Data Science Technology Stack

Rapid Information Factory (RIF) Ecosystem:

Rapid Information Factory (RIF) System is a technique and tool which is used for processing the data in the development. The Rapid Information Factory is a massive parallel data processing platform capable of processing theoretical unlimited size data sets.

The Rapid Information Factory (RIF) platform supports five high-level layers:

- **Functional Layer.**

  The functional layer is the core processing capability of the factory.

  Core functional data processing methodology is the **R-A-P-T-O-R** framework.

  - ***Retrieve Super Step.***

    The retrieve super step supports the interaction between external data sources and the factory.

  - ***Assess Super Step.***

    The assess super step supports the data quality clean-up in the factory.

  - ***Process Super Step.***

    The process super step converts data into data vault.

- ***Transform Super Step.***

  The transform super step converts data vault via sun modeling into dimensional modeling to form a data warehouse.

- ***Organize Super Step.***

  The organize super step sub-divides the data warehouse into data marts.

- ***Report Super Step.***

  The report super step is the Virtualization capacity of the factory.

- **Operational Management Layer.**
- **Audit, Balance and Control Layer.**
- **Utility Layer.**

  Common components supporting other layers.

  - *Maintenance Utilities.*
  - *Data Utilities.*
  - *Processing Utilities.*
- **Business Layer.**

  Contains the business requirements (Functional and Non-functional)

Data Science Storage Tools:

- Data Science ecosystem has a bunch of series of tools which are used to build your solution. By using this tools and techniques you will get rapid information in advanced for its better capability and new development will occur each day.

- There are two basic data processing tools to perform the practical of data science as given below:

- Schema on write ecosystem.

    - Traditional Relational Database Management System requires a schema before loading the data. Schema basically denotes the organizational data which is like a blueprint, describing how the database should be constructed.

    - Schema is a single structure which represents logical view of entire database. It represents how the data is organized and related between them.

    - It is responsible of the database designer to design the database perfect to understand the logic and structure with the help of programmer.

    - Relational Database Management System is used and designed to store the data.

    - To Retrieve the data from the relational database system, you need to run the specific structure query language to perform these tasks.

- A traditional database management system only works with schema and it will work once the schema is described and there is only one point of view to describe and view the data into the database.

- It stores a dense of data and all the data are stored into the datastore and schema on write widely use methodology to store the dense data.

- Schema on write schemas are build with the purpose which makes them change and maintain the data into the database.

- When there is a lot of raw data which are available for the processing, during, some of the data are lost and it makes them weak for future analysis.

- If some important data are not stored into the database then you cannot process the data for further data analysis.

❖ Schema on read ecosystem.

- Schema on read ecosystem does not need schema, without this you can load the data into the database.

- This type of schema stores the minimal data with values into the database and some of the important schema are applied during the query phase.

- It has the capabilities to store the structure, semi-structure, unstructured data and it has potential to apply most of the flexibilities when we request the query during the execution.

- These types of ecosystem are applicable for both experimental and exploration of data to retrieve the data from the schema or structure.

- Schema on read generate the fresh and new data and increase the speed of data generation as well as reduce the cycle time of data availability of actionable information.

- These types of ecosystem that means schema on read and schema on write are very useful and essential for data scientist and engineering personal for better understanding about data preparation, modeling, development, and deployment of data into the production.

- When you apply schema on read on structure, un-structure, and semi-structure, it would generate very slow result because it does not have the schema fast retrieval of data into the data warehouse.

- Schema on read follow the agile way of working and it has capabilities and potential to work like NoSQL database as it works in the environment.

- Some time schema on read through the error during the query time because there are three type of data stored into the database like structure, un-structure, and semi-structure. There is no better process and rule and regulation for fast and better retrieval of data from structure database.

Data Lake:

- A Data Lake is storage repository of large amount of raw data that means structure, semi-structure, unstructured data.

- This is the place where you can store three types of data structure, semi-structure, unstructured data with no fix amount of limit and storage to store the data.

- If we compare schema on write and data lake then we will find that schema on write store the data into the data warehouse in predefined database on the other hand data lake store the less data structure to store the data into the database .

- Data Lake follow to store less data into the structure database because it follows the schema on read process architecture to store the data.

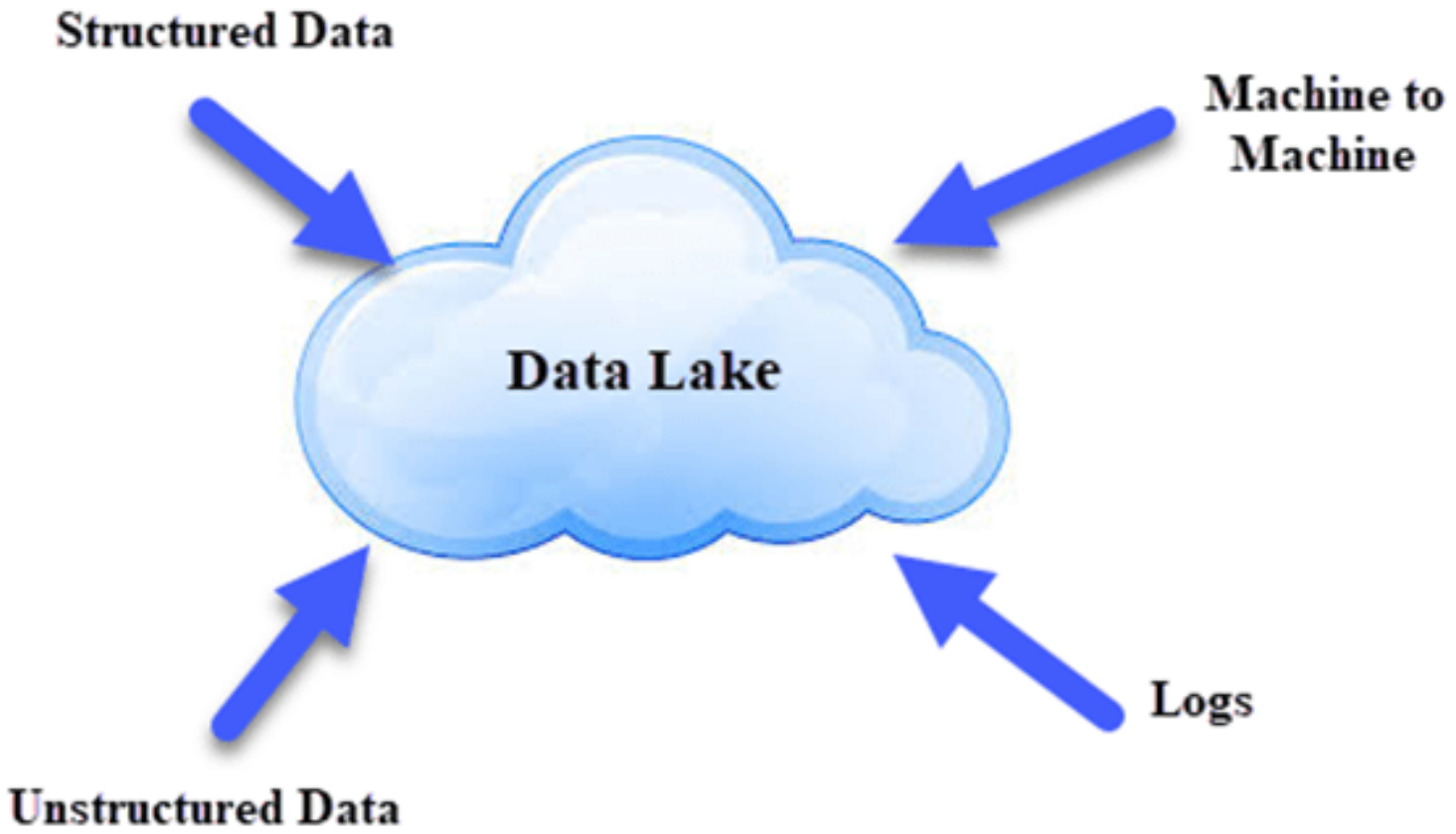
- Data Lake allow us to transform the raw data that means structure, semi-structure, unstructured data into the structure data format so that SQL query could be performed for the analysis.

- Most of the time data lake are deployed by using the distributed data object storage database which enable the schema on read so that business analytics and data mining tools and algorithms can be applied on the data.

- Retrieval of data is so fast because there is no schema applied. Data must be access without any failure or any complex reason.

- Data Lake is similar to real time river or lake where the water comes from different- different places and at the last all the small- small river and lake are merged into the big river or lake where large amount of water are stored, whenever there is need of water then it can be used by anyone.

- It is low cost and effective way to store the large amount of data stored into centralized database for further organizational analysis and deployment.



Data Vault:

- Data Vault is a database modeling method which is designed to store the long-term historical storage amount of data and it can controlled by using the data vault.

- In Data Vault, data must come from different sources and it is designed in such a ways that data could be loaded in parallel ways so that large amount of data implementation can be done without any failure or any major design.

- Data Vault is the process of transforming the schema on read data lake into schema on write data lake.

- Data Vault are designed schema on read query request and after that it would be converted into the data lake because schema on read increase the speed of generating new data for the better analysis and implementation.

- Data Vault store a single version of data and does not distinguish between good data and bad data.

- Data Lake and Data Vault are built by using the three main component or structure of data i.e. Hub, Link and satellite.

1. Hub :

   ❖ Hub has unique business key with low amount of data to be changed and meta data that means data is the main source of generating the hubs.

   ❖ Hub contains surrogate key for each metadata information and each hub items i.e. origin of this business key.

   ❖ Hub contains a set of unique business key that will never change over a period manner.

- There are different types of hubs like person hub, time hub, object hub, event hub, locations hub. The Time hub contains ID Number, ID Time Number, ZoneBasekey , DateTimekey, DateTimeValue and all these links are interconnected to each other like Time-Person, Time-Object, Time-Event, Time-Location, Time-Links etc.

- The Person hub contains IDPersonNumber, FirstName, SecondName, LastName, Gender, TimeZone, BirthDateKey, BirthDate and all these links are interconnected to each other like Person- Time, Person-Object, Person-Location, Person-Event, Person-Link etc.

- The Object hub contains IDObjectNumber, ObjectBaseKey, ObjectNumber, ObjectValue and all these links are interconnected to each other like Object-Time, Object-Link,Object-Event,Object-Location,Object-Person etc.

- The Eventhub contains IDEventNumber, EventType, EventDescription and all these links are interconnected to each other like Event-Person, Event-Location, Event-Object, Event-Time etc.

- The Location hub contains IDLocationNumber, ObjectBaseKey, LocationNumber , LocationName, Longitude and Latitude all these links are interconnected to each other like Location-Person, Location-Time, Location-Object ,Location-event etc.

Link :

- Link plays a very important role during transaction and association of business key. The Table relate to each other depending upon the attribute of table like that one to one relationship, one to many relationships, Many to One relationship, Many to many relationships.

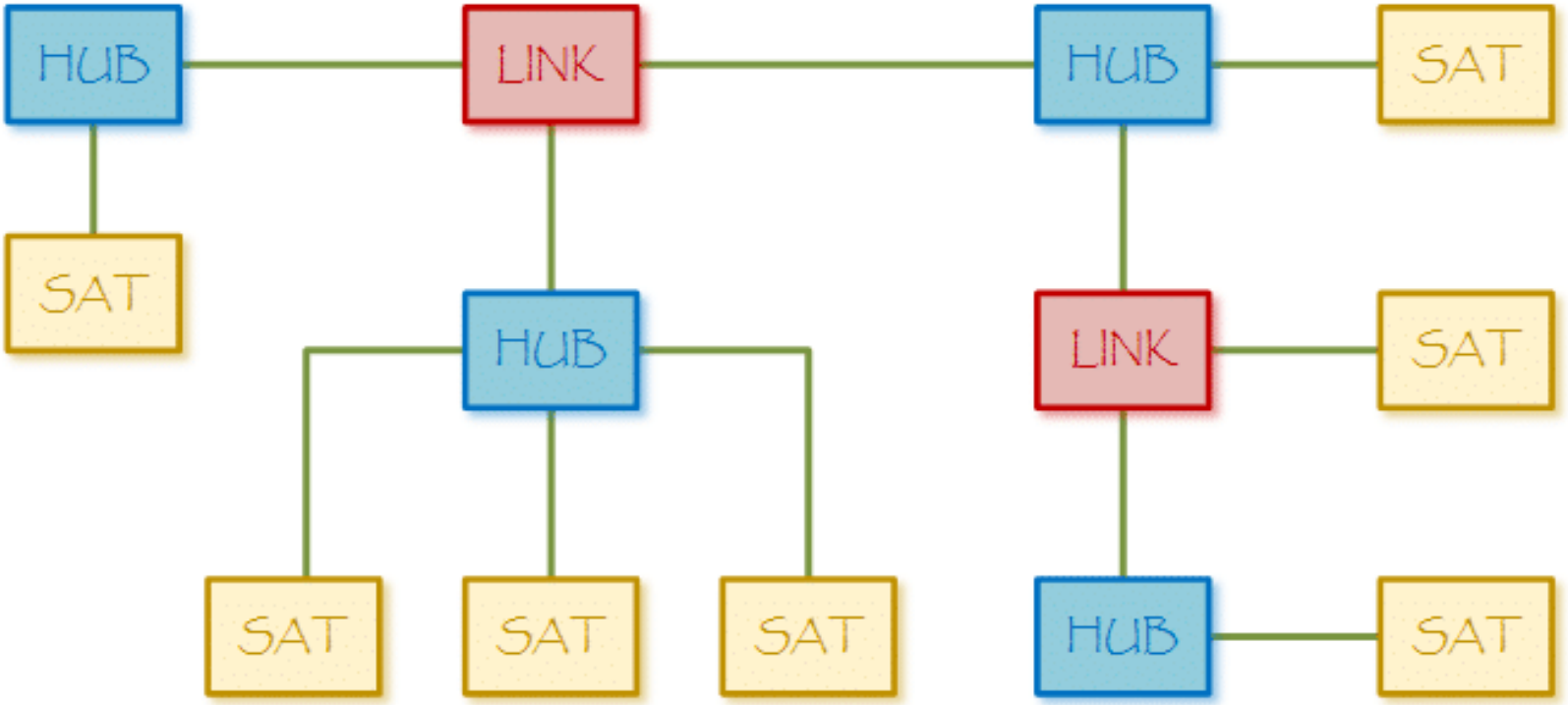
- Link represent and connect only element in the business relationships because when one node or link relate to one or another link on that time data transfers smoothly.

Satellites :

- When the hubs and links produce and form the structure of satellites which store no chronological structure of data means then it would not provide the information about the mean, median, mode, maximum, minimum, sum of the data.

- Satellites are the strong structure of data that store a detailed information about the related data or business characteristics key and stores large volume of data vault.

- The combinations of all these three i.e. hub, link, and satellites are formed together to help the data analytics and data scientists and data engineer to store the business structure, types of information or data into it.



Data Science Processing Tools:

- The process of transforming the data, data lake to data vault and then transferring the data vault into data warehouse.
- Most of the data scientist and data analysis, data engineer uses these data science processing tool to process and transfer the data vault into data warehouse.

1. **Spark :**

- Apache Spark is an open source clustering computing framework. The word open source means it is freely available on internet and just go on internet and type apache spark and you can get freely source code, you can download and use according to your wish.

- Apache Spark was developed at AMP Lab of university of California, Berkeley and after that all the code and data was donated to Apache Software Foundation to keep doing changes over a time and make it more effective, reliable, portable that will run on all the platform.

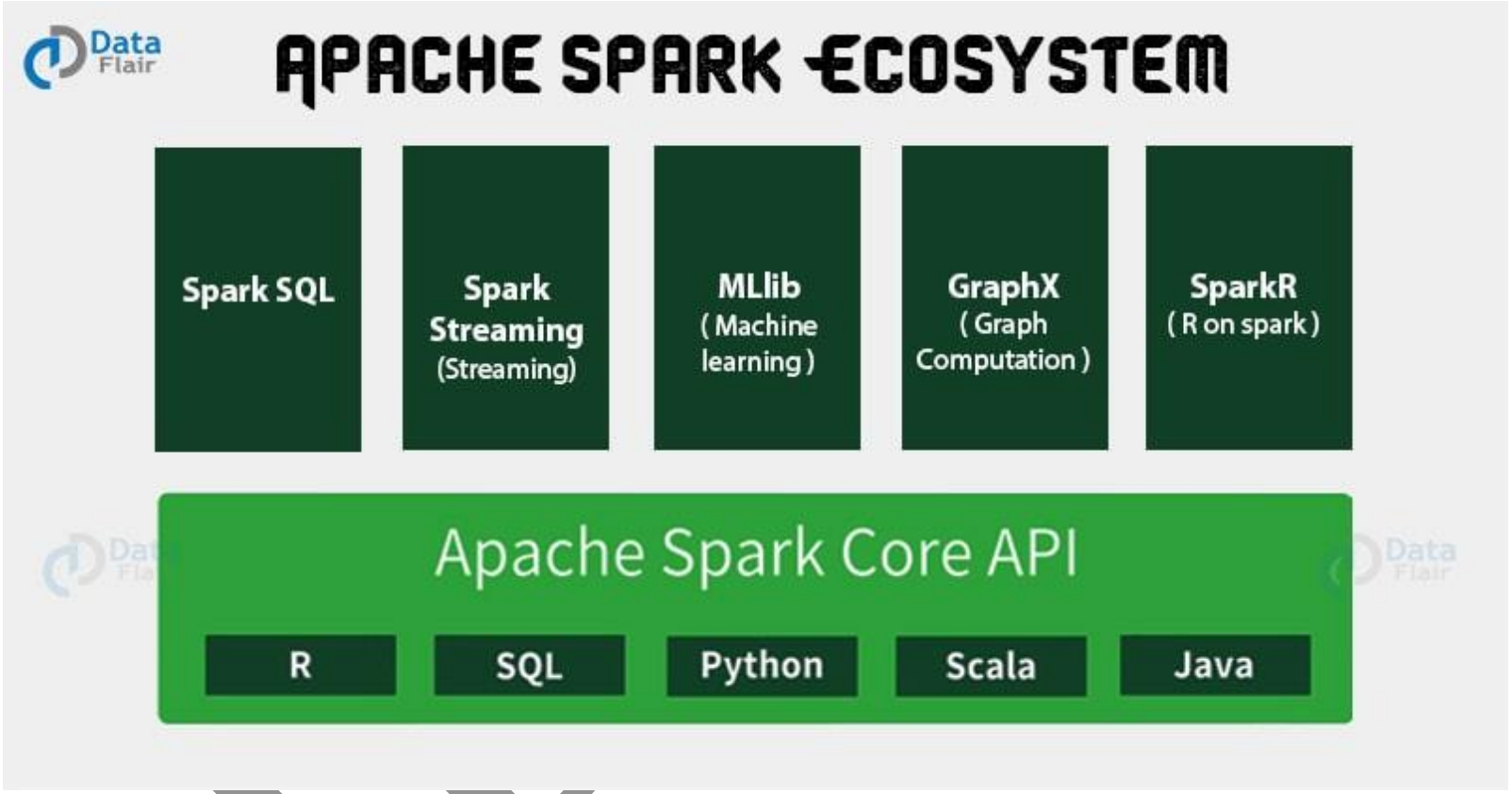
- Apache Spark provide an interface for the programmer and developer to directly interact with the system and make data parallel and compatible with data scientist and data engineer.

- Apache Spark has the capabilities and potential, process all types and variety of data with repositories including Hadoop distributed file system, NoSQL database as well as apache spark.

- IBM are hiring most of the data scientist and data engineer, who has more knowledge and information about apache spark project so that innovation could perform an easy way and will come up with more feature and changing.

- Apache Spark has potential and capabilities to process the data very fast and hold the data in memory and transfer the data into memory data processing engine.

- It has been built on top of the Hadoop distributed file system which make the data more efficient, more reliable and make it more extendable on the Hadoop map reduce.


Data Flair
APACHE SPARK ECOSYSTEM
Spark SQL
Spark Streaming (Streaming)
MLlib (Machine learning)
GraphX (Graph Computation)
SparkR (R on spark)
Apache Spark Core API
R
SQL
Python
Scala
Java

2. **Spark Core**:

- Spark Core is base and foundation for over all of the project development and provide some most important Information like distributed task, dispatching, scheduling and basic Input and output functionalities.

- By using spark core, you can have more complex queries that will help us to work with complex environment.

- The distributed nature of spark ecosystem enables you the same processing data on a small cluster, to go for hundreds or thousand of nodes without making any changes.

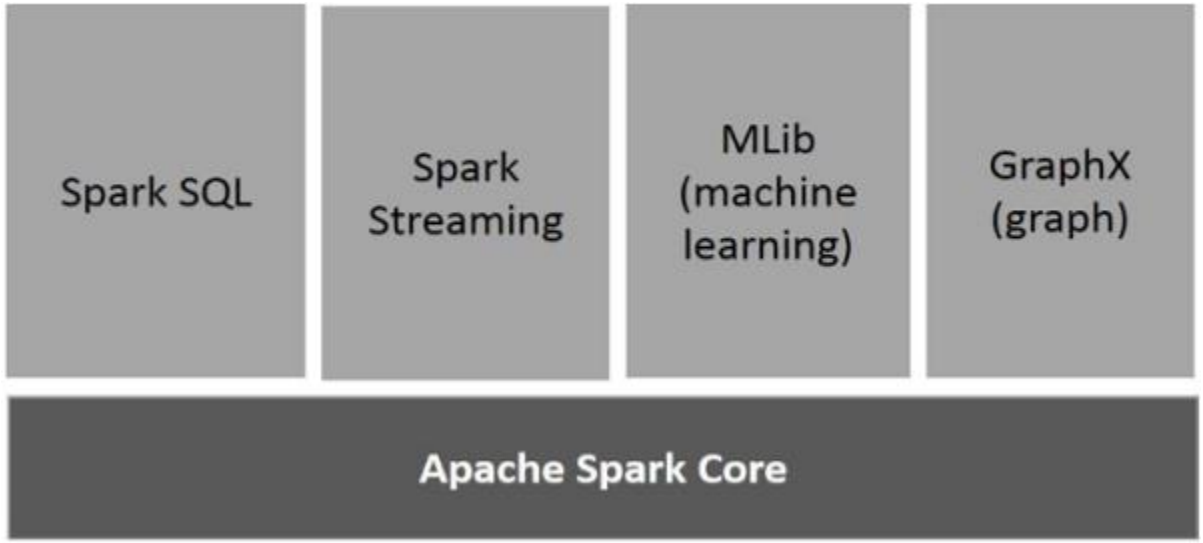
- Apache Spark uses Hadoop in two ways one is storage and second one is for processing purpose.

- Spark is not a modified version of Hadoop distributed file system, because it depend upon the Hadoop which has its own feature and tool for data storage and data processing.

- Apache Spark has a lot of feature which makes it compatible and reliable. Speed is one the most important feature of spark that means with the help of spark, your application are able to run on directly on Hadoop and it is 100 times much faster in the memory.

- Apache Spark Core support many more language and it has its own built in function and API in java, Scala, python that means you can write the application by using the java, python, C++, Scala etc.

- Apache Spark Core has come up with advanced analytics that means it does not support the map and reduce the potential and capabilities to support SQl Queries, Machine Learning and graph Algorithms.



3. **Spark SQL:**

- Spark SQL is a component on top of the Spark Core that presents data abstraction called Data Frames.
- Spark SQL is fast clustering data abstraction, so that data manipulation can be done for fast computation.
- It enables the user to run SQL/HQL on top of the spark and by suing this, we can process the structure, unstructured and semi structured data.

- Apache Spark SQL provide a much relationship between relational database and procedural processing. This comes, when we want to load the data from traditional way into data lake ecosystem.

- Spark SQL is Apache Spark's module for working with structured and semi-structured data and it originated to overcome the limitation of apache hive.

- It always dependent upon the MapReduce engine of Hadoop for execution and processing of data and allows the batch-oriented operation.

- Hive lags in performance uses to MapReduce jobs for executing ad-hoc process and hive does not allow you to resume a job processing, if it fails in the middle.

- Spark performs better operation than hive in many situation. Latency in the terms of hours and CPU reservation time.

- You can integrate the Spark SQL and querying structured, semi-structured data inside the apache spark.

- Spark SQL follows the RDD Model and it also support large job and middle query fault tolerance.

- You can easily connect the Spark SQL with the JDBC and ODBC for better connectivity of business purpose.

4. **Spark Streaming:**

- Apache Spark Streaming enables powerful interactive and data analytics application for live streaming data. In Streaming, data is not fixed and data comes from different source continuously.

- Stream divide the incoming input data into the small-small unit of data for further data analytics and data processing for next level.

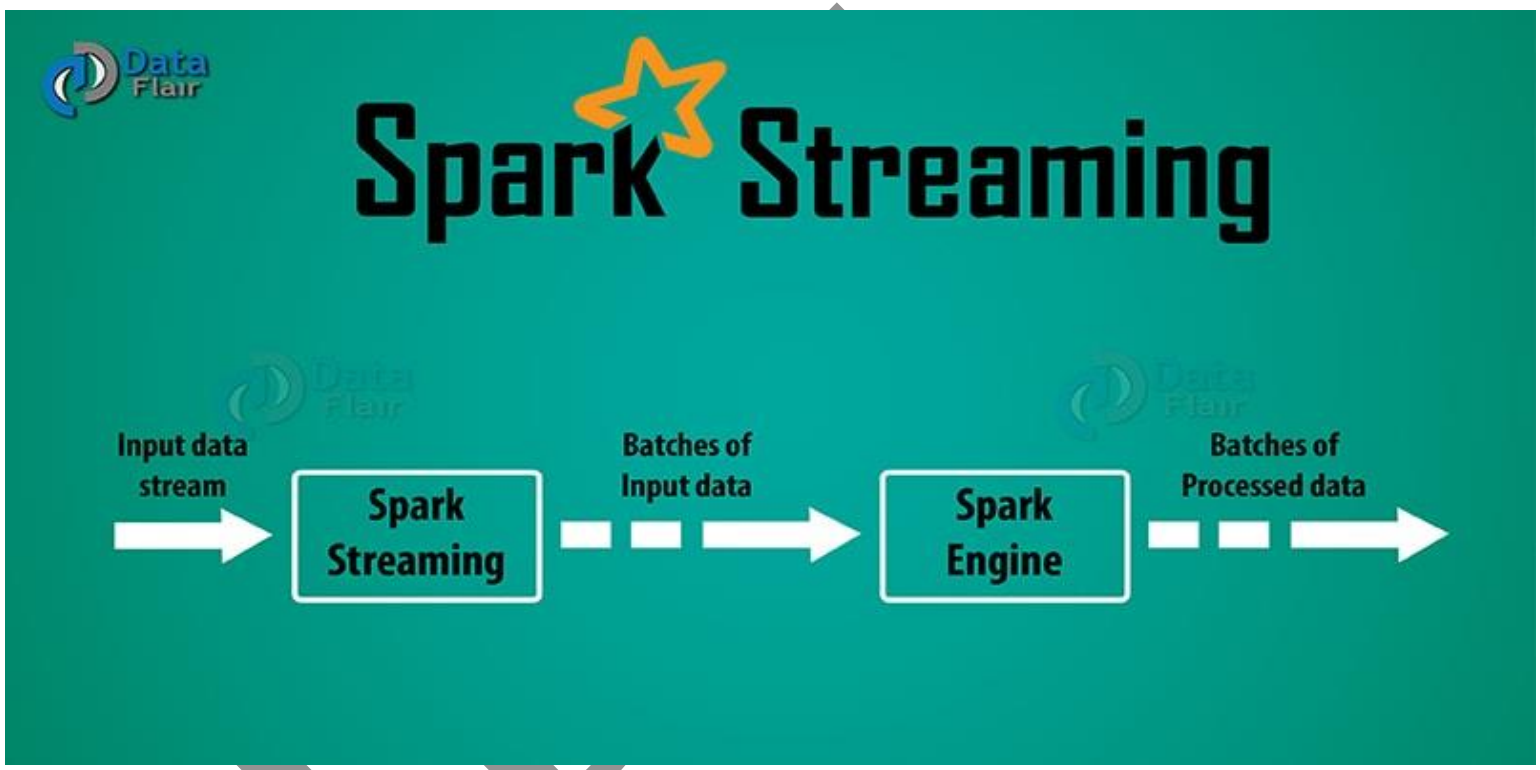
- There are multilevel of processing involved in it. Live streaming data are received and divided into small-small parts or batches and these small-small of data or batches are then processed or mixed by the spark engine to generate or produced the final level of streaming of data.

- Processing of data in the system in Hadoop has very high latency means that data is not received on timely manner and it is not suitable for real time processing requirement.

- Processing of data is generated by storm, if it is not happened again. But this type of mistake and latency give the data loss and repetition of records processing.

- Most of scenario, Hadoop are used for data batching and Apache Spark are used for the live streaming of data.

- Apache Streaming provide and help us to fix these types of issue and provides reliable, portable, scalable, efficiency, and integration of the system.



5. **GraphX:**

Unedited Version: Data science

- GraphX is very powerful graph processing tool application programming interface for apache spark analytics engine.

- GraphX is a new component in a spark for graphs and graphs-parallel computation.
- GraphX follow the ETL process that means Extract, transform and Load, exploratory analysis, iterative graph computation within a single system.

- The usage can be seen in the Facebooks, LinkedIn connection, google map, and internet routers use these types of tool for better response and analysis.

- Graph is an abstract data types that means it is used to implement the directed and undirected graph concepts from the mathematics in the graph theory concept.

- In the graph theory concept, each data associate with some other data with edge like numeric value.

- Every edge and node or vertex have user defined properties and values associated with it.

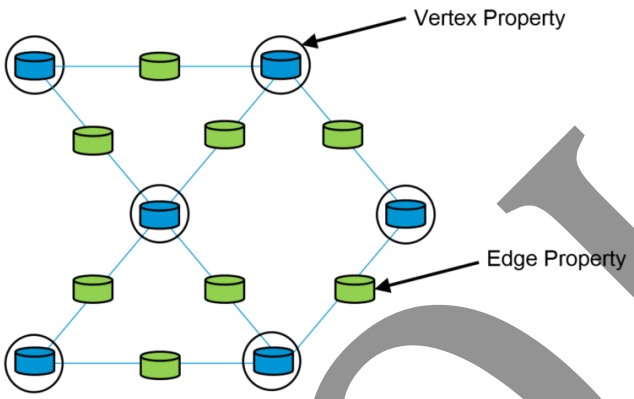


Figure: *Property Graph*

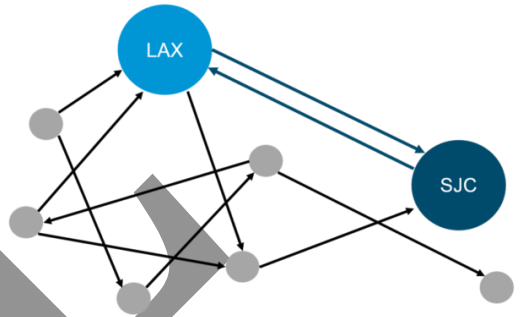


Figure: *An example of property graph*

- GraphX has more **flexibilities** to work with graph and computation. Graph follow the ETL process that means Extract, transform and Load, exploratory analysis, iterative graph computation within a single system.

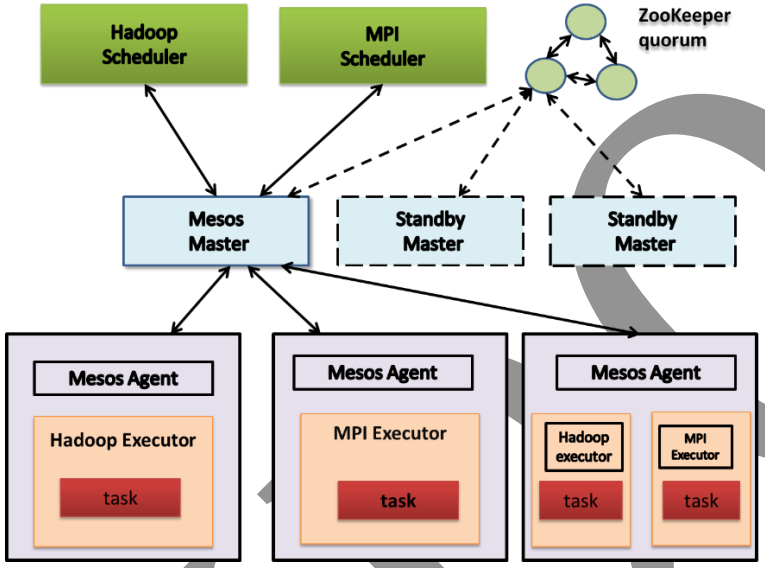
- Speed is one of the most important point in the point of Graph and it is comparable with the fastest graph system while when there is any fault tolerance and provide ease of use.

- We can choose lots of more feature that comes with more flexibilities and reliability and it provide library of graph algorithms.

6. **Mesos:**

- Apache Mesos is an open source cluster manager and it was developed by the universities of California, Berkeley.
- It provides all the required resource for the isolation and sharing purpose across all the distributed application.
- The software we are using for Mesos, provide resources sharing in a fine-grained manner so that improving can be done with the help of this.
- Mesosphere enterprises DC/OS is the enterprise version of Mesos and this run specially on Kafka, Cassandra, spark and Akka .
- It can handle the workload in distributed environments by suing the dynamic sharing and isolation manner.
- Apache Mesos could be deployed and manageable of large amount of data distribution scale cluster environment.
- Whatever the data are available in the existing system, it will be grouped together with the machine or node of the cluster into a single cluster so that load could be optimized.

- Apache Mesos is totally opposite to the virtualizations because in virtualization one physical resource is going to be shared with multiple virtual resource but in Mesos multiple, physical resource are grouped together to form a single virtual machine.



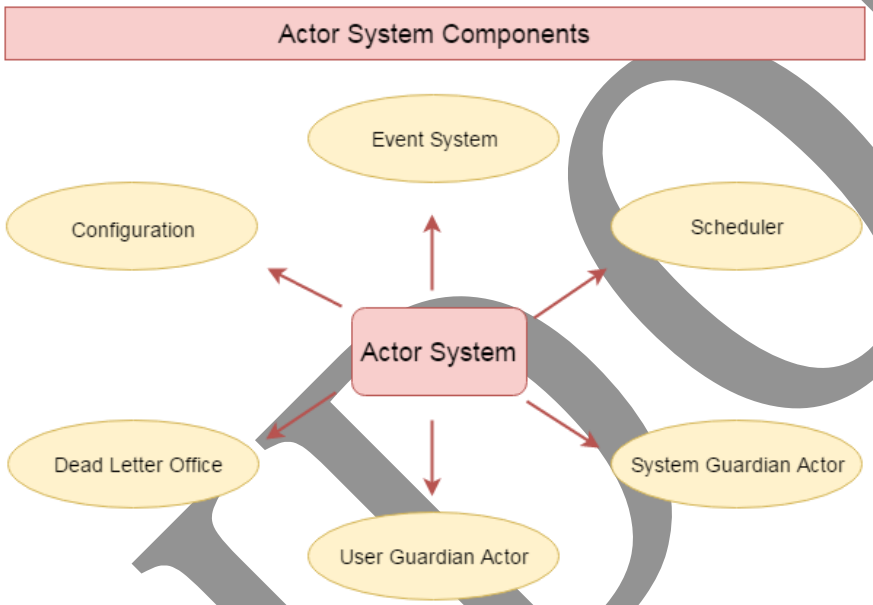
7. **Akka:**

- Akka is an actor-based message driven runtime for running concurrency, elasticity, and resilience processes.

- The actor can be controlled and limited to perform the intended task only. Akka is an open source library or toolkit.

- Apache Akka is used to create distributed and fault tolerance and it can be integrated to this library into the java virtual machine or JVM to support the language.

- Akka could be integrated with the Scala programming language and it is written in the Scala and it help us and developers to deal with external locking and threat management.



- The Actor is an entity which communicate with another actor by passing the massage to each other and it has its own state and behavior.

- In object-oriented programming like that everything is an object same thing is here like Akka is an actor based driven system.

- In other way we can say that Actor is an object that include and incapsulate it states and behavior.

8. **Cassandra**:

- Apache Cassandra is an open source distributed database system that is designed for storing and managing large amount of data across commodity servers.

- Cassandra can be used for both real time operational data store for online transaction data application.

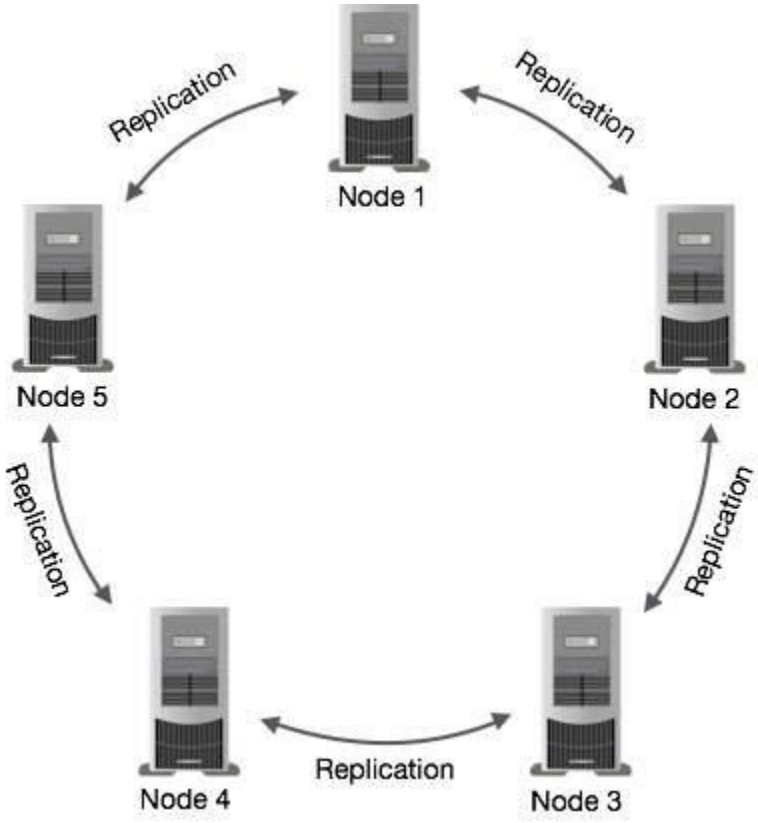
- Cassandra is designed for to have peer to peer process continues nodes instead of master or named nodes to ensure that there should not be any single point of failure.

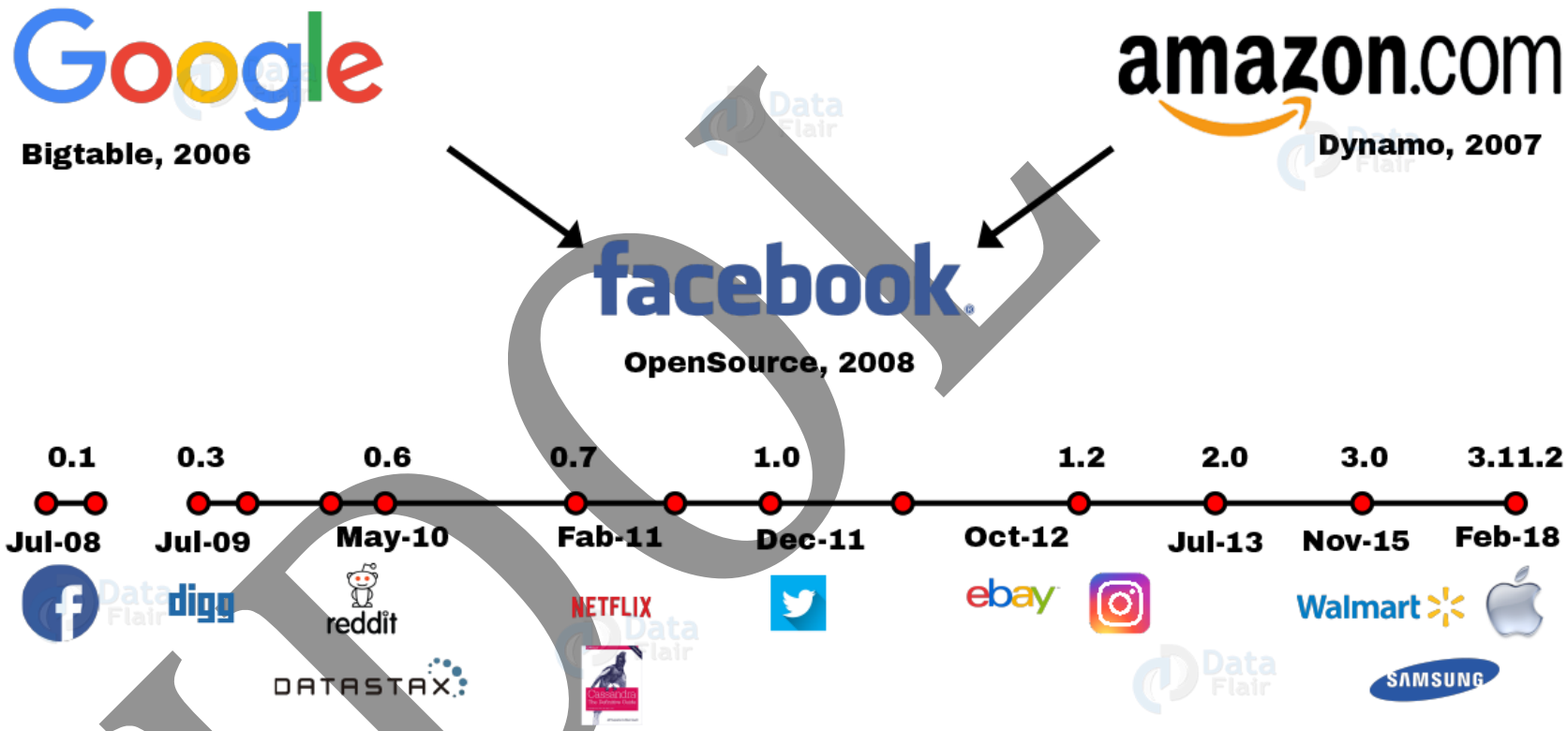
- Apache Cassandra is a highly scalable, high performance distributed database designed to handle large amounts of data and it is type of NoSQL Database.

- A NoSQL database is a database that provide mechanism to store and retrieve the data from the database than relational database.
- NoSQL database uses different data structure compared to relational database and it support very simple query language.

- NoSQL Database has no Schema and does not provide the data transaction.

- Cassandra is being used by some of the most important companies like Facebook, Twitter, Cisco, Netflix and much more.

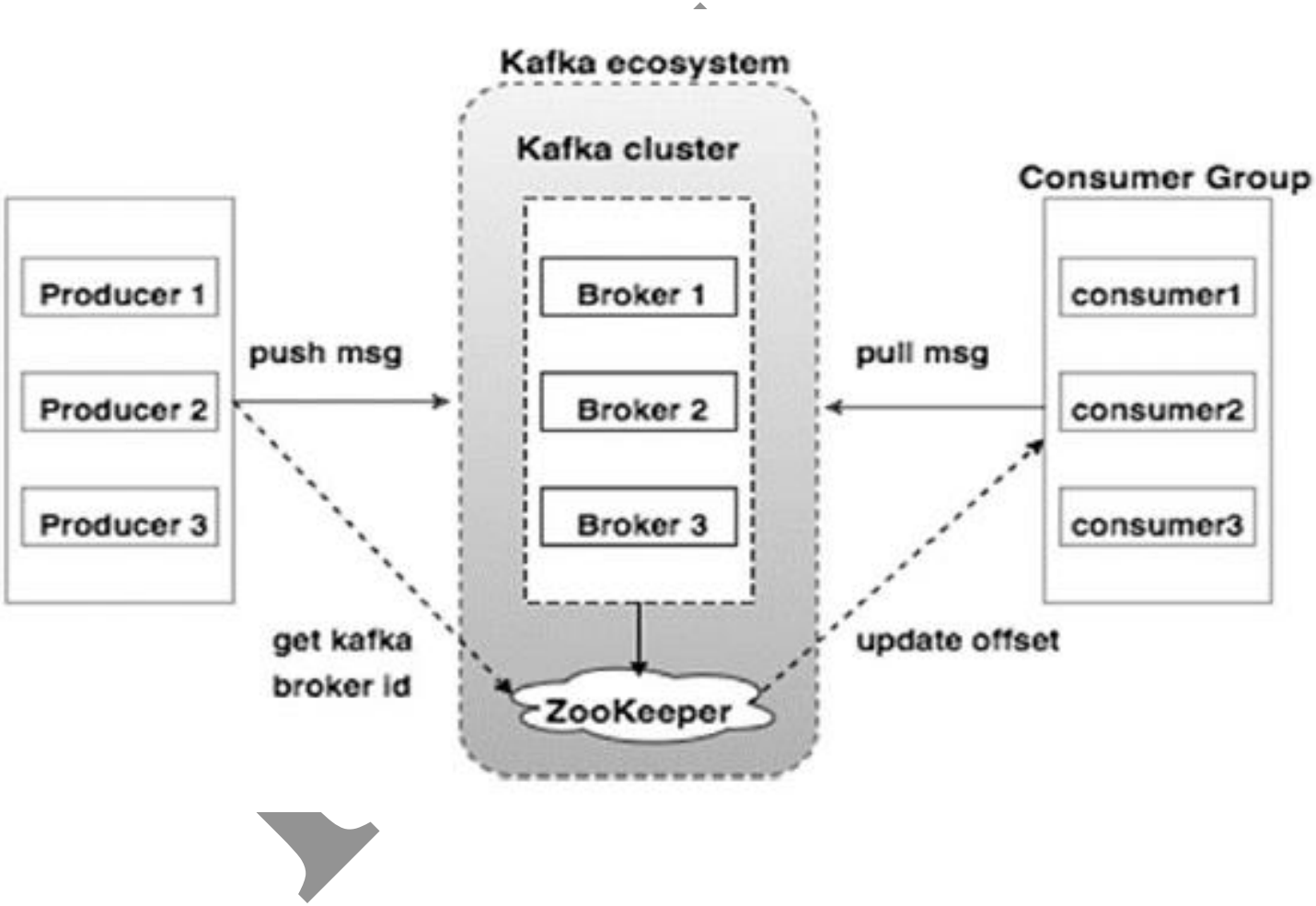




- The component of the Cassandra is Node, Data Center, Commit Log, Cluster, Mem-Table, SS Table, Bloom Filter.

9. **Kafka:**

- Kafka is a high messaging backbone that enables communication between data processing entities and Kafka is written in java and Scala language.
- Apache Kafka is highly scalable, reliable, fast, and distributed system. Kafka is suitable for both offline and online message consumption.
- Kafka messages are stored on the hard disk and replicated within the cluster to prevent the data loss.
- Kafka is distributed, partitioned, replicated and fault tolerant which make it more reliable.
- Kafka messaging system scales easily without down time which make it more scalable. Kafka has high throughput for both publishing and subscribing messages, and it can store data up to TB.
- Kafka has unique platform for handling the real time data for feedback and it can handle large amount of data to diverse consumers.
- Kafka persists all data to the disk, which essentially means that all the writes go to the page cache of the OS (RAM). This makes it very efficient to transfer data from page cache to a network socket.


Kafka ecosystem
Kafka cluster
Consumer Group
Producer 1
Producer 2
Producer 3
push msg
Broker 1
Broker 2
Broker 3
pull msg
consumer1
consumer2
consumer3
get kafka
broker id
ZooKeeper
update offset

Different Programming languages in data science processing:

1. Elastic Search:

- Elastic Search is an open source, distributed search and analytical engine designed.
- Scalability mean that it can scale any point of view, reliability means that it should be trustable, stress free management.

- Combine the power of search and power of analytics so that developers, programmers, data engineer and data scientist could work with very smoothly with structures, un-structured, and time series data.

- Elastics search is an open source that means anyone can download and work with it and it is developed by using java language and most of the big organization are using this search engine for their need.

- It enables the user to expand the very large amount of data at very high speed.

- It is used for the replacement of the documents and data store in the database like mongo dB etc.

- Elastic search is one of the popular search engines and mostly used by the recent organization like google, stack Overflow, GitHub and much more.

- Elastic Search is an open source search engine and is available under the hive version 2.0.

2. R Language:

- R is a programming language and it is used for statistical computing and graphics purpose.
- R Language are used by data engineer, data scientist, statisticians, and data miners for developing the software and performing data analytics.
- There is core requirement before learning the R Language and some depend on library and package concept that you should know about it and know how to work upon it easily.
- The related packages are of R Language is sqldf, forecast, dplyr, stringer, lubridate, ggplot2, reshape etc.
- R language is freely available, and it comes with General Public License and it supports many of the platform like windows, Linux/Unix, Mac.
- R language has built in capability to support and can be implemented and integrated with procedural language written in c, c++, java, .Net, and python.
- R Language has capacity and potential for handling data and data storage.

3 Scala:

- Scala is a general-purpose programming language and it support functional programming and a strong type statics type system.
- Most of the data science project and framework are build by using the Scala programming language because it has so many capabilities and potential to work with it.
- Scala integrate the feature of object-oriented language and its function because Scala can be written in java, c++, python language.
- Types and behavior of objects are described by the class and class can be extended by another class by using its properties.
- Scala support the high-level functions and function can be called by another function by using and written the function in a code.
- Once the Scala program is ready to compile and executive, Scala program convert into the byte code (machine understandable language) with the help of java virtual machine.
- This means that Scala and Java Programs can be complied and executed by using the JVM. So, we can easily say that it can be moved from Java to Scala and vice-versa.

- Scala enables you to use and import all the class, object and its behavior and function because Scala and java run with the help of Java Virtual Machine and you can create its own class and object.

- Instead of writing thousands of codes, Scala reduce the code in such a way that it can be readable, reliable, portable and reduce lines of code and support the developer and programmers to type the code in easy way.

4 Python:

- Python is a programming language and it can used on a server to create web application.

- Python can be used for web development, mathematics, software development and it is used to connect the database and create and modify the data.

- Python can handle the large amount of data and it is capable and potential to perform the complex task on data.

- Python is reliable, portable, and flexible to work on different platform like windows, mac and Linux etc.
- As compare to the other programming language , python is easy to learn and can perform the simple as well as complex task and it has the capabilities to reduce the line of code and help the programmer and developers to work with is easily friendly manner.

- Python support object-oriented programming language, functional and work with structure data.

- Python support dynamics data type and can be supported by dynamics type checking.

- Python is an interpreter and it has the philosophy and statements that it reduces the line of code.

Vermeulen-Krennwallner-Hillman-Clark

- Vermeulen-krennwallner-Hillman-Clark is small group like VKHCG and has a small size international company and it consist of 4 subcomponent 1. Vermeulen PLC, 2. Krennwallner AG, 3. Hillman Ltd 4. Clark Ltd.

1. Vermeulen PLC:

- Vermeulen PLC is a data processing company which process all the data within the group companies.

- This is the company for which we hire most of the data engineer and data scientist to work with it.

- This company supplies data science tool, Network, server and communication system , internal and external web sites, decision science and process automation.

2. Krennwallner AG:

- This is an advertising and media company which prepares advertising and media information which is required for the customers.

- Krennwallner supplies advertising on billboards, make Advertising and content management for online delivery etc.

- By using the number of record and data which are available on internet for media stream, it takes the data from there and make an analysis on this according to that it searches which of the media stream are watched by customer, how many time and which is most watchable content on internet.

- By using the survey, it specifies and choose content for the billboards, make and understand how many times customer are visited for which channel.

3. Hillman Ltd:

- This is logistic and supply chain company and it is used to supply the data around the worldwide for the business purpose.

- This include client warehouse, international shipping, home – to – home logistics.

4. Clark Ltd:

- This is the financial company which process all financial data which is required for financial purpose includes Support Money, Venture Capital planning and allow to put your money on share market.

Scala:

- Scala is a general-purpose programming language and it support functional programming and a strong type statics type system.
- Most of the data science project and framework are built by using the Scala programming language because it has so many capabilities and potential to work with it.
- Scala integrate the feature of object-oriented language and its function because Scala can be written in java, C++, python language.
- Types and behavior of objects are described by the class and class can be extended by another class by using its properties.
- Scala support the high-level functions and function can be called by another function by using and written the function in a code.

Apache Spark:

- Apache Spark is an open source clustering computing framework. The word open source means it is freely available on internet and just go on internet and type apache spark and you will get freely source code are available there, you can download and according to your wish.

- Apache Spark was developed at AMP Lab of university of California, Berkeley and after that all the code and data was donated to Apache Software Foundation for keep doing changes over a time and make it more effective , reliable, portable that will run all the platform.

    - Apache Spark provide an interface for the programmer and developer to directly interact with the system and make data parallel and compatible with data scientist and data engineer.

    - Apache Spark has the capabilities and potential, process all types and variety of data with repositories including Hadoop distributed file system, NoSQL database as well as apache spark.

    - IBM are hiring most of the data scientist and data engineer to whom has more knowledge and information about apache spark project so that innovation could be perform an easy way and will come up with more feature and changing.

Apache Mesos:

- Apache Mesos is an open source cluster manager and it was developed by the universities of California, Berkeley.

- It provides all the required resource for the isolation and sharing purpose across all the distributed application.

- The software we are using for Mesos, provide resources sharing in a fine-grained manner so that improvement can be done with the help of this.

- Mesosphere enterprises DC/OS is the enterprise version of Mesos and this run specially on Kafka, Cassandra, spark and Akka.

Akka:

- Akka is an actor-based message driven runtime for running concurrency, elasticity, and resilience processes.

- The actor can be controlled and limited to perform the intended task only. Akka is an open source library or toolkit.

- Apache Akka is used to create distributed and fault tolerant and it can be integrated to the library into the java virtual machine or JVM to support the language.

- Akka could be integrated with the Scala programming language and it is written in the Scala and it help us and developers to deal with external locking and threat management.

Apache Cassandra:

- Apache Cassandra is an open source distributed database system that is designed for storing and managing large amount of data across commodity servers.

- Cassandra can be used for both real time operational data store for online transaction data application.

- Cassandra is designed to have peer to peer process continuing nodes instead of master or named nodes to ensure that there should not be any single point of failure.

- Apache Cassandra is a highly scalable, high performance distributed database designed to handle large amounts of data and it is type of NoSQL Database.

- A NoSQL database is a database that provide mechanism to store and retrieve the data from the database than relational database.

Kafka:

- Kafka is a high messaging backbone that enables communication between data processing entities and Kafka is written in java and Scala language.

- Apache Kafka is highly scalable, reliable, fast, and distributed system. Kafka is suitable for both offline and online message consumption.

- Kafka messages are stored on the hard disk and replicated within the cluster to prevent the data loss.

- Kafka is distributed, partitioned, replicated and fault tolerant which make it more reliable.

- Kafka messaging system scales easily without down time which make it more scalable. Kafka has high throughput for both publishing and subscribing messages, and it can store data up to TB.

Python:

- Python is a programming language and it can used on a server to create web application.

- Python can be used for web development, mathematics, software development and it is used to connect the database and create and modify the data.

- Python can handle the large amount of data and it is capable to perform the complex task on data.

- Python is reliable, portable, and flexible to work on different platform like windows, mac, and Linux etc.

- Python can be installed on all the operating system example windows, Linux and mac operating system and it can work on all these platforms for better understanding and learning purpose.

- You can earn much more knowledge by installing and working all three platform for data science and data engineering.

- To working and installing the data science required package in python, **Ubuntu** run the following command below:

- sudo apt-get install python3 python3-pip python3-setuptools

- To working and installing the data science required package in python, **Linux** run the following command below:

  - sudo yum install python3 python3-pip python3-setuptools
  - To work and install the data science required package in python, **Windows** run the following command below:

  - https://www.python.org/downloads/

  - **Python Libraries:**

  - Python library is a collection of functions and methods that allows you to perform many actions without writing your code.

  - **Pandas:**

  - Pandas stands for panel data and it is the core library for data manipulation and data analysis.

- It consists of single and multidimensional data for data analysis.
- How to install pandas in **UBUNTU** by using the following commands:
- sudo apt-get install python-pandas
- How to install pandas in **LINUX** by using the following commands:
- yum install python-pandas
- How to install pandas in **WINDOWS** by using the following commands:
- pip install pandas

Matplotlib:

- Matplotlib is used for data visualization and is one of the most important packages of python.
- Matplotlib is used to display and visualize the 2D data and it is written in python.
- It can be used for python, Jupiter, notebook and web application server also.
- How to install Matplotlib Library for **UBUNTU** in python by using the following command:

- sudo apt-get install python-matplotlib
- How to install Matplotlib Library for **LINUX** in python by using the following command:
- Sudo yum install python-matplotlib
- How to install Matplotlib Library for **WINDOWS** in python by using the following command:
- pip install matplotlib

NumPy:

- NumPy is the fundamental package of python language and is used for the numerical purpose.
- NumPy is used with the SciPy and Matplotlib package of python and it is freely available on internet.

SymPy:

- Sympy is a python library and which is used for symbolic mathematics and it can be used with complex algebra formula.

R:

- R is a programming language and it is used for statistical computing and graphics purpose.

- R Language is used by data engineer, data scientist, statisticians, and data miners for developing the software and performing data analytics.

- There is core requirement before learning the R Language and some depend on library and package concept that you should know about it and know how to work upon it easily.

- The related packages are of R Language is sqldf, forecast, dplyr, stringer, lubridate, ggplot2, reshape etc.

Unit 2

Chapter 3. Three Management Layers

Unit Structure

3.0 Objectives

- The objective is to explain in detail the core operations of the Three Management Layers i.e. Operational Management Layer, Audit, Balance, and Control Layer & the Functional Layers

3.1 Introduction

- The Three Management Layers are a very important part of the framework.
- They watch the overall operations in the data science ecosystem and make sure that things are happening as per plan.

- If things are not going as per plan then it has contingency actions in place for recovery or cleanup.

3.2 Operational Management Layer

- Operations management is one of the areas inside the ecosystem responsible for designing and controlling the process chains of a data science environment.
- This layer is the center for complete processing capability in the data science ecosystem.
- This layer stores what you want to process along with every processing schedule and workflow for the entire ecosystem.
- This area enables us to see an integrated view of the entire ecosystem. It reports the status each and every processing in the ecosystem. This is where we plan our data science processing pipelines.
- We record the following in the operations management layer:
    - Definition and Management of Data Processing stream
    - Eco system Parameters
    - Overall Process Scheduling
    - Overall Process Monitoring
    - Overall Communication
    - Overall Alerting

- **Definition and Management of Data Processing stream**
    - The processing-stream definitions are the building block of the data science environment.
    - This section of the ecosystem stores all currently active processing scripts.
    - Management refers to Definition management, it describes the workflow of the scripts throughout the ecosystem, it manages the correct execution order according to the workflow designed by the data scientist.
- **Eco system Parameters**
    - The processing parameters are stored in this section, here it is made sure that a single location is made available for all the system parameters.

- In any production system, for every existing customer, all the parameters can be placed together in a single location and then calls could be made to this location every time the parameters are needed.
- Two ways to maintain a central location for all parameters are:
  1. Having a text file which we can import into every processing script.
  2. A standard parameter setup script that defines a parameter database which we can import into every processing script.
- Example: an ecosystem setup phase

```
if sys.platform == 'linux':
Base=os.path.expanduser('~') + '/VKHCG'
else:
Base='C:/VKHCG'
################################################################
print('################################')
print('Working Base :',Base, ' using ', sys.platform)
print('################################')
################################################################
sFileDir=Base + '/01-Vermeulen/01-Retrieve/01-EDS/02-Python'
if not os.path.exists(sFileDir):
os.makedirs(sFileDir)
################################################################
sFileName=Base + '/01-Vermeulen/00-RawData/Country_Currency.xlsx'
```

- **Overall Process Scheduling**
  - Along with other things the scheduling plan is stored in this section, it enables a centralized control and visibility of the complete scheduling plan for the entire system.
  - One of the scheduling methods is a Drum-Buffer-Rope method.

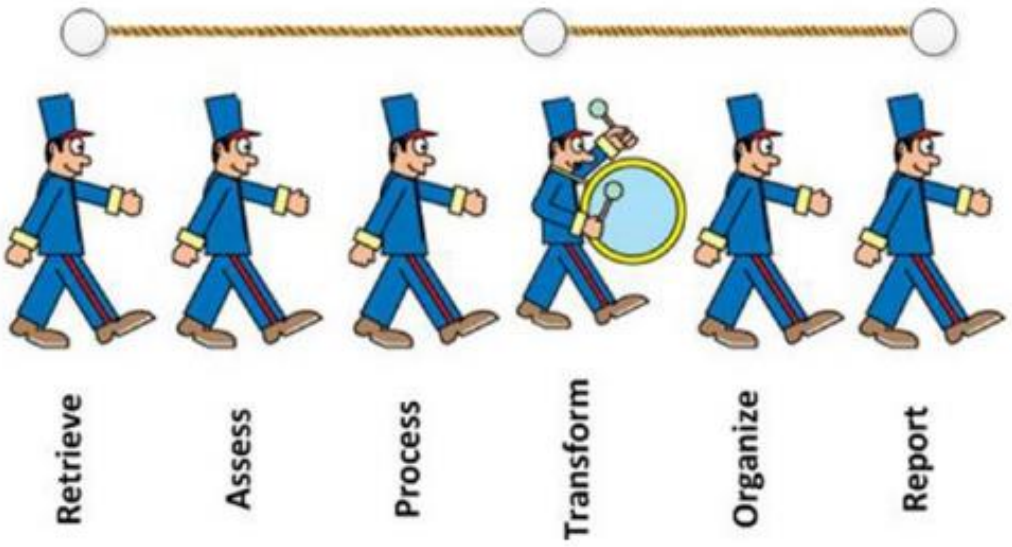


Fig: Original Drum-Buffer-Rope use

- **The Drum-buffer-rope Method**
    - It is a standard practice to identify the slowest process among all.
    - Once identified it is then used to control the speed of the complete pipeline
    - This is done by tying or binding the remaining processes of the pipeline to this process.
    - The method implies that
        - the “drum” is placed at the slow part of the pipeline, to give the processing pace,
        - the “rope” is attached to all the processes from beginning to end of the pipeline, this makes sure that no processing is done that is not attached to the drum.
    - This approach ensures that all the processes in the pipeline complete more efficiently, as no process is entering or leaving the process pipeline without been recorded by the drum’s beat.
- **Process Monitoring**
    - The central monitoring process makes sure that there is a single unified view of the complete system.
    - We should always ensure that the monitoring of our data science is being done from a single point.
    - With no central monitoring running different data science processes on the same ecosystem will make managing a difficult task.

- **Overall Communication**
    - The Operations management handles all communication from the system, it makes sure that any activities that are happening are communicated to the system.
    - To make sure that we have all our data science processes tracked we may use a complex communication process.
- **Overall Alerting**

- The alerting section of the Operations management layer uses communications to inform the correct status of the complete system to the correct person, at the correct time.

3.3 Audit, Balance, and Control Layer

- Any process currently under executing is controlled by the audit, balance, and control layer.
- It is this layer that has the engine that makes sure that every processing request is completed by the ecosystem according to the plan.
- This is the only area where you can observe which processes are is currently running within your data scientist environment.
- It records the following information:
    - Process-execution statistics
    - Balancing and controls
    - Rejects- and error-handling
    - Fault codes management

3.3.1 Audit

- An audit refers to an examination of the ecosystem that is systematic and independent
- This sublayer records which processes are running at any given specific point within the ecosystem.
- Data scientists and engineers use this information collected to better understand and plan future improvements to the processing to be done.
- the audit in the data science ecosystem, contain of a series of observers which record prespecified processing indicators related to the ecosystem.
- The following are good indicators for audit purposes:
    - Built-in Logging
        - Debug Watcher
        - Information Watcher
        - Warning Watcher
        - Error Watcher
        - Fatal Watcher
    - Basic Logging
        - Process Tracking
        - Data Provenance
        - Data Lineage

- **Built-in Logging**
    - It is always a good thing to design our logging in an organized prespecified location, this ensures that we capture every relevant log entry in one location.
    - Changing the internal or built-in logging process of the data science tools should be avoid as this complicate any future upgrades complex and will prove very costly to correct.
    - A built-in Logging mechanism along with a cause-and-effect analysis system allows you to handle more than 95% of all issues that can rise in the ecosystem.
    - Since there are five layers it would be a good practice to have five watchers for each logging locations independent of each other as described below:
- **Debug Watcher**
    - This level of logging is the maximum worded logging level.
    - Any debug logs if discovered in the ecosystem, it should raise an alarm, indicating that the tool is using some precise processing cycles to perform the necessary low-level debugging.
- **Information Watcher**
    - The information watcher logs information that is beneficial to the running and management of a system.
    - It is advised that these logs be piped to the central Audit, Balance, and Control data store of the ecosystem.
- **Warning Watcher**
    - Warning is usually used for exceptions that are handled or other important log events.
    - Usually this means that the issue was handled by the tool and also took corrective action for recovery.
    - It is advised that these logs be piped to the central Audit, Balance, and Control data store of the ecosystem.
- **Error Watcher**

    - An Error logs all unhandled exceptions in the data science tool.
    - An Error is a state of the system. This state is not good for the overall processing, since it normally means that a specific step did not complete as expected.
    - In case of an error the ecosystem should handle the issue and take the necessary corrective action for recovery.
    - It is advised that these logs be piped to the central Audit, Balance, and Control data store of the ecosystem.
- **Fatal Watcher**
    - Fatal is a state reserved for special exceptions or conditions for which it is mandatory that the event causing this state be identified immediately.
    - This state is not good for the overall processing, since it normally means that a specific step did not complete as expected.
    - In case of an fatal error the ecosystem should handle the issue and take the necessary corrective action for recovery.
    - It is advised that these logs be piped to the central Audit, Balance, and Control data store of the ecosystem.
- **Basic Logging**: Every time a process is executed this logging allows you to log everything that occurs to a central file.
- **Process Tracking**
    - For Process Tracking it is advised to create a tool that will perform a controlled, systematic and independent examination of the process for the hardware logging.
    - There may be numerous server-based software that monitors system hardware related parameters like voltage, fan speeds, temperature sensors and clock speeds of a computer system.
    - It is advised to use the tool which your customer and you bot are most comfortable to work with.
    - It is also advised that logs generated should be used for cause-and-effect analysis system.
- **Data Provenance**

- For every data entity all the transformations in the system should be tracked so that a record can be generated for activity.
- This ensures two things: 1. that we can reproduce the data, if required, in the future and 2. That we can supply a detailed history of the data's source in the system throughout its transformation.

- **Data Lineage**
  - This involves keeping records of every change whenever it happens to every individual data value in the data lake.
  - This help us to figure out the exact value of any data item in the past.
  - This is normally accomplished by enforcing a valid-from and valid-to audit entry for every data item in the data lake.

3.3.2 Balance

- The balance sublayer has the responsibility to make sure that the data science environment is balanced between the available processing capability against the required processing capability or has the ability to upgrade processing capability during periods of extreme processing.
- In such cases the on-demand processing capability of a cloud environment becomes highly desirable.

3.3.3 Control

- The execution of the current active data science processes is controlled by the control sublayer.
- The control elements of the control sublayer are a combination of:
  - the control element available in the Data Science Technology Stack's tools and
  - a custom interface to control the overarching work.
- When processing pipeline encounters an error, the control sublayer attempts a recovery as per our prespecified requirements else if recovery does not work out it will schedule a cleanup utility to undo the error.
- The cause-and-effect analysis system is the core data source for the distributed control system in the ecosystem.

3.4 Yoke Solution

- The yoke solution is a custom design
- Apache Kafka is developed as an open source stream processing platform. Its function is to deliver a platform that is unified, has high-throughput and low-latency for handling real-time data feeds.
- Kafka provides a publish-subscribe solution that can
- handle all activity-stream data and processing. The Kafka environment enables you to send
- messages between producers and consumers that enable you to transfer control between different parts of your ecosystem while ensuring a stable process.

3.4.1 Producer

- The producer is the part of the system that generates the requests for data science processing, by creating structures messages for each type of data science process it requires.
- The producer is the end point of the pipeline that loads messages into Kafka.

3.4.2 Consumer

- The consumer is the part of the process that takes in messages and organizes them for processing by the data science tools.
- The consumer is the end point of the pipeline that offloads the messages from Kafka.

3.4.3 Directed Acyclic Graph Scheduling

- This solution uses a combination of graph theory and publish-subscribe stream data processing to enable scheduling.
- You can use the Python NetworkX library to resolve any conflicts, by simply formulating the graph into a specific point before or after you send or receive messages via Kafka.
- That way, you ensure an effective and an efficient processing pipeline

DAG code

3.4.4 Cause-and-Effect Analysis System

- The cause-and-effect analysis system is the part of the ecosystem that

- collects all the logs, schedules, and other ecosystem-related information and
- enables data scientists to evaluate the quality of their system.

3.5 Functional Layer

- The functional layer of the data science ecosystem is the largest and most essential layer for
- programming and modeling. Any data science project must have processing elements in this

3.6 Data Science Process

- Following are the five fundamental data science process steps.
    - Begin process by asking a What if question
    - Attempt to guess at a probably potential pattern
    - Create a hypothesis by putting together observations
    - Verify the hypothesis using real-world evidence
    - Promptly and regularly collaborate with subject matter experts and customers as and when you gain insights
- **Begin process by asking a What if question –** Decide what you want to know, even if it is only the subset of the data lake you want to use for your data science, which is a good start.
- **Create a hypothesis by putting together observations –** Use your experience or insights to guess a pattern you want to discover, to uncover additional insights from the data you already have
- **Gather Observations and Use Them to Produce a Hypothesis –** A hypothesis, it is a proposed explanation, prepared on the basis of limited evidence, as a starting point for further investigation.
- **Verify the hypothesis using real-world evidence**- Now, we verify our hypothesis by comparing it with real-world evidence
- **Promptly and regularly collaborate with subject matter experts and customers as and when you gain insights –** Things that are communicated with experts may include technical aspects like workflows or more specifically data formats & data schemas.

- **Data structures in the functional layer of the ecosystem are:**
    - **Data schemas and data formats**: Functional data schemas and data formats deploy onto the data lake's raw data, to perform the required schema-on-query via the functional layer.
    - **Data models:** These form the basis for future processing to enhance the processing capabilities of the data lake, by storing already processed data sources for future use by other processes against the data lake.
    - **Processing algorithms:** The functional processing is performed via a series of well-designed algorithms across the processing chain.
    - **Provisioning of infrastructure:** The functional infrastructure provision enables the framework to add processing capability to the ecosystem, using technology such as Apache Mesos, which enables the dynamic previsioning of processing work cells.

- The processing algorithms and data models are spread across six super steps for processing the data lake.

1. **Retrieve**: This super step contains all the processing chains for retrieving data from the raw data lake into a more structured format.

2. **Assess:** This super step contains all the processing chains for quality assurance and additional data enhancements.

3. **Process:** This super step contains all the processing chains for building the data vault.

4. **Transform:** This super step contains all the processing chains for building the data warehouse from the core data vault.

5. **Organize:** This super step contains all the processing chains for building the data marts from the core data warehouse.

6. **Report:** This super step contains all the processing chains for building virtualization and reporting of the actionable knowledge.

3.7 Review Question

1. Explain in detail the function of Operational Management Layer
2. Give an overview of the Drum-buffer-rope Method
3. Give an overview of the functions of Audit, Balance, and Control Layer
4. Explain the different ways of implementing the Built-in Logging in the Audit phase.
5. Explain the different ways of implementing the Basic Logging in the Audit phase.
6. Explain Directed Acyclic Graph Scheduling
7. List & Explain the data structures in the functional layer of the ecosystem
8. Explain the fundamental data science process steps
9. List the super steps for processing the data lake.

3.8 References

Andreas François Vermeulen, “Practical Data Science - A Guide to Building the Technology Stack for Turning Data Lakes into Business Assets”

Unit 2
Chapter 2. Retrieve Super step

Unit Structure

4.0 Objectives

- The objective of this chapter is to explain in detail the core operations in the Retrieve Super step.
- This chapter explains important guidelines which if followed will prevent the data lake turning into a data swamp.
- This chapter explains another important example related to shipping terminology called Incoterm

- Finally this chapter explains the different possible data sources to retrieve data from.

4.1 Introduction

- The Retrieve super step is a practical method for importing a data lake consisting of different external data sources completely into the processing ecosystem.
- The Retrieve super step is the first contact between your data science and the source systems.
- The successful retrieval of the data is a major stepping-stone to ensuring that you are performing good data science.
- Data lineage delivers the audit trail of the data elements at the lowest granular level, to ensure full data governance.
- Data governance supports metadata management for system guidelines, processing strategies, policies formulation, and implementation of processing.
- Data quality and master data management helps to enrich the data lineage with more business values, if you provide complete data source metadata.
- The Retrieve super step supports the edge of the ecosystem, where your data science makes direct contact with the outside data world. I will recommend a current set of data structures that you can use to handle the deluge of data you will need to process to uncover critical business knowledge.

4.2 Data Lakes

- A company's data lake covers all data that your business is authorized to process, to attain an improved profitability of your business's core accomplishments.
- The data lake is the complete data world your company interacts with during its business life span.
- In simple terms, if you generate data or consume data to perform your business tasks, that data is in your company's data lake.

- Just as a lake needs rivers and streams to feed it, the data lake will consume an unavoidable deluge of data sources from upstream and deliver it to downstream partners

4.3 Data Swamps

- Data swamps are simply data lakes that are not managed.
- They are not to be feared. They need to be tamed.
- Following are four critical steps to avoid a data swamp.

    1. Start with Concrete Business Questions
    2. Data Quality
    3. Audit and Version Management
    4. Data Governance

4.3.1. Start with Concrete Business Questions

- Simply dumping a horde of data into a data lake, with no tangible purpose in mind, will result in a big business risk.
- The data lake must be enabled to collect the data required to answer your business questions.
- It is suggested to perform a comprehensive analysis of the entire set of data you have and then apply a metadata classification for the data, stating full data lineage for allowing it into the data lake.

4.3.2. Data Quality

- More data points do not mean that data quality is less relevant.
- Data quality can cause the invalidation of a complete data set, if not dealt with correctly.

4.3.3. Audit and Version Management

- You must always report the following:
    - Who used the process?
    - When was it used?
    - Which version of code was used?

4.3.4. Data Governance

- The role of data governance, data access, and data security does not go away with the volume of data in the data lake.
- It simply collects together into a worse problem, if not managed.
- Data Governance can be implemented in the following ways:
    - Data Source Catalog
    - Business Glossary
    - Analytical Model Usage

4.3.4.1. Data Source Catalog

- Metadata that link ingested data-to-data sources are a must-have for any data lake.
- Data processing should include the following rules:
    - **Unique data catalog number**
        - use YYYYMMDD/ NNNNNN/NNN.
        - E.g. 20171230/000000001/001 for data first registered into the metadata registers on December 30, 2017, as data source 1 of data type 1.
        - This is a critical requirement.
    - **Short description (It should be under 100 characters)**
        - Country codes and country names
        - Ex. ISO 3166 defines Country Codes as per United Nations Sources
    - **Long description (It should kept as complete as possible)**
        - Country codes and country names used by your organization as standard for country entries
    - **Contact information for external data source**
        - ISO 3166-1:2013 code lists from www.iso.org/iso-3166-country-codes.html
    - **Expected frequency**

        - Irregular i.e., no fixed frequency, also known as ad hoc, every minute, hourly, daily, weekly, monthly, or yearly.
        - Other options are near-real-time, every 5 seconds, every minute, hourly, daily, weekly, monthly, or yearly.
    - **Internal business purpose**
        - Validate country codes and names.

4.3.4.2. Business Glossary

- The business glossary maps the data-source fields and classifies them into respective lines of business.
- This glossary is a must-have for any good data lake.
- The business glossary records the data sources ready for the retrieve processing to load the data.
- Create a data-mapping registry with the following information:
    - **Unique data catalog number:** use YYYYMMDD/ NNNNNN/NNN.
    - **Unique data mapping number:** use NNNNNNN/ NNNNNNNNN. E.g., 0000001/000000001 for field 1 mapped to
    - internal field 1
    - **External data source field name:** States the field as found in the raw data source
    - **External data source field type:** Records the full set of the field's data types when loading the data lake
    - **Internal data source field name:** Records every internal data field name to use once loaded from the data lake
    - **Internal data source field type:** Records the full set of the field's types to use internally once loaded
    - **Timestamp of last verification of the data mapping:** use YYYYMMDD-HHMMSS-SSS that supports timestamp down to a thousandth of a second.

4.3.4.3. Analytical Model Usage

- Data tagged in respective analytical models define the profile of the data that requires loading and guides the data scientist to what additional processing is required.
- The following data analytical models should be executed on every data set in the data lake by default.
    - Data Field Name Verification
    - Unique Identifier of Each Data Entry
    - Data Type of Each Data Column
    - Histograms of Each Column
    - Minimum Value
    - Maximum Value
    - Mean
    - Median
    - Mode
    - Range
    - Quartiles
    - Standard Deviation
    - Skewness
    - Missing or Unknown Values
    - Data Pattern
- The models can be applied using R or Python, we will use R
- The data set used to demonstrate the models is INPUT_DATA.csv
- **Data Field Name Verification**
    - This is used to validate and verify the data field's names in the retrieve processing in an easy manner.
    - Example

```
library(table)

set_tidy_names(INPUT_DATA, syntactic = TRUE, quiet = FALSE)
```

    - Reveals field names that are not easy to use
- **Unique Identifier of Each Data Entry**
    - Allocate a unique identifier within the system that is independent of the given file name.
    - This ensures that the system can handle different files from different paths and keep track of all data entries in an effective manner.

- Then allocate a unique identifier for each record or data element in the files that are retrieved.

- Example: To add the unique identifier, run the following command

```
INPUT_DATA_with_ID =
Row_ID_to_column(INPUT_DATA_FIX, var = "Row_ID")
```

- **Data Type of Each Data Column**

  - Determine the best data type for each column, to assist you in completing the business glossary, to ensure that you record the correct import processing rules.

  - Example: To find datatype of each column

```
sapply(INPUT_DATA_with_ID, typeof)
```

- **Histograms of Each Column**

  - I always generate a histogram across every column, to determine the spread of the data value.

  - Example: to compute histogram

```
library(data.table)

country_histogram=data.table(Country=unique(INPUT_DATA_with_ID[is.na
(INPUT_DATA_with_ID ['Country']) == 0, ]$Country))
```

- **Minimum Value**

  - Determine the minimum value in a specific column.

  - Example: find minimum value

```
min(country_histogram$Country)
or
sapply(country_histogram[,'Country'], min, na.rm=TRUE)
```

- **Maximum Value**

- Determine the maximum value in a specific column.
- Example: find maximum value

```
max(country_histogram$Country)
or
sapply(country_histogram[,'Country'], max, na.rm=TRUE)
```

- **Mean**
  - If the column is numeric in nature, determine the average value in a specific column.
  - Example: find mean of latitude

```
sapply(lattitue_histogram_with_id[,'Latitude'], mean, na.rm=TRUE)
```

- **Median**
  - Determine the value that splits the data set into two parts in a specific column.
  - Example: find median of latitude

```
sapply(lattitue_histogram_with_id[,'Latitude'], median, na.rm=TRUE)
```

- **Mode**
  - Determine the value that appears most in a specific column.
  - Example: Find mode for column country

```
INPUT_DATA_COUNTRY_FREQ =
data.table(with(INPUT_DATA_with_ID, table(Country)))
```

- **Range**
  - For numeric values, you determine the range of the values by taking the maximum value and subtracting the minimum value.

- Example: find range of latitude

```
sapply(lattitue_histogram_with_id[,'Latitude'], range, na.rm=TRUE
```

- **Quartiles**
  - These are the base values that divide a data set in quarters. This is done by sorting the data column first and then splitting it in groups of four equal parts.
  - Example: find quartile of latitude

```
sapply(lattitue_histogram_with_id[,'Latitude'], quantile,
na.rm=TRUE)
```

- **Standard Deviation**
  - the standard deviation is a measure of the amount of variation or dispersion of a set of values.
  - Example: find standard deviation of latitude

```
sapply(lattitue_histogram_with_id[,'Latitude'], sd, na.rm=TRUE)
```

- **Skewness**
  - Skewness describes the shape or profile of the distribution of the data in the column.
  - Example: find skewness of latitude

```
library(e1071)
skewness(lattitue_histogram_with_id$Latitude, na.rm = FALSE,
type = 2)
```

- **Missing or Unknown Values**
  - Identify if you have missing or unknown values in the data sets. Example: find missing value in country column

```
missing_country=data.table(Country=unique(INPUT_DATA_with_ID[is.na(I
NPUT_DATA_with_ID ['Country']) == 1, ]))
```

- **Data Pattern**
    - I have used the following process for years, to determine a pattern of the data values themselves.
    - Here is my standard version:
    - Replace all alphabet values with an uppercase case A, all numbers with an uppercase N, and replace any spaces with a lowercase letter band all other unknown characters with a lowercase u.
    - As a result, “Data Science 102” becomes "AAAAbAAAAAAAbNNNu.” This pattern creation is beneficial for designing any specific assess rules.

4.4 Training the Trainer Model

- To prevent a data swamp, it is essential that you train your team also. Data science is a team effort.
- People, process, and technology are the three cornerstones to ensure that data is curated and protected.
- You are responsible for your people; share the knowledge you acquire from this book. The process I teach you, you need to teach them. Alone, you cannot achieve success.
- Technology requires that you invest time to understand it fully. We are only at the dawn of major developments in the field of data engineering and data science.
- Remember: A big part of this process is to ensure that business users and data scientists understand the need to start small, have concrete questions in mind, and realize that there is work to do with all data to achieve success.

4.5 Shipping Terminologies

In this section we discuss two things : shipping terms and Incoterm 2010.

4.5.1 Shipping Terms

- These determine the rules of the shipment, the conditions under which it is made. Normally, these are stated on the shipping manifest.

- Following are the terms used:

    - **Seller -** The person/company sending the products on the shipping manifest is the seller. This is not a location but a legal entity sending the products.

    - **Carrier -** The person/company that physically carries the products on the shipping manifest is the carrier. Note that this is not a location but a legal entity transporting the products.

    - **Port -** A Portis any point from which you have to exit or enter a country. Normally, these are shipping ports or airports but can also include border crossings via road. Note that there are two ports in the complete process. This is important. There is a port of exit and a port of entry.

    - **Ship -** Ship is the general term for the physical transport method used for the goods. This can refer to a cargo ship, airplane, truck, or even person, but it must be identified by a unique allocation number.

    - **Terminal -** A terminalis the physical point at which the goods are handed off for the next phase of the physical shipping.

    - **Named Place -** This is the location where the ownership is legally changed from seller to buyer. This is a specific location in the overall process. Remember this point, as it causes many legal disputes in the logistics industry.

    - **Buyer -** The person/company receiving the products on the shipping manifest is the buyer. In our case, there will be warehouses, shops, and

customers. Note that this is not a location but a legal entity receiving the products.

4.5.2 Incoterm 2010

- Incoterm 2010 is a summary of the basic options, as determined and published by a standard board

- This option specifies which party has an obligation to pay if something happens to the product being shipped (i.e. if the product is damaged or destroyed in route before it reaches to the buyer)

- **EXW—Ex Works**

    - Here the seller will make the product or goods available at his premises or at another named place. This term EXW puts the minimum obligations on the seller of the product /item and maximum obligation on the buyer.

    - Here is the data science version: If I were to buy an item a local store and take it home, and the shop has shipped it EXW—Ex Works, the moment I pay at the register, the ownership is transferred to me. If anything happens to the book, I would have to pay to replace it.

- **FCA—Free Carrier**

    - In this condition, the seller is expected to deliver the product or goods, that are cleared for export, at a named place.

    - The data science version: : If I were to buy an item at an overseas duty-free shop and then pick it up at the duty-free desk before taking it home, and the shop has shipped it FCA— Free Carrier—to the duty-free desk, the moment I pay at the register, the ownership is transferred to me, but if anything happens to the book between the shop and the duty-free desk, the shop will have to pay.

- It is only once I pick it up at the desk that I will have to pay, if anything happens. So, the moment I take the book, the transaction becomes EXW, so I have to pay any necessary import duties on arrival in my home country

- **CPT—Carriage Paid To**

  - Under this term, the seller is expected to pay for the carriage of product or goods up to the named place of destination.

  - The moment the product or goods are delivered to the first carrier they are considered to be delivered, and the risk gets transferred to the buyer.

  - All the costs including origin costs, clearance of export and freight costs for carriage till the place of named destination have to be paid by the seller to the named place of destination. This is could be anything like the final destination like the buyer's facility, or a port of at the destination country. This has to be agreed upon by both seller and buyer in advance.

  - The data science version: : If I were to buy an item at an overseas store and then pick it up at the export desk before taking it home and the shop shipped it CPT—Carriage Paid To—the duty desk for free, the moment I pay at the register, the ownership is transferred to me, but if anything happens to the book between the shop and the duty desk of the shop, I will have to pay.

  - It is only once I have picked up the book at the desk that I have to pay if anything happens. So, the moment I take the book, the transaction becomes EXW, so I must pay any required export and import duties on arrival in my home country.

- **CIP - Carriage & Insurance Paid**

  - The seller has to get insurance for the goods for shipping the goods.

  - The data science version If I were to buy an item at an overseas store and then pick it up at the export desk before taking it home, and the shop has shipped it CPT—Carriage Paid To— to the duty desk for free, the moment I

pay at the register, the ownership is transferred to me. However, if anything happens to the book between the shop and the duty desk at the shop, I have to take out insurance to pay for the damage.

- It is only once I have picked it up at the desk that I have to pay if anything happens. So, the moment I take the book, it becomes EXW, so I have to pay any export and import duties on arrival in my home country. Note that insurance only covers that portion of the transaction between the shop and duty desk.

- **DAT—Delivered at a Terminal**

  - According to this term the seller has to deliver and unload the goods at a named terminal. The seller assumes all risks till the delivery at the destination and has to pay all incurred costs of transport including export fees, carriage, unloading from the main carrier at destination port, and destination port charges.

  - The terminal can be a port, airport, or inland freight interchange, but it must be a facility with the capability to receive the shipment. If the seller is not able to organize unloading, it should consider shipping under DAP terms instead. All charges after unloading (for example, import duty, taxes, customs and on-carriage costs) are to be borne by buyer.

  - The data science version. If I were to buy an item at an overseas store and then pick it up at a local store before taking it home, and the overseas shop shipped it—Delivered at Terminal (Local Shop)—the moment I pay at the register, the ownership is transferred to me.

  - However, if anything happens to the book between the payment and the pickup, the local shop pays. It is picked up only once at the local shop. I have to pay if anything happens. So, the moment I take it, the transaction becomes EXW, so I have to pay any import duties on arrival in my home.

- **DAP—Delivered at Place**

- Under this option the seller delivers the goods at a given place of destination. Here, the risk will pass from seller to buyer from destination point.

- Packaging cost at the origin has to be paid by the seller also all the legal formalities in the exporting country will be carried out by the seller at his own expense.

- Once the goods are delivered in the destination country the buyer has to pay for the customs clearance.

- Here is the data science version. If I were to buy 100 pieces of a particular item from an overseas web site and then pick up the copies at a local store before taking them home, and the shop shipped the copies DAP-Delivered At Place (Local Shop)— the moment I paid at the register, the ownership would be transferred to me. However, if anything happened to the item between the payment and the pickup, the web site owner pays. Once the 100 pieces are picked up at the local shop, I have to pay to unpack them at store. So, the moment I take the copies, the transaction becomes EXW, so I will have to pay costs after I take the copies.

- **DDP—Delivered Duty Paid**

  - Here the seller is responsible for the delivery of the products or goods to an agreed destination place in the country of the buyer. The seller has to pay for all expenses like packing at origin, delivering the goods to the destination, import duties and taxes, clearing customs etc.

  - The seller is not responsible for unloading. This term **DDP** will place the minimum obligations on the buyer and maximum obligations on the seller. Neither the risk nor responsibility is transferred to the buyer until delivery of the goods is completed at the named place of destination.

  - Here is the data science version. If I were to buy an item in quantity 100 at an overseas web site and then pick them up at a local store before taking them home, and the shop shipped DDP—Delivered Duty Paid (my home)—

the moment I pay at the till, the ownership is transferred to me. However, if anything were to happen to the items between the payment and the delivery at my house, the store must replace the items as the term covers the delivery to my house.

4.6 Other Data Sources /Stores

- While performing data retrieval you may have to work with one of the following data stores

- **SQLite**

    - This requires a package named sqlite3.

- **Microsoft SQL Server**

    - Microsoft SQL server is common in companies, and this connector supports your connection to the database. Via the direct connection, use

```
from sqlalchemy import create_engine
engine =
create_engine('mssql+pymssql://scott:tiger@hostname:port/folder')
```

- **Oracle**

    - Oracle is a common database storage option in bigger companies. It enables you to load data from the following data source with ease:

```
from sqlalchemy import create_engine
engine =
create_engine('oracle://andre:vermeulen@127.0.0.1:1521/vermeulen')
```

- **MySQL**

    - MySQL is widely used by lots of companies for storing data. This opens that data to your data science with the change of a simple connection string.

- There are two options. For direct connect to the database, use

```
from sqlalchemy import create_engine
engine =
    create_engine('mysql+mysqldb://scott:tiger@localhost/vermeulen')
```

- **Apache Cassandra**
  - Cassandra is becoming a widely distributed database engine in the corporate world.
  - To access it, use the Python package cassandra.

```
from cassandra.cluster import Cluster
cluster = Cluster()
session = cluster.connect('vermeulen')
```

- **Apache Hadoop**
  - Hadoop is one of the most successful data lake ecosystems in highly distributed data Science.
  - The pydoop package includes a Python MapReduce and HDFS API for Hadoop.
- **Pydoop**
  - It is a Python interface to Hadoop that allows you to write MapReduce applications and interact with HDFS in pure Python
- Microsoft Excel
  - Excel is common in the data sharing ecosystem, and it enables you to load files using this format with ease.
- **Apache Spark**
  - Apache Spark is now becoming the next standard for distributed data processing. The universal acceptance and support of the processing

ecosystem is starting to turn mastery of this technology into a must-have skill.

- **Apache Hive**
    - Access to Hive opens its highly distributed ecosystem for use by data scientists

- **Luigi**
    - Luigi enables a series of Python features that enable you to build complex pipelines into batch jobs. It handles dependency resolution and workflow management as part of the package.
    - This will save you from performing complex programming while enabling good quality processing

- **Amazon S3 Storage**
    - S3, or Amazon Simple Storage Service (Amazon S3), creates simple and practical methods to collect, store, and analyze data, irrespective of format, completely at massive scale. I store most of my base data in S3, as it is cheaper than most other methods.
    - **Package s3 -** Python's s3 module connects to Amazon's S3 REST API
    - **Package Boot -** The Boto package is another useful too that connects to Amazon's S3 REST API

- **Amazon Redshift**
    - Amazon Redshift is cloud service that is a fully managed, petabyte-scale data warehouse.
    - The Python package redshift-sqlalchemy, is an Amazon Redshift dialect for sqlalchemy that opens this data source to your data science

- **Amazon Web Services**
    - The boto3package is an Amazon Web Services Library Python package that provides interfaces to Amazon Web Services

4.7 Review Question

1. Explain the Retrieve Super step.
2. Explain Data Lakes and Data Swamps.
3. Explain the general rules for data source catalog.
4. State and explain the four critical steps to avoid data swamps.
5. Why is it necessary to train the data science team?
6. Explain the following shipping terms:
    i. Seller,
    ii. Carrier,
    iii. Port,
    iv. Ship,
    v. Terminal, Named Place,
    vi. Buyer.
7. Explain the following shipping terms with example:
    i. Ex Works
    ii. Free Carrier
    iii. Carriage Paid To
    iv. Carriage and Insurance Paid To
    v. Delivered at Terminal
    vi. Delivered at Place

vii. Delivery Duty Paid

8. List and explain the different data stores used in data science.

4.8 References

Books:

- Andreas François Vermeulen, “Practical Data Science - A Guide to Building the Technology Stack for Turning Data Lakes into Business Assets”

Websites:

- https://www.aitworldwide.com/incoterms
- Incoterm: https://www.ntrpco.com/what-is-incoterms-part2/

M.Sc IT Sem I Data Science

Unit – III

Chapter - V

Assess Superstep

Unit Structure

5.0 Objectives

This chapter makes you understand the following concepts:

- Dealing with errors in data
- Principles of data analysis
- Different ways to correct errors in data

5.1 Assess Superstep

Data quality problems result in a 20% decrease in worker productivity and explain why 40% of business initiatives fail to achieve set goals. Incorrect data can harm a reputation, misdirect resources, slow down the retrieval of information, and lead to false insights and missed opportunities.

For example, if an organization has the incorrect name or mailing address of a prospective client, their marketing materials could go to the wrong recipient. If sales data is attributed to the wrong SKU or brand, the company might invest in a product line with less than stellar customer demand.

Data profiling is the process of examining, analyzing and reviewing data to collect statistics surrounding the quality and hygiene of the dataset. Data quality refers to the accuracy, consistency, validity and completeness of data. Data profiling may also be known as data archeology, data assessment, data discovery or data quality analysis.

5.2 Errors

Errors are the norm, not the exception, when working with data. By now, you've probably heard the statistic that 88% of spreadsheets contain errors. Since we cannot safely assume that any of the data we work with is error-free, our mission should be to find and tackle errors in the most efficient way possible.

5.2.1 Accept the Error

If an error falls within an acceptable standard (i.e., Navi Mumbai instead of Navi Mum.), then it could be accepted and move on to the next data entry. But remember that if you accept the error, you will affect data science techniques and algorithms that perform classification, such as binning, regression, clustering, and decision trees, because these processes assume that the values in this example are not the same. This option is the easy option, but not always the best option.

5.2.2 Reject the Error

Unless the nature of missing data is 'Missing completely at random', the best avoidable method in many cases is deletion.

a. Listwise: In this case, rows containing missing variables are deleted.

User	Device	OS	Transactions
A	Mobile	Android	5
B	Mobile	Windows	3
~~C~~	~~Tablet~~	~~NA~~	~~4~~
~~D~~	~~NA~~	~~Android~~	~~1~~
E	Mobile	iOS	2

In the above case, the entire observation for User C and User D will be ignored for listwise deletion.

b. Pairwise: In this case, only the missing observations are ignored and analysis is done on variables present.

User	Device	OS	Transactions
A	Mobile	Android	5
B	Mobile	Windows	3
C	Tablet	~~NA~~	4

D	~~NA~~	Android	1
E	Mobile	iOS	2

In the above case, 2 separate sample data will be analyzed, one with the combination of User, Device and Transaction and the other with the combination of User, OS and Transaction. In such a case, one won't be deleting any observation. Each of the samples will ignore the variable which has the missing value in it.

Both the above methods suffer from loss of information. Listwise deletion suffers the maximum information loss compared to Pairwise deletion. But, the problem with pairwise deletion is that even though it takes the available cases, one can't compare analyses because the sample is different every time.

Use reject the error option if you can afford to lose a bit of data. This is an option to be used only if the number of missing values is 2% of the whole dataset or less.

5.2.3 Correct the Error

Identify the Different Error Types

We are going to look at a few different types of errors. Let's take the example of a sample of people described by a number of different variables:

First Name	**Email**	**Date of Birth**	**Country**	**Height**
Rakesh	rakesh@example.com	23/01/1990	India	1.49m
Samuel	samuel_329@example.com	20/09/2001		1.67m
Rob	choupipoune@supermail.eu	12 Sept. 1984	Madagascar	5'2
Mark	marco23@example.com, mc23@supermail.eu	10/02/1978	24	1.65m
Harry	helloworld@mail.example.com	04/25/1975	Germany	1.34m
Honey	honey2019@supermail.eu	01/01/1970	Canada	2.8m
Samuël	samuel_329@example.com		Benin	1.45m

Can you point out a few inconsistencies? Write them down a few and check your answers below!

1. First, there are empty cells for the "country" and "date of birth variables". We call these **missing attributes**.
2. If you look at the "Country" column, you see a cell that contains 24. "24" is definitely not a country! This is known as a **lexical error**.
3. Next, you may notice in the "Height" column that there is an entry with a different unit of measure. Indeed, Rodney's height is recorded in feet and inches while the rest are recorded in meters. This is an **irregularity error** because the unit of measures are not uniform.
4. Mark has two email addresses. It's is not necessarily a problem, but if you forget about this and code an analysis program based on the assumption that each person has only one email address, your program will probably crash! This is called a **formatting error**.
5. Look at the "date of birth" variable. There is also a **formatting error** here as Rob's date of birth is not recorded in the same format as the others.

6. Samuel appears on two different rows. But, how can we be sure this is the same Samuel? By his email address, of course! This is called a **duplication error**. But look closer, Samuel's two rows each give a different value for the "height variable": 1.67m and 1.45m. This is called a **contradiction error**.
7. Honey is apparently 9'1". This height diverges greatly from the normal heights of human beings. This value is, therefore, referred to as an **outlier**.

The term **outlier** can indicate two different things: an **atypical** value and an **aberration**.

Deal With These Errors

When it comes to cleansing data sets, there is no set rule. Everything you do depends on how you plan to use your data. No two data analysts will cleanse the same data set the same way—not if their objectives are different!

So there's no set rule, but I can give you a few pointers:

1. Missing attributes will be addressed in the following chapter.
2. For the invalid country, it's possible to supply a list of authorized countries in advance, then eliminate all of the values that are not found on this list (hint: 24 will not be found). Such a list is often referred to as a dictionary.
3. For irregularity errors, it's more complicated! You can, for example, set a fixed format (here: a decimal number followed by the letter "m" for "meter") and eliminate values that don't adhere to it. But we can do better, by first detecting what unit the value is expressed in (meters or centimeters) then converting everything to the same unit.
4. For the formatting error of the duplicate email address, it all depends on what you want to do. If you won't be looking at emails in your future analysis, there's no need to correct this error. If, on the other hand, you want to know the proportion of people whose address ends in, for example @example.com, or @supermail.eu, etc., then you can choose between:
    1. Taking the first email address and forgetting the second one.
    2. Keeping all email addresses.
5. Let's move on to the Date of Birth variable. There are many different formats; each country has its own custom when it comes to writing dates (India and North America, for example, do not use the same format). Add to this the problem of time zones! In our case, the simplest solution would be to eliminate dates that are not in the desired format month/day/year.
6. Duplicates.
7. Outliers!

5.2.4 Create a Default Value

NaN is the default missing value marker for reasons of computational speed and convenience. This is a sentinel value, in the sense that it is a dummy data or flag value that can be easily detected and worked with using functions in pandas.

5.3 Analysis of Data

One of the causes of data quality issues is in source data that is housed in a patchwork of operational systems and enterprise applications. Each of these data sources can have scattered or misplaced values, outdated and duplicate records, and inconsistent (or undefined) data standards and formats across customers, products, transactions, financials and more.

Data quality problems can also arise when an enterprise consolidates data during a merger or acquisition. But perhaps the largest contributor to data quality issues is that the data are being entered, edited, maintained, manipulated and reported on by people.

To maintain the accuracy and value of the business-critical operational information that impact strategic decision-making, businesses should implement a data quality strategy that embeds data quality techniques into their business processes and into their enterprise applications and data integration.

5.3.1 Completeness:

Completeness is defined as expected comprehensiveness. Data can be complete even if optional data is missing. As long as the data meets the expectations then the data is considered complete.

For example, a customer's first name and last name are mandatory but middle name is optional; so a record can be considered complete even if a middle name is not available.

Questions you can ask yourself: Is all the requisite information available? Do any data values have missing elements? Or are they in an unusable state?

5.3.2 Consistency

Consistency means data across all systems reflects the same information and are in synch with each other across the enterprise.

Examples:

A business unit status is closed but there are sales for that business unit.

Employee status is terminated but pay status is active.

Questions you can ask yourself: Are data values the same across the data sets? Are there any distinct occurrences of the same data instances that provide conflicting information?

5.3.3 Timeliness

Timeliness referes to whether information is available when it is expected and needed. Timeliness of data is very important. This is reflected in:

- Companies that are required to publish their quarterly results within a given frame of time
- Customer service providing up-to date information to the customers
- Credit system checking in real-time on the credit card account activity

The timeliness depends on user expectation. Online availability of data could be required for room allocation system in hospitality, but nightly data could be perfectly acceptable for a billing system.

5.3.4 Conformity

Conformity means the data is following the set of standard data definitions like data type, size and format. For example, date of birth of customer is in the format “mm/dd/yyyy”
Questions you can ask yourself: Do data values comply with the specified formats? If so, do all the data values comply with those formats?
Maintaining conformance to specific formats is important.

5.3.5 Accuracy

Accuracy is the degree to which data correctly reflects the real world object OR an event being described. Examples:

- Sales of the business unit are the real value.
- Address of an employee in the employee database is the real address.

Questions you can ask yourself: Do data objects accurately represent the “real world” values they are expected to model? Are there incorrect spellings of product or person names, addresses, and even untimely or not current data?
These issues can impact operational and advanced analytics applications.

5.3.6 Integrity

Integrity means validity of data across the relationships and ensures that all data in a database can be traced and connected to other data.
For example, in a customer database, there should be a valid customer, addresses and relationship between them. If there is an address relationship data without a customer then that data is not valid and is considered an orphaned record.
Ask yourself: Is there are any data missing important relationship linkages?
The inability to link related records together may actually introduce duplication across your systems.

5.4 Practical Actions

In Unit 2, you have been introduced to the Python package pandas. The package enables several automatic error-management features.

5.4.1 Missing Values in Pandas

Following are four basic processing concepts.

1. Drop the Columns Where All Elements Are Missing Values
2. Drop the Columns Where Any of the Elements Is Missing Values
3. Keep Only the Rows That Contain a Maximum of Two Missing Values
4. Fill All Missing Values with the Mean, Median, Mode, Minimum

5.4.1.1. Drop the Columns Where All Elements Are Missing Values

Importing data

Step 1: Importing necessary libraries

```
import os
import pandas as pd
```

Step 2: Changing the working directory

```
os.chdir("D:\Pandas")
```

Pandas provides various data structures and operations for manipulating numerical data and time series. However, there can be cases where some data might be missing. In Pandas missing data is represented by two values:

- None: None is a Python singleton object that is often used for missing data in Python code.
- NaN: NaN (an acronym for Not a Number), is a special floating-point value recognized by all systems that use the standard IEEE floating-point representation

Pandas treat None and NaN as essentially interchangeable for indicating missing or null values. In order to drop a null values from a dataframe, we used dropna() function this function drop Rows/Columns of datasets with Null values in different ways.

Syntax:

```
DataFrame.dropna(axis=0, how='any', thresh=None, subset=None, inplace=False)
```

Parameters:

- *axis*: axis takes int or string value for rows/columns. Input can be 0 or 1 for Integer and 'index' or 'columns' for String.
- *how*: how takes string value of two kinds only ('any' or 'all'). 'any' drops the row/column if ANY value is Null and 'all' drops only if ALL values are null.
- *thresh*: thresh takes integer value which tells minimum amount of na values to drop.
- *subset*: It's an array which limits the dropping process to passed rows/columns through list.
- *inplace*: It is a boolean which makes the changes in data frame itself if True.

Let's take an example of following dataframe:

	A	B	C	D
0	NaN	2.0	NaN	0

1	3.0	4.0	NaN	1
2	NaN	NaN	NaN	5

Here, column C is having all NaN values. Let's drop this column. For this use the following code.

Code:

```
import pandas as pd
import numpy as np

df = pd.DataFrame([[np.nan, 2, np.nan, 0], [3, 4, np.nan, 1],
                   [np.nan, np.nan, np.nan, 5]],
                  columns=list('ABCD'))
df # it will print the data frame

df.dropna(axis=1, how='all')  # this code will delete the columns with all null values.
Here, axis=1 means columns and how='all' means drop the columns with all NaN values.
```

Output:

	A	B	D
0	NaN	2.0	0
1	3.0	4.0	1
2	NaN	NaN	5

5.4.1.2. Drop the Columns Where Any of the Elements Is Missing Values

Let's consider the same dataframe again:

	A	B	C	D
0	NaN	2.0	NaN	0
1	3.0	4.0	NaN	1
2	NaN	NaN	NaN	5

Here, column A, B and C are having all NaN values. Let's drop these columns. For this use the following code.

Code:

```
import pandas as pd
import numpy as np

df = pd.DataFrame([[np.nan, 2, np.nan, 0], [3, 4, np.nan, 1],
                   [np.nan, np.nan, np.nan, 5]],
                  columns=list('ABCD'))
df # it will print the data frame
```

df.dropna(axis=1, how='any') # this code will delete the columns with all null values.
Here, axis=1 means columns and how=’any’ means drop the columns with one or nore NaN values.

Output:

	D
0	0
1	1
2	5

5.4.1.3. Keep Only the Rows That Contain a Maximum of Two Missing Values

Let’s consider the same dataframe again:

	A	B	C	D
0	NaN	2.0	NaN	0
1	3.0	4.0	NaN	1
2	NaN	NaN	NaN	5

Here, row 2 is having more than 2 NaN values. So, this row will get dropped. For this use the following code.

Code:

```
# importing pandas as pd
import pandas as pd
import numpy as np

df = pd.DataFrame([[np.nan, 2, np.nan, 0], [3, 4, np.nan, 1],
                   [np.nan, np.nan, np.nan, 5]],
                  columns=list('ABCD'))
df

df.dropna(thresh=2)
  # this code will delete the rows with more than two null values.
```

Here, thresh=2 means maximum two NaN will be allowed per row.

Output:

	A	B	C	D
0	NaN	2.0	NaN	0
1	3.0	4.0	NaN	1

5.4.1.4. Fill All Missing Values with the Mean, Median, Mode, Minimum

Another approach to handling missing values is to impute or estimate them. Missing value imputation has a long history in statistics and has been thoroughly researched. In essence,

imputation uses information and relationships among the non-missing predictors to provide an estimate to fill in the missing value. The goal of these techniques is to ensure that the statistical distributions are tractable and of good enough quality to support subsequent hypothesis testing. The primary approach in this scenario is to use multiple imputations; several variations of the data set are created with different estimates of the missing values. The variations of the data sets are then used as inputs to models and the test statistic replicates are computed for each imputed data set. From these replicate statistics, appropriate hypothesis tests can be constructed and used for decision making.

A simple guess of a missing value is the mean, median, or mode (most frequently appeared value) of that variable.

Replacing Nan values with mean:

In pandas, .fillna can be used to replace NA's with a specified value.
Let's take an example:

	Apple	Orange	Banana	Pear
Basket 1	10	NaN	30	40
Basket 2	7	14	21	28
Basket 3	55	NaN	8	12
Basket 4	15	14	NaN	8
Basket 5	7	1	1	NaN
Basket 6	NaN	4	9	2

Here, we can see NaN in all the columns. Let's fill it by their mean. For this, use the following code:

```
import pandas as pd
import numpy as np

df = pd.DataFrame([[10, np.nan, 30, 40], [7, 14, 21, 28], [55, np.nan, 8, 12],
                   [15, 14, np.nan, 8], [7, 1, 1, np.nan], [np.nan, 4, 9, 2]],
                  columns=['Apple', 'Orange', 'Banana', 'Pear'],
                  index=['Basket1', 'Basket2', 'Basket3', 'Basket4',
                       'Basket5', 'Basket6'])

df

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

Output:

	Apple	Orange	Banana	Pear
Basket 1	10	8.25	30	40
Basket 2	7	14	21	28
Basket 3	55	8.25	8	12
Basket 4	15	14	13.8	8

Basket 5	7	1	1	18
Basket 6	18.8	4	9	2

Here, the mean of Apple Column = (10 + 7 + 55 + 15 + 7)/5 = 18.8. So, Nan value is replaced by 18.8. Similarly, in Orange Column Nan's are replaced with 8.25, in Banana's column Nan replaced with 13.8 and in Pear's column it is replaced with 18.

Replacing Nan values with median:

Let's take an example:

	Apple	Orange	Banana	Pear
Basket 1	10	NaN	30	40
Basket 2	7	14	21	28
Basket 3	55	NaN	8	12
Basket 4	15	14	NaN	8
Basket 5	7	1	1	NaN
Basket 6	NaN	4	9	2

Here, we can see NaN in all the columns. Let's fill it by their median. For this, use the following code:

```
import pandas as pd
import numpy as np

df = pd.DataFrame([[10, np.nan, 30, 40], [7, 14, 21, 28], [55, np.nan, 8, 12],
                   [15, 14, np.nan, 8], [7, 1, 1, np.nan], [np.nan, 4, 9, 2]],
                  columns=['Apple', 'Orange', 'Banana', 'Pear'],
                  index=['Basket1', 'Basket2', 'Basket3', 'Basket4',
                         'Basket5', 'Basket6'])

df

df.fillna(df.median())
```

Output:

	Apple	Orange	Banana	Pear
Basket 1	10	9.0	30	40
Basket 2	7	14	21	28
Basket 3	55	9.0	8	12
Basket 4	15	14	9.0	8
Basket 5	7	1	1	12.0
Basket 6	10.0	4	9	2

Here, the median of Apple Column = (7, 7, 10, 15, 55) = 10. So, Nan value is replaced by 10. Similarly, in Orange Column Nan's are replaced with 9, in Banana's column Nan replaced with 9 and in Pear's column it is replaced with 12.

Replacing Nan values with mode:

Let's take an example:

	Apple	Orange	Banana	Pear
Basket 1	10	NaN	30	40
Basket 2	7	14	8	28
Basket 3	55	NaN	8	12
Basket 4	15	14	NaN	12
Basket 5	7	1	1	NaN
Basket 6	NaN	4	9	2

Here, we can see NaN in all the columns. Let's fill it by their mode. For this, use the following code:

```
import pandas as pd
import numpy as np

df = pd.DataFrame([[10, np.nan, 30, 40], [7, 14, 8, 28], [55, np.nan, 8, 12],
                   [15, 14, np.nan, 12], [7, 1, 1, np.nan], [np.nan, 4, 9, 2]],
                  columns=['Apple', 'Orange', 'Banana', 'Pear'],
                  index=['Basket1', 'Basket2', 'Basket3', 'Basket4',
                         'Basket5', 'Basket6'])

df

for column in df.columns:
    df[column].fillna(df[column].mode()[0], inplace=True)

df
```

Output:

	Apple	Orange	Banana	Pear
Basket 1	10	14	30	40
Basket 2	7	14	8	28
Basket 3	55	14	8	12
Basket 4	15	14	8	12
Basket 5	7	1	1	12
Basket 6	7.0	4	9	2

Here, the mode of Apple Column = (10, 7, 55, 15, 7) = 7. So, Nan value is replaced by 7. Similarly, in Orange Column Nan's are replaced with 14, in Banana's column Nan replaced with 8 and in Pear's column it is replaced with 12.

Replacing Nan values with min:

Let's take an example:

	Apple	Orange	Banana	Pear
Basket 1	10	NaN	30	40
Basket 2	7	14	21	28
Basket 3	55	NaN	8	12
Basket 4	15	14	NaN	8
Basket 5	7	1	1	NaN
Basket 6	NaN	4	9	2

Here, we can see NaN in all the columns. Let's fill it by their minimum value. For this, use the following code:

```
import pandas as pd
import numpy as np

df = pd.DataFrame([[10, np.nan, 30, 40], [7, 14, 21, 28], [55, np.nan, 8, 12],
                   [15, 14, np.nan, 8], [7, 1, 1, np.nan], [np.nan, 4, 9, 2]],
                  columns=['Apple', 'Orange', 'Banana', 'Pear'],
                  index=['Basket1', 'Basket2', 'Basket3', 'Basket4',
                       'Basket5', 'Basket6'])

df

df.fillna(df.min())
```

Output:

	Apple	Orange	Banana	Pear
Basket 1	10	1	30	40
Basket 2	7	14	21	28
Basket 3	55	1	8	12
Basket 4	15	14	1	8
Basket 5	7	1	1	2
Basket 6	7	4	9	2

Here, the minimum of Apple Column = (10, 7, 55, 15, 7) = 7. So, Nan value is replaced by 7. Similarly, in Orange Column Nan's are replaced with 1, in Banana's column Nan replaced with 1 and in Pear's column it is replaced with 2.

5.5 Let us Sum up

This chapter focuses on dealing with errors in data. The main concepts related to errors are: accept the errors. This is very crucial. Another way is to reject the errors. This step can be used if you can take this risk and not more than 10-15% data is to be compromised. Another way is to correct the error. To correct the errors, there are different practical solutions available like using different error correction methods.

Principles of data analysis were also discussed.

Practical Solutions to solve the missing values were also covered like Drop the Columns Where All Elements Are Missing Values, Drop the Columns Where Any of the Elements Is

Missing Values, and keep Only the Rows That Contain a Maximum of Two Missing Values, Fill All Missing Values with the Mean, Median, Mode, Minimum, and Maximum of the Particular Numeric Column

5.6 List of References

- Python for Data Science For Dummies, by Luca Massaron John Paul Mueller (Author), ISBN-13 : 978-8126524938, Wiley
- Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython, 2nd Edition by William McKinney (Author), ISBN-13 : 978-9352136414 , Shroff/O'Reilly
- Data Science From Scratch: First Principles with Python, Second Edition by Joel Grus, ISBN-13 : 978-9352138326, Shroff/O'Reilly
- Data Science from Scratch by Joel Grus, ISBN-13 : 978-1491901427 , O'Reilly
- Data Science Strategy For Dummies by Ulrika Jagare, ISBN-13 : 978-8126533367 , Wiley
- Pandas for Everyone: Python Data Analysis, by Daniel Y. Chen, ISBN-13 : 978-9352869169, Pearson Education
- Practical Data Science with R (MANNING) by Nina Zumel, John Mount, ISBN-13 : 978-9351194378, Dreamtech Press

5.7 Exercise Questions

Q.1 Explain error

Q.2 Explain the different ways to deal with errors.

Q.3 Explain the principles of data analysis.

Q.4 How you will handle missing values in Pandas? Explain.

Q.5 Write a python program to Drop the Columns Where All Elements Are Missing Values.

Q.6 Write a python program to Drop the Columns Where Any of the Elements Is Missing Values.

Q.7 Write a python program to keep Only the Rows That Contain a Maximum of Two Missing Values.

Q.8 Write a python program to Fill All Missing Values with the Mean of the particular column.

Q.9 Write a python program to Fill All Missing Values with the Median of the particular column.

Q.10 Write a python program to Fill All Missing Values with the Mode of the particular column.

Q.11 Write a python program to Fill All Missing Values with the Minimum of the particular column.

Q.12 Write a python program to Fill All Missing Values with the Maximum of the particular column.

M.Sc IT Sem I Data Science

Unit – III

Assess Superstep

Chapter - VI

Unit Structure

6.0 Objectives

This chapter will make you understand the practical concepts of:

- Assess superstep
- Python NetworkX Library used to draw network routing graphs
- Python Schedule library to schedule various jobs

6.1 Engineering a Practical Assess Superstep

Let us first consider an example of Network routing:
Python uses a library called NetworkX for network routing.

To use NetworkX library, first install the library on your machine by using following command on your command prompt:

pip install NetworkX

NetworkX is a Python package for the creation, manipulation, and study of the structure, dynamics, and functions of complex networks.

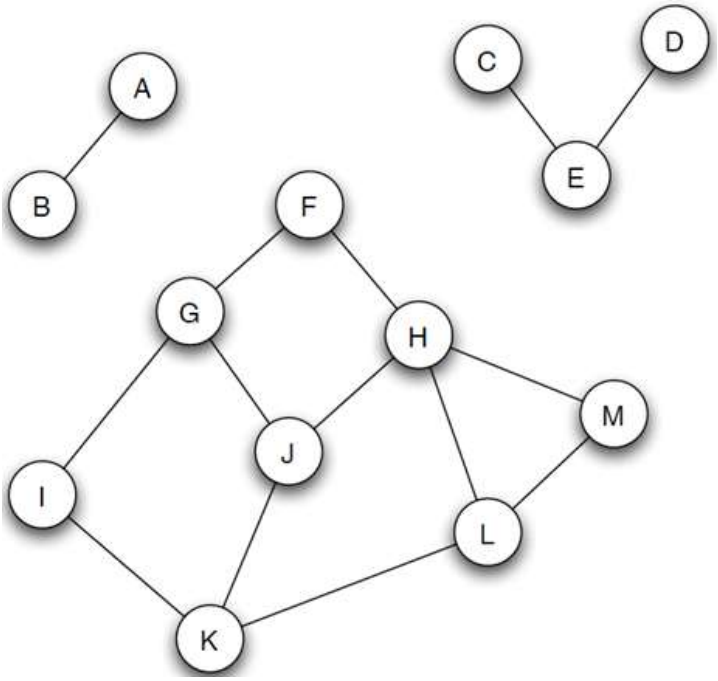
NetworkX provides:

- tools for the study of the structure and dynamics of social, biological, and infrastructure networks;
- a standard programming interface and graph implementation that is suitable for many applications;
- a rapid development environment for collaborative, multidisciplinary projects;
- an interface to existing numerical algorithms and code written in C, C++, and FORTRAN; and
- the ability to painlessly work with large nonstandard data sets.

With NetworkX you can load and store networks in standard and nonstandard data formats, generate many types of random and classic networks, analyze network structure, build network models, design new network algorithms, draw networks, and much more.

Graph Theory

In the Graph Theory, a graph has a finite set of vertices (V) connected to two-elements (E). Each vertex (v) connecting two destinations, or nodes, is called a link or an edge. Consider the Graph of bike paths below: sets {K,L}, {F,G}, {J,H}, {H,L}, {A,B}, and {C,E} are examples of edges.

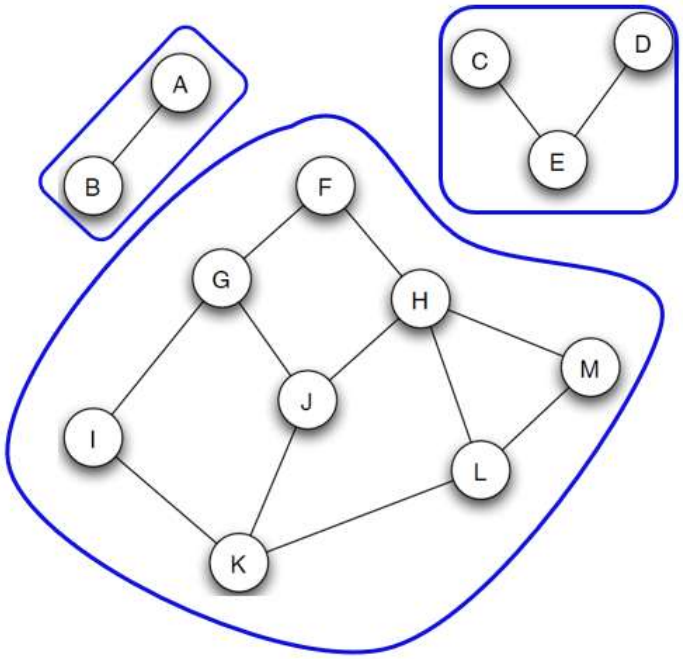


The total number of edges for each node is the degree of that node. In the Graph above, M has a degree of 2 ({M,H} and {M,L}) while B has a degree of 1 ({B,A}). Degree is described formally as:

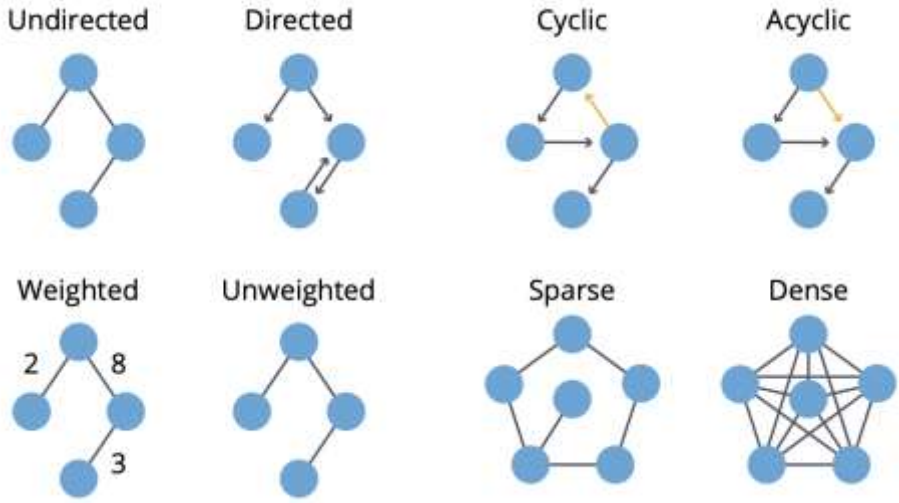
$$\begin{aligned} d_i(g) &= \text{number of links involving } i \\ &= \sum_{j=1}^{n} g_{ij} \end{aligned}$$

Connections through use of multiple edges are called paths. {F, H, M, L, H, J, G, I} is an example of a path. A simple path is when a path does not repeat a node — formally known as Eulerian path. {I, G, J, H, F} is an example of a simple path. The shortest simple path is called Geodesic. Geodesic between I and J is {I, G, J} or {I, K, J}. Finally, a cycle is when a path's start and end points are the same (ex. {H,M,L,H}). In some notebooks, a cycle is formally referred to as Eulerian cycle.

Not all networks in a Graph system are interconnected. This disconnection is when components are formed. As shown in the graph below, a component is formed only when every node has a path to other nodes.



Neo4J's book on graph algorithms provides a clear summary :



For example:

Creating a graph

Create an empty graph with no nodes and no edges.

```
import networkx as nx
G = nx.Graph()

# By definition, a `Graph` is a collection of nodes (vertices) along with identified pairs of
```

nodes (called # edges, links, etc). In NetworkX, nodes can be any [hashable] object e.g., a
text string, an image, an # XML object, another Graph, a customized node object, etc.
Nodes
The graph `G` can be grown in several ways. NetworkX includes many graph generator
functions # and facilities to read and write graphs in many formats.
To get started # though we'll look at simple manipulations. You can add one node at a
time,

```
G.add_node(1)
```

or add nodes from any [iterable] container, such as a list

```
G.add_nodes_from([2, 3])
```

Nodes from one graph can be incorporated into another:

```
H = nx.path_graph(10)
G.add_nodes_from(H)
```

`G` now contains the nodes of `H` as nodes of `G`.
In contrast, you could use the graph `H` as a node in `G`.

```
G.add_node(H)
```

The graph `G` now contains `H` as a node. This flexibility is very powerful as it allows
graphs of graphs, graphs of files, graphs of functions and much more. It is worth thinking
about how to structure # your application so that the nodes are useful entities. Of course
you can always use a unique identifier # in `G` and have a separate dictionary keyed by
identifier to the node information if you prefer.

Edges
`G` can also be grown by adding one edge at a time,

```
G.add_edge(1, 2)
e = (2, 3)
G.add_edge(*e)  # unpack edge tuple*
```

by adding a list of edges,

```
G.add_edges_from([(1, 2), (1, 3)])
```

```
# or by adding any ebunch of edges.  An *ebunch* is any iterable container of edge-tuples.
# An edge-tuple can be a 2-tuple of nodes or a 3-tuple with 2 nodes followed by an edge
# attribute dictionary, e.g.,
# `(2, 3, {'weight': 3.1415})`.  Edge attributes are discussed further below.

G.add_edges_from(H.edges)

# There are no complaints when adding existing nodes or edges.
# For example, after removing all # nodes and edges,

G.clear()

# we add new nodes/edges and NetworkX quietly ignores any that are already present.

G.add_edges_from([(1, 2), (1, 3)])
G.add_node(1)
G.add_edge(1, 2)
G.add_node("spam")       # adds node "spam"
G.add_nodes_from("spam")  # adds 4 nodes: 's', 'p', 'a', 'm'
G.add_edge(3, 'm')

# At this stage the graph `G` consists of 8 nodes and 3 edges, as can be seen by:

G.number_of_nodes()
G.number_of_edges()

# # Examining elements of a graph
# We can examine the nodes and edges. Four basic graph properties facilitate reporting: #`G.nodes`,
# `G.edges`, `G.adj` and `G.degree`.  These are set-like views of the nodes, edges, neighbors
# (adjacencies), and degrees of nodes in a graph. They offer a continually updated read-only
#view into the graph structure. They are also dict-like in that you can look up node and edge
#data attributes via the views and iterate with data attributes using methods `.items()`,
#`.data('span')`.
# If you want a specific container type instead of a view, you can specify one.
# Here we use lists, though sets, dicts, tuples and other containers may be better in other
#contexts.

list(G.nodes)
list(G.edges)
```

```
list(G.adj[1])  # or list(G.neighbors(1))
G.degree[1]  # the number of edges incident to 1

# One can specify to report the edges and degree from a subset of all nodes using an #nbunch.
# An *nbunch* is any of: `None` (meaning all nodes),  a node, or an iterable container of nodes that is # not itself a node in the graph.

G.edges([2, 'm'])
G.degree([2, 3])

# # Removing elements from a graph
# One can remove nodes and edges from the graph in a similar fashion to adding.
# Use methods `Graph.remove_node()`, `Graph.remove_nodes_from()`, #`Graph.remove_edge()`
# and `Graph.remove_edges_from()`, e.g.

G.remove_node(2)
G.remove_nodes_from("spam")
list(G.nodes)
G.remove_edge(1, 3)

# # Using the graph constructors
# Graph objects do not have to be built up incrementally - data specifying
# graph structure can be passed directly to the constructors of the various graph classes.
# When creating a graph structure by instantiating one of the graph
# classes you can specify data in several formats.

G.add_edge(1, 2)
H = nx.DiGraph(G)   # create a DiGraph using the connections from G
list(H.edges())
edgelist = [(0, 1), (1, 2), (2, 3)]
H = nx.Graph(edgelist)

# # What to use as nodes and edges
# You might notice that nodes and edges are not specified as NetworkX
# objects.  This leaves you free to use meaningful items as nodes and
# edges. The most common choices are numbers or strings, but a node can
# be any hashable object (except `None`), and an edge can be associated
# with any object `x` using `G.add_edge(n1, n2, object=x)`.
```

```
# As an example, `n1` and `n2` could be protein objects from the RCSB Protein Data Bank,
#and `x` # could refer to an XML record of publications detailing experimental observations
#of their interaction.
# We have found this power quite useful, but its abuse can lead to surprising behavior
#unless one is # familiar with Python.
# If in doubt, consider using `convert_node_labels_to_integers()` to obtain a more
traditional graph with # integer labels.  Accessing edges and neighbors
# In addition to the views `Graph.edges`, and `Graph.adj`, access to edges and neighbors is
#possible using subscript notation.

G = nx.Graph([(1, 2, {"color": "yellow"})])
G[1]  # same as G.adj[1]
G[1][2]
G.edges[1, 2]

# You can get/set the attributes of an edge using subscript notation
# if the edge already exists.

G.add_edge(1, 3)
G[1][3]['color'] = "blue"
G.edges[1, 2]['color'] = "red"
G.edges[1, 2]

# Fast examination of all (node, adjacency) pairs is achieved using
# `G.adjacency()`, or `G.adj.items()`.
# Note that for undirected graphs, adjacency iteration sees each edge twice.

FG = nx.Graph()
FG.add_weighted_edges_from([(1, 2, 0.125), (1, 3, 0.75), (2, 4, 1.2), (3, 4, 0.375)])
for n, nbrs in FG.adj.items():
  for nbr, eattr in nbrs.items():
      wt = eattr['weight']
      if wt < 0.5: print(f"({n}, {nbr}, {wt:.3})")

# Convenient access to all edges is achieved with the edges property.

for (u, v, wt) in FG.edges.data('weight'):
   if wt < 0.5:
       print(f"({u}, {v}, {wt:.3})")

# # Adding attributes to graphs, nodes, and edges
```

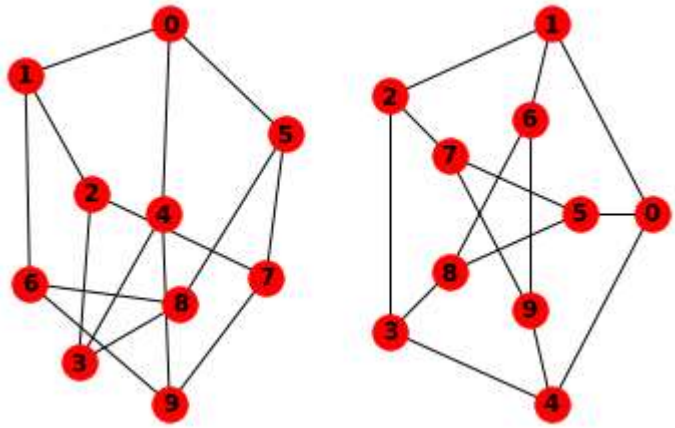
```
#
# Attributes such as weights, labels, colors, or whatever Python object you like,
# can be attached to graphs, nodes, or edges.
#
# Each graph, node, and edge can hold key/value attribute pairs in an associated
# attribute dictionary (the keys must be hashable).  By default these are empty,
# but attributes can be added or changed using `add_edge`, `add_node` or direct
# manipulation of the attribute dictionaries named `G.graph`, `G.nodes`, and
# `G.edges` for a graph `G`.
# ## Graph attributes
# Assign graph attributes when creating a new graph

G = nx.Graph(day="Friday")
G.graph

# Or you can modify attributes later

G.graph['day'] = "Monday"
G.graph

# # Node attributes
# Add node attributes using `add_node()`, `add_nodes_from()`, or `G.nodes`

G.add_node(1, time='5pm')
G.add_nodes_from([3], time='2pm')
G.nodes[1]
G.nodes[1]['room'] = 714
G.nodes.data()

# Note that adding a node to `G.nodes` does not add it to the graph, use
# `G.add_node()` to add new nodes. Similarly for edges.
# # Edge Attributes
# Add/change edge attributes using `add_edge()`, `add_edges_from()`,
# or subscript notation.

G.add_edge(1, 2, weight=4.7 )
G.add_edges_from([(3, 4), (4, 5)], color='red')
G.add_edges_from([(1, 2, {'color': 'blue'}), (2, 3, {'weight': 8})])
G[1][2]['weight'] = 4.7
G.edges[3, 4]['weight'] = 4.2
```

```
# The special attribute `weight` should be numeric as it is used by
# algorithms requiring weighted edges.
#  Directed graphs
# The `DiGraph` class provides additional methods and properties specific
# to directed edges, e.g.,
# `DiGraph.out_edges`, `DiGraph.in_degree`,
# `DiGraph.predecessors()`, `DiGraph.successors()` etc.
# To allow algorithms to work with both classes easily, the directed versions of
# `neighbors()` is equivalent to `successors()` while `degree` reports
# the sum of `in_degree` and `out_degree` even though that may feel
# inconsistent at times.

DG = nx.DiGraph()
DG.add_weighted_edges_from([(1, 2, 0.5), (3, 1, 0.75)])
DG.out_degree(1, weight='weight')
DG.degree(1, weight='weight')
list(DG.successors(1))
list(DG.neighbors(1))

# Some algorithms work only for directed graphs and others are not well
# defined for directed graphs.  Indeed the tendency to lump directed
# and undirected graphs together is dangerous.  If you want to treat
# a directed graph as undirected for some measurement you should probably
# convert it using `Graph.to_undirected()` or with

H = nx.Graph(G)  # create an undirected graph H from a directed graph G

# # Multigraphs
# NetworkX provides classes for graphs which allow multiple edges
# between any pair of nodes.  The `MultiGraph` and
# `MultiDiGraph`
# classes allow you to add the same edge twice, possibly with different
# edge data.  This can be powerful for some applications, but many
# algorithms are not well defined on such graphs.
# Where results are well defined,
# e.g., `MultiGraph.degree()` we provide the function.  Otherwise you
# should convert to a standard graph in a way that makes the measurement well defined.

MG = nx.MultiGraph()
MG.add_weighted_edges_from([(1, 2, 0.5), (1, 2, 0.75), (2, 3, 0.5)])
dict(MG.degree(weight='weight'))
```

```
GG = nx.Graph()
for n, nbrs in MG.adjacency():
  for nbr, edict in nbrs.items():
      minvalue = min([d['weight'] for d in edict.values()])
      GG.add_edge(n, nbr, weight = minvalue)

nx.shortest_path(GG, 1, 3)

# # Graph generators and graph operations
# In addition to constructing graphs node-by-node or edge-by-edge, they
# can also be generated by
# 1. Applying classic graph operations, such as:
# 1. Using a call to one of the classic small graphs, e.g.,
# 1. Using a (constructive) generator for a classic graph, e.g.,
# like so:

K_5 = nx.complete_graph(5)
K_3_5 = nx.complete_bipartite_graph(3, 5)
barbell = nx.barbell_graph(10, 10)
lollipop = nx.lollipop_graph(10, 20)

# 1. Using a stochastic graph generator, e.g, like so:

er = nx.erdos_renyi_graph(100, 0.15)
ws = nx.watts_strogatz_graph(30, 3, 0.1)
ba = nx.barabasi_albert_graph(100, 5)
red = nx.random_lobster(100, 0.9, 0.9)

# 1. Reading a graph stored in a file using common graph formats,
#    such as edge lists, adjacency lists, GML, GraphML, pickle, LEDA and others.

nx.write_gml(red, "path.to.file")
mygraph = nx.read_gml("path.to.file")

# For details on graph formats see Reading and writing graphs
# and for graph generator functions see Graph generators
# # Analyzing graphs
# The structure of `G` can be analyzed using various graph-theoretic functions such as:

G = nx.Graph()
G.add_edges_from([(1, 2), (1, 3)])
```

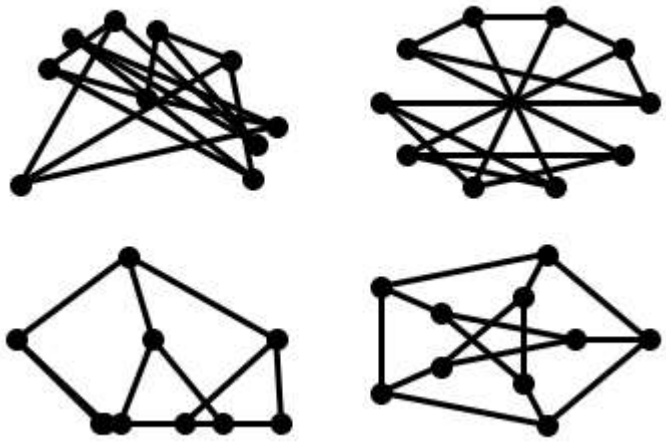
```
G.add_node("spam")      # adds node "spam"
list(nx.connected_components(G))
sorted(d for n, d in G.degree())
nx.clustering(G)

# Some functions with large output iterate over (node, value) 2-tuples.
# These are easily stored in a [dict](https://docs.python.org/3/library/stdtypes.html#dict)
# structure if you desire.

sp = dict(nx.all_pairs_shortest_path(G))
sp[3]

# See Algorithms for details on graph algorithms supported.
# # Drawing graphs
# NetworkX is not primarily a graph drawing package but basic drawing with
# Matplotlib as well as an interface to use the open source Graphviz software
# package are included.  These are part of the `networkx.drawing` module and will
# be imported if possible.
# First import Matplotlib's plot interface (pylab works too)

import matplotlib.pyplot as plt

# To test if the import of `networkx.drawing` was successful draw `G` using one of

G = nx.petersen_graph()
plt.subplot(121)
nx.draw(G, with_labels=True, font_weight='bold')
plt.subplot(122)
nx.draw_shell(G, nlist=[range(5, 10), range(5)], with_labels=True, font_weight='bold')

# when drawing to an interactive display.  Note that you may need to issue a Matplotlib

plt.show()

options = {
   'node_color': 'black',
   'node_size': 100,
   'width': 3,
}
plt.subplot(221)
nx.draw_random(G, **options)
```

```
plt.subplot(222)
nx.draw_circular(G, **options)
plt.subplot(223)
nx.draw_spectral(G, **options)
plt.subplot(224)
nx.draw_shell(G, nlist=[range(5,10), range(5)], **options)

# You can find additional options via `draw_networkx()` and
# layouts via `layout`.
# You can use multiple shells with `draw_shell()`.

G = nx.dodecahedral_graph()
shells = [[2, 3, 4, 5, 6], [8, 1, 0, 19, 18, 17, 16, 15, 14, 7], [9, 10, 11, 12, 13]]
nx.draw_shell(G, nlist=shells, **options)

# To save drawings to a file, use, for example
nx.draw(G)
plt.savefig("path.png")

# writes to the file `path.png` in the local directory.
```
Output:
```
G = nx.petersen_graph()
plt.subplot(121)
nx.draw(G, with_labels=True, font_weight='bold')
plt.subplot(122)
nx.draw_shell(G, nlist=[range(5, 10), range(5)], with_labels=True, font_weight='bold')
```

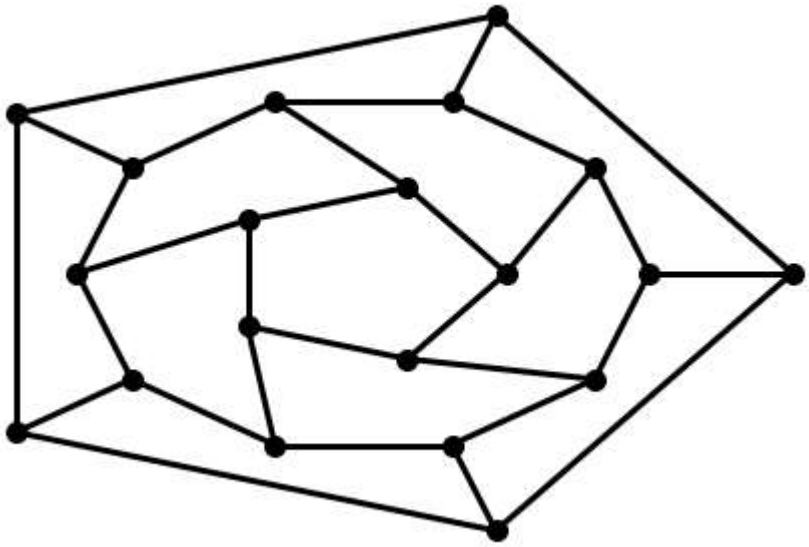



```
plt.show()
options = {
    'node_color': 'black',
    'node_size': 100,
```

```
    'width': 3,
}
plt.subplot(221)
nx.draw_random(G, **options)
plt.subplot(222)
nx.draw_circular(G, **options)
plt.subplot(223)
nx.draw_spectral(G, **options)
plt.subplot(224)
nx.draw_shell(G, nlist=[range(5,10), range(5)], **options)
```

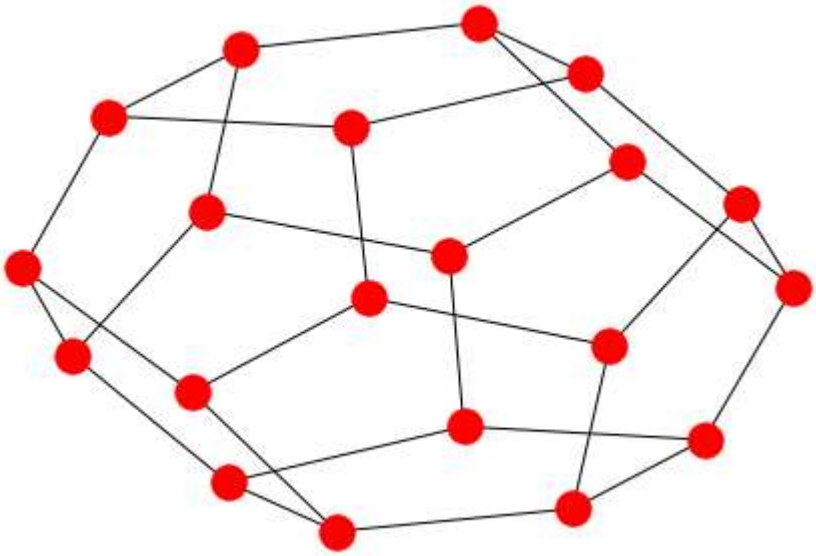


```
G = nx.dodecahedral_graph()
shells = [[2, 3, 4, 5, 6], [8, 1, 0, 19, 18, 17, 16, 15, 14, 7], [9, 10, 11, 12, 13]]
nx.draw_shell(G, nlist=shells, **options)
```



```
nx.draw(G)
```

plt.savefig("path.png")



Building a DAG for Scheduling Jobs

Python Schedule Library

Schedule is in-process scheduler for periodic jobs that use the builder pattern for configuration. Schedule lets you run Python functions (or any other callable) periodically at pre-determined intervals using a simple, human-friendly syntax.

Schedule Library is used to schedule a task at a particular time every day or a particular day of a week. We can also set time in 24 hours format that when a task should run. Basically, Schedule Library matches your systems time to that of scheduled time set by you. Once the scheduled time and system time matches the job function (command function that is scheduled) is called.

Installation

$ pip install schedule

schedule.Scheduler class

- schedule.every(interval=1) : Calls every on the default scheduler instance. Schedule a new periodic job.
- schedule.run_pending() : Calls run_pending on the default scheduler instance. Run all jobs that are scheduled to run.
- schedule.run_all(delay_seconds=0) : Calls run_all on the default scheduler instance. Run all jobs regardless if they are scheduled to run or not.

- schedule.idle_seconds() : Calls idle_seconds on the default scheduler instance.
- schedule.next_run() : Calls next_run on the default scheduler instance. Datetime when the next job should run.
- schedule.cancel_job(job) : Calls cancel_job on the default scheduler instance. Delete a scheduled job.
- schedule.Job(interval, scheduler=None) class

A periodic job as used by Scheduler.

Parameters:

- interval: A quantity of a certain time unit
- scheduler: The Scheduler instance that this job will register itself with once it has been fully configured in Job.do().

Basic methods for Schedule.job

- at(time_str) : Schedule the job every day at a specific time. Calling this is only valid for jobs scheduled to run every N day(s). Parameters: time_str – A string in XX:YY format.

Returns: The invoked job instance

- do(job_func, *args, **kwargs) : Specifies the job_func that should be called every time the job runs. Any additional arguments are passed on to job_func when the job runs. Parameters: job_func – The function to be scheduled. Returns: The invoked job instance
- run() : Run the job and immediately reschedule it. Returns: The return value returned by the job_func
- to(latest) : Schedule the job to run at an irregular (randomized) interval. For example, every(A).to(B).seconds executes the job function every N seconds such that A <= N <= B.

For example

```
# Schedule Library imported
import schedule
import time

# Functions setup
def placement():
   print("Get ready for Placement at various companies")

def good_luck():
   print("Good Luck for Test")

def work():
   print("Study and work hard")

def bedtime():
```

```
    print("It is bed time go rest")

def datascience():
    print("Data science with python is fun")

# Task scheduling
# After every 10mins datascience() is called.
schedule.every(10).minutes.do(datascience)

# After every hour datascience() is called.
schedule.every().hour.do(datascience)

# Every day at 12am or 00:00 time bedtime() is called.
schedule.every().day.at("00:00").do(bedtime)

# After every 5 to 10 mins in between run work()
schedule.every(5).to(10).minutes.do(work)

# Every monday good_luck() is called
schedule.every().monday.do(good_luck)

# Every tuesday at 18:00 placement() is called
schedule.every().tuesday.at("18:00").do(placement)

# Loop so that the scheduling task
# keeps on running all time.
while True:

    # Checks whether a scheduled task
    # is pending to run or not
    schedule.run_pending()
    time.sleep(1)
```

6.2 References:

- Python for Data Science For Dummies, by Luca Massaron John Paul Mueller (Author), ISBN-13 : 978-8126524938, Wiley
- Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython, 2nd Edition by William McKinney (Author), ISBN-13 : 978-9352136414 , Shroff/O'Reilly
- Data Science From Scratch: First Principles with Python, Second Edition by Joel Grus, ISBN-13 : 978-9352138326, Shroff/O'Reilly
- Data Science from Scratch by Joel Grus, ISBN-13 : 978-1491901427 , O'Reilly

- Data Science Strategy For Dummies by Ulrika Jagare, ISBN-13 : 978-8126533367 , Wiley
- Pandas for Everyone: Python Data Analysis, by Daniel Y. Chen, ISBN-13 : 978-9352869169, Pearson Education
- Practical Data Science with R (MANNING) by Nina Zumel, John Mount, ISBN-13 : 978-9351194378, Dreamtech Press

6.3 Exercise Questions

Q.1 Write Python program to create the network routing diagram from the given data.

Q.2 Write a Python program to build directed acyclic graph.

Q.3 Write a Python program to pick the content for Bill Boards from the given data.

Q.4 Write a Python program to generate visitors data from the given csv file.

UNIT IV
CHAPTER 1: PROCESS SUPERSTEP

Structure:

4.1.1 Objectives

The objective of this chapter to learn Time-Person-Object-Location-Event(T-P-O-L-E) design principle and various concepts that are use to create/define relationship among this data.

4.1.2 Introduction

The Process superstep uses the assess results of the retrieve versions of the data sources into a highly structured data vault. These data vaults form the basic data structure for the rest of the data science steps.

The Process superstep is the amalgamation procedure that pipes your data sources into five primary classifications of data.

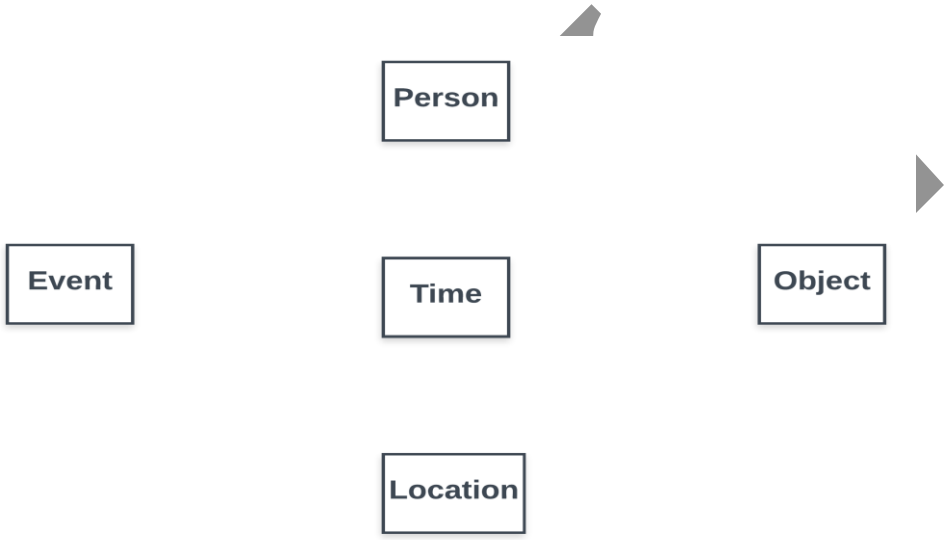


Figure 4.1-1. Categories of data

4.1.3 Data Vault

Data Vault modelling is a technique to manage long term storage of data from multiple operation system. It stores historical data in the database.

4.1.3.1 Hubs

Data vault hub is used to store business key. These keys do not change over time. Hub also contains a surrogate key for each hub entry and metadata information for a business key.

4.1.3.2 Links

Data vault links are join relationship between business keys.

4.1.3.3 Satellites

Data vault satellites stores the chronological descriptive and characteristics for a specific section of business data. Using hub and links we get model structure but no chronological characteristics. Satellites consist of characteristics and metadata linking them to their specific hub.

4.1.3.4 Reference Satellites

Reference satellites are referenced from satellites that can be used by other satellites to prevent redundant storage of reference characteristics.

4.1.4 Time-Person-Object-Location-Event Data Vault

We will use Time-Person-Object-Location-Event (T-P-O-L-E) design principle.
All five sections are linked with each other, resulting into sixteen links.

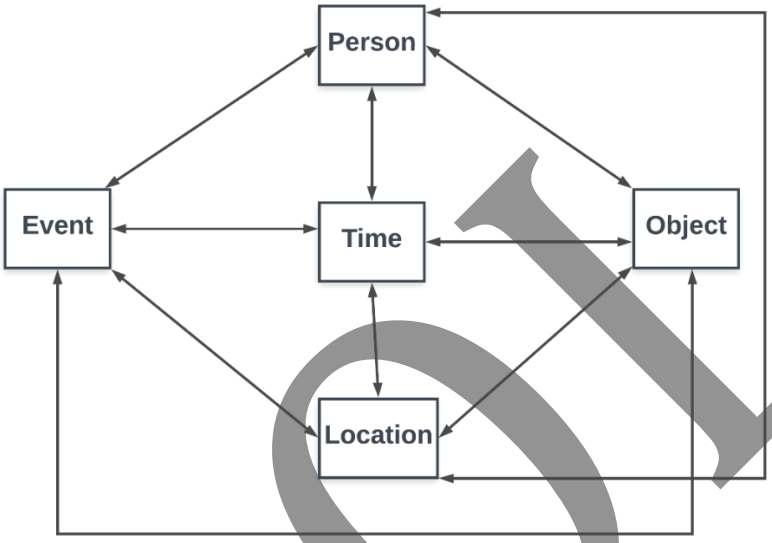


Figure 4.1-2. Time-Person-Object-Location-Event high-level design

4.1.5 Time Section

Time section contain data structure to store all time related information.
For example, time at which event has occurred.

4.1.5.1Time Hub

This hub act as connector between time zones.
Following are the fields of time hub.

```
CREATE TABLE [Hub-Time] (
IDNumber VARCHAR (100) PRIMARY KEY,
IDTimeNumber Integer,
ZoneBaseKey VARCHAR (100),
DateTimeKey VARCHAR (100),
DateTimeValue DATETIME
);
```

4.1.5.2 Time Links

Time Links connect time hub to other hubs.

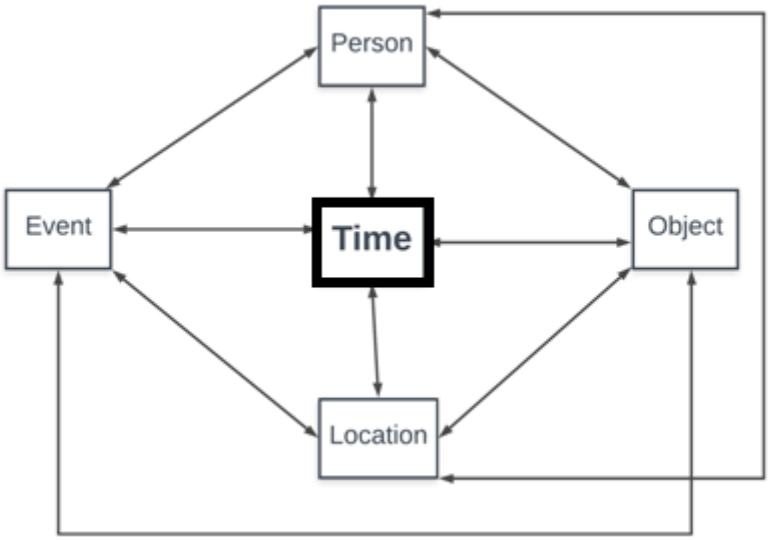


Figure 4.1-3. Time link

Following are the time links that can be stored as separate links.

- **Time-Person Link**
  - This link connects date-time values from time hub to person hub.
  - Dates such as birthdays, anniversaries, book access date, etc.
- **Time-Object Link**
  - This link connects date-time values from time hub to object hub.
  - Dates such as when you buy or sell car, house or book, etc.
- **Time-Location Link**
  - This link connects date-time values from time hub to location hub.
  - Dates such as when you moved or access book from post code, etc.
- **Time-Event Link**
  - This link connects date-time values from time hub to event hub.
  - Dates such as when you changed vehicles, etc.

4.1.5.3 Time Satellites

Following are the fields of time satellites.

```
CREATE TABLE [Satellite-Time-<Time Zone>] (
IDZoneNumber VARCHAR (100) PRIMARY KEY,
IDTimeNumber INTEGER,
ZoneBaseKey VARCHAR (100),
DateTimeKey VARCHAR (100),
UTCDateTimeValue DATETIME,
Zone VARCHAR (100),
DateTimeValue DATETIME
);
```

Time satellite can be used to move from one time zone to other very easily. This feature will be used during Transform superstep.

4.1.6 Person Section

Person section contains data structure to store all data related to person.

4.1.6.1 Person Hub

Following are the fields of Person hub.

```
CREATE TABLE [Hub-Person] (
IDPersonNumber INTEGER,
FirstName VARCHAR (200),
SecondName VARCHAR (200),
LastName VARCHAR (200),
Gender VARCHAR (20),
TimeZone VARCHAR (100),
BirthDateKey VARCHAR (100),
BirthDate DATETIME
);
```

4.1.6.2 Person Links

Person Links connect person hub to other hubs.

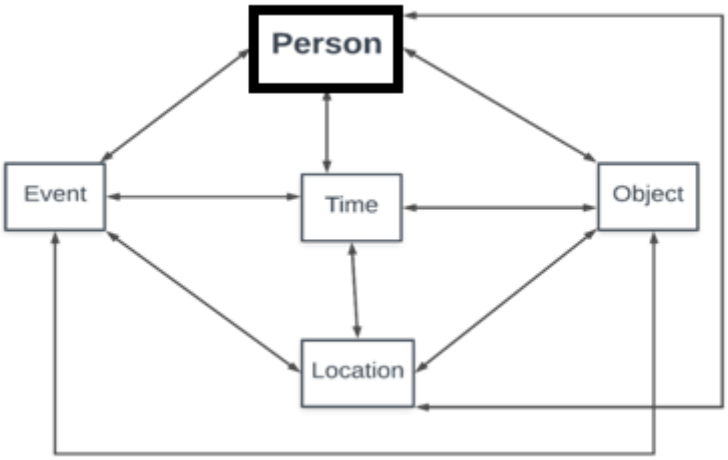


Figure 4.1-4. Person Link

Following are the person links that can be stored as separate links.

- **Person-Time Link**
  - This link contains relationship between person hub and time hub.
- **Person-Object Link**
  - This link contains relationship between person hub and object hub.
- **Person-Location Link**
  - This link contains relationship between person hub and location hub.
- **Person-Event Link**
  - This link contains relationship between person hub and event hub.

4.1.6.3 Person Satellites

Person satellites are part of vault. Basically, it is information about birthdate, anniversary or validity dates of ID for respective person.

```
CREATE TABLE [Satellite-Person-Gender] (
PersonSatelliteID VARCHAR (100),
IDPersonNumber INTEGER,
FirstName VARCHAR (200),
SecondName VARCHAR (200),
LastName VARCHAR (200),
BirthDateKey VARCHAR (20),
Gender VARCHAR (10),
);
```

4.1.4.1 Object Section

Object section contains data structure to store all data related to object.

4.1.4.1.1 Object Hub

Object hub represent a real-world object with few attributes.
Following are the fields of object hub.

```
CREATE TABLE [Hub-Object-Species] (
IDObjectNumber INTEGER,
ObjectBaseKey VARCHAR (100),
ObjectNumber VARCHAR (100),
ObjectValue VARCHAR (200),
);
```

4.1.4.1.2Object Links

Object Links connect object hub to other hubs.

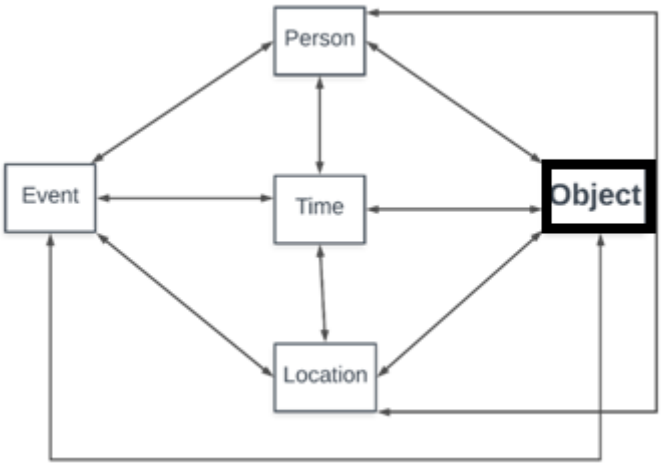


Figure 4.1-5. Object Link

Following are the object links that can be stored as separate links.

- **Object-Time Link**
  - This link contains relationship between Object hub and time hub.

- **Object-Person Link**
  - This link contains relationship between Object hub and Person hub.
- **Object-Location Link**
  - This link contains relationship between Object hub and Location hub.
- **Object-Event Link**
  - This link contains relationship between Object hub and event hub.

4.1.4.1.3 Object Satellites

Object satellites are part of vault. Basically, it is information about ID,UUID, type, key, etc. for respective object.

```
CREATE TABLE [Satellite-Object-Make-Model] (
IDObjectNumber INTEGER,
ObjectSatelliteID VARCHAR (200),
ObjectType VARCHAR (200),
ObjectKey VARCHAR (200),
ObjectUUID VARCHAR (200),
Make VARCHAR (200),
Model VARCHAR (200)
);
```

4.1.8 Location Section

Location section contains data structure to store all data related to location.

4.1.8.1 Location Hub

The location hub consists of a series of fields that supports a GPS location. The locationhub consists of the following fields:

```
CREATE TABLE [Hub-Location] (
IDLocationNumber INTEGER,
ObjectBaseKey VARCHAR (200),
LocationNumber INTEGER,
LocationName VARCHAR (200),
Longitude DECIMAL (9, 6),
Latitude DECIMAL (9, 6)
);
```

4.1.8.2Location Links

Location Links connect location hub to other hubs.

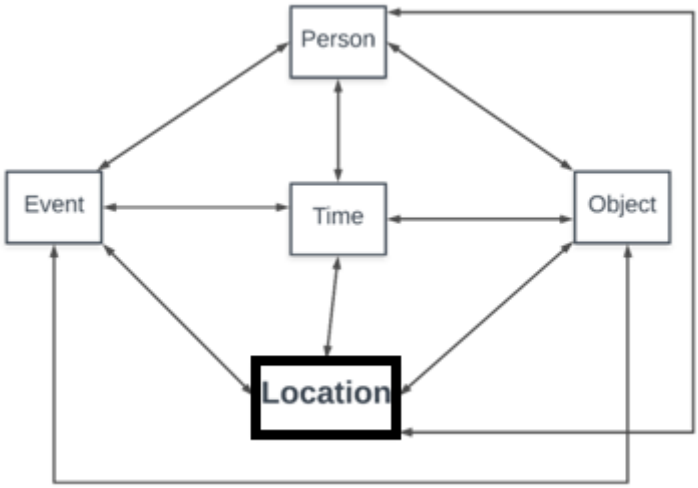


Figure 4.1-6. Location Link

Following are the location links that can be stored as separate links.

- **Location-Time Link**
  - This link contains relationship between location hub and time hub.
- **Location-Person Link**
  - This link contains relationship between location hub and person hub.
- **Location-Object Link**
  - This link contains relationship between location hub and object hub.
- **Location-Event Link**
  - This link contains relationship between location hub and event hub.

4.1.8.3 Location Satellites

Location satellites are part of vault that contains locations of entities.

```
CREATE TABLE [Satellite-Location-PostCode] (
IDLocationNumber INTEGER,
LocationSatelliteID VARCHAR (200),
LocationType VARCHAR (200),
LocationKey VARCHAR (200),
LocationUUID VARCHAR (200),
CountryCode VARCHAR (20),
PostCode VARCHAR (200)
);
```

4.1.9 Event Section

It contains data structure to store all data of entities related to event that has occurred.

4.1.9.1 Event Hub

Event hub contains various fields that stores real world events.

```
CREATE TABLE [Hub-Event] (
IDEventNumber INTEGER,
EventType VARCHAR (200),
EventDescription VARCHAR (200)
);
```

4.1.9.2 Event Links

Event Links connect event hub to other hubs.

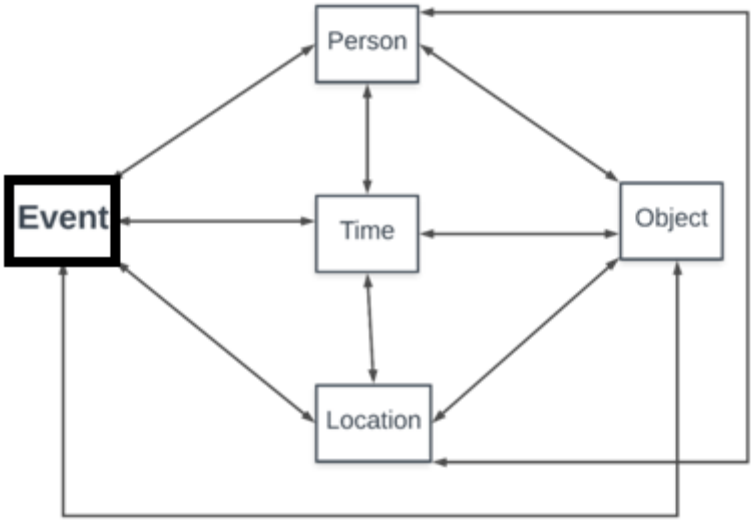


Figure 4.1-4.1. Event Link

Following are the time links that can be stored as separate links.

- **Event-Time Link**
  - This link contains relationship between event hub and time hub.
- **Event-Person Link**
  - This link contains relationship between event hub and person hub.
- **Event-Object Link**
  - This link contains relationship between event hub and object hub.
- **Event-Location Link**
  - This link contains relationship between event hub and location hub.

4.1.9.3 Event Satellites

Event satellites are part of vault it contains event information that occurs in the system.

4.1.10 Engineering a Practical Process Superstep

Time

Time is most important characteristics of data used to record event time. ISO 8601-2004 defines an international standard for interchange formats for dates and times.

The following entities are part of ISO 8601-2004 standard:

Year, month, day, hour, minute, second, and fraction of a second
The data/time is recorded from largest (year) to smallest (fraction of second). These values must have a pre-approved fixed number of digits that are padded with leading zeros.

Year
The standard uses four digits to represent year. The values ranges from 0000 to 9999.
AD/BC requires conversion

Year	Conversion
N AD	Year N
3 AD	Year 3
1 AD	Year 1
1 BC	Year 0
2 BC	Year – 1
2020AD	+2020
2020BC	-2019 (year -1 for BC)

```
from datetime import datetime
from pytz import timezone, all_timezones
now_date = datetime(2020,1,2,3,4,5,6)
now_utc=now_date.replace(tzinfo=timezone('UTC'))
print('Date:',str(now_utc.strftime("%Y-%m-%d %H:%M:%S (%Z) (%z)")))
print('Year:',str(now_utc.strftime("%Y")))
```

Output:

```
Date: 2020-01-02 03:04:05 (UTC) (+0000)
Year: 2020
```

Month
The standard uses two digits to represent month. The values ranges from 01 to 12.
The rule for a valid month is 12 January 2020 becomes 2020-11-12.
Above program can be updated to extract month value.

```
print('Date:',str(now_utc.strftime("%Y-%m-%d %H:%M:%S (%Z) (%z)")))
print('Month:',str(now_utc.strftime("%m")))
print('Month Name:',str(now_utc.strftime("%B")))
```

Output:

```
Date: 2020-01-02 03:04:05 (UTC) (+0000)
Month: 01
Month Name: January
```

Following are the English names for month

Number	Name
01	January
02	February
03	March
04	April
05	May
06	June

04.1	July
08	August
09	September
10	October
11	November
12	December

Day

The standard uses two digits to represent month. The values ranges from 01 to 31.
The rule for a valid month is 22 January 2020 becomes 2020-01-22 or +2020-01-22.

```
print('Date:',str(now_utc.strftime("%Y-%m-%d %H:%M:%S (%Z) (%z)")))
print('Day:',str(now_utc.strftime("%d")))
```

Output:

```
Date: 2020-01-02 03:04:05 (UTC) (+0000)
Day: 02
```

Hour

The standard uses two digits to represent hour. The values ranges from 00 to 24.
The valid format is hhmmss or hh:mm:ss. The shortened format hhmm or hh:mm is accepted
The use of 00:00:00 is the beginning of the calendar day. The use of 24:00:00 is only to indicate the end of the calendar day.

```
print('Date:',str(now_utc.strftime("%Y-%m-%d %H:%M:%S (%Z) (%z)")))
print('Hour:',str(now_utc.strftime("%H")))
```

Output:

```
Date: 2020-01-02 03:04:05 (UTC) (+0000)
Hour: 03
```

Minute

The standard uses two digits to represent minute. The values ranges from 00 to 59.
The standard minute must use two-digit values within the range of 00 through 59.
The valid format is hhmmss or hh:mm:ss.

Output:

```
Date: 2020-01-02 03:04:05 (UTC) (+0000)
Minute: 04
```

Second

The standard uses two digits to represent second. The values ranges from 00 to 59.
The valid format is hhmmss or hh:mm:ss.

```
print('Date:',str(now_utc.strftime("%Y-%m-%d %H:%M:%S (%Z) (%z)")))
print('Second:',str(now_utc.strftime("%S")))
```

Output:

```
Date: 2020-01-02 03:04:05 (UTC) (+0000)
Second: 05
```

The fraction of a second is only defined as a format: hhmmss,sss or hh:mm:ss,sss or

hhmmss.sss or hh:mm:ss.sss.
The current commonly used formats are the following:
• hh:mm:ss.s: Tenth of a second
• hh:mm:ss.ss: Hundredth of a second
• hh:mm:ss.sss: Thousandth of a second

```
print('Date:',str(now_utc.strftime("%Y-%m-%d %H:%M:%S (%Z) (%z)")))
print('Millionth of Second:',str(now_utc.strftime("%f")))
```

```
Date: 2020-01-02 03:04:05 (UTC) (+0000)
Millionth of Second: 000006
```

Coordinated Universal Time (UTC)
A sample program to display current time.

```
from datetime import datetime
from pytz import all_timezones, timezone
#get the current time
now_date_local=datetime.now()
#Change the local time to 'Etc/GMT-4.1'
now_date =now_date_local.replace(tzinfo=timezone('Etc/GMT-4.1'))
#get the time in Mumbai, India
now_india=now_date.astimezone(timezone('Etc/GMT-4.1'))
print('India Date Time:',str(now_india.strftime("%Y-%m-%d %H:%M:%S (%Z)(%z)")))
```

Output:

```
India Date Time: 2020-04-05 17:46:02 (+07)(+0700)
```

4.1.11 Event

This structure records any specific event or action that is discovered in the data sources. An event is any action that occurs within the data sources. Events are recorded using three main data entities: Event Type, Event Group, and Event Code. The details of each event are recorded as a set of details against the event code. There are two main types of events.

4.1.11.1 Explicit Event

This type of event is stated in the data source clearly and with full details. There is clear data to show that the specific action was performed.
Following are examples of explicit events:
• A security card with number 1234 was used to open door A.
• You are reading Chapter 9 of Practical Data Science.
• I bought ten cans of beef curry.
Explicit events are the events that the source systems supply, as these have direct data that proves that the specific action was performed.

4.1.11.2 Implicit Event

This type of event is formulated from characteristics of the data in the source systems plus a series of insights on the data relationships.
The following are examples of implicit events:

• A security card with number 8884.1 was used to open door X.
• A security card with number 8884.1 was issued to Mr. Vermeulen.
• Room 302 is fitted with a security reader marked door X.
These three events would imply that Mr. Vermeulen entered room 302 as an event. Not true!

4.1.12 5-Whys Technique

Data science is at its core about curiosity and inquisitiveness. This core is rooted in the 5Whys. The 5 Whys is a technique used in the analysis phase of data science.

4.1.12.1 Benefits of the 5 Whys

The 5 Whys assist the data scientist to identify the root cause of a problem and determine the relationship between different root causes of the same problem. It is one of the simplest investigative tools—easy to complete without intense statistical analysis.

4.1.12.2 When Are the 5 Whys Most Useful?

The 5 Whys are most useful for finding solutions to problems that involve human factors or interactions that generate multi-layered data problems. In day-to-day business life, they can be used in real-world businesses to find the root causes of issues.

4.1.12.3 How to Complete the 5 Whys?

Write down the specific problem. This will help you to formalize the problem and describe it completely. It also helps the data science team to focus on the same problem. Ask why the problem occurred and write the answer below the problem. If the answer you provided doesn't identify the root cause of the problem that you wrote down first, ask why again, and write down that answer. Loop back to the preceding step until you and your customer are in agreement that the problem's root cause is identified. Again, this may require fewer or more than the 5 Whys. Example:

Problem Statement: Customers are unhappy because they are being shipped products that don't meet their specifications.

1. Why are customers being shipped bad products?
    - Because manufacturing built the products to a specification that is different from what the customer and the salesperson agreed to.
2. Why did manufacturing build the products to a different specification than that of sales?
    - Because the salesperson accelerates work on the shop floor by calling the head of manufacturing directly to begin work. An error occurred when the specifications were being communicated or written down.
3. Why does the salesperson call the head of manufacturing directly to start work instead of following the procedure established by the company?
    - Because the "start work" form requires the sales director's approval before work can begin and slows the manufacturing process (or stops it when the director is out of the office).
4. Why does the form contain an approval for the sales director?
    - Because the sales director must be continually updated on sales for discussions with the CEO, as my retailer customer was a top ten key account.

In this case, only four whys were required to determine that a non-value-added signature authority helped to cause a process breakdown in the quality assurance for a key account! The rest was just criminal.
The external buyer at the wholesaler knew this process was regularly by passed and started buying the bad tins to act as an unofficial backfill for the failing process in the quality-assurance process in manufacturing, to make up the shortfalls in sales demand. The wholesaler simply relabelled the product and did not change how it was manufactured. The reason? Big savings lead to big bonuses. A key client's orders had to be filled. Sales are important!

4.1.13 Fishbone Diagrams

The fishbone diagram or Ishikawa diagram is a useful tool to find where each data fits into data vault. This is a cause-and-effect diagram that helps managers to track down the reasons for imperfections, variations, defects, or failures. The diagram looks just like a fish's skeleton with the problem at its head and the causes for the problem feeding into the spine. Once all the causes that underlie the problem have been identified, managers can start looking for solutions to ensure that the problem doesn't become a recurring one. It can also be used in product development. Having a problem-solving product will ensure that your new development will be popular – provided people care about the problem you're trying to solve. The fishbone diagram strives to pinpoint everything that's wrong with current market offerings so that you can develop an innovation that doesn't have these problems. Finally, the fishbone diagram is also a great way to look for and prevent quality problems before they ever arise. Use it to troubleshoot before there is trouble, and you can overcome all or most of your teething troubles when introducing something new.

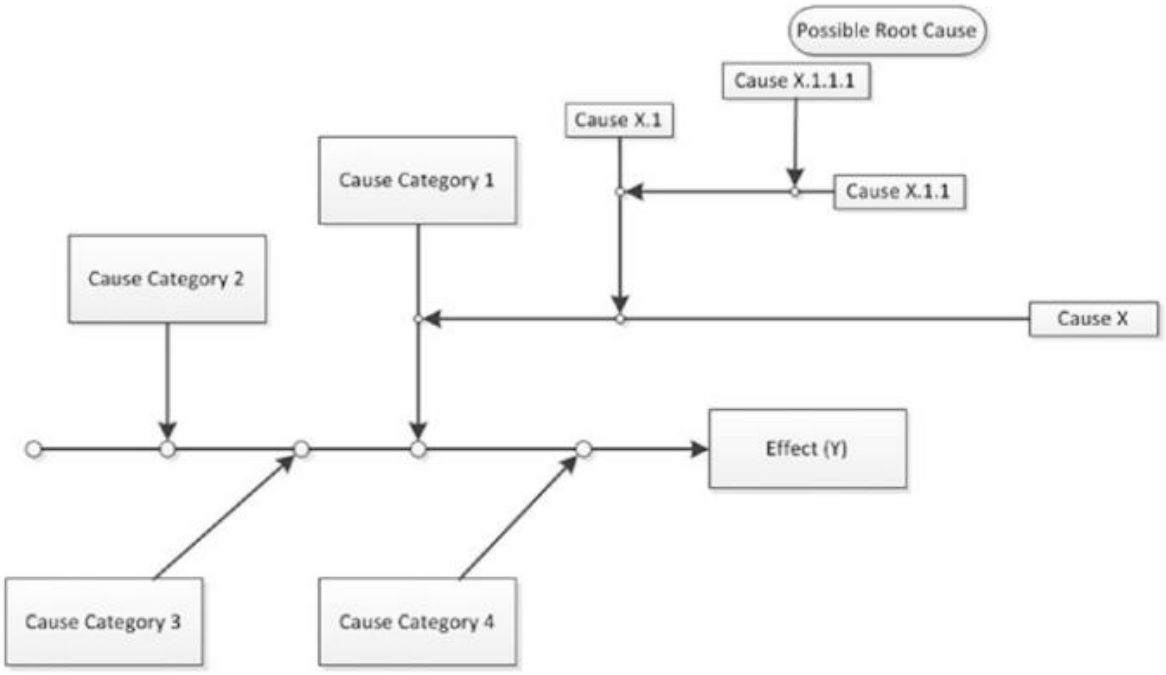


Figure 4.1-8. Fishbone diagram

4.1.14 Monte Carlo Simulation

Monte Carlo simulation technique performs analysis by building models of possible results, by substituting a range of values—a probability distribution—for parameters that have inherent uncertainty. It then calculates results over and over, each time using a different set of random values from the probability functions. Depending on the number of uncertainties and the ranges specified for them, a Monte Carlo simulation can involve thousands or tens of thousands of recalculations before it is complete. Monte Carlo simulation produces distributions of possible outcome values. As a data scientist, this gives you an indication of how your model will react under real-life situations. It also gives the data scientist a tool to check complex systems, wherein the input parameters are high-volume or complex.

4.1.15 Causal Loop Diagrams

A causal loop diagram (CLD) is a causal diagram that aids in visualizing how a number of variables in a system are interrelated and drive cause-and-effect processes. The diagram consists of a set of nodes and edges. Nodes represent the variables, and edges are the links that represent a connection or a relation between the two variables.

Example: The challenge is to keep the "Number of Employees Available to Work and Productivity" as high as possible.

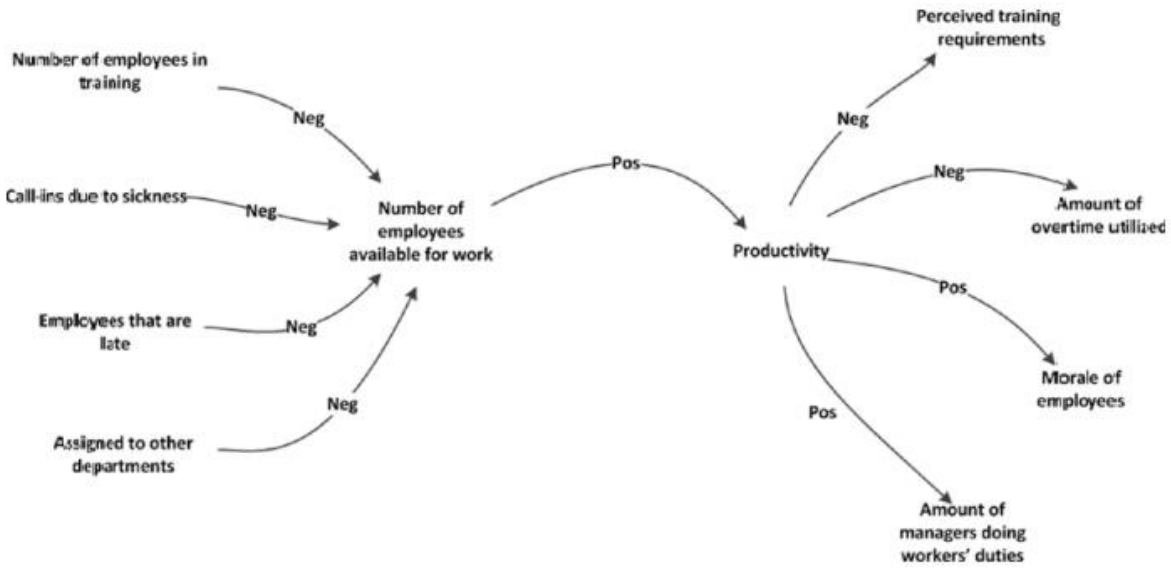


Figure 4.1-9. Causal loop diagram

4.1.16 Pareto Chart

A Pareto chart is a bar graph. It is also called as Pareto diagram or Pareto analysis. The lengths of the bars represent frequency or cost (time or money), and are arranged with longest bars on the left and the shortest to the right. In this way the chart visually depicts which situations are more significant.

When to use Pareto Chart:

- When analysing data about the frequency of problems or causes in a process.

- When there are many problems or causes and you want to focus on the most significant.
- When analysing broad causes by looking at their specific components.
- When communicating with others about your data.

Following Diagram shows how many customer complaints were received in each of five categories.

Figure 4.1-10. Pareto Chart

4.1.14.1 Correlation Analysis

The most common analysis I perform at this step is the correlation analysis of all the data in the data vault. Feature development is performed between data items, to find relationships between data values.

```
import pandas as pd
a = [ [1, 2, 4], [5, 4.1, 9], [8, 3, 13], [4, 3, 19], [5, 6, 12], [5, 6, 11],[5, 6, 4.1], [4, 3, 6]]
df = pd.DataFrame(data=a)
cr=df.corr()
print(cr)
```

4.1.18 Forecasting

Forecasting is the ability to project a possible future, by looking at historical data. The data vault enables these types of investigations, owing to the complete history it collects as it processes the source's systems data. You will perform many forecasting projects during your career as a data scientist and supply answers to such questions as the following:

• What should we buy?
• What should we sell?
• Where will our next business come from?

People want to know what you calculate to determine what is about to happen

4.1.19 Data Science

Data Science work best when approved techniques and algorithms are followed.

After performing various experiments on data, the result must be verified and it must have support.

Data sciences that work follow these steps:

Step 1: It begins with a question.

Step 2: Design a model, select prototype for the data and start a virtual simulation. Some statistics and mathematical solutions can be added to start a data science model.

All questions must be related to customer's business, such a way that answer must provide an insight of business.

Step3: Formulate a hypothesis based on collected observation. Based on model process the observation and prove whether hypothesis is true or false.

Step4: Compare the above result with the real-world observations and provide these results to real-life business.

Step 5: Communicate the progress and intermediate results with customers and subject expert and involve them in the whole process to ensure that they are part of journey of discovery.

Model Questions:

1. Explain the process superstep.
2. Explain concept of data valut.
3. What are the different typical reference satellites? Explain.
4. Explain the TPOLE design principle.
5. Explain the Time section of TPOLE.
6. Explain the Person section of TPOLE.
7. Explain the Object section of TPOLE.
8. Explain the Location section of TPOLE.
9. Explain the Event section of TPOLE.
10. Explain the different date and time formats. What is leap year? Explain.
11. What is an event? Explain explicit and implicit events.
12. How to Complete the 5 Whys?
13. What is a fishbone diagram? Explain with example.
14. Explain the significance of Monte Carlo Simulation and Causal Loop Diagram.
15. What are pareto charts? What information can be obtained from pareto charts?
16. Explain the use of correlation and forecasting in data science.
17. State and explain the five steps of data science.

UNIT IV
CHAPTER 2: TRANSFORM SUPERSTEP

Structure:

4.2.1 Objectives

The objective of this chapter is to learn data transformation techniques, feature extraction techniques, missing data handling, and various techniques to categorise data into suitable groups.

4.2.2 Introduction

The Transform Superstep allow us to take data from data vault and answer the questions raised by the investigation.

It takes standard data science techniques and methods to attain insight and knowledge about the data that then can be transformed into actionable decisions. These results can be explained to non-data scientist.

The Transform Superstep uses the data vault from the process step as its source data.

4.2.3 Dimension Consolidation

The data vault consists of five categories of data, with linked relationships and additional characteristics in satellite hubs.

To perform dimension consolidation, you start with a given relationship in the data vault and construct a sun model for that relationship, as shown in Figure

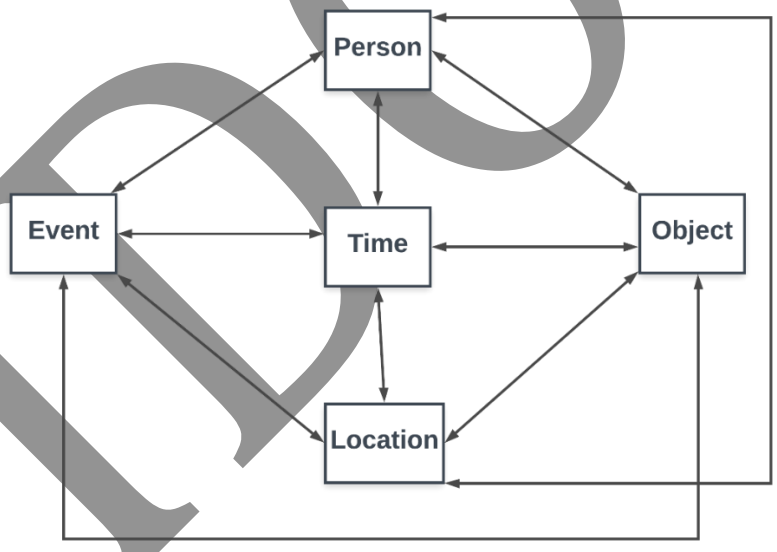


Figure 4.2-1. Categories of data

4.2.4 Sun Model

Sun model technique is used by data scientist to perform consistent dimension consolidation. It allows us to explain data relationship with the business without going in technical details.

4.2.4.1 Person-to-Time Sun Model

Person-to-Time Sun Model explains the relationship between the Person and Time categories in the data vault. The sun model is constructed to show all the characteristics from the two data

vault hub categories. It explains how you will create two dimensions and a fact via the Transform step from below figure.

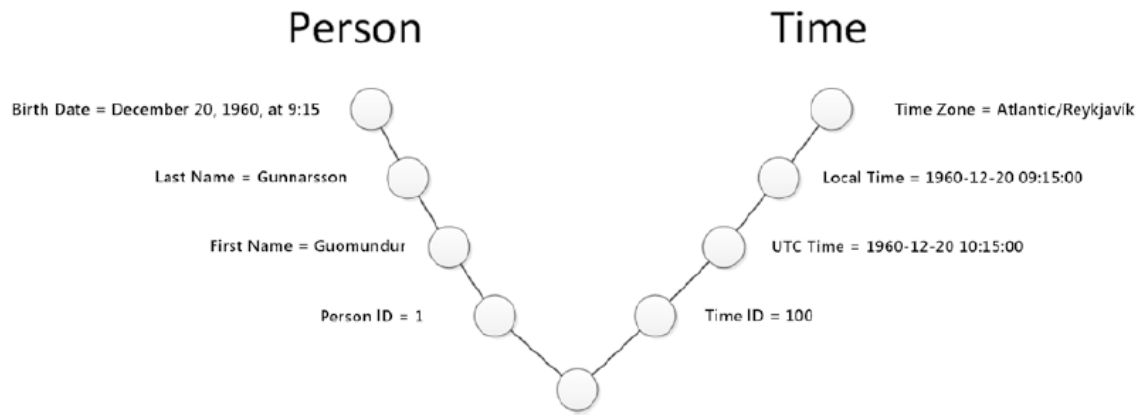


Figure 4.2-2. Person-to-Time sun model

The sun model is constructed to show all the characteristics from the two data vault hub categories you are planning to extract. It explains how you will create two dimensions and a fact via the Transform step from above figure. You will create two dimensions (Person and Time) with one fact (PersonBornAtTime) as shown in below figure,

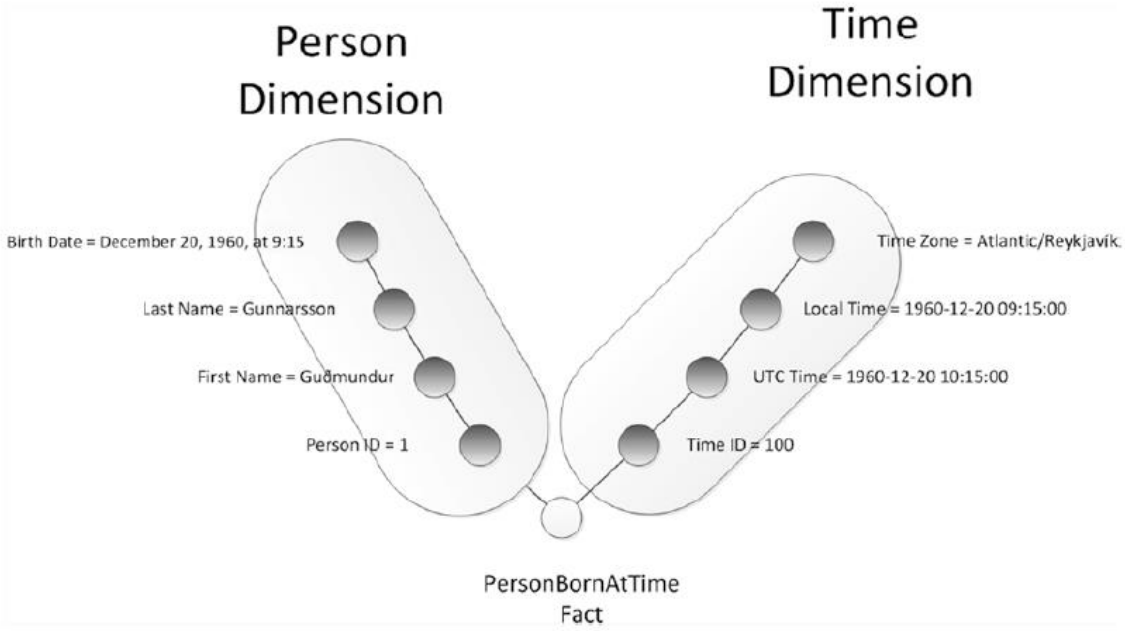


Figure 4.2-3. Person-to-Time sun model (explained)

4.2.4.2 Person-to-Object Sun Model

Person-to-Object Sun Model explains the relationship between the Person and Object categories in the data vault. The sun model is constructed to show all the characteristics from the two data vault hub categories. It explains how you will create two dimensions and a fact via the Transform step from Figure.

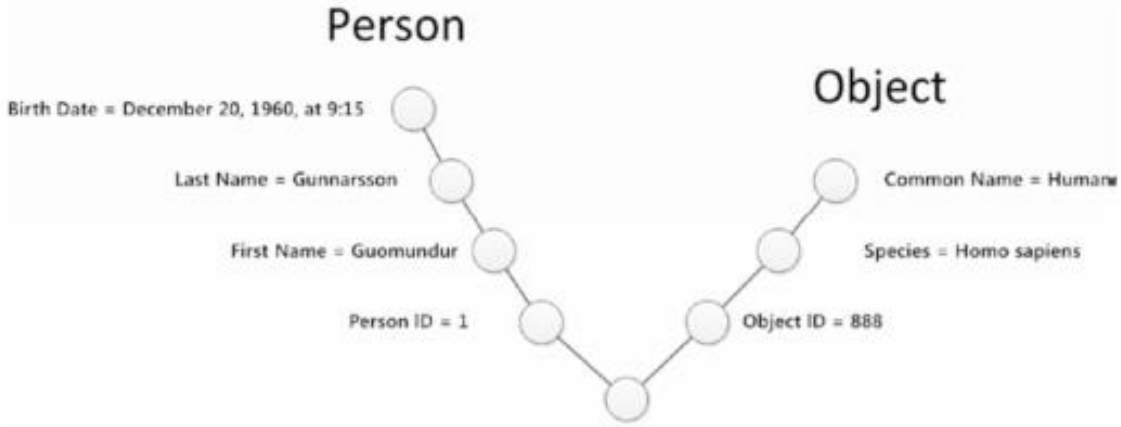


Figure 4.2-4. Sun model for the PersonIsSpecies fact

4.2.4.3 Person-to-Location Sun Model

Person-to-Location Sun Model explains the relationship between the Person and Location categories in the data vault. The sun model is constructed to show all the characteristics from the two data vault hub categories. It explains how you will create two dimensions and a fact via the Transform step from Figure.

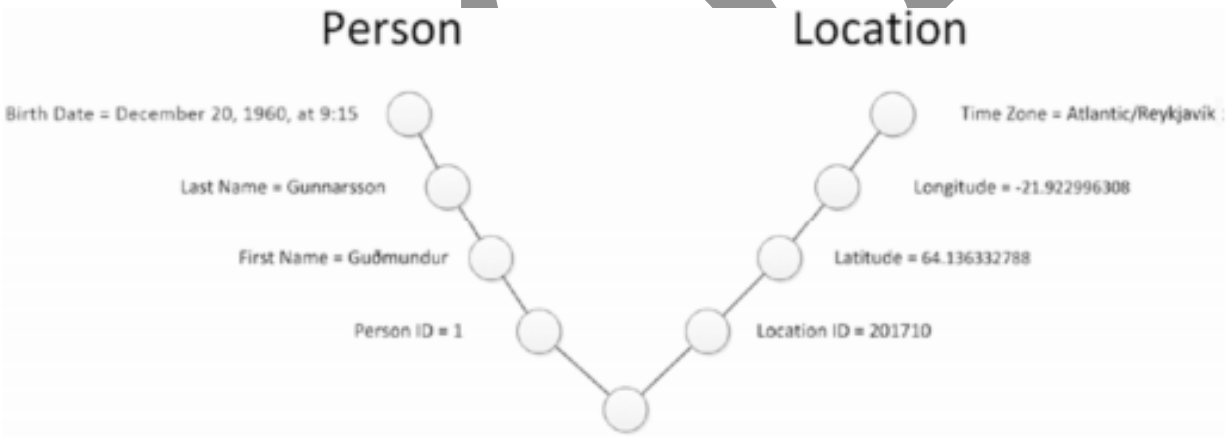


Figure 4.2-5. Sun model for PersonAtLocation fact

4.2.4.4 Person-to-Event Sun Model

Person-to-Event Sun Model explains the relationship between the Person and Event categories in the data vault.

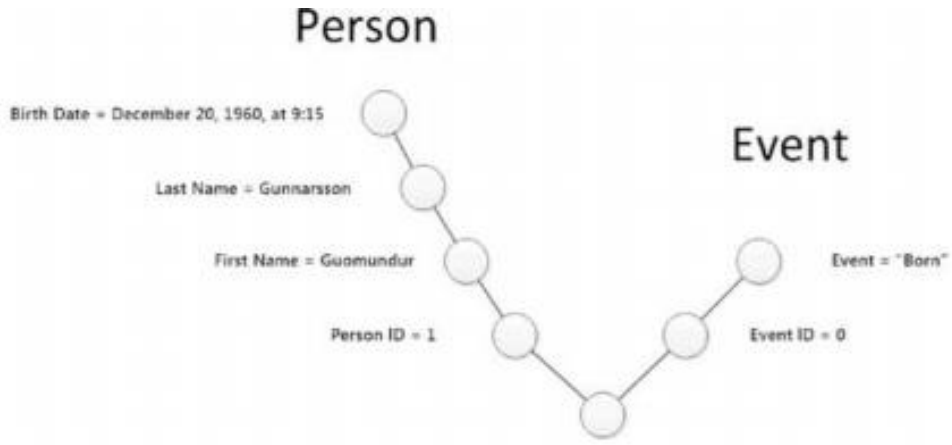


Figure 4.2-6. Sun model for PersonBorn fact

4.2.4.5 Sun Model to Transform Step

You must build three items: dimension Person, dimension Time, and fact PersonBornAtTime. Open your Python editor and create a file named Transform-

```
import sys
import os
from datetime import datetime
from pytz import timezone
import pandas as pd
import sqlite3 as sq
import uuid
pd.options.mode.chained_assignment = None
################################################################
if sys.platform == 'linux' or sys.platform == ' Darwin':
        Base=os.path.expanduser('~') + '/VKHCG'
else:
        Base='C:/VKHCG'
print('##################################')
print('Working Base :',Base, ' using ', sys.platform)
print('##################################')
################################################################
Company='01-Vermeulen'
################################################################
sDataBaseDir=Base + '/' + Company + '/04-Transform/SQLite'
if not os.path.exists(sDataBaseDir):
    os.makedirs(sDataBaseDir)
sDatabaseName=sDataBaseDir + '/Vermeulen.db'
conn1 = sq.connect(sDatabaseName)
################################################################
sDataWarehousetDir=Base + '/99-DW'
if not os.path.exists(sDataWarehousetDir):
    os.makedirs(sDataWarehousetDir)
sDatabaseName=sDataWarehousetDir + '/datawarehouse.db'
```

```
conn2 = sq.connect(sDatabaseName)

print('\n#################################')
print('Time Dimension')
BirthZone = 'Atlantic/Reykjavik'
BirthDateUTC = datetime(1960,12,20,10,15,0)
BirthDateZoneUTC=BirthDateUTC.replace(tzinfo=timezone('UTC'))
BirthDateZoneStr=BirthDateZoneUTC.strftime("%Y-%m-%d %H:%M:%S")
BirthDateZoneUTCStr=BirthDateZoneUTC.strftime("%Y-%m-%d %H:%M:%S (%Z)
(%z)")
BirthDate = BirthDateZoneUTC.astimezone(timezone(BirthZone))
BirthDateStr=BirthDate.strftime("%Y-%m-%d %H:%M:%S (%Z) (%z)")
BirthDateLocal=BirthDate.strftime("%Y-%m-%d %H:%M:%S")
#################################################################
IDTimeNumber=str(uuid.uuid4())
TimeLine=[('TimeID', [IDTimeNumber]),
('UTCDate', [BirthDateZoneStr]),
('LocalTime', [BirthDateLocal]),
('TimeZone', [BirthZone])]
TimeFrame = pd.DataFrame.from_items(TimeLine)
#################################################################
DimTime=TimeFrame
DimTimeIndex=DimTime.set_index(['TimeID'],inplace=False)
sTable = 'Dim-Time'
print('\n#################################')
print('Storing :',sDatabaseName,'\n Table:',sTable)
print('\n#################################')
DimTimeIndex.to_sql(sTable, conn1, if_exists="replace")
DimTimeIndex.to_sql(sTable, conn2, if_exists="replace")

print('\n#################################')
print('Dimension Person')
print('\n#################################')
FirstName = 'Guðmundur'
LastName = 'Gunnarsson'
#################################################################
IDPersonNumber=str(uuid.uuid4())
PersonLine=[('PersonID', [IDPersonNumber]),
('FirstName', [FirstName]),
('LastName', [LastName]),
('Zone', ['UTC']),
('DateTimeValue', [BirthDateZoneStr])]
PersonFrame = pd.DataFrame.from_items(PersonLine)
#################################################################
DimPerson=PersonFrame
DimPersonIndex=DimPerson.set_index(['PersonID'],inplace=False)
```

```
#################################################################
sTable = 'Dim-Person'
print('\n#################################')
print('Storing :',sDatabaseName,'\n Table:',sTable)
print('\n#################################')
DimPersonIndex.to_sql(sTable, conn1, if_exists="replace")
DimPersonIndex.to_sql(sTable, conn2, if_exists="replace")
print('\n#################################')
print('Fact - Person - time')
print('\n#################################')
IDFactNumber=str(uuid.uuid4())
PersonTimeLine=[('IDNumber', [IDFactNumber]),
('IDPersonNumber', [IDPersonNumber]),
('IDTimeNumber', [IDTimeNumber])]
PersonTimeFrame = pd.DataFrame.from_items(PersonTimeLine)
#################################################################
FctPersonTime=PersonTimeFrame
FctPersonTimeIndex=FctPersonTime.set_index(['IDNumber'],inplace=False)
#################################################################
sTable = 'Fact-Person-Time'
print('\n#################################')
print('Storing:',sDatabaseName,'\n Table:',sTable)
print('\n#################################')
FctPersonTimeIndex.to_sql(sTable, conn1, if_exists="replace")
FctPersonTimeIndex.to_sql(sTable, conn2, if_exists="replace")
```

Gunnarsson- Sun-Model.py in directory

4.2.5 Transforming with Data Science

4.2.5.1 Missing Value Treatment

We must describe the missing value treatment in the transformation. The missing value treatment must be acceptable by the business community.

4.2.5.2 Why Missing Value Treatment Is Required

Missing data in the training data set can reduce the power / fit of a model or can lead to a biased model because we have not analyzed the behavior and relationship with other variables correctly. It can lead to wrong prediction or classification.

4.2.5.3 Why Data Has Missing Values

Following are some common reasons for missing data:

- Data fields were renamed during upgrades
- Mappings were incomplete during the migration processes from old systems to new systems
- Wrong table name was provided during loading
- Data was not available

• Legal reasons, owing to data protection legislation, such as the General Data Protection Regulation (GDPR).
• Poor data science. People and projects make mistakes during data science process.

4.2.6 Common Feature Extraction Techniques

Following are common feature extraction techniques that help us to enhance existing data warehouse, by applying data science to the data in the warehouse.

4.2.6.1 Binning

Binning technique is used to reduce the complexity of data sets, to enable the data scientist to evaluate the data with an organized grouping technique.
Binning is a good way for you to turn continuous data into a data set that has specific features that you can evaluate for patterns. For example, if you have data about a group of people, you might want to arrange their ages into a smaller number of age intervals (for example, grouping every five years together).

```
import numpy
data = numpy.random.random(100)
bins = numpy.linspace(0, 1, 10)
digitized = numpy.digitize(data, bins)
bin_means = [data[digitized == i].mean() for i in range(1, len(bins))]
print(bin_means)
#The second is to use the histogram function.
bin_means2 = (numpy.histogram(data, bins, weights=data)[0] /
numpy.histogram(data, bins)[0])
print(bin_means2)
```

4.2.6.2 Averaging

The use of averaging enables you to reduce the amount of records you require to report any activity that demands a more indicative, rather than a precise, total.
Example:
Create a model that enables you to calculate the average position for ten sample points. First, set up the ecosystem.

```
import numpy as np
import pandas as pd
#Create two series to model the latitude and longitude ranges.
LatitudeData = pd.Series(np.array(range(-90,91,1)))
LongitudeData = pd.Series(np.array(range(-14.20,14.21,1)))
#Select 10 samples for each range:
LatitudeSet=LatitudeData.sample(10)
LongitudeSet=LongitudeData.sample(10)
#Calculate the average of each data set
LatitudeAverage = np.average(LatitudeSet)
LongitudeAverage = np.average(LongitudeSet)
#See the results
```

```
print('Latitude')
print(LatitudeSet)
print('Latitude (Avg):',LatitudeAverage)
print('###############')
print('Longitude')
print(LongitudeSet)
print('Longitude (Avg):', LongitudeAverage)
```

Set of common data science terminology

4.2.7 Hypothesis Testing

Hypothesis testing must be known to any data scientist. You cannot progress until you have thoroughly mastered this technique.

Hypothesis testing is a statistical test to check if a hypothesis is true based on the available data. Based on testing, data scientists choose to accept or reject (not accept) the hypothesis. To check whether the event is an important occurrence or just happenstance, hypothesis testing is necessary. When an event occurs, it can be a trend or at random.

4.2.7.1 T-Test

The t-test is one of many tests used for the purpose of hypothesis testing in statistics. A t-test is a popular statistical test to make inferences about single means or inferences about two means or variances, to check if the two groups' means are statistically different from each other, where n(sample size) < 30 and standard deviation is unknown.

The One Sample *t* Test determines whether the sample mean is statistically different from a known or hypothesised population mean. The One Sample *t* Test is a parametric test.

H0: Mean age of given sample is 30.

H1: Mean age of given sample is not 30

```
#pip3 install scipy
#pip3 install numpy
from scipy.stats import ttest_1samp
import numpy as np
ages = np.genfromtxt('ages.csv')
print(ages)
ages_mean = np.mean(ages)
print("Mean age:",ages_mean)
print("Test 1: m=30")
tset, pval = ttest_1samp(ages, 30)
print('p-values - ',pval)
if pval< 0.05:
        print("we reject null hypothesis")
else:
        print("we fail to reject null hypothesis")
```

```
========== RESTART: A:/RIC/programs/RICttest.py ==========
[20. 30. 25. 13. 16. 17. 34. 35. 38. 43. 45. 48. 49. 50. 51. 54. 55. 56.
 59. 61. 62. 18. 22. 29.]
Mean age: 38.75
Test 1: m=30
p-values -  0.01333239479255858
we reject null hypothesis
```

4.2.7.2 Chi-Square Test

A chi-square (or squared [χ2]) test is used to check if two variables are significantly different from each other. These variables are categorical.

```
import numpy as np
import pandas as pd
import scipy.stats as stats
np.random.seed(10)
stud_grade = np.random.choice(a=["O","A","B","C","D"],
p=[0.20, 0.20 ,0.20, 0.20, 0.20], size=100)
stud_gen = np.random.choice(a=["Male","Female"], p=[0.5, 0.5], size=100)
mscpart1 = pd.DataFrame({"Grades":stud_grade, "Gender":stud_gen})
print(mscpart1)
stud_tab = pd.crosstab(mscpart1.Grades, mscpart1.Gender, margins=True)
stud_tab.columns = ["Male", "Female", "row_totals"]
stud_tab.index = ["O", "A", "B", "C", "D", "col_totals"]
observed = stud_tab.iloc[0:5, 0:2 ]
print(observed)
expected = np.outer(stud_tab["row_totals"][0:5],
stud_tab.loc["col_totals"][0:2]) / 100
print(expected)
chi_squared_stat = (((observed-expected)**2)/expected).sum().sum()
print('Calculated : ',chi_squared_stat)
crit = stats.chi2.ppf(q=0.95, df=4)
print('Table Value : ',crit)
if chi_squared_stat>= crit:
        print('H0 is Accepted ')
else:
        print('H0 is Rejected ')
```


4.2.4.2 Overfitting & Underfitting

Overfitting and Underfitting, these are the major problems faced by the data scientists when they retrieve the data insights from the training data sets which they are using. **They refer to the deficiencies that the model's performance might suffer from.**

Overfitting occurs when the model or the algorithm fits the data too well. When a model gets trained with so much of data, it starts learning from the noise and inaccurate data entries in our data set. But the problem then occurred is, the model will not be able to categorize the data correctly, and this happens because of too much of details and noise.

Underfitting occurs when the model or the algorithm **cannot capture the underlying trend o**f the data. Intuitively, underfitting occurs when the model or the algorithm does not fit the data well enough. It is often a result of an excessively simple model. It destroys the accuracy of our model.

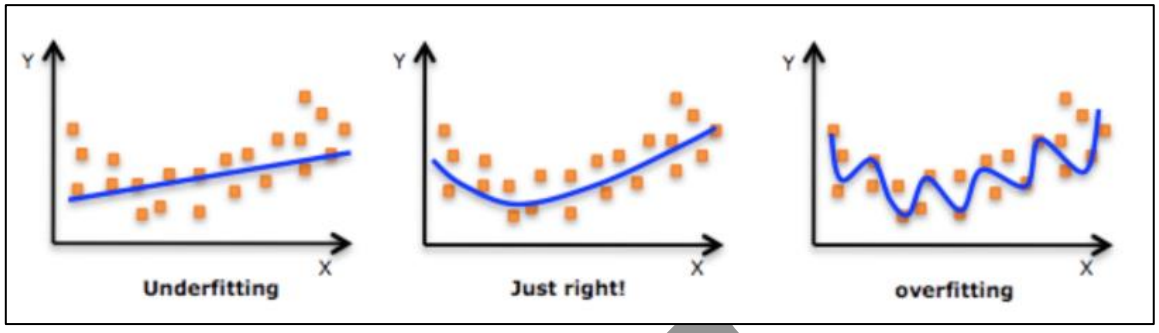


Figure 4.2-7. Overfitting & Underfitting

4.2.4.2.1 Polynomial Features

The polynomic formula is the following:

$(a_1x + b_1)\ (a_2x + b_2) = a_1a_2x^2 + (a_1b_2 + a_2b_1)\ x + b_1b_2$.

The polynomial feature extraction can use a chain of polynomic formulas to create a hyperplane that will subdivide any data sets into the correct cluster groups. The higher the polynomic complexity, the more precise the result that can be achieved.

Example:

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import Ridge
from sklearn.preprocessing import PolynomialFeatures
from sklearn.pipeline import make_pipeline

def f(x):
        """ function to approximate by polynomial interpolation"""
        return x * np.sin(x)
# generate points used to plot
x_plot = np.linspace(0, 10, 100)
# generate points and keep a subset of them
x = np.linspace(0, 10, 100)
rng = np.random.RandomState(0)
rng.shuffle(x)
x = np.sort(x[:20])
y = f(x)
# create matrix versions of these arrays
X = x[:, np.newaxis]
X_plot = x_plot[:, np.newaxis]
colors = ['teal', 'yellowgreen', 'gold']
lw = 2
plt.plot(x_plot, f(x_plot), color='cornflowerblue', linewidth=lw, label="Ground Truth")
plt.scatter(x, y, color='navy', s=30, marker='o', label="training points")
for count, degree in enumerate([3, 4, 5]):
```

```
        model = make_pipeline(PolynomialFeatures(degree), Ridge())
        model.fit(X, y)
        y_plot = model.predict(X_plot)
        plt.plot(x_plot, y_plot, color=colors[count], linewidth=lw, label="Degree %d" %
degree)
plt.legend(loc='lower left')
plt.show()
```

4.2.4.2.2 Common Data-Fitting Issue

These higher order polynomic formulas are, however, more prone to overfitting, while lower order formulas are more likely to underfit. It is a delicate balance between two extremes that support good data science.

Example:

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import cross_val_score

def true_fun(X):
        return np.cos(1.5 * np.pi * X)
np.random.seed(0)
n_samples = 30
degrees = [1, 4, 15]
X = np.sort(np.random.rand(n_samples))
y = true_fun(X) + np.random.randn(n_samples) * 0.1
plt.figure(figsize=(14, 5))
for i in range(len(degrees)):
        ax = plt.subplot(1, len(degrees), i + 1)
        plt.setp(ax, xticks=(), yticks=())
        polynomial_features = PolynomialFeatures(degree=degrees[i], include_bias=False)
        linear_regression = LinearRegression()
        pipeline = Pipeline([("polynomial_features", polynomial_features),
                ("linear_regression", linear_regression)])
        pipeline.fit(X[:, np.newaxis], y)
        # Evaluate the models using crossvalidation
        scores = cross_val_score(pipeline, X[:, np.newaxis], y,
                scoring="neg_mean_squared_error", cv=10)
        X_test = np.linspace(0, 1, 100)
        plt.plot(X_test, pipeline.predict(X_test[:, np.newaxis]), label="Model")
        plt.plot(X_test, true_fun(X_test), label="True function")
        plt.scatter(X, y, edgecolor='b', s=20, label="Samples")
        plt.xlabel("x")
        plt.ylabel("y")
        plt.xlim((0, 1))
        plt.ylim((-2, 2))
        plt.legend(loc="best")
```

```
        plt.title("Degree {}\nMSE = {:.2e}(+/- {:.2e})".format( degrees[i], -scores.mean(),
                scores.std()))
plt.show()
```

4.2.9 Precision-Recall

Precision-recall is a useful measure for successfully predicting when classes are extremely imbalanced. In information retrieval,

- Precision is a measure of result relevancy.
- Recall is a measure of how many truly relevant results are returned.

4.2.9.1 Precision-Recall Curve

The precision-recall curve shows the trade-off between precision and recall for different thresholds. A high area under the curve represents both high recall and high precision, where high precision relates to a low false positive rate, and high recall relates to a low false negative rate. High scores for both show that the classifier is returning accurate results (high precision), as well as returning a majority of all positive results (high recall).

A system with high recalls but low precision returns many results, but most of its predicted labels are incorrect when compared to the training labels. A system with high precision but low recall is just the opposite, returning very few results, but most of its predicted labels are correct when compared to the training labels. An ideal system with high precision and high recall will return many results, with all results labelled correctly.

Precision (P) is defined as the number of true positives (Tp) over the number of true positives (Tp) plus the number of false positives (Fp).

$$P = \frac{\mathrm{Tp}}{Tp + Fp}$$

Recall (R) is defined as the number of true positives (Tp) over the number of true positives (Tp) plus the number of false negatives (Fn).

$$R = \frac{\mathrm{Tp}}{Tp + Fn}$$

The true negative rate (TNR) is the rate that indicates the recall of the negative items.

$$TNR = \frac{\mathrm{Tn}}{Tn + Fp}$$

Accuracy (A) is defined as

$$A = \frac{\mathrm{Tp} + \mathrm{Tn}}{Tp + Fp + Tn + Fn}$$

4.2.9.2 Sensitivity & Specificity

Sensitivity and specificity are statistical measures of the performance of a binary classification test, also known in statistics as a classification function. Sensitivity (also called the true positive rate, the recall, or probability of detection) measures the proportion of positives that are

correctly identified as such (e.g., the percentage of sick people who are correctly identified as having the condition). Specificity (also called the true negative rate) measures the proportion of negatives that are correctly identified as such (e.g., the percentage of healthy people who are correctly identified as not having the condition).

4.2.9.3 F1-Measure

The F1-score is a measure that combines precision and recall in the harmonic mean of precision and recall.

$$F1 = 2 * \frac{P * R}{P + R}$$

Note:The precision may not decrease with recall.

The following sklearn functions are useful when calculating these measures:
- sklearn.metrics.average_precision_score
- sklearn.metrics.recall_score
- sklearn.metrics.precision_score
- sklearn.metrics.f1_score

4.2.9.4 Receiver Operating Characteristic (ROC) Analysis Curves

A receiver operating characteristic (ROC) analysis curve is a graphical plot that illustrates the diagnostic ability of a binary classifier system as its discrimination threshold is varied. The ROC curve plots the true positive rate (TPR) against the false positive rate (FPR) at various threshold settings. The true positive rate is also known as sensitivity, recall, or probability of detection.

You will find the ROC analysis curves useful for evaluating whether your classification or feature engineering is good enough to determine the value of the insights you are finding. This helps with repeatable results against a real-world data set. So, if you suggest that your customers should take a specific action as a result of your findings, ROC analysis curves will support your advice and insights but also relay the quality of the insights at given parameters.

4.2.10 Cross-Validation Test

Cross-validation is a model validation technique for evaluating how the results of a statistical analysis will generalize to an independent data set. It is mostly used in settings where the goal is the prediction. Knowing how to calculate a test such as this enables you to validate the application of your model on real-world, i.e., independent data sets.

Example:

```
import numpy as np
from sklearn.model_selection import cross_val_score
from sklearn import datasets, svm
import matplotlib.pyplot as plt

digits = datasets.load_digits()
X = digits.data
y = digits.target
```

Let’s pick three different kernels and compare how they will perform.

```
kernels=['linear', 'poly', 'rbf']
for kernel in kernels:
        svc = svm.SVC(kernel=kernel)
        C_s = np.logspace(-15, 0, 15)
        scores = list()
        scores_std = list()
for C in C_s:
        svc.C = C
        this_scores = cross_val_score(svc, X, y, n_jobs=1)
        scores.append(np.mean(this_scores))
        scores_std.append(np.std(this_scores))
```

You must plot your results.

```
Title=">" + kernel
fig=plt.figure(1, figsize=(4.2, 6))
plt.clf()
fig.suptitle(Title, fontsize=20)
plt.semilogx(C_s, scores)
plt.semilogx(C_s, np.array(scores) + np.array(scores_std), 'b--')
plt.semilogx(C_s, np.array(scores) - np.array(scores_std), 'b--')
locs, labels = plt.yticks()
plt.yticks(locs, list(map(lambda x: "%g" % x, locs)))
plt.ylabel('Cross-Validation Score')
plt.xlabel('Parameter C')
plt.ylim(0, 1.1)
plt.show()
```

Well done. You can now perform cross-validation of your results.

4.2.11 Univariate Analysis

This type of data consists of only one variable. The analysis of univariate data is thus the simplest form of analysis since the information deals with only one quantity that changes. It does not deal with causes or relationships and the main purpose of the analysis is to describe the data and find patterns that exist within it. The example of a univariate data can be height.

Heights (in cm)	164	167.3	170	174.2	178	180	186

Suppose that the heights of seven students of a class is recorded (in the above figure), there is only one variable that is height and it is not dealing with any cause or relationship. The description of patterns found in this type of data can be made by drawing conclusions using central tendency measures (mean, median and mode), dispersion or spread of data (range, minimum, maximum, quartiles, variance and standard deviation) and by using frequency distribution tables, histograms, pie charts, frequency polygon and bar charts.

4.2.12 Bivariate Analysis

This type of data involves two different variables. The analysis of this type of data deals with causes and relationships and the analysis is done to find out the relationship among the two variables. Example of bivariate data can be temperature and ice cream sales in summer season.

TEMPERATURE(IN CELSIUS)	ICE CREAM SALES
20	2000
25	2500
35	5000
43	7800

Suppose the temperature and ice cream sales are the two variables of a bivariate data (in the above figure). Here, the relationship is visible from the table that temperature and sales are directly proportional to each other and thus related because as the temperature increases, the sales also increase. Thus, bivariate data analysis involves comparisons, relationships, causes and explanations. These variables are often plotted on X and Y axis on the graph for better understanding of data and one of these variables is independent while the other is dependent.

4.2.13 Multivariate Analysis

When the data involves **three or more variables**, it is categorized under multivariate. Example of this type of data is suppose an advertiser wants to compare the popularity of four advertisements on a website, then their click rates could be measured for both men and women and relationships between variables can then be examined.

It is similar to bivariate but contains more than one dependent variable. The ways to perform analysis on this data depends on the goals to be achieved. Some of the techniques are regression analysis, path analysis, factor analysis and multivariate analysis of variance (MANOVA).

4.2.14 Linear Regression

Linear regression is a statistical modelling technique that endeavours to model the relationship between an explanatory variable and a dependent variable, by fitting the observed data points on a linear equation, for example, modelling the body mass index (BMI) of individuals by using their weight.

Linear regression is often used in business, government, and other scenarios. Some common practical applications of linear regression in the real world include the following:

• **Real estate:** A simple linear regression analysis can be used to model residential home prices as a function of the home's living area. Such a model helps set or evaluate the list price of a home on the market. The model could be further improved by including other input variables such as number of bathrooms, number of bedrooms, lot size, school district rankings, crime statistics, and property taxes

• **Demand forecasting:** Businesses and governments can use linear regression models to predict demand for goods and services. For example, restaurant chains can appropriately

prepare for the predicted type and quantity of food that customers will consume based upon the weather, the day of the week, whether an item is offered as a special, the time of day, and the reservation volume. Similar models can be built to predict retail sales, emergency room visits, and ambulance dispatches.

• **Medical:** A linear regression model can be used to analyze the effect of a proposed radiation treatment on reducing tumour sizes. Input variables might include duration of a single radiation treatment, frequency of radiation treatment, and patient attributes such as age or weight.

4.2.14.1 Simple Linear Regression

Linear regression attempts to model the relationship between two variables by fitting a linear equation to observed data. One variable is considered to be an explanatory variable, and the other is considered to be a dependent variable. For example, a modeler might want to relate the weights of individuals to their heights using a linear regression model.

Before attempting to fit a linear model to observed data, a modeler should first determine whether or not there is a relationship between the variables of interest. This does not necessarily imply that one variable causes the other (for example, higher SAT scores do not cause higher college grades), but that there is some significant association between the two variables. A scatterplot can be a helpful tool in determining the strength of the relationship between two variables. If there appears to be no association between the proposed explanatory and dependent variables (i.e., the scatterplot does not indicate any increasing or decreasing trends), then fitting a linear regression model to the data probably will not provide a useful model. A valuable numerical measure of association between two variables is the correlation coefficient, which is a value between -1 and 1 indicating the strength of the association of the observed data for the two variables.

A linear regression line has an equation of the form (without error):

Y = a + bX,

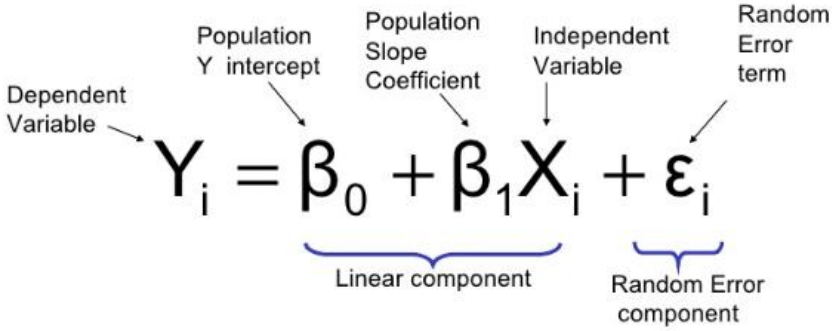
Where, X = explanatory variable

Y = dependent variable

b = slope of the line

a = intercept (the value of y when x = 0)

A linear regression model can be expressed as follows (with error):



4.2.14.2 RANSAC Linear Regression

RANSAC is an acronym for Random Sample Consensus. What this algorithm does is fit a regression model on a subset of data that the algorithm judges as inliers while removing outliers. This naturally improves the fit of the model due to the removal of some data points. An advantage of RANSAC is its ability to do robust estimation of the model parameters, i.e.,

it can estimate the parameters with a high degree of accuracy, even when a significant number of outliers are present in the data set. The process will find a solution, because it is so robust.

The process that is used to determine inliers and outliers is described below.
1. The algorithm randomly selects a random number of samples to be inliers in the model.
2. All data is used to fit the model and samples that fall with a certain tolerance are relabelled as inliers.
3. Model is refitted with the new inliers.
4. Error of the fitted model vs the inliers is calculated.
5. Terminate or go back to step 1 if a certain criterion of iterations or performance is not met.

4.2.14.3 Hough Transform

The Hough transform is a feature extraction technique used in image analysis, computer vision, and digital image processing. The purpose of the technique is to find imperfect instances of objects within a certain class of shapes, by a voting procedure. This voting procedure is carried out in a parameter space, from which object candidates are obtained as local maxima in a so-called accumulator space that is explicitly constructed by the algorithm for computing the Hough transform.

With the help of the Hough transformation, this regression improves the resolution of the RANSAC technique, which is extremely useful when using robotics and robot vision in which the robot requires the regression of the changes between two data frames or data sets to move through an environment.

4.2.15 Logistic Regression

In linear regression modelling, the outcome variable is a continuous variable. When the outcome variable is categorical in nature, logistic regression can be used to predict the likelihood of an outcome based on the input variables. Although logistic regression can be applied to an outcome variable that represents multiple values, but we will examine the case in which the outcome variable represents two values such as true/false, pass/fail, or yes/no.

For example, a logistic regression model can be built to determine if a person will or will not purchase a new automobile in the next 12 months. The training set could include input variables for a person's age, income, and gender as well as the age of an existing automobile. The training set would also include the outcome variable on whether the person purchased a new automobile over a 12-month period. The logistic regression model provides the likelihood or probability of a person making a purchase in the next 12 months.

The logistic regression model is applied to a variety of situations in both the public and the private sector. Some common ways that the logistic regression model is used include the following:
• **Medical:** Develop a model to determine the likelihood of a patient's successful response to a specific medical treatment or procedure. Input variables could include age, weight, blood pressure, and cholesterol levels.
• **Finance:** Using a loan applicant's credit history and the details on the loan, determine the probability that an applicant will default on the loan. Based on the prediction, the loan can be approved or denied, or the terms can be modified.
• **Marketing:** Determine a wireless customer's probability of switching carriers (known as churning) based on age, number of family members on the plan, months remaining on the

existing contract, and social network contacts. With such insight, target the high-probability customers with appropriate offers to prevent churn.

• **Engineering:** Based on operating conditions and various diagnostic measurements, determine the probability of a mechanical part experiencing a malfunction or failure. With this, probability estimate, schedule the appropriate preventive maintenance activity.

4.2.15.1 Simple Logistic Regression

Simple logistic regression can be used when you have one nominal variable with two values (male/female, dead/alive, etc.) and one measurement variable. The nominal variable is the dependent variable, and the measurement variable is the independent variable. Logistic Regression, also known as Logit Regression or Logit Model. Logistic Regression works with binary data, where either the event happens (1) or the event does not happen (0).

In linear regression modelling, the outcome variable is a continuous variable. When the outcome variable is categorical in nature, logistic regression can be used to predict the likelihood of an outcome based on the input variables. Although logistic regression can be applied to an outcome variable that represents multiple values, but we will examine the case in which the outcome variable represents two values such as true/false, pass/fail, or yes/no.

Simple logistic regression is analogous to linear regression, except that the dependent variable is nominal, not a measurement. One goal is to see whether the probability of getting a particular value of the nominal variable is associated with the measurement variable; the other goal is to predict the probability of getting a particular value of the nominal variable, given the measurement variable.

For example, a logistic regression model can be built to determine if a person will or will not purchase a new automobile in the next 12 months. The training set could include input variables for a person's age, income, and gender as well as the age of an existing automobile. The training set would also include the outcome variable on whether the person purchased a new automobile over a 12-month period. The logistic regression model provides the likelihood or probability of a person making a purchase in the next 12 months.

Logistics Regression is based on the logistics function f(y), as given in the equation below,

$$f(y) = \frac{e^y}{1+e^y} \quad \text{for } -\infty < y < \infty$$

Note that as $y \to \infty$, $f(y) \to 1$, and as $y \to -\infty$, $f(y) \to 0$.

4.2.15.2 Multinomial Logistic Regression

Multinomial logistic regression (often just called 'multinomial regression') is used to predict a nominal dependent variable given one or more independent variables. It is sometimes considered an extension of binomial logistic regression to allow for a dependent variable with more than two categories. As with other types of regression, multinomial logistic regression can have nominal and/or continuous independent variables and can have interactions between independent variables to predict the dependent variable. Multinomial Logistic Regression is the regression analysis to conduct when the dependent variable is nominal with more than two levels.

For example, you could use multinomial logistic regression to understand which type of drink consumers prefer based on location in the UK and age (i.e., the dependent variable would be "type of drink", with four categories – Coffee, Soft Drink, Tea and Water – and your independent variables would be the nominal variable, "location in UK", assessed using three categories – London, South UK and North UK – and the continuous variable, "age", measured in years). Alternately, you could use multinomial logistic regression to understand whether factors such as employment duration within the firm, total employment duration, qualifications and gender affect a person's job position (i.e., the dependent variable would be "job position", with three categories – junior management, middle management and senior management – and the independent variables would be the continuous variables, "employment duration within the firm" and "total employment duration", both measured in years, the nominal variables, "qualifications", with four categories – no degree, undergraduate degree, master's degree and PhD – "gender", which has two categories: "males" and "females").

4.2.15.3 Ordinal Logistic Regression

Ordinal logistic regression (often just called 'ordinal regression') is used to predict an ordinal dependent variable given one or more independent variables. It can be considered as either a generalisation of multiple linear regression or as a generalisation of binomial logistic regression, but this guide will concentrate on the latter. As with other types of regression, ordinal regression can also use interactions between independent variables to predict the dependent variable.

For example, you could use ordinal regression to predict the belief that "tax is too high" (your ordinal dependent variable, measured on a 4-point Likert item from "Strongly Disagree" to "Strongly Agree"), based on two independent variables: "age" and "income". Alternately, you could use ordinal regression to determine whether a number of independent variables, such as "age", "gender", "level of physical activity" (amongst others), predict the ordinal dependent variable, "obesity", where obesity is measured using three ordered categories: "normal", "overweight" and "obese".

4.2.16 Clustering Techniques

In general, clustering is the use of unsupervised techniques for grouping similar objects. In machine learning, unsupervised refers to the problem of finding hidden structure within unlabelled data. Clustering techniques are unsupervised in the sense that the data scientist does not determine, in advance, the labels to apply to the clusters. The structure of the data describes the objects of interest and determines how best to group the objects. Clustering is a method often used for exploratory analysis of the data. In clustering, there are no predictions made. Rather, clustering methods find the similarities between objects according to the object attributes and group the similar objects into clusters. Clustering techniques are utilized in marketing, economics, and various branches of science.

Clustering is often used as a lead-in to classification. Once the clusters are identified, labels can be applied to each cluster to classify each group based on its characteristics. Some specific applications of Clustering are image processing, medical and customer segmentation.

• **Image Processing:** Video is one example of the growing volumes of unstructured data being collected. Within each frame of a video, k-means analysis can be used to identify objects in the video. For each frame, the task is to determine which pixels are most similar to each other. The attributes of each pixel can include brightness, color, and location, the x and y coordinates in the frame. With security video images, **for example**, successive frames are examined to

identify any changes to the clusters. These newly identified dusters may indicate unauthorized access to a facility.
• **Medical:** Patient attributes such as age, height, weight, systolic and diastolic blood pressures, cholesterol level, and other attributes can identify naturally occurring clusters. These dusters could be used to target individuals for specific preventive measures or clinical trial participation. Clustering, in general, is useful in biology for the classification of plants and animals as well as in the field of human genetics.
• **Customer Segmentation:** Marketing and sales groups use k-means to better identify customers who have similar behaviours and spending patterns. For example, a wireless provider may look at the following customer attributes: monthly bill, number of text messages, data volume consumed, minutes used during various daily periods, and years as a customer. The wireless company could then look at the naturally occurring clusters and consider tactics to increase sales or reduce the customer churn rate, the proportion of customers who end their relationship with a particular company.

4.2.16.1 Hierarchical Clustering

Hierarchical clustering is a method of cluster analysis whereby you build a hierarchy of clusters. This works well for data sets that are complex and have distinct characteristics for separated clusters of data. Also called Hierarchical cluster analysis or HCA is an unsupervised clustering algorithm which involves creating clusters that have predominant ordering from top to bottom.

For example: All files and folders on our hard disk are organized in a hierarchy.

The algorithm groups similar objects into groups called clusters. The endpoint is a set of clusters or groups, where each cluster is distinct from each other cluster, and the objects within each cluster are broadly similar to each other.

This clustering technique is divided into two types:
1. Agglomerative Hierarchical Clustering
2. Divisive Hierarchical Clustering

Agglomerative Hierarchical Clustering:
The Agglomerative Hierarchical Clustering is the most common type of hierarchical clustering used to group objects in clusters based on their similarity. It's also known as AGNES (Agglomerative Nesting). It's a "bottom-up" approach: each observation starts in its own cluster, and pairs of clusters are merged as one moves up the hierarchy.

How does it work?
1. Make each data point a single-point cluster → forms N clusters
2. Take the two closest data points and make them one cluster → forms N-1 clusters
3. Take the two closest clusters and make them one cluster → Forms N-2 clusters.
4. Repeat step-3 until you are left with only one cluster.

Divisive Hierarchical Clustering:
In Divisive or DIANA (DIvisive ANAlysis Clustering) is a top-down clustering method where we assign all of the observations to a single cluster and then partition the cluster to two least similar clusters. Finally, we proceed recursively on each cluster until there is one cluster for each observation. So this clustering approach is exactly opposite to Agglomerative clustering.

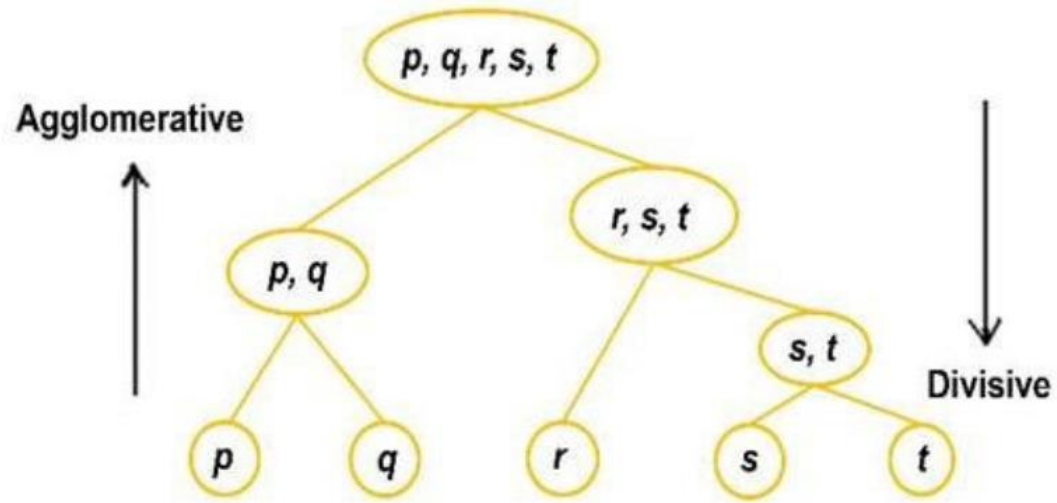


Figure 4.2-4.2. Agglomerative and Divisive

4.2.16.2 Partitional Clustering

A partitional clustering is simply a division of the set of data objects into non overlapping subsets (clusters), such that each data object is in exactly one subset. Partitional clustering decomposes a data set into a set of disjoint clusters. Given a data set of N points, a partitioning method constructs K ($N \geq K$) partitions of the data, with each partition representing a cluster. That is, it classifies the data into K groups by satisfying the following requirements: (1) each group contains at least one point, and (2) each point belongs to exactly one group. Notice that for fuzzy partitioning, a point can belong to more than one group.

Many partitional clustering algorithms try to minimize an objective function. For example, in K-means and K-medoids the function (also referred to as the distortion function) is:

$$\sum_{i=1}^{K}\sum_{j=1}^{|C_i|}\text{Dist}(x_j, \text{center}(i)),$$

4.2.17 ANOVA

The ANOVA test is the initial step in analysing factors that affect a given data set. Once the test is finished, an analyst performs additional testing on the methodical factors that measurably contribute to the data set's inconsistency. The analyst utilizes the ANOVA test results in an f-test to generate additional data that aligns with the proposed regression models. The ANOVA test allows a comparison of more than two groups at the same time to determine whether a relationship exists between them.

Example:

A BOGOF (buy-one-get-one-free) campaign is executed on 5 groups of 100 customers each. Each group is different in terms of its demographic attributes. We would like to determine whether these five respond differently to the campaign. This would help us optimize the right campaign for the right demographic group, increase the response rate, and reduce the cost of the campaign.

The analysis of variance works by comparing the variance between the groups to that within the group. The core of this technique lies in assessing whether all the groups are in fact part of one larger population or a completely different population with different characteristics.

The Formula for ANOVA is:

$$F = \frac{MST}{MSE}$$

where:

F = ANOVA coefficient

MST = Mean sum of squares due to treatment

MSE = Mean sum of squares due to error

There are two types of ANOVA: **one-way (or unidirectional) and two-way.** One-way or two-way refers to the number of independent variables in your analysis of variance test. A one-way ANOVA evaluates the impact of a sole factor on a sole response variable. It determines whether all the samples are the same. The one-way ANOVA is used to determine whether there are any statistically significant differences between the means of three or more independent (unrelated) groups.

A two-way ANOVA is an extension of the one-way ANOVA. With a one-way, you have one independent variable affecting a dependent variable. With a two-way ANOVA, there are two independents. For example, a two-way ANOVA allows a company to compare worker productivity based on two independent variables, such as salary and skill set. It is utilized to observe the interaction between the two factors and tests the effect of two factors at the same time.

4.2.14.2 Decision Trees

A decision tree (also called prediction tree) uses a tree structure to specify sequences of decisions and consequences. Given input X = {x1,x2,...xn}, the goal is to predict a response or output variable Y. Each member of the set {x1,x2,...xn} is called an input variable. The prediction can be achieved by constructing a decision tree with test points and branches. At each test point, a decision is made to pick a specific branch and traverse down the tree. Eventually, a final point is reached, and a prediction can be made. Due to its flexibility and easy visualization, decision trees are commonly deployed in data mining applications for classification purposes.

The input values of a decision tree can be categorical or continuous. A decision tree employs a structure of test points (called nodes) and branches, which represent the decision being made. A node without further branches is called a leaf node. The leaf nodes return class labels and, in some implementations, they return the probability scores. A decision tree can be converted into a set of decision rules. In the following example rule, income and mortgage_amount are input variables, and the response is the output variable default with a probability score.

IF income <50,000 AND mortgage_amount > 100K
THEN default = True WITH PROBABILITY 75%

Decision trees have two varieties: classification trees and regression trees. Classification trees usually apply to output variables that are categorical—often binary—in nature, such as yes or no, purchase or not purchase, and so on. Regression trees, on the other hand, can apply to output variables that are numeric or continuous, such as the predicted price of a consumer good or the likelihood a subscription will be purchased.

Example:

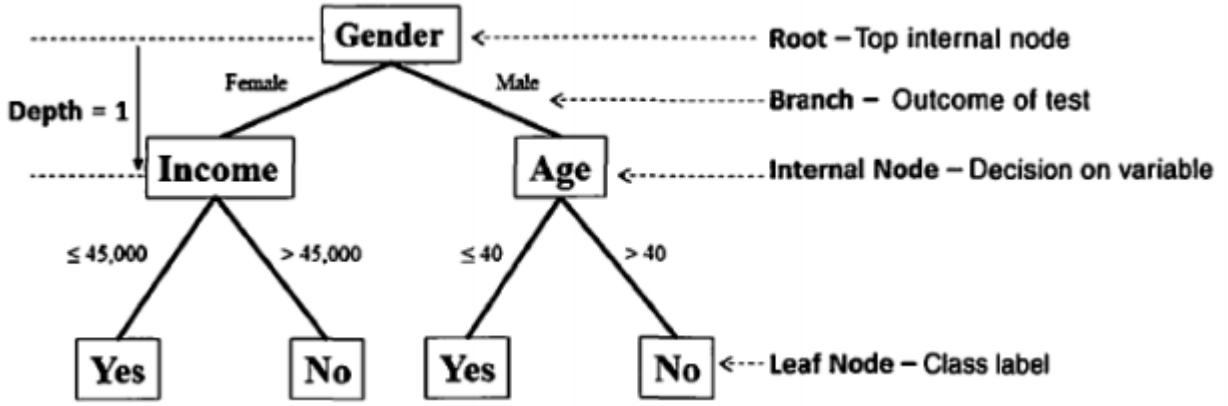


Figure 4.2-9 Decision Tree

The above figure shows an example of using a decision tree to predict whether customers will buy a product. The term branch refers to the outcome of a decision and is visualized as a line connecting two nodes. If a decision is numerical, the "greater than" branch is usually placed on the right, and the "less than" branch is placed on the left. Depending on the nature of the variable, one of the branches may need to include an "equal to" component.

Internal nodes are the decision or test points. Each internal node refers to an input variable or an attribute. The top internal node is called the root. The decision tree in the above figure is a binary tree in that each internal node has no more than two branches. The branching of a node is referred to as a split.

The depth of a node is the minimum number of steps required to reach the node from the root. In above figure for example, nodes Income and Age have a depth of one, and the four nodes on the bottom of the tree have a depth of two. Leaf nodes are at the end of the last branches on the tree. They represent class labels—the outcome of all the prior decisions. The path from the root to a leaf node contains a series of decisions made at various internal nodes.

The decision tree in the above figure shows that females with income less than or equal to $45,000 and males 40 years old or younger are classified as people who would purchase the product. In traversing this tree, age does not matter for females, and income does not matter for males.

Where decision tree is used?

• Decision trees are widely used in practice.
• To classify animals, questions (like cold-blooded or warm-blooded, mammal or not mammal) are answered to arrive at a certain classification.
• A checklist of symptoms during a doctor's evaluation of a patient.
• The artificial intelligence engine of a video game commonly uses decision trees to control the autonomous actions of a character in response to various scenarios.

• Retailers can use decision trees to segment customers or predict response rates to marketing and promotions.
• Financial institutions can use decision trees to help decide if a loan application should be approved or denied. In the case of loan approval, computers can use the logical if - then statements to predict whether the customer will default on the loan.

Model Questions:

1. Explain the transform superstep.
2. Explain the Sun model for TPOLE.
3. Explain Person-to-Time Sun Model.
4. Explain Person-to-Object Sun Model.
5. Why does data have missing values? Why do missing values need treatment? What methods treat missing values?
6. What is feature engineering? What are the common feature extraction techniques?
7. What is Binning? Explain with example.
8. Explain averaging and Latent Dirichlet Allocation with respect to the transform step of data science.
9. Explain hypothesis testing, t-test and chi-square test with respect to data science.
10. Explain over fitting and underfitting. Discuss the common fitting issues.
11. Explain precision recall, precision recall curve, sensitivity, specificity and F1 measure.
12. Explain Univariate Analysis.
13. Explain Bivariate Analysis.
14. What is Linear Regression? Give some common application of linear regression in the real world.
15. What is Simple Linear Regression? Explain.
16. Write a note on RANSAC Linear Regression.
17. Write a note on Logistic Regression.
18. Write a note on Simple Logistic Regression.
19. Write a note on Multinomial Logistic Regression.
20. Write a note on Ordinal Logistic Regression.
21. Explain CLustering techniques.
22. Explain Receiver Operating Characteristic (ROC) Analysis Curves and cross validation test.
23. Write a note on ANOVA.
24. Write a note on Decision Trees.

Unit V
Chapter 1
Transform Super step:

9.1 Objectives:

The objective of this chapter is to learn the data transformation where it brings data to knowledge and converts results into insights.

9.2 Introduction:

The Transform superstep allows to take data from the data vault and formulate answers to questions. The transformation step is the data science process that converts results into meaningful insights.

9.3 Overview:

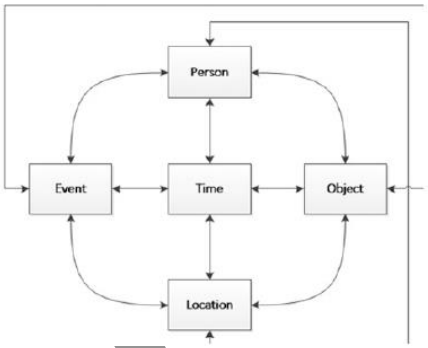
As to explain the below scenario is shown.

Data is categorised in to 5 different dimensions:

1. Time
2. Person
3. Object
4. Location
5. Event

9.4 Dimension Consolidation

The data vault consists of five categories of data, with linked relationships and additional characteristics in satellite hubs.



9.5 The SUN Model

The use of sun models is a technique that enables the data scientist to perform consistent dimension consolidation, by explaining the intended data relationship with the business, without exposing it to the technical details required to complete the transformation processing.

The sun model is constructed to show all the characteristics from the two data vault hub categories you are planning to extract. It explains how you will create two dimensions and a fact via the Transform step.

9.6 Transforming with data science

9.6.1 Missing value treatment

You must describe in detail what the missing value treatments are for the data lake transformation. Make sure you take your business community with you along the journey. At the end of the process, they must trust your techniques and results. If they trust the process, they will implement the business decisions that you, as a data scientist, aspire to achieve.

Why Missing value treatment is required?

Explain with notes on the data traceability matrix why there is missing data in the data lake. Remember: Every inconsistency in the data lake is conceivably the missing insight your customer is seeking from you as a data scientist. So, find them and explain them. Your customer will exploit them for business value.

Why Data Has Missing Values ?

The 5 Whys is the technique that helps you to get to the root cause of your analysis. The use of cause-and-effect fishbone diagrams will assist you to resolve those questions. I have found the following common reasons for missing data:

- Data fields renamed during upgrades
- Migration processes from old systems to new systems where mappings were incomplete
- Incorrect tables supplied in loading specifications by subject-matter expert
- Data simply not recorded, as it was not available
- Legal reasons, owing to data protection legislation, such as the General Data Protection Regulation (GDPR), resulting in a not-to-process tag on the data entry
- Someone else's "bad" data science. People and projects make mistakes, and you will have to fix their errors in your own data science.

9.6.2 Techniques of outlier detection and Treatment

9.7 Hypothesis testing

Hypothesis testing is not precisely an algorithm, but it's a must-know for any data scientist. You cannot progress until you have thoroughly mastered this technique. Hypothesis testing is the process by which statistical tests are used to check if a hypothesis is true, by using data. Based on hypothetical testing, data scientists choose to accept or reject the hypothesis. When an event occurs, it can be a trend or happen by chance. To check whether the event is an important occurrence or just happenstance, hypothesis testing is necessary.

There are many tests for hypothesis testing, but the following two are most popular. T-test and Chi-Square test.

9.8 Chi-square test

There are two types of chi-square tests. Both use the chi-square statistic and distribution for different purposes:

A chi-square goodness of fit test determines if a sample data matches a population. For more details on this type, see: Goodness of Fit Test.

A chi-square test for independence compares two variables in a contingency table to see if they are related. In a more general sense, it tests to see whether distributions of categorical variables differ from each another. A very small chi square test statistic means that your observed data fits your expected data extremely well. In other words, there is a relationship. A very large chi square test statistic means that the data does not fit very well. In other words, there isn't a relationship.

9.9 Univariate Analysis.

Univariate analysis is the simplest form of analysing data. “Uni” means “one”, so in other words your data has only one variable. It doesn’t deal with causes or relationships (unlike regression) and its major purpose is to describe; It takes data, summarizes that data and finds patterns in the data.

Univariate analysis is used to identify those individual metabolites which, either singly or multiplexed, are capable of differentiating between biological groups, such as separating tumour bearing mice from nontumor bearing (control) mice. Statistical procedures used for this analysis include a t-test, ANOVA, Mann–Whitney U test, Wilcoxon signed-rank test, and logistic regression. These tests are used to individually or globally screen the measured metabolites for an association with a disease.

9.10 Bivariate Analysis

Bivariate analysis is when two variables are analysed together for any possible association or empirical relationship, such as, for example, the correlation between gender and graduation with a data science degree? Canonical correlation in the experimental context is to take two sets of variables and see what is common between the two sets. Graphs that are appropriate for bivariate analysis depend on the type of variable. For two continuous variables, a scatterplot is a common graph. When one variable is categorical and the other continuous, a box plot is common, and when both are categorical, a mosaic plot is common.

9.11 Multivariate Analysis

A single metabolite biomarker is generally insufficient to differentiate between groups. For this reason, a multivariate analysis, which identifies sets of metabolites (e.g., patterns or clusters) in the data, can result in a higher likelihood of group separation. Statistical methods for this analysis include unsupervised methods, such as principle component analysis (PCA) or cluster analysis, and supervised methods, such as latent Dirichlet allocation (LDA), partial least squares (PLS), PLS Discriminant Analysis (PLS-DA), artificial neural network (ANN), and machine learning methods. These methods provide an overview of a large dataset that is useful for identifying patterns and clusters in the data and expressing the data to visually highlight similarities and differences. Unsupervised methods may reduce potential bias since the classes are unlabelled.

Regardless of one's choice of method for statistical analysis, it is necessary to subsequently validate the identified potential biomarkers and therapeutic targets by examining them in new and separate sample sets (for biomarkers), and in vitro and/or in vivo experiments evaluating the identified pathways or molecules (for therapeutic targets)

9.12 Linear Regression

9.13 Logistic Regression

Logistic regression is one another technique, used for converting binary classification (dichotomous) problem to linear regression problems. Logistic regression is a predictive analysis technique. This could be difficult to interpret so there are tools available for it. It is used to describe data and to explain the relationship between one dependent binary variable and one or more nominal, ordinal, interval or ratio-level independent variables.

It can answer complex but dichotomous questions such as:

- Probability of getting attending college (YES or NO), provided syllabus is over but faculty is interesting but boring at times depends upon the mood and behaviour of students in class.
- Probability of finishing the lunch sent my mother (YES or NO), depends upon multiple aspects, a) mood b) better options available c) food taste d) scolding by mom e) friends open treat and so further.

Hence, Logistic regression predicts the probability of an outcome that can only have two values (YES or NO)

9.14 Clustering Techniques

Clustering is an unsupervised learning model, similar to classification, it helps creating different set of classes together. It groups the similar types together by creating/identifying the clusters of the similar types. Clustering is a task of dividing homogeneous data types or population or groups. It does so by identifying the similar data types or nearby data elements over graph. In classification the classes are defined with the help algorithms or predefined classes are used and then the data inputs is considered, while in clustering the algorithm inputs itself decides with the help of inputs, the number of clusters depending upon it similarity traits. These similar set of inputs forms a group and called as clusters. Clustering is more dynamic model in terms of grouping.

Basic comparison for clustering and classification is given as below:

PARAMETERS	CLASSIFICATION	CLUSTERING

Fundamental	This model function classifies the data into one of given pre-defined definite classes.	This function maps the data into one of the multiple clusters where the arrangement of data items is relies on the similarities between them.
Involved in	Supervised learning	Unsupervised learning
Training sample	Provided	Not provided

We can classify clustering into two categories :

Soft clustering and hard clustering. Let me give one example to explain the same. For an instance we are developing a website for writing blogs. Now your blog belongs to a particular category such as : Science, Technology, Arts, Fiction etc. It might be possible that the article which is written could belong or relate 2 or more categories. Now, in this case if we restrict our blogger to choose one of the category then we would call this as “hard or strict clustering method”, where a user can remain in any one of the category. Let say this work is done automated by our piece of code and it chooses categories on the basis of the blog content. If my algorithm chooses any one of the given cluster for the blog then it would be called as “hard or strict clustering”. In contradiction to this if my algorithm chooses to select more than one cluster for the blog content then it would be called as “ soft or loose clustering” method.

Clustering methods should consider following important requirements:

- Robustness
- Flexibility
- Efficiency

Clustering algorithms/Methods:

There are several clustering algorithms/Methods available, of which we will be explaining a few:

- Connectivity Clustering Method: This model is based on the connectivity between the data points. These models are based on the notion that the data points closer in data space exhibit more similarity to each other than the data points lying farther away.
- Clustering Partition Method : it works on divisions method, where the division or partition between data set is created. These partitions are predefined non-empty sets. This is suitable for a small dataset.

- Centroid Cluster Method: This model revolve around the centre element of the dataset. The closest data points to the centre data point (centroid) in the dataset is considered to form a cluster. K-Means clustering algorithm is the best fit example of such model.
- Hierarchical clustering Method: This method describes the tree based structure (nested clusters) of the clusters. In this method we have clusters based on the divisions and their sub divisions in a hierarchy (nested clustering). The hierarchy can be pre-determined based upon user choice. Here number of clusters could remain dynamic and not needed to be predetermined as well.
- Density-based Clustering Method: In this method the density of the closest dataset is considered to form a cluster. The more number of closer data sets (denser the data inputs), the better the cluster formation. The problem here comes with outliers, which is handled in classifications (support vector machine) algorithm.

9.15 ANOVA

9.16 Principal Component Analysis (PCA)

9.17 Decision Trees

A decision tree represents classification. Decision tree learning is the most promising technique for supervised classification learning. Since it is a decision tree it is mend to take decision and being a learning decision tree it trains itself and learns from the experience of set of input iterations. These input iterations are also well known as “input training sets” or “training set data”.

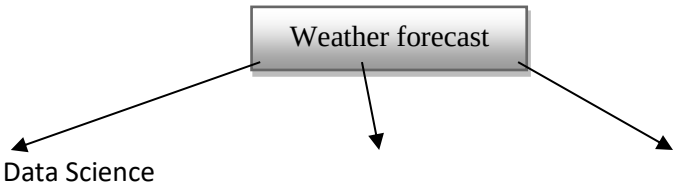
Decision trees predict the future based on the previous learning and input rule sets. It taken multiple input values and returns back the probable output with the single value which is considered as a decision. The input/output could be continuous as well as discrete. A decision tree takes its decision based on the defined algorithms and the rule sets.

For example you want to take a decision to buy a pair of shoes. We start with few set of questions:

1. Do we need one?
2. What would be the budget?
3. Formal or informal?
4. Is it for a special occasion?
5. Which colour suits me better?
6. Which would be the most durable brand?
7. Shall we wait for some special sale or just buy one since its needed?

And similar more questions would give us a choice for selection. This prediction works on the classification where the choice of outputs are classified and the possibility of occurrence is decided on the basis of the probability of the occurrence of that particular output.

Example:



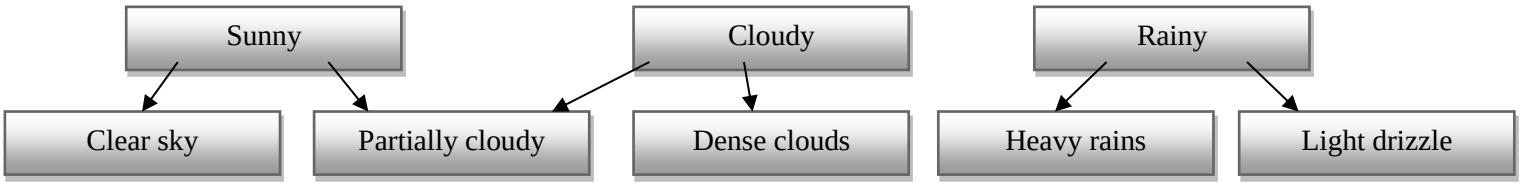


Fig: Example showing the decision tree of weather forecast.

The above figure shows how a decision needs to be taken in a weather forecast scenario where the day is specified as Sunny, Cloudy or Rainy. Depending upon the metrics received by an algorithm it will take the decision. The metrics could be humidity, sky visibility and others. We can also see the cloudy situation having two possibilities of having partially cloudy and dense clouds, wherein having partial clouds is also a subset of a Sunny day. Such occurrences make decision tree bivalence.

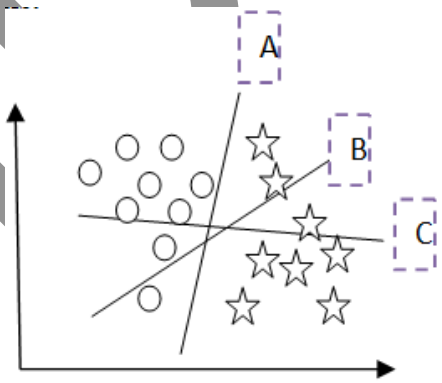
9.18 Support Vector Machines

Support vector machine is an algorithm which is used for classification in a supervised learning algorithm example. It does classification of the inputs received on the basis of the rule-set. It also works on Regression problems.

Classification is needed to differentiate 2 or more sets of similar data.

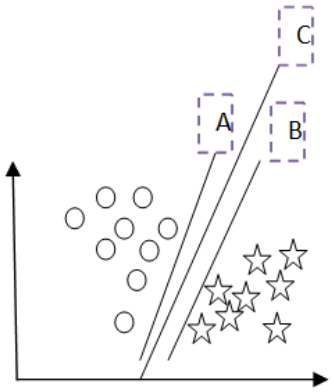
Let us understand how it works.

Scene one:



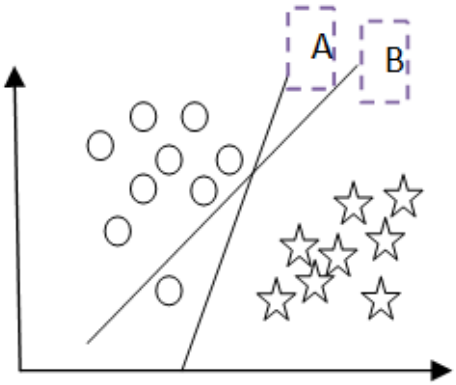
The above scene shows A, B and C as 3 line segments creating hyper planes by dividing the plane. The graph here shows the 2 inputs circles and stars. The inputs could be from two classes. Looking at the scenario we can say that A is the line segment which is diving the 2 hyper planes showing 2 different input classes.

Scene two:



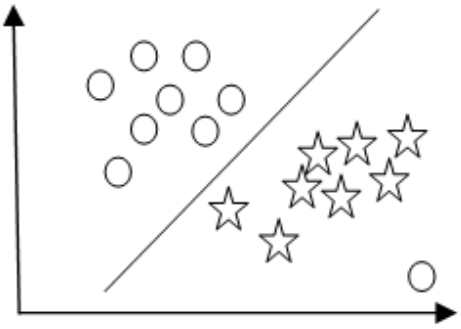
In the scene 2 we can see another rule, the one which cuts the better halves is considered. Hence , hyper plane C is the best choice of the algorithm.

Scene three:



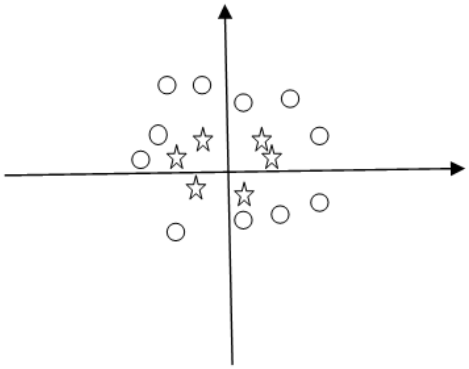
Here in scene 3, we see one circle overlapping hyper plane A , hence according to rule 1 of scene 1 we will choose B which is cutting the co-ordinates into 2 better halves.

Scene four:

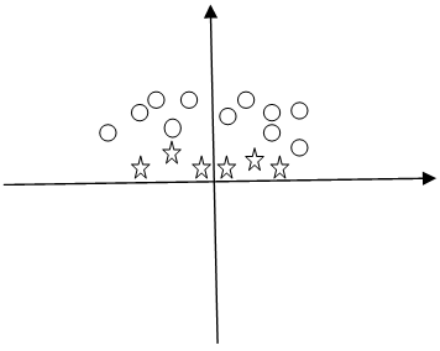


Scene 4 shows one hyper plane dividing the 2 better halves but there exist one extra circle co-ordinate in the other half hyperplane. We call this as an outlier which is generally discarded by the algorithm.

Scene five:



Scene 5 shows another strange scenario where we have the co-ordinates at all 4 quadrants. In this scenario we will fold the x-axis and cut y axis into two halves and transfer the stars and circle on one side of the quadrant and simplify the solution. The representation is shown below:



This gives us again a chance to divide the 2 classes into 2 better halves with using a hyperplane. In the above scenario we have scooped out the stars from the circle co-ordinates and shown it as a different hyperplane.

9.19 Networks, Clusters, and Grids
9.20 Data Mining
9.21 Pattern Recognition
9.22 Machine Learning
9.23 Bagging Data
9.24 Random Forests
9.25 Computer Vision (CV)
9.26 Natural Language Processing (NLP)
9.27 Neural Networks

Artificial Neural Networks; the term is quite fascinating when any student starts learning it. Let us break down the term and know its meaning.

Artificial = means "man made",

Neural = comes from the term Neurons in the brain; a complex structure of nerve cells with keeps the brain functioning. Neurons are the vital part of human brain which does simple input/output to complex problem solving in the brain.

Network = A connection of two entities (here in our case "Neurons", not just two but millions of them).

What is a Neural Network ?

Neural network is a network of nerve cells in the brain. There are about 100 million neurons in our brain. Let us know few more facts about human brain.

Well we all have one 1200 gms of brain (appx). Try weighing it. ?.. That's a different thing that few have kidney beans Inside their skull.?? For those who think they have a tiny little brain. Let me share certain things with you.

Our brain weighs appx 1200-1500 gms.

- It has a complex structure of neurons (nerve cells) with a lot of grey matter. Which is very essential to keep you working fine.
- There are around 100 billion neurons in our brain, which keeps our brain and body functioning by constantly transferring data 24×7 till we are alive.
- The data transfer is achieved by the exchange of electrical or chemical signals with the help of synapses (junction between two neurons).
- They exchange about 1000 trillion synaptic signals per second, which is equivalent to 1 trillion bit per second computer processor.
- The amount of energy generated with these synaptic signal exchange is enough to light a bulb of 5 volts.
- A human brain hence can also store upto 1000 terabytes of data.
- The information transfer happen with the help of the synaptic exchange.
- It takes 7 years to replace every single neuron in the brain. So we tend to forget the content after 7 years, since during the synaptic exchange there is a loss of energy, means loss of information. If we do not recall anything for 7 years then that information is completely erased from our memory.
- Similar to these neurons computer scientist build a complex Artificial neural network using the array of logic gates. The most preferred gates used are XOR gates.

- The best part about Artificial brain is that it can store and cannot forget like human brain and we can store much more information than an individual brain. Unfortunately that has a bigger side effect. Trust me “forgetting is better”. Certain things in our lives we should forget so we can move forward. Imagine if you would have to live with every memory in life. Both negative and positive. Things will haunt you and you will start the journey of psychotic disorders.. ??

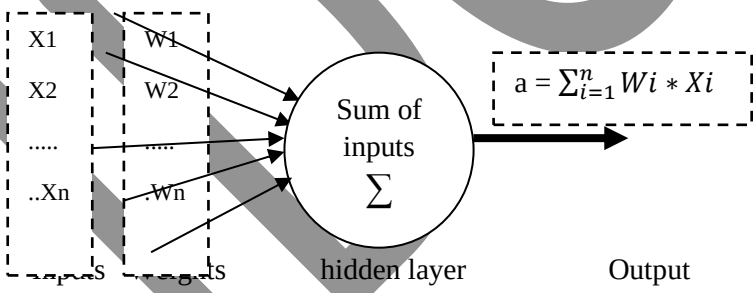
Scary ? Isn’t it !!

Well there are many assumptions with their probable outcome. We always need to look at the positive side of it. Artificial neural networks(ANN) are very useful for solving complex problems and decision making. ANN is an artificial representation of a human brain that tries to simulate its various functions such as learning, calculating , understanding, decision making and many more. Unfortunately it could not reach the exact human brain like function. It is a connection of logical gates which uses mathematical computational model to work and give the output.

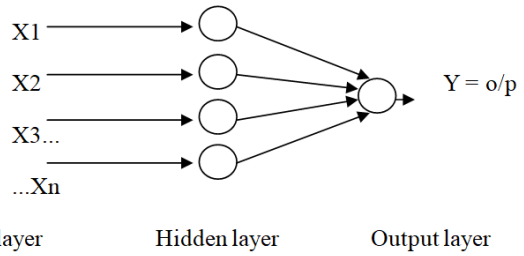
After WWII, in year 15.143 , Warren McCulloch and Walter Pitts modelled artificial neuron to perform computation. They did this by developing a neuron from a logic gate.

Here, the neuron is actually a processing unit, it calculates the weighted sum of the input signal to the neuron to generate the activation signal a, given by :

$$a = \sum_{i=1}^{N} w_i x_i$$



A single Artificial Neuron representation.



Another representation showing detailed multiple neurons working.

Here it shows that, the inputs of all neurons is calculated along with their weights. Hence the weighted sum of all the inputs XiWi (X1W1, X2W2, X3W3.....XnWn), Where X represents input signals and W represents weights is considered as an output to the equation “a”.

These neurons are connected in a long logical network to create a polynomial function(s). So that to calculate multiple complex problems.

In the architecture, more element needs to be added that is a threshold.

Threshold defines the limits to the model. Threshold is defined as THETA (Θ) in neural network model. It is added or subtracted to the output depending upon the model definitions.

This theta defines additional limits acting as a filter to the inputs. With the help of which we can filter out unwanted stuff and get more focus on the needed ones. Another fact about theta is that its value is dynamic according to the environment. For an instance it can be understood as + or - tolerance value in semiconductors / resistors.

9.28 TensorFlow.