



**SIDDHARTHA INSTITUTE OF TECHNOLOGY & SCIENCES**  
**(UGC - AUTONOMOUS)**

(Approved by AICTE, New Delhi & Affiliated to JNTUH, Hyderabad)  
Accredited by \*NBA, NAAC with 'A+' Grade, \*nirf Ranked and ISO Certified Institution  
Narapally, Korremula Road, Ghatkesar, Medchal-Malkajiri (Dist), Telangana-500 008



**Organizing By**

Department of CSE (AIML, DS, CS, IoT & SE)

*In Association with "Brain O Vision"*  
**and Approved by AICTE**



*One Week National Level Student Workshop*  
**on**  
*"Data Science using Python"*

**AGENDA & CURRICULAM**

- ▶ Day 1 - Introduction to Data Science and Python Basics
- ▶ Day 2 - Data Manipulation and Visualization
- ▶ Day 3 - Data Analysis with Python
- ▶ Day 4 - Introduction to Machine Learning
- ▶ Day 5 - Practical Machine Learning in Python



**Chief Guest**

**Dr. Buddha Chandrasekhar**  
Chief Coordinating Officer - AICTE



**19<sup>th</sup> – 23<sup>rd</sup> FEB 2024**  
**7.00 PM - 8.30 PM**

**Registration Fee: ₹09/- ONLY**

<https://brainovision.in/student-workshop-2024/>

**Chief Patron**

**Sri. C. R. Jagadish Rao**  
Chairman, SITS

**Patron**

**Dr. M. Janardhan**  
Principal, SITS

**Co- Patron**

**Dr. Srinivas Bachu**  
Professor, HOD-ECE & Dean IQAC

**Convener**

**Mrs. I. Vasantha Kumari**  
Assoc. Prof., & HOD-CSE(CS, IoT & SE)

**Co-ordinators**

**Mr. S. Venkata Satya Krishna,**  
Assoc. Prof., & HOD-CSE(AIML & DS)

**Ms. Y. Sowjanya,**  
Asst. Prof. - CSE(SE)

**E Certificates will be provided for participants**

**Select Host College as "SIDDHARTHA INSTITUTE OF TECHNOLOGY & SCIENCES**  
**(UGC - AUTONOMOUS)"**

**Day 1 –(19.02.2024)**

## **INTRODUCTION ABOUT AICTE: -**

All India Council for Technical Education (AICTE) is a national-level Apex Advisory Body to conduct a survey on the facilities available for technical education and to promote development in the country in a coordinated and integrated manner.

## **CHIEF GUEST:**

**BUDHA CHANDRASEKHAR SIR - CHIEF COORDINATING OFFICER of AICTE**



Buddha Chandrasekhar sir is focused to make India as the Education and Knowledge capital of the world by providing the best Education, Technology, Skilling, Internships, Entrepreneurship, Startup, and innovation eco-system to the youth of India.

He is expertised in Technologies, Ideas, Multi-domain business knowledge, Entrepreneur, Leader with a Go-getter attitude.

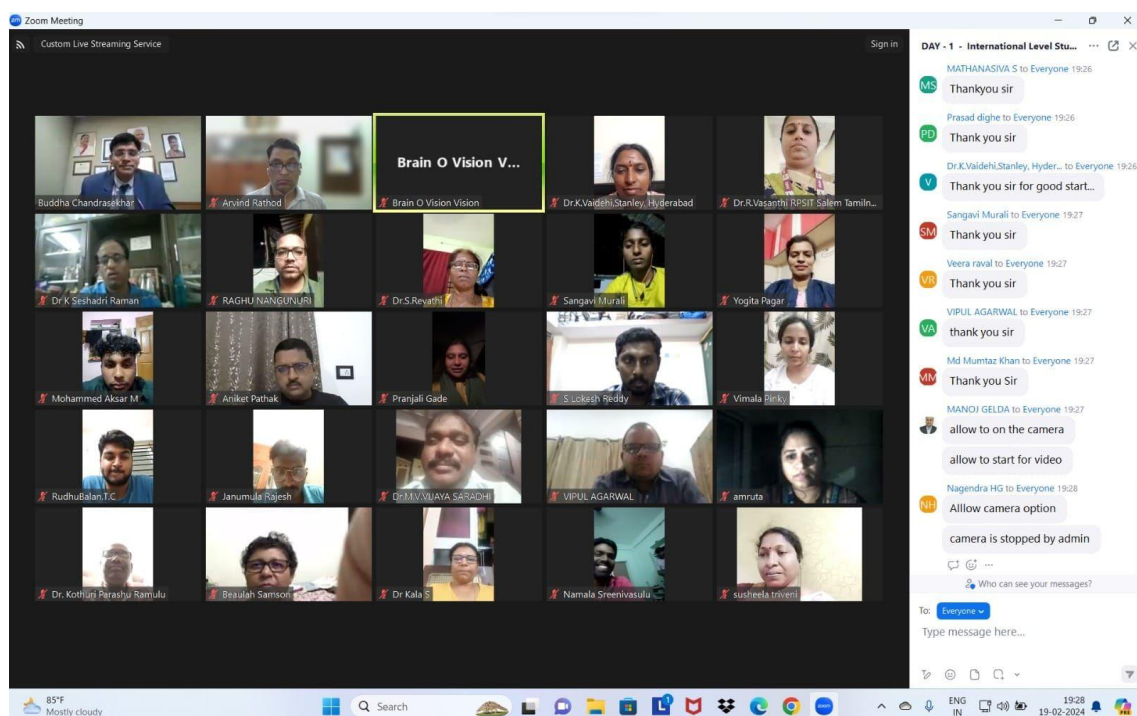
Sir has led the team and played a major role in developing the revolutionary tool **ANUVADINI - CHUTKI**. Real time language translator made in INDIA.

Voice & Document AI Translation Tools consisting of a multitude of features and functionalities desires to close this gap arising due to the language barriers. The tool has support for 22 regional Indian & foreign languages helping break language barriers & unifying India and the World under the principles of Ek Bharat Shreshtha Bharat and One Earth, One Family, One Future!

Chandrasekhar sir focusing on strategy, technology, and business transformation. He is an expert in managing advanced technologies, processes, cybersecurity, and people. Buddha Chandrasekhar Architected 180+ technology-based systems.

He is leading the Ministry of Education National Educational Alliance for Technology (NEAT) Initiative, Internship Enterprise Portal, Artificial Intelligence Language translation project, Indian Knowledge Systems, distributed 450 Crore free digital content to Higher Education Students and leading various student up-skilling initiatives.

## Co-Ordinators



## TOPICS COVERED:

- 1. Introduction to Data Science and Python Basics**
- 2. What is data science?**
- 3. Role of data science in various industries.**
- 4. Introduction to Python programming.**
- 5. Python variables, data types, and basic operations.**

## **SESSION 1:**

### **Introduction to Data Science and Python Basics**

Data science is a field that involves using statistical and computational techniques to extract insights and knowledge from data. It encompasses a wide range of tasks, including data cleaning and preparation, data visualization, statistical modelling, machine learning, and more. Data scientists use these techniques to discover patterns and trends in data, make predictions, and support decision-making. They may work with a variety of data types, including structured data (such as numbers and dates in a spreadsheet) and unstructured data (such as text, images, or audio). Data science is used in a wide range of industries, including finance, healthcare, retail, and more.

Data science is a multidisciplinary field that uses statistical and computational methods to extract insights and knowledge from data. It involves a combination of skills and knowledge from various fields such as statistics, computer science, mathematics, and domain expertise.

The process of data science involves several steps, including data collection, cleaning, exploration, analysis, and interpretation. These steps are often iterative, and the process may be refined based on the results obtained.

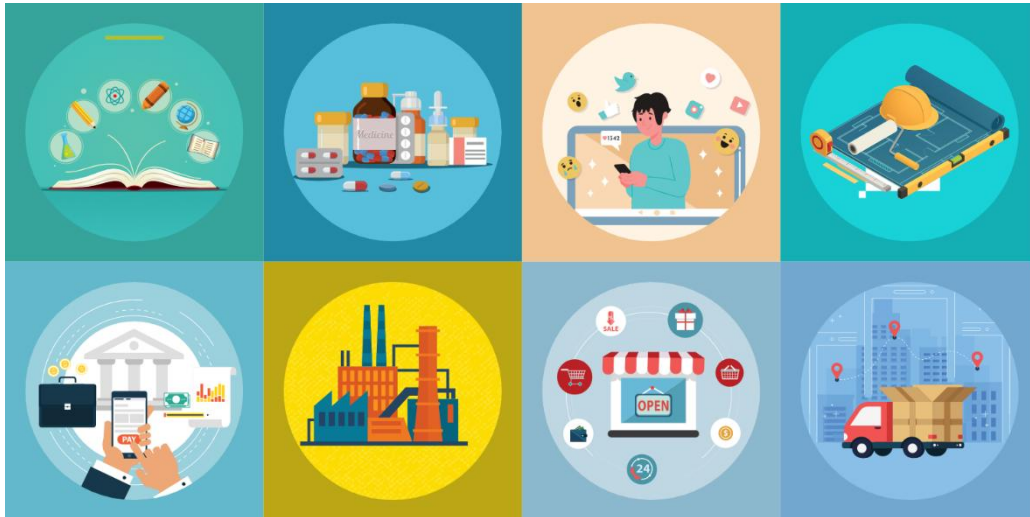
#### **What is data science?**

Data Science is kind a blended with various tools, algorithms, and machine learning principles. Most simply, it involves obtaining meaningful information or insights from structured or unstructured data through a process of analysing, programming and business skills. It is a field containing many elements like mathematics, statistics, computer science, etc. Those who are good at these respective fields with enough knowledge of the domain in which you are willing to work can call themselves as Data Scientist. It's not an easy thing to do but not impossible too. You need to start from data, it's visualization, programming, formulation, development, and deployment of your model. In the future, there will be great hype for data scientist jobs. Taking in that mind, be ready to prepare yourself to fit in this world.

- Data science is a field that involves using statistical and computational techniques to extract insights and knowledge from data.
- It is a multi-disciplinary field that encompasses aspects of computer science, statistics, and domain-specific expertise. Data scientists use a variety of tools and methods, such as machine learning, statistical modeling, and data visualization, to analyze and make predictions from data.
- They work with both structured and unstructured data, and use the insights gained to inform decision making and support business operations.
- Data science is applied in a wide range of industries, including finance, healthcare, retail, and more. It helps organizations to make data-driven decisions and gain a competitive advantage.

## **Role of data science in various industries.**

- Data science has been effective in tackling many real-world problems and is being increasingly adopted across industries to power more intelligent and better-informed decision-making.
- With the increased use of computers for day-to-day business and personal operations, there is a demand for intelligent machines, can learn human behavior and work patterns.
- This brings Data science and big data analytics to the forefront.



Almost all industries can benefit from data science and analytics. However, below are some industries that are better poised to make use of data science and analytics.

1. Retail
2. Medicine
3. Banking and Finance
4. Construction
5. Transportation
6. Communications, Media, and Entertainment
7. Education
8. Manufacturing and Natural Resources
9. Government
10. Energy and Utilities
11. Outsourcing Industry

## **Introduction to Python programming.**

Python is a widely used general-purpose, high level programming language. It was created by Guido van Rossum in 1991 and further developed by the Python Software Foundation. It was designed with an emphasis on code readability, and its syntax allows programmers to express their concepts in fewer lines of code. Python is a programming language that lets you work quickly and integrate systems more efficiently.

## **LANGUAGE FEATURES**

- Interpreted
- Platform Independent
- Free and Open Source
- High-level Language.
- Simple
- Embeddable
- Robust:
- Rich Library Support

## **Organizations using Python :**

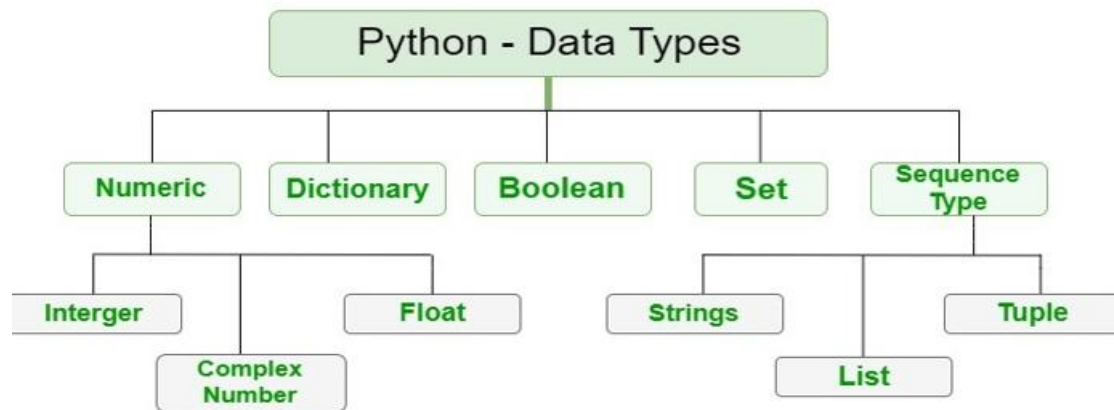
1. Google(Components of Google spider and Search Engine)
2. Yahoo(Maps)
3. YouTube
4. Mozilla
5. Dropbox
6. Microsoft
7. Cisco
8. Spotify
9. Quora

## **Python variables, data types, and basic operations.**

Data types are the classification or categorization of data items. It represents the kind of value that tells what operations can be performed on a particular data. Since everything is an object in Python programming, data types are classes and variables are instances (objects) of these classes.

The following are the standard or built-in data types in Python:

- Numeric
- Sequence Type
- Boolean
- Set
- Dictionary
- Binary Types( memoryview, bytearray, bytes)



### Python type() Function:

- To define the values of various data types and check their data types we use the **type()** function.

This code assigns variable 'x' different values of various data types in Python. It covers **string, integer, float, complex, list, tuple, range, dictionary, set, frozenset, boolean, bytes, bytearray, memoryview**, and the special value '**None**' successively. Each assignment replaces the previous value, making 'x' take on the data type and value of the most recent assignment.

### Numeric Data Types in Python:

- The numeric data type in Python represents the data that has a numeric value. A numeric value can be an integer, a floating number, or even a complex number. These values are defined as Python int, Python float, and Python complex classes in Python.
  - **Integers** – This value is represented by int class. It contains positive or negative whole numbers (without fractions or decimals). In Python, there is no limit to how long an integer value can be.
  - **Float** – This value is represented by the float class. It is a real number with a floating-point representation. It is specified by a decimal point. Optionally, the character e or E followed by a positive or negative integer may be appended to specify scientific notation.
  - **Complex Numbers** – A complex number is represented by a complex class. It is specified as (real part) + (imaginary part)j. For example – 2+3j

### Sequence Data Type in Python:

- The sequence Data Type in Python is the ordered collection of similar or different data types. Sequences allow storing of multiple values in an organized and efficient fashion. There are several sequence types in Python –
  - Python String
  - Python List
  - Python Tuple

## String Data Type

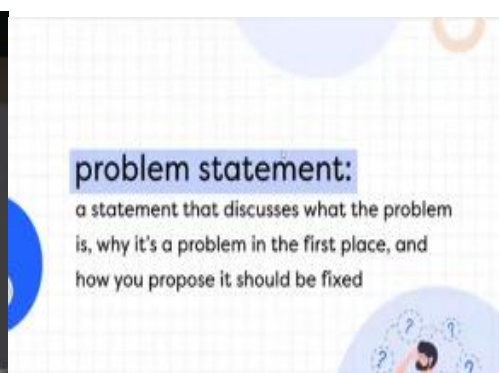
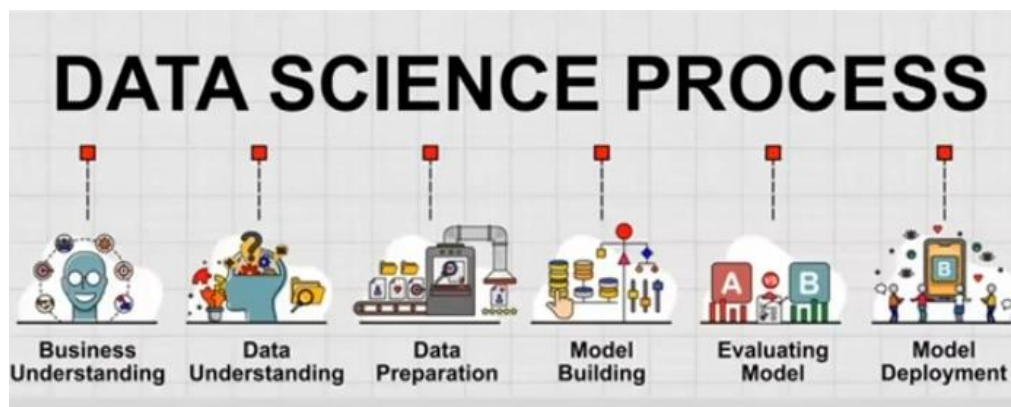
- Strings in Python are arrays of bytes representing Unicode characters.
- A string is a collection of one or more characters put in a single quote, double-quote, or triple-quote. In Python there is no character data type, a character is a string of length one. It is represented by str class.

### Creating String

Strings in Python can be created using single quotes, double quotes, or even triple quotes.

**Example:** This Python code showcases various string creation methods. It uses single quotes, double quotes, and triple quotes to create strings with different content and includes a multiline string.

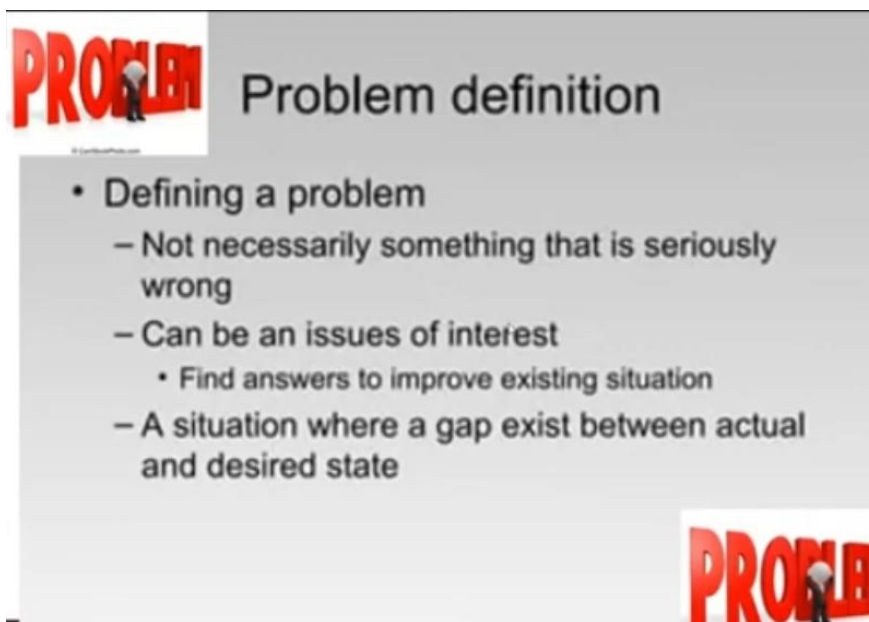
## Shared Slides from Presentations





## ***DOMAIN STUDY***

It aims to **identify features** common to a domain of applications, **selecting** and **abstracting** the objects and operations that characterize those features.



**PROBLEM** Problem definition

- Defining a problem
  - Not necessarily something that is seriously wrong
  - Can be an issues of interest
    - Find answers to improve existing situation
  - A situation where a gap exist between actual and desired state

**PROBLEM**

**DAY 2 –(20.02.2024)**

## **DATA MANIPULATION AND VISUALIZATION**

### **TOPICS COVERED:**

- 1. Data Manipulation with Python**
- 2. Working with data structures (lists, dictionaries, pandas data frames).**
- 3. Data loading and manipulation.**
- 4. Data Visualization with Python**
- 5. Introduction to Data Visualization**
- 6. Creating charts and Graphs using libraries like Matplotlib.**

**Session: 2**

## **Data Manipulation with Python**

In Machine Learning, the model requires a dataset to operate, i.e. to train and test. But data doesn't come fully prepared and ready to use. There are discrepancies like Nan/ Null / NA values in many rows and columns. Sometimes the data set also contains some of the rows and columns which are not even required in the operation of our model. In such conditions, it requires proper cleaning and modification of the data set to make it an efficient input for our model. We achieve that by practicing **Data Wrangling** before giving data input to the model. Today, we will get to know some methods using Pandas which is a famous library of Python. And by using it we can make out data ready to use for training the model and hence getting some useful insights from the results.

### **Installing Pandas**

Before moving forward, ensure that Pandas is installed in your system. If not, you can use the following command to install it:

```
pip install pandas
```

### **Creating DataFrame**

Let's dive into the programming part. Our first aim is to create a Pandas dataframe in Python, as you may know, pandas is one of the most used libraries of Python.

## **Working with data structures (lists, dictionaries, pandas data frames).**

### **Lists**

Lists in Python are implemented as **dynamic mutable arrays** which hold an **ordered** collection of items.

First, in many programming languages, arrays are data structures that contain a collection of elements of the same data types (for instance, all elements are integers). However, in Python, lists can contain heterogeneous data types and objects. For instance, integers, strings, and even functions can be stored within the same list. Different elements of a list can be accessed by integer indices where the first element of a list has the index of 0. This property derives from the fact that in Python, lists are ordered, which means they retain the order in which you insert the elements into the list.

Next, we can arbitrarily add, remove, and change elements in the list. For instance, the `append()` method adds a new element to a list, and the `.remove()` method removes an element from a list. Furthermore, by accessing a list's element by index, we can change it to another element. For more detail on different list methods, please refer to the documentation.

Finally, when creating a list, we do not have to specify in advance the number of elements it will contain; therefore, it can be expanded as we wish, making it dynamic.

Lists are useful when we want to store a collection of different data types and subsequently add, remove, or perform operations on each element of the list (by looping through them). Furthermore, lists are useful to store other data structures (and even other lists) by creating, for instance, lists of dictionaries, tuples, or lists. It is very common to store a table as a list of lists (where each inner list represents a table's column) for subsequent data analysis.

Thus, the **pros of lists** are:

- They represent the easiest way to store a collection of related objects.
- They are easy to modify by removing, adding, and changing elements.
- They are useful for creating nested data structures, such as a list of lists/dictionaries.

However, they also have **cons**:

- They can be pretty slow when performing arithmetic operations on their elements. (For speed, use NumPy's arrays.)
- They use more disk space because of their under-the-hood implementation.

## Dictionaries

Dictionaries in Python are very similar to real-world dictionaries. These are **mutable** data structures that contain a collection of **keys** and, associated with them, **values**. This structure makes them very similar to word-definition dictionaries. For example, the word dictionary (our key) is associated with its definition (value) in Oxford online dictionary: a book or electronic resource that gives a list of the words of a language in alphabetical order and explains what they mean, or gives a word for them in a foreign language.

Dictionaries are used to quickly access certain data associated with a **unique** key. Uniqueness is essential, as we need to access only certain pieces of information and not confuse it with other entries. Imagine we want to read the definition of Data Science, but a dictionary redirects us to two different pages: which one is the correct one? Note that technically we can create a dictionary with two or more identical keys, although due to the nature of dictionaries, it is not advisable.

We use dictionaries when we are able to associate (in technical terms, **to map**) a unique key to certain data, and we want to access that data **very quickly** (in constant time, no matter the dictionary size). Moreover, dictionary values can be pretty complex. For example, our keys can be customer names, and their personal data (values) can be dictionaries with the keys like "Age," "Hometown," etc.

Thus, the **pros of dictionaries** are:

- They make code much easier to read if we need to generate key:value pairs. We can also do the same with a list of lists (where inner lists are pairs of "keys" and "values"), but this looks more complex and confusing.
- We can look up a certain value in a dictionary very quickly. Instead, with a list, we would have to read the list before we hit the required element. This difference grows drastically if we increase the number of elements.

However, **their cons** are:

- They occupy a lot of space. If we need to handle a large amount of data, this is not the most suitable data structure.
- In Python 3.6.0 and later versions, dictionaries remember the order of element insertions. Keep that in mind to avoid compatibility issues when using the same code in different versions of Python.

### Examples:

Let's now take a look at a few examples.

First, we can create a dictionary with curly brackets ({} ) or the dict() constructor:

Now let's access an element in a dictionary. We can do this with the same method as lists:

Next, we can also modify dictionaries -- for example, by adding new key:value pairs:

As we can see, a new key, "Violet", has been added.

It's also possible to remove elements from a dictionary, so look for a way to do so by reading the documentation. Furthermore, you can read a more in-depth tutorial on Python dictionaries (with tons of examples) or have a look at DataQuest's dictionary lesson.

### Sets

Sets in Python can be defined as mutable dynamic collections of **immutable unique** elements. The elements contained in a set must be immutable. Sets may seem very similar to lists, but in reality, they are very different.

First, they may **only contain unique elements**, so no duplicates are allowed. Thus, sets can be used to remove duplicates from a list. Next, like sets in mathematics, they have unique operations which can be applied to them, such as set union, intersection, etc. Finally, they are very efficient in checking whether a specific element is contained in a set.

Thus, the pros of sets are:

- We can perform unique (but similar) operations on them.
- They are significantly faster than lists if we want to check whether a certain element is contained in a set.

But their cons are:

- Sets are intrinsically unordered. If we care about keeping the insertion order, they are not our best choice.
- We cannot change set elements by indexing as we can with lists.

### Tuples

Tuples are almost identical to lists, so they contain an ordered collection of elements, except for one property: they are **immutable**. We would use tuples if we needed a data structure that, once created, cannot be modified anymore. Furthermore, tuples can be used as dictionary keys if all the elements are immutable.

Other than that, tuples have the same properties as lists. To create a tuple, we can either use round brackets (()) or the tuple() constructor. We can easily transform lists into tuples and vice versa (recall that we created the list l4 from a tuple).

The **pros of tuples** are:

- They are immutable, so once created, we can be sure that we won't change their contents by mistake.
- They can be used as dictionary keys if all their elements are immutable.

The **cons of tuples** are:

- We cannot use them when we have to work with modifiable objects; we have to resort to lists instead.
- Tuples cannot be copied.
- They occupy more memory than lists.

### **Data loading and manipulation.**

- Data Manipulation is the process of manipulating (creating, arranging, deleting) data points in a given data to get insights much easier. We know that about 90% of the data we have are unstructured.
- Data manipulation is a fundamental step in data analysis, data mining, and data preparation for machine learning and is essential for making informed decisions and drawing conclusions from raw data.

To make use of these data points, we perform data manipulation. It involves:

- Creating a database
- SQL for structured data manipulation
- NoSQL languages like MongoDB for unstructured data manipulation.

### **Steps Required to Perform Data Manipulation**

The steps we perform in Data Manipulation are:

- **Mine the data and create a database:** The data is first mined from the internet, either with API requests or Web Scraping, and these data points are structured into a database for further processing.
- **Perform data preprocessing:** The Data acquired from mining is still a little rough and may have incorrect values, missing values, and some outliers. In this step, all these problems are taken care of, either by deleting the rows or, by adding the mean values in all missing areas (Note: This is only in the case of numerical data.)
- **Arrange the data:** After the data has been Preprocessed, it is arranged accordingly to make analysis of data easier.

- **Transform the data:** The data in question is transformed, either by changing datatypes or transposing data in some cases.
- **Perform Data Analysis:** Work with the data to view the result. Create visualizations or an output column to view the output.

**Tools Used in Data Manipulation** Many tools are used in Data Manipulation. Some most popularly known tools with no-code/code Data manipulation functionalities are:

- **MS Excel** – MS Excel is one of the most popular tools used for data manipulation. It provides a huge array/ variety for freedom/ manipulation of data.
- **Power BI** – It is a tool used to create interactive dashboards easily. It is provided by Microsoft and can be coded into it.
- **Tableau** – Tableau has a similar functionality as Power BI, but it is also a data analysis tool where you can manipulate data to create stunning visualizations.

## Operations of Data Manipulation

Data Manipulation follows the 4 main operations,

### CRUD (Create, Read, Update and Delete).

- It is used in many industries to improve the overall output.

In most DML, there is some version of the CRUD operations where:

- **Create:** To create a new data point or database.
- **Read:** Read the data to understand where we need to perform data manipulation.
- **Update:** Update missing/wrong data points with the correct ones to encourage data to be streamlined.
- **Delete:** Deletes the rows with missing data points/ erroneous/ misclassified data.

These **4 main operations** are performed in different ways seen below:

- **Data Preprocessing:** Most of the raw data that is mined may contain errors, missing values and mislabeled data. This will hamper the final output if it is not dealt with in the initial stages.
- **Structuring data** (if it is unstructured): If there's any sort of data available in the database which can be structured into a table to query them effectively, we sort those data into tables for greater efficiency.
- **Reduce the number of features:** As we know, data analysis is inherently computationally intensive. As a result, one of the reasons to perform data manipulation is to find out the optimum number of features needed for getting the result, while discarding the other features. Some techniques used here are, Principal Component Analysis (PCA), Discrete Wavelet Transform and so on.
- **Clean the data:** Delete unnecessary data points or outliers which may affect the final output. This is done to streamline the output.

- **Transforming data:** Some insights into data can be improved by transforming the data. This may involve transposing data, and arranging/rearranging them.

## Session 3:

### Data Visualization with Python

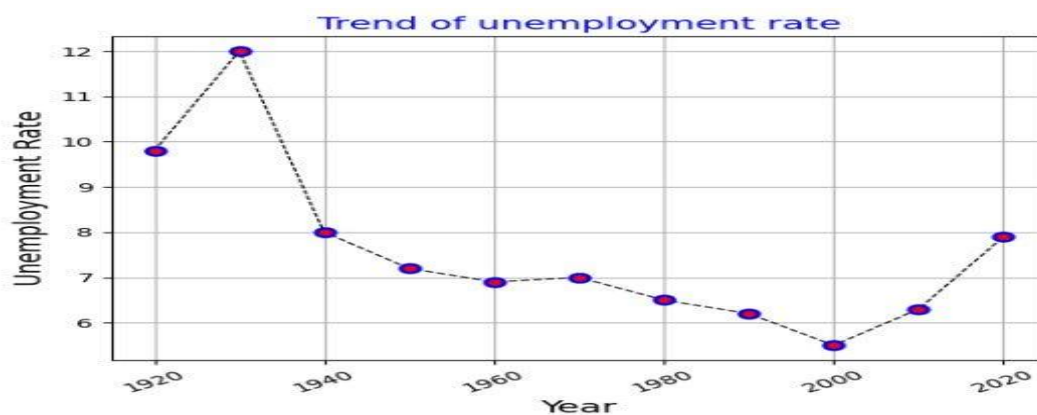
- Data Visualization is the art of presenting complex and unorganized data into a form that is easier to comprehend, visually appealing, and actionable. It can be a mammoth task to make sense out of data in the form of tables.
- This is when visualizing it in the form of charts, graphs, and maps becomes important. A clear representation of data makes it easy to understand the patterns, trends, and outliers in the data.

There can be a lot of graphs and charts to represent the data. But what type of visual would be best suited for a particular type of data? This is the question that we will try to answer in the article. This article mainly focuses on visualizing data through Python. There can be many useful data visualization tools like Tableau, PowerBI, Qlikview, etc. to do the same.

There are a lot of libraries that can be used in Python to visualize data. Some of the most common are matplotlib, seaborn, plotly, GGPlot, Bokeh, Geoplotlib, and PyGal. Here, we will mostly be focussing on the **matplotlib**, **seaborn**, and **plotly** for our visualizations.

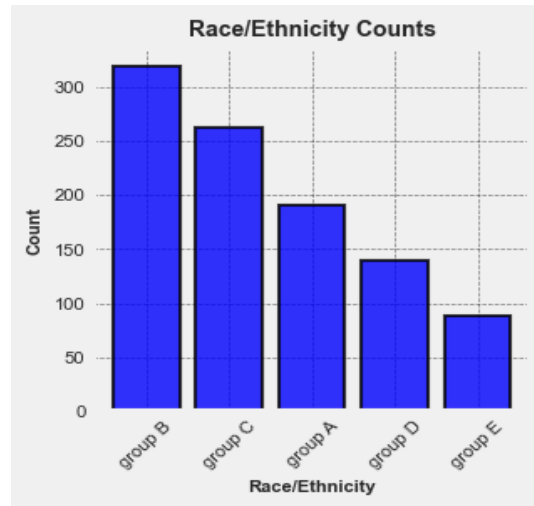
Let's now get into the details of different types of visualizations and when and how should we use them:

#### 1- Line Chart



It is one of the most common charts that is used to observe a single or multiple variables with the change of another variable. Basically, it is used in trend analysis and time series analysis. It is always a best practice to use line charts with continuous data, as the value of one variable can be found by projecting the other variable on the line at any point.

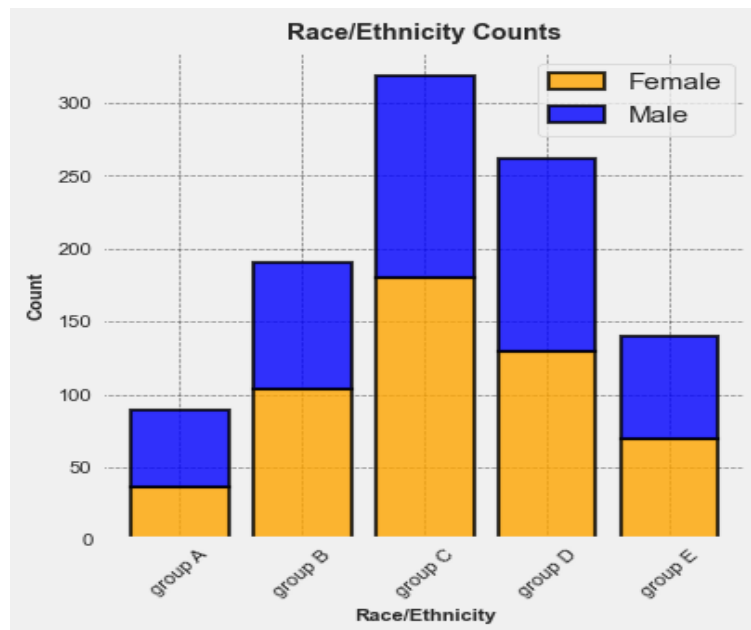
#### 2- Bar Chart



A bar chart is also one of the most used charts. It uses bars to show the change in the value of a particular variable wrt the other. This type of chart is generally used discrete or categorical data. A bar chart can be horizontal or vertical.

Now, in some cases we might need to add another categorical variable and show its variation our visualization. In such cases we use a **stacked bar chart**. Different bars, representing data of different categories present in that variable, are stacked one on top of the other.

The total length of the bar is equal to the sum of the values of all different categories in that variable. A stacked bar chart would show how the data is distributed among different categories.



### 3- Area Chart

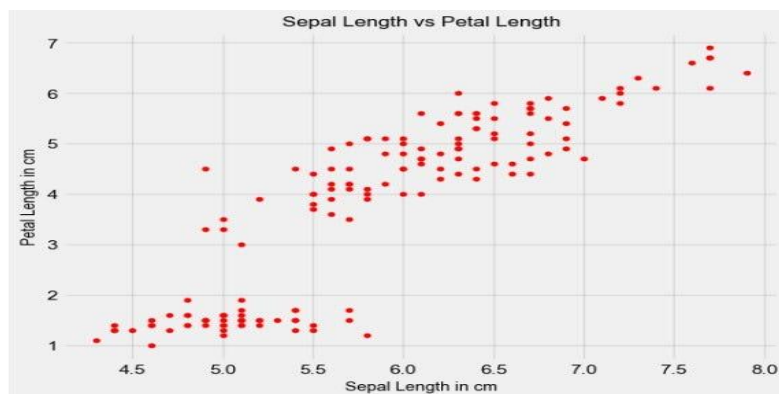




Area charts are quite similar to line charts except for the fact that the area under the line and x-axis is shaded with a color. This type of chart is effective in doing trend and time series analysis. However, in case of more than one category it becomes a little difficult to observe as the areas overlap. In this case it is advisable to add transparency to the charts for proper visuals.

Unlike line charts, we can also plot a stacked area chart. Just like stacked column charts, stacked area charts also show how different categories of a variable vary over time. Here, I am using the same Unemployment Rate dataset used for line chart.

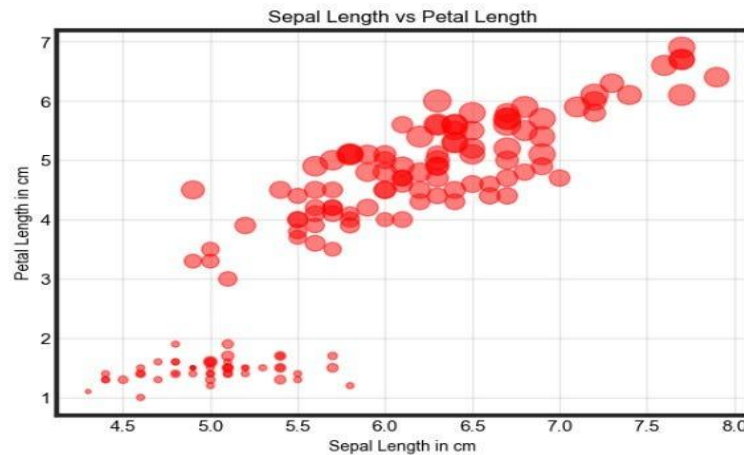
#### 4- Scatter Plot



In a scatter plot, the data is scattered between the two axes. These type of plots are basically used to study the correlation between the two variables. If one variable increases or decreases with other, they are positively correlated. If one variable increases as the other decreases and vice versa, they are negatively correlated. If there is no such relation, they aren't related. It is one of the most common charts to observe the relation between two variables.

If there are more than one categories in a variable, they can be shown by different colors. We can also plot a **trend line**, extrapolating which we can predict the future values. Here, I have the famous iris dataset.

## 5- Bubble Chart



A bubble chart is similar to a scatter plot but is used when we need to observe the relation between three variables. In this chart, the size of the bubble corresponds to the third variable.

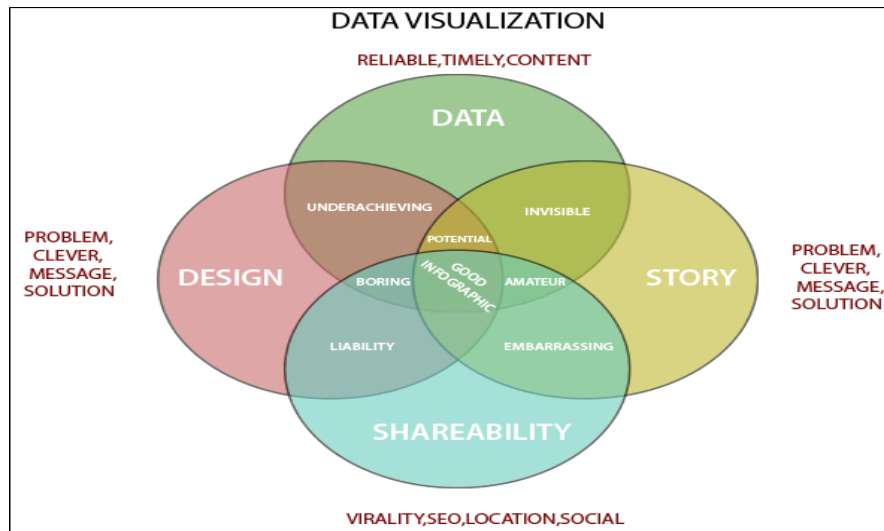
### Introduction to data visualization.

- Data visualization is a graphical representation of quantitative information and data by using visual elements like graphs, charts, and maps.
- Data visualization convert large and small data sets into visuals, which is easy to understand and process for humans.
- Data visualization tools provide accessible ways to understand outliers, patterns, and trends in the data.
- Data visualizations are common in your everyday life, but they always appear in the form of graphs and charts. The combination of multiple visualizations and bits of information are still referred to as Infographics.
- Data visualizations are used to discover unknown facts and trends. You can see visualizations in the form of line charts to display change over time. Bar and column charts are useful for observing relationships and making comparisons. A pie chart is a great way to show parts-of-a-whole. And maps are the best way to share geographical data visually.

Today's data visualization tools go beyond the charts and graphs used in the Microsoft Excel spreadsheet, which displays the data in more sophisticated ways such as dials and gauges, geographic maps, heat maps, pie chart, and fever chart.

### What makes Data Visualization Effective?

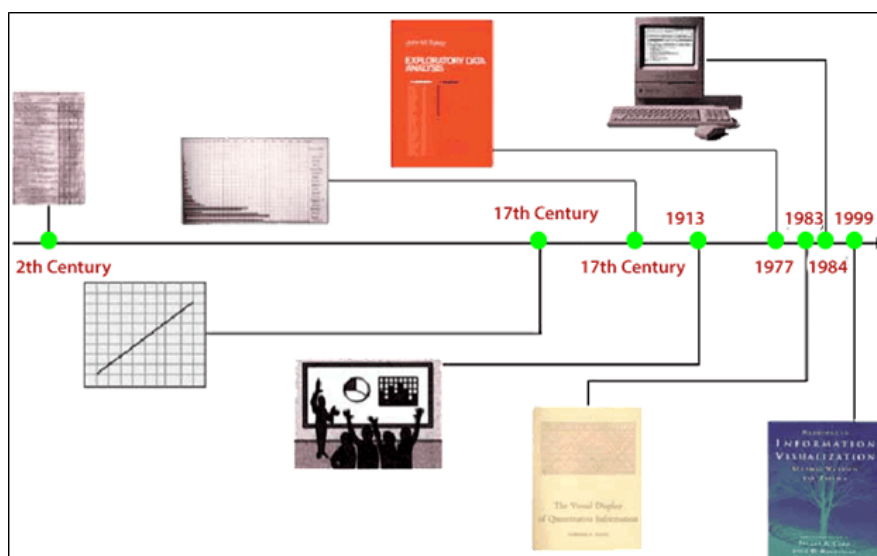
Effective data visualization are created by communication, data science, and design collide. Data visualizations did right key insights into complicated data sets into meaningful and natural.



- To craft an effective data visualization, you need to start with clean data that is well-sourced and complete. After the data is ready to visualize, you need to pick the right chart.
- After you have decided the chart type, you need to design and customize your visualization to your liking. Simplicity is essential - you don't want to add any elements that distract from the data.

### History of Data Visualization

- The concept of using picture was launched in the 17th century to understand the data from the maps and graphs, and then in the early 1800s, it was reinvented to the pie chart.
- Several decades later, one of the most advanced examples of statistical graphics occurred when **Charles Minard** mapped Napoleon's invasion of Russia.
- The map represents the size of the army and the path of Napoleon's retreat from Moscow - and that information tied to temperature and time scales for a more in-depth understanding of the event.
- Computers made it possible to process a large amount of data at lightning-fast speeds. Nowadays, data visualization becomes a fast-evolving blend of art and science that certain to change the corporate landscape over the next few years.



## Importance of Data Visualization

- Data visualization is important because of the processing of information in human brains. Using graphs and charts to visualize a large amount of the complex data sets is more comfortable in comparison to studying the spreadsheet and reports.
- Data visualization is an easy and quick way to convey concepts universally. You can experiment with a different outline by making a slight adjustment.

## Data visualization have some more specialties such as:

- Data visualization can identify areas that need improvement or modifications.
- Data visualization can clarify which factor influence customer behavior.
- Data visualization helps you to understand which products to place where.
- Data visualization can predict sales volumes.

Data visualization tools have been necessary for democratizing data, analytics, and making data-driven perception available to workers throughout an organization. They are easy to operate in comparison to earlier versions of BI software or traditional statistical analysis software. This guide to a rise in lines of business implementing data visualization tools on their own, without support from IT.

## Why Use Data Visualization?

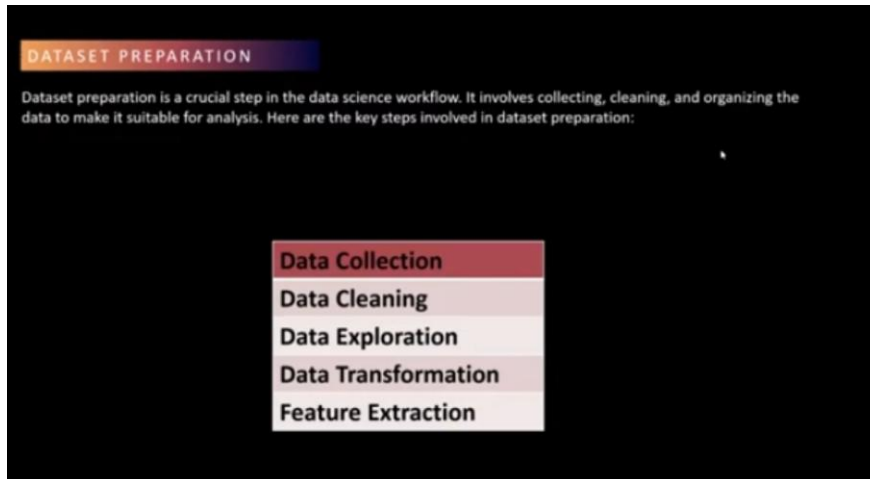
1. To make easier in understand and remember.
2. To discover unknown facts, outliers, and trends.
3. To visualize relationships and patterns quickly.
4. To ask a better question and make better decisions.
5. To competitive analyze.
6. To improve insights.

## Shared Slides from Presentations

**Feature engineering is the most important art in machine learning which creates a huge difference between a good model and a bad model.**

**STEPS THAT ARE INVOLVED WHILE SOLVING ANY PROBLEM IN MACHINE LEARNING ARE AS FOLLOWS:**

<b>Feature Creation</b>
<b>Feature Selection</b>
<b>Feature Transformation</b>
<b>Feature Encoding</b>
<b>Feature Extraction</b>



**Day 3 – (21.2.2024)**

## **DATA ANALYSIS WITH PYTHON**

### **TOPICS COVERED:**

- 1. Data Analysis with Pandas.**
- 2. Data cleaning and preparing.**
- 3. Exploratory data analysis (EDA) with Pandas.**

### **Session 4:**

#### **Data Analysis with Pandas**

- **Pandas** are the most popular python library that is used for data analysis.
- It provides highly optimized performance with back-end source code purely written in **C** or **Python**.

We can analyze data in Pandas with:

- **Pandas Series**
- **Pandas DataFrames**

#### **Pandas Series**

**Series** in Pandas is one dimensional(1-D) array defined in pandas that can be used to store any data type.

#### **Data cleaning and preparation.**

While the techniques used for data cleaning may vary according to the types of data your company stores, you can follow these basic steps to map out a framework for your organization.

**Step 1:** Remove duplicate or irrelevant observations

Remove unwanted observations from your dataset, including duplicate observations or irrelevant observations. Duplicate observations will happen most often during data collection. When you combine data sets from multiple places, scrape data, or receive data from clients or multiple departments, there are opportunities to create duplicate data. De-duplication is one of the largest areas to be considered in this process. Irrelevant observations are when you notice observations that do not fit into the specific problem you are trying to analyze. For example, if you want to analyze data regarding millennial customers, but your dataset includes older generations, you might remove those irrelevant observations. This can make analysis more efficient and minimize distraction from your primary target—as well as creating a more manageable and more performant dataset.

### **Step 2:** Fix structural errors

Structural errors are when you measure or transfer data and notice strange naming conventions, typos, or incorrect capitalization. These inconsistencies can cause mislabeled categories or classes. For example, you may find “N/A” and “Not Applicable” both appear, but they should be analyzed as the same category.

### **Step 3:** Filter unwanted outliers

Often, there will be one-off observations where, at a glance, they do not appear to fit within the data you are analyzing. If you have a legitimate reason to remove an outlier, like improper data-entry, doing so will help the performance of the data you are working with. However, sometimes it is the appearance of an outlier that will prove a theory you are working on. Remember: just because an outlier exists, doesn't mean it is incorrect. This step is needed to determine the validity of that number. If an outlier proves to be irrelevant for analysis or is a mistake, consider removing it.

### **Step 4:** Handle missing data

You can't ignore missing data because many algorithms will not accept missing values. There are a couple of ways to deal with missing data. Neither is optimal, but both can be considered.

1. As a first option, you can drop observations that have missing values, but doing this will drop or lose information, so be mindful of this before you remove it.
2. As a second option, you can input missing values based on other observations; again, there is an opportunity to lose integrity of the data because you may be operating from assumptions and not actual observations.
3. As a third option, you might alter the way the data is used to effectively navigate null values.

### **Step 5:** Validate and QA

At the end of the data cleaning process, you should be able to answer these questions as a part of basic validation:

- Does the data make sense?
- Does the data follow the appropriate rules for its field?
- Does it prove or disprove your working theory, or bring any insight to light?
- Can you find trends in the data to help you form your next theory?
- If not, is that because of a data quality issue?

False conclusions because of incorrect or “dirty” data can inform poor business strategy and decision-making. False conclusions can lead to an embarrassing moment in a reporting meeting when you realize your data doesn’t stand up to scrutiny. Before you get there, it is important to create a culture of quality data in your organization. To do this, you should document the tools you might use to create this culture and what data quality means to you.



### Components of quality data

Determining the quality of data requires an examination of its characteristics, then weighing those characteristics according to what is most important to your organization and the application(s) for which they will be used.

#### 5 characteristics of quality data

1. **Validity.** The degree to which your data conforms to defined business rules or constraints.
2. **Accuracy.** Ensure your data is close to the true values.
3. **Completeness.** The degree to which all required data is known.
4. **Consistency.** Ensure your data is consistent within the same dataset and/or across multiple data sets.
5. **Uniformity.** The degree to which the data is specified using the same unit of measure.

### Advantages and benefits of data cleaning

Having clean data will ultimately increase overall productivity and allow for the highest quality information in your decision-making. Benefits include:

- Removal of errors when multiple sources of data are at play.

- Fewer errors make for happier clients and less-frustrated employees.
- Ability to map the different functions and what your data is intended to do.
- Monitoring errors and better reporting to see where errors are coming from, making it easier to fix incorrect or corrupt data for future applications.
- Using tools for data cleaning will make for more efficient business practices and quicker decision-making.

Data preparation is the process of cleaning and transforming raw data prior to processing and analysis. It is an important step prior to processing and often involves reformatting data, making corrections to data, and combining datasets to enrich data.

Data preparation is often a lengthy undertaking for data engineers or business users, but it is essential as a prerequisite to put data in context in order to turn it into insights and eliminate bias resulting from poor data quality.

For example, the data preparation process usually includes standardizing data formats, enriching source data, and/or removing outliers.

### **Benefits of data preparation in the cloud**

76% of data scientists say that data preparation is the worst part of their job, but efficient, accurate business decisions can only be made with clean data.

Data preparation helps:

- **Fix errors quickly** Data preparation helps catch errors before processing. After data has been removed from its original source, these errors become more difficult to understand and correct.
- **Produce top-quality data** Cleaning and reformatting datasets ensures that all data used in analysis will be of high quality.
- **Make better business decisions** Higher-quality data that can be processed and analyzed more quickly and efficiently leads to more timely, efficient, better-quality business decisions.

Additionally, as data and data processes move to the cloud, data preparation moves with it for even greater benefits, such as:

- **Superior scalability** Cloud data preparation can grow at the pace of the business. Enterprises don't have to worry about the underlying infrastructure or try to anticipate their evolutions.
- **Future proof** Cloud data preparation upgrades automatically so that new capabilities or problem fixes can be turned on as soon as they are released. This allows organizations to stay ahead of the innovation curve without delays and added costs.
- **Accelerated data usage and collaboration** Doing data prep in the cloud means it is always on, doesn't require any technical installation, and lets teams collaborate on the work for faster results.



Additionally, a good, cloud-native data preparation tool will offer other benefits (like an intuitive and simple-to-use GUI) for easier and more efficient preparation.

## Data preparation steps

The specifics of the data preparation process vary by industry, organization, and need, but the workflow remains largely the same.

### 1. Gather data

The data preparation process begins with finding the right data. This can come from an existing data catalog or data sources can be added ad-hoc.

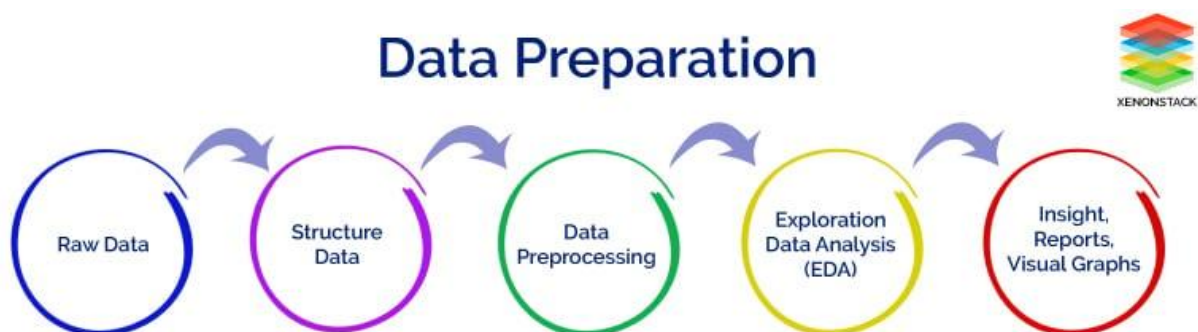
### 2. Discover and assess data

After collecting the data, it is important to discover each dataset. This step is about getting to know the data and understanding what has to be done before the data becomes useful in a particular context.

Discovery is a big task, but Talend's data preparation platform offers visualization tools which help users profile and browse their data.

## Exploratory data analysis (EDA) with Pandas.

The Pandas library offers numerous intuitive methods that can be successfully applied for exploratory data analysis. EDA is an important step in any data science project. It helps grasp the size and structure of the data, identify the critical variables and potential issues with them, explore the statistics of the data, find the correlation between different columns, and discover hidden patterns in the data for further investigation. EDA in pandas allows running a few lines of simple code to efficiently solve all these tasks and more.



## Introduction to Exploratory Data Analysis

Exploratory data analysis (EDA) is a vital initial step of any data analysis or machine learning project. It is necessary for:

- getting an overall understanding of the data, including first insights
- identifying the size of the dataset, its structure, and the features that are crucial for the project goal
- gathering fundamental statistics of the data

- detecting potential issues to fix (such as missing values, duplicates, or outliers)

The pandas library is the most popular Python package for conducting data wrangling and data analysis. It offers a vast spectrum of analytical tools and methods to explore the data from all angles. Performing EDA in pandas is an easy task because of the intuitive names and syntaxes of its functions, their flexibility, and comprehensive library documentation.

## **EDA Functions in Pandas**

Let us take a look at the essential methods that allow for conducting fast and efficient EDA in pandas. Note that some of those functions are applicable only for data frames, others – only for series, and others – for both. In the context of EDA in pandas, we usually talk about a series object as an individual column of a dataframe.

### **Head() and tail()**

By default, the head() method returns the first five and tail() – the last five rows of a dataframe or a series. To return the number of rows different from five, we need to pass in that number.

### **Sample()**

By default, it returns a random row of a dataframe or a series. To return a certain number of random rows, we need to pass in that number.

### **Shape**

For a dataframe, it returns a tuple with the number of rows and columns. For a series, it returns a one-element tuple with the number of rows.

### **Size**

Returns the number of elements in a dataframe or a series. For a series object, it makes more sense to use size rather than shape. The obtained information is the same in both cases, but size returns it in a more handy form – as an integer rather than a one-element tuple.

### **Info()**

Returns overall information about a dataframe, including the index data type, the number of rows and columns, column names, indices, and data types, the number of non-null values by column, and memory usage.

### **Describe()**

Returns the major statistics of a dataframe or a series, including the number of non-null values, the minimum, maximum, and mean values, and percentiles. For a dataframe, it returns the information by column, and by default, only for numeric columns. To include the statistics for the columns of an object type as well, we need to pass in include='all'. For object

columns, the method returns the number of non-null values, the number of unique values, the most frequent value, and the number of times it is encountered in the corresponding column.

## **Dtypes**

Returns the data types of a dataframe by column. If a column contains mixed data types or if all its values are None, the returned data type of that column will be an object. This also includes a special case of a column containing Boolean and null (NaN) or None values.

For a series, we can interchangeably use either dtypes or dtype for EDA in pandas.

## **Select\_dtypes()**

Returns a subset of the columns of a dataframe based on the provided column data type (or types). We have to specify the data type (or types) either to include into the subset (using the include parameter) or exclude from it (exclude).

Some typical data types are number, int, float, object, bool, category, and datetime. To specify several data types to include or exclude, we pass in a list of those data types.

## **Columns**

Returns the column names of a dataframe.

## **Count()**

Returns the count of non-null values in a dataframe or a series. For a dataframe, by default, returns the results by column. Passing in axis=1 or axis='columns' will give the results by row.

## **Unique() and nunique()**

The unique, () method returns the unique values of a series, while nunique() – the number of unique values in a dataframe or a series.

For a dataframe, nunique(), by default, returns the results by column. Otherwise, passing in axis=1 or axis='columns' will give the results by row.

## **Is\_unique**

Returns True if all the values in a series are unique.

## **IsNull() and isna()**

Both methods return a boolean same-sized object showing which values are null (True) and which are not (False). These functions apply to both series and dataframes.

IsNull() and isna() work best when chained with sum() (e.g., df.isnull().sum()) returning the number of null values by column for a dataframe or their total number for a series. For a dataframe, the method chaining df.isnull().sum().sum() gives the total number of null values.

### **hasnans**

Returns if a series contains at least one null value.

### **Value\_counts()**

Returns the count of each unique value in a series. By default, the outputs are not normalized, or sorted in descending order, and the null values are not considered. To override the defaults, we can set the optional parameters normalize, ascending, and dropna accordingly.

### **Nsmallest() and nlargest()**

By default, nsmallest() returns the five smallest while nlargest() – the five largest values of a series together with their indices. To return the number of values different from five, we need to pass in that number.

### **Is\_monotonic\_increasing and is\_monotonic\_decreasing**

The first method returns True if the values of a series are monotonically increasing, and the second – if they are decreasing.

### **Corr()**

This method of EDA in pandas applies both to dataframes and series, but in a slightly different way. For a dataframe (df.corr()), it returns column pairwise correlation, excluding null values. For a series (Series1.corr(Series2)), this method returns the correlation of that series with another one, excluding null values.

### **Plot()**

Allows the creation of simple plots of various kinds for a data frame or a series. The main parameters are x, y, and kind. The popular types of supported plots are line, bar, barh, hist, box, area, density, pie, and scatter.

In general, pandas are not the best choice for creating compelling visualizations in Python. However, for the purposes of EDA in pandas, the plot() method works just fine.

## **Day 4– (22.02.2024)**

### **Introduction to Machine Learning**

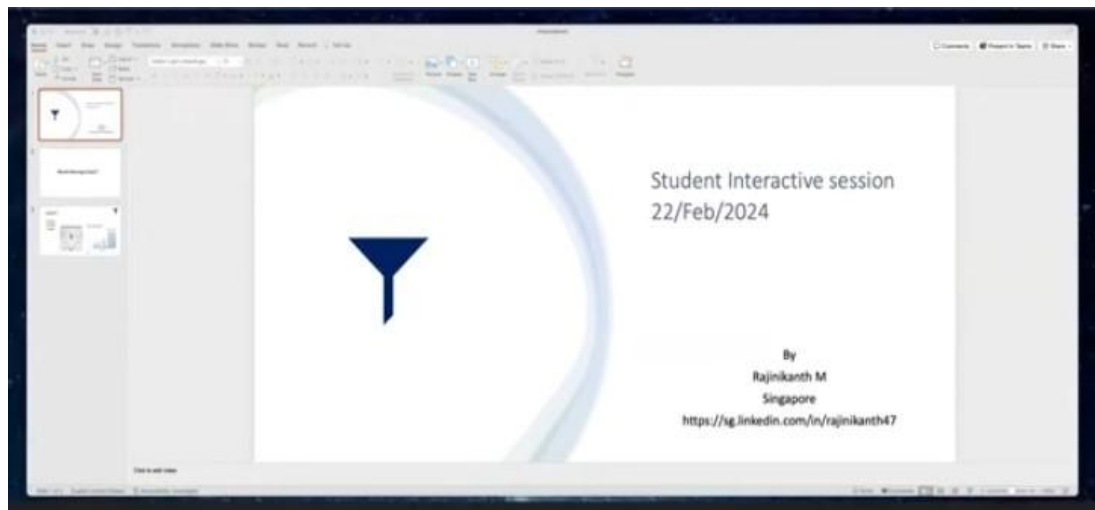
#### **TOPICS COVERED**

- 1. Introduction to Machine Learning**
- 2. Understanding machine Learning**
- 3. Supervised vs. unsupervised Learning**

## Session 5:

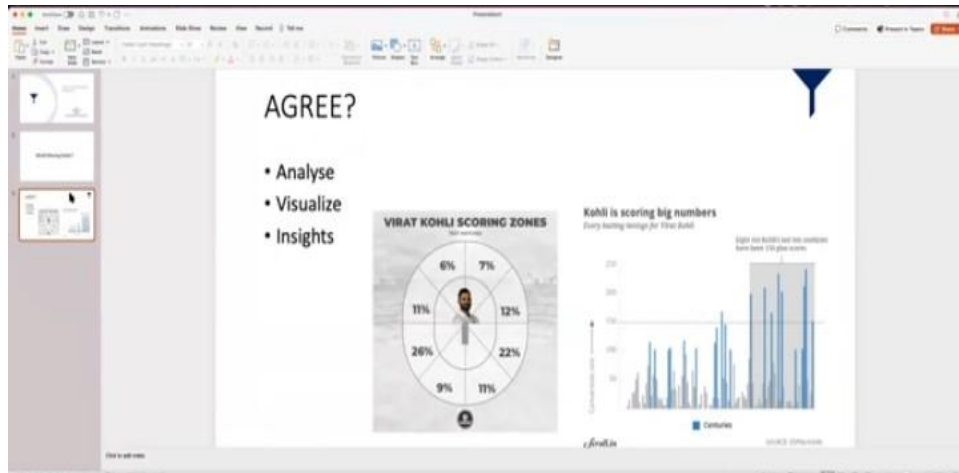
### Introduction to Machine Learning

- Machine learning is programming computers to optimize a performance criterion using example data or past experience. We have a model defined up to some parameters, and learning is the execution of a computer program to optimize the parameters of the model using the training data or past experience.
- The model may be predictive to make predictions in the future, or descriptive to gain knowledge from data.
- The field of study known as machine learning is concerned with the question of how to construct computer programs that automatically improve with experience.



### Understanding machine learning

- In [supervised learning](#), the machine is trained on a set of labeled data, which means that the input data is paired with the desired output. The machine then learns to predict the output for new input data. Supervised learning is often used for tasks such as classification, regression, and object detection.
- In unsupervised learning, the machine is trained on a set of unlabeled data, which means that the input data is not paired with the desired output. The machine then learns to find patterns and relationships in the data. Unsupervised learning is often used for tasks such as [clustering](#), dimensionality reduction, and anomaly detection.



```

1 @startuml
2
3
4 Kohli --> WhatsApp: Using the Smart watch
5 WhatsApp --> AppServer: [30 min] REST API - Send Data (Sensor data:ex GPS :
  Lat, lng
6 AppServer --> AppServer: Clean Data (validation lng=123 lat= null) json - py/
  json...etc
7 AppServer --> AppServer: predict data |
8 opt Kohli Running
9 AppServer --> AppServer: Predict the direction
10 else Kohli Resting
11 AppServer --> AppServer: Resting where??
12 end
13
14
15 @enduml

```

**Day 5 – (23.02.2024)**

## **Machine Learning with Scikit-Learn and Practical Implementation**

### **TOPICS COVERED:**

- 1. Machine Learning with Scikit-Learn and Practical Implementation**
- 2. Introduction to Scikit-Learn Library**
- 3. Building, training, and evaluating machine Learning models.**
- 4. Applying machine learning to real-world dataset**
- 5. Valedictory and Closing.**

**Session: 6**

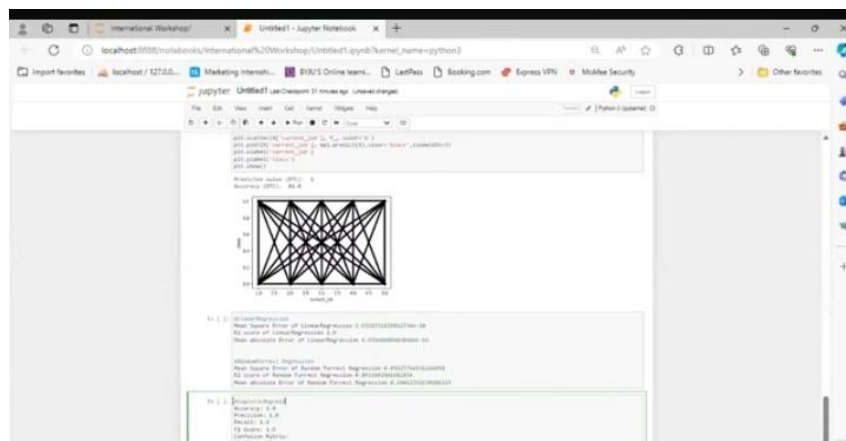
## Machine Learning with Scikit-Learn and Practical Implementation

The [scikit-learn library](#) is one of the most popular platforms for everyday machine learning and data science. The reason is because it is built upon Python, a fully featured programming language.

- Simple and efficient tools for predictive data analysis
- Accessible to everybody, and reusable in various contexts
- Built on NumPy, SciPy, and matplotlib

### What is Scikit-Learn in Python?

- Scikit-Learn is a free machine learning library for Python. It supports both supervised and unsupervised machine learning, providing diverse algorithms for classification, regression, clustering, and dimensionality reduction.
- The library is built using many libraries you may already be familiar with, such as NumPy and SciPy. It also plays well with other libraries, such as Pandas and Seaborn.



### Building, training, and evaluating machine learning models.

- Building machine learning models with Keras is all about assembling together layers, data-processing building blocks, much like we would assemble Lego bricks.
- These layers allow us to specify the sequence of transformations we want to perform on our input. As our learning algorithm takes in a single text input and outputs a single classification, we can create a linear stack of layers using the [Sequential model API](#).

```
[ ]: #LogisticRegression
Accuracy: 1.0
Precision: 1.0
Recall: 1.0
F1 Score: 1.0
Confusion Matrix:
[[ 631  0]
 [  0 1369]]

#DecisionTree Classification
Accuracy: 0.818
Precision: 0.8214971209213052
Recall: 0.9379108838568299
F1 Score: 0.8758526603001365
Confusion Matrix:
[[ 352 279]
 [  85 1284]]
```

$$Recall = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

**Recall:** Out of all the predictions, how many are actually correct. This metric basically tells us how good our model is at identifying relevant samples.

### Certification courses & job opportunities:

The screenshot shows the Naukri.com website with search filters on the left and job listings in the center. The filters include Work mode (Work from office, Remote, Hybrid, Temp. WFH), Experience (0-10 years), and Department (Engineering - Soft., IT & Information Se., Engineering - Hard., Other). The job listings include:

- AWS Cloud Architect - Infrastructure, AWS, Provisioning** by Photon (4.1 rating, 277 reviews). Location: Kolkata, Mumbai, New Del... Experience: 6-11 yrs. Skills: Automation, Linux, Configuration management, Agile, Network security.
- Inviting applications For the role of AWS Cloud Architect** by Genpact (4.0 rating, 2370 reviews). Location: Hyderabad. Experience: 3-7 yrs. Skills: AWS, Linux Administration, Architecture, Application, Administration.
- AWS Cloud Architect** by NET Cyber. Location: Kolkata, Mumbai, New Del... Experience: 8-10 yrs.

On the right, there is a 'blazeclan' advertisement for 'WE ARE HIRING Cloud Ops Engineer' and a 'See 31 jobs in Featured Companies' section listing CGI, SKP, Mphasis, Infosys, Tech Mahindra, epiq, Coforge, and Proapt.



## United nations department of economic and social affairs:



## India Book of Records Achievement Celebrations

For National Level Faculty Development Program on Cloud Infrastructure (21st to 25th Aug 2023)





## STUDENT FEEDBACK

- The course was presented in a professional way with opportunity to discuss the scenarios within our working environment.
- Our training session was excellent! Your energy and enthusiasm were infectious, and you kept us engaged throughout the entire session.
- A lot of information to fit in to 1 day although all very relevant to my domain.
- The session was informative, interactive and the instructor expressed an understanding of roles in the team clarifying what we expected to do and Not to do.
- I was able to relate to the scenarios and experiences shared by the instructor and this will positively impact on my decision making and actions in future.
- I am looking forward to training a gain on the topic–Dealing with the post positive handling stress.
- Very good and professional trainer I enjoyed and found this course most helpful.

**Thank you.**