counts normalized to form a probability density
cumulative	If True, then a histogram is computed where each bin gives the counts in that bin plus all bins for smaller values.
histtype	The type of histogram to draw. Default is ‘bar’ ▪ ‘bar’ is a traditional bar-type histogram. If multiple data are given the bars are arranged side by side. ▪ ‘barstacked’ is a bar-type histogram where multiple data are stacked on top of each other. ▪ ‘step’ generates a lineplot that is by default unfilled. ▪ ‘stepfilled’ generates a lineplot that is by default filled.

Example-

```
In [26]: class_age=[16,18,17,26,25,25,15,14,12,9]
         bins=[0,2,4,6,8,10,12,14,16,18,20,22,24,26,28]

In [27]: plt.hist(class_age,bins=bins)

Out[27]: (array([0., 0., 0., 0., 1., 0., 1., 2., 2., 1., 0., 0., 2., 1.]),
          array([ 0,  2,  4,  6,  8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28]),
          <BarContainer object of 14 artists>)
```



3.PieChart-

- *A Pie Chart can only display one series of data.* Pie charts show the size of items (called wedge) in one data series, proportional to the sum of the items. The data points in a pie chart are shown as a percentage of the whole pie.
- The pie chart looks best if the figure and axes are square, or the Axes aspect is equal.

- Parameters-

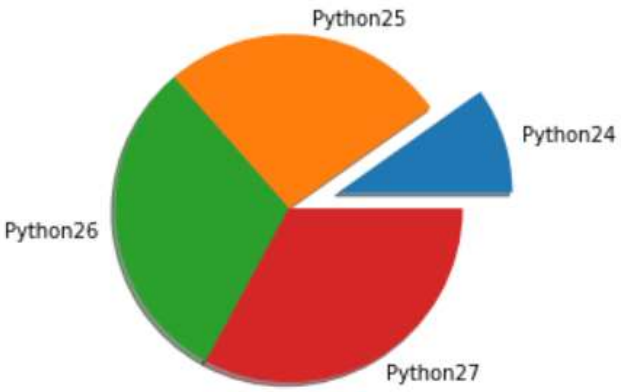
Following table lists down the parameters foe a pie chart −

x	array-like. The wedge sizes.
labels	list. A sequence of strings providing the labels for each wedge.
Colors	A sequence of matplotlibcolorargs through which the pie chart will cycle. If None, will use the colors in the currently active cycle.
Autopct	string, used to label the wedges with their numeric value. The label will be placed inside the wedge. The format string will be fmt%pct.

Example-

```
In [45]: Students_Count=[25,68,78,84]
         Batch_No=['Python24','Python25','Python26','Python27']
```

```
In [47]: plt.pie(Students_Count,labels=Batch_No,shadow=True,explode=(0.3,0,0,0))
```



4.Scatter Plot-

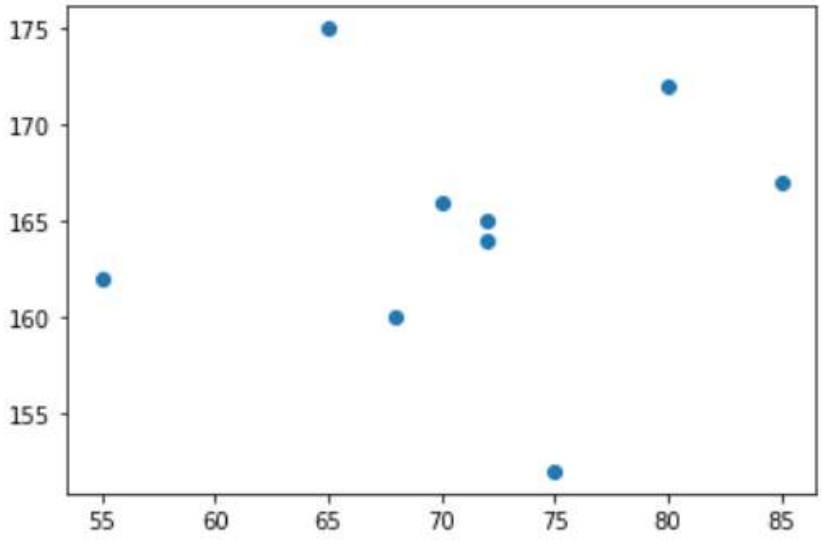
- Scatter plots are used to plot data points on horizontal and vertical axis in the attempt to show how much one variable is affected by another. Each row in the data table is represented by a marker the position depends on its values in the columns set on the X and Y axes. A third variable can be set to correspond to the color or size of the markers, thus adding yet another dimension to the plot.
- *This plot is also used for Outlier detection and for Univariate analysis and Bivariate analysis.*

Ex.1-

```
In [55]: weight=[55,65,68,72,70,72,85,75,80]
         height=[162,175,160,164,166,165,167,152,172]
```

```
In [56]: plt.scatter(weight,height)
```

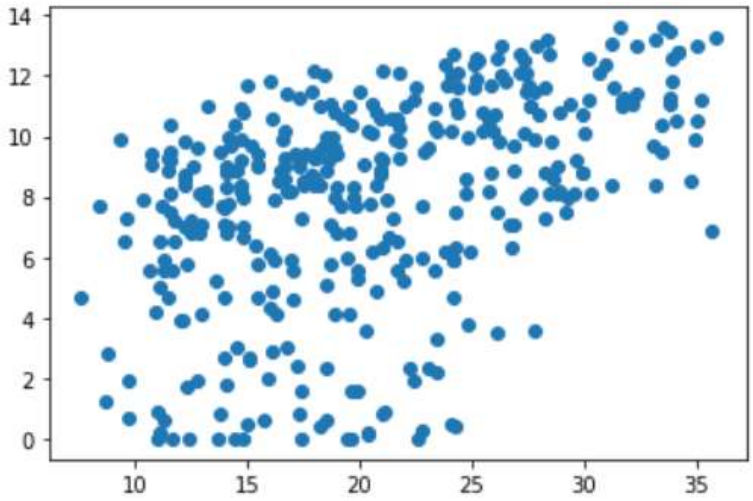
```
Out[56]: <matplotlib.collections.PathCollection at 0x255aced8310>
```



Ex.2-

```
In [64]: plt.scatter(df['MaxTemp'],df['Sunshine'])
```

```
Out[64]: <matplotlib.collections.PathCollection at 0x255afb84520>
```



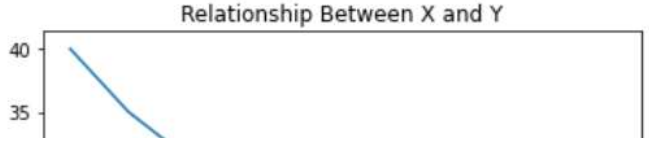
5.Line Graph-

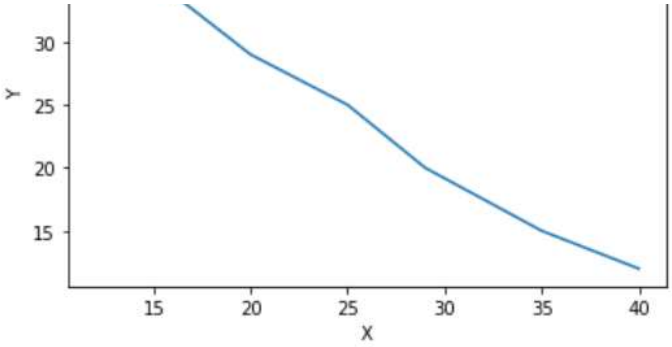
```
In [111]: x=[12,15,20,25,29,35,40]
```

```
In [112]: y=[40,35,29,25,20,15,12]
```

```
In [113]: plt.plot(x,y)
          plt.xlabel('X')
          plt.ylabel('Y')
          plt.title('Relationship Between X and Y')
```

```
Out[113]: Text(0.5, 1.0, 'Relationship Between X and Y')
```



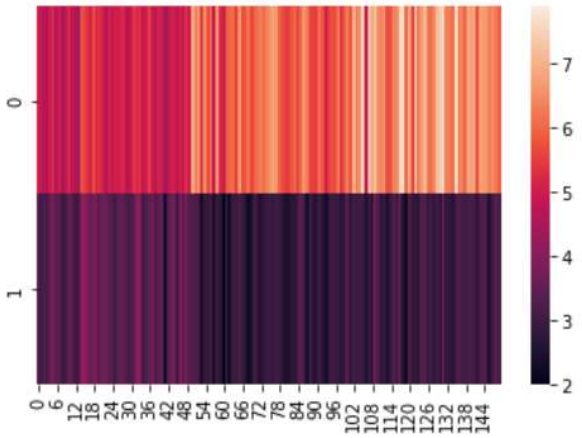


6.Heat Map-

- Heatmap is basically used for feature selection and to find co-relation between features.

Ex.1-

```
In [18]: df5=sns.load_dataset('iris')

In [24]: sns.heatmap(data=(df5['sepal_length'],df5['sepal_width']))
Out[24]: <AxesSubplot:>
```



Numpy -

- NumPy stands for numeric python which is a python package for the computation and processing of the multidimensional and single dimensional array elements.
- **Travis Oliphant** created NumPy package in 2005 by injecting the features of the ancestor module Numeric into another module Numarray.
- It is an extension module of Python which is mostly written in C. It provides various functions which are capable of performing the numeric computations with a high speed.
- NumPy provides various powerful data structures, implementing multi-dimensional arrays and matrices. These data structures are used for the optimal computations regarding arrays and matrices.
- With the revolution of data science, data analysis libraries like NumPy, SciPy, Pandas, etc. have seen a lot of growth. With a much easier syntax than other programming languages, python is the first choice language for the data scientist.

NumPy provides a convenient and efficient way to handle the vast amount of data. *NumPy is very convenient with Matrix multiplication and data reshaping.* NumPy is fast which makes it reasonable to work with a large set of data.

- There are the following advantages of using NumPy for data analysis.
  1. NumPy performs array-oriented computing.
  2. It efficiently implements the multidimensional arrays.
  3. It performs scientific computations.
  4. It is capable of performing Fourier Transform and reshaping the data stored in multidimensional arrays.
  5. NumPy provides the in-built functions for linear algebra and random number generation.
- Nowadays, NumPy in combination with SciPy and Mat-plotlib is used as the replacement to MATLAB as Python is more complete and easier programming language than MATLAB.
- **Why we are using Numpy when we have List..??**
  1. Its occupy less memory compare to list.
  2. Its actually pretty fast.
  3. It is very convenient to work with NumPy.

1.Create One Dimensional Array-

- Syntax-

np.array([1,2,3,4])

```
In [2]: import numpy as np

In [3]: a=np.array([1,2,3,4])

In [4]: a
Out[4]: array([1, 2, 3, 4])
```

2. Create Two Dimensional Array-

```
In [7]: b=np.array([(1,2,3,4),(5,6,7,8)])

In [8]: b
Out[8]: array([[1, 2, 3, 4],
               [5, 6, 7, 8]])
```

3.Find The Dimensions Of The Array-

- Syntax=

<array>.ndim

```
In [18]: b
Out[18]: array([[1, 2, 3, 4],
                [5, 6, 7, 8]])
```

```
In [12]: b.ndim

Out[12]: 2
```

4.Find Byte size of Array-

- Syntax-

<array>.itemsize

```
In [20]: b

Out[20]: array([[1, 2, 3, 4],
                [5, 6, 7, 8]])

In [19]: b.itemsize

Out[19]: 4
```

5.Find Data Type Of Array-

- Syntax-

<array>.dtype

```
In [22]: b

Out[22]: array([[1, 2, 3, 4],
                [5, 6, 7, 8]])

In [21]: b.dtype

Out[21]: dtype('int32')
```

6.Find Shape Of Array-

- Shape of array nothing but no of columns and rows present in array.
- Size of Array will give no of elements present in array.

```
In [25]: b

Out[25]: array([[1, 2, 3, 4],
                [5, 6, 7, 8]])

In [24]: b.size

Out[24]: 8

In [26]: b.shape

Out[26]: (2, 4)
```

7.Reshape Of Arrays-

- When you change number of columns and rows that is called reshaping.

```
In [27]: c=np.array([(1,2,3,4,5),(4,5,6,7,8)])

In [28]: c

Out[28]: array([[1, 2, 3, 4, 5],
```

```
       [4, 5, 6, 7, 8]])
```

```
In [29]: c.shape
```

```
Out[29]: (2, 5)
```

```
In [36]: c.reshape(5,2)
```

```
Out[36]: array([[1, 2],
                [3, 4],
                [5, 4],
                [5, 6],
                [7, 8]])
```

8.Slicing-

- Slicing is basically extracting set of elements from your array.
- When we say 0: this actually means all the rows that will include 0 as well.

```
In [39]: c
```

```
Out[39]: array([[1, 2, 3, 4, 5],
                [4, 5, 6, 7, 8]])
```

```
In [50]: c[0:,0]          #if we have to fetch zeroth index from each rows
```

```
Out[50]: array([1, 4])
```

```
In [51]: c[1:,1]         #if we have to fetch first index value from 2nd row
```

```
Out[51]: array([5])
```

```
In [52]: c[0:1]          #if we have to fetch only first row from array
```

```
Out[52]: array([[1, 2, 3, 4, 5]])
```

```
In [54]: c[1:2]         #if we have to fetch only second row from array
```

```
Out[54]: array([[4, 5, 6, 7, 8]])
```

```
In [62]: c[0:2]         #if we have to fetch first and second row from array
```

```
Out[62]: array([[1, 2, 3, 4, 5],
                [4, 5, 6, 7, 8]])
```

9.Find Min,Max,Mean,Standard Deviation of array

- We can perform Min,Max,Mean,Standard Deviation operations on array .

```
In [69]: c
```

```
Out[69]: array([[ 1,  2,  3,  4,  5],
                [ 4,  5,  6,  7,  8],
                [ 7,  8,  9, 10, 11],
                [ 5,  6,  7,  9, 10]])
```

```
In [82]: c.max(),c.min(),c.sum(),c.mean(),c.std(),
```

```
Out[82]: (11, 1, 127, 6.35, 2.6884010117540127)
```

10.Axis-

- axis =0 >>Refer to Columns
- axis =1>> Refer to Rows

```
In [83]: c

Out[83]: array([[ 1,  2,  3,  4,  5],
                [ 4,  5,  6,  7,  8],
                [ 7,  8,  9, 10, 11],
                [ 5,  6,  7,  9, 10]])

In [85]: c.sum(axis=0)

Out[85]: array([17, 21, 25, 30, 34])

In [86]: c.sum(axis=1)

Out[86]: array([15, 30, 45, 37])
```

11.Arithematic Operations on Arrays-

- We can perform Matrix addition, Subtraction,Multiplication and Division on array

```
In [97]: c+d,c-d,c*d,c/d

Out[97]: (array([[ 2,  4,  6,  8, 10],
                 [ 8, 10, 12, 14, 16],
                 [14, 16, 18, 20, 22],
                 [10, 12, 14, 18, 20]]),
          array([[0, 0, 0, 0, 0],
                 [0, 0, 0, 0, 0],
                 [0, 0, 0, 0, 0],
                 [0, 0, 0, 0, 0]]),
          array([[  1,   4,   9,  16,  25],
                 [ 16,  25,  36,  49,  64],
                 [ 49,  64,  81, 100, 121],
                 [ 25,  36,  49,  81, 100]]),
          array([[1., 1., 1., 1., 1.],
                 [1., 1., 1., 1., 1.],
                 [1., 1., 1., 1., 1.],
                 [1., 1., 1., 1., 1.]]))
```

12.Concatenate Arrays-

- We have two methods of stack i.e Vertical stack and Horizantal stack

```
In [100]: np.vstack((c,d))

Out[100]: array([[ 1,  2,  3,  4,  5],
                 [ 4,  5,  6,  7,  8],
                 [ 7,  8,  9, 10, 11],
                 [ 5,  6,  7,  9, 10],
                 [ 1,  2,  3,  4,  5],
                 [ 4,  5,  6,  7,  8],
                 [ 7,  8,  9, 10, 11],
                 [ 5,  6,  7,  9, 10]])

In [101]: np.hstack((c,d))

Out[101]: array([[ 1,  2,  3,  4,  5,  1,  2,  3,  4,  5],
                 [ 4,  5,  6,  7,  8,  4,  5,  6,  7,  8],
                 [ 7,  8,  9, 10, 11,  7,  8,  9, 10, 11],
                 [ 5,  6,  7,  9, 10,  5,  6,  7,  9, 10]])
```

13.Convert Multidimensional array to single column-

```
In [106]: c
```

```
Out[106]: array([[ 1,  2,  3,  4,  5],
                 [ 4,  5,  6,  7,  8],
                 [ 7,  8,  9, 10, 11],
                 [ 5,  6,  7,  9, 10]])
```

```
In [105]: np.ravel(c)
```

```
Out[105]: array([ 1,  2,  3,  4,  5,  4,  5,  6,  7,  8,  7,  8,  9, 10, 11,  5,  6,
                  7,  9, 10])
```

Exploratory Data Analysis-

- If we want to explain EDA in simple terms, it means trying to understand the given data much better, so that we can make some sense out of it.
- *In statistics, **exploratory data analysis** is an approach to analyzing data sets to summarize their main characteristics, often with visual methods. A statistical model can be used or not, but primarily EDA is for seeing what the data can tell us beyond the formal modeling or hypothesis testing task.*
- EDA in Python uses data visualization to draw meaningful patterns and insights. It also involves the preparation of data sets for analysis by removing irregularities in the data.
- Based on the results of EDA, companies also make business decisions, which can have repercussions later.
- If EDA is not done properly then it can hamper the further steps in the machine learning model building process.
- *If EDA is not done properly then it can hamper the further steps in the machine learning model building process.*
- *If done well, it may improve the efficiency of everything we do next.*
- EDA contains below topics-

1. Data Sourcing.
2. Data Cleaning.
3. Univariate analysis.
4. Bivariate analysis.
5. Multivariate analysis.

A)Data Sourcing-

- Data Sourcing is the process of finding and loading the data into our system. Broadly there are two ways in which we can find data.

1. Private Data-
2. Public Data-

Private Data-

- As the name suggests, private data is given by private organizations. There are some security and privacy concerns attached to it. This type of data is used for mainly organizations internal analysis.

Public Data-

- This type of Data is available to everyone. We can find this in government websites and public

- This type of Data is available to everyone. We can find this in government websites and public organizations etc. Anyone can access this data, we do not need any special permissions or approval.
- The very first step of EDA is Data Sourcing, we have seen how we can access data and load into our system. Now, the next step is how to clean the data.

B)Data Cleaning-

- After completing the Data Sourcing, the next step in the process of EDA is **Data Cleaning**. It is very important to get rid of the irregularities and clean the data after sourcing it into our system.
- Irregularities are of different types of data.

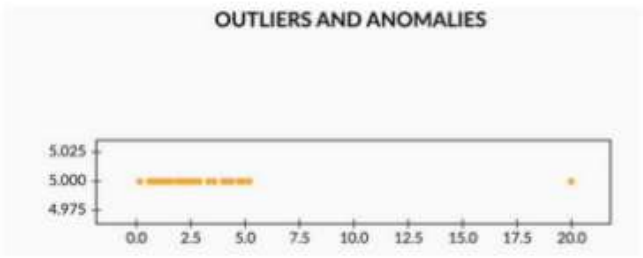
1. Missing Values-
2. Incorrect Format-
3. Incorrect Headers-
4. Anomalies/Outliers-
5. Standardizing Values-

i)Missing Values-

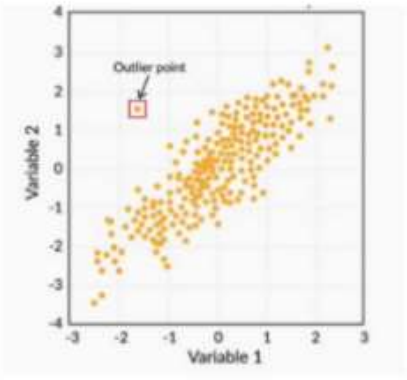
- If there are missing values in the Dataset before doing any statistical analysis, we need to handle those missing values.
- There are mainly three types of missing values.
    1. MCAR(Missing completely at random): These values do not depend on any other features.
    2. MAR(Missing at random): These values may be dependent on some other features.
    3. MNAR(Missing not at random): These missing values have some reason for why they are missing.

ii)Anomalies/Outliers-

- *Outliers are the values that are far beyond the next nearest data points.*
- There are two types of outliers:
    1. Univariate outliers: Univariate outliers are the data points whose values lie beyond the range of expected values based on one variable.
    2. Multivariate outliers: While plotting data, some values of one variable may not lie beyond the expected range, but when you plot the data with some other variable, these values may lie far from the expected value.



Univariate Outlier



Multivariate Outlier

- So, after understanding the causes of these outliers, we can handle them by dropping those records or imputing with the values or leaving them as is, if it makes more sense.

iii)Standardizing Values-

- To perform data analysis on a set of values, we have to make sure the values in the same column should be on the same scale. For example, if the data contains the values of the top speed of different companies' cars, then the whole column should be either in meters/sec scale or miles/sec scale.

C)Univariate Analysis-

- If we analyze data over a single variable/column from a dataset, it is known as Univariate Analysis.
- If the column or variable is of numerical then we'll analyze by calculating its mean, median, std, etc. We can get those values by using the describe function.
- *We use to **Pie chart** and **Bar charts** for analysis of categorical ordered/unordered Univariate variables.*

i)Categorical Unordered Univariate Analysis-

- An unordered variable is a categorical variable that has no defined order.

ii)Categorical Ordered Univariate Analysis-

- Ordered variables are those variables that have a natural rank of order.

D)Bivariate Analysis-

- If we analyze data by taking two variables/columns into consideration from a dataset, it is known as Bivariate Analysis.

i)Numeric -Numeric Analysis-

- Analyzing the two numeric variables from a dataset is known as numeric-numeric analysis. We can analyze it in three different ways.
  - Scatter Plot-
  - Pair Plot-
  - Correlation Matrix-It is difficult to see the relation between three numerical variables in a single graph. In those cases, we'll use the correlation matrix and then we use this matrix for mapping of Heatmap.

ii)Numeric - Categorical analysis-

- Analyzing the one numeric variable and one categorical variable from a dataset is known as numeric-categorical analysis. *We analyze them mainly using* mean, median, and box plots.

iii) Categorical- Categorical-

- Analyzing the two categorical dataset is known as categorical-categorical analysis

---

E)Multivariate Analysis-

- If we analyze data by taking more than two variables/columns into consideration from a dataset, it is known as Multivariate Analysis.
- For Multivariate data analysis we can use Heat map by using pivot table.

Conclusions-

- This is how we'll do Exploratory Data Analysis. Exploratory Data Analysis (EDA) helps us to look beyond the data. The more we explore the data, the more the insights we draw from it. As a

data analyst, almost 80% of our time will be spent understanding data and solving various business problems through EDA.

- [ ] **Random Oversampling and Under-sampling for Imbalanced classification-**

- Imbalanced datasets are those where there is a severe skew in the class distribution, such as 1:100 or 1:1000 examples in the minority class to the majority class.
- This bias in the training dataset can influence many machine learning algorithms, leading some to ignore the minority class entirely. This is a problem as it is typically the minority class on which predictions are most important.
- One approach to addressing the problem of class imbalance is to randomly resample the training dataset. The two main approaches to randomly resampling an imbalanced dataset are *to delete examples from the majority class, called undersampling, and to duplicate examples from the minority class, called oversampling.*
- In short we can say that,
  - **Random Oversampling**: Randomly add duplicate examples in the minority class.
  - **Random Under-sampling**: Randomly delete examples in the majority class.
- **Random Re-Sampling -** *Applying re-sampling strategies to obtain a more balanced data distribution is an effective solution to the imbalance problem.*
- **Example1**- For Oversampling .

1.Load data set-

```
In [3]: df=pd.read_csv('creditcard.csv')
```

```
In [4]: df
```

Out[4]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	...
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	...
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	...
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	...
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	...

2.Check DataFrame Shape-

```
In [5]: df.shape
```

```
Out[5]: (284807, 31)
```

3.Find values counts from targeted features-

```
In [6]: df['Class'].value_counts()
```

```
Out[6]: 0    284315
        1       492
```

```
Name: Class, dtype: int64
```

4.Split Target feature and Independent Feature-

- There are several ways for splitting,

1. Using iloc - df.ioc[: , 0 :30]
2. Using list Comprehensions- [col for col in df.columns if col != 'Class']
3. Using Drop - df.drop(' Class ')

- Split Data as X= Independent Features and Y = Target Feature

```
In [10]: X=df.drop('Class',axis=1)
```

```
In [11]: X
```

Out[11]:

V5	V6	V7	V8	V9	...	V20	V21	V22	V23	V24	V25	V26	V27	V28	Amount
-0.338321	0.462388	0.239599	0.098698	0.363787	...	0.251412	-0.018307	0.277838	-0.110474	0.066928	0.128539	-0.189115	0.133558	-0.021053	149.62
0.060018	-0.082361	-0.078803	0.085102	-0.255425	...	-0.069083	-0.225775	-0.638672	0.101288	-0.339846	0.167170	0.125895	-0.008983	0.014724	2.69
-0.503198	1.800499	0.791461	0.247676	-1.514654	...	0.524980	0.247998	0.771679	0.909412	-0.689281	-0.327642	-0.139097	-0.055353	-0.059752	378.66

```
In [12]: Y=df['Class']
```

```
In [14]: Y.shape
```

```
Out[14]: (284807,)
```

5.Check Value counts from Target Feature-

```
In [15]: Y.value_counts()
```

```
Out[15]: 0    284315
         1       492
         Name: Class, dtype: int64
```

6.Sampling -

- As we can observed that from Values counts ,data is imbalance so we have to use do Over-sampling.
- So install module package imblearn in your machine, and import class- RandomOverSampler and create object of class and give customize percentage of matching/duplication of data.

```
In [21]: from imblearn.over_sampling import RandomOverSampler
```

```
In [ ]: obj=RandomOverSampler(0.85)
```

---

7. Fit Samples-

- Fit samples to X_new and Y_Nnew
- After fitting of samples, we can see that Target features have near about same datapoints for 0 and 1.

```
In [32]: X_new,Y_new=obj.fit_resample(0.85)

In [37]: X.shape

Out[37]: (284807, 30)

In [35]: Y_new.value_counts()

Out[35]: 0    284315
         1    213236
         Name: Class, dtype: int64
```

Example 2- For Under-Sampling-

8. Import Class NearMiss-

- import class NearMiss,and create object .

```
In [38]: from imblearn.under_sampling import NearMiss

In [39]: obj1=NearMiss(0.85)

         C:\Users\Abhimanyu Devadhe\anaconda3\lib\site-packages\imblearn\utils\_validation
         =0.85 as keyword args. From version 0.9 passing these as positional arguments wil
           warnings.warn(
```

---

9. Fit Samples -

- Fit samples to X_New_Under and Y_New_Under,and then check value count of target feature.

```
In [40]: X_New_Under,Y_New_Under=obj.fit_resample(X,Y)

In [41]: Y_New_Under.value_counts()

Out[41]: 0    284315
         1    213236
         Name: Class, dtype: int64
```

- [ ] **Synthetic Minority Oversampling Technique (SMOTE)-**

1.Problem Statement-

- While dealing with classification problems the percentage of classes in the total samples an important role however there are scenarios you may be dealing with imbalance dataset i.e Presence of minority class of in dataset.

2.Challenges Related to imbalanced Datasets-

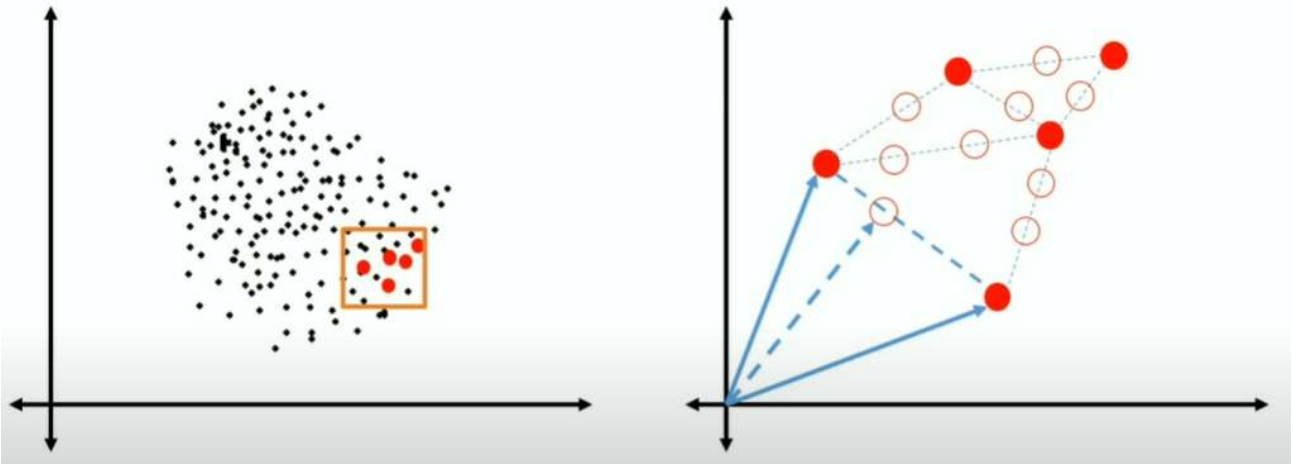
- Biased Predication.
- Misleading Accuracy.

3.Some Examples-

- Credit cards frauds.
- Manufacturing Defects.
- Rare diseases diagnosis.
- Natural Disaters.
- Enrolment to premier institutes.

4.Solution-

- For dealing with imbalanced dataset on of the way is to resample the dataset by either decreasing the the majority class or increasing minority class observations.
- **SMOTE-**It creates new synthetic observations or new datapoints .some are the steps below,



1)Identify the feature vector and its nearest neighbour.
2)Take the difference between two.
3)Multiply the difference with random number by in between 0 and 1.
4)Identify the new point on the line segment by adding random number to feature vector.
5)Repeat the process for identified feature vector.

Missing Values-

- If there are missing values in the Dataset before doing any statistical analysis, we need to handle those missing values.
- There are mainly three types of missing values.
  1. MCAR(Missing completely at random): These values do not depend on any other features.
  2. MAR(Missing at random): These values may be dependent on some other features.
  3. MNAR(Missing not at random): These missing values have some reason for why they are missing.
- Find Actions to be taken according Null values percentage in Feature-
  1. 80 to 90 % Null values- Drop feature.
  2. 40 to 60 % Null values- Build internal machine learning model for fill null values.
  3. 20 to 40 % Null values- Use Statistical techniques- Mean,Median and Mode.
- It is not hard code rule,just good practice for data analysis.

1.Read data -

```
In [43]: df=pd.read_csv('Bengaluru_House_Data.csv')
```

```
In [44]: df
```

Out[44]:

	area_type	availability	location	size	society	total_sqft	bath	balcony	price
0	Super built-up Area	19-Dec	Electronic City Phase II	2 BHK	Coomee	1056	2.0	1.0	39.07
1	Plot Area	Ready To Move	Chikka Tirupathi	4 Bedroom	Theanmp	2600	5.0	3.0	120.00
2	Built-up Area	Ready To Move	Uttarahalli	3 BHK	NaN	1440	2.0	3.0	62.00
3	Super built-up Area	Ready To Move	Lingadheeranahalli	3 BHK	Soiewre	1521	3.0	1.0	95.00
4	Super built-up Area	Ready To Move	Kothanur	2 BHK	NaN	1200	2.0	1.0	51.00

2.Check shape of DataFrame-

```
In [46]: df.shape
Out[46]: (13320, 9)
```

3.Check Null values of each feature-

```
In [47]: df.isnull().sum()
Out[47]: area_type          0
         availability       0
         location           1
         size              16
         society         5502
         total_sqft         0
         bath              73
         balcony          609
         price              0
         dtype: int64
```

- We can observed that there is 5 feature in dataset which contains null values,and we have to fill or drop null values features according to scenario.

4.Drop society feature -

- We are dropping society feature from Dataframe.

```
In [48]: df_drop_society=df.drop('society',axis=1)
```

```
In [49]: df_drop_society
```

Out[49]:

	area_type	availability	location	size	total_sqft	bath	balcony	price
0	Super built-up Area	19-Dec	Electronic City Phase II	2 BHK	1056	2.0	1.0	39.07
1	Plot Area	Ready To Move	Chikka Tirupathi	4 Bedroom	2600	5.0	3.0	120.00
2	Built-up Area	Ready To Move	Uttarahalli	3 BHK	1440	2.0	3.0	62.00
3	Super built-up Area	Ready To Move	Lingadheeranahalli	3 BHK	1521	3.0	1.0	95.00

5.Fill null values for feature-

- We are dropped society from dataframe, but still we have 4 features which have null values.
- Now we have to fill user defined/customized/mean values in balcony feature.

```
In [52]: df_drop_society['balcony'].unique()
Out[52]: array([ 1.,  3., nan,  2.,  0.])
```

```
In [57]: df_drop_society['balcony']=df_drop_society['balcony'].fillna(df_drop_society['balcony'].mean())
```

```
In [58]: df_drop_society
```

Out[58]:

	area_type	availability	location	size	total_sqft	bath	balcony	price
0	Super built-up Area	19-Dec	Electronic City Phase II	2 BHK	1056	2.0	1.000000	39.07
1	Plot Area	Ready To Move	Chikka Tirupathi	4 Bedroom	2600	5.0	3.000000	120.00
2	Built-up Area	Ready To Move	Uttarahalli	3 BHK	1440	2.0	3.000000	62.00
3	Super built-up Area	Ready To Move	Lingadheeranahalli	3 BHK	1521	3.0	1.000000	95.00

- Now we have 3 feature which contains null values.

```
In [59]: df_drop_society.isnull().sum()
Out[59]: area_type        0
         availability     0
         location         1
         size            16
         total_sqft       0
         bath            73
         balcony          0
         price            0
```

6.Check feature have proper formatting-

- Check data types of all features,

```
In [63]: df_drop_society.dtypes
Out[63]: area_type        object
         availability     object
         location         object
         size             object
         total_sqft       object
         bath            float64
         balcony         float64
         price           float64
```

- We can observed that total_sqft should be have int data type but due to improper data entry it have object(string) data type.
- so we have to change data type of total_sqft, use linear regression for finding mean of range of sqft feet.

```
In [64]: df_drop_society['total_sqft'].unique()
```

```
Out[64]: array(['1056', '2600', '1440', ..., '1133 - 1384', '774', '4689'],
               dtype=object)
```

```
In [82]: import re
         def test(x):
             obj=re.findall(r'\d+',x)
             if len(obj)>1:
                 return (int(obj[0])+int(obj[1]))/2
             return obj[0]
```

```
In [83]: df_drop_society['total_sqft_new']=df_drop_society['total_sqft'].apply(test)
```

```
In [84]: df_drop_society
```

Out[84]:

	area_type	availability	location	size	total_sqft	bath	balcony	price	total_sqft_new
0	Super built-up Area	19-Dec	Electronic City Phase II	2 BHK	1056	2.0	1.000000	39.07	1056
1	Plot Area	Ready To Move	Chikka Tirupathi	4 Bedroom	2600	5.0	3.000000	120.00	2600
2	Built-up Area	Ready To Move	Uttarahalli	3 BHK	1440	2.0	3.000000	62.00	1440
3	Super built-up Area	Ready To Move	Lingadheeranahalli	3 BHK	1521	3.0	1.000000	95.00	1521
4	Super built-up Area	Ready To Move	Kothanur	2 BHK	1200	2.0	1.000000	51.00	1200

- We have cleaned data from feature total_sqft and make new column as total_sqft_new.
- Now we can change data type of total_sqft_new as int,

```
In [89]: df_drop_society.dtypes
```

```
Out[89]: area_type          object
         availability       object
         location           object
         size               object
         total_sqft         object
         bath              float64
         balcony           float64
         price             float64
         total_sqft_new     object
         dtype: object
```

```
In [94]: df_drop_society['total_sqft_new']=df_drop_society['total_sqft_new'].astype('int')
```

```
In [95]: df_drop_society.dtypes
```

```
Out[95]: area_type          object
         availability       object
         location           object
         size               object
         total_sqft         object
         bath              float64
         balcony           float64
         price             float64
         total_sqft_new      int32
```

Handling categorical Values-

- As computer has its own language, machine learning algorithms work on numerical data. Whenever we have categorical data we have to Encode this data for better understanding to algorithm and achieve optimum accuracy from model.
- So have below Encode techniques to handle categorical data and make it useful for the

machine learning algorithm to get insightful information.

1.One -Hot Encoding-

- One-Hot Encoding is a very handy and popular technique for treating categorical features. This is based on creating additional features by its unique values. *Every unique value in this is a added feature and values are assigned as 1 or 0 based on the presence of it in a row.*
- In Python it can be implemented as:

```
In [31]: df
```

Out[31]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_Hist
0	LP001032	Male	No	0	Graduate	No	4950	0.0	125	360	
1	LP001824	Male	Yes	1	Graduate	No	2882	1843.0	123	480	
2	LP002928	Male	Yes	0	Graduate	No	3000	3416.0	56	180	
3	LP001814	Male	Yes	2	Graduate	No	9703	0.0	112	360	
4	LP002244	Male	Yes	0	Graduate	No	2333	2417.0	136	360	

- Use get dummies class from pandas -

```
In [44]: col=[col for col in df if df[col].dtypes=='object']
```

```
In [45]: pd.get_dummies(df[col])
```

Out[45]:

Dependents_3+	Education_Graduate	Education_Not Graduate	Self_Employed_No	Self_Employed_Yes	Property_Area_Rural	Property_Area_Semiurban	Property_Area_Urban
0	1	0	1	0	0	0	1
0	1	0	1	0	0	1	0
0	1	0	1	0	0	1	0
0	1	0	1	0	0	0	1

- in above example we have encode all features which have object data type.
- **Assumptions:**
  1. There are finite set of features.
  2. Where no ordinal relationship exists between the categories of variable.
- **Advantages:**

  1) Easy to use.

  2) Creates no bias as assumption of any ordering between the categories.
- **Disadvantages:**

  1) Can result in an increase in number of features resulting in performance issues.

  2)It Increases Feature space.

2.Ordinal Number Encoding-

- This is a popular technique, in which each label is assigned a unique integer based on alphabetical ordering.
- This is easiest way and used in most of the data where there is natural relation between the categories of ordinal values.

```
In [ ]: from sklearn.preprocessing import LabelEncoder
```

```
In [ ]: obj=LabelEncoder()
```

```
In [60]: df['Property_Area_Ordinal']=obj.fit_transform(df['Property_Area'])
```

```
In [63]: df['Property_Area_Ordinal'].value_counts()
```

```
Out[63]: 1    146
         2    125
```

```
0    113
Name: Property_Area_Ordinal, dtype: int64
```

```
In [64]: df['Property_Area'].value_counts()
```

```
Out[64]: Semiurban    146
         Urban        125
         Rural        113
         Name: Property_Area, dtype: int64
```

- We can observed that there are 3 categories in 'Property_Area',so 0,1,2 label allocated as per alphabetical order of categories.
- **Assumptions:**

1. The integer values are having natural ordered relationship between each other. Like Current>Ex>Never.

- **Advantages:**

1) Easy to use

2) Easily reversible.

3) Doesn't increase feature space .

- **Disadvantages:**

1) May result in unexpected results if the ordering of number is not related in any order.

3.Count Or Frequency Encoding-

- In this type of encoding the count of existence of each category in the variable is determined. Each category is then replaced by the frequency of it.

```
In [69]: Property_Area_Count=df['Property_Area'].value_counts().to_dict()
```

```
In [78]: df['Property_Area_Count']=df['Property_Area'].map(Property_Area_Count)
```

```
In [79]: df.set_index('Property_Area_Count',inplace=True)
```

```
In [81]: df['Property_Area'].value_counts()
```

```
Out[81]: Semiurban    146
         Urban        125
         Rural        113
         Name: Property_Area, dtype: int64
```

```
In [82]: df
```

Out[82]:

Property_Area_Count	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	C
125	LP001032	Male	No	0	0	No	4950	
146	LP001824	Male	Yes	1	0	No	2882	
146	LP002928	Male	Yes	0	0	No	3000	
125	LP001814	Male	Yes	2	0	No	9703	

- **Assumptions:**

There is no category of variable having similar frequency.

- **Advantages**

1. Easy to use.
2. Easily reversible

2. Easily reversible.
3. Doesn't increase feature space .

- **Disadvantages**

1. Will not be able to handle if frequencies are same for two or more categories.

Outlier Removing Methods-

- An **outlier** is an observation that lies abnormally far away from other values in a dataset. Outliers can be problematic because they can affect the results of an analysis.

A) How to Identify Outliers From data set-

- Before you can remove outliers, you must first decide on what you consider to be an outlier. There are two common ways to do so:

1) Use the interquartile Range-

- The interquartile range (IQR) is the difference between the 75th percentile (Q3) and the 25th percentile (Q1) in a dataset. It measures the spread of the middle 50% of values.
- You could define an observation to be an outlier if it is 1.5 times the interquartile range greater than the third quartile (Q3) or 1.5 times the interquartile range less than the first quartile (Q1).
- Formula-

Outliers = Observations > Q3 + 1.5*IQR or Observations < Q1 – 1.5*IQR

2) Use z-scores.

- A z-score tells you how many standard deviations a given value is from the mean. We use the following formula to calculate a z-score:

$$\mathbf{z} = (X - \mu) / \sigma$$

where:

X is a single raw data value
μ is the population mean
σ is the population standard deviation

- You could define an observation to be an outlier if it has a z-score less than -3 or greater than 3.

Outliers = Observations with z-scores > 3 or < -3

B) How to Remove Outliers form Dataframe-

- Once you decide on what you consider to be an outlier, you can then identify and remove them from a dataset.

1.Interquartile Range Method-

- Create DataFrame-

```
In [2]: #Create Dataframe with three columns A,B,C
```

```
In [3]: np.random.seed(10)
        df = pd.DataFrame(np.random.randint(0, 10, size=(100, 3)), columns=['A', 'B', 'C'])
```

```
In [4]: df.head(10)
```

Out[4]:

	A	B	C
0	9	4	0
1	1	9	0

2	1	8	9

- Find Q1,Q2 and Interquartile range for each column-

```
In [5]: from scipy.stats import stats

In [6]: Q1=df.quantile(q=0.25)         # 1st Quadrant

In [7]: Q3=df.quantile(q=0.75)         # 3rd Quadrant

In [8]: Qrtl_Range=df.apply(stats.iqr) #Inter Quartile Range

In [9]: Clean_Data=df[~((df<(Q1-1.5*Qrtl_Range)) | (df>(Q3+1.5*Qrtl_Range))).any(axis=1)]
```

```
In [11]: Clean_Data
```

Out[11]:

	A	B	C
0	9	4	0
1	1	9	0
2	1	8	9

- Like this we can find outliers and remove from data set.

2.Z- Score Method-

- Create DataFrame-

```
In [2]: #Create Dataframe with three columns A,B,C

In [3]: np.random.seed(10)
        df = pd.DataFrame(np.random.randint(0, 10, size=(100, 3)), columns=['A', 'B', 'C'])

In [4]: df.head(10)
```

Out[4]:

	A	B	C
0	9	4	0
1	1	9	0
2	1	8	9

- find absolute value of z-score and keep rows in dataframe with all z-scores less than absolute value of 3

```
In [130]: #find absolute value of z-score for each observation
          z = np.abs(stats.zscore(df))

In [131]: #only keep rows in dataframe with all z-scores less than absolute value of 3
          data_clean = df[(z<3).all(axis=1)]

In [132]: #find how many rows are left in the dataframe
          data_clean.shape
```

Out[132]: (100, 3)

C) When To Remove Outliers-

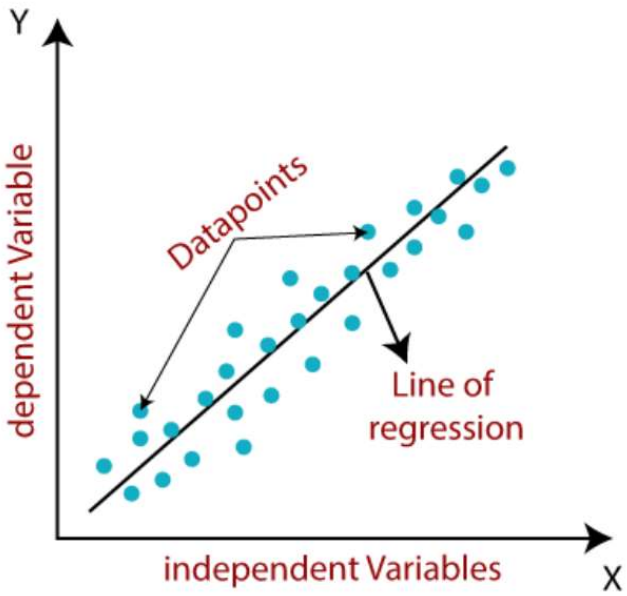
- If one or more outliers are present in your data, you should first make sure that they're not a result of data entry error. Sometimes an individual simply enters the wrong data value when recording data.
- If the outlier turns out to be a result of a data entry error, you may decide to assign a new value to it such as Mean or Median of the dataset.
- If the value is a true outlier, you may choose to remove it if it will have a significant impact on your overall analysis. Just make sure to mention in your final report or analysis that you removed an outlier.

- [ ] **Machine Learning Algorithms-**

- Machine Learning algorithms are the programs that can learn the hidden patterns from the data, predict the output, and improve the performance from experiences on their own.
- Different algorithms can be used in machine learning for different tasks, such as simple linear regression that can be used **for prediction problem**s like **stock market prediction,** and **the KNN algorithm can be used for classification problems**.

1.Linear Regression-

- Linear regression is a statistical model used to predict the relationship between Independent and dependent variables.
- Linear regression makes predictions for continuous/real or numeric variables such as **sales, salary, age, product price,** etc.
- Linear regression algorithm shows a linear relationship between a dependent (y) and one or more independent (X) variables, hence called as linear regression. Since linear regression shows the linear relationship, which means it finds how the value of the dependent variable is changing according to the value of the independent variable.



- **Regression-**Regression analysis is form of predictive modelling techniques which investigates the relationship between a dependent and independent variable.

- [ ] **Types of Linear Regression**

- Linear regression can be further divided into two types of the algorithm:

1.Simple Linear Regression:

- If a single independent variable is used to predict the value of a numerical dependent variable, then such a Linear Regression algorithm is called Simple Linear Regression.

2.Multiple Linear regression:

- If more than one independent variable is used to predict the value of a numerical dependent variable, then such a Linear Regression algorithm is called Multiple Linear Regression.

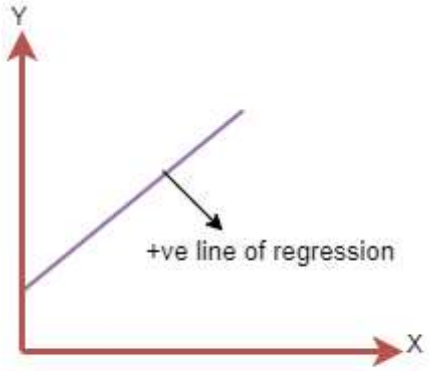
☐ **Basic Terminology in Linear Regression-**

1.Linear Regression Line

- A linear line showing the relationship between the dependent and independent variables is called a **regression line**. A regression line can show two types of relationship:

A)Positive Linear Relationship:

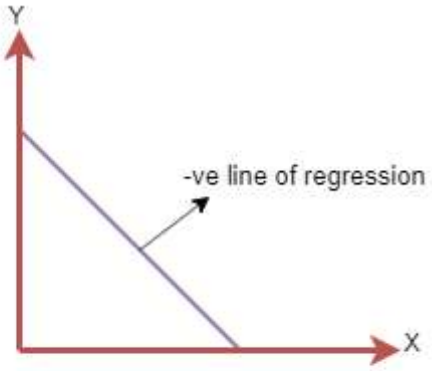
- If the dependent variable increases on the Y-axis and independent variable increases on X-axis, then such a relationship is termed as a Positive linear relationship.



The line equation will be: $Y= a_0+a_1x$

B)Negative Linear Relationship:

- If the dependent variable decreases on the Y-axis and independent variable increases on the X-axis, then such a relationship is called a negative linear relationship.



The line of equation will be: $Y= -a_0+a_1x$

2.Feature spaces-

- is a collection of Independent and Dependent feature.

3.ML Models-

- Models are nothing but lines,surfaces,hyper surfaces in your feature space.

4.Line equation -

- Y=mX+c it tells you how X and Y are related to each other. m is slope,Y is Dependent feature,X is independent variable.When you give data to Model it will find m and C for you and collectively called as coefficients of models.

5. Collinearity-

- In linear regression model assumes that each dimensions are independent to other but in but in reality they influence each other that problem is called Collinearity.
- Before building Model we have to check relation between each independent feature to the target feature.Those features are strongly related to target feature only these features we have to consider for analysis.

6.Coefficient of Co-relation-

- To measure strength of relation between independent variable and target variable for that we use matric called Coefficient of Co-relation.

7.Covariance-

- Covariance is a measure of the relationship between two random variables. Covariance between Independent and Target feature should be strong but Covariance between independent variables should be zero.
- We tested R values for every independent feature with the Target variable and we select only those whose R values either close to - 1 or +1. and those dimensions which have close value to 0 are useless dimensions for analysis in Linear Models.

8.How actually Algorithms Woks..??

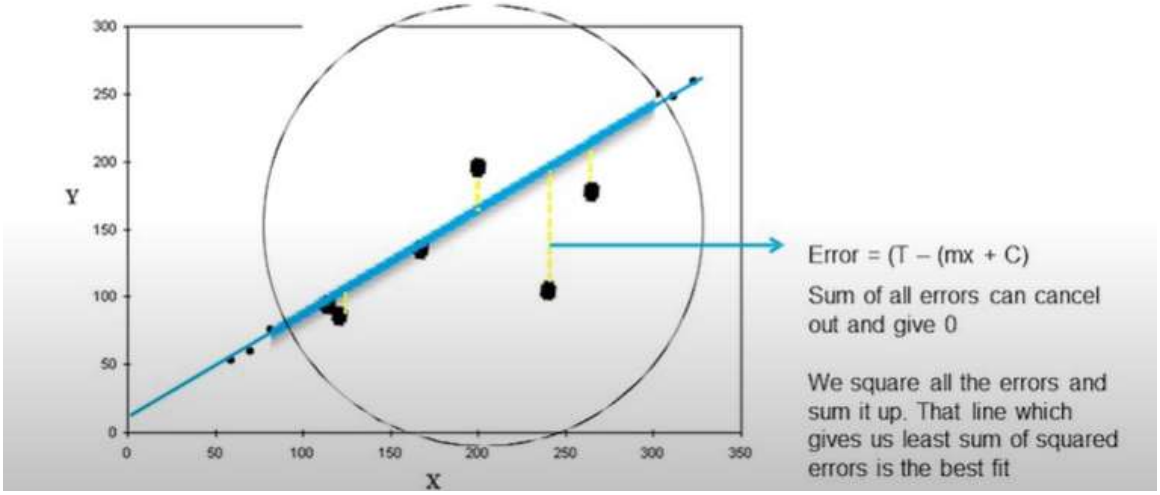
- Algorithms start with some random m and c , and it will evaluate different possible lines before its find best fit line.

9.What is Best Fit Line..?

- Amongst evaluated different possible lines the line goes through maximum data points and minimizes distance between other points and line called as Best Fit Line.
- When working with linear regression, our main goal is to find the best fit line that means the error between predicted values and actual values should be minimized. The best fit line will have the least error.
- The different values for weights or the coefficient of lines (m) gives a different line of regression, so we need to calculate the best values for m to find the best fit line, so to calculate this we use cost function.

10.Error-

- distance between other points and line called as Errors.



11.Mean Squared Errors-

- For Linear Regression, we use the **Mean Squared Error (MSE)** cost function, which is the average of squared error occurred between the predicted values and actual values.

12. Residuals-

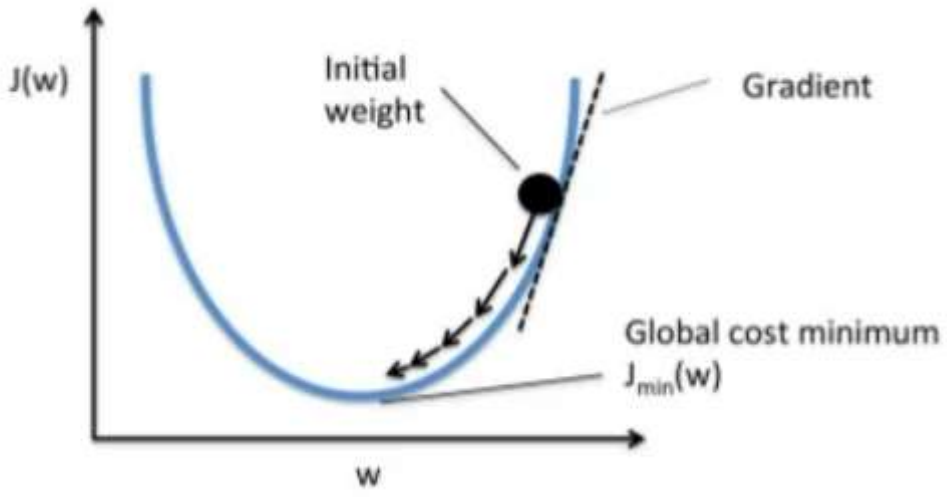
- The distance between the actual value and predicted values is called residual. If the observed points are far from the regression line, then the residual will be high, and so cost function will high. If the scatter points are close to the regression line, then the residual will be small and hence the cost function.

12.Objective Of Model-

- The algorithm will find out Best Fit Line from the infinite number of possibilities which have minimum Sum Of Squared Error with the help of process called as Gradient Descent.

13.Gradient Descent-

- Gradient descent is used to minimize the MSE by calculating the gradient of the cost function.
- A regression model uses gradient descent to update the coefficients of the line by reducing the cost function.
- It is done by a random selection of values of coefficient and then iteratively update the values to reach the minimum cost function.
- Internally makes use of partial derivatives ,partial derivative is nothing but dy/dx.For small changing in m whether it is positive or direction from random m i.e dx and with respective m changes what is behaviour of error i.e dy calculated.
- We are just doing with m, because we use mathematical term Partial derivative.we can also calculate error w.r.t C. It finds out to the next line to the next line until it reaches global minima.and this point it will get best fit line.



14.Learning Step-

- In process of jumping one to another for finding best fit line there is concept Learning Step. As jump towards your goal the steps become smaller and smaller this algorithm is called as Bold Driver algorithm and is part of Gradient Descent.

15.Coefficient Of Determination/R squared-

- Once we build a model we have to see how good the model is or how reliable the model is for that we use another matric called as Coefficient of Determination.and this is represented as R

squared. R squared ranges between 0 and 1.

- The Goodness of fit determines how the line of regression fits the set of observations. The process of finding the best model out of various models is called **optimization**. It can be achieved by R Squared Method.
- It measures the strength of the relationship between the dependent and independent variables on a scale of 0-100%.
- R-Squared is a ratio between variance explained by your model to the total variance in the datapoint
- R-Squared value is a statistical measure of how close the are to the fitted regression line.
- High R squared value is considered as good linear model.

17.Assumptions-

1.Assumption Of Linearity-

- Assumes a linear relation between the dependent/target variable and the independent /predictor variables.

2.Assumption of normality of the error distribution-

- The errors should be normally distributed across the model.

3.Small or no Collinearity between the features-

- The model assumes either little or no collinearity between the features or independent variables.

18.Advantages and Disadvantages of Linear Regression-

Linear Regression Model -

Advantages –

1. Simple to implement and easier to interpret the outputs coefficients

Disadvantages -

1. Assumes a linear relationships between dependent and independent variables. That is, it assumes there is a straight-line relationship between them
2. Outliers can have huge effects on the regression
3. Linear regression assume independence between attributes
4. Linear regression looks at a relationship between the mean of the dependent variable and the independent variables.
5. Just as the mean is not a complete description of a single variable, linear regression is not a complete description of relationships among variables
6. Boundaries are linear

19.Application Of Linear regression in Business-

- Evaluating Trends and Sales Estimates.
- Analyzing the impact of price changes.
- Assessment of risk in financial services and insurance domain.

☐ **What is Ensemble Method..??**

- An ensemble method is a technique which uses multiple independent similar or different

models/weak learners to derive an output or make some predictions. For e.g. A random forest is an ensemble of multiple decision trees. An ensemble can also be built with a combination of different models like random forest, SVM, Logistic regression etc.

- [ ] **Implementation Of Linear Regression-**

1.Reading Data-

In [1]:
```
import pandas as pd
```

In [2]:
```
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
plt.rcParams['figure.figsize']=(20.0,10.0)
```

In [21]:
```
df=pd.read_csv('C:\\Users\Abhimanyu Devadhe\Desktop\\iris.csv')
```

In [23]:
```
df.head()
```

Out[23]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa

2.Separating Data in Dependent and Independent Variable-

- Considering Sepal length as Dependant variable and Petal Length as In dependant variable.

In [29]:
```
y=df[["SepalLengthCm"]]
```

In [32]:
```
X=df[["PetalLengthCm"]]
```

In [42]:
```
X.head(2)
```

Out[42]:

	PetalLengthCm
0	1.4
1	1.4

In [43]:
```
y.head(2)
```

Out[43]:

	SepalLengthCm
0	5.1
1	4.9

3.Divide dataset X and y for Training set and Testing Set-

In [44]:
```
from sklearn.model_selection import train_test_split
```

```
In [45]: X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.3)
```

- We have to import package train_test_split from library sklearn.model_selection.
- we have to store result in four variables as X_train,X_test,y_train and y_test.

4. Build a Linear regression Model-

```
In [48]: from sklearn.linear_model import LinearRegression

In [49]: LR_Model=LinearRegression()

In [50]: LR_Model.fit(X_train,y_train)
Out[50]: LinearRegression()
```

- Import package LinearRegression from sklearn.linear_model.

5.Check Coefficient of Intercept(c) and Coefficient of slope(m)-

```
In [60]: LR_Model.intercept_
Out[60]: array([4.25898748])

In [61]: LR_Model.coef_
Out[61]: array([[0.41615084]])
```

6.Predict Output-

```
In [51]: y_Pred=LR_Model.predict(X_test)

In [57]: y_test.head(5),y_Pred[0:5]
Out[57]: (     SepalLengthCm
          81             5.5
          0              5.1
          64             5.6
          63             6.1
          148            6.2,
          array([[5.79874561],
                 [4.84159866],
                 [5.75713052],
                 [6.21489645],
                 [6.50620204]]))
```

- Predict y_Pred top on X_test.

7.Find error in prediction-

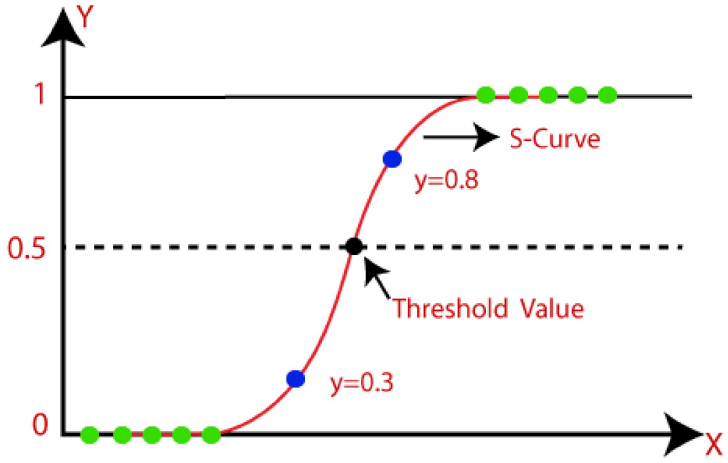
```
In [58]: from sklearn.metrics import mean_squared_error

In [59]: mean_squared_error(y_test,y_Pred)
Out[59]: 0.15251407548635823
```

- We have to find mean_squared_error for checking accuracy of model,so in that case we observe mean_squared_error is less about 0.152.

---

2. Logistic Regression-

- Logistic Regression is one of the most popular machine learning algorithms used for binary classification.
- Logistic regression is one of the most popular Machine Learning algorithms, which comes under the Supervised Learning technique. It is used for predicting the categorical dependent variable using a given set of independent variables.
- Logistic regression predicts the output of a categorical dependent variable. Therefore the outcome must be a categorical or discrete value. It can be either Yes or No, 0 or 1, true or False, etc. but instead of giving the exact value as 0 and 1, **it gives the probabilistic values which lie between 0 and 1**.
- Logistic Regression is much similar to the Linear Regression except that how they are used. Linear Regression is used for solving Regression problems, whereas **Logistic regression is used for solving the classification problems**.
- In Logistic regression, instead of fitting a regression line, we fit an "S" shaped logistic function, which predicts two maximum values (0 or 1).
- The curve from the logistic function indicates the likelihood of something such as whether the cells are cancerous or not, a mouse is obese or not based on its weight, etc.
- Logistic Regression is a significant machine learning algorithm because it has the ability to provide probabilities and classify new data using continuous and discrete datasets.
- Logistic Regression can be used to classify the observations using different types of data and can easily determine the most effective variables used for the classification. The below image is showing the logistic function:



- Note: Logistic regression uses the concept of predictive modeling as regression; therefore, it is called logistic regression, but is used to classify samples; Therefore, it falls under the classification algorithm.

- [ ] **Logistic Function (Sigmoid Function):**
  - The sigmoid function is a mathematical function used to map the predicted values to probabilities.
  - It maps any real value into another value within a range of 0 and 1.
  - The value of the logistic regression must be between 0 and 1, which cannot go beyond this

limit, so it forms a curve like the "S" form. The S-form curve is called the Sigmoid function or the logistic function.
    - In logistic regression, we use the concept of the threshold value, which defines the probability of either 0 or 1. Such as values above the threshold value tends to 1, and a value below the threshold values tends to 0.

- [ ] **Assumptions for Logistic Regression:**
    - The dependent variable must be categorical in nature.
    - The independent variable should not have multi-collinearity.

- [ ] **Logistic Regression Equation:**

- The Logistic regression equation can be obtained from the Linear Regression equation. The mathematical steps to get Logistic Regression equations are given below:
- We know the equation of the straight line can be written as:

$$y = b_0 + b_1x_1 + b_2x_2 + b_3x_3 + \cdots + b_nx_n$$

- In Logistic Regression y can be between 0 and 1 only, so for this let's divide the above equation by (1-y):

$$\frac{y}{1-y}$$; 0 for y= 0, and infinity for y=1

- But we need range between -[infinity] to +[infinity], then take logarithm of the equation it will become:

$$log\left[\frac{y}{1-y}\right] = b_0 + b_1x_1 + b_2x_2 + b_3x_3 + \cdots + b_nx_n$$

- The above equation is the final equation for Logistic Regression.

- [ ] **Type of Logistic Regression:**

- On the basis of the categories, Logistic Regression can be classified into three types:
    - **Binomial:** In binomial Logistic regression, there can be only two possible types of the dependent variables, such as 0 or 1, Pass or Fail, etc.
    - **Multinomial:** In multinomial Logistic regression, there can be 3 or more possible unordered types of the dependent variable, such as "cat", "dogs", or "sheep"
    - **Ordinal:** In ordinal Logistic regression, there can be 3 or more possible ordered types of dependent variables, such as "low", "Medium", or "High".

- [ ] **Evaluation Of Classification Model-**

- Logistic Regression employs all different sets of metrics. Here, we deal with probabilities and categorical values.
- Confusion matrix is the most crucial metric commonly used to evaluate classification models. It's quite confusing but make sure you understand it by heart. The skeleton of a confusion matrix looks like this:

	1 (Predicted)	0 (Predicted)
1		

1 (Actual)	True Positive (TP)	False Negative (FN)
0 (Actual)	False Positive (FP)	True Negative (TN)

- As you can see, the confusion matrix avoids "confusion" by measuring the actual and predicted values in a tabular format. In table above, Positive class = 1 and Negative class = 0.
  Following are the metrics we can derive from a confusion matrix:
- **Accuracy -** It determines the overall predicted accuracy of the model. It is calculated as Accuracy = (True Positives + True Negatives)/(True Positives + True Negatives + False Positives + False Negatives)
- **True Positive Rate (TPR) -** It indicates how many positive values, out of all the positive values, have been correctly predicted. The formula to calculate the true positive rate is (TP/TP + FN). Also, TPR = 1 - False Negative Rate. It is also known as Sensitivity or Recall.
- **False Positive Rate (FPR) -** It indicates how many negative values, out of all the negative values, have been incorrectly predicted. The formula to calculate the false positive rate is (FP/FP + TN). Also, FPR = 1 - True Negative Rate.
- **True Negative Rate (TNR) -** It indicates how many negative values, out of all the negative values, have been correctly predicted. The formula to calculate the true negative rate is (TN/TN + FP). It is also known as Specificity.
- **False Negative Rate (FNR) -** It indicates how many positive values, out of all the positive values, have been incorrectly predicted. The formula to calculate false negative rate is (FN/FN + TP).
- **Precision:** It indicates how many values, out of all the predicted positive values, are actually positive. It is formulated as:(TP / TP + FP).
- **F1 Score:** F score is the harmonic mean of precision and recall. It lies between 0 and 1. Higher the value, better the model. It is formulated as 2((precision*recall) / (precision+recall)).
- **ROC curve-**ROC and AUC curve basically used for to fix threshold value and Plot in between True Positive Rate(Y axis) and False Positive Rate(X axis).With tunning threshold values we obtain datapoints and combinedly plotted on graph.Threshold values are fix w.r.t to domain and required severity of TPR and FPR only.
- **AUC -**Area Under curve,Area under ROC curve called as AUC And the more the area under the curve the better the model is yours.

- [ ] **Why Is Logistic Regression Called "Logistic Regression" and Not "Logistic Classification"?**

- Logistic regression uses the concept of predictive modeling as regression; therefore, it is called logistic regression, but is used to classify samples; Therefore, it falls under the classification algorithm.
- Logistic regression borrows its name from the logistic function and linear regression algorithm. Linear regression does not work well with classification problems.
- Logistic regression uses the logistic function which squashes the output range between 0 and 1 and makes use of hypothesis function of the linear regression algorithm.

1.and makes use of hypothesis function of the linear regression algorithm.

Implementation Of Logistic Regression-

1.Import Data-

In [1]:
```
import pandas as pd
import numpy as np
```

In [3]:
```
df= pd.read_csv("C:\\Users\Abhimanyu Devadhe\Desktop\\HR_comma_sep.csv")
```

In [4]:
```
df
```

Out[4]:

	satisfaction_level	last_evaluation	number_project	average_montly_hours	time_spend_company	Work_accident	left	p
0	0.38	0.53	2	157	3	0	1	
1	0.80	0.86	5	262	6	0	1	
2	0.11	0.88	7	272	4	0	1	

2. Convert Categorical feature into Numerical feature-

- with the help of get_dummies function we can did 'one hot encoding' to convert categorical features in numerical feature.

In [17]:
```
df1=pd.get_dummies(df.salary)
```

In [19]:
```
df2=pd.get_dummies(df.sales)
```

In [22]:
```
df3=pd.concat([df,df1,df2],axis=1)
```

In [28]:
```
df_Final=df3.drop(["sales","salary"],axis=1)
```

In [29]:
```
df_Final
```

Out[29]:

	satisfaction_level	last_evaluation	number_project	average_montly_hours	time_spend_company	Work_accident	left
0	0.38	0.53	2	157	3	0	1
1	0.80	0.86	5	262	6	0	1

- There is salary and sales are categorical features in our dataset we are converted into numerical feature.

3.Split dataset in X and y as Independent and Dependent features-

In [31]:
```
X=df_Final.drop(["left"],axis=1)
```

In [33]:
```
Y=df_Final[["left"]]
```

- left column is considered as dependent feature and others are independent feature.

4.Split X and Y for Training and Testing-

In [41]:
```
from sklearn.model_selection import train_test_split
```

In [42]:
```
X_train,X_test,y_train,y_test=train_test_split(X,Y,test_size=0.2)
```

5. Build Model-

```
In [44]: from sklearn.linear_model import LogisticRegression

In [46]: LR_Model=LogisticRegression()

In [47]: LR_Model.fit(X_train,y_train)

         C:\Users\Abhimanyu Devadhe\anaconda3\lib\site-packages\sklearn\utils\validation.py
         y was passed when a 1d array was expected. Please change the shape of y to (n sam
```

- from sklearn.linear_model import LogisticRegression.

6.Predict y_test w.r.t X_test-

```
In [49]: y_pred=Model.predict(X_test)

In [56]: len(y_pred)
Out[56]: 3000

In [55]: len(y_test)
Out[55]: 3000
```

7.Evaluation of Model-

i)Confusion Matrix-

```
In [58]: from sklearn.metrics import confusion_matrix,precision_score,recall_score,f1_score,accuracy_score

In [59]: cm=confusion_matrix(y_test,y_pred)

In [60]: cm
Out[60]: array([[2099,  169],
                [ 465,  267]], dtype=int64)
```

ii)accuracy_score-

```
In [61]: accuracy_score(y_test,y_pred)
Out[61]: 0.7886666666666666
```

iii)precision_score-

```
In [62]: precision_score(y_test,y_pred)
Out[62]: 0.6123853211009175
```

iv)recall_score-

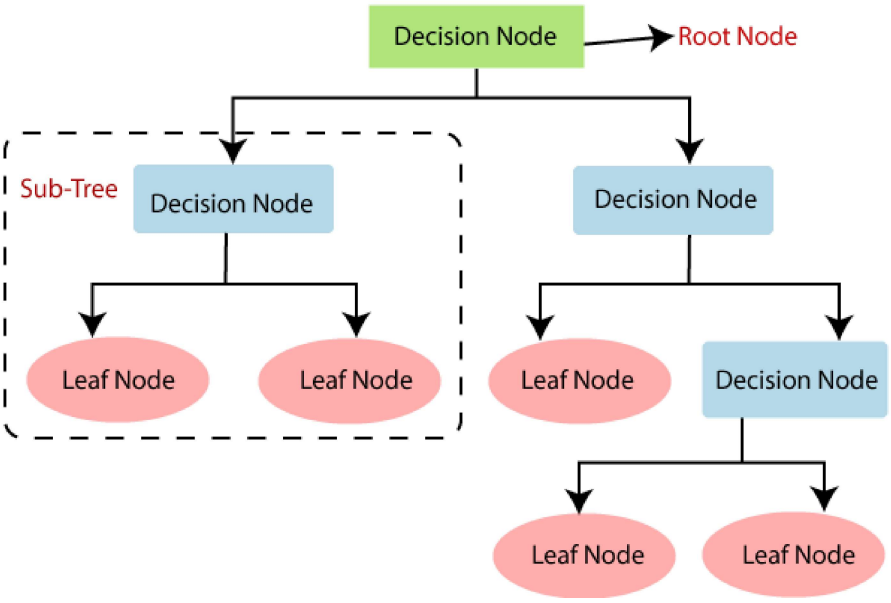
```
In [63]: recall_score(y_test,y_pred)
Out[63]: 0.36475409836065575
```

v)f1_score-

```
In [64]: f1_score(y_test,y_pred)
Out[64]: 0.45719178082191786
```

3.Desicion Tree-

- Decision Tree is a **Supervised learning technique** that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where **internal nodes represent the features of a dataset, branches represent the decision rules** and **each leaf node represents the outcome.**
- In a Decision tree, there are two nodes, which are the **Decision Node** and **Leaf Node.** Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches.
- The decisions or the test are performed on the basis of features of the given dataset.
- ***It is a graphical representation for getting all the possible solutions to a problem/decision based on given conditions.***
- It is called a decision tree because, similar to a tree, it starts with the root node, which expands on further branches and constructs a tree-like structure.
- In order to build a tree, we use the **CART algorithm,** which stands for **Classification and Regression Tree algorithm.**
- A decision tree simply asks a question, and based on the answer (Yes/No), it further split the tree into subtrees.
- Below diagram explains the general structure of a decision tree:
- Note: A decision tree can contain categorical data (YES/NO) as well as numeric data.



- [ ] **Why use Decision Trees?**

- There are various algorithms in Machine learning, so choosing the best algorithm for the given dataset and problem is the main point to remember while creating a machine learning model. Below are the two reasons for using the Decision tree:

1. Decision Trees usually mimic human thinking ability while making a decision, so it is easy to

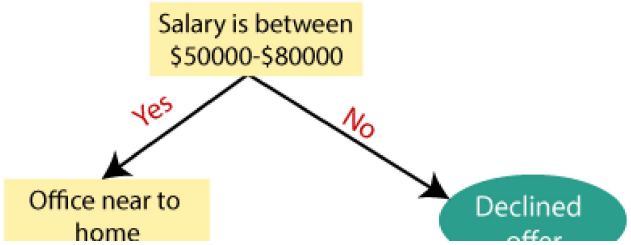
1. Decision trees usually mimic human thinking ability while making a decision, so it is easy to understand.
2. The logic behind the decision tree can be easily understood because it shows a tree-like structure.

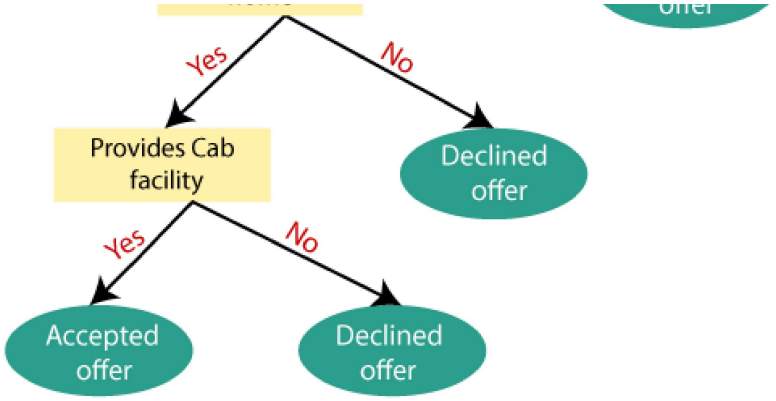
- [ ] **Decision Tree Terminologies**

1. **Root Node:** Root node is from where the decision tree starts. It represents the entire dataset, which further gets divided into two or more homogeneous sets.
2. **Leaf Node:** Leaf nodes are the final output node, and the tree cannot be segregated further after getting a leaf node.
3. **Splitting:** Splitting is the process of dividing the decision node/root node into sub-nodes according to the given conditions.
4. **Branch/Sub Tree:** A tree formed by splitting the tree.
5. **Pruning:** Pruning is the process of removing the unwanted branches from the tree.
6. **Parent/Child node:** The root node of the tree is called the parent node, and other nodes are called the child nodes.

- [ ] **How does the Decision Tree algorithm Work?**

- In a decision tree, for predicting the class of the given dataset, the algorithm starts from the root node of the tree. This algorithm compares the values of root attribute with the record (real dataset) attribute and, based on the comparison, follows the branch and jumps to the next node.
- For the next node, the algorithm again compares the attribute value with the other sub-nodes and move further. It continues the process until it reaches the leaf node of the tree. The complete process can be better understood using the below algorithm:
- **Step-1:** Begin the tree with the root node, says S, which contains the complete dataset.
- **Step-2:** Find the best attribute in the dataset using **Attribute Selection Measure (ASM).**
- **Step-3:** Divide the S into subsets that contains possible values for the best attributes.
- **Step-4:** Generate the decision tree node, which contains the best attribute.
- **Step-5:** Recursively make new decision trees using the subsets of the dataset created in step -3. Continue this process until a stage is reached where you cannot further classify the nodes and called the final node as a leaf node.
- **Example:** Suppose there is a candidate who has a job offer and wants to decide whether he should accept the offer or Not. So, to solve this problem, the decision tree starts with the root node (Salary attribute by ASM). The root node splits further into the next decision node (distance from the office) and one leaf node based on the corresponding labels. The next decision node further gets split into one decision node (Cab facility) and one leaf node. Finally, the decision node splits into two leaf nodes (Accepted offers and Declined offer). Consider the below diagram:





- [ ] **Attribute Selection Measures**

- While implementing a Decision tree, the main issue arises that how to select the best attribute for the root node and for sub-nodes. So, to solve such problems there is a technique which is called as **Attribute selection measure or ASM.** By this measurement, we can easily select the best attribute for the nodes of the tree. There are two popular techniques for ASM, which are:

A) Information Gain

B) Gini Index

1. Information Gain:

- Information gain is the measurement of changes in entropy after the segmentation of a dataset based on an attribute.
- It calculates how much information a feature provides us about a class.
- According to the value of information gain, we split the node and build the decision tree.
- A decision tree algorithm always tries to maximize the value of information gain, and a node/attribute having the highest information gain is split first. It can be calculated using the below formula:

Information Gain= Entropy(S)- [(Weighted Avg) *Entropy(each feature)

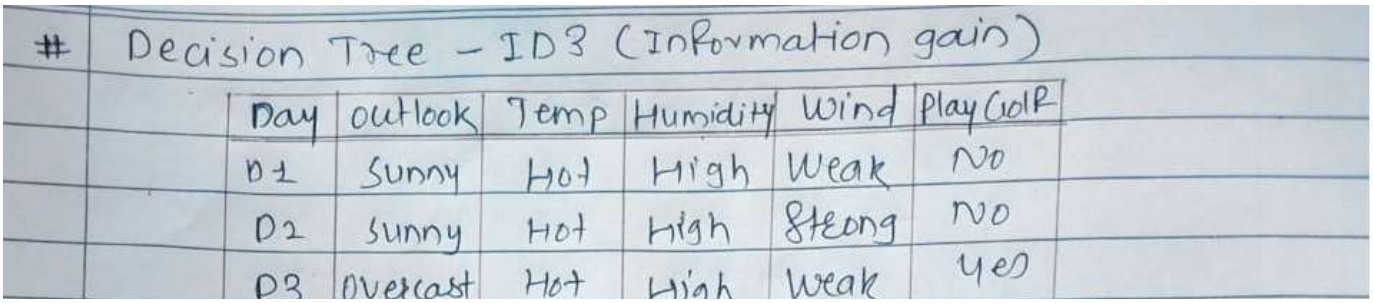
- **Entropy:** Entropy is a metric to measure the impurity in a given attribute. It specifies randomness in data. Entropy can be calculated as:

```
Entropy(s)= -P(yes)log2 P(yes)- P(no) log2 P(no)
```

Where,

- **S= Total number of samples**
- **P(yes)= probability of yes**
- **P(no)= probability of no**

- [ ] **Analytical Solution for finding Entropy for Dataset and Individual attribute.**



Decision Tree - ID3 (Information gain)

Day	outlook	Temp	Humidity	Wind	PlayGolf
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Steong	No
D3	Overcast	Hot	High	Weak	Yes

D4	Rain	Mild	High	Weak	yes
D5	Rain	Cool	Nomal	Weak	yes
D6	Rain	Cool	Normal	Steong	No
D7	Overcast	Cool	Normal	Steong	yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	yes
D10	Rain	Mild	Normal	Weak	yes
D11	Sunny	Mild	Normal	Steong	yes
D12	Overcast	Mild	High	Steong	yes
D13	Overcast	Hot	Normal	Weak	yes
D14	Rain	Mild	High	Steong	No

- FirsHy we need to understand which attribute giving maximum information out of available attributes
- So in this case we have four attributes, Out four attributes we need to calculate information gain of every attribute.
- The attribute having maximum information gain we will consider as "root Node". and then we will start to building tree.
-

① Attibute → outlook

values (outlook) = sunny, overcast, Rain

- IF you want to calculate Information gain of attribute then we have to calculate Entropy of whole dataset and Entropy of values from

classmate

individual attribute.

(A) Entropy = $-p/(p+n)\log_2(p/(p+n)) - n/(p+n)\log_2(n/(p+n))$

where, p = positive, n = negative

i) Entrop of Entire Dataset(S) —

S=(9 LP), 5 (N) → play Golf

$$S = -\frac{9}{14}\log_2\frac{9}{14} - \frac{5}{14}\log_2\frac{5}{14}$$

14 14 14 14

$S = 0.94$

ii) Entropy of sunny =
(2(P), 3(n))

$$\text{Entropy\_sunny} = -\frac{2}{5}\log_2\frac{2}{5} - \frac{3}{5}\log_2\frac{3}{5}$$

Entrop_sunny = 0.971

iii) Entropy of overcast –
(4(P), 0(n))

$$\text{Entrop\_overcast} = -\frac{4}{4}\log_2\frac{4}{4} - \frac{0}{4}\log_2\frac{0}{4}$$

Entroy_overcast = 0

iv) Entropy of Rain –
(3(P), 2(n))

$$\text{Entropy\_Rain} = -\frac{3}{5}\log_2\frac{3}{5} - \frac{2}{5}\log_2\frac{2}{5}$$

$= 0.971$

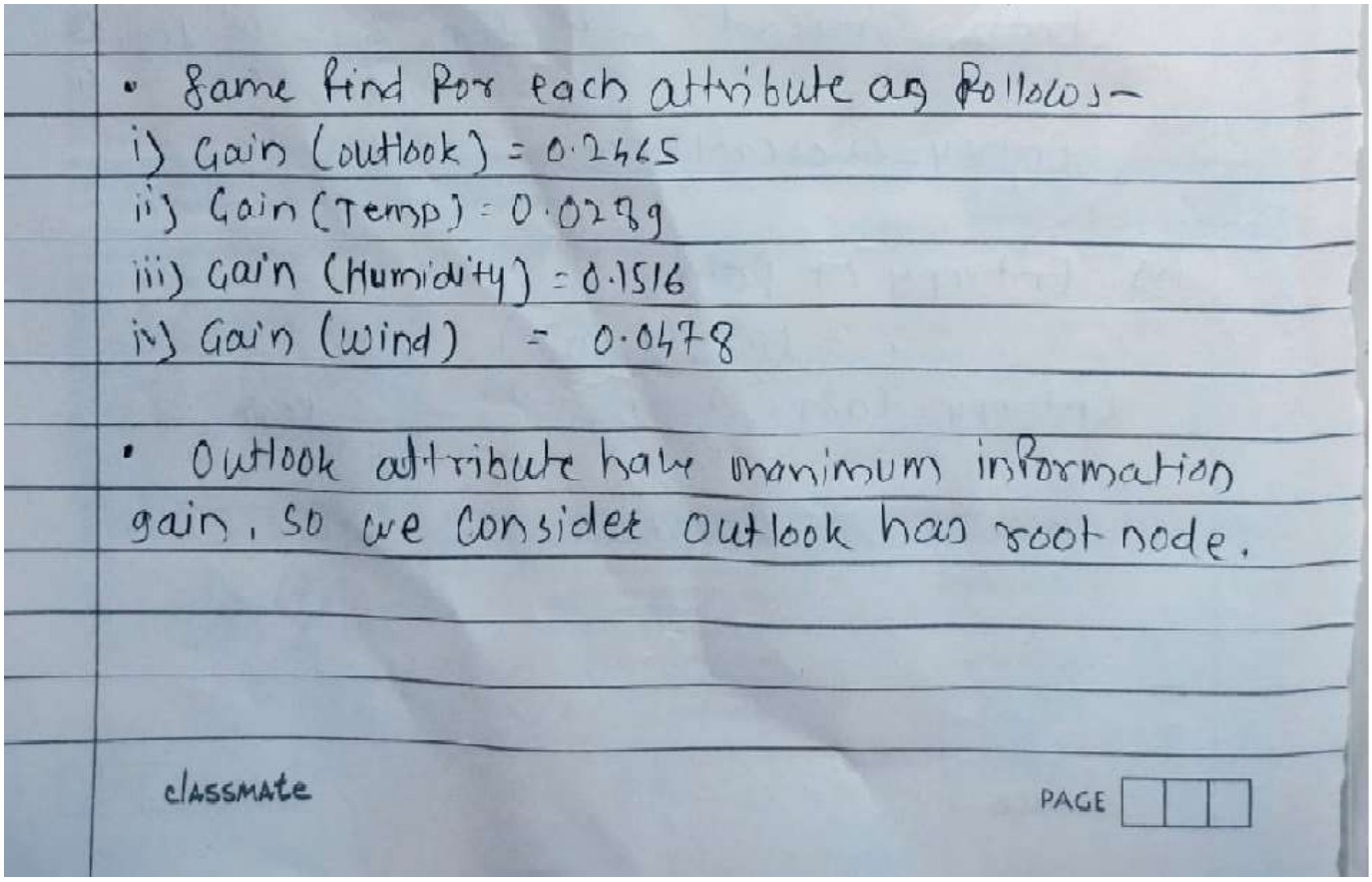
(B) Information Gain →

Gain(outlook) = Entropy of whole Dataset − [(P_sunny + n_sunny)/p+n × Entropy_sunny + (P_overcast + n_overcast)/p+n × Entropy_overcast + (P_rain + n_rain)/p+n × Entropy_rain]

$$= 0.94 - \left(\frac{5}{14} \times 0.971 + \frac{4}{14} \times 0 + \frac{5}{14} \times 0.971\right)$$

$= 0.94 - 0.6935$

i) Gain = 0.2465 --- gain for outlook



2. Gini Index:

- Gini index is a measure of impurity or purity used while creating a decision tree in the CART(Classification and Regression Tree) algorithm.
- An attribute with the low Gini index should be preferred as compared to the high Gini index.
- It only creates binary splits, and the CART algorithm uses the Gini index to create binary splits.
- Gini index can be calculated using the below formula:

```
Gini Index= 1- ∑jPj2
```

- [ ] **Pruning: Getting an Optimal Decision tree**

- *Pruning is a process of deleting the unnecessary nodes from a tree in order to get the optimal decision tree.*
- A too-large tree increases the risk of overfitting, and a small tree may not capture all the important features of the dataset. Therefore, a technique that decreases the size of the learning tree without reducing accuracy is known as Pruning. There are mainly two types of tree **pruning** technology used:

1. Cost Complexity Pruning

2.Reduced Error Pruning.

- **Advantages of Pruning a Decision Tree**

1.Pruning reduces the complexity of the final tree and thereby reduces overfitting.

2.Explainability — Pruned trees are shorter, simpler, and easier to explain.

- **Limitations of Pruning**

1.Similar to Lasso regularization, there is no real disadvantage. However, pruning does come with a high computational cost.

- [ ] **Advantages of the Decision Tree**

- It is simple to understand as it follows the same process which a human follow while making any decision in real-life.
- It can be very useful for solving decision-related problems.
- It helps to think about all the possible outcomes for a problem.
- There is less requirement of data cleaning compared to other algorithms.

- [ ] **Disadvantages of the Decision Tree**

- The decision tree contains lots of layers, which makes it complex.
- It may have an overfitting issue, which can be resolved using the **Random Forest algorithm.**
- For more class labels, the computational complexity of the decision tree may increase.

- [ ] **Implementation Of Decision Tree-**

1.Load Dataset-

```
In [3]: import seaborn as sns
```

```
In [4]: df=pd.read_csv("golf-dataset.csv")
```

```
In [5]: df
```

Out[5]:

	Outlook	Temp	Humidity	Windy	Play Golf
0	Rainy	Hot	High	False	No
1	Rainy	Hot	High	True	No

- golf dataset is commonly used dataset for decision tree.

2.Check datatype of features-

```
In [7]: df.dtypes
```

```
Out[7]: Outlook      object
        Temp         object
        Humidity     object
        Windy          bool
        Play Golf    object
        dtype: object
```

```
In [24]: df1=df["Windy"].astype("str")
```

```
In [28]: df2=df.drop(["Windy"],axis=1)
```

```
In [29]: df3=pd.concat([df1,df2],axis=1)
```

```
In [32]: df3.dtypes
```

```
Out[32]: Windy        object
         Outlook      object
         Temp         object
         Humidity     object
         Play Golf    object
         dtype: object
```

- datatype of should be string for encoding of features.

3.Split dataset into X and Y and Encode Features-

```
In [45]: X=df3.drop(["Play Golf"],axis=1)   #Dependent feature is removed.
```

```
In [46]: X_Final=pd.get_dummies(X)      #One hot encoding for Independent Features.

In [48]: Y=df3[["Play Golf"]]        #Considering Play Gold as Target Feature

In [55]: Y_Final=Y["Play Golf"].map({"Yes":1,"No":0}) #Encoding for Target feature
```

- play golf feature is our target feature and others are independent feature.

4.Split dataset for Training and Testing-

```
In [58]: X_train,X_test,y_Train,y_test=train_test_split(X_Final,Y_Final,test_size=0.15)
```

5. Build Model-

A) Build Model on Information Gain ASM-

```
In [59]: from sklearn.tree import DecisionTreeClassifier

In [69]: Entr_Model=DecisionTreeClassifier(criterion="entropy")

In [70]: DT_Model=Model.fit(X_train,y_Train)
```

- use criterion as entropy.

B) Build Model on Gini Index-

```
In [83]: Gini_Model=DecisionTreeClassifier(criterion="gini")

In [84]: DT_Model2=Gini_Model.fit(X_train,y_Train)
```

6.Predict Play golf w.r.t X_test for both model-

```
In [92]: y_Pred1=DT_Model1.predict(X_test)

In [93]: y_Pred2=DT_Model2.predict(X_test)
```

7.Validation of Models-

A) For Information Gain Model-

```
In [100]: confusion_matrix(y_test,y_Pred1)
Out[100]: array([[1, 1],
                 [0, 1]], dtype=int64)

In [101]: precision_score(y_test,y_Pred1)
Out[101]: 0.5

In [102]: accuracy_score(y_test,y_Pred1)
```

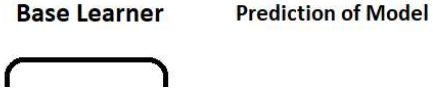
```
Out[102]: 0.6666666666666666

In [103]: recall_score(y_test,y_Pred1)

Out[103]: 1.0

In [104]: f1_score(y_test,y_Pred1)

Out[104]: 0.6666666666666666
```

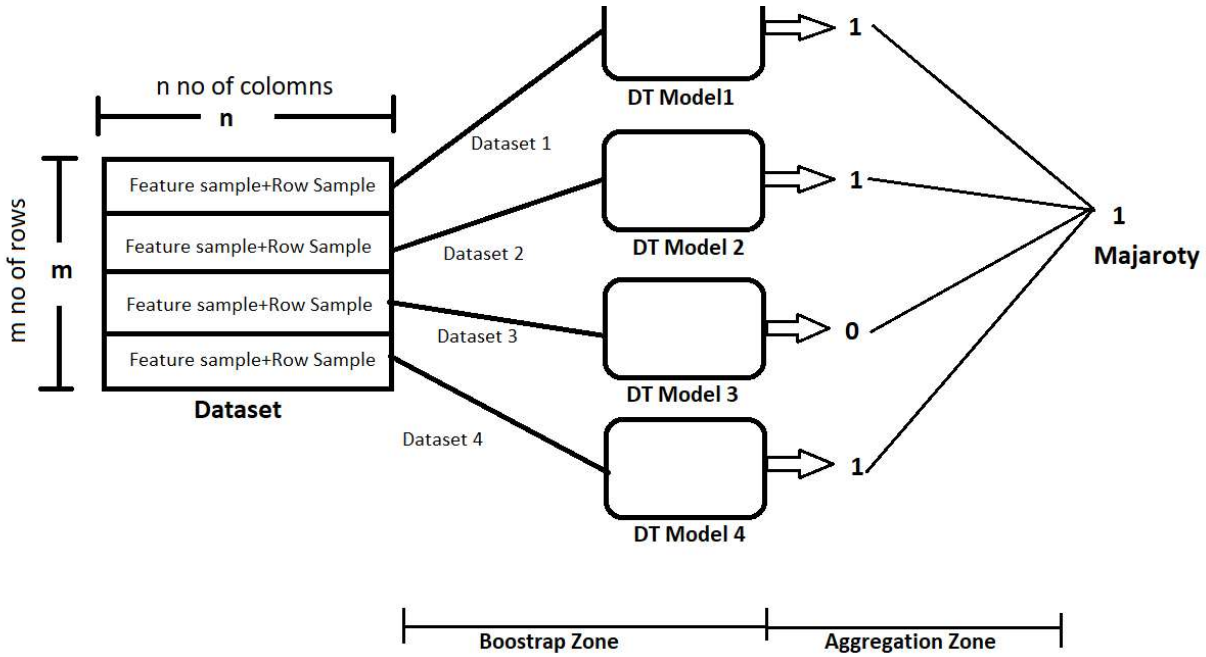
B) For Gini Index Model-

```
In [105]: confusion_matrix(y_test,y_Pred2)

Out[105]: array([[1, 1],
                 [0, 1]], dtype=int64)

In [106]: precision_score(y_test,y_Pred2)

Out[106]: 0.5

In [107]: accuracy_score(y_test,y_Pred2)

Out[107]: 0.6666666666666666

In [108]: recall_score(y_test,y_Pred2)

Out[108]: 1.0

In [110]: f1_score(y_test,y_Pred2)

Out[110]: 0.6666666666666666
```

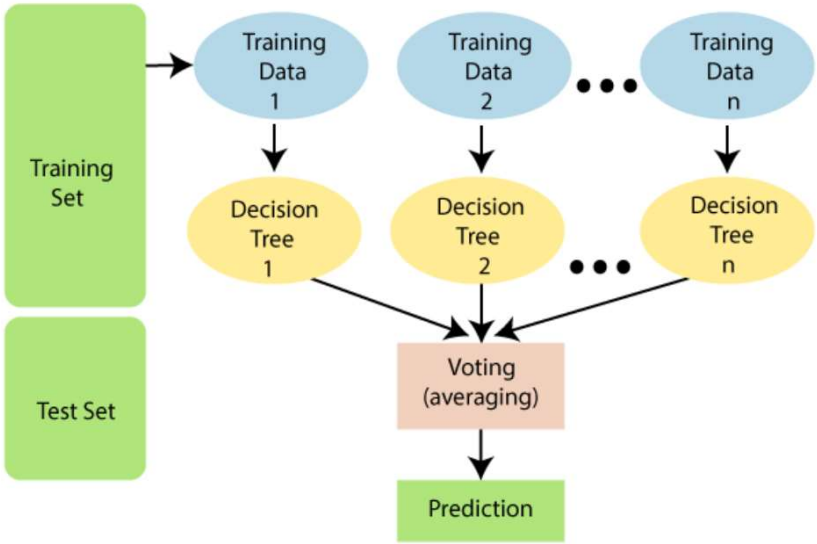
4.<u>Random Forest</u>-

- Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of **ensemble learning,** which is a process of *combining multiple classifiers to solve a complex problem and to improve the performance of the model.*
- As the name suggests, ***"Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset."*** Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output.
- **The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting.**
- Random Forest is basically based on ensemble technic called as bagging(Bootstrap Aggregation).





- Observer above scenario,we have one dataset have n no of columns as m no of rows.
- And we have 4 no of models which are basically Decision tree.
- From dataset we will randomly pick some no of features and rows for each individual model with the help of 'Row Sampling with Replacement' method.
- With the help of provided dataset points model has trained and ready for prediction.
- Then we provide test data for testing of model and model predicts 0 and 1 w.r.t test data. (assuming Binary classification).
- For final prediction we use Aggregation method,in which our final prediction will be majority prediction of models.Majority is used for classification model.
- When we use RF for regression model use Mean or median at the place of majority.
- The below diagram explains the working of the Random Forest algorithm:



- [ ] **Assumptions for Random Forest**

- Since the random forest combines multiple trees to predict the class of the dataset, it is possible that some decision trees may predict the correct output, while others may not. But together, all the trees predict the correct output. Therefore, below are two assumptions for a better Random forest classifier:
    - There should be some actual values in the feature variable of the dataset so that the classifier can predict accurate results rather than a guessed result.

- The predictions from each tree must have very low correlations.

- [ ] **Why use Random Forest?**

- Below are some points that explain why we should use the Random Forest algorithm:
  - It takes less training time as compared to other algorithms.
  - It predicts output with high accuracy, even for the large dataset it runs efficiently.
  - It can also maintain accuracy when a large proportion of data is missing.

- [ ] **How does Random Forest algorithm work?**

- Random Forest works in two-phase first is to create the random forest by combining N decision tree, and second is to make predictions for each tree created in the first phase.
- The Working process can be explained in the below steps and diagram:

Step-1: Select random K data points from the training set.

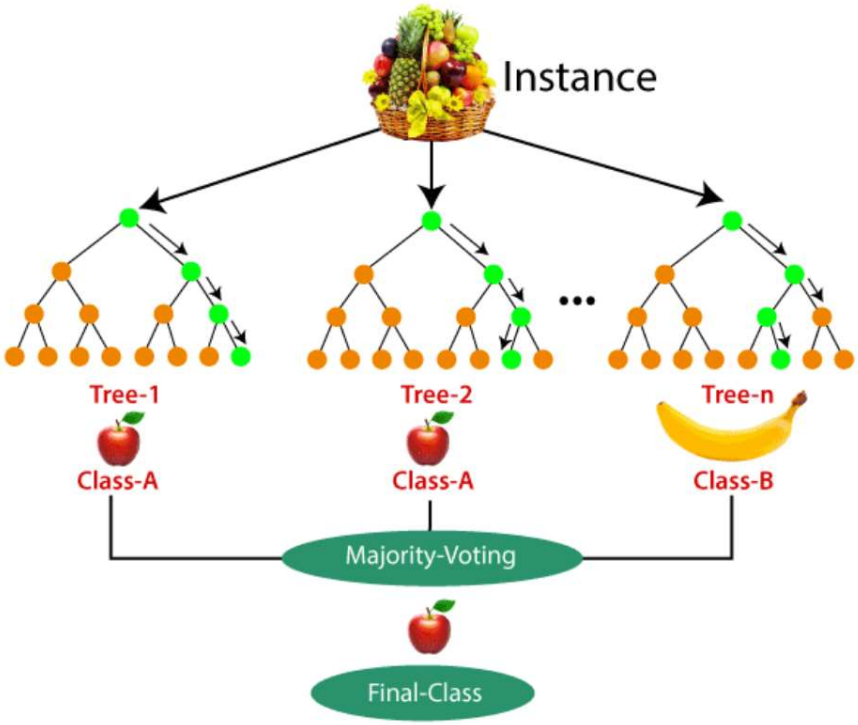
Step-2: Build the decision trees associated with the selected data points (Subsets).

Step-3: Choose the number N for decision trees that you want to build.

Step-4: Repeat Step 1 & 2.

Step-5: For new data points, find the predictions of each decision tree, and assign the new data points to the category that wins the majority votes.

- The working of the algorithm can be better understood by the below example:
- **Example:** Suppose there is a dataset that contains multiple fruit images. So, this dataset is given to the Random forest classifier. The dataset is divided into subsets and given to each decision tree. During the training phase, each decision tree produces a prediction result, and when a new data point occurs, then based on the majority of results, the Random Forest classifier predicts the final decision. Consider the below image:



- [ ] **Applications of Random Forest**

- There are mainly four sectors where Random forest mostly used:

1. **Banking:** Banking sector mostly uses this algorithm for the identification of loan risk.
2. **Medicine:** With the help of this algorithm, disease trends and risks of the disease can be identified.
3. **Land Use:** We can identify the areas of similar land use by this algorithm.

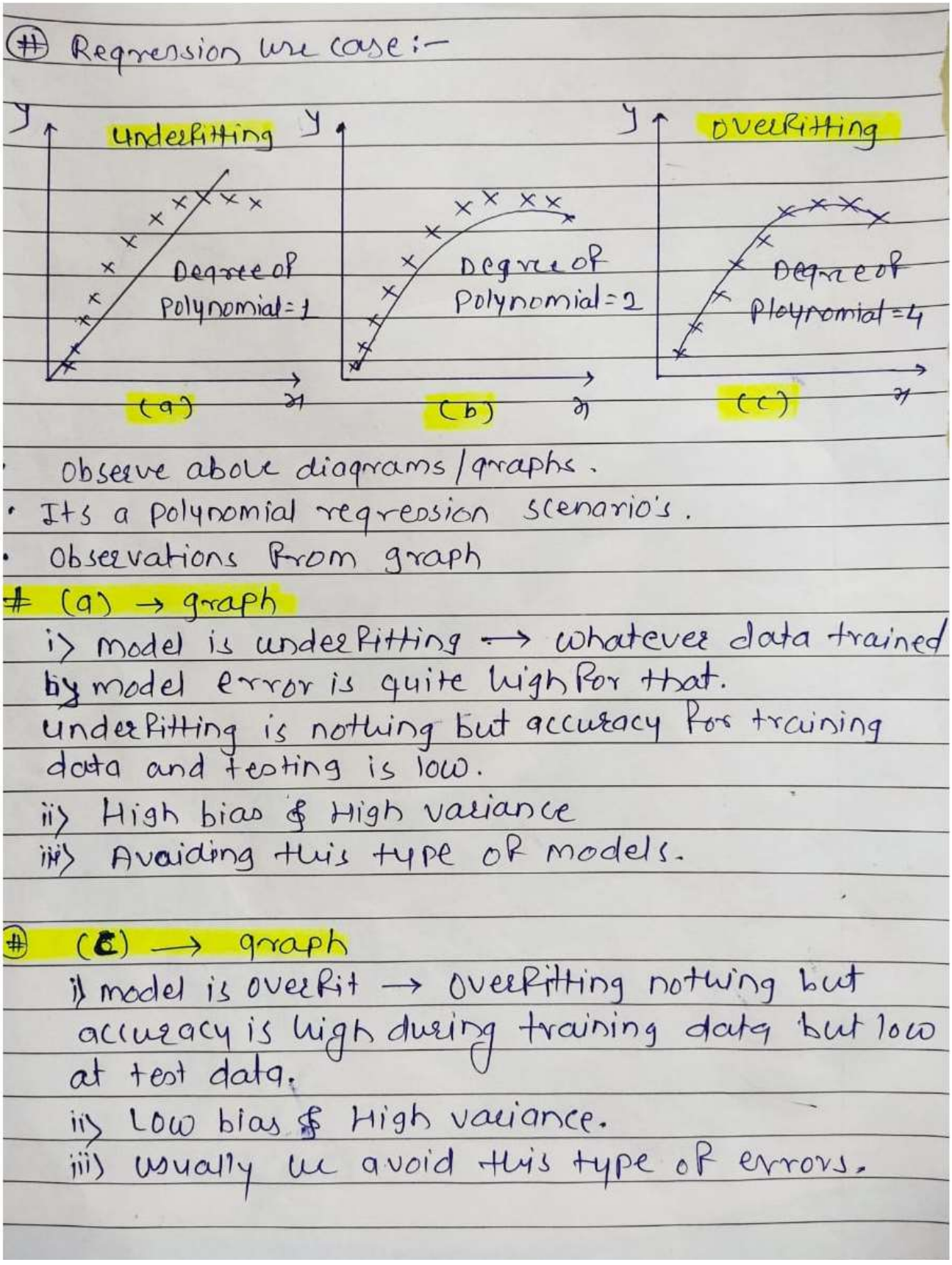
4. **Marketing:** Marketing trends can be identified using this algorithm.

- [ ] **Advantages of Random Forest**

- Random Forest is capable of performing both Classification and Regression tasks.
- It is capable of handling large datasets with high dimensionality.
- It enhances the accuracy of the model and prevents the overfitting issue.

- [ ] **Disadvantages of Random Forest**

- Although random forest can be used for both classification and regression tasks, it is not more suitable for Regression tasks.



(#) (b) → graph

i) low bias & low variance

ii) It is generalized model, usually we are expecting this type of model.

iii) model should give some accuracy for training dataset as well as testing dataset.

(#) Bias → Bias defines as Errors of the training dataset.

- For generalized ML model, model should be low bias

(#) Variance → Variance defines as Errors of the test dataset.

- For generalized ML model, model should be low variance.

(#) Decision Tree :-

- Decisiontree disguised row of tree itself completely to his depths, Its takes all features and then its starts splitting to its complete depth.
- In DT accuracy at training datase is too high but at testing it will be low, so DT model have overfitting issue and have low bias and high variance.

(#) Random Forest :-

- In Random Forest we have multiple DT in parallel.

- In Random Forest, we are using multiple DT and have low bias and high variance each.
- In RF we have use Bootstrap Aggregation technique and have multiple DT in parallel.
- So high variance converted into low variance
- RF have low bias and low variance.

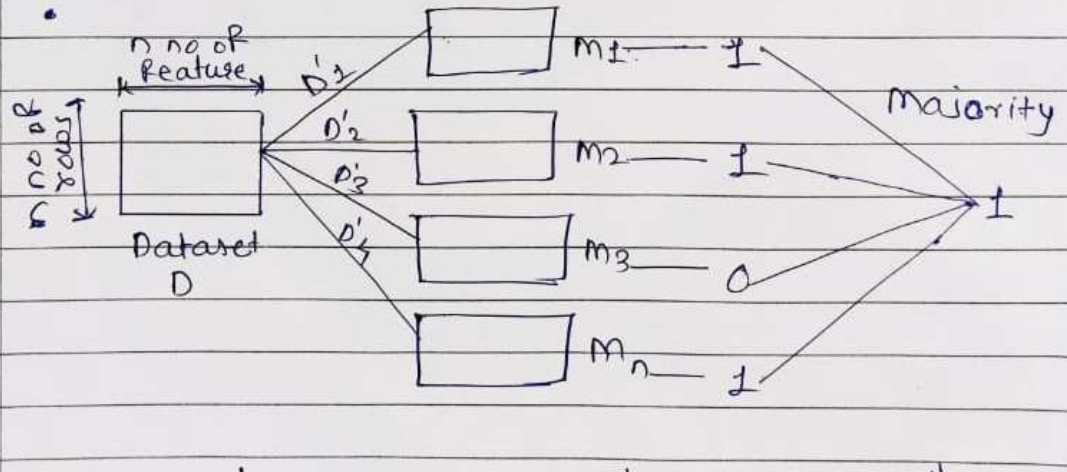
Ensemble Techniques :-

i) Ensembles → Ensembles means combining multiple ML models for final prediction.

- There are two techniques in Ensembles
  A] Bagging & B] Boosting.

A] Bagging :-

- Also called as bootstrap Aggregation.
- This technique used in Random Forest.



- We have a dataset D and with the help of 'Row sampling with Replacement' we have provide random data to each model.
- Model will train and give prediction in Binary form (assuming binary classification)
- Then we find majority from that and this will be our final prediction

---

- [ ] **Cross Validation in ML-**

- In machine learning, we couldn't fit the model on the training data and can't say that the model will work accurately for the real data. For this, we must assure that our model got the correct patterns from the data, and it is not getting up too much noise. For this purpose, we use the cross-validation technique.
- Cross-validation is a technique in which we train our model using the subset of the data-set and then evaluate using the complementary subset of the data-set.
- **Methods of Cross Validation-**

1.Validation

- In this method, we perform training on the 50% of the given data-set and rest 50% is used for

In this method, we perform training on the 50% of the given data-set and rest 50% is used for the testing purpose. The major drawback of this method is that we perform training on the 50% of the dataset, it may possible that the remaining 50% of the data contains some important information which we are leaving while training our model i.e higher bias.

2.LOOCV (Leave One Out Cross Validation)

- In this method, we perform training on the whole data-set but leaves only one data-point of the available data-set and then iterates for each data-point. It has some advantages as well as disadvantages also.An advantage of using this method is that we make use of all data points and hence it is low bias.The major drawback of this method is that it leads to higher variation in the testing model as we are testing against one data point. If the data point is an outlier it can lead to higher variation. Another drawback is it takes a lot of execution time as it iterates over 'the number of data points' times.

3.K-Fold Cross Validation-

- In this method, we split the data-set into k number of subsets(known as folds) then we perform training on the all the subsets but leave one(k-1) subset for the evaluation of the trained model. In this method, we iterate k times with a different subset reserved for testing purpose each time.
- Note-It is always suggested that the value of k should be 10 as the lower value of k is takes towards validation and higher value of k leads to LOOCV method.
- **Example-**The diagram below shows an example of the training subsets and evaluation subsets generated in k-fold cross-validation. Here, we have total 25 instances. In first iteration we use the first 20 percent of data for evaluation, and the remaining 80 percent for training([1-5] testing and [5-25] training) while in the second iteration we use the second subset of 20 percent for evaluation, and the remaining three subsets of the data for training([5-10] testing and [1-5 and 10-25] training), and so on.


```
Total instances: 25
Value of k     : 5

No. Iteration              Training set observations                    Testing set observations
 1      [ 5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24]   [0 1 2 3 4]
 2      [ 0  1  2  3  4 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24]   [5 6 7 8 9]
 3      [ 0  1  2  3  4  5  6  7  8  9 15 16 17 18 19 20 21 22 23 24]   [10 11 12 13 14]
```

```
4      [ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 20 21 22 23 24]   [15 16 17 18 19]
5      [ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19]   [20 21 22 23 24]
```

Comparison of train/test split to cross-validation

- **Advantages of train/test split:**
  1. This runs K times faster than Leave One Out cross-validation because K-fold cross-validation repeats the train/test split K-times.
  2. Simpler to examine the detailed results of the testing process.
- **Advantages of cross-validation:**
  1. More accurate estimate of out-of-sample accuracy.
  2. More "efficient" use of data as every observation is used for both training and testing.
- **Python code for k fold cross-validation-**

```
# as required packages are not found.
# importing cross-validation from sklearn package.
  from sklearn import cross_validation
# value of K is 10.
  data = cross_validation.KFold(len(train_set), n_folds=10, indices=False)
```

- [ ] **Interview Notes-**