ਜਗਤ ਗੁਰੂ ਨਾਨਕ ਦੇਵ
ਪੰਜਾਬ ਸਟੇਟ ਓਪਨ ਯੂਨੀਵਰਸਿਟੀ
ਪਟਿਆਲਾ

JAGAT GURU NANAK DEV
PUNJAB STATE OPEN UNIVERSITY, PATIALA

(Established by Act No. 19 of 2019 of the Legislature of State of Punjab)

The Motto of the University
(SEWA)

SKILL ENHANCEMENT **EMPLOYABILITY** **WISDOM**
ACCESSIBILITY

M.SC. (COMPUTER SCIENCE)
SEMESTER-III
COURSE: INTRODUCTION TO DATA SCIENCE

COURSE CODE: MSCS-3-04T

ADDRESS: C/28, THE LOWER MALL, PATIALA-147001
WEBSITE: www.psou.ac.in

M.Sc. (Computer Science)
Semester-3
MSCS-3-04T: Introduction to Data Science

Total Marks: 100
External Marks: 70
Internal Marks: 30
Credits: 4
Pass Percentage: 40%

INSTRUCTIONS FOR THE PAPER SETTER/EXAMINER

1. The syllabus prescribed should be strictly adhered to.
2. The question paper will consist of three sections: A, B, and C. Sections A and B will have four questions from the respective sections of the syllabus and will carry 10 marks each. The candidates will attempt two questions from each section.
3. Section C will have fifteen short answer questions covering the entire syllabus. Each question will carry 3 marks. Candidates will attempt any ten questions from this section.
4. The examiner shall give a clear instruction to the candidates to attempt questions only at one place and only once. Second or subsequent attempts, unless the earlier ones have been crossed out, shall not be evaluated.
5. The duration of each paper will be three hours.

INSTRUCTIONS FOR THE CANDIDATES

Candidates are required to attempt any two questions each from the sections A and B of the question paper and any ten short q questions from Section C. They have to attempt questions only at one place and only once. Second or subsequent attempts, unless the earlier ones have been crossed out, shall not be evaluated.

Course: Introduction to Data Science	
Course Code: MSCS-3-04T	
Course Outcomes (COs) After the completion of this course, the students will be able to:	
CO1	Understand tools and techniques to analyze and extract insights from data received from different data sources such as social media, IoT devices, and sensors.
CO2	Understand the general techniques and frameworks that can be used to handle special types of data, such as acoustic, image, sensor, and network data
CO3	Apply mathematical or logical operations to the data to derive new insights.
CO4	Apply tools for understanding complex data structures and relationships.
CO5	Explore various applications of data science in the field of business, energy, health care, biotechnology, manufacturing, telecommunication, pharmaceuticals etc.

SECTION-A

Unit I: Data Science: A discipline, Landscape-Data to Data science, Data Growth-issues and challenges, data science process. foundations of data science. Messy data, Anomalies and artefacts in datasets. Cleaning data.

Unit II: Introduction data acquisition: Structured Vs Unstructured data, data preprocessing techniques including data cleaning, selection, integration, transformation and reduction, data mining, interpretation.

Unit III: Representation of Data: Special types-acoustic, image, sensor and network data. Problems when handling large data – General techniques for handling large data, Distributing data storage and processing with Frameworks.

Unit IV: Data Science Ethics: Doing good data science, Owners of the data, valuing different aspects of privacy, getting informed consent, the five Cs, diversity, inclusion, future trends.

SECTION-B

Unit V: Data Wrangling Combining and Merging Data Sets: Reshaping and Pivoting, Data Transformation, String manipulations, Regular Expressions.

Unit VI: Data Aggregation and Group Operations: Group by Mechanics, Data Aggregation, Group Wise Operations, Transformations, Pivot Tables, Cross Tabulations, Date and Time data types.

Unit VII: Data Modeling: Basics of Generative modeling and Predictive modeling. Charts-histograms, scatter plots, time series plots etc. Graphs, 3D Visualization and Presentation.

Unit VIII: Applications of Data Science: Business, Insurance, Energy, Health care, Biotechnology, Manufacturing, Utilities, Telecommunication, Travel, Governance, Gaming, Pharmaceuticals, Geospatial analytics and modeling

Reference Books:

- Sinan Ozdemir, “Principles of Data Science”, Packt Publishing, 2016.
- Joel Grus, “Data Science from Scratch”, O’Reilly, 2016.
- Foster Provost & Tom Fawcett, “Data Science for Business”, O’Reilly, 2013.
- Roger D. Peng & Elizabeth Matsui, “The Art of Data Science”, Lean Publishing, 2015.

INTRODUCTION TO DATA SCIENCE

UNIT I: FOUNDATION OF DATA SCIENCE

STRUCTURE

1.0 OBJECTIVES

1. Introduction to Data Science
2. Familiarize with the issues and challenges in Data Growth
3. Familiarize with data Science Process
4. Familiarize with the concepts of data like clean and messy data, data artifacts and anomalies in data.

1.1 DATA SCIENCE-A DISCIPLINE

Data Science is a blend of various tools, algorithms and machine learning principles with the goal to discover hidden patterns from raw data. It is related to data mining, machine learning and Big Data It is an area that manages, manipulates, extracts, and interprets knowledge from tremendous amount of data. Data science (DS) is a multidisciplinary field of study with a goal to address the challenges in big data. Big Data has given rise to Data Science and is a therefore, a multidisciplinary field of study with goal to address the challenges in big data. Following are some main reasons for using data science:

- Data Science is useful in making the huge amount of raw and unstructured data into meaningful and useful data.
- Major companies like Google, Amazon, Netflix, etc, are using data science to handle the huge data, to improve on the better customer experience by being able to analyze and make predictions about trends and customer interests.
- Data science is used in almost every domain, therefore increasing the demand of a data science expert in each domain, leading to increased employability.

1.2 DATA TO DATA SCIENCE

The growth of data has been seen since 2010, due to the growth in the number of data generating devices like smart phones, wearables, Internet of things, etc. The availability of more data publicly from social media sites like Facebook, YouTube, twitter etc, business transactions, sensors, audio, video, photos etc. This enormous growth of data led to the concept of Big data, which is a term used for collection of large and complex data sets. As the data has increased, so did the need for its storage. Until 2010, the main focus was building framework and solutions to store data, which was successfully solved by HADOOP and other frameworks. In the present time, it has become difficult to process this large and complex data using traditional data management techniques such as, for example, the RDBMS (relational database management systems). This rise in the use of data, sparked the use of Data Science. Data science makes this possible, as it is a multidisciplinary study of data collection for analysis, prediction, learning and prevention. Data Science involves using methods to analyse massive amounts of data and extract the knowledge from the raw data and process it using scientific methods, algorithms and computer programming. Data science is an exciting, interdisciplinary field that is revolutionizing the way companies approach every

facet of their business. It helps in discovering hidden patterns from the raw data with the help of mathematics, statistics and data analysis.

1.3 DATA GROWTH ISSUES AND CHALLENGES

During the last decade, the most challenging problem the world envisaged is the big data problem[3].The data is growing at a much faster rate than computational speeds. Social activities, business transactions, sales figures, scientific experiments, biological explorations along with the sensor devices, customer logs are among the contributors of voluminous data which is referred as Big data. Big data is beneficial to the society and business but at the same time, it brings challenges like data quality, storage, lack of data scientists and data analysts. Big Data is characterized by five V's namely *volume*, *variety*, *velocity*, *veracity* and *value*. Consequently, the challenges these characteristics bring are being seen in *data capture*, *curation*, *storage*, *search*, *sharing*, *transfer*, and *visualization*. Each of the characteristic og the big data is a challenge in itself. The following section discusses the various characteristics of big data:

i. **Volume** refers to the enormous size of data. Big data refers to data volumes in the range of exabytes and beyond e.g. In the year 2016, the estimated global mobile traffic was 6.2 Exabytes (6.2 billion GB) per month. Also, by the year 2020 we will have almost 40000 Exabytes of data. One of the most pressing challenges of Big Data is storing all these huge sets of data properly. The amount of data being stored in data centers and databases of companies is increasing rapidly. As these data sets grow exponentially with time, which gets extremely difficult to handle.Most of the data is unstructured and comes from documents, videos, audios, text files and other sources. Such volumes exceed the capacity of current on-line storage systems and processing systems. Traditional database systems is not able to capture, store and analyse this large amount of data. Therefore, as the volume of the data increases, storage becomes an issue. To handle large volume of data, techniques like compression, tiering, and deduplication are being used. Compression is used for reducing the number of bits in the data, thus reducing its overall size. Deduplication is the process of removing duplicate and unwanted data from a data set.Storage solutions have been provided by HADOOP framework, but represent long term challenges that require research and new paradigms.

ii. **Velocity** refers to the high speed of accumulation of data. There is a massive and continuous flow of data from sources like machines, networks, social media, mobile phones etc. This determines the potential of data that how fast the data is generated and processed to meet the demands e.g. there are more than 3.5 billion searches per day are made on Google. FaceBook users are also increasing by 22% (approx.) year by year. Data is streaming in, at unprecedented speed and must be dealt with in a timely manner. RFID tags, sensors and smart metering are driving the need to deal with torrents of data in near-real time. The data has to be available at the right time to make business decisions accurately. Reacting quickly enough to deal with data velocity is a challenge for most organizations.

iii. **Variety** refers to nature of data. Data is available in varied formats and heterogenous sources. It can be structured, semi-structured and unstructured data. It is a challenge to find ways of governing, merging and managing these diverse forms of data.

- **Structured Data:** The data that is organized and has a formal structure such as in a row, column form at like ina excel sheet or relational tables is called structured data. This data can be stored in a relational database. Some examples of structured data are educational records, health records, bank statements, stock information, and geolocations etc.
- **Semi-structured Data:** This kind of data is semi organized i.e. it has no formal structure; it has features of both structured and unstructured data. It has a flexible structure, which means that data of any structure can be captured without making changes to the database schema or code. The functionality of the database is not affected by adding or removing data.While semi-structured data increases flexibility, the lack of a fixed schema also creates challenges in storage and indexing. The challenges increase as the volume of such data increases.Log files are categorized as semi-structured data. Other examples include emails, zipped files and webpages etc.
- **Unstructured Data**: This kind of data does not map to any standard. It cannot be fit into any standard fields. E.g. the data on social media platforms is unstructured data. It may contain both images and videos. Other examples of unstructured data are photos, video and audio files, social media content, satellite imagery, presentations, PDFs, open-ended survey responses, websites, data from IoT devices, mobile data, weather data, and conversation transcripts.

iv. **Veracity** here means quality of data, i.e. the extent to which data being used to analyse a problem is meaniful. In other words, by veracity, it meant how accurate the data is, or is the data trustworthy to be used for analysis. It refers to inconsistencies and uncertainty in data. The available data may be messy and thus controlling the quality and accuracy becomes another challenge. The analysis of a problem is totally dependent on the collected data. If the gathered data is inaccurate or incomplete or has any missing values, then it may not be able to provide real or valuable insight about the problem.Therefore, Veracity refers to the level of trust in the collected data. It is big challenge , because the accuracy of the analysis depends on the accuracy of the colleceted data. Therefore, it is very important to clean the data to get accurate analysis of data.

v. **Variability:** Variabilty means the extent to which the data varies or in other words it is the variation in data. In addition to the increasing velocities and varieties of data, data flows can be highly inconsistent with periodic peaks. Big Data is also variable because of the multitude of data dimensions resulting from multiple disparate data

types and sources e.g. Data in bulk could create confusion whereas less amount of data could convey half or Incomplete Information. The variation in data should be minimum i.e the difference between the real and idle situation must be minimum. Variability of data can be challenging to manage.

vi. **Value:** The bulk data having no value is of no good to the company, unless you turn it into something useful. Data in itself is of no use or importance but it needs to be converted into something valuable to extract information. The ability to transform the bulk data into business and make this enormous data of use to business, for monetization. It is a challenge to connect and correlate relationships, hierarchies and multiple data linkages of the big data.

Among all the other concerns, the biggest challenge in data science ids the security issue. The most common data security issue is data theft, especially for organizations that maintain sensitive data like financial information or customers‘ personal information. With the exponential increase in the usage of internet, enormous data is available , hence, data security is the of utmost importance. Hence, companies need to follow the three fundamentals of data security namely *Confidentiality*, *Integrity* and *Accessibility*.

1.4 FOUNDATION OF DATA SCIENCE

Data science involves using methods to analyse massive amounts of data and extract the knowledge it contains. Data Science is an interdisciplinary field focused on extracting knowledge from datasets which are large in size, applying the knowledge from data to solve problems in a wide range of application domains. Mathematics, statistics, computer science, and domain knowledge are the foundations of Data Science.

The main components of Data Science are given below:

1. Statistics: To analyse the large amount of numerical data and to find the meaningful insights from it, knowledge of statistics is required. Math and statistics expertise is a fundamental component of data science that allows practitioners to find meaningful patterns in data that yield actionable insights.

2. Domain Expertise: Data is available and applicable in various domains, therefore domain expertise or specialized knowledge of a specific area is required to get best results. There may be situations which demand taking the right decision, especially when dealing with messy data, it is very important to take the appropriate decision depending on the domain, because the same data may be used with a different approach in different domains, hence best approach cannot be known without the appropriate knowledge of the domain. Domain expertise is very important to get accurate results.

3. Data engineering: Data engineering is a part of data science, which involves acquiring, storing, retrieving, and transforming the data. Data engineering also includes metadata (data about data) to the data.

4. Visualization: Data visualization is meant by representing data in a visual context so that people can easily understand the significance of data. Data visualization makes it easy to access the huge amount of data in visuals.

5. Advanced computing: Advanced computing involves designing, writing, debugging, and maintaining the source code of computer programs. Machine Learning and Deep learning techniques are required for modelling and to make predictions about unforeseen/future data.

1.5 TOOLS FOR DATA SCIENCE

Some tools required for data science are as follows:

- **Data Analysis tools:** R, Python, Statistics, SAS, Jupyter, R Studio, MATLAB, Excel, RapidMiner.
- **Data Warehousing:** ETL, SQL, Hadoop, Informatica/Talend, AWS Redshift
- **Data Visualization tools:** R, Jupyter, Tableau, Cognos.
- **Machine learning tools:** Spark, Mahout, Azure ML studio.

1.6 APPLICATIONS OF DATA SCIENCE

Data Science is used by most organizations for predictive analysis, price optimization, and customer satisfaction. Some areas where data science is being used are Health Care, Finance, Security, Airline Routing, Manufacturing, Speech Recognition, Advertisement, Security, Fraud detection, Banking, Internet of Things etc. Some use cases are given below:

- Genomic Data provides deeper understanding of Genetic issues and reactions to particular drugs and diseases.
- Logistics companies like DHL,Fedex have discovered the best time and routes to ship cost effectively.
- Predict employee attrition and understand the variables that influence employee turnover.
- Airline companies can now easily predict flight delays and notify the passengers before time.
- Banks can make better decisions by predict risk analysis, fraud detection and better customer management

Data Science has created a strong foothold in several industries.Some case studies, not limited, where data science has been applied and has shown tremendous results are pharmaceuticals industries, manufacturing, production, scientific studies, education, travel etc

1.7 DATA SCIENCE PROCESS

Data Science is the area of study which involves extracting insights from vast amounts of data by the use of various scientific methods, algorithms, and processes. Data Science is an interdisciplinary field that requires expertise from various fileds like math, statistice, domain knowledge, computer programming etc to extract knowledge from structured or unstructured data. Data science provides a practical solution to a business problem. The *data science life cycle* is essentially comprises of **data collection**, **data cleaning**, **exploratory data analysis**, **model building** and **model deployment**. The structured approach to data science helps in maximizing the chances of success in a data science project at the lowest cost. The data science process typically consists of the following steps:

I. Business Understanding: To prepare a solution model for a problem, business understanding is most important. A good model can be designed to get a practical solution with the best understanding of the business. The main purpose here, is making sure all the stakeholders understand the what, how, and why of the project. It involves two main steps namely *defining research goals* and *preparing a project charter*.

i. Defining Research Goals: It is very essential to understand the business goals to be able to define the research goal that states the purpose of assignment in a clear and focused manner. The data by for the same can be gathered by repeatedly asking questions and enquiring about business expectations, identifying the research goals so that everybody knows what to do and can agree on the best course of action. The outcome should be a clear research goal, a good understanding of the context, well-defined deliverables, and a plan of action with a timetable. This information is then best placed in a project charter

ii. Creating Project Charter: A project charter has the information gathered while setting the research goal. The project charter must contain a clear research goal, the project mission and context, method for analysis, information about the resources required for project completion, proof that the project is achievable, or proof of concepts, deliverables and a measure of success and also a timeline for the project.

This information is useful to make an estimation of the project costs and the data and people required for the project to become a success.

II. Data Acquisition or Data Collection or Data Retrieval: This phase involves data gathering from various sources like webservers, logs, databases, API‘s and online repositories. It involves acquiring data from all the identified internal & external sources. One should start with data collection from internal sources. The data may be available in many forms ranging from simple text to database records. The data may be structured as well as unstructured. Data may be acquired from sources outsise the

organization also. Some sources may be available free of cost or some may be paid. Data collection is a tiresome and a time-consuming task.

III. Data preparation: Data collection is an error-prone process; in this phase, the quality of the data is enhanced and it is prepared for use in subsequent steps. This phase consists of three subphases: **data cleansing, data integration data transformation.**

i) ***Data Cleansing*** involves cleansing of data collected in the previous phase. It involves removing false values from a data source, removing inconsistencies across data sources, checking misspelled attributes, missing and duplicate values and typing errors, fixing capital letter mismatches as most programming languages are case sensitive (e.g India and india), removing redundant white spaces, sanity checks (check for impossible values like six-digit phone number etc) and outliers (an outlier is an observation that seems to be distant from other observations). It should be ensured that most errors are corrected in this phase to make the data usable and get better results.

ii) ***Data Integration*** enriches data sources by combining information from multiple data sources. Data comes from several sources and in this sub step the focus is on integrating these different sources. The data can be combined in the following ways:

- ***Joining Tables:*** To combine information about some data in one table with the information available in another table. e.g a table may contain information about purchase of a particular product and the other table may contain information about people who have purchased that product. This can be done using join command in SQL.
- ***Appending or Stacking:*** To add observations from one table to another table e.g. a table may contain the information about the purchase of a particular product in the year 2000 and the other contains the similar data in the year 2001, then appending means to add the records of 2001 to the table containing the records of 2000. This can be done using the union function in SQL.
- ***View:*** View in SQL can be used to virtually combine two tables. This saves on the space requirement
- ***Aggregate Functions:*** Aggregate functions may be used as per requirement for combining data.

iii) ***Data transformation*** ensures that the data is in a suitable format to be used in the project model. Sometimes, the number of variables may have to be reduced, It involves modification of data so that it takes a suitable shape. Tools like talend and informatica can be used for transformation.

IV. **Exploratory Data Analysis (EDA) or Data exploration:** Data exploration is concerned with building a deeper understanding of the data. It involves the understanding of how variables interact with each other, the distribution of the data, and whether there are outliers. It involves defining and refining the selection of feature variables that will be used in model development. This is the most important step as it involves understanding of the data which will further be used for modelling. Various techniques like visualization, tabulation, clustering, and other modelling techniques can be used for exploratory analysis.

V. **Data Modelling or Model building:** Model building is an iterative process. In this phase, domain knowledge, and insights about the data are used for modelling. It involves identifying the data model that best fits the business requirement. For this, various machine learning techniques like KNN, Naïve's, decision tree etc may be applied on data. The model is trained on the data set and the testing of the selected model is done. Data modelling can be done using Python, R, SAS. Most models consist of the following main steps:

   i. ***Selection of a modelling technique and variables to enter in the model:*** After the exploratory analysis, the variables to be used are known, so in this phase, the variables can be used to build the model. A model that suits the project requirement has to be chosen.
   ii. ***Execution of the model***: The chosen model has to be implemented by coding. Python is most used language for coding as it has many inbuilt libraries.
   iii. **Diagnosis and model comparison:** In this step, the best performing model or the model with lowest errors is chosen from among the multiple models that are built.

VI. **Presentation and automation:**After the data is analysed and a data model built for it, in this stage, the results are presented. These results can take many forms, ranging from presentations to research reports.

VII. **Deployment & Maintenance:** Before the final deployment in the production environment, testing is done in preproduction environment. After deployment, the reports are used for real time analysis.

The Data Science life cycle is an Iterative process, there is often a need to step back and rework certain findings. If the step 1(business understanding) is performed dedicatedly, rework can be prevented. The figure 1 below describes the Data Science model.

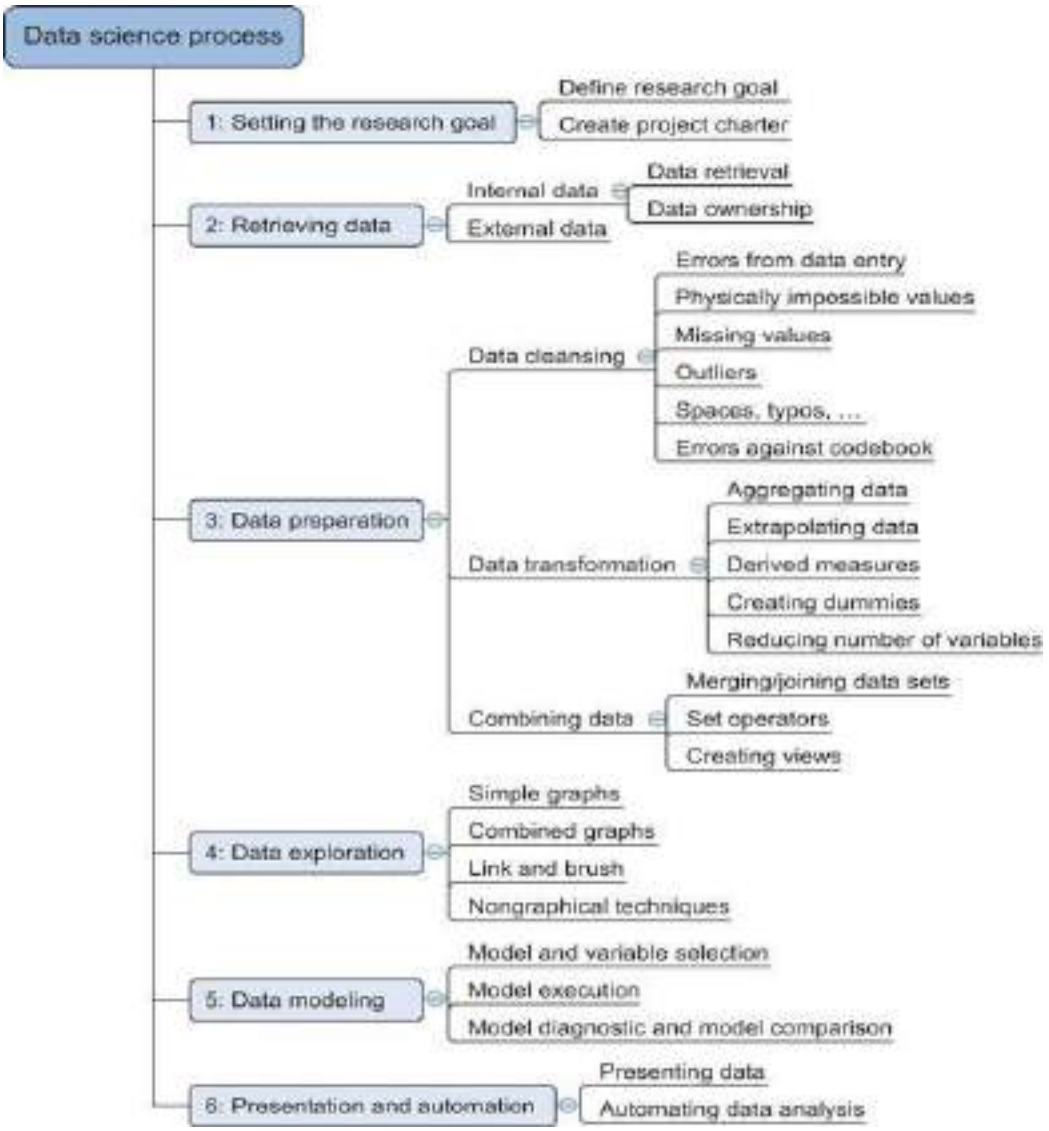


Fig1source: https://livebook.manning.com/book/introducing-data-science/chapter-2/8

1.8 MESSY DATA

Data Science has become an indispensable part of many businesses and industries. The backbone of data science is data. There has been a surge in the volume and variety of data, but not all the data is usable.The Data that is not in usable form or it is impossible to obtain clearly interpretable information from it is called messy data. In other words, incomplete, inaccurate, inconsistent, and duplicate data is called dierty or messy data. Some examples of messy data are missing data, unstructured data, multiple variables in one column, variables stored in wrong places, observations split incorrectly or left together against normalization rules, switched columns and rows, extra spaces etc. Data science and its algorithms are clean and precise, but the data on which they operate come from the real world, are inherently messy i.e. the data which requires some preparation before you can use them effectively and it is not easy to find clean data. The quality of insights you derive from data depends on the validity of that data, so some preparation is required.

1.8.1 Types of Messy Data

i. **Duplicate data** : A record that shares data with another record in the marketing database is called duplicate data. The database may contain a record that is a total replica of some record or it may be a partially duplicate. Partial duplicate records are the records that may contain the same name, phone number, email, or address as another record, but also contain other non-matching data. Sometimes partial duplicates are created by human error — when a customer or member of your team enters information manually. When duplicate *fields* increase, cleansing grows exponentially. Duplicate data is a big problem in a data-driven organization. It is very important to clean and get rid of duplicates in any data.

ii. **Outdated/Stale Data**: The information that is incorrect, incomplete, or no longer in use is referred as Outdated or stale data. Outdata is worst than having no data. It is very important to have updated data for decision making.Outdated data can have a very impact on the decisions of an organization.

iii. **Incomplete Data:** Incomplete dataanother kind of very common occurring dirty data. A record that lacks key fields on master data records such as industry type, title or last names, etc. which are useful for business is considered incomplete. For example imagine trying to sell geolocation software to a prospect who is located at ―N/A‖.

iv. **Inaccurate/Incorrect Data**: Collecting information about your customers helps in better understanding them and making informed decisions to satisfy them. This can be only possible if data is collected properly, completely, and accurately and can also lead to costly blunders.
Incorrect data: It occurs when the field values are generated outside of the valid range of values. For example, when filling a month field the range should encompass ranges from 1 to 12, or a zip code in an Indian city should be maximum 6 digits only.
Inaccurate data: There are many instances where the data on a field is correct but inaccurate considering the business context. Inaccurate data can lead to costly interruptions. For example, errors in a customer‘s address can lead to the delivery of the product at the wrong location even though the address on which it was delivered is correct.

v. **Inconsistent Data:** Inconsistent data are also known as data redundancy is when the same field value is stored in different places, which leads to inconsistency. For example, companies have customer information on multiple systems, and data is not kept in sync. The problem with inconsistent data can be explained, for example, if you want to target all ―Vice President‖ for an upcoming email marketing campaign. Since _V.P‘ _v.p‘ _VP‘ & _Vice Pres‘ all mean the same thing, however, these would only be included in the campaign if all these variations are included in the campaign list. Inconsistent data hinders analytics and makes segmentation difficult when you have to consider all variables of the same title, industry, etc.

1.9 ANOMALIES AND ARTEFACTS IN DATASETS

1.9.1 Anomaly is a deviation in data from the expected value for a metric at a given point in time. An unexpected change within these data patterns, or an event that does not conform to the expected data pattern, is considered an *anomaly*. Anomaly detection is any process that finds the outliers (items that do not belong to the dataset) of a dataset. The term anomaly is also referred to as outlier. Outliers are the data objects that stand out among other objects in the data set and do not conform to the normal behaviour in a data set. Anomaly detection is a technique for finding an unusual point or pattern in a given set. Anomaly detection is a data science application that combines multiple data science tasks like classification, regression, and clustering.There are three kinds of anomalies namely: point anomaly, contextual anomaly, and collective anomalies.

- **Point Anomaly:** If a single instance in a given dataset is different from others with respect to its attributes, it is called a point anomaly i.e. when a single instance of data is anomalous, it deviates largely from the rest of the set e.g. detecting credit card fraud based on ―amount spent.‖
- **Contextual anomaly:** If the data is anomalous in some context, it is called contextual anomaly. This type of anomaly is common in time-series data. In the absence of a context, all the data points look normal. E.g. if the context of the temperature is recorder in December and a high temperature reading is seen in December month, which is an abnormal phenomenon.
- **Collective anomalies** can be formed due to a combination of many instances i.e. a set of data instances collectively helps in detecting anomalies. For example, sequence data in network log or an attempt to copy data form a remote machine to a local host unexpectedly, an anomaly that would be flagged as a potential cyber-attack.

Anomaly detection refers to the problem of finding patterns in data that do not conform to expected behaviour. It is a technique for finding an unusual point or pattern in a given set. These nonconforming patterns are often referred to as anomalies, outliers, discordant observations, exceptions, aberrations, surprises, peculiarities, or contaminants in different application domains. Anomaly detection is commonly used for:

- Data cleaning
- Intrusion detection
- Fraud detection
- Systems health monitoring
- Event detection in sensor networks
- Ecosystem disturbances

1.9.2 Artifact It is a data flaw caused by equipment, techniques or conditions. Common sources of data flaws include hardware or software errors, conditions such as electromagnetic interference and flawed designs such as an algorithm prone to miscalculations. Some common data artifacts are:

- **Digital Artifacts:** Flaws or errors in digital media, documents and data records caused by data processing e.g a distorted camera recording.
- **Visual Artifacts:** Flaws in visualizations such as user interfaces.
- **Compression Artifacts**: Flaws in data due to lossy compression.
- **Statistical Artifacts**: Flaw such as a bias in statistical data.
- **Sonic Artifacts**: Unwanted sound in a recording.

1.10 CLEANING DATA

Data cleaning is a key part of data science. Clean data increases overall productivity and allows for the highest quality information in decision-making. Data cleaning is the process of fixing or removing incorrect, corrupted, incorrectly formatted, duplicate, or incomplete data within a dataset. If data is messy, the outcomes and algorithms are unreliable. It can lead to poor business strategy and decision-making. The data can have many irrelevant and missing parts. To handle this part, data cleaning is done. It involves handling of missing data, noisy data etc. During cleansing, missing values may either be filled or removed depending on the data. There are many other types of errors that may need to be cleaned in dirty data. These errors can include:

- missing data
- unstructured data
- multiple variables in one column
- variables stored in the wrong places
- observations split incorrectly or left together against normalization rules
- switched columns and rows
- extra spaces

1.10.1 Challenges with Dirty Data:

- **Time-consuming:** Dirty data needs to be cleaned to make is usable, analysts spend up to 80% of the total analysis process cleaning dirty data. It is not only time consuming but expensive also.
- **Technical Constraints**: In the case of more advanced data projects, organizations must hire costly data scientists or data engineers with advanced programming skills—

only to have them spend the majority of their time cleaning data. Programming languages are effective in wrangling big, complex data, but they limit data preparation to a small group of people, which creates a big bottleneck.

- **It's error-prone:** Datasets need to be prepared effectively as well as efficiently. There is a possibility of errors to remain until reviewed by another person it's often difficult (or near impossible) for others to revise the data cleaning work.

1.10.2 Data Cleansing Techniques

The choice of data cleaning techniques relies on a lot of factors like the kind of data being dealth with. Whether the data is numeric values or strings. Multiple techniques may be used for cleansing different kinds of data to get better results. Some dat cleansing activities are:

- **Removing Irrelevant Values**

The first and foremost thing is to remove irrelevant data from your system. Any useless or irrelevant data is the onethat is not required for analysis. It might not fit the context of the problem. E.g to measure the average age of sales staff, their email address are not required and are hence considered irrelevant or useless in this context.

- **Remove Duplicate Values**

Duplicate data only increases the volume of data and tends to waste time. Duplicate values could be present in your system for several reasons like you may have combined the data of multiple sources or, perhaps the person submitting the data repeated a value mistakingly. Some user clicked twice on _enter' when they were filling an online form. This duplicate data should be removed as soon as it is found.

- **Avoid Typos**

Typos are a result of human error. They are essential to fix because models treat different values differently. Strings rely a lot on their spellings and cases. Another error similar to typos is of strings' size. They might need to be padded to maintain the format. For example, a dataset might require to have 5-digit numbers only. So if you have any value which only has four digits such as _1234', zero must be added in the beginning to increase its number of digits as _01234'. An additional error with strings is of white spaces. Extra white spaces should be removed.

- **Convert Data Types**

Data types should be uniform across the dataset. The following things should be kept in mind when converting data types:

  - Keep numeric values as numerics
  - Check whether a numeric is a string or not. If you entered it as a string, it would be incorrect.
  - If you can't convert a specific data value, you should enter _NA value' or something of this sort.

- **Take Care of Missing Values**

Missing values can lead to wrong interpretations. Therefore, missing values should be handled well. A particular column in your dataset may have too many missing values. The decision whether to delete the misiing values or to fill them is based on the domain knowledge. Sometimes, it is recommended to get rid of missing values, but sometimes it is not advisable depending upon the problem situation, but missing values shouldn‘t be ignored because they will contaminate data, and hamper the accuracy of the results. There are multiple ways to deal with missing values:

Imputing Missing Values: means assuming the approximate value. Methods like linear regression or median can be used to calculate the missing value. Another method to impute missing values is to copy the data from a similar dataset. This method is called _Hot-deck imputation‘.

Highlighting Missing Values: Imputation isn‘t always the best measure to take care of missing values. Many experts argue that it only leads to more mixed results as they are not _real‘. So, you can take another approach and inform the model that the data is missing. The missing valurs can be highlighted or flagged. For example, your records may not have many answers to a specific question of your survey because your customer didn‘t want to answer it in the first place. If the missing value is numeric, you can use 0. Just make sure that you ignore these values during statistical analysis. On the other hand, if the missing value is a categorical value, you can fill _missing‘.

1.11 SUMMARY

This chapter introduces the concept of data science as a discipline. The various issues faced due to the growth in data are discussed. The importance of cleaning data and the anomalies found in data are also discussed.

1.12 PRACTICE QUESTIONS

Q1. What are the foundations of data science?

Q2. Discuss the areas where data science is applicable?

Q3. Discuss the various challenges due to growth in data.

Q4. Explain briefly the data Science Process.

Q5. Define Messy data.

INTRODUCTION TO DATA SCIENCE

UNIT II: DATA ACQUISITION AND PROCESSING

STRUCTURE

2.0 OBJECTIVES

1. Introduction to Data Acquisition

2. To familiarize with the different types of data

3. To provide the concept of data pre-processing and familiarize with the various reprocessing techniques

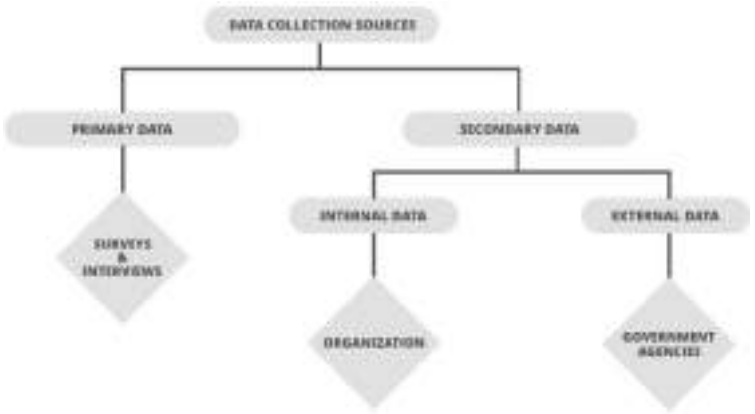
4. To familiarize with the concept of data mining and data interpretation

2.1 INTRODUCTION TO DATA ACQUISITION

Data acquisition or data collection is the process of acquiring, collecting, extracting, and storing the voluminous amount of data which may be in the structured or unstructured form like text, video, audio, XML files, records, or other image files. In other words, the process of gathering data and making it useful by techniques like filtering and cleaning as per the business requirement is termed as data acquisition. It is the most important step or we can say it is the initial step to be performed before beginning of the process of big data analysis. The data must be acquired from different valid sources. It must be suitable to the problem in hand. It is also important to acquire up to date data to get accurate results.

The actual data is then divided mainly into two types known as:

i. Primary data
ii. Secondary data



i. Primary Data:

The data which is raw, original, and is extracted directly from the official sources is known as primary data. This type of data is collected directly by performing techniques such as questionnaires, interviews, and surveys. The data collected must be according to the demand and requirements of the problem on which analysis is performed so as to get the accurate analysis of the problem. Some methods of collecting primary data:

a) Interview method

The data collected during this process is through interviewing the target audience by a person called interviewer and the person who answers the interview is known as the

interviewee. The answers to the questions asked are noted down in the form of notes, audio, or video and this data is stored for processing. This data can be both structured and unstructured like personal interviews or formal interviews through telephone, face to face, email, etc.

b) Survey method

The survey method is the process of research where a list of relevant questions are asked and answers are noted down in the form of text, audio, or video. The survey method can be obtained in both online and offline mode like through website forms and email. Then that survey answers are stored for analyzing data. Examples are online surveys or surveys through social media polls.

c) Observation method

In this method, the researcher keenly observes the behavior and practices of the target audience using some data collecting tool and stores the observed data in the form of text, audio, video, or any raw formats. In this method, the data is collected directly by posting a few questions on the participants. For example, observing a group of customers and their behavior towards the products. The data obtained will be sent for processing.

4. Experimental method:

The experimental method is the process of collecting data through performing experiments, research, and investigation. The most frequently used experiment methods are CRD, RBD, LSD, FD.

- **CRD- Completely Randomized design** is a simple experimental design used in data analytics which is based on randomization and replication. It is mostly used for comparing the experiments.
- **RBD- Randomized Block Design** is an experimental design in which the experiment is divided into small units called blocks. Random experiments are performed on each of the blocks and results are drawn using a technique known as analysis of variance (ANOVA). RBD was originated from the agriculture sector.
- **LSD – Latin Square Design** is an experimental design that is similar to CRD and RBD blocks but contains rows and columns. It is an arrangement of NxN squares with an equal amount of rows and columns which contain letters that occurs only once in a row. Hence the differences can be easily found with fewer errors in the experiment. Sudoku puzzle is an example of a Latin square design.
- **FD- Factorial design** is an experimental design where each experiment has two factors each with possible values and on performing trail other combinational factors are derived.

ii. Secondary Data

Secondary data is the data which has already been collected and reused again for some valid purpose. This type of data is previously recorded from primary data and it has two types of sources named internal source and external source.

- **Internal source**

These types of data can easily be found within the organization such as market record, a sales record, transactions, customer data, accounting resources, etc. The cost and time consumption is less in obtaining internal sources.

- **External source**

The data which can‘t be found at internal organizations and can be gained through external third party resources is external source data. The cost and time consumption is more because this contains a huge amount of data. Examples of external sources are Government publications, news publications, Registrar General of India, planning commission, international labor bureau, syndicate services, and other non-governmental publications.

Other sources:

- **Sensors data:** With the advancement of IoT devices, the sensors of these devices collect data which can be used for sensor data analytics to track the performance and usage of products.
- **Satellites data:** Satellites collect a lot of images and data in terabytes on daily basis through surveillance cameras which can be used to collect useful information.

The characteristics of big data namely volume, velocity, variety, and value and very important for the acquisition of data. In data science, there is a variety of data that we need to deal with. The data is majorly characterized as:

- **Structured Data:** The data which is available in some stand format is called structured data. Structured data is organized data. It is stored in the row and column structure like in a relational database and excel sheets. Some examples of structured data are names, numbers, geolocations, addresses etc.

- **Unstructured Data**: This data is unorganized data. There is no particular format for unstructured data. It is available in a variety of formats like texts, pictures, videos etc. Unstructured data is more difficult to search and requires processing to become understandable.

- **Semi- Structured data:** This data is basically a semi-organised data. It is generally a form of data that do not conform to the formal structure of data. Log files are the examples of this type of data.

2.2 DATA PREPROCESSING AND TECHNIQUES

The data acquired needs to be preprocessed to make it usable. The raw data may be inconsistent, erroneous, incomplete and not in the required format. These issues need to be resolved to make the data usable. Data Preprocessing is a collaborative term used for the activities involved in transforming the real-world data or raw data into a usable form to make it more valuable and to get it in the required format. The preprocessed data is cleaner and more valuable, and hence used as final training set. Data preprocessing is a very essential step, as more clean and inconsistent data we get, better shall be the final output. In other words, data processing improves the quality of data. The huge size of data collected from heterogenous sources leads to anomalous data. Data preprocessing has become a vital and most fundamental step considering the fact that high quality data leads to better models and predictions. Data preprocessing techniques are majorly categorized in the following methods:

i. Data Cleaning
ii. Data Integration
iii. Data Transformation
iv. Data Reduction

2.2.1 Data Cleaning involves removing duplicate data, filling missing values, identifying outliers, smoothening noise and remove data inconsistencies. The following steps must be followed for cleaning data:

- **Remove duplicate or irrelevant observations**
  It is very important to remove inconsistencies in dataset like removing unwanted observations including duplicate and irrelevant data entries. Duplicate values may be caused at the time of data collection e.g. the names of the countries may have repeated valued like *NewZealand* and *New Zealand*; *Pakistan* and *pakistan*. Inconsistencies in capitalizations and trailing white spaces are very common in text data, The data set can be cleaned using available function in Python or R. functions like unique, sort, lower(), upper(), strip() can be used to handle inconsistencies

- **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 mislabelled categories or classes. For example, you may find ―N/A‖ and ―Not Applicable‖ both appear, but they should be analysed as the same category. Set the single date format in case there are multiple date formats in a single column.
- **Filter unwanted outliers** is essential to improve the performance of the data. Outliers are the data objects that stand out among other objects in the data set and do not conform to the normal behaviour in a data set. If an outlier is found to be irrelevant it must be handled. Some techniques to remove outliers from huge datasets are: boxplots,Z-score and interquantile range. The outliers need to be handled as per the problem. They can be either removed or the outlier is capped or else it can be imputed.
- **Handle missing data:** During analysis of data, it is important to understand why the missing values exist. Whether the missing value exist because they were not recorded or they imply that data does not exist for the missing values. Missing data may be handled by using the following ways:

  i) In case, the values were not recorded, it becomes essential to fill up the values by guessing based on the other values in that column and row. This is called ***imputation.*** E.g. if a value is missing for gender, it is understood that the value was not recorded, so we need to analyse data and give it a value, we cannot leave the value blank in this case. Therefore, you can input missing values based on other observations; but there is a chance of losing integrity of the data because these values are being filled based on assumptions and not actual observations.

ii) If a value is missing because it doesn't exist e.g. the height of the oldest child of someone who doesn't have any children. In this case, it would make more sense to either leave it empty or to add a third value like *NA* or *NaN*. The NA or NaN values should be replaced with 0. Panda's fillna() function to fill in missing values in a data frame can be used.

iii) In case, it is not possible to figure out the reason for the missing, then that particular value can be dropped. Pandas function, dropna() can be used to do this, but doing this can drop or lose information, so be mindful of what is being removed or dropped this before removing it.

- **Noisy Data:** Random variance in the data is called noise. It is generated due to faulty data collection or errors in data collection. It adversely affects the analysis of the data.The following methods called smoothening techniques are used to handle noisy data:

  a) Binning Method: This method is also known as discretization, is used to smooth sorted data values by consulting the values around it i.e., the neighbouring values. It is a local smoothening method, since it refers to the neighbouring values. In this method, the entire data is divided into equal segments called bins or buckets. Smoothing by binning is done by one of the following methods:

  - **Smoothening by Bin Means:** In smoothing by bin means, each value in a bin is replaced by the mean value of the bin.
  - **Smoothening by Bin Median:** In smoothing by bin means, each value in a bin is replaced by the median value of the bin.
  - **Smoothening by Bin Boundary:** In smoothing by bin boundaries, the minimum and maximum values in a given bin are identified as the bin boundaries. Each bin value is then replaced by the closest boundary value.

Algorithm:

1. Sort the dataset.
2. Partition the dataset into ‗ n‘ segments. Each segment should contain approximately same number of data elements. Partitioning can be done using either of the following methods namely: equal width binning, equal frequency binning or entropy-based binning.
3. Calculate the arithmetic mean or media or replace by boundary (min and max value)
4. Replace each data element in each bin by the calculated mean/ median/ boundaries.

Example: Given data:18,22,6,6,9,14,20,21,12,18,18,16. Illustrate binning by mean, median and boundary replacement. Given bin depth=3

Step1: Sort the data: 6,6,9,12,14,16,18,18,18,20,21,22,

Step 2: Partition the data into equal frequency bins of size of bin depth(n/d) where n= no. of elements and d= bin depth

N/D=12/3=4 bins

Bin 1: 6, 6 , 9
Bin 2: 12,14, 16
Bin 3: 18,18,18
Bin 4: 20, 21, 22

Step 3: Calculate Arithmetic mean

Arithmetic mean= Sum of observations ÷ number of observations

Bin 1= (6+6+9)/3=21/3=7
Bin 2= (12+14+16)/3= 42/3=14
Bin 3=(18+18+18)/3= 54/3=18
Bin 4= (20+21+22)/3= 63/3=21

Step 4: Replace each data element in each bin by the calculated mean

Bin 1: 7, 7, 7
Bin 2: 14,14,14
Bin 3: 18,18,18
Bin 4: 21, 21, 21

- **Binning using Median:** In this method, Step 1 and step 2 are same.

Step 3: Calculate Median (50% percentile)

Median is the observation 2 in each bin

Step 4: Replace each data element in each bin by the calculated mean

Bin 1: 6, 6, 6
Bin 2: 14,14,14
Bin 3: 18,18,18
Bin 4: 21, 21, 21

- **Binning using Boundary Values:** In this, we keep the minimum as well as maximum values.

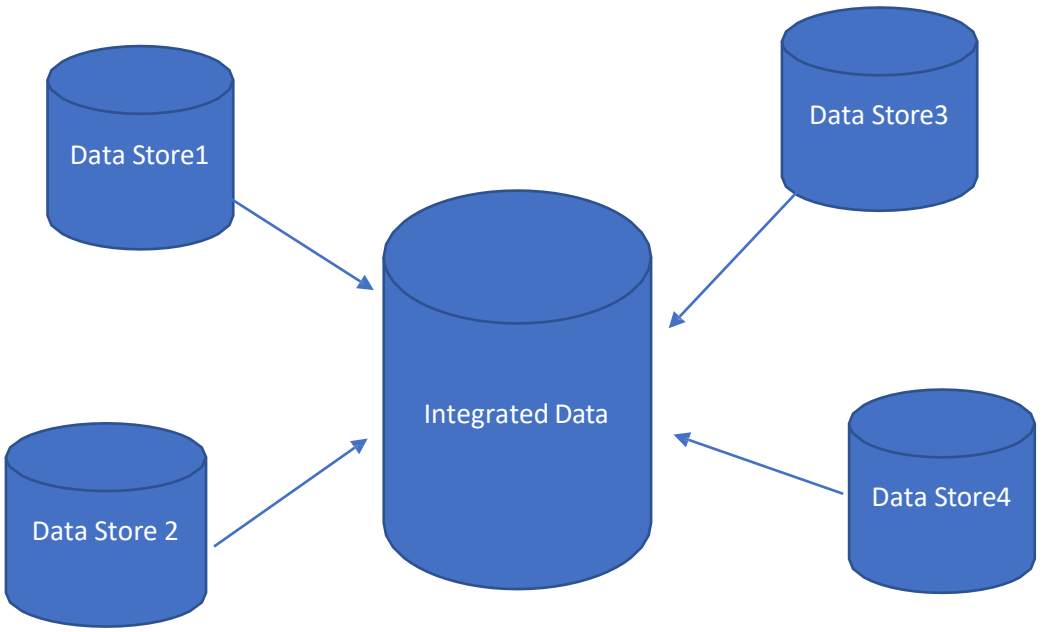
Bin 1: 6, 6, 9
Bin 2: 12,12,16
Bin 3: 18,18,18
Bin 4: 20, 20, 22

b) Regression: Regression is used to find a mathematical equation to fit the data to smooth out the noise. Regression may be linear or multiple. Linear regression is used to find the best line that fits two variables such that one predicts the other. Multiple linear regression involves more than two variables are involved.

c) Clustering: Cluster analysis in data mining refers to detecting out the category of things that are identical to each other in the category but are discrete from the things in another category. Clustering assists to divide information into numerous groups. This approach is useful for organizing similar data in groups or clusters. The outliers may be detected by clustering. Data Integration makes data more comprehensive and more usable.

2.2.2 Data Integration

Data Integration is a vital step in which the data acquired from multiple sources is integrated into a single centralized location. Data Integration makes data comprehensive and more usable.



The most common approaches to integrate data are:

a) Data Consolidation
b) Data Propagation
c) Data Virtualization
d) Data Warehousing

a) Data consolidation means to consolidate data from several separate sources into one data store, so that it is available to all the stake holders to enable better decision making. It involves eliminating redundancies, removing inaccuracies before consolidating to a single store. The most common data consolidation techniques are ETL (Extract, Transform, Load), Data virtualization and Data warehousing.

- **ETL** is the most widely used data consolidation technique. The data is first extracted from multiple sources, then it is transformed into an understandable format by using various functions like sorting, aggregating, cleaning etc and then

transfer it to a centralized store like another database or data ware house. The ETL process cleans, filters, and transforms data, and then applies business rules before data populates the new source. ETL is further of two types namely Real time ETL used in real time systems and Batch processing ETL used for high volume databases.

- **Data Virtualization:** In this method, the data stays in the original location, but changes are made in a virtual manner and can be seen in a consolidated manner by the users. It is a logical layer that amalgamates data from various sources without performing actual ETL process. It is an abstraction such that only the required data is visible to the users without requiring technical details about the location or structure of the data source. It provides enhanced data security.
- **Data Warehousing** is the integration of data from multiple sources to a centralized source to facilitate decision making, reporting and query handling. A centralized source of data enables better decision making. Data warehousing is the secure electronic storage of information by a business or other organization. The goal of data warehousing is to create a trove of historical data that can be retrieved and analyzed to provide useful insight into the organization's operations.

b) Data Propagation involves copying data from one location i.e., source to another location i.e., target location. It is event driven. These applications usually operate online and push data to the target location. They are majorly useful for real time data movement such as workload balancing, backup and recovery. Data propagation can be done asynchronously or synchronously.

The two methods for data propagation are: Enterprise Application Integration (EAI) and Enterprise Data Replication (EDR). The key advantage of data propagation is that it can be used for real-time / near-realtime data movement and can also be used for workload balancing, backup and recovery. EAI is used majorly for the exchange of messages and transactions in real-time business transaction processing; whereas for transfer of voluminous data between databases, is used.

c) Data Virtualization: In this, data is not stored in a single location, but is abstracted and can be viewed as unified view of data from multiple sources. Data Federation is a form of data virtualization, supported by Enterprise information technology (EII). EII products have evolved from two different technological backgrounds – relational DBMS and XML, but the current trend of the industry is to support both approaches, via SQL (ODBC and JDBC) and XML (XML Query Language - XQuery - and XML Path Language - XPath) data interfaces.

d) Data Warehousing is the integration of data from multiple sources to a centralized source to facilitate decision making, reporting and query handling. A centralized source of data enables better decision making.

2.2.3 Data Transformation

After the data has been acquired, it is cleaned as discussed above. The clean data may not be in a standard format. The process of changing the structure and format of data to make it more usable is called data transformation. Data transformation may be constructive, destructive, aesthetic or structural.

- *Constructive Data Transformation* involves adding, copying, and replicating data.
- *Destructive Data Transformation* involves deleting fields and records.
- *Aesthetic Data Transformation* involves standardizing salutations or street names.
- *Structural Data Transformation* involves renaming, moving, and combining columns in a database.

Data Transformation may require smoothing, aggregation, discretization, attribute Construction, generalization and normalization to make data manageable.

Scripting languages like Python or domain-specific languages like SQL are usually used for data transformation.

2.2.4 Data Reduction

It is the process of reducing the volume of data such that data integrity is preserved. i.e., the volume of data is reduced but the results of data mining before and after mining are the same. Data reduction increasing the efficiency of data mining. It helps in reducing the storage requirement, reduced computation time and removal of redundancy. The various techniques used for data mining are: *Dimensionality reduction, numerosity reduction and data compression.*

2.2.4.1 Dimensionality Reduction: is the transformation of data from high dimensional space to a low dimensional space. It is difficult to visualize data that has higher number of features. Sometimes, most of these features are correlated, and hence redundant. This is where the need of dimensionality reduction arises. In other words, dimensionality reduction is the process of reducing the number of random variables under consideration, by obtaining a set of principal variables. e.g., 3-dimensional data may be reduced to a 2D data. The various methods used for dimensionality reduction include:

- **Wavelet Transformation** is mostly used in image compression. It is a lossy method for dimensionality reduction, where a data vector X is transformed into another vector X‘, such that both X and X‘ represent the same length. The result of wavelet transform can be truncated, unlike its original, thus achieving dimensionality reduction. Wavelet transforms are well suited for data cube, sparse data or data which is highly skewed.

- **Principal Component Analysis (PCA**) is applied to skewed and sparse data. In this method, the entire data set is represented by few independent tuples with ‗n‘

attributes. This method was introduced by Karl Pearson. It works on a condition that while the data in a higher dimensional space is mapped to data in a lower dimension space, the variance of the data in the lower dimensional space should be maximum. It involves the following steps:

i. Construct the covariance matrix of the data.
ii. Compute the eigenvectors of this matrix.
iii. Eigenvectors corresponding to the largest eigenvalues are used to reconstruct a large fraction of variance of the original data.

Hence, lesser number of eigenvectors are left.

- **Attribute Subset Selection:** In this method, a subset of some selected attributes is created for reducing the volume of data. The goal of attribute subset selection is to find a minimum set of attributes such that dropping of those irrelevant attributes does not much affect the utility of data and the cost of data analysis could be reduced.
  Attribute Subset Selection is done by the following methods:
  1. Stepwise Forward Selection.
  2. Stepwise Backward Elimination.
  3. Combination of Forward Selection and Backward Elimination.
  4. Decision Tree Induction.

2.2.4.2 Numerosity Reduction: is the reduction of original data and its representation in a smaller form. It can be done in two ways: parametric and non-parametric numerosity reduction.

i) **Parametric Numerosity Reduction:** In this method, only the data parameters are stored, instead of the entire original data. The data is represented using some model to estimate the data, so that only parameters of data are required to be stored, instead of actual data. Regression and Log-Linear methods are used for creating such models.

ii) **Non- Parametric Numerosity Reduction** methods are used for storing reduced representations of the data include *histograms*, *clustering*, *sampling* and *data cube aggregation.*

- **Histogram** is the data representation in terms of frequency. It uses binning to approximate data distribution and is a popular form of data reduction.
- **Clustering** divides the data into groups/clusters. This technique partitions the whole data into different clusters. In data reduction, the cluster representation of the data are used to replace the actual data. It also helps to detect outliers in data.
- **Sampling** can be used for data reduction because it allows a large data set to be represented by a much smaller random data sample (or subset).

- **Data Cube Aggregation** involves moving the data from detailed level to a fewer number of dimensions. The resulting data set is smaller in volume, without loss of information necessary for the analysis task.

2.2.4.3 Data Compression is a technique in which the original data is compressed for reduction. This compressed data can again be reconstructed to form the original data. If the data is reconstructed without losing any information, then it is a _lossless' data reduction.

2.3 DATA MINING

Data mining is the process that helps in extracting information from a given data set to identify trends, patterns, and useful data. The objective of using data mining is to make data-supported decisions from enormous data sets. Different types of data can be mined such as Data stored in database, data warehouse, transactional data and other types of data such as data streams, engineering design data, sequence data, graph data, spatial data, multimedia data, and more. The data mining process breaks down into five steps:

1. An organization collects data and loads it into a data warehouse.
2. The data are then stored and managed, either on in-house servers or in a cloud service.
3. Business analysts, management teams, and information technology professionals access and organize the data.
4. Application software sorts the data.
5. The end-user presents the data in an easy-to-share format, such as a graph or table.

In recent data mining projects, various major data mining techniques have been developed and used, including association, classification, clustering, prediction, sequential patterns, and regression.

- **Association** is a data mining technique useful to discover a link between two or more items. It uses if-then statements to show the probability of interactions between data items within large data sets. It finds a hidden pattern in the data set. It is commonly used to help sales correlations in data or medical data sets.
- **Clustering** is a data mining technique in which clusters are created of objects that share the same characteristics. Clusters relate to hidden patterns. It is based on unsupervised learning.
- **Classification** classifies items or variables in a data set into predefined groups or classes. It is based on unsupervised learning.
- **Prediction** is used in predicting the relationship that exists between independent and dependent variables as well as independent variables alone. It can be used to predict future trends. It analyses past events or instances in the right sequence to predict a future event.
- **Sequential patterns** is a data mining technique specialized for evaluating sequential data to discover sequential patterns. It comprises of finding interesting subsequence in a set of sequences, where the stake of a sequence can be measured in terms of different criteria like length, occurrence frequency, etc.

- **Regression:** Regression analysis is the data mining process is used to identify and analyze the relationship between variables because of the presence of the other factor. It is used to define the probability of the specific variable. Regression, primarily a form of planning and modeling. For example, we might use it to project certain costs, depending on other factors such as availability, consumer demand, and competition. Primarily it gives the exact relationship between two or more variables in the given data set.

2.3.1 Data Mining Applications

Data mining is useful in predictions, classification, clustering and trends, which makes it applicable to many domains like health care, fraud detection, education, market basket analysis, manufacturing, sales, customer relationship management, finance and banking etc.

2.4 DATA INTERPRETATION

The process of reviewing data through some predefined processes to be able to assign some meaning to the data and arrive at a relevant conclusion is called data interpretation. It involves taking the result of data analysis, making inferences on the relations studied, and using them to reach conclusions and develop recommendations. Data interpretation is done by analyst to make inferences from the data. There are two ways to interpret data namely Qualitative and Quantitative.

Qualitative Data Interpretation: Qualitative data also called categorical data, does not contain numbers, it consists of text, pictures, observations, symbols etc. The interpretation of patterns and themes in qualitative data is done using qualitative methods.

Quantitative Data Interpretation is done on quantitative data i.e. numerical data. It involves statistical methods such as mean, standard deviation, variance, frequency distribution, regression etc. Data analysis and interpretation, regardless of method and qualitative/quantitative status, may include the following characteristics:

- Data identification and explanation
- Comparing and contrasting of data
- Identification of data outliers
- Future predictions

Data Interpretation is helpful in improving processes by identifying problems. Data interpretations is used for predicting trends after studying the patterns in the data. It is helpful in better decision making.

2.5 SUMMARY

This chapter introduces the various facets of data and the varius data preprocessing techniques. A brief introduction to dat mining and data interpretation is provided in this chapter.

2.6 PRACTICE QUESTIONS

Q1. What are the various types of data?
Q2.Explain the concept of data preprocessing. What are the techniques used for preprocessing of data?
Q3. Write a short note on Data mining.
Q4. What are the techniques used for data mining?
Q5. What is the importance of data Interpretation?

INTRODUCTION TO DATA SCIENCE

UNIT III: REPRESENTATION OF DATA

STRUCTURE

3.0 OBJECTIVES

1. To familiarize with the data representation and types of data
2. To familiarize with the general techniques of handling data

3.1 BIG DATA ANALYTICS

The assortment of any kind of data which is large, highly complex and difficult to manage and handle in the real time scenario with the traditional management techniques becomes the Big Data. The most extensively technique for the management of data was Relational DBMS which was a kind of dataset that fits in all the scenarios for the storage, manipulation and interpretation of data but it failed in case of big data. A groundbreaking study in 2013 reported 90% of the entirety of the world‘s data has been created within the previous two years. In the previous two years researchers have composed and handled data which is 9 times the data being processed in the previous 1000 years of the survival. As per the statistical institute, already 2.7 zettabytes of data have been generated and by 2025 it might escalate to a limit beyond belief which ca be 100zettabytes or even more.

What do we do with all of this data? How do we make it useful to us? What are its real-world applications? These questions are the domain of data science.

3.2 DATA SCIENCE - WORKING

In order to handle the query for complete in depth and a sophisticated approach for the exploration of the raw unprocessed data into the real time multiple disciplines with the expertise in machine learning and AI based dimensions, data science gives the solution with the most advanced means and methods. The scrutiny of the relevant information from the irrelevant raw data and pass on or forward only the most vital data for the enhancing the computing efficiency with the revolution in the fields of engineering, mathematics, statistics, advanced computing and visualizations. The data science is the basic field of exploration of data where the researchers or the data scientists also rely heavily on artificial intelligence for the creation of simulated models and then predicting the values of the parameters with the algorithms designed for manipulation of data into information. The working in the field of the data science is thereby based on the types of data being explored and manipulation needed e.g., if the simple data in form of tables is available then RDBMS is used and if image data is present then RDBMS fails.

3.3 FACETS OF DATA

In data science and big data, the data required for the manipulation and interpretation changes with the change in the types of tools and techniques applied in the algorithm. The types of the data explored in the field of data science can be categorized as follows:

- Structured
- Unstructured
- Natural language
- Machine-generated
- Streaming

- Acoustic data, video, and images
- Sensor Data
- Graph-based/ Network data

As in the previous chapter we have explored the data categories structured versus unstructured data. Let‘s talk about the rest of the representation of data and the explore the characteristics. The structured data is the main type of data explored in a model and can be expressed as a record with a field of fixed length and dimension. The RDBMS and the Excel data is usually the structured data with known or used defined datatypes with structural information. Bu the unstructured data is data that may not fit any kind of mathematical or simulated model for the study of data due to the context-specific or varying nature of the stored raw data. One example of unstructured data is the email.

3.3.1 Natural Language Data

Another special type of unstructured data is the Natural language data with the challenging approach to progression the data and manipulate it. The handling of large amount of natural language data and processing in itself necessitates the data scientists to acquire the knowledge of specific data science techniques and linguistics. The natural language

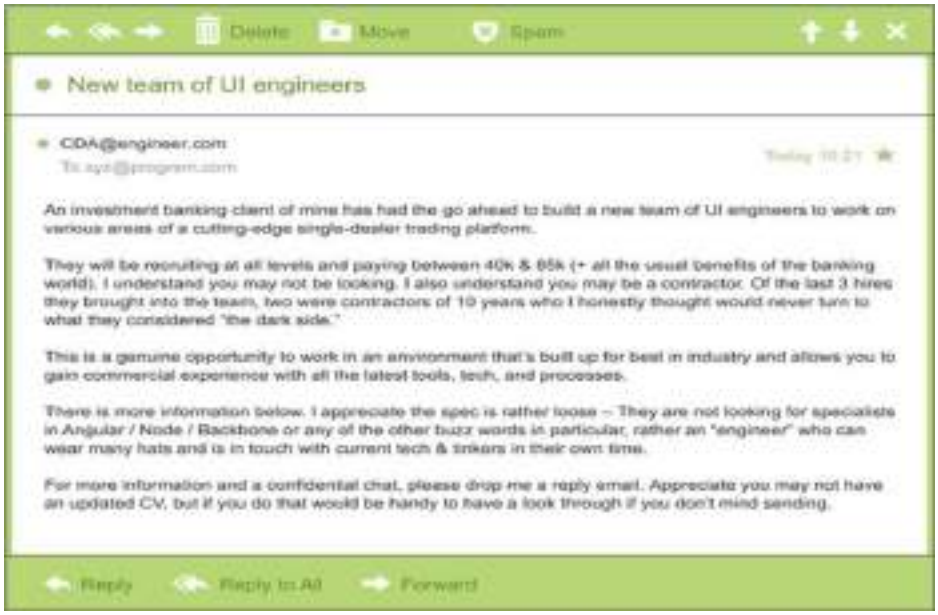


Fig. 3.1.1 Perfect example of Natural Language data [12]

processing community has had success in entity recognition, topic recognition, summarization, text completion, and sentiment analysis, but models trained in one domain don‘t generalize well to other domains. Even state-of-the-art techniques aren‘t able to decipher the meaning of every piece of text. The meaning of the same words can vary when coming from someone upset or joyous. This is illustrated in figure 3.1.

3.3.2 Machine Generated Data

The machine or the computer has the capability to generate the data based on the requirement of the developer or the application developed. The data generated by the machine or any robotic process without the intervention of a human is a fast process and also help to forecast certain information for the application, or other machine without human intervention. Machine-generated data is becoming a major data resource and will continue to do so. Wikibon has forecast that the market value of the industrial Internet (a term coined by Frost & Sullivan to refer to the integration of complex physical machinery with networked sensors and software) will be approximately $540 billion in 2020. IDC (International Data Corporation) has estimated there will be 26 times more connected things than people in 2020. This network is commonly referred to as the internet of things.The analysis of machine data relies on highly scalable tools, due to its high volume and speed. Examples of machine data are web server logs, call detail records, network event logs, and telemetry. This is illustrated by figure 3.2.

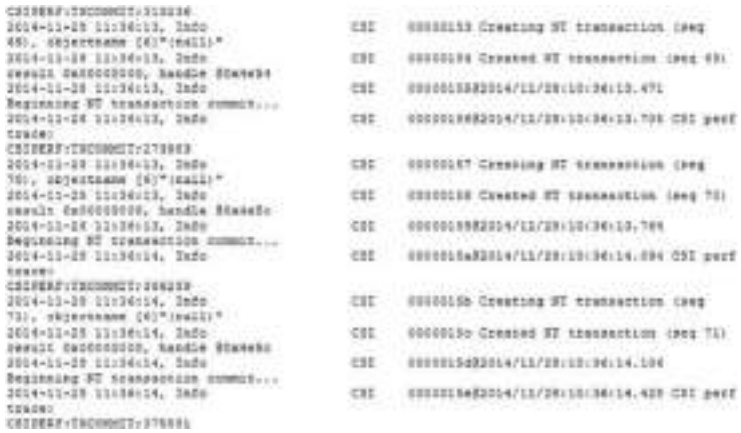


Fig. 3.1.2 Example of machine generated data [12]

3.3.3 Streaming Data

The streaming data when being administered for the information can take nay form and format which is an interesting property. This type of data only gets loaded into the server or the data warehouse when any relevant activity occurs or there‘s an interpretable change in the parameters. The data is not accessed in batch mode. This type of data is not actually a different category but the system for the processing of such varying formats needs to be adaptive to the minutest variations in the information to be handled.

Examples are the ―What‘s trending‖ on Twitter, live sporting or music events, and the stock market.

3.3.4 Acoustic, video and image data

The world of animation and digitization has developed multimedia datasets with audio, video and images. These types of datsets can be stored, handled and interpreted with the Object-Oriented Databases. The databases include the class inheritance and interface in a pre-specified format for handling the complexity of the data and finding its application in Digital libraries, video-on demand, news-on demand, musical database, etc. The other type of

information in the multimedia datasets can be represented and reproduced as sensory experiences - touch, sense and hear which are typically leading to storing a huge amount of data handled by digitizing. Furthermore, data compression for images and sounds can exploit limits on human senses performed by throwing away information not needed for good-quality experience which is performed by compression. There are certain limitations when you deal with such a large amount of data are described below:

1. **Range is limited and might lead to misinformation**
    - only certain pitches and loudness can be heard
    - only certain kinds of light are visible, and there must be enough / not too much light
2. **Discrimination due to the descriptive features of the data**
    - pitches, loudness, colors, intensities can‘t be distinguished unless they are different enough (color1, color2)
3. **Coding the information of the sensory data into digital world.**
    - nervous systems ―encode‖ experience, e.g., rods and cones in the eye

3.3 REPRESENTATION OF IMAGE DATA

The image data is expanding vastly and due to enhancement in the cameras the encoding is needed for computation and manipulation of image data. The techniques are vector array-based encoding and bit map-based encoding.

Vector graphics encode the image sequences as a series of lines or curves. The process is expensive in term of image computation but smoothly rescales.

Bit map encode the image as an array of pixels. The encoding process is cost effective in terms of computation but scales inefficiently leading to loss of image data.

The Basic idea of image data is the array of receptors where each receptor records a pixel by ―counting‖ the number of photons that strike it during exposure. The Red, green, blue recorded separately at each point on image produced by group of three receptors where each receptor is behind a color filter.

Representation of Acoustic data: In recent years, the analysis of acoustic characteristics of speech and sound has been one of the areas that data mining has found its way through. The present research study is also related to this topic which aims to detect the gender of the speaker by using the acoustic feature of his voice. When an instrument is played or a voice speaks, periodic (many times per second) changes occur in air pressure, which we interpret as acoustics. For the representation of the acoustic data compression is needed. Codecs (compression/decompression) implement various compression/decompression techniques which are either lossy or lossless. The lossy compression of the acoustic data may lead to loss of certain information as it is non-repetitive: MPEG (like JPEG) a family of perceptually-based techniques are all lossy techniques. While the other type of techniques is where the information of the acoustic data is preserved in which WMA Lossless, ALAC, MPEG-4 ALS methods are applied.

The encoding of data in form of the sounds heard or the visuals captured are highly challenging for the data scientists to process the information. Some tasks performed by the human brain have been of great difficulty to the computers for the recognition of the object in the images. MLBAM (Major League Baseball Advanced Media) announced in 2014 that they'll increase video capture to approximately 7 TB per game for the purpose of live, in-game analytics. The higher resolution cameras and acquiring sensors have the capability to capture the motion of ball and the player in a real time scenario e.g., the path taken by a defender relative to two baselines.

Recently a company called DeepMind succeeded at creating an algorithm that's capable of learning how to play video games. This algorithm takes the video screen as input and learns to interpret everything via a complex process of deep learning. It's a remarkable feat that prompted Google to buy the company for their own Artificial Intelligence (AI) development plans. The learning algorithm takes in data as it's produced by the computer game; it's streaming data.

3.3.5 Sensor Data

Any input acquired from the physical environment which is detected and responded from the devices as an output is the collaborative approach of the sensor data. The extracted data from the sensor devices act as an output for a real time system or as an input to another system for the performance of any activity. These sensors can be used to perceive any type of physical element with the approach to detect the events or changes in the environment. A sensor is always used with other electronics, as simple as a lamp or as complex as a computer. Advanced chip technology makes it possible to integrate all the required functions at low cost, in a small volume and with low energy consumption. The number of sensors around us is increasing rapidly. Estimates vary, but many expect that by 2030 more than 500 billion sensors will be connected to each other via the Internet of Things (IoT).

The exponential growth of the IoT based systems leads to ever demanding rise in the input sensor devices which are responsible for the collection, storage and interpretation of the captured data. In addition, consumers, organizations, governments and companies themselves produce more and more data, for example on social media. The amount of data is growing exponentially. People speak of Big Data when they work with one or more datasets that are too large to be maintained with regular database management systems.

The pros of applying sensor data to the input devices is that the decisions of the IoT devices are subjected to information accessed from the evidences and not from any kind of irrelevant details and subjective experiences. This type of knowledge base makes the system cost effective with the enhanced streamlined processes, boosted product quality and better services. By combining data intelligently and by interpreting / translating, new insights are created that can be used for new services, applications and markets. This information can also be combined with data from various external sources, such as weather data or demographics.

3.3.6 Graph/ Network Data

―Graph data‖ is the type of data which can be represented as graph with a special

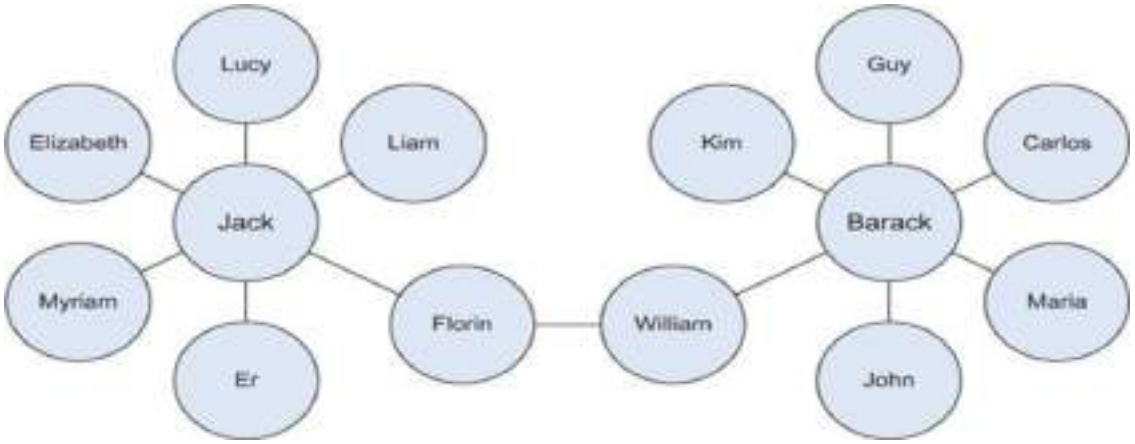


Fig. 3.1.3 Friends in social network are example of Graph Network[12]

characteristic of comprising the mathematical graph theory into the mined information. ―Graph‖ in the network data generally represents a model in the statistical and mathematical domain with pair wise relationship in the constituent objects. Graph or network data is, in short, data that focuses on the relationship or adjacency of objects. The representation of the graph models includes the nodes, edges, and relationships between the stored data in the nodes. Graph-based data is a natural way to represent social networks, and its structure allows you to calculate specific metrics such as the influence of a person and the shortest path between two people.

Examples of graph-based data can be found on many social media websites (figure 3.3). For instance, on LinkedIn you can see who you know at which company. Your follower list on Twitter is another example of graph-based data. Graph databases are used to store graph-based data and are queried with specialized query languages such as SPARQ.

3.4 MULTIPLE PROBLEMS IN HANDLING LARGE DATASETS

The multiple problems in the large datasets are discussed by numerous data scientists with the need to understand the growing demand of the data and handling the mathematical as well as statistical operations for the manipulation of the data. In the last two years, over 90% of the world‘s data was created, and with 2.5 quintillion bytes of data generated daily, it is clear that the future is filled with more data, which can also mean more data problem in context to the following:

- Collecting, storing, sharing and securing data
- Creating and utilizing meaningful insights from their data.

Some common big data problems and the respective solutions have been discussed below.

1. Lack of Understanding

The lack of understanding of the mined data in the data science-based companies might lead to knocked down the performance in many areas. Many of the major areas for the data-

based companies were: depreciate the expenses of mining information, innovate new ideas for interpretation, new product launching, enhance performance and so on. Despite the benefits, companies have been slow to adopt data for a data centric approach.

Solution: Follow a top-down approach for the introduction and manipulation of the data science based on the procedures followed up. In case of lack of a data science professional, the consultancy services or an IT proficient with data science knowledge should be hired to get a better understanding.

2. High Cost of Data Solutions

The companies have understood that buying and maintaining of necessary components make the system less cost effective. In addition to cost of the servers and the software-based storage, the high-end cost of the data science experts makes the system time consuming.

Solution: The solution is to understand the need and use of the data with a collaborative method to find a goal, conduct a research or solution and implement the execution with a plan.

3. Too Many Choices

Coined as the ―paradox of choice,‖ Schwartz explains how option overload can cause inaction on behalf of a buyer. In the world of data and data tools, the options are almost as widespread as the data itself, so it is understandably overwhelming when deciding the solution that‘s right for the business, especially when it will likely affect all departments and hopefully be a long-term strategy.

Solution: Like understanding data, a good solution is to leverage the experience of your in-house expert, perhaps a CTO. If that‘s not an option, hire a consultancy firm to assist in the decision-making process. Use the internet and forums to source valuable information and ask questions.

4. Complex Systems for Managing Data

The systems to understand the data management and finding a relevant solution for the manipulation of data is in itself a problem Due to the vast expanse of the different types of data with the IT teams creating their own data during the process of data handling results in increased complexity.

Solution: Find a solution with a single command center, implement automation whenever possible, and ensure that it can be remotely accessed 24/7.

5. Security Gaps

Another important aspect of the data science is the security of the data and the biasing of the large amount of data is always possible. In order to handle it the encryption and decryption must be performed with the data store with proper storage.

Solution: The data need to be handled with automated security updates of the data warehouse and automated backups.

6. Low Quality and Inaccurate Data

Having data is only useful when it‘s accurate. Low quality data not only serves no purpose, but it also uses unnecessary storage and can harm the ability to gather insights from clean data.

A few ways that data can be considered low quality is:

- Inconsistent formatting (which will take time to correct and can happen when the same elements are spelled differently like ―US‖ versus ―U.S.‖),
- Missing data (i.e. a first name or email address is missing from a database of contacts),
- Inaccurate data (i.e. it‘s just not the right information or the data has not be updated).
- Duplicate data (i.e. the data is being double counted)
- If data is not maintained or recorded properly, it‘s just like not having the data in the first place.

Solution: Begin by defining the necessary data you want to collect (again, align the information needed to the business goal). Cleanse data regularly and when it is collected from different sources, organize and normalize it before uploading it into any tool for analysis. Once you have your data uniform and cleansed, you can segment it for better analysis.

7. Compliance Hurdles

When collecting information, security and government regulations come into play. With the somewhat recent introduction of the General Data Protection Regulation (GDPR), it‘s even more important to understand the necessary requirements for data collection and protection, as well as the implications of failing to adhere. Companies have to be compliant and careful in how they use data to segment customers for example deciding which customer to prioritize or focus on. This means that the data must: be a representative sample of consumers, algorithms must prioritize fairness, there is an understanding of inherent bias in data, and Big Data outcomes have to be checked against traditionally applied statistical practices.

Solution: The only solution to adhere to compliance and regulation is to be informed and well-educated on the topic. There‘s no way around it other than learning because in this case, ignorance is most certainly not bliss as it carries both financial and reputational risk to your business. If you are unsure of any regulations or compliance you should consult expert legal and accounting firms specializing in those rules.

3.5 General Techniques for Handling Large Data

The multiple challenges have been discussed in the section above and the solutions for the defies are categorized based on the algorithms and out of memory errors. Never-ending algorithms, out-of-memory errors, and speed issues are the most common challenges you face when working with large data. In this section, we‘ll investigate solutions to overcome or alleviate these problems.

The solutions can be divided into three categories: using the correct algorithms, choosing the right data structure, and using the right tools. (Figure 3.4)

There is no such relationship between the problems discussed in the section above and

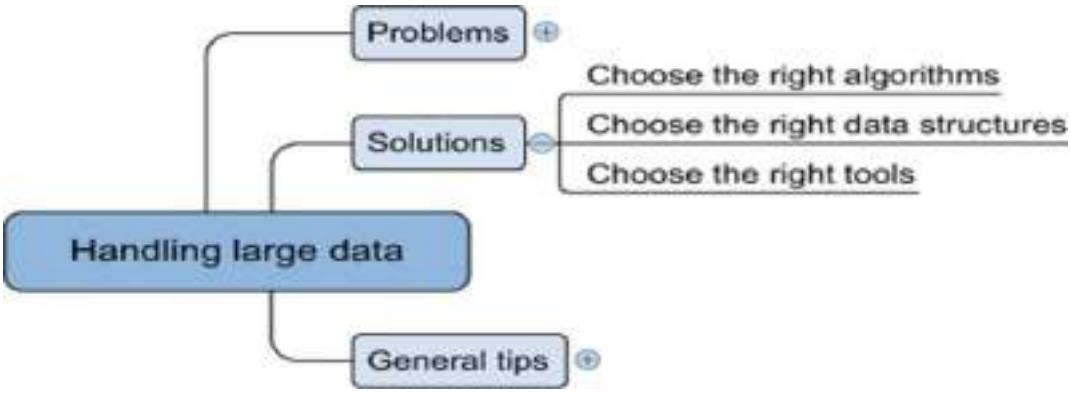


Fig. 3.1.4 Overview of solutions for handling large data sets[12]

the solutions to be incorporated. There are multiple solutions to a given problem which can also handle the issue of memory and computational overhead. This type of data science solutions can be generalized for challenges in the exploration of data. For instance, the compression and decompression of the data set help resolve the memory issues but this also affects computation speed with a shift from the slow hard disk to the fast CPU. Contrary to RAM (random access memory), the hard disc will store everything even after the power goes down, but writing to disc costs more time than changing information in the fleeting RAM. When constantly changing the information, RAM is thus preferable over the (more durable) hard disc.

3.5.1 Choosing the Right Algorithm

T

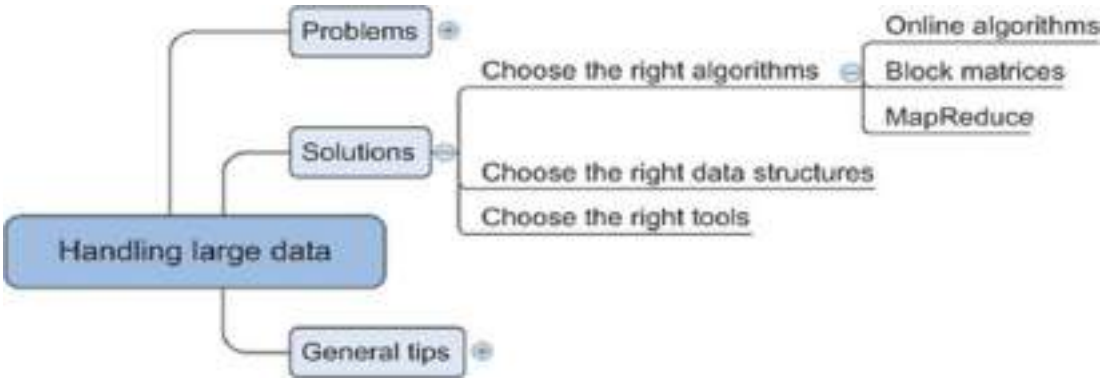


Fig. 3.1.5 The algorithms for handling the Big Data[12]

The selection of an effective algorithm for the processing of the data can provide better results as compared to the enhanced hardware. In order to perform the predictive or

selective analysis the best-chosen algorithm need not necessarily load the complete data into the memory or RAM, it rather supports the parallel computations with parallel or distributed databases. In this section three types of algorithms have been discussed that can perform the computations parallelly reducing the computation or memory overhead: online algorithms, block algorithms, and MapReduce algorithms, as shown in figure 3.5.

Several, but not all, machine learning algorithms can be trained using one observation at a time instead of taking all the data into memory. The model can be trained based on the current parameters and the previous parameter values can be made to forget by the algorithm. This technique of ―use and forget ‖ helps to attain high memory effectiveness in the system. This way as the new data values is acquired by the algorithm, the previous values are forgotten or overwritten but the effect can be witnessed or observed in the performance metrics of the proposed model. Most online algorithms can also handle mini-batches; this way, the data science exert or the developer can feed the batches of 10 to 1,000 observations at one single instance and then applying the sliding window protocol to access the data. This learning can be handled by multiple means discussed as follows:

- Full batch learning (also called statistical learning) —Feed the algorithm all the data at once.
- Mini-batch learning —Feed the algorithm a spoonful (100, 1000, ..., depending on what your hardware can handle) of observations at a time.
- Online learning —Feed the algorithm one observation at a time.

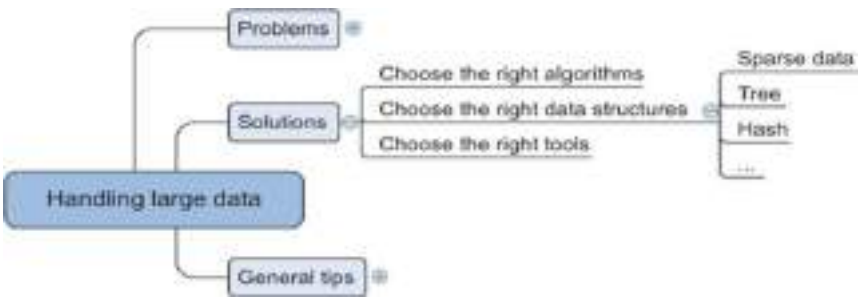


Fig. 3.1.6 Data structures applied in the data science[12]

3.5.2 Right Data Structure

The algorithms as discussed can enhance the performance and execution for the manipulation of the data in the warehouse. This process in the data science field actually leads to fragmentation in the raw data so that is the reason the structures for the storage of the data is equally important for the data scientist or a data science researcher. Data structures have different storage requirements, but also influence the performance of CRUD (create, read, update, and delete) and other operations on the data set.

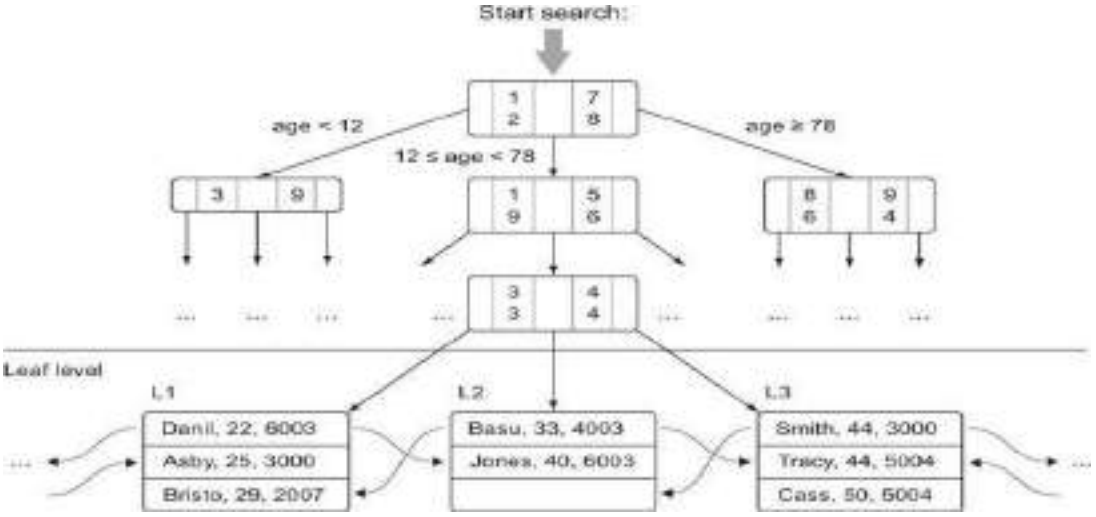
- **Sparse Data**

Most of the IT professional and the data scientist understand the representation through a sparse matrix. Similarly, the sparse data illustration can be done by giving a minimal detail about the data as compared to the total number of entries. As per figure 3.7, the conversion of the data from the text to binary or an image to binary can be expressed as the sparse data. Imagine a set of 100,000 completely unrelated Twitter tweets. Most of them probably have fewer than 30 words, but together they might have hundreds or thousands of distinct words. For this reason, the text documents are processed for the stop words, cut into fragments and stored as vectors instead of the binary information. The basic idea is that any word present in the tweet is expressed as 1 and not in tweet is expressed as 0 resulting in sparse data indeed. But the matrix generated would require equal memory as compared to any other matrix even though it has a little information.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

- **Tree Structure**

Trees is a special type of data structure that has faster retrieval of information in comparison to the table or sparse data. In these data structures the root node is the first directory for accessing the information and the child nodes are the sub directories of the root node. The information can be accessed form the child or leaf nodes by either using pointers or indexing of the tree structure. The figure 3.8 helps the researcher to understand the process of information retrieval in the tree structure.



- **Hash Tables**

The process of the calculation of the key for each data entry and allotting the key to the relevant bucket is the important process of the hash table structure for th storage of data. The process of storage and handling in the hash table makes it more reliable source for the retrieval of information based on the key value. This way you can quickly retrieve the information by looking in the right bucket when you encounter the data. Dictionaries in Python are a hash table implementation, and they‘re a close relative of key-value stores. You‘ll encounter them in the last example of this chapter when you build a recommender system within a database. Hash tables are used extensively in databases as indices for fast information retrieval.

3.5.2 Choosing the Right Tools

As discussed earlier in section 3.4.1 and 3.4.2 with the need to have a right algorithm

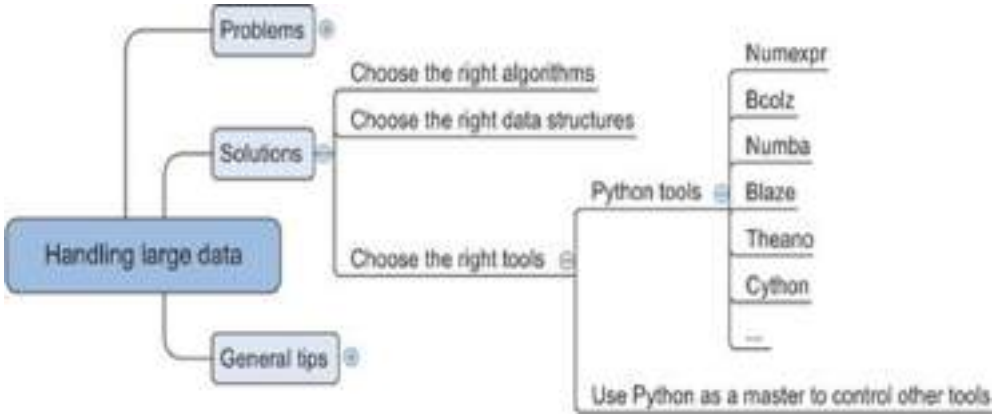


Fig. 3.1.7 Tools applied by data scientist for handing data[12]

and the right data structure, the requirement of a good tool is also important. The right tool can be a Python library or at least a tool that‘s controlled from Python, as shown figure 3.9.

Python has a number of libraries that can help you deal with large data. They range from smarter data structures over code optimizers to just-in-time compilers. The following is a list of libraries we like to use when confronted with large data:

Cython —The closer to the actual hardware of a computer, the more vital it is for the computer to know what types of data it has to process. For a computer, adding 1 + 1 is different from adding 1.00 + 1.00. The first example consists of integers and the second consists of floats, and these calculations are performed by different parts of the CPU. Python needs no specifications of the types of the data but the compiler can itself interpret the values based on the integer or float. This is although a slow process and so a superset of python can be used for the solution which forces the user to define the data type before the execution and hence can be implemented much faster. See http://cython.org/ for more information on Cython.

Numexpr —Numexpr is at the core of many of the big data packages, as is NumPy for in-memory packages. Numexpr is a numerical expression evaluator for NumPy but can be many times faster than the original NumPy. https://github.com/pydata/numexpr.

Numba —Numba is the package which compiles (just-in-time compiling) the code for the data manipulation and handling before actually executing it which makes it faster and less prone to errors. With the user gets a platform to develop a high level code wit the speed of the compilation as in C. http://numba.pydata.org/.

Bcolz —Bcolz helps you overcome the out-of-memory problem that can occur when using NumPy. It can store and work with arrays in an optimal compressed form. It not only slims down your data need but also uses Numexpr in the background to reduce the calculations needed when performing calculations with bcolz arrays. http://bcolz.blosc.org/.

Blaze — Blaze is ideal in case the data scientist use the power of a database backend but like the ―Pythonic way‖ of working with data. Blaze will translate your Python code into SQL but can handle many more data stores than relational databases such as CSV, Spark, and others. Blaze delivers a unified way of working with many databases and data libraries. http://blaze.readthedocs.org/en/latest/index.html.

Theano —Theano enables you to work directly with the graphical processing unit (GPU) and do symbolical simplifications whenever possible, and it comes with an excellent just-in-time compiler. On top of that it‘s a great library for dealing with an advanced but useful mathematical concept: tensors. http://deeplearning.net/software/theano.

3.6 DISTRIBUTING DATA STORAGE AND PROCESSING WITH FRAMEWORKS

New big data technologies such as Hadoop and Spark make it much easier to work with and control a cluster of computers. Hadoop can scale up to thousands of computers, creating a cluster with petabytes of storage. This enables businesses to grasp the value of the massive amount of data available. ―Big data Analytics‖ is a phrase that was coined to refer to amounts of datasets that are so large, traditional data processing software simply can‘t manage them. For example, big data is used to pick out trends in economics, and those trends and patterns are used to predict what will happen in the future. These vast amounts of data require more robust computer software for processing, best handled by data processing frameworks. These are the top preferred data processing frameworks, suitable for meeting a variety of different needs of businesses.

3.6.1 Hadoop

This is an open-source batch processing framework that can be used for the distributed storage and processing of big data sets. Hadoop relies on computer clusters and modules that have been designed with the assumption that hardware will inevitably fail, and those failures should be automatically handled by the framework.

There are four main modules within Hadoop. Hadoop Common is where the libraries and utilities needed by other Hadoop modules reside. The Hadoop Distributed File System (HDFS) is the distributed file system that stores the data. Hadoop YARN (Yet Another Resource Negotiator) is the resource management platform that manages the

computing resources in clusters, and handles the scheduling of users‘ applications. The Hadoop MapReduce involves the implementation of the MapReduce programming model for large-scale data processing.

Hadoop operates by splitting files into large blocks of data and then distributing those datasets across the nodes in a cluster. It then transfers code into the nodes, for processing data in parallel. The idea of data locality, meaning that tasks are performed on the node that stores the data, allows the datasets to be processed more efficiently and more quickly. Hadoop can be used within a traditional onsite datacenter, as well as through the cloud.

3.6.2 Apache Spark

Apache Spark is a batch processing framework that has the capability of stream processing, as well, making it a hybrid framework. Spark is most notably easy to use, and it‘s easy to write applications in Java, Scala, Python, and R. This open-source cluster-computing framework is ideal for machine-learning, but does require a cluster manager and a distributed storage system. Spark can be run on a single machine, with one executor for every CPU core. It can be used as a standalone framework, and you can also use it in conjunction with Hadoop or Apache Mesos, making it suitable for just about any business.

Spark relies on a data structure known as the Resilient Distributed Dataset (RDD). This is a read-only multiset of data items that is distributed over the entire cluster of machines. RDDs operate as the working set for distributed programs, offering a restricted form of distributed shared memory. Spark is capable of accessing data sources like HDFS, Cassandra, HBase, and S3, for distributed storage. It also supports a pseudo-distributed local mode that can be used for development or testing.

The foundation of Spark is Spark Core, which relies on the RDD-oriented functional style of programming to dispatch tasks, schedule, and handle basic I/O functionalities. Two restricted forms of shared variables are used: broadcast variables, which reference read-only data that has to be available for all the nodes, and accumulators, which can be used to program reductions. Other elements included in Spark Core are:

Spark SQL, which provides domain-specific language used to manipulate Data Frames.

Spark Streaming, which uses data in mini-batches for RDD transformations, allowing the same set of application code that is created for batch analytics to also be used for streaming analytics. Spark MLlib, a machine-learning library that makes the large-scale machine learning pipelines simpler. GraphX, which is the distributed graph processing framework at the top of Apache Spark.

3.6.3 Apache Storm

This is another open-source framework, but one that provides distributed, real-time stream processing. Storm is mostly written in Clojure, and can be used with any programming language. The application is designed as a topology, with the shape of a Directed Acyclic Graph (DAG). Spouts and bolts act as the vertices of the graph. The idea behind Storm is to define small, discrete operations, and then compose those operations into a topology, which acts as a pipeline to transform data.

3.6.4 Samza

Samza is another open-source framework that offers near a real-time, asynchronous framework for distributed stream processing. More specifically, Samza handles immutable streams, meaning transformations create new streams that will be consumed by other components without any effect on the initial stream. This framework works in conjunction with other frameworks, using Apache Kafka for messaging and Hadoop YARN for fault tolerance, security, and management of resources.

3.6.5 Flink

Flink is a hybrid framework, open-source, and stream processes, but can also manage batch tasks. It uses a high-throughput, low-latency streaming engine that is written in Java and Scala, and the runtime system that is pipelined allows for the execution of both batch and stream processing programs. The runtime also supports the execution of iterative algorithms natively. Flink‘s applications are all fault-tolerant and can support exactly-once semantics. Programs can be written in Java, Scala, Python, and SQL, and Flink offers support for event-time processing and state management.

3.7 SUMMARY

Data science involves using methods to analyze massive amounts of data and extract the knowledge it contains. You can think of the relationship between big data and data science as being like the relationship between crude oil and an oil refinery. Data science and big data evolved from statistics and traditional data management but are now considered to be distinct disciplines.

The foremost aspect for the data scientist to conduct the refining of the raw data is the representation of data dealing with the problems of big data. Another aspect of the dealing with the problems of big data is to perform the processing of the big data with the frameworks. For this multiple special type of data have been explored.

There are multiple problems being faced by the data scientists for the processing of the raw data and mining it into useful and relevant information. Luckily, there are pragmatic solutions that companies can take to overcome their data problems and thrive in the data-driven economy. The problems and the relevant solutions have been discussed in the chapter. Data processing frameworks are not intended to be one-size-fits-all solutions for businesses. Hadoop was originally designed for massive scalability, while Spark is better with machine learning and stream processing. A good IT services consultant can evaluate your needs and offer advice. What works for one business may not work for another, and to get the best possible results, you may find that it‘s a good idea to use different frameworks for different parts of your data processing.

3.8 PRACTICE QUESTIONS

Q1. Discuss the various representation techniques of different data types.
Q2.What are the issues that are faced in handling large data?
Q3. Explain briefly the techniques to handle large data.
Q4.What are the popular frameworks for big data storage?

UNIT IV: DATA ACQUISITION AND PROCESSING

STRUCTURE

4.1 OBJECTIVE

1. To familiarize with the Data Science Ethics
2. To familiarize with different aspects of privacy
3. To familiarize with the future trends of data Science

4.2 DATA SCIENCE ETHICS

The skill to extract or mine the relevant patterns and the transforming capability to revolutionize the products in data science helps to bring a positive change in the social and technological sphere making it ethically neutral. It does not come with its own perspective of either: what is correct or incorrect; nor: what is good or bad in using it. There is no such kind of a value-based framework while the companies working on the data store have a value-based system for the handling of the information. Anything which is private or protected is not for anyone to access except the administrator or the data scientist himself/herself. The problems of the ethical mishandling of data and the relevant solutions to the problems must be able to amalgamate with the ethics of the companies working in Bigdata.

The future of the technology world is in the hands of machine learning techniques with AI based systems where data science is the solution for the data on which these systems are trained. The data science is kind of fuel for he working intelligent systems as there training can be done on millions of raw data and all of it can be mined or extracted from the data science sources through the extractive approach of the data scientists. Data Ethics is a rapidly improvising field-of-study. The IT professionals and the data scientist work for the collection, sharing and the manipulation of data which is done by keeping ethics in the exploration of data and are sometimes even forced to deal with the ethics to avoid any kind of negative public opinion. Loss of ethics sometimes can prove to be alienating to the reputation and work culture of any kind of organization.

4.3 DATA SCIENCE – GOOD ASPECTS FOR TECHNOLOGY

Data science involves a plethora of disciplines and expertise areas to produce a holistic, thorough and refined look into raw data. The professionals might be trained for the handling, manipulation and the interpretation of data but the most important and vital part of the data science is to perform the scrutiny of the relevant information from the disarrayed or unorganized form of raw data and only interconnect or forward only those data values which are of vital importance for the invention or the improvisation of the existing system. These reasons are sufficient enough for the data scientists to completely depend upon the existing technological advancement in the field of machine learning or deep learning with the ability to create the simulated models and predict the values in the models with the proposed algorithms and techniques.

To acclimatize the organization with the ethical data science, there‘s a need to understand what actually are the ethics about data science with the cost requirement for the implementation of ethical values and what can be the ways to execute such practices.

There are several ways and means with the respective solutions for the same.

Firstly, the data scientist needs to understand about the daintiness of the information in the data store. With this the relevant implementation or the operations can be performed on the data keeping in the view the consequences with improper or unacceptable access of the data for information. Indeed, the security perspective in computer or network has proved many times that ignoring the consequences of accessing the data unethically might cause a trouble to the data professionals or the company as a whole in terms of loss of reputation or money.

With the available systems it is difficult for any data scientist to predict the inevitable unethical uses of the data but with the AI based techniques especially using machine learning and deep learning, a prediction can be made for the unintended consequences. For example, Facebook‘s ―Year in Review‖ that reminded people of deaths and other painful events. This can be handled by the data professionals in the field of data science by keeping in view the patterns of the data to be explored and how to think of better means to represent the data with a new or enhanced approach.

Another important step to stop the data mining if the administrator finds any kind of problem in the production line. This idea goes back to Toyota‘s Kanban: any assembly line worker can stop the line if they see something going wrong. The line doesn‘t restart until the problem is fixed. Workers don‘t have to fear consequences from management for stopping the line; they are trusted, and expected to behave responsibly

The issue lurking behind all of these concerns is, of course, corporate culture. Corporate environments can be hostile to anything other than short-term profitability. That‘s a consequence of poor court decisions and economic doctrine, particularly in the U.S. But that inevitably leads us to the biggest issue: how to move the needle on corporate culture. Susan Etlinger has suggested that, in a time when public distrust and disenchantment is running high, ethics is a good investment. Upper-level management is only starting to see this; changes to corporate culture won‘t happen quickly. Users want to engage with companies and organizations they can trust not to take unfair advantage of them. Users want to deal with companies that will treat them and their data responsibly, not just as potential profit or engagement to be maximized. Those companies will be the ones that create space for ethics within their organizations. We, the data scientists, data engineers, AI and ML developers, and other data professionals, have to demand change. We can‘t leave it to people that ―do‖ ethics. We can‘t expect management to hire trained ethicists and assign them to our teams. We need to live ethical values, not just talk about them. We need to think carefully about the consequences of our work. We must create space for ethics within our organizations. Cultural change may take time, but it will happen—if we are that change. That‘s what it means to do good data science

4.4 OWNERS OF DATA

Data owners are either individuals or teams who make decisions such as who has the right to access and edit data and how it's used. Owners may not work with their data every day, but are responsible for overseeing and protecting a data domain.

A data owner is responsible for the data in a particular data domain. They may belong to the steering committee and ensure that the data under their view is governed throughout the organization. Data owners approve data glossaries and definitions as well as initiate data quality activities.

Owning' data – who owns data, what's capable of being owned, and what rights and responsibilities ownership attracts – is gaining a lot of attention.

4.4.1 Responsibilities of the Data Owner

The first responsibility of the data owner is to classify the data correctly. Once a classification has been set it is up to the data owner to determine who has access to the data. Usually, this access is based upon roles as opposed to individuals.

The data classification is one of the most important steps. Data classification has a different meaning for different organizations but at the basic level it is knowing the type of data a company has, determining its value, and categorizing it. For example, if your company has a secret sauce or original intellectual property it may be considered ―top secret‖ or ―confidential‖. The reason to label or classify it as ―top secret‖ or ―confidential‖ is so it can be handled and ultimately protected appropriately. In addition to not putting the correct controls on data there is the potential to retain data for longer than needed or destroy data before it should be based on laws or contractual commitments. Many people have an opinion on how data should be classified or labeled but at the end of the day it is the responsibility of the data owner who is ultimately accountable for the data to make the final decision. The data owner will have the most knowledge of the use of the data and the value to the company. It is advisable for the data owner to get input from various sources like the data custodian or data users but the data owner has complete control over the data.

- Who has access to the data? Clarify the roles of people who can access the data. Example: Employees can see an organization chart with departments, manager names, and titles but not salary information (Classification = internal). But a very limited audience like HR should only have access to salary data, performance data, or social security numbers (Classification = confidential).
- How is the data secured? Sensitive data elements within HR documentation have been classified to be confidential and therefore it requires additional security controls to protect it. Some of the additional controls to secure confidential data stored in electronic medium could include being saved in a location on the network with appropriate safeguards to prevent unauthorized access (secure folders protected by passwords).
- How long the data is retained? Many industries require that data be retained for a certain length of time. For example, the finance industry requires a seven-year retention period and some health care industries requires a 100-year retention period. Data owners need to know the regulatory requirements for their data and if there is no clear guidance on retention then it should be based off the company‘s retention policy or best practices.

- How data should be destroyed? Based on the classification of the data there should be clear guidance on how to dispose or destroy the data. For example:

Information Medium	Public	Internal	Confidential
Hard Copy	Place in recycling bins or trash receptacles.	Place in secured shedding bins or manually shred.	Place in secured shedding bins or manually shred with a cross-cut shredder or pulped. A record must be maintained that indicates the records disposed of and the data of disposal.
Electronic records	Electronic records can be deleted normally.	Electronic records need to be overwritten to 1‘s and 0‘s or with a secure delete option.	Electronic records need to be degaussed off magnetic media after securely deleted or the physical media should be destroyed with a record maintained that indicates the records disposed of and the date of disposal.

- What data needs to be encrypted? Data owners should decide whether their data needs to be encrypted. To make this determination the data owner should know the applicable laws or regulation requirements set that must be complied with. A good example of a regulation requirement is set by the Payment Card Industry (PCI) Data Security Standard and it requires that the transmission of cardholder data across open, public networks must be encrypted.

4.4.2 The Importance of Assigning Data Owners

In most organizations, as data passes through different teams and systems, assigning data owners can be cumbersome. However, this is a critical step for GDPR compliance.

Here‘s why assigning data owners is important:

1. Accountability — Ownership creates accountability. Since GDPR introduces many controls on personal data, assigning responsibilities ensures that data will be continuously monitored for compliance by the owners.

2. Defining policies — As they have a vested interest in the integrity of their data, owners focus on defining policies (for example retention or deletion policies) and standards that ensure the alignment of their data to the GDPR.

3. Creating trusted data — Data ownership is a key ingredient to gain customer trust and achieve measurable business benefits. Poor data could easily result in bad customer

experiences and ultimately losing customers. In particular, when personal data is not reconciled into a data subject 360° view, compliance with data subject access rights, such as rights of portability or rights to be forgotten cannot be fully achieved.

4. Eliminating redundancies — As organizations strive to put the appropriate governance framework in place for GDPR, one common frustration is a loss of productivity. This issue stems from multiple teams addressing the same problem either because there isn‘t a clear understanding of data or they‘re not even aware that the problem has been resolved by another team. Federated ownership eliminates these painful issues.

4.4.3 Identification of Data Owners: Three Questions to Ask

The mandatory introduction of a data protection officer (DPO) role by GDPR in most organizations effectively creates a master data owner. Each and every element in a data taxonomy needs an individual owner, however, and there is little likelihood that a single DPO can hold this responsibility on such a large scale. In addition, this could create a security issue. Delegation and segregation of duties are needed.

Asking the right questions helps. Once these questions have been answered, the data owner should become clearer:

1. Who is most impacted by data accuracy?
2. Who has authority to decide the next step?
3. Who owns the related data attributes?

4.5 DIFFERENT ASPECTS OF PRIVACY

First there is the data where security and privacy has always mattered and for which there is already an existing and well galvanized body of law in place. Foremost among these is classified or national security data where data usage is highly regulated and enforced. Other data for which there exists a considerable body of international and national law regulating usage includes:

Proprietary Data – specifically the data that makes up the intellectual capital of individual businesses and gives them their competitive economic advantage over others, including data protected under copyright, patent, or trade secret laws and the sensitive, protected data that companies collect on behalf of its customers;

Infrastructure Data - data from the physical facilities and systems – such as roads, electrical systems, communications services, etc. – that enable local, regional, national, and international economic activity; and

Controlled Technical Data - technical, biological, chemical, and military-related data and research that could be considered of national interest and be under foreign export restrictions.

It may be possible to work with publicly released annualized and cleansed data within these areas without a problem, but the majority of granular data from which significant insight can be gleaned is protected. In most instances, scientists, researchers, and other authorized developers take years to appropriately acquire the expertise, build the personal relationships,

and construct the technical, procedural and legal infrastructure to work with the granular data before implementing any approach. Even using publicly released datasets within these areas can be restricted, requiring either registration, the recognition of or affiliation to an appropriate data governing body, background checks, or all three before authorization is granted.

The second group of data that raises privacy and security concerns is personal data. Commonly referred to as Personally Identifiable Information (PII), it is any data that distinguishes individuals from each other. It is also the data that an increasing number of digital approaches rely on, and the data whose use tends to raise the most public ire. Personal data could include but is not limited to an individual‘s:

- Government issued record data (social security numbers, national or state identity numbers, passport records, vehicle data, voting records, etc.);
- Law enforcement data (criminal records, legal proceedings, etc.);
- Personal financial, employment, medical, and education data;
- Communication records (phone numbers, texts data, message records, content of conversations, time and location, etc.);
- Travel data (when and where traveling, carriers used, etc.);
- Networks and memberships (family, friends, interests, group affiliations, etc.);
- Location data (where a person is and when);
- Basic contact information (name, address, e-mail, telephone, fax, twitter handles, etc.);
- Internet data (search histories, website visits, click rates, likes, site forwards, comments, etc.);
- Media data (which shows you‘re watching, music you‘re listening to, books or magazines you‘re reading, etc.);
- Transaction data (what you‘re buying or selling, who you‘re doing business with, where, etc.); and
- Bio and activity data (from personal mobile and wearable devices).

In industries where being responsible for handling highly detailed personal data is the established business norm – such as in the education, medical and financial fields – there are already government regulations, business practices and data privacy and security laws that protect data from unauthorized usage, including across new digital platforms. But in many other industries, particularly in data driven industries where personal data has been treated as proprietary data and become the foundation of business models, there is currently little to no regulation. In the new normal, the more that a data approach depends on data actively or passively collected on individuals, the more likely that consumers will speak up and demand privacy protection, even if they previously gave some form of tacit approval to use their data.

` Despite this new landscape, there are lots of different ways to use personal data, some of which may not trigger significant privacy or security concerns. This is particularly true in cases where individuals willingly provide their data or data cannot be attributed to an

individual. Whether individuals remain neutral to data approaches tends to be related to the level of control they feel they have over how their personal data is used. Some organizations that collect personal data extensively, such as Facebook and Google, work to increasingly provide their users with methods to control their own data. But for others, the lack of due diligence on data privacy in their approaches has already had their effect.

A third category of data needing privacy consideration is the data related to good people working in difficult or dangerous places. Activists, journalists, politicians, whistle-blowers, business owners, and others working in contentious areas and conflict zones need secure means to communicate and share data without fear of retribution and personal harm.

4.6 FIVE C' S OF DATA SCIENCE

What does it take to build a good data product or service? Not just a product or service that's useful, or one that's commercially viable, but one that uses data ethically and responsibly.

Users lose trust because they feel abused by malicious ads; they feel abused by fake and misleading content, and they feel abused by ―act first, and apologize profusely later‖ cultures at many of the major online companies. And users ought to feel abused by many abuses they don't even know about. Why was their insurance claim denied? Why weren't they approved for that loan? Were those decisions made by a system that was trained on biased data? The slogan goes, ―Move fast and break things.‖ But what if what gets broken is society?

Data collection is a big business. Data is valuable: ―the new oil,‖ as the Economist proclaimed. We've known that for some time. But the public provides the data under the assumption that we, the public, benefit from it. We also assume that data is collected and stored responsibly, and those who supply the data won't be harmed. Essentially, it's a model of trust. But how do you restore trust once it's been broken? It's no use pretending that you're trustworthy when your actions have proven that you aren't. The only way to get trust back is to be trustworthy, and regaining that trust once you've lost it takes time.

There's no simple way to regain users' trust, but a ―golden rule‖ for data as a starting point: ―treat others' data as data scientist would like other to treat their data.‖ However, implementing a ―golden rule‖ in the actual research and development process is challenging. The golden rule isn't enough by itself. There has to be certain guidelines to force discussions with the application development teams, application users, and those who might be harmed by the collection and use of data. Five framing guidelines help us think about building data products. We call them the five Cs: consent, clarity, consistency, control (and transparency), and consequences (and harm).

4.6.1 Consent

The trust between the people who are providing data and the people who are using it cannot be established without agreement about what data is being collected and how that data will be used . Agreement starts with obtaining consent to collect and use data. Unfortunately, the agreements between a service's users (people whose data is collected) and the service itself (which uses the data in many ways) are binary (meaning that you either accept or decline)

and lack clarity. In business, when contracts are being negotiated between two parties, there are multiple iterations (redlines) before the contract is settled. But when a user is agreeing to a contract with a data service, you either accept the terms or you don‘t get access. It‘s non-negotiable.

Data is frequently collected, used, and sold without consent. This includes organizations like Acxiom, Equifax, Experian, and Transunion, who collect data to assess financial risk, but many common brands also connect data without consent. In Europe, Google collected data from cameras mounted on cars to develop new mapping products. AT&T and Comcast both used cable set top boxes to collect data about their users, and Samsung collected voice recordings from TVs that respond to voice commands.

4.6.2 Clarity

Clarity is closely related to consent. Users must have clarity about what data they are providing, what is going to be done with the data, and any downstream consequences of how their data is used. All too often, explanations of what data is collected or being sold are buried in lengthy legal documents that are rarely read carefully, if at all. Observant readers of Eventbrite‘s user agreement recently discovered that listing an event gave the company the right to send a video team, and exclusive copyright to the recordings. And the only way to opt out was by writing to the company. The backlash was swift once people realized the potential impact, and Eventbrite removed the language.

Facebook users who played Cambridge Analytica‘s ―This Is Your Digital Life‖ game may have understood that they were giving up their data; after all, they were answering questions, and those answers certainly went somewhere. But did they understand how that data might be used? Or that they were giving access to their friends‘ data behind the scenes? That‘s buried deep in Facebook‘s privacy settings. It really doesn‘t matter which service you use; you rarely get a simple explanation of what the service is doing with your data, and what consequences their actions might have. Unfortunately, the process of consent is often used to obfuscate the details and implications of what users may be agreeing to. And once data has escaped, there is no recourse.

4.6.3 Consistency and Trust

Trust requires consistency over time. You can‘t trust someone who is unpredictable. They may have the best intentions, but they may not honor those intentions when you need them to. Or they may interpret their intentions in a strange and unpredictable way. And once broken, rebuilding trust may take a long time. Restoring trust requires a prolonged period of consistent behavior.

Consistency, and therefore trust, can be broken either explicitly or implicitly. An organization that exposes user data can do so intentionally or unintentionally. In the past years, we‘ve seen many security incidents in which customer data was stolen: Yahoo!, Target, Anthem, local hospitals, government data, and data brokers like Experian, the list grows longer each day. Failing to safeguard customer data breaks trust—and safeguarding data means nothing if not consistency over time.

4.6.4 Control and Transparency

All too often, users have no effective control over how their data is used. They are given all-or-nothing choices, or a convoluted set of options that make controlling access overwhelming and confusing. It‘s often impossible to reduce the amount of data collected, or to have data deleted later. A major part of the shift in data privacy rights is moving to give users greater control of their data. For example, Europe‘s General Data Protection Regulation (GDPR) requires a user‘s data to be provided to them at their request and removed from the system if they so desire.

4.6.5 Consequences

Data products are designed to add value for a particular user or system. As these products increase in sophistication, and have broader societal implications, it is essential to ask whether the data that is being collected could cause harm to an individual or a group. The unforeseen consequences and the ―unknown unknowns‖ about using data and combining data sets have been witnessed frequently. Risks can never be eliminated completely. However, many unforeseen consequences and unknown unknowns could be foreseen and known, if only people had tried. All too often, unknown unknowns are unknown because we don‘t want to know.

While Strava and AOL triggered a chain of unforeseen consequences by releasing their data, it‘s important to understand that their data had the potential to be dangerous even if it wasn‘t released publicly. Collecting data that may seem innocuous and combining it with other data sets has real-world implications. It‘s easy to argue that Strava shouldn‘t have produced this product, or that AOL shouldn‘t have released their search data, but that ignores the data‘s potential for good. In both cases, well-intentioned data scientists were looking to help others. The problem is that they didn‘t think through the consequences and the potential risks.

Many data sets that could provide tremendous benefits remain locked up on servers. Medical data that is fragmented across multiple institutions limits the pace of research. And the data held on traffic from ride-sharing and gps/mapping companies could transform approaches for traffic safety and congestion. But opening up that data to researchers requires careful planning.

4.7 DIVERSITY – INCLUSION

Reports of AI gone wrong abound, and Responsible AI has started to take a foothold in business — Gartner has even added Responsible AI as a new category on its Hype Cycle for Emerging Technologies. Yet when talking about solutions, increasing diversity and making data science a more inclusive field unfortunately don‘t often top the list. Noelle Silver, Head of Instruction, Data Science, Analytics, and Full Stack Web Development at HackerU, is looking to change that.

A 2018 study revealed that only 15% of data scientists are women, and sadly, a 2020 study found exactly the same results: it seems we haven‘t managed to move the needle. While

diversity obviously encompasses more than just women, few studies have been able to quantify other types of representation in the field. Inclusivity is similarly difficult to quantify, both in terms of people working on technology and the ways in which technology can be accessed by all. But that doesn‘t mean there aren‘t solutions.

Problem

―The reality is that when we train machine learning models with a bunch of data, it‘s going to make predictions based on that data. If that data comes from a room of people that look the same, talk the same, act the same, they‘re all friends — it‘s not a bad scenario. In the moment, you feel like things are good. No one is really seeing any problems; you don‘t feel any friction. It‘s very misleading, especially in artificial intelligence. So, you go to market.

The problem, though, is not everyone looks like you, or talks like you, or thinks like you. So even though you found a community of people that built this software that thinks the same, as soon as you go to market and someone other than that starts using it, they start to feel that friction.‖

Solution

Of course, there‘s no easy, magic bullet solution to this problem, but foundations of a good start are:

- Committing the time and resources to practice inclusive engineering: This includes, but certainly isn‘t limited to, doing whatever it takes to collect and use diverse datasets.
- Create an experience that welcomes more people to the field: This might mean looking at everything from education to hiring practices.
- Think beyond regulations: Simply being compliant doesn‘t necessarily mean experiences are optimized.

4.8 FUTURE TRENDS

Trend 1: Smarter, faster, more responsible AI

Within the current pandemic context, AI techniques such as machine learning (ML), optimization and natural language processing (NLP) are providing vital insights and predictions about the spread of the virus and the effectiveness and impact of countermeasures. AI and machine learning are critical realigning supply and the supply chain to new demand patterns.

Trend 2: Decline of the dashboard

Dynamic data stories with more automated and consumerized experiences will replace visual, point-and-click authoring and exploration. As a result, the amount of time users spends using predefined dashboards will decline. The shift to in-context data stories means that the most relevant insights will stream to each user based on their context, role or use. These dynamic

insights leverage technologies such as augmented analytics, NLP, streaming anomaly detection and collaboration.

Trend 3: Decision intelligence

Decision intelligence brings together a number of disciplines, including decision management and decision support. It encompasses applications in the field of complex adaptive systems that bring together multiple traditional and advanced disciplines. It provides a framework to help data and analytics leaders design, compose, model, align, execute, monitor and tune decision models and processes in the context of business outcomes and behavior. Explore using decision management and modeling technology when decisions need multiple logical and mathematical techniques, must be automated or semi-automated, or must be documented and audited.

Trend 4: X analytics

Gartner coined the term ―X analytics‖ to be an umbrella term, where X is the data variable for a range of different structured and unstructured content such as text analytics, video analytics, audio analytics, etc.

Data and analytics leaders use X analytics to solve society‘s toughest challenges, including climate change, disease prevention and wildlife protection with analytics capabilities available from their existing vendors, such as cloud vendors for image, video and voice analytics, but recognize that innovation will likely come from small disruptive startups and cloud providers.

Trend 5: Augmented data management

Augmented data management uses ML and AI techniques to optimize and improve operations. It also converts metadata from being used in auditing, lineage and reporting to powering dynamic systems. Augmented data management products can examine large samples of operational data, including actual queries, performance data and schemas. Using the existing usage and workload data, an augmented engine can tune operations and optimize configuration, security and performance. Data and analytics leaders should look for augmented data management enabling active metadata to simplify and consolidate their architectures, and also increase automation in their redundant data management tasks.

Trend 6: Cloud is a given

As data and analytics moves to the cloud, data and analytics leaders still struggle to align the right services to the right use cases, which leads to unnecessary increased governance and integration overhead. Data and analytics leaders need to prioritize workloads that can exploit cloud capabilities and focus on cost optimization and other benefits such as change and innovation acceleration when moving to cloud.

Trend 7: Data and analytics worlds collide

The collision of data and analytics will increase interaction and collaboration between historically separate data and analytics roles. This impacts not only the technologies and

capabilities provided, but also the people and processes that support and use them. The spectrum of roles will extend from traditional data and analytics roles in IT to information explorer, consumer and citizen developer as an example. To turn the collision into a constructive convergence, incorporate both data and analytics tools and capabilities into the analytics stack.

Trend 8: Data marketplaces and exchanges

Data marketplaces and exchanges provide single platforms to consolidate third-party data offerings. These marketplaces and exchanges provide centralized availability and access (to X analytics and other unique data sets, for example) that create economies of scale to reduce costs for third-party data. To monetize data assets through data marketplaces, data and analytics leaders should establish a fair and transparent methodology by defining a data governance principle that ecosystems partners can rely on.

Trend 9: Blockchain in data and analytics

Blockchain technologies address two challenges in data and analytics. First, blockchain provides the full lineage of assets and transactions. Second, blockchain provides transparency for complex networks of participants. Data and analytics should position blockchain technologies as supplementary to their existing data management infrastructure by highlighting the capabilities mismatch between data management infrastructure and blockchain technologies.

Trend 10: Relationships form the foundation of data and analytics value

Graph analytics is a set of analytic techniques that allows for the exploration of relationships between entities of interest such as organizations, people and transactions. It helps data and analytics leaders find unknown relationships in data and review data not easily analyzed with traditional analytics.

4.9 SUMMARY

When combined with ML algorithms, these technologies can be used to comb through thousands of data sources and documents that could help medical and public health experts rapidly discover new possible treatments or factors that contribute to more negative outcomes for some patients.

Data and analytics leaders need to evaluate opportunities to incorporate graph analytics into their analytics portfolios and applications to uncover hidden patterns and relationships. In addition, consider investigating how graph algorithms and technologies can improve your AI and ML initiatives.

4.10 PRACTICE QUESTIONS

Q1. Write a short note on Data Science Ethics.

Q2. What are the privacy aspects of data?

Q3. What are the five C‘s of data Science

Q4. Discuss briefly the future trends in Data Science.

REFERENCES

1. https://www.knowledgehut.com/blog/big-data/5-best-data-processing-frameworks
2. Mark J. Costello, Peter T. Harris, Bryony Pearce, Andrea Fiorentino, Jean-François Bourillet, Sarah M. Hamylton, ―A Glossary of Terminology Used in Marine Biology, Ecology, and Geology‖, Michael I. Goldstein, Dominick A. DellaSala, Encyclopedia of the World's Biomes, Elsevier, 2020, Pages 471-478, ISBN 9780128160978, https://doi.org/10.1016/B978-0-12-409548-9.11944-X.
3. Big Data Analytics, Advances in Intelligent Systems and Computing. Springer Singapore (Print ISBN: 978-981-10-6619-1 Electronic ISBN: 978-981-10-6620-7)
4. https://www.solvexia.com/blog/15-big-data-problems-you-need-to-solve
5. https://blog.quantela.com/the-ethical-challenges-of-a-data-science-practitioner/
6. https://towardsdatascience.com/5-key-ai-problems-related-to-data-privacy-f39558290530
7. https://www.oreilly.com/radar/the-five-cs
8. https://builtin.com/data-science
9. https://www.edx.org/course/subject/data-science
10. https://www.geeksforgeeks.org/types-of-sources-of-data-in-data-mining/
11. https://internetofthingsagenda.techtarget.com/definition/sensor-data
12. https://3bplus.nl/how-do-smart-devices-work-sensors-iot-big-data-and-ai/
13. Introducing Data science, big data, machine learning, and more, using python tools, Cielen D., Meysman A., Ali M., Manning Publications Co ., 2016, ISBN: 9781633430037
14. https://www.andrew.cmu.edu/user/nbier/15110/lectures/lec15a_sound_video.pdf
15. https://simplicable.com/new/data-artifact
16. https://www.geeksforgeeks.org/different-sources-of-data-for-data-analysis/
17. https://nptel.ac.in/courses/106/105/106105174/
18.

INTRODUCTION TO DATA SCIENCE

UNIT V: DATA WRANGLING COMBINING AND MERGING DATA SETS

STRUCTURE

5.0 OBJECTIVES

The main goal of this module is to help students learn, understand and practice the data science approaches, which include the study of latest data science tools with latest programming languages. The main objectives of this module are data wrangling which includes data discovery, structuring, cleaning, enriching, validating and publishing, combining and merging datasets, data reshaping, pivoting, transformation, string manipulation operations and regular expression.

5.1 INTRODUCTION

Data science become a buzzword that everyone talks about the data science. Data science is an interdisciplinary field that combines different domain expertise, computer programming skills, mathematics and statistical knowledge to find or extract the meaningful or unknown patterns from unstructured and structure dataset.

Data science is useful for extraction, preparation, analysis and visualization of various information. Various scientific methods can be applied to get insight in data.

Data science is all about using data to solve problems. Data has become the fuel of industries. It is most demandable field of 21st century. Every industry require data to functioning, searching, marketing, growing, expanding their business.

The application of areas of data science are health care, fraud detection, disease predicting, real time shipping routes, speech recognition, targeting advertising, gaming and many more.

5.2 DATA WRANGLING

Data wrangling is a key component of any data science project. Data wrangling is a process where one transforms *"row"* data for making it more suitable for analysis and it will improve the quality of the data. Data wrangling is the process of collecting, gathering and transforming of raw data into appropriate format for accessing, analyzing, easy understanding and further processing for better and quick decision making. It is also known as Data Munging or Data Pre-Processing.

Data wrangling is a crucial first steps in the preparation of data for broad analysis of huge amount of data or big data. It requires significant amount of time and efforts. If data wrangling is properly conducted, it gives you insights into the nature of the data. It is not just a single time process but it is an iterative or repetitive process. Each step in the data wrangling process exposes the new potential ways for the data re-wrangling towards deriving the goal of complex data manipulation and analysis.

5.2.1 Steps for Data Wrangling

Data wrangling process includes six core activities:

- Discovery
- Structuring
- Cleaning
- Enriching
- Validating

- Publishing

- **Discovery:**

  Data discovering is an umbrella term. It describes the process to understand the dataset and insight into it. It involves the collection and evaluation of data from various sources and is often used to identify the spot trends and detecting the patterns of the data and gain instant insight.

  Now a day‟s large business organization or companies have a large amount of data related to the customers, suppliers, production, sells, marketing etc.

  For example, if a company have a customer database, you can identify that the most of the customers are from which part of the city or state or country.

- **Structuring:**

  Data is coming from the various sources with the difference formats. Structuring is necessary because of the different size and shape of the data. Data structuring is the process or actions that change the form or schema of the dataset. Data splitting into the columns, deleting some fields from the dataset, pivoting rows are the form of data structuring.

- **Cleaning:**

  Before start the data manipulation or data analysis, you need to perform the data cleaning. Data cleaning is the process to identify the data quality related issues, such as missing or mismatched values, duplicate records etc. and apply the appropriate transformation to correct, replace or delete those values or records from the dataset to make high quality of data.

- **Enriching:**

  The data is useful for decision making process in the business. The data needed to take business related decision can be stored into multiple files. To gather all necessary insights into single file and you need to enriched your existing dataset by performing joining and aggregating multiple data sources.

- **Validating:**

  After completion of data cleaning and data enriching, you need to check the accuracy of the data. If dataset is not accurate, it might be creating a problem. It is necessary to do the data validation. This is the final check that any missing or mismatched data was not corrected during the transformation process. It is also need to validate that output dataset has the intended structure and content before publishing it.

- **Publishing:**

  After successful completion of data discovering, structuring, cleaning, enriching and validating‟s, it‟s a time to published the wrangled output data for the further analytics processes, if any. The published data can be uploaded in the

organization‟s software or store into the file in a specific location where organizations peoples knows it is ready to use.

5.2.2 Tools for Data Wrangling

There are some tools available for the data wrangling. Some of the popular tools are as follow:

- **Python**
  Python is a most popular general-purpose high-level programming language. There are many popular Python libraries available for data science. The pandas is a most popular and open source library and it becomes a game changer for data science. It is a very fast, flexible, powerful and easy to used library which includes Data Frame to perform more complex operation such as data joining, data merging, data transformation etc. for in data science.

- **R**
  R is also popular, open source and more power tool for data science and management. It also supports many libraries such as dplyr, tidyr, ggplot2 etc. for data manipulation and data visualization.

- **Tabula**
  Tabula is a tool for liberating data tables trapped inside the PDF files. It allows the user to upload the files in PDF format and extract the selected rows and columns from any tables available in the PDF file. It supports to extract this data from PDF to CSV or Microsoft Excel file format.

- **DataWrangler**
  It is an interactive tool for data cleaning. It takes the read word data and transform it into data tables which can be used for further processing or analysis. It also supports to export the data tables in Microsoft Excel, R, Tabula etc.

- **OpenRefine**
  OpenRefine, previously known as GoogleRefine. It is a Java based open source powerful tool for manipulates the huge data. It is used for data loading, understanding, cleaning and transforming from one format to another format. It also supports to extending the data with web services

.

- **CSVKit**
  CSVKit is a suite for command line tools for converting and working with CSV file. It supports the covert the data from Excel to CSV, JSON to CSV, query with SQL etc.

5.3 COMBINING DATASET

Combining is the process to put two or more dataset together for further processing. The pandas library of Python provides easy functionality to combining the dataset together. Here we learn the combining dataset with concat, merge and join functions using pandas library.

- **Concat:** The concat() function is used for combining dataset across the rows or columns.
- **Merge:** The merge() function is used for combining the dataset on common columns.
- **Join:** The join() function is used for combining the dataset on key column or an index.

5.4 CONCATENATING DATASET

The ***"concat"*** function is used to perform the concatenation operation with the data frame along with an axis. The datasets are just stitched together along with axis (rows axis and column axis). Here we concatenate the datasets using pandas.

Syntax:

```
pd.concat(objs, axis, join, join_axes, ignore_index, keys)
```

Here,

- **objs:** This is a sequence or mapping of Series, DataFrame, objects.
- **axis:** This is an axis to concatenate. This value is {*0, 1, ...*}, default is 0.
- **join:** This is used how to handle indexes on another axis(es). This value is {*„inner", „outer"*}, default is *„outer"*. The outer is used for union operation and inner is used for intersection operation.
- **join_axes:** This is the list of index objects. Its specific indexes to use for the other (n-1) axes instead of performing inner/outer set logic.
- **ignore_index:** This value is Boolean type, default is False. If this value is True, do not use the index values on the concatenation axis. The resulting axis will be labeled 0, ..., n-1.
- **keys:** This is a sequence to add an identifier to the result indexes, default is None.

Example: Here we perform the concatenation operation using two different data frames i.e. df1 and df2.

The below code creates the two different data frame df1 and df2.

```
# Importing pandas library
import pandas as pd

# First dataframe creation
df1 = pd.DataFrame({
      "Name":["Rahul","Shreya","Pankaj","Monika","Kalpesh"],
      "Age":[20, 19, 24, 25, 25],
      "Gender":["Male","Female","Male","Male","Female"],
```

```
        "Course":["B.E.","B.Tech.","MCA","M.Tech.","M.E."]},
        index = [1, 2, 3, 4, 5])

# Second dataframe creation
df2 = pd.DataFrame({
        "Name":["Mayank","Jalpa","Sanjana","Vimal","Raj"],
        "Age":[22, 21, 24, 26, 23],
        "Gender":["Male","Female","Female","Male","Male"],
        "Course":["MBA","MCA","B.E.","B.Tech.","M.Sc."]},
        index = [1, 2, 3, 4, 5])
```

Now we display both data frames df1 and df2.
The below code will display data frame df1.

```
# Display first dataframe
df1
```

The above code will give the following output.

	Name	Age	Gender	Course
1	Rahul	20	Male	B.E.
2	Shreya	19	Female	B.Tech.
3	Pankaj	24	Male	MCA
4	Monika	25	Male	M.Tech.
5	Kalpesh	25	Female	M.E.

The below code will display data frame df2.

```
# Display second dataframe
df2
```

The above code will give the following output.

	Name	Age	Gender	Course
1	Mayank	22	Male	MBA
2	Jalpa	21	Female	MCA
3	Sanjana	24	Female	B.E.
4	Vimal	26	Male	B.Tech.
5	Raj	23	Male	M.Sc.

Now we perform the concatenation operation on both data frames.
The below code will perform the concatenation operation on both the data frame df1 and df2.

```
# Concatenation of both dataframe
df3 = pd.concat([df1, df2])
df3
```

The above code will give the following output, which concatenate the five records of data frame df1 and five records of data frame df2 into single data frame df3 with ten records.

	Name	Age	Gender	Course
1	Rahul	20	Male	B.E.
2	Shreya	19	Female	B.Tech.
3	Pankaj	24	Male	MCA
4	Monika	25	Male	M.Tech.
5	Kalpesh	25	Female	M.E.
1	Mayank	22	Male	MBA
2	Jalpa	21	Female	MCA
3	Sanjana	24	Female	B.E.
4	Vimal	26	Male	B.Tech.
5	Raj	23	Male	M.Sc.

Now we perform the concatenation with axis as an argument.
The below code will perform the concatenation operation on both data frame df1 and df2 with using axis as an argument.

```
# Concatenation of both dataframe with axis as an argument
df3 = pd.concat([df1, df2],axis=1)
df3
```

The above code will give the following output, which concatenate the data frame df1 and data frame df2 into single data frame df3 horizontally with ***axis=1*** as an argument.

SN	Name	Age	Gender	Course	Name	Age	Gender	Course
1	Rahul	20	Male	B.E.	Mayank	22	Male	MBA
2	Shreya	19	Female	B.Tech.	Jalpa	21	Female	MCA
3	Pankaj	24	Male	MCA	Sanjana	24	Female	B.E.
4	Monika	25	Male	M.Tech.	Vimal	26	Male	B.Tech.
5	Kalpesh	25	Female	M.E.	Raj	23	Male	M.Sc.

Now we perform the concatenation with keys as an argument which is associated with specific keys.
The below code will perform the concatenation operation on both data frame df1 and df2 with keys as an argument.

```
# Concatenation of both dataframe with keys as an argument
df3 = pd.concat([df1, df2], keys=['x','y'])
df3
```

The above code will give the following output, which concatenate the data frame df1 and data frame df2 into single data frame df3 with ***x*** and ***y*** as ***keys*** argument.

		Name	Age	Gender	Course
x	**1**	Rahul	20	Male	B.E.
	2	Shreya	19	Female	B.Tech.
	3	Pankaj	24	Male	MCA
	4	Monika	25	Male	M.Tech.
	5	Kalpesh	25	Female	M.E.
y	**1**	Mayank	22	Male	MBA
	2	Jalpa	21	Female	MCA
	3	Sanjana	24	Female	B.E.
	4	Vimal	26	Male	B.Tech.
	5	Raj	23	Male	M.Sc.

Now we perform the concatenation with ***keys*** and ***ignore_index*** as an argument. It follows its own indexing if we set ignore_index is True.
The below code will perform the concatenation operation on both data frame df1 and df2 with ***keys*** and ***ignore_index*** as an argument.

```
# Concatenation of both dataframe using keys argument
df3 = pd.concat([df1, df2], keys=['x','y'], ignore_index=True)
df3
```

The above code will give the following output, which concatenate data frame df1 and data frame df2 into single data frame df3 using ***x*** and ***y*** as ***keys*** arguments and ***ignore_index=True*** argument.

	Name	Age	Gender	Course
0	Rahul	20	Male	B.E.
1	Shreya	19	Female	B.Tech.
2	Pankaj	24	Male	MCA
3	Monika	25	Male	M.Tech.
4	Kalpesh	25	Female	M.E.
5	Mayank	22	Male	MBA
6	Jalpa	21	Female	MCA
7	Sanjana	24	Female	B.E.
8	Vimal	26	Male	B.Tech.
9	Raj	23	Male	M.Sc.

Now we perform the concatenation using append() function.
The below code will perform the concatenation operation on both data frame df1 and df2. The data frame df2 is appended with the data frame df1.

```
# Concatenation of both dataframe using append
df1.append(df2)
```

The above code will give the following output, which concatenate the data frame df2 with the data frame df1.

	Name	Age	Gender	Course
1	Rahul	20	Male	B.E.
2	Shreya	19	Female	B.Tech.
3	Pankaj	24	Male	MCA
4	Monika	25	Male	M.Tech.
5	Kalpesh	25	Female	M.E.
1	Mayank	22	Male	MBA
2	Jalpa	21	Female	MCA
3	Sanjana	24	Female	B.E.
4	Vimal	26	Male	B.Tech.
5	Raj	23	Male	M.Sc.

5.5 MERGING DATASET

The word ***"merge"*** and ***"join"*** both are used relatively interchangeable in SQL, R and Pandas. Both merge and join doing the similar things, but there are separate "merge" and "join" functions in Pandas.

The merging/joining is the process of bringing two or more datasets together into single dataset and aligning the rows from each dataset based on the common attributes or columns.

The ***„merge"*** function is used to perform the merging operation with the data frame. Here we merge the datasets using pandas.

Syntax:

```
pd.merge(left, right, how, on, left_on, right_on, left_index, right_index, sort)
```

Here,

- **left:** This is the first data frame.
- **right:** This is the second data frame.
- **how:** This is the method how to perform the merge operation. The values of this field are one of *„left", „right", „inner", „outer"*, default is *„inner"*.
- **On:** This is the name of column in which action to be perform. This column must be available in both left and right data frame object.
- **left_on:** This is the name of column from left data frame to use as keys.
- **right_on:** This is the name of column from right data frame to use as keys.
- **left_index:** This is using the index (row label) from left data frame as its join keys, if it is True.
- **right_index:** This is used the index (row label) from right data frame as its join keys, if it is True.

- **sort:** This is use to sort the result data frame by join keys in specific order. The value of this field is Boolean, default is True.

Example: Here we perform the merge operation using two different data frames i.e. left and right.

The below code creates two different data frames left and right.

```
# Importing pandas library
import pandas as pd

# Left dataframe creation
left = pd.DataFrame({
      "Rno":[1, 2, 3, 4, 5],
      "Name":["Rahul","Shreya","Pankaj","Monika","Kalpesh"],
      "Course":["B.E.","B.Tech.","MCA","M.Tech.","M.E."]})

# Right dataframe creation
right = pd.DataFrame({
      "Rno":[1, 2, 3, 4, 5],
      "Name":["Mayank","Jalpa","Sanjana","Vimal","Raj"],
      "Course":["B.E.","MBA","MCA","B.Tech.","M.Sc."]})
```

Now we display both data frame left and right.

The below code will display data frame left.

```
# Display left dataframe
left
```

The above code will give the following output.

	Rno	Name	Course
0	1	Rahul	B.E.
1	2	Shreya	B.Tech.
2	3	Pankaj	MCA
3	4	Monika	M.Tech.
4	5	Kalpesh	M.E.

The below code will display the data frame right.

```
# Display right dataframe
right
```

The above code will give the following output.

	Rno	Name	Course
0	1	Mayank	B.E.
1	2	Jalpa	MBA
2	3	Sanjana	MCA
3	4	Vimal	B.Tech.
4	5	Raj	M.Sc.

Now we perform the merge operation on both data frame using ***on*** as an argument.

The below code will perform the merge operation on both data frame left and right using single ***on*** key as an argument.

```
# Merge both left and right dataframe using single on key
pd.merge(left, right, on='Rno')
```

The above code will give the following output, which merge both data frame left and right using single ***on*** key as an argument.

	Rno	Name_x	Course_x	Name_y	Course_y
0	1	Rahul	B.E.	Mayank	B.E.
1	2	Shreya	B.Tech.	Jalpa	MBA
2	3	Pankaj	MCA	Sanjana	MCA
3	4	Monika	M.Tech.	Vimal	B.Tech.
4	5	Kalpesh	M.E.	Raj	M.Sc.

The below code will perform the merge operation on both data frame left and right using multiple ***on*** key as an argument.

```
# Merge left and right dataframe using multiple on keys
pd.merge(left, right, on=['Rno','Course'])
```

The above code will give the following output, which merge both data frame left and right using multiple ***on*** key as an argument.

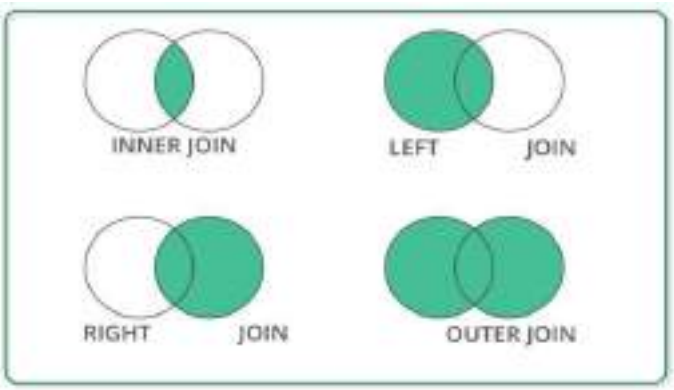
	Rno	Name_x	Course	Name_y
0	1	Rahul	B.E.	Mayank
1	3	Pankaj	MCA	Sanjana

Now we perform the merge operation using ***how*** as an argument. This argument specifies how to determine which keys are to be included in the resulting data frame or table. If the combination does not appear in any of the data frame or table, NA will be display in joined table.

The merge methods are same as SQL join equivalent as below:

Merge Method	SQL Join Equivalent	Description
left	Left Outer Join	Use keys from left object
right	Right Outer Join	Use keys from right object
inner	Inner Join	Use intersection of keys
outer	Full Outer Join	Use unions of keys

It will represent graphically as follows:



The below code will perform the merge operation on both data frame left and right ***on*** course using ***how=„left"*** method.

```
# Merge both left and right dataframe using on and how
pd.merge(left, right, on='Course', how='left')
```

The above code will give the following output, which merge both data frame left and right using left join. It will display left data frame records plus common records as follows:

	Rno_x	Name_x	Course	Rno_y	Name_y
0	1	Rahul	B.E.	1.0	Mayank
1	2	Shreya	B.Tech.	4.0	Vimal
2	3	Pankaj	MCA	3.0	Sanjana
3	4	Monika	M.Tech.	NaN	NaN
4	5	Kalpesh	M.E.	NaN	NaN

The below code will perform the merge operation on both data frame left and right ***on*** course using ***how=„right"*** method.

```
# Merge both left and right dataframe using on and how
pd.merge(left, right, on='Course', how='right')
```

The above code will give the following output, which merge both data frame left and right using right join. It will display right data frame records plus common records as follows:

	Rno_x	Name_x	Course	Rno_y	Name_y
0	1.0	Rahul	B.E.	1	Mayank
1	NaN	NaN	MBA	2	Jalpa
2	3.0	Pankaj	MCA	3	Sanjana
3	2.0	Shreya	B.Tech.	4	Vimal
4	NaN	NaN	M.Sc.	5	Raj

The below code will perform the merge operation on both data frame left and right ***on*** course using ***how=„inner"*** method.

```
# Merge both left and right dataframe using on and how
pd.merge(left, right, on='Course', how='inner')
```

The above code will give the following output, which merge both data frame left and right using inner join. It performs the intersection operation on both data frame, which will display only common records as follows:

	Rno_x	Name_x	Course	Rno_y	Name_y
0	1	Rahul	B.E.	1	Mayank
1	2	Shreya	B.Tech.	4	Vimal
2	3	Pankaj	MCA	3	Sanjana

The below code will perform the merge operation on both data frame left and right ***on*** course using ***how=„over"*** method.

```
# Merge both left and right dataframe using on and how
pd.merge(left, right, on='Course', how='outer')
```

The above code will give the following output, which merge both data frame left and right using outer join. It performs the union operation on both data frame, which will display left data frame records, right data frame records and common records as follows:

	Rno_x	Name_x	Course	Rno_y	Name_y
0	1.0	Rahul	B.E.	1.0	Mayank
1	2.0	Shreya	B.Tech.	4.0	Vimal
2	3.0	Pankaj	MCA	3.0	Sanjana
3	4.0	Monika	M.Tech.	NaN	NaN
4	5.0	Kalpesh	M.E.	NaN	NaN
5	NaN	NaN	MBA	2.0	Jalpa
6	NaN	NaN	M.Sc.	5.0	Raj

5.6 RESHAPING DATASET

The way in which dataset is arranged into rows and columns is referred as the shape of data. In a dataset, each row represents one observation in a vertical or long data and each column is considered a variable with multiple distinct values.

It is needed to convert or transform the dataset from one format to another format, which is called reshaping of dataset.

Reshaping is the process to change the shape or structure of datasets, such as convert ***"wide"*** data tables into ***"long".*** The below figure shows the reshaping process graphically.

The data frame will be reshaped by using melt(), stack(), unstack() and pivot() functions as follows:

The below code creates and display data frame df1.

```
# Importing pandas library
import pandas as pd

# Dataframe creation
df = pd.DataFrame({
      "Rno":[1, 2, 3, 4, 5],
      "Name":["Rajan","Shital","Mayur","Mittal","Mahesh"],
      "Age":[25, 27, 24, 25, 21]})

# Display dataframe
df
```

The data frame output as follows:

	Rno	Name	Age
0	1	Rajan	25
1	2	Shital	27
2	3	Mayur	24
3	4	Mittal	25
4	5	Mahesh	21

5.6.1 Using melt() Function

We can reshape the data frame using this function. This function is used to wide data frame columns into rows.

The below code will perform the reshape operation using melt function.

```
# Performing melt function on dataframe
df.melt()
```

The above code will give the following output.

	Variable	value
0	Rno	1
1	Rno	2
2	Rno	3
3	Rno	4
4	Rno	5
5	Name	Rajan
6	Name	Shital
7	Name	Mayur
8	Name	Mittal
9	Name	Mahesh
10	Age	25
11	Age	27
12	Age	24
13	Age	25
14	Age	21

5.6.2 Using stack() and unstack() Function

We can reshape the data frame using these functions. The stack() function is used to increase the level of index in a data frame.

The below code will perform the reshape operation using stack function.

```
# Performing stack function on dataframe
df.stack()
```

The above code will give the following output.

0	Rno	1
	Name	Rajan
	Age	25
1	Rno	2
	Name	Shital
	Age	27
2	Rno	3
	Name	Mayur
	Age	24

3	Rno	4
	Name	Mittal
	Age	25
4	Rno	5
	Name	Mahesh
	Age	21
dtype: object		

The unstack() function is used to do the revert back changes in a data frame was perform by the stack() function.

The below code will perform the reshape operation using unstack function.

```
# Performing unstack function on dataframe
df.unstack()
```

The above code will give the following output.

Rno	0	1
	1	2
	2	3
	3	4
	4	5
Name	0	Rajan
	1	Shital
	2	Mayur
	3	Mittal
	4	Mahesh
Age	0	25
	1	27
	2	24
	3	25
	4	21
dtype: object		

5.6.3 Using pivot() Function

We can reshape the data frame using this function. This function is used to reshape the data frame based on the specified column in a data frame.

The below code will perform the reshape operation on **"Rno"** column using pivot function.

```
# Performing pivot function on dataframe
df.pivot(columns='Rno')
```

The above code will give the following output.

	Name					Age				
Rno	**1**	**2**	**3**	**4**	**5**	**1**	**2**	**3**	**4**	**5**
0	Rajan	NaN	NaN	NaN	NaN	25.0	NaN	NaN	NaN	NaN
1	NaN	Shital	NaN	NaN	NaN	NaN	27.0	NaN	NaN	NaN
2	NaN	NaN	Mayur	NaN	NaN	NaN	NaN	24.0	NaN	NaN
3	NaN	NaN	NaN	Mittal	NaN	NaN	NaN	NaN	25.0	NaN
4	NaN	NaN	NaN	NaN	Mahesh	NaN	NaN	NaN	NaN	21.0

5.7 DATA TRANSFORMATION

Data is collected from the various sources and combine it into a unified data frame. This data frame has large number of columns with different data types.

Data transformation is the process to transform or convert the data as per required format for further processing as and when needed.

The data transformation includes add new columns, find NaN values, drop NaN, replace with mean value, field encoding and decoding, column splitting etc.

5.7.1 Data Frame Creation

To perform the various transformation operation on data, first we have to create the data frame.

The below code will create and display a data frame df which contains the three columns such as **"Name", "Gender"** and **"City"**.

```
#importing pandas library
import pandas as pd

# Dataframe creation
df = pd.DataFrame({
   "Name":["Jayesh Patel","Priya Shah","Vijay Sharma"],
   "Gender": ["Male","Female","Male"],
   "City": ["Rajkot","Delhi","Mumbai"]})

# Display dataframe
df
```

The above code will create and display data frame as follows:

	Name	Gender	City
0	Jayesh Patel	Male	Rajkot
1	Priya Shah	Female	Delhi
2	Vijay Sharma	Male	Mumbai

5.7.2 Missing Value

The dataset contains the many rows and columns. There are some cells in a dataset that have NA or empty cell. This is called missing data in a dataset.

It is needed first to check that missing value before further processing. The common and very simple method to handle this missing value is to delete the rows which contain missing values.

The below code will check the data to containing the missing value or not.

```
df.isna().sum()
```

Here there are no any missing values in each column so it returns zero value in each column.

If there are any missing values in each column, it returns the number of missing values.

```
Name      0
Gender    0
City      0
dtype: int64
```

The below code will drop all the rows which containing missing value.

```
df = df.dropna()
```

The below code will drop the columns where all elements are missing values.

```
df.dropna(axis=1, how='all')
```

The below code will drop the columns where any of the elements containing missing values.

```
df.dropna(axis=1, how='any')
```

The below code will keep only the rows which contains maximum two missing values.

```
df.dropna(thresh=2)
```

The below code will fill all missing values with mean value of the particular column.

```
df.fillna(df.mean())
```

The below code will fill any missing values in specified column with median value of the particular column. Here we taken “Age” column for example.

```
df['Age'].fillna(df['Age'].median())
```

The below code will fill any missing values in specified column with mode value of the particular column. Here we taken “Age” column for example.

```
df['Age'].fillna(df['Age'].mode())
```

5.7.3 Encoding

The dataset contains both numerical and categorical value. The categorical data is not much useful for data processing or analytics. It is needed to encoding the categorical value into numeric value.

Data encoding is the process to convert a categorical variable into a numerical form.

Here we discuss the label encoding which is simply converting each value in a column to a number. Our dataset has **“Gender”** column which has only two values **“male”** and **“female”** It encodes like this:

- Male → 0
- Female → 1

The below code will replace the “Male” to 0 and “Female” to 1.

```
df=df.Gender.replace({"Male":0,"Female":1})
df
```

The above code will display data frame as follows:

```
0   0
1   1
2   0
Name: Gender, dtype: int64
```

5.7.4 Inset New Column

It is needed to add one or more columns in an existing dataset.

The below code will insert a new column in a data frame and display it.

```
df.insert(1, "Age", [21, 23, 24], True)
df
```

The above code will insert a new column **“Age”** with the values **21, 23** and **24** respectively at second column in a data frame. The newly added column data frame will display as follow:

	Name	Age	Gender	City
0	Jayesh Patel	21	Male	Rajkot
1	Priya Shah	23	Female	Delhi
2	Vijay Sharma	24	Male	Mumbai

5.7.5 Split Column

It is needed to split one column into two or more different columns. The process to create two or more different columns from single column in a dataset is called column splitting. Sometimes the **"Full Name"** column of dataset may be need to split into **"First Name"** and **"Last Name"** as a separate column.

The below code will split the column into two different columns and display new data frame.

```
df[['First Name','Last Name']] = df.Name.str.split(expand=True)
df
```

The above code will create two different columns (i.e. First Name, Last Name) from the single column **"Name"** of dataset. The newly split data frame will be display as follow:

	Name	Age	Gender	City	First Name	Last Name
0	Jayesh Patel	21	Male	Rajkot	Jayesh	Patel
1	Priya Shah	23	Female	Delhi	Priya	Shah
2	Vijay Sharma	24	Male	Mumbai	Vijay	Sharma

5.8 STRING MANIPULATION

String manipulation is the process of handling and analyzing the strings. The various operation can be performed on string such as string modification, parsing of string, string conversion etc. The various in-built functions available for string manipulation in different language. He we perform some common string manipulation operations in Python.

We take the following strings as an example.

```
str1 = "Data Science"
str2 = "using"
str3 = "Python"
str4 = "2021"
str5 = "   Data Science   "
```

Function	Description	Example	Output
capitalize()	It converts the first character of string into upper case	str2.capitalize()	'Using'
casefold()	It converts the string into lower case	str1.casefold()	'data science'
center()	It returns the string in center of the specified size	str1.center(15)	' Data Science '
count()	It returns the number of times a specified value occurs in a string	str1.count("a")	2
endswith()	It returns true if the string ends with the specified value	str1.endswith("nce")	True
find()	It searches the string for a specified value and returns the position of where it is found	str1.find("i")	7
format()	It formats the specified values in a string	str4.format()	'2021'
index()	It searches the string for a specified value and returns the position of where it was found	str1.index("c")	6
isalnum()	It returns True if all the characters in a string are alphanumeric	str4.isalnum()	True
isalpha()	It returns True if all characters in a string are alphabet	str2.isalpha()	True
isdigit()	It returns True if all characters in a string are digit	str4.isdigit()	True
islower()	It returns True if all characters in a string are lower case	str2.islower()	True
isnumeric()	It returns True if all characters in a string are numeric	str4.isnumeric()	True
isprintable()	It returns True if all characters in a string are printable	str1.isprintabl()	True
isspace()	It returns True if all characters in a string are whitespaces	str1.isspace()	False
istitle()	It returns True if the string follows the title case rules	str1.istitle()	True
isupper()	It returns True if all characters in a string are upper case letter	str1.isupper()	False
len(string)	It returns the length of a string	len(str1)	12
lower()	It converts the string into lower case	str1.lower()	'data science'
lstrip()	It returns the string with left trim version	str5.lstrip()	'Data Science '
replace(old,new)	It replaces the old string with new string	str1.replace("Science","Analytics")	'Data Analytics'
rfind()	It searches the string for a specified value and returns the last position of	str1.rfind("e")	11

	where it was found		
rindex()	It searches the string for a specified value and returns the last index position of where it is found	str1.rindex("a")	3
rstrip()	It returns the string with right trim version	str5.rstrip()	' Data Science'
split()	It splits the string with specified separator and returns a list	str1.split(" ")	['Data', 'Science']
startswith()	It returns true if the string starts with the specified value	str3.startswith ("P")	True
strip()	It returns the both left and right trim version	str5.strip()	'Data Science'
swapcase()	It returns the swaps cases, lower case becomes upper case and vice versa	str1.swapcase ()	'dATA sCIENCE'
title()	It converts the first character of each word to upper case	str2.title()	'Using'
upper()	It converts a string into upper case	str1.upper()	'DATA SCIENCE'
zfill()	It returns the string with filled by 0 for specified number of times in a string at the beginning	str3.zfill(10)	'0000Python'
+	It concatenates or join the two strings	str1+" "+str2+" "+str3	'Data Science using Python'
*	It repeated the string using n times	str3 * 3	'PythonPythonPython'
string[0]	It returns the first character of a string	str1[0]	'D'
string[7]	It returns the eighth character of a string	str1[7]	'i'
string[2:8]	It returns the string from third character to eighth character	str1[2:8]	'ta Sci'
string[3:]	It returns the string from fourth character to last character	str1[3:]	'a Science'
string[:8]	It returns the string from first character to eighth character	str1[:8]	'Data Sci'
string[-4:]	It returns the last four character of a sting	str1[-4:]	'ence'
string[:-4]	It removes the last four character of a string	str1[:-4]	'Data Sci'

5.9 REGULAR EXPRESSION

Regular expression or RegEx is generally used to identify whether a sequence or character or pattern is exists in a given string or not. It is also used to identify the position of such pattern in a string or file. It mainly used to find and replace specific patterns in a string or file. It helps to manipulate text-based datasets.

Python has a built-in package called ***re***, to work with Regular Expression.

The ***re*** package of Python has set of functions to search the string for matching. The common Python RegEx functions are as follow:

Function	Description
findall	It returns a list of all match values.
search	It returns a match object if match found anywhere in the string.
split	It returns a list and spit the string where each match found
sub	It replaces the one or more matches with a specified string.

5.9.1 The findall() Function

This function returns a list which contains all match values.

Example:

```
import re
string = "Working with Data Science using Python"
result = re.findall("th",string)
result
```

The list contains all match values in an order of they are found.

Output:

```
['th', 'th']
```

Example:

```
import re
string = "Working with Data Science using Python"
result = re.findall("ds",string)
result
```

If no matches found, an empty list will return.

Output:

```
[]
```

5.9.2 The search() Function

This function returns a match object if match found anywhere in the string. Only first occurrence of match will be returned, if there are more than one match found. The none will return if no match found.

Example:

```
import re
string = "Working with Data Science using Python"
result = re.search("wi",string)
```

```
result
```

It found the match value "***wi***".

Output:

```
<re.Match object; span=(8, 10), match='wi'>
```

Example:

```
import re
string = "Working with Data Science using Python"
result = re.search("\s",string)
result
```

It found the match value "**\s**" i.e. space.

Output:

```
<re.Match object; span=(7, 8), match=' '>
```

5.9.3 The split() Function

This function returns a list of where the string has been split at each match found.

Example:

```
import re
string = "Working with Data Science using Python"
result = re.split("\s",string)
result
```

It found the match value **"\s"** i.e. space.

Output:

```
['Working', 'with', 'Data', 'Science', 'using', 'Python']
```

Example:

```
import re
string = "Working with Data Science using Python"
result = re.split("\s",string,2)
result
```

It found the match value **"\s"** i.e. space. But here the string will split into first 2 occurrence of found only. The remaining string will print as it is.

Output:

```
['Working', 'with', 'Data Science using Python']
```

5.9.4 The sub() Function

This function replaces the string with the specified text at each match found.

It will print same string, if not match found.

Example:

```
import re
string = "Working with Data Science using Python"
result = re.sub("\s", "-",string)
result
```

It found the match value **"\s"** i.e. space. Every **"\s"** (space) will replace with the **"-"** (dash).

Output:

```
'Working-with-Data-Science-using-Python'
```

Example:

```
import re
string = "Working with Data Science using Python"
result = re.sub("\s","-",string,2)
result
```

It found the match value **"\s"** i.e. space. But here **"\s"** (space) will replace with the **"-"** (dash) in first two occurrence of found only. The remaining string will print as it is.

Output:

```
'Working-with-Data Science using Python'
```

5.10 SUMMARY

The students will learn many things related to data pre-processing in this module and they will be able to perform the various data science related operation using Python.

- Ability to do the data wrangling which includes data discovery structuring, cleaning, enriching, validating and publishing.
- Ability to do the combining different datasets using concat, merge and join function with different arguments.
- Ability to do the reshaping of dataset using melt, stack and unstack and pivot functions.
- Ability to work with missing value, encoding categorical data into numerical value, splitting dataset etc.
- Ability to perform the various functions related to string and regular expression.

REFERENCES

Books

1. Davy Cielen, Arno D. B. Meysman, Mohamed Ali : Introducing Data Science, Manning Publications Co.
2. Stephen Klosterman (2019) : Data Science Projects with Python, Packt Publishing

3. Jake VanderPlas (2017) : Python Data Science Handbook: Essential Tools for Working with Data, O‟Reilly
4. Wes McKinnery and Pandas Development Team (2021) : pandas : powerful Python data analysis toolkit, Release 1.2.3

Web References

1. https://www.geeksforgeeks.org
2. https://www.tutorialspoint.com
3. https://www.w3schools.com
4. https://pandas.pydata.org
5. https://pbpython.com

QUESTIONS

Short Answer:

1. What is data wrangling?
2. What is data cleaning?
3. List tools for data wrangling.
4. What is merging dataset?
5. What is reshaping dataset?

Long Answer:

1. Explain steps for data wrangling process.
2. Explain concat function with example.
3. Explain merge operation with syntax and example.
4. Explain reshaping dataset different functions.
5. Explain string function with example.
6. Explain regular expression with example.

PRACTICALS

1. Create and display a data frame.
2. Create a data frame and find null values and remove it.
3. Create a data frame and inert new column in data frame.
4. Create a data frame and convert categorical data into numerical values.
5. Create a data frame and split the column.
6. Create data frames and combine using concat function.
7. Create data frames and merge with different arguments.
8. Create data frame and reshape using melt function.
9. Perform string manipulation operations on string.
10. Perform regular expression functions on string.

INTRODUCTION TO DATA SCIENCE

UNIT VI: AGGREGATION AND GROUP OPERATIONS GROUP BY MECHANICS

STRUCTURE

6.0 OBJECTIVES

The main goal of this module is to help students learn, understand and practice the data science approaches, which include the study of latest data science tools with latest programming . The main objectives of this module are data aggregation, group wise operation including data splitting, applying and combining, data transformation using lamda function, pivot table, cross tabulation using two-way and three-way cross table and data and time data type.

6.1 INTRODUCTION

Data science become a buzzword that everyone talks about the data science. Data science is an interdisciplinary field that combines different domain expertise, computer programming skills, mathematics and statistical knowledge to find or extract the meaningful or unknown patterns from unstructured and structure dataset.

Data science is useful for extraction, preparation, analysis and visualization of various information. Various scientific methods can be applied to get insight in data.

Data science is all about using data to solve problems. Data has become the fuel of industries. It is most demandable field of 21st century. Every industry require data to functioning, searching, marketing, growing, expanding their business.

The application of areas of data science are health care, fraud detection, disease predicting, real time shipping routes, speech recognition, targeting advertising, gaming and many more.

6.2 Data Aggregation

Data aggregation is the process to gather the raw data and express in a summarised form for statistical analysis. Data may be collected from various sources and combine into a summary format for data analysis.

A dataset contains large amount of data in a rows and columns. There are thousands or more data records are in a single dataset. Data aggregation will useful to access and process the large amount of data quickly. Aggregate data can be access quickly to gain insight instead of accessing all the data records. A single raw of aggregated data can represent this large number of data records over a given time period to calculate the statistics such as sum, minimum, maximum, average and count.

- **Sum :** This function add all the specified data to get a total.
- **Min :** This function displays the lowest value of each specified category.
- **Max :** This function displays the highest value of each specified category.
- **Average :** This function calculates the average value of the specific data.
- **Count :** This function counts the total number of data entries for each category.

Data aggregation provides more insight information based on related cluster of data. For example, a company want to know the sales performance of different district, they would aggregate the sales data based on the district. Data can aggregate by date also, if you want to know the trends over a period of months, quarters, years, etc.

Example: Here, we will perform the aggregation operation on data frame.

The below code will create and display data frame df.

```
#importing pandas library
import pandas as pd

# Dataframe creation
df = pd.DataFrame({
   "Rno":[1,2,3,4,5,6,7,8,9,10],
   "Maths":[67,83,74,91,55,70,86,81,92,67],
   "Physics":[56,67,72,84,89,79,90,89,92,82],
   "Chemistry":[81,88,78,69,74,72,83,90,58,68],
   "Biology":[90,83,86,75,68,79,67,71,91,89],
   "English":[60,55,63,71,88,75,91,82,85,80]})

# Display dataframe
df
```

The above code will give the following output.

	Rno	Maths	Physics	Chemistry	Biology	English
0	1	67	56	81	90	60
1	2	83	67	88	83	55
2	3	74	72	78	86	63
3	4	91	84	69	75	71
4	5	55	89	74	68	88
5	6	70	79	72	79	75
6	7	86	90	83	67	91
7	8	81	89	90	71	82
8	9	92	92	58	91	85
9	10	67	82	68	89	8

Now, we perform the aggregation using min, max and sum.

The below code will find min and max value of different subject of data frame df.

```
df.aggregate(["min","max"])
```

The above code will give the following output.

	Rno	Maths	Physics	Chemistry	Biology	English
Min	1	55	56	58	67	55
Max	10	92	92	90	91	91

The below code will find min and max value and calculate average and sum value of different subject of data frame df.

```
df.agg(["min","max","average","sum"])
```

The above code will give the following output.

	Rno	Maths	Physics	Chemistry	Biology	English
min	1.0	55.0	56.0	58.0	67.0	55.0
max	10.0	92.0	92.0	90.0	91.0	91.0
average	5.5	76.6	80.0	76.1	79.9	75.0
sum	55.0	766.0	800.0	761.0	799.0	750.0

Now, we perform the different aggregation functions on different columns.

The below code will find min and max value of “Maths” subject max and sum of “Physics” subject and min, median and std of “English” subject of data frame df.

```
df.agg({"Maths":["min","max"],
    "Physics":["max","sum"],
    "English":["min","median","std"]})
```

The above code will give the following output.

Here, NaN is display where the specific function is not applying for particular subject.

	Maths	Physics	English
max	92.0	92.0	NaN
median	NaN	NaN	77.500000
min	55.0	NaN	55.000000
std	NaN	NaN	12.400717
sum	NaN	800.0	NaN

Now, we will apply sum function on each column.

The below code will calculate sum of each column of data frame df.

```
df.sum()
```

The above code will give the following output.

Here, the sum of score of each subject will be display.

```
Rno          55
Maths        766
Physics      800
Chemistry    761
Biology      799
English      750
dtype: int64
```

Now, we will apply min function on each column.

The below code will find min value of each column of data frame df.

```
df.min()
```

The above code will give the following output.

Here, the min value of score of each subject will be display.

```
Rno          1
Maths        55
Physics      56
Chemistry    58
Biology      67
English      55
dtype: int64
```

Now, we will apply max function on each column.

The below code will find max value of each column of data frame df.

```
df.max()
```

The above code will give the following output.

Here, the max value of score of each subject will be display.

```
Rno          10
Maths        92
Physics      92
Chemistry    90
Biology      91
English      91
dtype: int64
```

Now, we will apply mean function on each column.

The below code will calculate mean value of each column of data frame df.

```
df.mean()
```

The above code will give the following output.

Here, the mean value of score of each subject will be display.

```
Rno          5.5
Maths        76.6
Physics      80.0
Chemistry    76.1
Biology      79.9
English      75.0
dtype: float64
```

Now, we will apply count function on each column.

The below code will count the numbers of values in each column of data frame df.

```
df.count()
```

The above code will give the following output.

Here, the total numbers of values of score of each subject will be display.

```
Rno          10
Maths        10
Physics      10
Chemistry    10
Biology      10
English      10
dtype: int64
```

Now we will apply std function on each column.

The below code will calculate standard deviation of each column of data frame df.

```
df.std()
```

The above code will give the following output.

Here, the standard deviation of score of each subject will be display.

```
Rno          3.027650
Maths       11.992590
Physics     11.718931
Chemistry    9.859570
Biology      9.230986
English     12.400717
dtype: float64
```

Now, we will apply describe function.

The below code will calculate the basic statistics of each column of data frame df.

```
df.describe()
```

The above code will give the following output.

	Rno	Maths	Physics	Chemistry	Biology	English
count	10.00000	10.00000	10.000000	10.00000	10.000000	10.000000
mean	5.50000	76.60000	80.000000	76.10000	79.900000	75.000000
std	3.02765	11.99259	11.718931	9.85957	9.230986	12.400717
min	1.00000	55.00000	56.000000	58.00000	67.000000	55.000000
25%	3.25000	67.75000	73.750000	69.75000	72.000000	65.000000
50%	5.50000	77.50000	83.000000	76.00000	81.000000	77.500000
75%	7.75000	85.25000	89.000000	82.50000	88.250000	84.250000
max	10.00000	92.00000	92.000000	90.00000	91.000000	91.000000

6.3 GROUP WISE OPERATION

The groupby() function is used to perform the group wise operation on a large dataset. This is a versatile and easy to use function which help to get summary of large dataset. The summary is easy to explore the dataset and shows the relationship between variables.

We can create a grouping of different categories and apply various functions to each category. This function is widely used in real data science projects which dealing with large

amounts of data. It has ability to aggregate data efficiently. This function refers to a process of involving one or more of the following steps:

- **Splitting:** It is a process in which we split the dataset into different groups based on some criteria.
- **Applying:** It is a process in which we apply different functions to each group independently. To apply the function to each group, we perform some operations:
  - **Aggregation:** It is a process to compute a statistical summary of the group such as sum, mean, median, etc.
  - **Transformation:** It is a process to perform some group specific computations and return a like-indexed such as filling NA within group with a value derived from each group.
  - **Filtration:** It is a process to remove some groups based on some criteria such as filtering out dataset based on group wise sum or mean.
- **Combining:** It is a process to combine different datasets after applying groupby function and results will store in a dataset.

The syntax of groupby function is as follows:

Syntax:

```
DataFrame.groupby(by=None, axis=0, level=None, as_index=True, sort=True, group_keys=True, squeeze=False, **kwargs)
```

Here,

- **by:** mapping, function, label, str
- **axis:** int, 0 or index and 1 or columns, default is 0, split along rows (0) or columns (1).
- **level:** int, level name, default is None, If the axis is a MultiIndex, group by a particular level or levels.
- **as_index:** boolean, default is True, for aggregated output, return object with group labels as the index.
- **sort:** boolean, default is True, sort group keys.
- **group_keys:** boolean, default is True, when calling apply, add group keys to index to identify pieces.
- **squeeze:** boolean, default is False, reduce the dimensionality of the return type, if possible

Example: Here, we will perform the groupby operation on data frame.

The below code will create and display data frame df.

```
# Importing pandas library
import pandas as pd

# Dataframe creation
df = pd.DataFrame({
```

```
"Product":["Mango","Corn","Orange","Cabbage","Mango","Corn","Watermelon","App
le","Pumkin","Mango",],
"Category":["Fruit","Vegetable","Fruit","Vegetable","Fruit","Vegetable","Fruit","Fruit
","Vegetable","Fruit"],
"Qty":[12, 5, 10, 2, 10, 3, 5, 8, 2, 10],
"Price":[350, 80, 320, 50, 200, 50, 280, 380, 60, 400]})



# Display dataframe
df
```

The above code will give the following output.

	Product	Category	Qty	Price
0	Mango	Fruit	12	350
1	Corn	Vegetable	5	80
2	Orange	Fruit	10	320
3	Cabbage	Vegetable	2	50
4	Mango	Fruit	10	200
5	Corn	Vegetable	3	50
6	Watermelon	Fruit	5	280
7	Apple	Fruit	8	380
8	Pumkin	Vegetable	2	60
9	Mango	Fruit	10	400

Now, we will perform the groupby operation on “Category” columns.

The below code will find sum of “Qty” and “Price” based on “Category” on data frame df.

```
df.groupby("Category").sum()
```

The above code will give the following output.

Here, the total Qty of “Fruit” category is 55 and total Price of “Fruit” category is 1930, while total Qty of “Vegetable” category is 12 and total Price of “Vegetable” category is 240.

Category	Qty	Price
Fruit	55	1930
Vegetable	12	240

Now, we will perform the groupby operation on “Product” columns.

The below code will find sum of “Qty” and “Price” based on different “Product” on data frame df.

```
df.groupby("Product").sum()
```

The above code will give the following output.

Product	Qty	Price
Apple	8	380
Cabbage	2	50
Corn	8	130
Mango	32	950
Orange	10	320
Pumkin	2	60
Watermelon	5	280

Now, we will perform the groupby operation on “Category” columns.

The below code will find mean of “Price” based on “Category” on data frame df.

```
df.groupby("Category")["Price"].mean()
```

The above code will give the following output.

```
Category
Fruit       321.666667
Vegetable    60.000000
Name: Price, dtype: float64
```

The below code will find mean of “Qty” based on “Category” on data frame df.

```
df.groupby("Category")["Qty"].mean()
```

The above code will give the following output.

```
Category
Fruit        9.166667
Vegetable    3.000000
Name: Qty, dtype: float64
```

Now, we will perform the groupby operation on “Product” columns.

The below code will find mean of “Price” based on “Product” on data frame df.

```
df.groupby("Product")['Price'].mean()
```

The above code will give the following output.

```
Product
Apple         380.000000
Cabbage        50.000000
Corn           65.000000
Mango         316.666667
Orange        320.000000
Pumkin         60.000000
Watermelon    280.000000
Name: Price, dtype: float64
```

Now, we will perform the groupby operation on “Category” columns.

The below code will find median of “Price” based on “Category” on data frame df.

```
df.groupby("Category")["Price"].median()
```

The above code will give the following output.

```
Category
Fruit        335
Vegetable     55
Name: Price, dtype: int64
```

The below code will find standard deviation of “Price” based on “Category” on data frame df.

```
df.groupby("Category")["Price"].std()
```

The above code will give the following output.

```
Category
Fruit        73.325757
Vegetable    14.142136
Name: Price, dtype: float64
```

6.4 Transformation

Transformation is a process in which we perform some group-specific computations and return a like-indexed. Transformation perform on a group or a column which returns an object that is indexed the same size of that is being grouped. Thus, the transform should return a result that is the same size as that of a group chunk.

Syntax:

```
DataFrame.transform(func, axis=0, *args, **kwargs)
```

Here,

- **func:** this is the function to use for data transformation.
- **axis:** the axis in which the transformation will perform, {0 or „index", 1 or „columns"}, default is 0.
- ***args:** positional arguments to pass in func.
- ****kwargs:** keyword arguments to pass in func.

Example: Here, we will perform some group specific operation and return a like-indexed.

The below code will create and display data frame df.

```
#importing pandas library
import pandas as pd

# Creating the DataFrame
df = pd.DataFrame({
    "A":[8, 7, 15, 12, 15],
    "B":[None, 22, 32, 9, 7],
    "C":[10, 6, None, 8, 14]})

# Display dataframe
df
```

The above code will give the following output.

	A	B	C

0	8	NaN	10.0
1	7	22.0	6.0
2	15	32.0	NaN
3	12	9.0	8.0
4	15	7.0	14.0

Now, we will perform the transform operation using lambda.

The below code will multiply by 5 to each value of all the columns A, B and C of data frame df.

```
result = df.transform(func = lambda x : x * 5)
result
```

The above code will give the following output.

	A	B	C
0	40	NaN	50.0
1	35	110.0	30.0
2	75	160.0	NaN
3	60	45.0	40.0
4	75	35.0	70.0

The below code will calculate the square root of value of all the columns A, B and C of data frame df.

```
result = df.transform(func = ["sqrt"])
result
```

The above code will give the following output.

	A	B	C
	sqrt	sqrt	sqrt
0	2.828427	NaN	3.162278
1	2.645751	4.690416	2.449490
2	3.872983	5.656854	NaN
3	3.464102	3.000000	2.828427
4	3.872983	2.645751	3.741657

The below code will calculate the exponential of value of all the columns A, B and C of data frame df.

```
result = df.transform(func = ["exp"])
result
```

The above code will give the following output.

	A	B	C
	exp	exp	exp
0	2.980958e+03	NaN	2.202647e+04
1	1.096633e+03	3.584913e+09	4.034288e+02
2	3.269017e+06	7.896296e+13	NaN
3	1.627548e+05	8.103084e+03	2.980958e+03
4	3.269017e+06	1.096633e+03	1.202604e+06

The below code will calculate both square root and exponential together of value of all the columns A, B and C of data frame df.

```
result = df.transform(func = ["sqrt","exp"])
result
```

The above code will give the following output.

	A		B		C	
	sqrt	exp	sqrt	exp	sqrt	exp

0	2.828427	2.980958e+03	NaN	NaN	3.162278	2.202647e+04
1	2.645751	1.096633e+03	4.690416	3.584913e+09	2.449490	4.034288e+02
2	3.872983	3.269017e+06	5.656854	7.896296e+13	NaN	NaN
3	3.464102	1.627548e+05	3.000000	8.103084e+03	2.828427	2.980958e+03
4	3.872983	3.269017e+06	2.645751	1.096633e+03	3.741657	1.202604e+06

The below code will create and display data frame df.

```
#importing pandas library
import pandas as pd

# Creating the DataFrame
df = pd.DataFrame({
"Team":["MI","CSK","RR","MI","KKR","KKR","MI","CSK","KKR",
"RR"],
"Score":[210,150,215,180,185,205,230,190,160,185]})

# Display dataframe
df
```

The above code will give the following output.

	Team	Score
0	MI	210
1	CSK	150
2	RR	215
3	MI	180
4	KKR	185
5	KKR	205
6	MI	230
7	CSK	190
8	KKR	160
9	RR	185

The below code will perform the transformation operation using groupby function.

Here, groupby function will apply on “Team” and calculate the “sum” of “Score” using transform function.

```
df.groupby("Team")["Score"].transform("sum")
```

The above code will give the following output.

Here, the sum of “Score” of each “Team” will be display.

```
0    620
1    340
2    400
3    620
4    550
5    550
6    620
7    340
8    550
9    400
Name: Score, dtype: int64
```

6.5 PIVOT TABLE

Pivot table is a statistical table which summarizes a substantial table like a big dataset. The summary in a pivot tables may include sum, min, max, mean, median or other statistical terms. The pivot() function provides general purpose pivoting with various data type such as string, numeric, etc. The pivot_table() function is used to create pivot table with aggregation of numeric data using data frame of pandas library of Python. The syntax of this function is as follow:

Syntax:

```
DataFrame.pivot(data, index=None, columns=None, values=None, aggfunc)
```

Here,

- **data:** dataframe object
- **index:** a column which has the same length as data. Keys to group by on the pivot table index.
- **columns:** a column which has the same length as data. Keys to group by on the pivot table column.
- **values:** column or list of columns to aggregate
- **aggfunc:** function to use for aggregation

The below example will create a pivot table using pivot_table() function of pandas library of Python.

Example :

The below code will create and display data frame df.

```
# Importing pandas library
import pandas as pd

# Dataframe creation
df = pd.DataFrame({
```

```
"Product":["Mango","Corn","Orange","Cabbage","Mango","Corn","Watermelon","App
le","Pumkin","Mango"],
"Category":["Fruit","Vegetable","Fruit","Vegetable","Fruit","Vegetable","Fruit","Fruit
","Vegetable","Fruit"],
"Qty":[12, 5, 10, 2, 10, 3, 5, 8, 2, 10],
"Price":[350, 80, 320, 50, 200, 50, 280, 380, 60, 400]})


# Display dataframe
df
```

The above code will give the following output.

	Product	Category	Qty	Price
0	Mango	Fruit	12	350
1	Corn	Vegetable	5	80
2	Orange	Fruit	10	320
3	Cabbage	Vegetable	2	50
4	Mango	Fruit	10	200
5	Corn	Vegetable	3	50
6	Watermelon	Fruit	5	280
7	Apple	Fruit	8	380
8	Pumkin	Vegetable	2	60
9	Mango	Fruit	10	400

Now, we will perform some examples of pivot table.

Example: To create a pivot table of total sales of each product.

```
# Pivot table of total sales of each product
tot_sales = df.pivot_table(index=["Product"], values=["Price"],aggfunc="sum")

# Display pivot table of total sales
print(tot_sales)
```

Here, we set the index as a “Product” and “sum” as an aggregate function to calculate the total sales of each product. The sum function will do the summation of each products. The above code will give the following output.

```
          Price
Product
Apple         380
Cabbage        50
Corn          130
Mango          950
Orange         320
Pumkin          60
Watermelon     280
```

Example: To create a pivot table of total sales of each category.

```
# Pivot table of total sales of each category
tot_sales = df.pivot_table(index=["Category"], values=["Price"], aggfunc="sum")


# Display pivot table of total sales
print(tot_sales)
```

Here, we set the index as a “Category” and “sum” as an aggregate function to calculate the total sales of each category. The above code will give the following output.

```
          Price
Category
Fruit        1930
Vegetable     240
```

Example: To create a pivot table of total sales of each product.

```
# Pivot table of total sales of both product and category
tot_sales = df.pivot_table(index=["Category","Product"], values=["Price"],
aggfunc="sum")

# Display pivot table of total sales
print(tot_sales)
```

Here, we set the index as a both “Category” and “Product” and “sum” as an aggregate function to calculate the total sales of each product. The above code will give the following output.

```
                  Price
Category  Product
Fruit     Apple       380
          Mango       950
          Orange      320
          Watermelon  280
Vegetable Cabbage     50
          Corn        130
          Pumkin      60
```

Example: To create a pivot table to find the minimum, maximum, mean and median of price of each category wise.

```
# Pivot table of min, max, mean and media of sales
tot_sales = df.pivot_table(index=["Category"], values=["Price"],
aggfunc={"min","max","mean","median"})

# Display pivot table of total sales
print(tot_sales)
```

Here, we set the index as a “Category” and “min”, “max”, “mean” and “median” as an aggregate function to calculate the minimum, maximum, mean and median of price of each category wise. The above code will give the following output.

```
          Price
            max        mean median    min
Category
Fruit      400.0  321.666667  335.0  200.0
Vegetable   80.0   60.000000   55.0   50.0
```

Example: To create a pivot table of total product count of each category.

```
# Pivot table of minimum, maximum and average sales
tot_sales = df.pivot_table(index=["Category", "Product"],
values=["Price"],aggfunc=["count"])

# Display pivot table of total sales
print(tot_sales)
```

Here, we set the index as a both “Category” and “Product” and “count” as an aggregate function to count total product of each category. The above code will give the following output.

		count
		Price
Category	**Product**	
Fruit	Apple	1
	Mango	3
	Orange	1
	Watermelon	1
Vegetable	Cabbage	1
	Corn	2
	Pumkin	1

6.6 CROSS TABULATIONS

The cross-tabulation method is used to calculate the simple cross-tabulation of two or more factors. The pandas provide crosstab() function to build a cross-tabulation table which shows the frequency with which certain groups of data appear. The syntax of crosstab() function in pandas is as follow:

Syntax:

```
pd.crosstab(index, columns, values=None, rownames=None, colnames=None, aggfunc=None, margins=False, margins_name='All', normalize=False, dropna=True)
```

Here,

- **index:** array-like, values to group by in the rows.
- **columns:** array-like, values to group by in the columns.
- **values:** array-like, optional, array of values to aggregate according to the factors.
- **rownames:** sequence, must match number of row arrays passed, default is None
- **colnames:** sequence, must match number of column arrays passed if passed, default is None
- **aggfuncs:** function, optional, if no values array is passed, its computers a frequency table.
- **margins:** boolean, add row / column margins (i.e. subtotals), default is False.
- **margins_name:** string, name of the row / column that will contain the subtotals if margins is True, default is “All”.
- **normalize:** boolean, {„all“, “index“, “columns“}, or {0,1}, normalize by dividing all values by the sum of values, default is False.
- **dropna:** boolean, do not include columns whose all entries are NaN, default is True.

The below example will create a cross-table using crosstab() function of pandas library of Python.

Example :

The below code will create and display data frame df.

```
# Importing pandas library
import pandas as pd

# Dataframe creation
df = pd.DataFrame({
"Name":["Rahul","Jyoti","Rupal","Rahul","Jyoti","Rupal",
"Rahul","Jyoti","Rupal","Rahul","Jyoti","Rupal","Rahul",
"Jyoti","Rupal","Rahul","Jyoti","Rupal"],
"Examination":["SEM-I","SEM-I","SEM-I","SEM-I","SEM-I",
"SEM-I","SEM-I","SEM-I","SEM-I","SEM-II","SEM-II",
"SEM-II","SEM-II","SEM-II","SEM-II","SEM-II","SEM-II",
"SEM-II"],
"Subject":["Physics","Physics","Physics","Chemistry",
"Chemistry","Chemistry","Biology","Biology","Biology",
"Physics","Physics","Physics","Chemistry","Chemistry",
"Chemistry","Biology","Biology","Biology"],
"Result":["PASS","PASS","FAIL","PASS","FAIL","PASS","FAIL",
"PASS","FAIL","PASS","PASS","PASS","FAIL","PASS","PASS",
"PASS","PASS","FAIL"]})

# Display dataframe
df
```

The above code will give the following output.

	Name	Examination	Subject	Result
0	Rahul	SEM-I	Physics	PASS
1	Jyoti	SEM-I	Physics	PASS
2	Rupal	SEM-I	Physics	FAIL
3	Rahul	SEM-I	Chemistry	PASS
4	Jyoti	SEM-I	Chemistry	FAIL
5	Rupal	SEM-I	Chemistry	PASS
6	Rahul	SEM-I	Biology	FAIL

7	Jyoti	SEM-I	Biology	PASS
8	Rupal	SEM-I	Biology	FAIL
9	Rahul	SEM-II	Physics	PASS
10	Jyoti	SEM-II	Physics	PASS
11	Rupal	SEM-II	Physics	PASS
12	Rahul	SEM-II	Chemistry	FAIL
13	Jyoti	SEM-II	Chemistry	PASS
14	Rupal	SEM-II	Chemistry	PASS
15	Rahul	SEM-II	Biology	PASS
16	Jyoti	SEM-II	Biology	PASS
17	Rupal	SEM-II	Biology	FAIL

Two-way cross table:

There are two columns is used to create a cross table is called two-way cross table.

Here, we will create a cross table of two columns “Subject” and “Result” as follow:

```
# Two-way cross table creation
pd.crosstab(df.Subject, df.Result, margins=True)
```

Here, we set margin=True to display the row wise sum and column wise sum of the cross table. The above code will give the following output.

Result	**FAIL**	**PASS**	**All**
Subject			
Biology	**3**	**3**	**6**
Chemistry	**2**	**4**	**6**
Physics	**1**	**5**	**6**
All	**6**	**12**	**18**

Three-way cross table:

There are three columns is used to create a cross table is called three-way cross table.

Here, we will create a cross table of three columns “Subject”, “Examination” and “Result” as follow:

```
# Three-way cross table creation
pd.crosstab([df.Subject, df.Examination], df.Result, margins=True)
```

Here, we set margin=True to display the row wise sum and column wise sum of the cross table. The above code will give the following output.

	Result	FAIL	PASS	All
Subject	**Examination**			
Biology	**SEM-I**	**2**	**1**	**3**
	SEM-II	**1**	**2**	**3**
Chemistry	**SEM-I**	**1**	**2**	**3**
	SEM-II	**1**	**2**	**3**
Physics	**SEM-I**	**1**	**2**	**3**
	SEM-II	**0**	**3**	**3**
All		**6**	**12**	**18**

6.7 DATE AND TIME DATA TYPE

Python does not have date and time data types, but it has a module named “datetime” can be imported to deal with the date and time. This is inbuilt module available in the Python. This module consists different classes to work with date and time. These classes provide different functions to work with dates, times and time intervals.

There are main six classes in datetime module:

Data Type	Description
Date	It is a date type object. It manipulates only date (i.e. day, month and year).
Time	It is a time object class. It manipulates only time of the any specific day (i.e. hour, minute, second, microsecond).
Datetime	It manipulates the combination of both time and date (i.e. day, month, year, hour, second, microsecond).
timedelta	It manipulates the duration expressing the different between two dates, times or datetime values in milliseconds.
Tzinfo	It is an abstract base class which provides time zone information.

timezone	It is a class that implements tzinfo abstract base class as a fixed offset from the UTC.

There are different format codes is used to formatting the data and time.

The format codes are as follows:

Directive	Description	Example
%a	Day of Week, short version	Fri
%A	Day of Week, full version	Friday
%w	Day of Week as a number from 0 to 6, Here 0 is Sunday	4
%d	Day of Month from 01 to 31	25
%b	Name of Month, short version	Mar
%B	Name of Month, full version	March
%m	Month as a number from 01 to 12	11
%y	Year, short version (in two digit)	21
%Y	Year, full version (in four digit)	2021
%H	Hour from 00 to 23 (in 24 hr format)	16
%I	Hour from 00 to 12 (in 12 hr format)	07
%p	AM or PM	AM
%M	Minute from 00 to 59	35
%S	Second from 00 to 59	45
%f	Microsecond from 000000 to 999999	234567
%z	UTC offset	+0100
%Z	Timezone	CST
%j	Day number of year from 001 to 365	325
%U	Week number of year, Sunday as the first day of week, from 00 to 53	40
%W	Week number of year, Monday as the first day of week, from 00 to 53	40
%c	Local version of date and time	Tue Mar 30 13:25:30 2021
%x	Local version of date (MM/DD/YY)	11/24/2021
%X	Local version of time (HH:MM:SS)	18:25:40

To get more insight and work with datetime modules, let‟s take few examples.

Example: To get current data and time.

```
# To get the current date and time
import datetime as dt
d = dt.datetime.now()
print(d)
```

The above code will give the following output.

```
2021-05-15 00:53:08.925167
```

Here, the now() function is used to display the current local date and time.

Example: To get the current date.

```
# To get the current date only
import datetime as dt
d = dt.date.today()
print(d)
```

The above code will give the following output.

```
2021-05-15
```

Here, the today() function is used to get the current local date.

Example: To get the todays date.

```
# To get the today‟s date, month and year separately
import datetime as dt
today = dt.date.today()
print(today)
print("Day :",today.day)
print("Month :",today.month)
print("Year :",today.year)
```

The above code will give the following output.

```
2021-05-15
Day : 15
Month : 5
Year : 2021
```

Here, the today() function is used to get the current local date and display day, month and year separately.

Example: To represent a date using date object.

```
# To represent a date using date object
import datetime as dt
d = dt.date(2021, 1, 26)
print(d)
```

The above code will give the following output.

```
2021-01-26
```

Here, the date is passed as an argument.

Example: To represent a date using timestamp.

```
# To get date from a timestamp
import datetime as dt
ts = dt.date.fromtimestamp(987654321)
print(ts)
```

The above code will give the following output.

```
2001-04-19
```

Here, the fromtimestamp() function is used converts seconds into equivalent date.

Example: To represent a time using time object.

```
# To represent a time using time object
import datetime as dt

# time(hour=0, minute=0, second=0)
t = dt.time()
print("Time :",t)

# time(hour, minute, second)
t = dt.time(10, 40, 55)
print("Time :",t)

# time(hour, minute, second)
t = dt.time(hour=10, minute=40, second=55)
print("Time :",t)

# time(hour, minute, second, microsecond)
t = dt.time(10, 40, 55, 123456)
print("Time :",t)

# time(hour, minute, second, microsecond)
t = dt.time(10, 40, 55, 123456)
```

```
print("Hour :",t.hour)
print("Minute :",t.minute)
print("Second :",t.second)
print("Microsecond :",t.microsecond)
```

The above code will give the following output.

```
Time : 00:00:00
Time : 10:40:55
Time : 10:40:55
Time : 10:40:55.123456
Hour : 10
Minute : 40
Second : 55
Microsecond : 123456
```

Here, the time() function with different arguments is used to get the time in different formats.

Example: To represent a datetime object.

```
# To represent a datetime using datetime object
import datetime as dt

# datetime(year, month, day)
dtformat = dt.datetime(2021, 5, 15)
print(dtformat)

# datetime(year,month,day,hour,minute,second,microsecond)
dtformat = dt.datetime(2021, 5, 15, 16, 35, 25, 234561)
print(dtformat)
```

The above code will give the following output.

```
2021-05-15 00:00:00
2021-05-15 16:35:25.234561
```

Here, the datetime() function is used to display the dates in different formats.

Example: To represent a datetime object using different format.

```
# To represent a datetime using datetime object
import datetime as dt

dtformat = dt.datetime(2021, 5, 15, 16, 35, 25, 234561)
print("Year : ",dtformat.year)
print("Month : ",dtformat.month)
```

```
print("Day : ",dtformat.day)
print("Hour : ",dtformat.hour)
print("Minute : ",dtformat.minute)
print("Timestamp : ",dtformat.timestamp())
```

The above code will give the following output.

```
Year :  2021
Month :  5
Day :  15
Hour :  16
Minute :  35
Timestamp :  1621076725.234561
```

Here, the datetime() function is used to display the dates separately in year, month, day, hours, minutes, seconds etc.

Example: To find the difference between to dates and times.

```
# Different between two dates and times
import datetime as dt

# date(year, month, day)
t1 = dt.date(year=2021, month=5, day=15)
t2 = dt.date(year=2020, month=7, day=25)
t3 = t1 - t2
print("Date Difference :",t3)
print("Type of t3 :",type(t3))

# date(year, month, day, hour, minute, second)
t1 = dt.datetime(year=2020, month=1, day=15, hour=8, minute=25, second=45)
t2 = dt.datetime(year=2021, month=4, day=25, hour=10, minute=30, second=50)
t3 = t1 - t2
print("Date Difference :",t3)
print("Type of t3 :",type(t3))
```

The above code will give the following output.

```
Date Difference : 294 days, 0:00:00
Type of t3 : <class 'datetime.timedelta'>
Date Difference : -467 days, 21:54:55
Type of t3 : <class 'datetime.timedelta'>
```

Here, the datetime() function is used to display the dates and perform the subtraction operation between two different dates.

Example: To use of timedelta object.

```
# Different between two timedelta objects
import datetime as dt

t1 = dt.timedelta(weeks=6, days=5, hours=9, minutes=45, seconds=10)
t2 = dt.timedelta(weeks=4, days=3, hours=5, minutes=25, seconds=35)
t3 = t1 - t2
print("Time Delta Difference : ",t3)

t1 = dt.timedelta(weeks=3, hours=10, minutes=45)
t2 = dt.timedelta(days=4, minutes=15, seconds=35)
t3 = t1 - t2
print("Time Delta Difference : ",t3)
```

The above code will give the following output.

```
Time Delta Difference :  16 days, 4:19:35
Time Delta Difference :  17 days, 10:29:25
```

Here, the timedelta() function is used to display the dates and perform the subtraction operation between two different dates.

Example: To represent the time in total seconds.

```
# Time duration in seconds
import datetime as dt
t = dt.timedelta(hours=9, minutes=35, seconds=15)
print("Time in Second :",t.total_seconds())
```

The above code will give the following output.

```
Time in Second : 34515.0
```

Here, the total_seconds() function is used to convert given times into second format.

Example: To use strftime() function for formatting.

```
# Use of strftime() function for date formatting
import datetime as dt

# current date and time
now = dt.datetime.now()
print("Current Date and Time :",now)
```

```
# time in HH:MM:SS format
f1 = now.strftime("%H:%M:%S")
print("Time format is :",f1)

# date and time format DD/MM/YY, HH:MM:SS format
f2 = now.strftime("%d/%m/%Y, %H:%M:%S")
print("Date :",f2)

# Date and time format MM/DD/YY, HH:MM:SS format
f3 = now.strftime("%m/%d/%Y, %H:%M:%S")
print("Date :",f3)
```

The above code will give the following output.

```
Current Date and Time : 2021-05-15 17:16:52.794301
Time format is : 17:16:52
Date : 15/05/2021, 17:16:52
Date : 05/15/2021, 17:16:52
```

Here, the strftime() function is used to display the dates in different format.

Example: To use strptime() function for formatting.

```
# Use of strptime() function for date formatting
import datetime as dt

# date in string format
dt_string = "15 May, 2021"
print("Date in string :",dt_string)

# date in object format
dt_object = dt.datetime.strptime(dt_string, "%d %B, %Y")
print("Date in object :",dt_object)
```

The above code will give the following output.

```
Date in string : 15 May, 2021
Date in object : 2021-05-15 00:00:00
```

Here, the strptime() function is used to string format date into object format date.

Example: To use different timezone.

```
# Use of time zone
import datetime as dt
import pytz
```

```
# Local time zone
local = dt.datetime.now()
print("Local Time Zone:",local.strftime("%m/%d/%y, %H:%M:%S"))

# London time zone
tz_London = pytz.timezone('Europe/London')
dt_London = dt.datetime.now(tz_London)
print("London Time Zone:",dt_London.strftime("%m/%d/%y, %H:%M:%S"))

# Newyork time zone
tz_NY = pytz.timezone('America/New_York')
dt_NY = dt.datetime.now(tz_NY)
print("New York Time Zone:",dt_NY.strftime("%m/%d/%y, %H:%M:%S"))
```

The above code will give the following output.

```
Local Time Zone: 05/15/21, 17:35:15
London Time Zone: 05/15/21, 13:05:15
New York Time Zone: 05/15/21, 08:05:15
```

Here, the pytz is used to set different timezone and strftime() function is used to display the dates in different format.

6.8 SUMMARY

The students will learn many things related to data aggregation and group wise operations in this module and they will be able to perform the various data science related operation using Python.

- Ability to perform the data aggregation using various functions such as min, max, sum, average, count etc.
- Ability to perform group wise operation on specific group such as splitting, applying and combining to calculate the mean, median, standard deviation group wise.
- Ability to do the statistical summary table using pivoting.
- Ability to work with cross tabulation using both two-way and three-way cross table.
- Ability to perform various operation on date and time data types.

REFERENCES

Books

5. Davy Cielen, Arno D. B. Meysman, Mohamed Ali : Introducing Data Science, Manning Publications Co.
6. Stephen Klosterman (2019) : Data Science Projects with Python, Packt Publishing
7. Jake VanderPlas (2017) : Python Data Science Handbook: Essential Tools for Working with Data, O‟Reilly

8. Wes McKinnery and Pandas Development Team (2021) : pandas : powerful Python data analysis toolkit, Release 1.2.3, 2021

Web References

6. https://www.geeksforgeeks.org
7. https://www.tutorialspoint.com
8. https://www.w3schools.com
9. https://pandas.pydata.org
10. https://pbpython.com

QUESTIONS

Short Answer:

6. What is data aggregation?
7. What is data splitting?
8. What is transformation?
9. What is pivot table?
10. What is cross tabulation?

Long Answer:

7. Explain data aggregation with different functions.
8. Explain groupby function with example.
9. Explain transform function with syntax and example.
10. Explain pivot table with example.
11. Explain cross tabulation with example.
12. Explain date and time data type with different format code.

PRACTICALS

1. Create a data frame and perform aggregation functions.
2. Create data frames and perform groupby function with different arguments.
3. Create data frames and perform transform function.
4. Create data frame and perform pivot table with different arguments.
5. Create data frame and prepare two-way and three-way cross table.
6. Perform various operation using date and time data types.

INTRODUCTION TO DATA SCIENCE

UNIT VII: DATA MODELING

STRUCTURE

7.0 OBJECTIVES

The main goal of this module is to help students learn, understand and practice the data science approaches, which include the study of latest data science tools with latest programming languages. The main objectives of this module are data modeling which includes the basics of generative modeling and predictive modeling techniques and data visualization which include different types of charts and plots like histogram, scatter plot, time series plot etc.

7.1 INTRODUCTION

Data science become a buzzword that everyone talks about the data science. Data science is an interdisciplinary field that combines different domain expertise, computer programming skills, mathematics and statistical knowledge to find or extract the meaningful or unknown patterns from unstructured and structure dataset.

Data science is useful for extraction, preparation, analysis and visualization of various information. Various scientific methods can be applied to get insight in data.

Data science is all about using data to solve problems. Data has become the fuel of industries. It is most demandable field of 21st century. Every industry require data to functioning, searching, marketing, growing, expanding their business.

The application of areas of data science are health care, fraud detection, disease predicting, real time shipping routes, speech recognition, targeting advertising, gaming and many more.

7.2 INTRODUCTION TO GENERATIVE MODELING

Generative models are the family of machine learning models that are used to describe how data is generated. There are mainly two different types of problems to work with machine learning or deep learning algorithms such as supervised learning and unsupervised learning.

In supervised learning problem, we have two variables such as independent variables (x) and the target variable (y). The examples of supervised learning are classification, regression, object detection etc.

In unsupervised learning problem, we have only independent variables (x). there are no target variable or label. It aims is to find some underlying patterns from the dataset. The examples of unsupervised learning are clustering, dimensionality reduction etc.

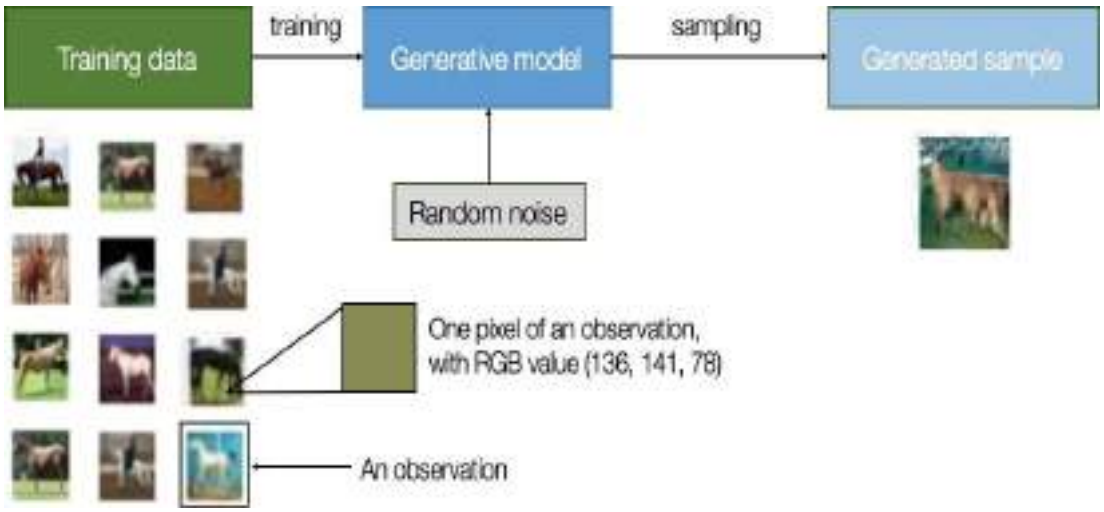
The generative model is an unsupervised learning problem in machine learning. It automatically discovers and learning the rules, regularities or patterns from the large input training dataset. This model learns to create a data that is look like as given. A generative model can be broadly defined as follows:

A generative model describes how a dataset is generated, in terms of a probabilistic model. By sampling from this model, we are able to generate new data.

Generative model can generate new data instances. If we have a dataset containing images of any animal and we may develop a model which can generate a new image of same animal that is never existing but still it looks like as real animal. This model has learned the general

rules that govern the appearance of a specific animal. A generative modelling is used to solve this kind of problems. A generative model process are as follows:

We require a dataset which consists many instants of the entity which we want to generate. This dataset is known as the training data and each data points is called as an observation. The following diagram represent a horse animal dataset.



Generative Model Process (Source: https://www.oreilly.com)

The existing dataset of horse images is used as a training dataset. Based on the training given to the existing dataset it built a generative model to create new images which look like as a real image.

7.3 INTRODUCTION TO PREDICTIVE MODELING

Predictive modeling is a mathematical approach to build models based on existing dataset, which will help to finding the future value or trend of a variable. The variety of statistical techniques including data mining and machine learning are used to estimate or predict the future outcomes.

The predictive modelling is used for every area such as

- Weather forecasting
- Price forecasting
- Demand forecasting
- Sales forecasting
- Customer targeting
- Financial modeling
- Risk assessment
- Market analysis

7.3.1 Types of Models

There are different predictive analytics models are developed for specific applications as follows:

- Classification Model
- Clustering Model
- Forecasting Model
- Time Series Model
- Outlier Model

- **Classification Model**

  This is a simplest and most commonly used predictive analytics model. It works on categorical information based on historical data.

  This model is used or apply in many industrial applications because it can easily retrain with new data as per the needs.

- **Clustering Model**

  This model is use to take the data and divide it into different nested smart groups based on some common attributes. It helps to divide or grouping things or data with shared characteristic or behaviors and take strategic decisions for each group. For example, the customers can be divided based on common attributes like purchasing methods, purchasing power, etc. for targeted marketing campaign to the customers.

- **Forecast Model**

  This is a very popular and most widely use model. It works with the metric value prediction, by estimating the value of new data based on learnings from historical data. It is also used to generate the numerical values and update where none or missing value found. This model can be applied wherever historical numerical data is available. It considers multiple input parameters.

  This model is used in many different business and industries. For example, the company‟s customer care department can predict how many supports calls they will receive per day.

- **Time Series Model**

  This model is focusses on data where time is an input parameter. This model is applied by using different data points which is taken from the previous year‟s data to develop a numerical metric that will used to predict the trends within a specified period of time.

  This model is used in many industries which want to see how a particular variable change over a time period. It also takes care about extraneous factors that might be affect the variable such as seasons or seasonal variable. For example, the shopping mall owners want to know the how many customers may visit the mall in week or month.

- **Outliers Model**

This model is work with anomalous data entries in a dataset. It works by finding unusual data, either in isolation or in relation with different categories and numbers. It is more useful in industries were identifying anomalies can save organization corers of rupees such as finance and retail. It is more effective in fraud detection because it can find the anomalies. Since an incidence of fraud is a deviation from the norm, this model is more likely to predict it before it occurs. For example, when identifying a fraud transaction, this model can assess the amount of money lost, purchase history, time, location etc.

7.3.2 Predictive Algorithms

The predictive analytics algorithms can be separated into two things: machine learning and deep learning. These both are subsets of artificial intelligence (AI).

Machine Learning: It deals structural data such as table or spreadsheets. It has both linear and non-linear algorithms. Linear algorithms are quickly train, while non-linear are better optimized for the problems they are to face.

Deep Learning: It is a subset of machine learning. It deals with unstructured data such as images, social media posts, text, audio and videos.

There are several algorithms can be used for machine learning predictive modelling. The most common algorithms are:

- Random Forest
- Generalized Linear Model (GLM) for Two Values
- Gradient Booster Model (GBM)
- K-Means
- Prophet

7.4 CHARTS

Charts is the representation of data in a graphical format. It helps to summarizing and presenting a large amount of data in a simple and easy to understandable formats. By placing the data in a visual context, we can easily detect the patterns, trends and correlations among them.

Python provides various easy to use multiple graphics libraries for data visualization with different features. These libraries are work with both small and large datasets.

Python has multiple graphics libraries with different features. Some of the most popular and commonly used Python data visualization libraries are :

- Matplotlib
- Pandas
- Seaborn
- ggplot
- Plotly

Matplotlib is a most popular, amazing and multi-platform data visualization library available in Python. Matplotlib consists a wide variety of plots like histogram, scatter plot, line or time series plot, bar chart etc.

7.4.1 Histogram

Histogram is a graphical representation of the distribution of numerical data. It contains a rectangular area to display the statistical information which is proportional to the frequency of a variable. It is an estimate of probability distribution of a continuous variable.

In a histogram, the data are binned and the count for each bin is represent. The number of bins is selected so that it is comparable to the typical number of samples in a bin. The bins are specified as consecutive and non-overlapping intervals of a variable. The numbers of bin can be customized also.

There are three basic steps to construct a histogram:

- Bin the range of values
- Distribute the range of values in a series of intervals
- Count the numbers of value into each interval

The ***hist()*** function is used to plot a histogram. It computes and draw the histogram of x values. There is some parameter to construct the histogram are as follows:

- ***bins:*** numbers of bin in the plot, optional
- ***range:*** lower and upper range of the bins.
- ***density***: density or count to populate the plot
- ***histtype:*** types of histogram plot such as bar, step and stepfilled, default is bar
- **align:** control to plot the histogram such as left, mid and right
- ***rwidth:*** relative width of a bar as a fraction of bin width
- ***color:*** it is a color spec or sequence of color specs
- ***orientation:*** horizontal or vertical representation, default is vertical

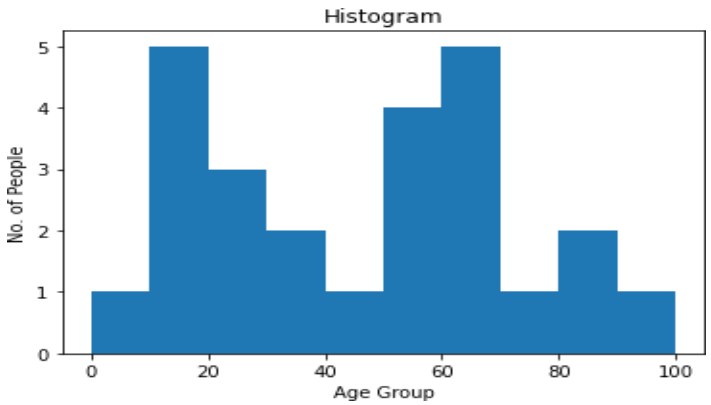
Example: Here we take an example of age of peoples.

The x-axis represents age group or bins and y-axis represents age.

```
# Importing library
import matplotlib.pyplot as plt

# Data values
age = [22, 55, 62, 45, 21, 22, 4, 12, 14, 64, 58, 68, 95, 85, 55, 38, 18, 37, 65, 59, 11, 15, 80, 75, 65]
bins = [0,10,20,30,40,50,60,70,80,90,100]



# Plotting the histogram with title and label
plt.hist(age, bins)
plt.xlabel("Age Group")
```

```
plt.ylabel("No. of People")
plt.title("Histogram")

# Show plot
plt.show()
```

The above code will plot histogram as follow:

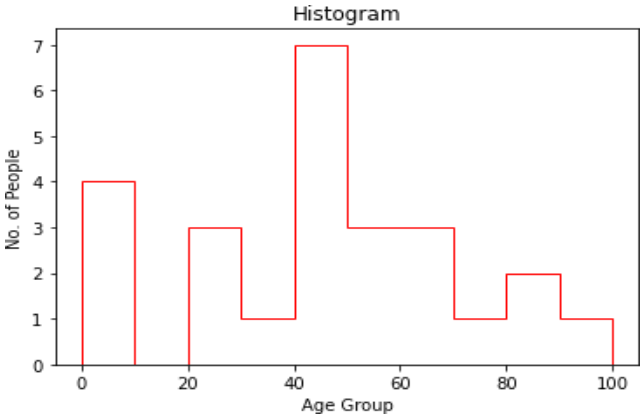


Here, the age is divided into different age group.

We can also set the type and color of histogram and using ***histtype*** and ***color*** as an argument.

```
import matplotlib.pyplot as plt

age = [22, 55, 62, 45, 21, 22, 34, 42, 42, 4, 2, 102, 95, 85, 55, 110, 120, 7, 65, 55,
111, 115, 80, 75, 65, 4, 44, 43, 42, 48]
bins = [0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100]

plt.hist(age, bins, histtype='step', rwidth=0.8, color="red")
plt.xlabel("Age Group")
plt.ylabel("No. of People")
plt.title("Histogram")
plt.show()
```

The above code will plot histogram as follow:



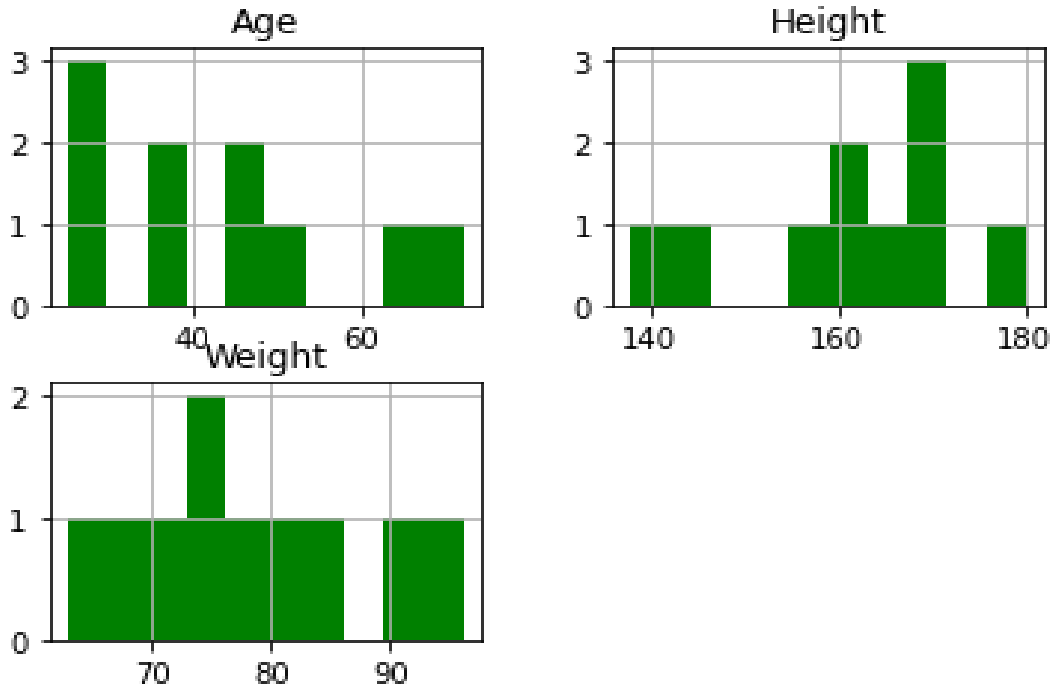
Here, the type of histogram is set to ***step*** and color is set to ***red***.

We can also create the multiple histogram of different columns as follows:

Here, we create a data frame with three different columns such as **"Age", "Height"** and **"Weight".**

```
import pandas as pd

df = pd.DataFrame({
    "Age":[25,38,45,29,65,52,46,72,28,35],
    "Height": [145,138,160,180,165,170,158,162,171,168],
    "Weight": [75,90,85,72,68,76,82,96,63,79]})

hist = df.hist(bins=10)
```

The above code will plot histogram as follow:



7.4.2 Scatter Plot

Scatter plot is diagram where each value in the dataset represent by a dot. It is set of dotted points to represent the individual data on both horizontal and vertical axis to reveal the distribution trends of data.

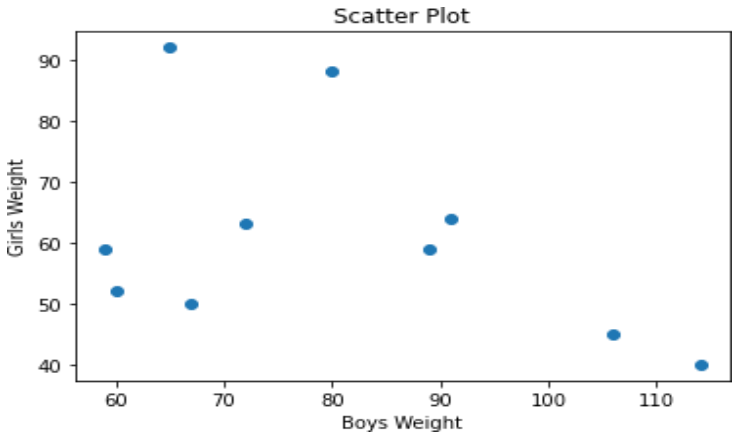
This plot is mostly used for large dataset to highlight the similarities in the dataset. It also shows the outliers and distribution of data.

The **scatter()** function is used to draw the scatter plot. This function plots one dot for each observation. It requires two different arrays of same length for both the x-axis and y-axis. we can also set the scatter plot title and labels on both the axis.

Example: Here we take an example of boys weight and girls weight. The x-axis represents **"Boys_Weight"** and y-axis represents **"Girls_Weight"**.

```
# Importing library
import matplotlib.pyplot as plt

# Data values
Boys_Weight = [67, 89, 72, 114, 65, 80, 91, 106, 60, 59]
Girls_Weight = [50, 59, 63, 40, 92, 88, 64, 45, 52, 59]

# Plotting scatter plot with title and label
plt.scatter(Boys_Weight,  Girls_Weight)
plt.title("Scatter Plot")
plt.xlabel("Boys Weight")
plt.ylabel("Girls Weight")

# Show plot
plt.show()
```

The above code will create scatter plot as follow:

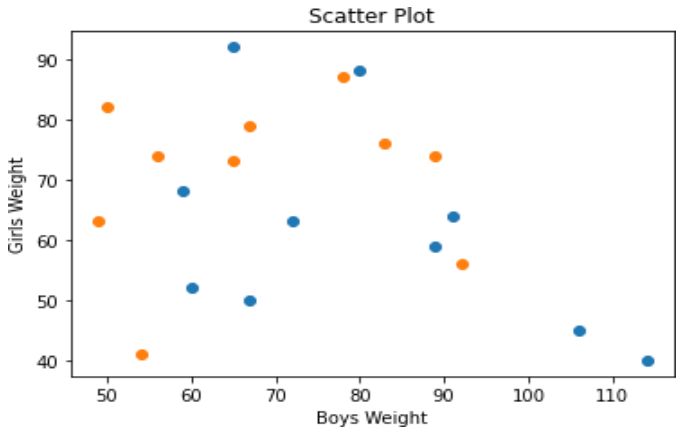


In above plot, we can see the relationship between boys and girls weight.

We can also compare the boys weight and girls weight of one class with another class.

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

Boys_Weight = [67, 89, 72, 114, 65, 80, 91, 106, 60, 59]
Girls_Weight = [50, 59, 63, 40, 92, 88, 64, 45, 52, 68]
plt.scatter(Boys_Weight, Girls_Weight)

Boys_Weight = [54, 67, 92, 56, 83, 65, 89, 78, 50, 49]
Girls_Weight = [41, 79, 56, 74, 76, 73, 74, 87, 82, 63]
plt.scatter(Boys_Weight, Girls_Weight)

plt.title("Scatter Plot")
plt.xlabel("Boys Weight")
plt.ylabel("Girls Weight")

plt.show()
```

The above code will create scatter plot as follow:



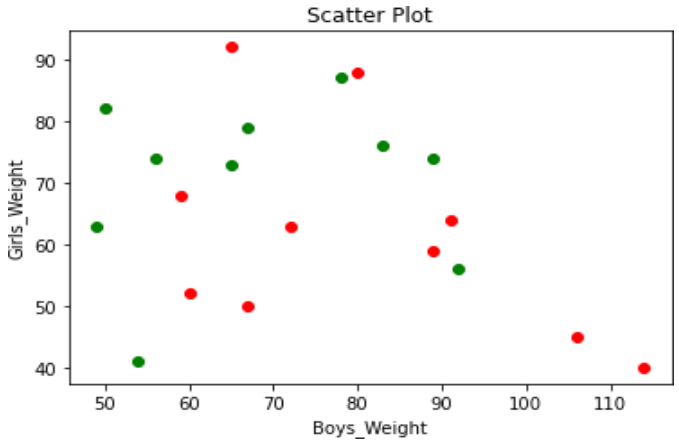
In above plot, we can see the relationship between boys and girls weight of one class with another class by using different colors.

We can also set or change the color of both the classes of data using ***color*** as an argument. Here we set the ***red*** color for first class students and ***green*** color for second class students.

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

Boys_Weight = [67, 89, 72, 114, 65, 80, 91, 106, 60, 59]
```

```
Girls_Weight = [50, 59, 63, 40, 92, 88, 64, 45, 52, 68]
plt.scatter(Boys_Weight, Girls_Weight, color="red")

Boys_Weight = [54, 67, 92, 56, 83, 65, 89, 78, 50, 49]
Girls_Weight = [41, 79, 56, 74, 76, 73, 74, 87, 82, 63]
plt.scatter(Boys_Weight, Girls_Weight, color="Green")

plt.title("Scatter Plot")
plt.xlabel("Boys_Weight")
plt.ylabel("Girls_Weight")

plt.show()
```

The above code will create scatter plot as follow:

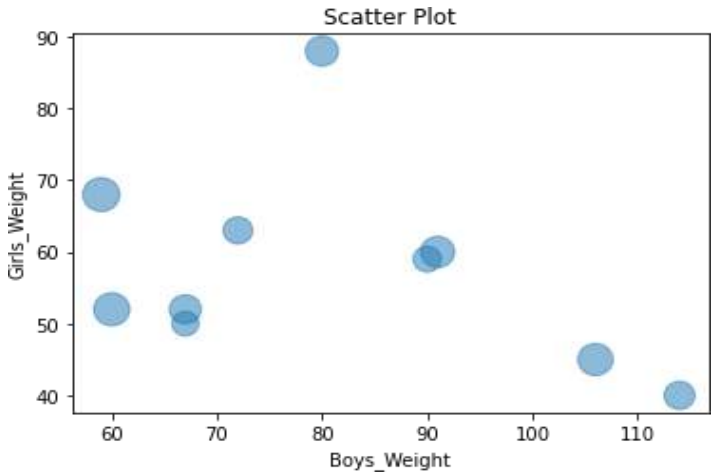


In above plot, we can see the relationship between both classes. Here, red color is used for first class students and green color is used for second class students.

We can also set or change the size of dots using ***s*** as an argument in scatter plot.

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

Boys_Weight = [67, 90, 72, 114, 67, 80, 91, 106, 60, 59]
Girls_Weight = [50, 59, 63, 40, 52, 88, 60, 45, 52, 68]
size = [200, 220, 240, 260, 280, 300, 320, 340, 360, 380]
plt.scatter(Boys_Weight, Girls_Weight, s=size, alpha=0.5)

plt.title("Scatter Plot")
plt.xlabel("Boys_Weight")
plt.ylabel("Girls_Weight")

plt.show()
```

The above code will create scatter plot as follow:

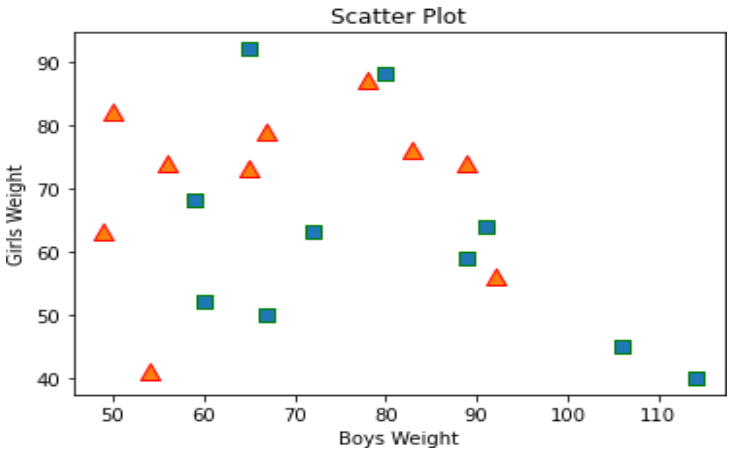


In above plot, we can see the size of each dots.

We can also set the shape instead of dots in scatter plot. Here we set the ***marker*** and ***edgecolor*** as an argument in scatter function to set the shape with edge color in scatter plot.

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

Boys_Weight = [67, 89, 72, 114, 65, 80, 91, 106, 60, 59]
Girls_Weight = [50, 59, 63, 40, 92, 88, 64, 45, 52, 68]
plt.scatter(Boys_Weight, Girls_Weight, marker ="s", edgecolor ="green", s=50)

Boys_Weight = [54, 67, 92, 56, 83, 65, 89, 78, 50, 49]
Girls_Weight = [41, 79, 56, 74, 76, 73, 74, 87, 82, 63]
plt.scatter(Boys_Weight, Girls_Weight, marker ="^", edgecolor ="red", s=100)

plt.title("Scatter Plot")
plt.xlabel("Boys Weight")
plt.ylabel("Girls Weight")

plt.show()
```

The above code will create scatter plot as follow:



In above plot, we can see the shape with edge color.

7.4.3 Line Chart

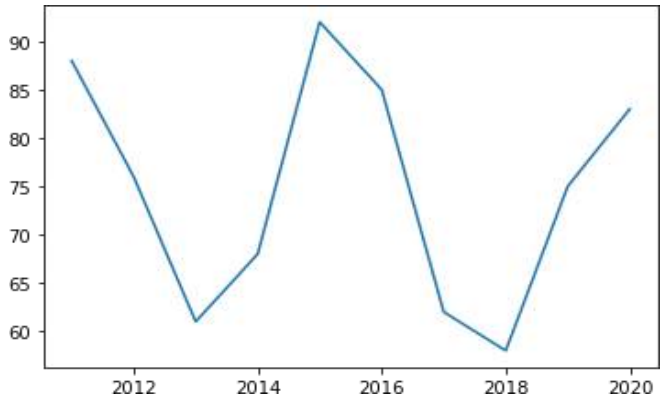
Line chart is used to shows the relation between two datasets on a different axis. There are multiple features available such as line color, line style, line width etc. It is also known as time series plot.

Matplotlib is most popular library for plotting different chart. Line chart is one of them. The ***plot()*** function is used to create a line chart. Here we will see some examples of line chart in Python.

Example: Here we take an example of numbers of students enroll in specific course in different year. The x-axis represents **"Year"** values and y-axis represents **"Student"**.

```
# Importing library
from matplotlib import pyplot as plt

# Data values
Year = [2011, 2012, 2013, 2014, 2015, 2016, 2017, 2018, 2019, 2020]
Student = [88,76,61,68,92,85,62,58,75,83]

# Plotting the line
plt.plot(Year, Student)

# Show plot
plt.show()
```

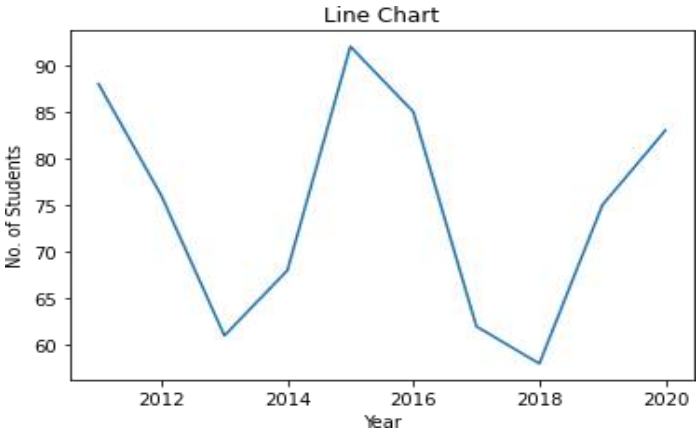
The above code will create line chart as follow:



In this chart, there is no label on both the axis and title of chart. Label is required to understand the dimensions of chart. The following code will create the line chart with ***title*** and ***labeled*** on both axes.

```
from matplotlib import pyplot as plt

Year = [2011, 2012, 2013, 2014, 2015, 2016, 2017, 2018, 2019, 2020]
Student = [88,76,61,68,92,85,62,58,75,83]

plt.title("Line Chart")
plt.xlabel("Year")
plt.ylabel("No. of Students")

plt.plot(Year, Student)

plt.show()
```

The above code will create line chart as follow:

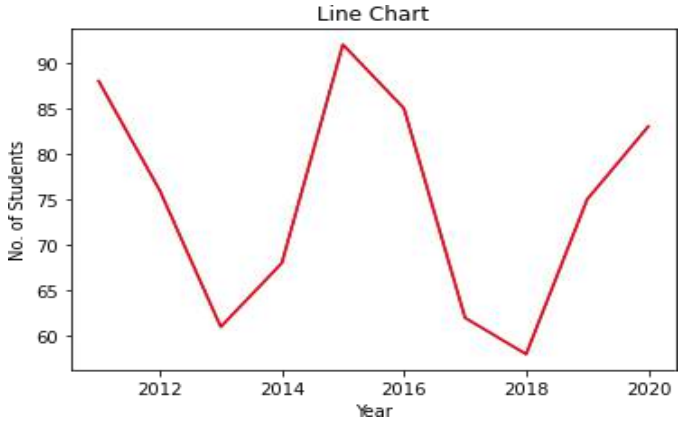


We can set the line color also using ***color*** as an argument. Here we set ***red*** color to the line.

```
from matplotlib import pyplot as plt
```

```
Year = [2011, 2012, 2013, 2014, 2015, 2016, 2017, 2018, 2019, 2020]
Student = [88,76,61,68,92,85,62,58,75,83]

plt.title("Line Chart")
plt.xlabel("Year")
plt.ylabel("No. of Students")

plt.plot(Year, Student)

plt.plot(Year, Student, 'red')

plt.show()
```

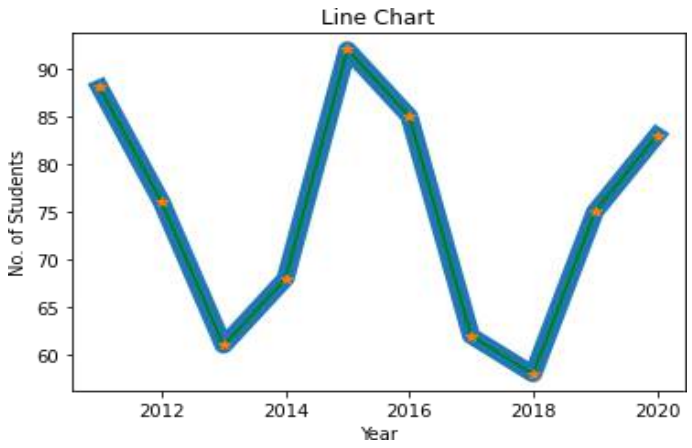
The above code will create line chart as follow:



We can set the line width also using ***linewidth*** or ***lw*** as an argument. Here we set ***10*** to the linewidth.

```
from matplotlib import pyplot as plt

Year = [2011, 2012, 2013, 2014, 2015, 2016, 2017, 2018, 2019, 2020]
Student = [88,76,61,68,92,85,62,58,75,83]

plt.title("Line Chart")
plt.xlabel("Year")
plt.ylabel("No. of Students")

plt.plot(Year, Student)

plt.plot(Year, Student, 'green', linewidth=10)
plt.plot(Year, Student,'*')
plt.show()
```

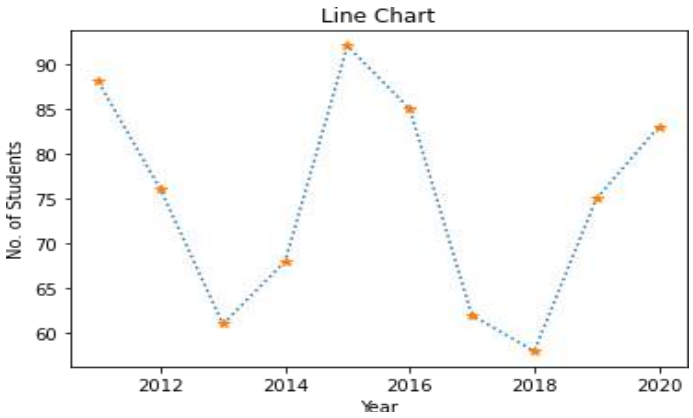
The above code will create line chart as follow:



We can set the line style also using ***linestyle*** or ***ls*** as an argument. There are various types of style available such as solid, dotted, dashed and dashdot. Here we set ***dotted*** as a linestyle in line chart.

```
from matplotlib import pyplot as plt

Year = [2011, 2012, 2013, 2014, 2015, 2016, 2017, 2018, 2019, 2020]
Student = [88,76,61,68,92,85,62,58,75,83]

plt.title("Line Chart")
plt.xlabel("Year")
plt.ylabel("No. of Students")

plt.plot(Year, Student, linestyle = 'dotted')

plt.plot(Year, Student,'*')

plt.show()
```

The above code will create line chart as follow:

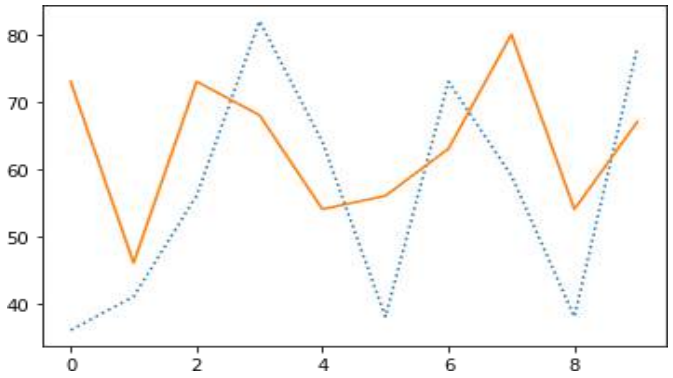


We can set the multiple lines in a single line chart. Here x-axis and y-axis represent the different values. We plot the line chart separately for both the axis. Here we set the ***dotted*** as a linestyle in x-axis.

```
from matplotlib import pyplot as plt

X = [36,41,56,82,64,38,73,59,38,78]
Y = [73,46,73,68,54,56,63,80,54,67]

plt.plot(X, linestyle = 'dotted')

plt.plot(Y)

plt.show()
```

The above code will create line chart as follow:

Here the dotted line represents the x-axis and solid line represent the y-axis.



7.4.4 Bar Chart

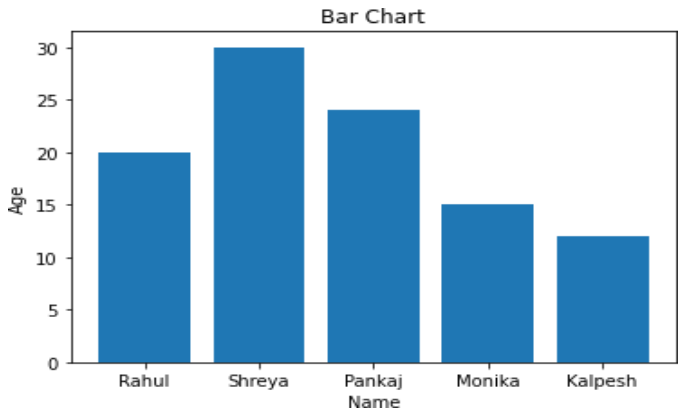
Bar chart or bar plot is representing the category of data with rectangular bars with different heights and lengths with reference to the values that they present. The ***bar()*** function is used to create a bar chart. The bar chart can be plotted both horizontally and vertically.

The bar chart describes the comparisons between distinct categories. One axis represents the particular categories being compared and another axis represent the measured values respected to those categories. The numerical values of variables in a dataset represent the height or length of bar.

Example: Here we take an example of students name and age. The x-axis represents **"Name"** and y-axis represents **"Age"**. Here we also set the chart title and labels on both the axis.

```
# Importing library
from matplotlib import pyplot as plt
import numpy as np

Name = np.array(["Rahul", "Shreya", "Pankaj", "Monika", "Kalpesh"])
Age = np.array([20, 30, 24, 15, 12])

# Labelling the axes and title
plt.title("Bar Chart")
plt.xlabel("Name")
plt.ylabel("Age")

# Plotting the bar
plt.bar(Name, Age)
```

The above code will create bar chart as follow:

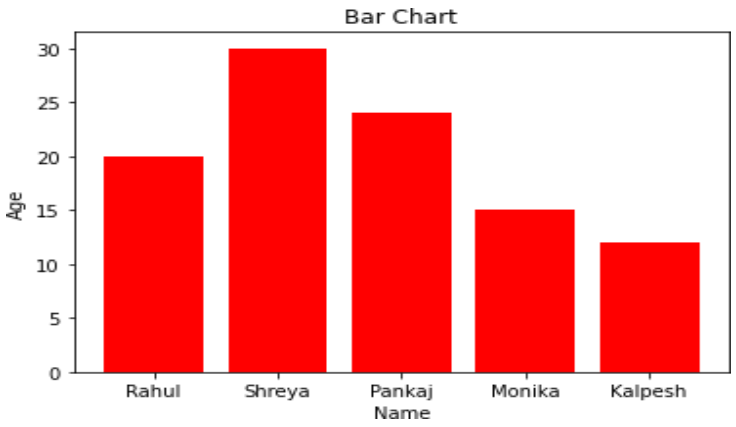


We can change the bar color also. We set ***color*** as an argument to change the color of bar. Here we set the ***red*** as a color of bar.

```
from matplotlib import pyplot as plt
import numpy as np

Name = np.array(["Rahul", "Shreya", "Pankaj", "Monika", "Kalpesh"])
Age = np.array([20, 30, 24, 15, 12])
```

```
plt.title("Bar Chart")
plt.xlabel("Name")
plt.ylabel("Age")

plt.bar(Name, Age, color="red")
```

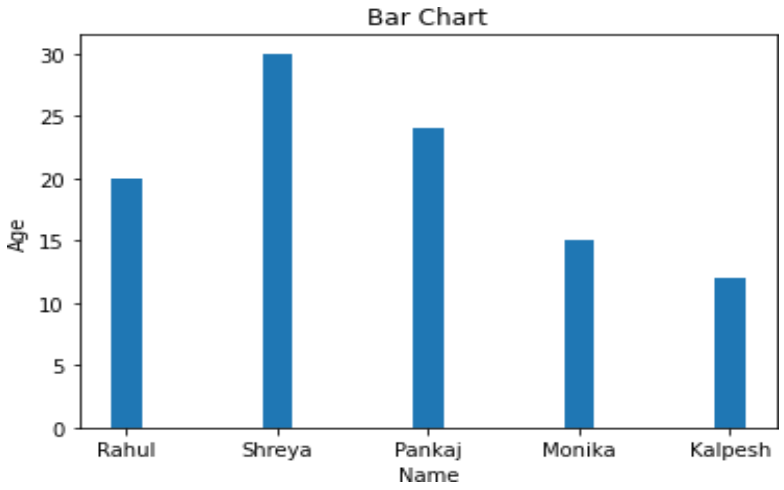
The above code will create bar chart as follow:



We can set the bar width also. We set ***width*** as an argument to set the width of bar. Here we set ***0.2*** as a width of bar.

```
from matplotlib import pyplot as plt
import numpy as np

Name = np.array(["Rahul", "Shreya", "Pankaj", "Monika", "Kalpesh"])
Age = np.array([20, 30, 24, 15, 12])

plt.title("Bar Chart")
plt.xlabel("Name")
plt.ylabel("Age")

plt.bar(Name, Age, width=0.2)
```

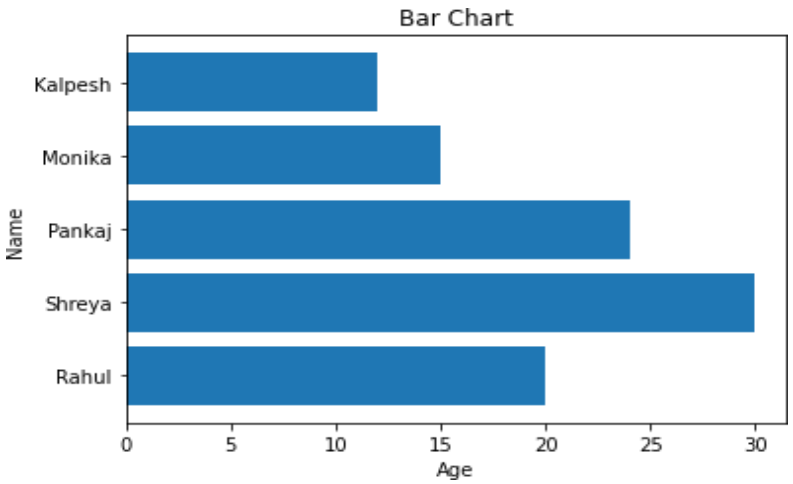
The above code will create bar chart as follow:



We can display bar horizontally also instead of vertically. We set the bar horizontally by using ***barh()*** function.

```
from matplotlib import pyplot as plt
import numpy as np

Name = np.array(["Rahul", "Shreya", "Pankaj", "Monika", "Kalpesh"])
Age = np.array([20, 30, 24, 15, 12])

plt.title("Bar Chart")
plt.xlabel("Age")
plt.ylabel("Name")

plt.barh(Name, Age)
```

The above code will create bar chart as follow:



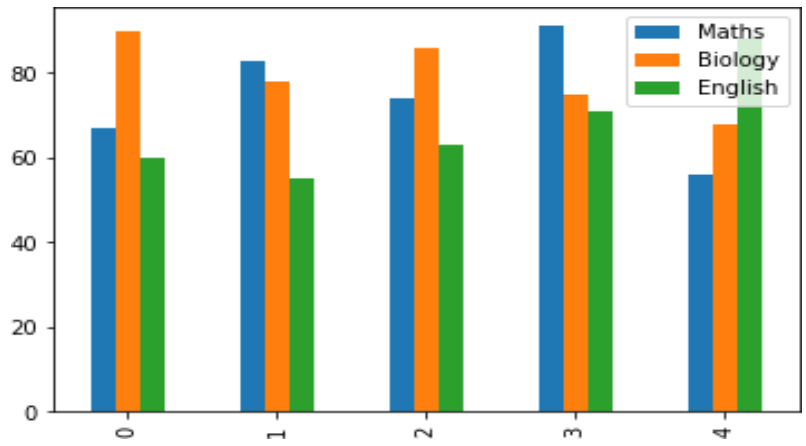
The multiple bar chart is used to represent the comparison among the different variables in a dataset. We can set the thickness and positions of bars also.

Here, we take an example of marks of different subject with the name of students. X-axis represents the students and y-axis represents the marks of different subjects.

```
from matplotlib import pyplot as plt
import pandas as pd

df = pd.DataFrame({
"Name":["Rahul","Shreya","Pankaj","Monika","Kalpesh"],
"Maths":[67,83,74,91,56],
"Biology":[90,78,86,75,68],
"English":[60,55,63,71,88]})

df.plot.bar()
```

The above code will create bar chart as follow:

Here, we show the comparisons of marks of different subjects of the students.



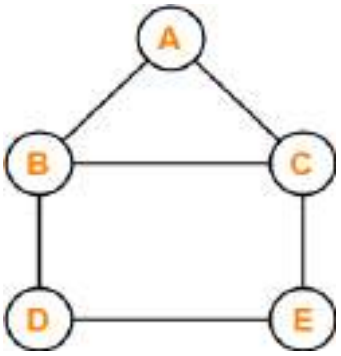
7.5 GRAPH

Graph is a pictorial representation of a set of objects. Some pairs of objects are connected through links. The interaction of link is denoted by points which is known as ***vertices***. The link which is used to connect the vertices is called ***edges***. We can perform some operation on graph such as:

- Display vertices
- Display edges
- Add new vertex
- Add new edge
- Create graph

The dictionary data type is used to present a graph in Python. The vertices of a graph are representing as the keys of dictionary and the links between the vertices also called edges which represent as the values of dictionary

Take the following graph as an example.



The above graph consists the following vertices (V) and edges (E).

```
V = {A, B, C, D, E}
E = {AB, AC, BC, BD, CE, DE}
```

The above graph represents using Python as below.

```
# Create the dictionary with graph elements
graph = { "a" : ["b","c"],
          "b" : ["a","c", "d"],
          "c" : ["a","b", "e"],
          "d" : ["b","e"],
          "e" : ["c","d"]
        }

# Print the graph
print(graph)
```

The code will give the following output.

```
{'a': ['b', 'c'], 'b': ['a', 'c', 'd'], 'c': ['a', 'b', 'e'], 'd': ['b', 'e'], 'e': ['c', 'd']}
```

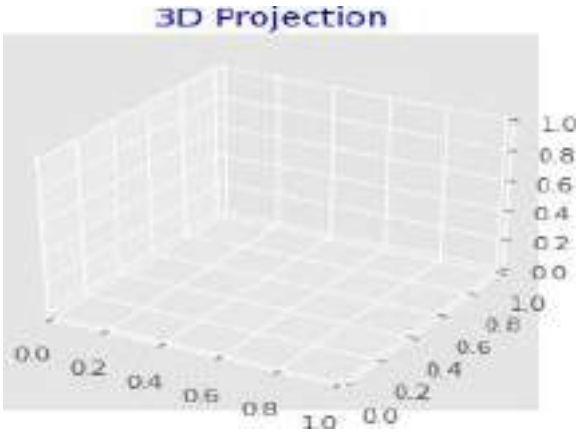
7.6 3D VISULIZATION AND PRESENTATION

The matplotlib library is most popular for data visualization in Python. It was initially designed for two-dimension plotting, but some three-dimension plotting utilities were built on top matplotlib‟s two-dimension display in later versions. Three dimensional plots are enabling by importing the mplot3d toolkit, which included with the main matplotlib.

A three-dimensional axis can be created by using the keyword ***projection="3d"*** as follows.

```
# Importing library
import matplotlib.pyplot as plt

# 3D projection plot
fig = plt.figure()
ax = plt.axes(projection ="3d")
plt.title("3D Projection", color="blue")
```

The above code will plot as follow:



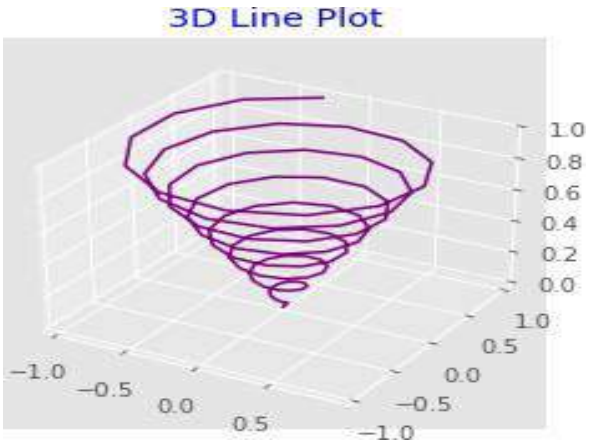
7.6.1 3D Line Plot

This is a most basic three-dimensional plot created using set of (x, y, z) triples. It is also known as time series plot. This plot is plotted using ***ax.plot3D*** function as follow:

```
# Importing library
from mpl_toolkits import mplot3d
import numpy as np
import matplotlib.pyplot as plt

# 3D projection
fig = plt.figure()
ax = plt.axes(projection = "3d")

# All three axis
z = np.linspace(0, 1, 100)
x = z * np.sin(50 * z)
y = z * np.cos(50 * z)

# 3D Line plotting
ax.plot3D(x, y, z, "purple")
ax.set_title("3D Line Plot", color="blue")
```

```
plt.show()
```

The above code will plot as follow:

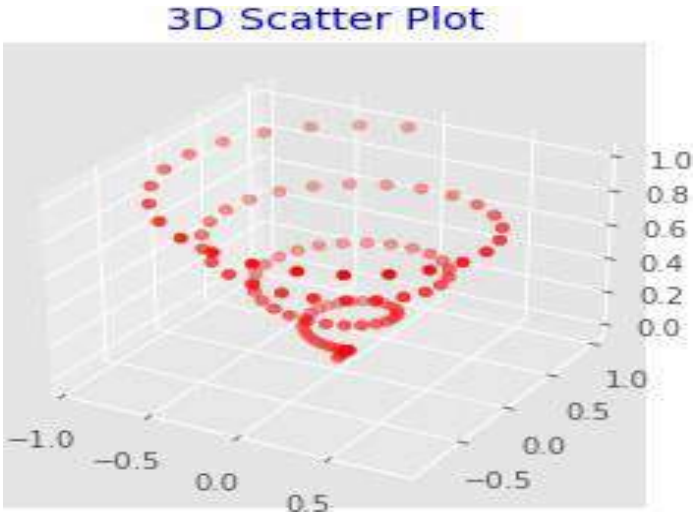


7.6.2 3D Scatter Plot

This is a basic three-dimensional plot created using set of (x, y, z) triples. It represents the data points on three axes to show the relationship between three variables. This plot is plotted using ***ax.scatter3D*** function as follow:

```
# Importing library
from mpl_toolkits import mplot3d
import numpy as np
import matplotlib.pyplot as plt

# 3D projection
fig = plt.figure()
ax = plt.axes(projection = "3d")

# All three axis
z = np.linspace(0, 1, 100)
x = z * np.sin(25 * z)
y = z * np.cos(25 * z)

# 3D scatter plotting
ax.scatter3D(x, y, z, color="red")
ax.set_title("3D Scatter Plot", color="blue")
plt.show()
```

The above code will plot as follow:

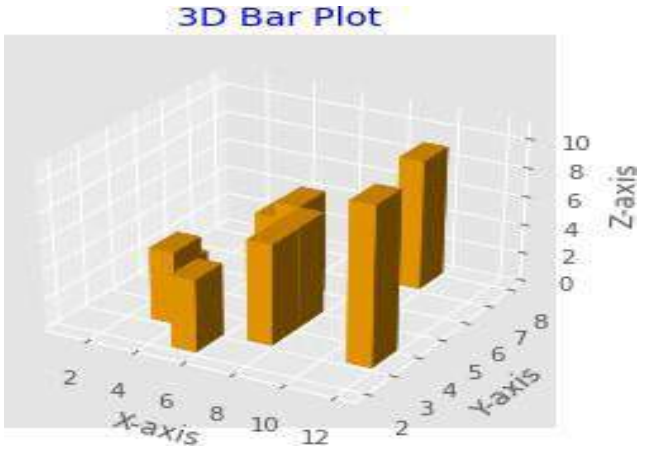


7.6.3 3D Bar Plot

The three-dimensional bar plot is used to compare the relationship between three variables. This plot is plotted using ***ax.bar3d*** function as follow:

```
# Importing library
from mpl_toolkits.mplot3d import axes3d
import matplotlib.pyplot as plt
import numpy as np
from matplotlib import style

# 3D projection
fig = plt.figure()
ax = fig.add_subplot(111, projection="3d")

# All three axis
x = [1,3,5,7,9,11,7,3,5,6]
y = [5,7,2,6,4,6,5,3,6,7]
z = np.zeros(10)

dx = np.ones(10)
dy = np.ones(10)
dz = [1,3,5,7,9,11,7,5,3,7]

# 3D Bar plotting
ax.bar3d(x, y, z, dx, dy, dz, color="orange")
ax.set_title("3D Bar Plot", color="blue")
ax.set_xlabel("X-axis")
ax.set_ylabel("Y-axis")
ax.set_zlabel("Z-axis")
plt.show()
```

The above code will plot as follow:

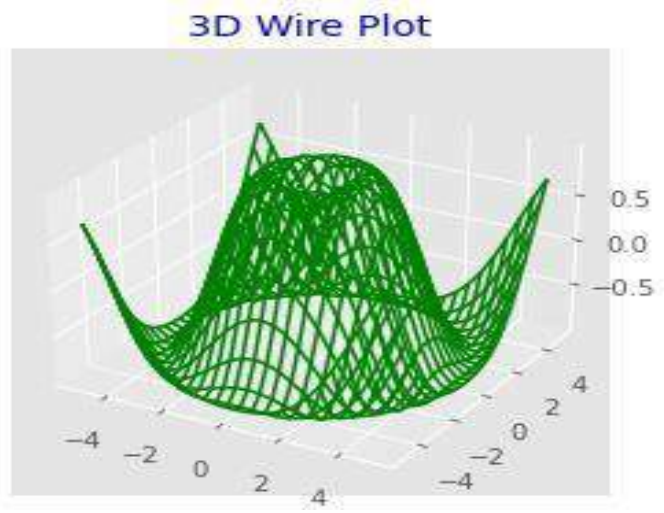


7.6.4 3D Wire Plot

This plot takes a grid of values and draws the lines between nearby points on three-dimensional surface. This plot is plotted using ***ax.plot_wireframe*** method as follow:

```
# Importing library
import numpy as np
import matplotlib.pyplot as plt

# 3D projection
fig = plt.figure()
ax = plt.axes(projection="3d")

# Function
def func(x, y):
    return np.sin(np.sqrt(x * x + y * y))

# All three axis
x = np.linspace(-5, 5, 25)
y = np.linspace(-5, 5, 25)
X, Y = np.meshgrid(x, y)
Z = func(X, Y)

# 3D wireframe plotting
ax.plot_wireframe(X, Y, Z, color="Green")
ax.set_title("3D Wire Plot", color="blue")
plt.show()
```

The above code will plot as follow:

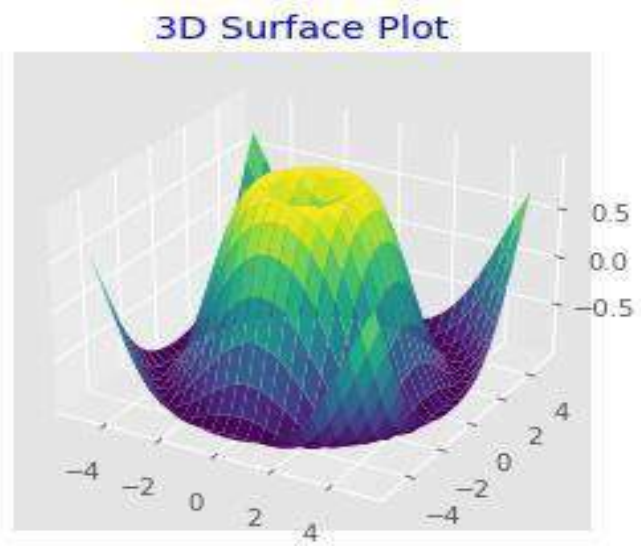


7.6.5 3D Surface Plot

This plot is like as wireframe plot, but each face of the wireframe is filled polygon. This plot shows the functional relationship between one dependent variable and two independent variables. This plot is plotted using ***ax.plot_surface*** method as follow:

```
# Importing library
import numpy as np
import matplotlib.pyplot as plt

# 3D projection
fig = plt.figure()
ax = plt.axes(projection="3d")

# Function
def func(x, y):
    return np.sin(np.sqrt(x * x + y * y))

# All three axis
x = np.linspace(-5, 5, 25)
y = np.linspace(-5, 5, 25)
X, Y = np.meshgrid(x, y)
Z = func(X, Y)

# 3D surface plotting
ax.plot_surface(X, Y, Z, rstride=1, cstride=1,cmap='viridis', edgecolor='none')
ax.set_title("3D Surface Plot", color="blue")
plt.show()
```

The above code will plot as follow:



7.7 SUMMARY

The students will learn many things in this module and they will be able to perform the various data science related operation using Python.

- Ability to do understand the generative and predictive modeling.
- Ability to plot the various types of charts including histogram, scatter plot, line or timeseries plot, bar plot from the dataset.
- Ability to prepare a graph.
- Ability to plot the various types of three-dimension graph.

REFERENCES

Books

11. Davy Cielen, Arno D. B. Meysman, Mohamed Ali: Introducing Data Science, Manning Publications Co.
12. Stephen Klosterman (2019) : Data Science Projects with Python, Packt Publishing
13. Jake VanderPlas (2017) : Python Data Science Handbook: Essential Tools for Working with Data, O‟Reilly
14. Wes McKinnery and Pandas Development Team (2021) : pandas : powerful Python data analysis toolkit, Release 1.2.3

Web References

1. https://www.oreilly.com
2. https://www.geeksforgeeks.org
3. https://www.tutorialspoint.com
4. https://www.w3schools.com
5. https://pandas.pydata.org
6. https://pbpython.com

7. https://jakevdp.github.io
8. https://matplotlib.org
9. https://www.logianalytics.com
10. https://seleritysas.com

QUESTIONS

Short Answer:

11. What is generative modeling?
12. What is predictive modeling?
13. List the types of predictive modeling.
14. What is chart?
15. What is histogram?
16. List types of charts.
17. What is graph?
18. List the 3D plots.

Long Answer:

13. Explain predictive modeling in details.
14. Explain histogram with example.
15. Explain scatter plot with example.
16. Explain line chart or time series plot with example.
17. Explain bar plot with example.
18. Explain three-dimensional plot with example.

PRACTICALS

7. Create and display histogram with different arguments.
8. Create and display scatter plot with different arguments.
9. Create and display time series plot with different arguments.
10. Create and display bar plot with different arguments.
11. Create a simple graph.
12. Create and display various 3D plots.