DMC 8301

MASTER OF COMPUTER APPLICATION

DATA SCIENCE

CENTRE FOR DISTANCE EDUCATION

ANNA UNIVERSITY

CHENNAI - 600 025

SYLLABUS

OBJECTIVES:

- Know the fundamental concepts of data science and analytics.
- Learn fundamental data analysis using R.
- Understand various data modeling techniques.
- Learn the basic and advanced features of open source big data tools and frameworks.
- Study various analytics on stream data.

UNIT I INTRODUCTION TO DATA SCIENCE AND BIG DATA

Introduction to Data Science – Data Science Process – Exploratory Data analysis – Big data: Definition, Risks of Big Data, Structure of Big Data – Web Data: The Original Big Data – Evolution Of Analytic Scalability – Analytic Processes and Tools – Analysis versus Reporting – Core Analytics versus Advanced Analytics– Modern Data Analytic Tools – Statistical Concepts: Sampling Distributions – Re-Sampling – Statistical Inference – Introduction to Data Visualization.

UNIT II DATA ANALYSIS USING R

Univariate Analysis: Frequency, Mean, Median, Mode, Variance, Standard Deviation, Skewness and Kurtosis – Bivariate Analysis: Correlation – Regression Modeling: Linear and Logistic Regression – Multivariate Analysis – Graphical representation of Univariate, Bivariate and Multivariate Analysis in R: Bar Plot, Histogram, Box Plot, Line Plot, Scatter Plot, Lattice Plot, Regression Line, Two-Way cross Tabulation.

UNIT III DATA MODELING

Bayesian Modeling – Support Vector and Kernel Methods – Principal Component Analysis – Introduction to NoSQL: CAP Theorem, MongoDB: RDBMS VsMongoDB, Mongo DB Database Model, Data Types and Sharding – Data Modeling in HBase: Defining Schema – CRUD Operations

UNIT IV DATA ANALYTICAL FRAMEWORKS

Introduction to Hadoop: Hadoop Overview – RDBMS versus Hadoop – HDFS

(Hadoop Distributed File System): Components and Block Replication – Introduction to MapReduce – Running Algorithms Using MapReduce – Introduction to HBase: HBase Architecture, HLog and HFile, Data Replication – Introduction to Hive, Spark and Apache Sqoop.

UNIT V STREAM ANALYTICS

Introduction To Streams Concepts – Stream Data Model and Architecture – Stream Computing – Sampling Data in a Stream – Filtering Streams – Counting Distinct Elements in a Stream – Estimating Moments – Counting Oneness in a Window – Decaying Window.

OUTCOMES:

On completion of the course, the students will be able to:

- Convert real world problems to hypothesis and perform statistical testing.
- Perform data analysis using R.
- Design efficient modeling of very large data and work with big data plat forms.
- Implement suitable data analysis for stream data.
- Write efficient Map Reduce programs for small problem solving methods.

REFERENCES:

1. Bill Franks, "Taming the Big Data Tidal Wave: Finding Opportunities in Huge Data Streamswith Advanced Analytics", John Wiley & sons, First Edition,2013.
2. Umesh R Hodeg hatta, Umesha Nayak, "Business Analytics Using R – A Practical Approach",Apress, First Edition,2017.
3. J. Leskowec, Anand Rajaraman, Jeffrey David Ullman, "Mining of Massive Datasets", Cambridge University Press, Second Edition,2014.
4. NishantGarg, "HBase Essentials", Packt, First Edition, 2014.

5. Rachel Schutt, Cathy O'Neil, “Doing Data Science”, O'Reilly, First Edition,2013

6. Foster Provost, Tom Fawcet, “Data Science for Business”, O'Reilly, First Edition,2013.

7. Bart Baesens, “Analytics in a Big Data World: The Essential Guide to Data Science and itsApplications”, Wiley, First Edition, 2014.

1.1 Introduction to Data Science

As the world entered the era of big data, the need for its storage also grew. It was the main challenge and concern for the enterprise industries until 2010. The main focus was on building a framework and solutions to store data. Now when Hadoop and other frameworks have successfully solved the problem of storage, the focus has shifted to the processing of this data. Data Science is the secret sauce here. Data Science is the future of Artificial Intelligence. Therefore, it is very important to understand what is Data Science and how can it add value to your business.

1.1.1 What is Data Science?

Data Science is a blend of various tools, algorithms, and machine learning principles with the goal to discover hidden patterns from the raw data. But how is this different from hacking skills, statisticians and substantive expertise on all these years? The answer lies in the difference between explaining and predicting.

Data science is not merely hacking—because when hackers finish debugging their Bash one-liners and Pig scripts, few of them care about non-Euclidean distance metrics.

And data science is not merely statistics, because when statisticians finish theorizing the perfect model, few could read a tab-delimited file into R if their job depended on it.

Data science is the civil engineering of data. Its acolytes possess a practical knowledge of tools and materials, coupled with a theoretical understanding of what's possible. Driscoll explain refers data science by Drew Conway's Venn diagram of from 2010, shown in figure 1.1. Also he mention the "Rise of the Data Scientist", which includes skillfully on Statistics (traditional analysis you're used to thinking about) and Data munging (parsing, scraping, and formatting data).

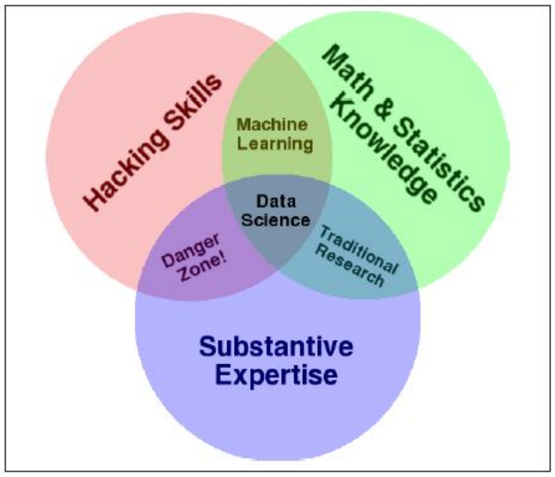


Figure 1.1: Drew Conway's Venn diagram

Data Analyst explains what is going on by processing history of the data. On the other hand, Data Scientist not only does the exploratory analysis to discover insights from it, but also uses various advanced machine learning algorithms to identify the occurrence of a particular event in the future. A Data Scientist will look at the data from many angles, sometimes angles not known earlier.

So, Data Science is primarily used to make decisions and predictions making use of predictive causal analytics, prescriptive analytics (predictive plus decision science) and machine learning.

- Predictive causal analytics: If you want a model that can predict the possibilities of a particular event in the future, you need to apply predictive causal analytics. Say, if you are providing money on credit, then the probability of customers making future credit payments on time is a matter of concern for you. Here, you can build a model that can perform predictive analytics on the payment history of the customer to predict if the future payments will be on time or not.
- Prescriptive analytics: If you want a model that has the intelligence of taking its own decisions and the ability to modify it with dynamic parameters, you certainly need prescriptive analytics for it. This relatively new field is all about providing advice. In other terms, it not only predicts but suggests a range of prescribed actions and associated outcomes, Example: Self-driving car.
- Machine learning for making predictions: If you have transactional data of a finance company and need to build a model to determine the future trend, then machine learning algorithms are the best bet. This falls under the paradigm of supervised learning. It is called supervised because you already have the data based on which you can train your machines. For example, a fraud detection model can be trained using a historical record of fraudulent purchases.
- Machine learning for pattern discovery: If you don't have the parameters based on which you can make predictions, and then you need to find out the hidden patterns within the dataset to be able to make meaningful predictions. This is nothing but the unsupervised model as you don't have any predefined labels for grouping. The most common algorithm used for pattern discovery is Clustering, Example: Consider the working process of a telephone company and you need to establish a network by putting towers in a region. Then, you can use the clustering technique to find those tower locations which will ensure that all the users receive optimum signal strength.

1.1.2 What is a Data Scientist, Really?

Data Scientists work in a variety of fields. Each is crucial to finding solutions to problems and requires specific knowledge. These fields include data acquisition, preparation, mining and modeling, and model maintenance. Data scientists take raw data; turn it into a goldmine of information with the help of machine learning algorithms that answer questions for businesses seeking solutions to their queries.

1.2 Data Science Process

Data Science is all about a systematic process used by Data Scientists to analyze, visualize and model large amounts of data. A data science process helps data scientists use the tools to find unseen patterns, extract data, and convert information to actionable insights that can be meaningful to the company. This aids companies and businesses in making decisions that can help in customer retention and profits. Further, a data science process helps in discovering hidden patterns of structured and unstructured raw data. The process helps in turning a problem into a solution by treating the business problem as a project. The general framework of Data Science Process shown in Figure 1.2, here we explain the workflow of Data Science Process, which involves step-by-step guide to help you get familiar with the process.

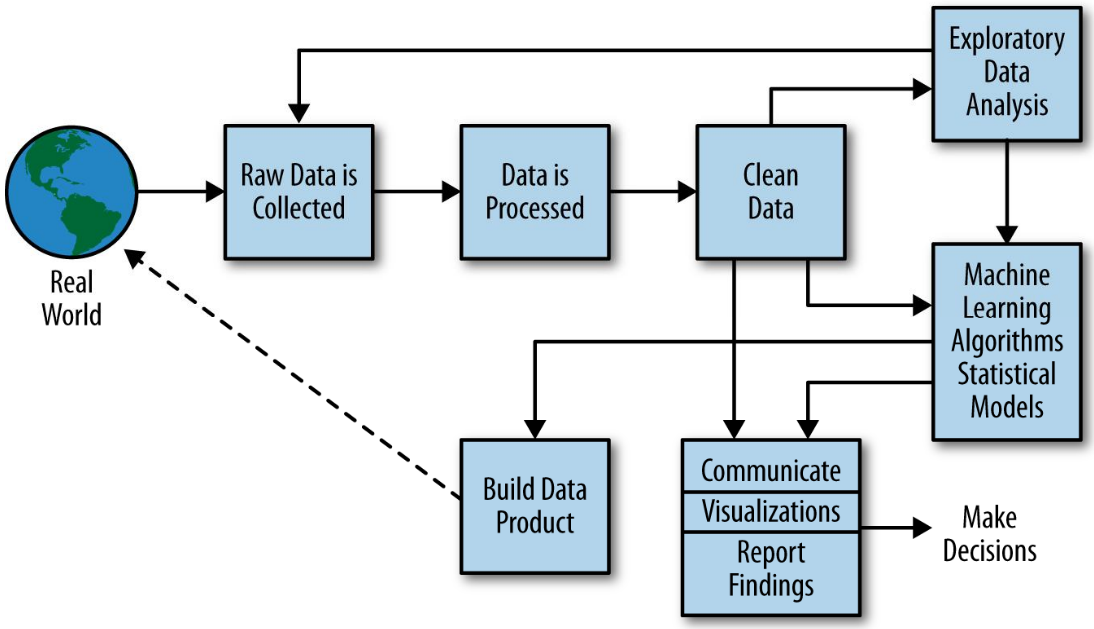


Figure 1.2: The data science process

Step 1: Framing the Problem

Before solving a problem, the pragmatic thing to do is to know what exactly the problem is. Data questions must be first translated to actionable business questions.

A great way to do this is to ask the right questions.

You should then figure out what the sales process looks like, and who the customers are. You need as much context as possible for your numbers to become insights.

- Who the customers are?

- How to identify them?
- Why are they interested in your products?
- What products they are interested in?

Step 2: Collect the raw data needed for your problem

Once you've defined the problem, you'll need data to give you the insights needed to turn the problem around with a solution. This part of the process involves thinking through what data you'll need and finding ways to get that data, whether it's querying internal databases, or purchasing external datasets.

You might find out that your company stores all of their sales data in a CRM or a customer relationship management software platform. You can export the CRM data in a CSV file for further analysis.

Step 3: Processing the Data to Analyze

When you have all the data you need, you will have to process it before going further and analyzing it. Data can be messy if it has not been appropriately maintained, leading to errors that easily corrupt the analysis. These issues can be values set to null when they should be zero or the exact opposite, missing values, duplicate values, and many more. You will have to go through the data and check it for problems to get more accurate insights.

You'll want to check for the following common errors:

- Missing values, perhaps customers without an initial contact date
- Corrupted values, such as invalid entries
- Timezone differences, perhaps your database doesn't take into account the different timezones of your users
- Date range errors, perhaps you'll have dates that makes no sense, such as data registered from before sales started

You will have to also look at the aggregate of all the rows and columns in the file and see if the values you obtain make sense. If it doesn't, you will have to remove or replace the data that doesn't make sense. Once you have completed the data cleaning process, your data will be ready for an exploratory data analysis (EDA)

Step 4: Exploring the Data

In this step, you will have to develop ideas that can help identify hidden patterns and insights. You will have to find more interesting patterns in the data, such as why sales of a particular product or service have gone up or down. You must analyze or notice this kind of data more thoroughly. This is one of the most crucial steps in a data science process.

Step 5: Performing In-depth Analysis

This step is usually the meat of your project, where you apply all the cutting-edge machinery of data analysis to unearth high-value insights and predictions.

Step 6: Communicate results of the analysis

All the analysis and technical results that you come up with are of little value unless you can explain to your stakeholders what they mean, in a way that's comprehensible and compelling. Data storytelling is a critical and underrated skill that you will build and use here.

1.3 Exploratory Data Analysis

Exploratory Data Analysis refers to the critical process of performing initial investigations on data so as to discover patterns, to spot anomalies, to test hypothesis and to check assumptions with the help of summary statistics and graphical representations.

EDA is primarily used to see what data can reveal beyond the formal modeling or hypothesis testing task and provides a better understanding of data set variables and the relationships between them. It can also help determine if the statistical techniques you are considering for data analysis are appropriate.

1.3.1 Why is exploratory data analysis important in data science?

The main purpose of EDA is to help look at data before making any assumptions. It can help identify obvious errors, as well as better understand patterns within the data, detect outliers or anomalous events, find interesting relations among the variables.

Data scientists can use exploratory analysis to ensure the results they produce are valid and applicable to any desired business outcomes and goals. EDA also helps stakeholders by confirming they are asking the right questions. EDA can help answer questions about standard deviations, categorical variables, and confidence intervals. Once EDA is complete and insights are drawn, its features can then be used for more sophisticated data analysis or modeling, including machine learning.

1.3.2 Types of exploratory data analysis

There are four primary types of EDA:

1. Univariate non-graphical: This is simplest form of data analysis, where the data being analyzed consists of just one variable. Since it's a single variable, it doesn't deal with causes or relationships. The main purpose of univariate analysis is to describe the data and find patterns that exist within it.

2. Univariate graphical: Non-graphical methods don't provide a full picture of the data. Graphical methods are therefore required. Common types of univariate graphics include:
   - Stem-and-leaf plots, which show all data values and the shape of the distribution.
   - Histograms, a bar plot in which each bar represents the frequency (count) or proportion (count/total count) of cases for a range of values.
   - Box plots, which graphically depict the five-number summary of minimum, first quartile, median, third quartile, and maximum.
3. Multivariate nongraphical: Multivariate data arises from more than one variable. Multivariate non-graphical EDA techniques generally show the relationship between two or more variables of the data through cross-tabulation or statistics.
4. Multivariate graphical: Multivariate data uses graphics to display relationships between two or more sets of data. The most used graphic is a grouped bar plot or bar chart with each group representing one level of one of the variables and each bar within a group representing the levels of the other variable.

1.4 Big Data

Big data is a term that describes large, hard-to-manage volumes of data – both structured and unstructured – that inundate businesses on a day-to-day basis. But it's not just the type or amount of data that's important; it's what organizations do with the data that matters. Big data can be analyzed for insights that improve decisions and give confidence for making strategic business moves.

Big data is often characterized by the three V's:

The large volume of data in many environments;

The wide variety of data types frequently stored in big data systems; and

The velocity at which much of the data is generated, collected and processed.

1.4.1 Why Is Big Data Important?

The importance of big data doesn't simply revolve around how much data you have. The value lies in how you use it. By taking data from any source and analyzing it, you can find answers that 1) streamline resource management, 2) improve operational efficiencies, 3) optimize product development, 4) drive new revenue and growth opportunities and 5) enable smart decision making. When you combine big data with high-performance analytics, you can accomplish business-related tasks such as:

- Determining root causes of failures, issues and defects in near-real time.
- Spotting anomalies faster and more accurately than the human eye.
- Improving patient outcomes by rapidly converting medical image data into insights.

- Recalculating entire risk portfolios in minutes.
- Sharpening deep learning models' ability to accurately classify and react to changing variables.
- Detecting fraudulent behavior before it affects your organization.

1.5 Risks of Big Data

The risks of Big Data are manifold, and organizations need to carefully plan for their use of Big Data solutions. These risks include strategic and business risks, such as operational impacts and cost overruns, as well as technical risks, such as data quality and security. Some of the key risks are listed below in table.

Risk Factor	**Description**
Loss of Strategic Focus	Organizations that jump into Big Data without specific goals in mind, or do not know what they are looking for from Big Data are at risk of big investment without commensurate results.
Operational Disruption	Big Data provides numerous potential insights from data; in fact, analytics companies are often judged by the novelty and quantity of insights that their algorithms can generate. However these insights can rapidly change or even contradict others and should be continuously tested and used as only one factor in decision making. Companies that chase Big Data insights too aggressively are at risk of making too many high-impact and complex business process changes without being able to accurately predict outcomes.
Poor Data Quality	Big Data often means Lots of Bad Data. This increases the risk of generating outputs and insights from data analysis that are wrong or even dangerous.
Cost Overruns	Implementation on Big Data can be expensive. They often require setup of multiple platforms and systems including cloud infrastructure, applications and complex integrations with existing systems. The combination of complexity and relative immaturity of these systems greatly increases the risk of cost overruns.
Security and Privacy	The very nature of Big Data increases its security risks. Big Data uses distributed systems, which requires multiple levels of protection, automated data transfers on a very frequent or real-time basis need to be constantly authenticated, data origin is often unknown or untracked because of the scale of the data.

1.6 Structure of Big Data

Big data structures can be divided into three categories – structured, unstructured, and semi-structured.

Structured data: It’s the data which follows a pre-defined format and thus, is straightforward to analyze. It conforms to a tabular format together with relationships between different rows and columns. You can think of SQL databases as a common example. Structured data relies on how data could be stored, processed, as well as, accessed. It’s considered the most “traditional” type of data storage.

Unstructured data: This type of big data comes with unknown form and cannot be stored in traditional ways and cannot be analyzed unless it’s transformed into a structured format. You can think of multimedia content like audios, videos, images as examples of unstructured data. It’s important to understand that these days; unstructured data is growing faster than other types of big data.

Semi-structured data: It’s a type of big data that doesn’t conform with a formal structure of data models. But it comes with some kinds of organizational tags or other markers that help to separate semantic elements, as well as, enforce hierarchies of fields and records within that data. You can think of JSON documents or XML files as this type of big data. The reason behind the existence of this category is semi-structured data is significantly easier to analyze than unstructured data. A significant number of big data solutions and tools come with the ability of reading and processing XML files or JASON documents, reducing the complexity of the analyzing process.

1.7 Web Data: The Original Big Data

In the world of Big Data, there's a lot of talk about unstructured data -- after all, "variety" is one of the three Vs. Often these discussions dwell on log file data, sensor output or media content. But what about data on the Web itself -- not data from Web APIs, but data on Web pages that were designed more for eyeballing than machine-driven query and storage?

Understanding web data is crucial in today’s environment because it gives business owners an opportunity to take advantage of the immense volume of data that’s available and gain key insights that would otherwise be impossible.

Now, let us discuss something about web data extraction. It is a process of collecting data from World Wide Web using some web scrapper, crawler, manual mining, etc. A web scrapper or crawler is a cutting tool for harvesting information available on internet. In other word web data extraction is a process of crawling websites and extract data from that page using a tool or programming. Web extraction is related to web indexing which refers to various methods of indexing the contents of web page using a bot or web crawler. A web crawler is an automated program, script or tool using that we can ‘crawl’ webPages to collect multiple information from websites.

In the world of big data, data comes from multiple sources and in huge amount. In which one source is web itself. Web data extraction is one of the medium of collecting data from this source i.e. web. Companies which are leveraging big data technology are using crawlers or programming to collect data. These data comes in bulk i.e. billions of records, or as a data dump. So, it needs to treat as big data and bring into Hadoop Eco system to get quick insight from it.

There are multiples areas where companies can explore web data extraction. Some areas are:

- In ecommerce, companies use web data extraction to monitor their competitor price and improve their product attributes. They also fetch data from different web sources to collect customer review and using Hadoop framework they do analysis – including sentiment analysis.
- Media companies use web scraping to collect recent and popular topics of interest from different social media and popular websites.
- Business directories use web scraping to collect information about the business profile, address, phone, location, zip code, etc.
- In healthcare sector, health physician scrap data from multiple websites to collect information on diseases, medicine, components, etc.

In this whole process, first step is web data extraction, that can be done using different scraping tools available in market (there are free and paid tools are available) or create custom script using programming language with the help of expert in scripting language like Python, ruby, etc.

Second step is to find insight from the data. For this, first we need to process the data using the right tool based on the size of the data and availability of the expert resources. Hadoop framework is the most popular and highly used tool for big data processing. Also, for sentimental analysis of those data, if needed, we need MapReduce which is one of the components of big data.

1.8. Evolution of Analytic Scalability

Big data is no longer just an impressive buzzword. It’s become essential to many companies’ success in today’s business landscape. We will discuss important technologies that make progress in the quest to tame the big data and the convergence of the analytic and data environments.

With the exponential increases in the volume of data being produced and processed, many companies’ databases are being overwhelmed with the deluge of data they are facing. To manage, store and process this overflow of data, a technique called “data scaling” has become necessary for many organizations dealing with exploding datasets. A scalable data platform accommodates rapid changes in the growth of data, either in traffic or volume. These platforms utilize added hardware or software to increase output and storage of data. When a company has a scalable data platform, it also is prepared for the potential of growth in its data needs.

1.8.1 When to Scale?

Scaling can be difficult, but absolutely necessary in the growth of a successful data-driven company. There are a few signs that it’s time to implement a scaling platform. When users begin complaining about slow performance, or service outages, it’s time to scale. Don’t wait for the problem to turn into major source of contention in the minds of your customers. This can have a massively negative impact on retaining those customers. If possible, try to anticipate the problem before it becomes severe. In addition to this, increased application latency, slow read queries rises and database writes are also important indicators that a scale is needed.

1.8.2. A History of Scalability

Until before 1900s, doing analytics was very, very difficult. To perform deep analysis, such as a predictive model, it required manually computing all of the statistics. For example, to perform a linear regression required manually computing a matrix and inverting the matrix by hand.

Slide rules eventually eased the situation, and then calculators came along in the 1970s and helped make it easier to utilize more data. But the volume manageable with a calculator is still trivially small. Computers, which entered the mainstream in the 1980s, are what began to enable people to finally move away from manual computations for good. As the decades have passed, data has moved far beyond the scale that people can handle manually. The amount of data has grown at least as fast as the computing power of the machines that process it.

As new big data sources where available to push the boundaries further. The history of big data is tied tightly to the capability of efficiently storing and managing larger and larger datasets, with size limitations expanding by orders of magnitude. Specifically, for each capability improvement, new database technologies were developed. The history of big data can be roughly split into the following stages

- Megabyte to Gigabyte: In the 1970s and 1980s, historical business data introduced the earliest ‘‘big data’’ challenge in moving from megabyte to gigabyte sizes. The urgent need at that time was to house that data and run relational queries for business analyses and reporting.
- Gigabyte to Terabyte: In the late 1980s, the popularization of digital technology caused data volumes to expand to several gigabytes or even a terabyte, which is beyond the storage and/or processing capabilities of a single large computer system.
- Terabyte to Petabyte: During the late 1990s, when the database community was admiring its ‘‘finished’’ work on the parallel database, the rapid development of Web 1.0 led the whole world into the Internet era, along with massive semi-structured or unstructured web pages holding terabytes or petabytes of data.
- Petabyte to Exabyte: Under current development trends, data stored and analyzed by big companies will undoubtedly reach the PB to exabyte magnitude soon. However, current

technology still handles terabyte to PB data; there has been no revolutionary technology developed to cope with larger datasets.

1.8.3 The Convergence of the Analytic and Data Environments

Both analytics and big data are on unstoppable trajectories right now, moving toward destinations that are beyond the greatest dreams of the first database (RDBMS) builders of the 1960s and 1970s.

The familiar business output from traditional data warehouses systems is a report containing a quantitative description of what was captured through historical transactions. This is extracted from the data via descriptive analytics, which provides hindsight (from past data) and oversight (from current data).

Modern big data systems contain a much broader picture of your business than does a traditional RDBMS. Big data systems combine data from multiple sources (inside and outside your organization), multiple channels (including social media, Web logs, and customer service interactions), and multiple viewpoints (context, content, sentiment, location, and time). As a consequence, it becomes possible to build predictive models of the behavior of objects (customers, machines etc.) within your business area.

The business output from our big data systems can therefore become much more than a report; the output could actually represent new knowledge (about the past, present, and future of your enterprise). This is discovered from the data via predictive analytics, which provides foresight about what is likely to occur regarding the objects within your business area.

Later, prescriptive analytics emerged, which provides insight into how objects behave, going beyond predictive models. Insight is needed in order to see beyond what has happened to understand objectively under what conditions a given object (customer, machine, etc.) will act or react in a certain way, perhaps even in a new way that was not seen in the historical data.

1.9. Analytic Processes and Tools

Data analytics is the science of analyzing raw datasets in order to derive a conclusion regarding the information they hold. It enables us to discover patterns in the raw data and draw valuable information from them.

Data analytics processes and techniques may use applications incorporating machine learning algorithms, simulation, and automated systems. The systems and algorithms work on the unstructured data for human use.

These findings are interpreted and used to help organizations understand their clients better, analyze their promotional campaigns, customize content, create content strategies, and develop products. Data analytics help organizations to maximize market efficiency and improve their earnings.

1.9.1 Process of Data Analytics

Below are the common steps involved in the data analytics method:

- Step 1: Determine the criteria for grouping the data
    - Data can be divided by a range of different criteria such as age, population, income, or sex. The values of the data can be numerical or categorical data.
- Step 2: Collecting the data
    - Data can be collected through several sources, including online sources, computers, personnel, and sources from the community.
- Step 3: Organizing the data
    - The data must be organized after it is collected so that it can be examined. Data organization can take place on a spreadsheet or other type of software that is capable of taking statistical data.
- Step 4: Cleaning the data
    - The data is first cleaned up to ensure that there is no overlap or mistake. Then, it is reviewed to make sure that it is not incomplete. Cleaning the data helps to fix or eliminate any mistakes before the data goes to a data analyst for analysis.

1.9.2. Data Analytics Types

The following are the four fundamental types of data analytics:

- **Descriptive Analytics** describes the happenings over time, such as whether the number of views increased or decreased and whether the current month's sales are better than the last one.
- **Diagnostic Analytics** focuses on the reason for the occurrence of any event. It requires hypothesizing and involves a much diverse dataset. It examines data to answer questions, such as "Did the weather impact the selling of beer?" or "Did the new ad strategy affect sales?"
- **Predictive Analytics** focuses on the events that are expected to occur in the immediate future. Predictive analytics tries to find answers to questions like, what happened to the sales in the last hot summer season? How many weather forecasts expect this year's hot summer?
- **Prescriptive Analytics** indicates a plan of action. If the chance of a hot summer calculated as the average of the five weather models is above 58%, an evening shift can be added to the brewery, and an additional tank can be rented to maximize the production.

1.9.3. Big Data Analytics tools

Big Data Analytics tools are very important for enterprises and large-scale industries because of the huge volume of data that will be generated and managed by modern organizational tools using Big Data tools. Big Data Analytics tools help businesses in saving time and money and also in gaining insights to make data-driven decisions.

Big Data analytics is the complete process of collecting, gathering, organizing, and analyzing huge sets of data (known as Big Data) to observe/identify the patterns and also other useful information needed for business decisions. The process – Big Data analytics helps organizations to better understand the information which is present within the sets of data. The guy who works as an Analyst working with Big Data typically will have the knowledge that comes from analyzing the data. There are different types of tools are available under Data Analytics that help to improve the data analyzing process that are data analysis, data cleansing, data mining, data visualization, data integration, data storage, and management.

For the process of big data analytics, there is a need for very High-Performance Analytics. Hence to analyze such a huge volume of data, specialized software tools are required for the Big Data analytics process and applications for predictive analytics, data mining, text mining, forecasting, and data optimization.

1.9.4. How Big Data Analytics works and its key technologies

As mentioned earlier, a big data analytics process is not a single activity that encompasses a huge volume of data. Instead, its advanced analytics can be applied to large data, but in reality, several types of different technologies work together to achieve the most value from information. Below are the biggest and important technologies involve in the big data analytics process:

- Data management
- Data mining
- Hadoop
- In-memory analytics
- Predictive analytics.
- Text mining

There 'N' number of Big Data Analytics tools, below is the list of some of the top tools used to store and analyze Big Data. These Big Data Analytics tools can be further be classified into two Storage and Querying/Analysis.

- Apache Hadoop: Apache Hadoop, a big data analytics tool that is a java based free software framework. It helps in the effective storage of a huge amount of data in a storage place known as a cluster. The special feature of this framework is it runs in parallel on a cluster and also has the ability to process huge data across all nodes in it.

There is a storage system in Hadoop popularly known as the Hadoop Distributed File System (HDFS), which helps to splits the large volume of data and distribute it across many nodes present in a cluster. It also performs the replication process of data in a cluster hence providing high availability and recovery from the failure – which increases the fault tolerance.

- KNIME: KNIME Analytics Platform is one of the leading open solutions for data-driven innovation. This tool helps in discovering the potential & hidden in a huge volume of data; it also performs mine for fresh insights or predicts the new futures. The KNIME Analytics Platform tool is a very much helpful toolbox for data scientists.
- OpenRefine: OpenRefine is introduced as Google Refine. This tool is one of the efficient tools to work on the messy and large volume of data that all include: cleansing data, transforming that data from one format to another, and also to perform extending it with web services and external data. The open refine tool helps explore large data sets easily.
- Orange: Orange is famous open-source data visualization and helps in data analysis for beginner and as well to the expert. This tool provides interactive workflows with a large toolbox option to create the same, which helps in the analysis and visualizing of data. An orange tool has many and different visualizations that include bar charts, trees, scatter plots, dendrograms, networks, and heat maps.
- RapidMiner : RapidMiner tool operates using visual programming, and also it is much capable of manipulating, analyzing, and modeling the data. RapidMiner tools make data science teams easier and productive by using an open-source platform for all their jobs like machine learning, data prep, and model deployment. Because of its uniformity in the data science platform makes accelerates in the building of complete analytical workflows in a single environment which helps in dramatically improving efficiency and short duration of time to value for data science projects.

1.9.5. Benefits of Data Analytics

- Decision making improves
  - Companies may use the information they obtain from data analytics to guide their decisions, leading to improved results. Data analytics removes a lot of guesswork from preparing marketing plans, deciding what material to make, creating goods, and more. With advanced data analytics technologies, new data can be constantly gathered and analyzed to enhance your understanding of changing circumstances.
- Marketing becomes more effective
  - When businesses understand their customers better, they will be able to sell to them more efficiently. Data analytics also gives businesses invaluable insights into how their marketing campaigns work so that they can fine-tune them for better results.
- Customer service improves

    - Data analytics provides businesses with deeper insight into their clients, helping them to customize customer experience to their needs, offer more customization, and create better relationships with them.
- The efficiency of operations increases
    - Data analytics will help businesses streamline their operations, save resources, and improve the bottom line. When businesses obtain a better idea of what the audience needs, they spend less time producing advertisements that do not meet the desires of the audience.

1.10. Analysis versus Reporting

Too many organizations mistakenly equate reporting with analysis. Let's understand what it is? Reports are important and can be valuable. Reports used correctly will add value. But reports have their limits, and it is important to understand what they are. In the end, an organization will need both reporting and analysis to succeed in taming big data, just as both reporting and analysis have been utilized to tame every other data source that's come along in the past.

Reporting

Reporting is the process of organizing data into informational summaries in order to monitor how different areas of a business are performing. A reporting environment is also called as business intelligence environment. Such an environment is where users go to select the reports they want to run, get the reports executed, and view the results. The reports may contain tables, graphs, and charts in any combination. The key factors that define a report include:

- A report will provide back to the user the data that was asked for.
- That data will be provided in a standardized, predefined format.
- There is no person involved in generating a report outside of the user who requested the report through his or her reporting interface.
- As a result, reports are fairly inflexible.

Note: An analysis can lead to reports, and reports can lead to an analysis. It is even possible to have an analysis based entirely off of reports. For example, you might run 10 reports, line them up on the desk, identify the key things you see in each, and write a summary of what you found and what it means. That's an analysis. It is the thought that a person puts into the business implications of data or statistics that makes an analysis. Data and statistics without any interpretation are useless.

Analysis

The process of exploring data and reports in order to extract meaningful insights, which can be used to better understand and improve business performance. The key points that define an analysis are:

- An analysis provides answers to the questions being asked.
- An analysis process takes any steps needed to get the answers to those questions.
- An analysis is therefore customized to the specific questions being addressed.
- An analysis involves a person who guides the process.
- By its very nature, the analysis process is flexible.

1.10.1 CONTRAST BETWEEN ANALYSIS AND REPORTING

The basis differences between Analysis and Reporting are as follows:

- Reporting translates raw data into information.
- Analysis transforms data and information into insights.
- reporting shows you what is happening
- While analysis focuses on explaining why it is happening and what you can do about it.
- Reports are like Robots n monitor and alter you and where as analysis is like parents - can figure out what is going on (hungry, dirty diaper, no pacifier, , teething, tired, ear infection, etc).
- Reporting and analysis can go hand-in-hand:
- Reporting provides no limited context about what is happening in the data. Context is critical to good analysis.
- Reporting translate a raw data into information
- Reporting usually raises a question – What is happening ?
- Analysis transforms the data into insights - Why is it happening ? What you can do about it?

1.11 CORE ANALYTICS VERSUS ADVANCED ANALYTICS

Core analytics tend to ask simple questions and provide simple answers. A core analytics process also called as business intelligence is going to investigate what happened, when it happened, and what the impact was. Advanced analytics goes further than core analytics. Advanced analytics includes everything from complex ad hoc SQL, to forecasting, to data mining, to predictive modeling.

The terms business intelligence and advanced analytics are often deployed as reference terms for business procedures designed to derive insight from operational data.

We can safely say that business intelligence and advanced analytics are both data-oriented management techniques that businesses of any size — from local food carts to global beverage manufacturers — can leverage to improve their operations.

The best way to understand distinctions between the two terms, though, is to think about the different questions they answer.

Business intelligence is generally used to explain why something happened in the past.

Advanced analytics is generally used to explain why something is happening in the present and what that will mean if trends continue.

We've established that business intelligence provides an insightful summary of past and current data. But advanced analytics goes a step further by using sophisticated modeling techniques to predict future events or discover patterns that can't be detected otherwise.

Advanced analytics tools can process this data and then perform numerous functions, including correlational analysis, regression analysis, forecasting analysis, text mining, image analytics, pattern matching, cluster analysis, multivariate statistics, and more.

This process can be used to answer specific business questions including:

- Why is this happening?
- What if these trends continue?
- What will happen next?
- What is the best that can happen?

Note: While business intelligence is focused on reporting and querying, advanced analytics is about optimizing, correlating, and predicting the next best action or the next most likely action.

1.12 Modern Data Analytic Tools

Modern Analytic Tools: Current Analytic tools concentrate on three classes:

- Batch processing tools
- Stream Processing tools and
- Interactive Analysis tools

Batch processing system:

- Batch Processing System involves collecting a series of processing jobs and carrying them out periodically as a group (or batch) of jobs.
- It allows a large volume of jobs to be processed at the same time.
- An organization can schedule batch processing for a time when there is little activity on their computer systems, for example overnight or at weekends.
- One of the most famous and powerful batch process-based Big Data tools is Apache Hadoop. It provides infrastructures and platforms for other specific Big Data applications.

Stream processing tools

- Stream processing envisioning (predicting) the life in data as and when it transpires.

- The key strength of stream processing is that it can provide insights faster, often within milliseconds to seconds. It helps understanding the hidden patterns in millions of data records in real time. It translates into processing of data from single or multiple sources in real or near-real time applying the desired business logic and emitting the processed information to the sink.
- Stream processing serves multiple and resolves in today's business arena.

Interactive Analysis -Big Data Tools

- The interactive analysis presents the data in an interactive environment, allowing users to undertake their own analysis of information.
- Users are directly connected to the computer and hence can interact with it in real time.
- The data can be reviewed, compared and analyzed in tabular or graphic format or both at the same time.

1.13 Statistical Concepts

Statistics, as an academic and professional discipline, is the collection, analysis and interpretation of data. Professionals who work with statistics also have to be able to communicate their findings. As such, statistics is a fundamental tool of data scientists, who are expected to gather and analyze large amounts of structured and unstructured data and report on their findings.

Data is raw information, and data scientists learn how to mine it, according to Data Science Central. Data scientists use a combination of statistical formulas and computer algorithms to notice patterns and trends within data. Then, they use their knowledge of social sciences and a particular industry or sector to interpret the meaning of those patterns and how they apply to real-world situations. The purpose is to generate value for a business or organization.

To become a data scientist, you must have a strong understanding of mathematics, statistical reasoning, computer science and information science. You must understand statistical concepts, how to use key statistical formulas, and how to interpret and communicate statistical results.

Basic Terminologies in Statistics:

Following are the basic terminologies in statistics to understand the statistical program.

- **Population:** A population is the set of resources from where we can collect data
- **Sample:** A Sample is nothing but a subset of the Population which is used for sampling of data and in inferential statistics to predict the outcome.
- **Variable:** A Variable can be a number, a characteristic, or a quantity that can be counted. It can be also called a data point.

- **Probability Distribution:** A probability distribution is a mathematical concept that primarily gives the probabilities of occurrence of different possible outcomes generally for an experiment conducted by statisticians.
- **Statistical Parameter:** Statistical or population parameter is basically a quantity that helps in indexing a family of probability distributions like the mean, median, or mode of a population.

1.13.1 Types of Statistics

There are two kinds of Statistics, which are descriptive Statistics and inferential Statistics. In descriptive Statistics, the Data or Collection Data are described in a summarized way, whereas in inferential Statistics, we make use of it in order to explain the descriptive kind. Both of them are used on a large scale. Also, there is another kind of Statistics where descriptive transitions into inferential Statistics.

Statistics is mainly divided into the following two categories.

- Descriptive Statistics
- Inferential Statistics

Descriptive Statistics

In the descriptive Statistics, the Data is described in a summarized way. The summarization is done from the sample of the population using different parameters like Mean or standard deviation. Descriptive Statistics are a way of using charts, graphs, and summary measures to organize, represent, and explain a set of Data.

- Data is typically arranged and displayed in tables or graphs summarizing details such as histograms, pie charts, bars or scatter plots.
- Descriptive Statistics are just descriptive and thus do not require normalization beyond the Data collected.

Inferential Statistics

In the Inferential Statistics, we try to interpret the Meaning of descriptive Statistics. After the Data has been collected, analyzed, and summarized we use Inferential Statistics to describe the Meaning of the collected Data.

- Inferential Statistics use the probability principle to assess whether trends contained in the research sample can be generalized to the larger population from which the sample originally comes.
- Inferential Statistics are intended to test hypotheses and investigate relationships between variables and can be used to make population predictions.

- Inferential Statistics are used to draw conclusions and inferences, i.e., to make valid generalizations from samples.

Example:

In a class, the Data is the set of marks obtained by 50 students. Now when we take out the Data average, the result is the average of 50 students' marks. If the average marks obtained by 50 students are 88 out of 100, on the basis of the outcome, we will draw a conclusion.

Mean, Median and Mode in Statistics

- **Mean**: Mean is considered the arithmetic average of a Data set that is found by adding the numbers in a set and dividing by the number of observations in the Data set.
- **Median**: The middle number in the Data set while listed in either ascending or descending order is the Median.
- **Mode**: The number that occurs the most in a Data set and ranges between the highest and lowest value is the Mode.

For n number of observations, we have

$$\text{Mean} = \overline{x} = \frac{\sum x}{n} = \sum xn$$

$$\text{Median} = \frac{\left[\frac{n}{2}+1\right]^{th} term}{2} \text{ if n is odd.}$$

$$\text{Median} = \frac{\left[\frac{n}{2}\right]^{th} term + \left[\frac{n}{2}+1\right]^{th} term}{2} \text{ if n is even.}$$

Mode = The value which occurs most frequently

Measures of Dispersion in Statistics

The measures of central tendency do not suffice to describe the complete information about a given Data. Therefore, the variability is described by a value called the measure of dispersion.

The different measures of dispersion include:

- The range in Statistics is calculated as the difference between the maximum value and the minimum value of the Data points.
- The quartile deviation that measures the absolute measure of dispersion. The Data points are divided into 3 quarters. Find the Median of the Data points. The Median of the Data points to the left of this Median is said to be the upper quartile and the Median of the

Data points to the right of this Median is said to be the lower quartile. Upper quartile - lower quartile is the interquartile range. Half of this is the quartile deviation.

- The Mean deviation is the statistical measure to determine the average of the absolute difference between the items in a distribution and the Mean or Median of that series.
- The standard deviation is the measure of the amount of variation of a set of values.

Stages of Statistics

- Collection of Data:
  - This is the first step of statistical Analysis where we collect the Data using different methods depending upon the case.
- Organizing the Collected Data:
  - In the next step, we organize the collected Data in a Meaningful manner. All the Data is made easier to understand.
- Presentation of Data:
  - In the third step we simplify the Data. These Data are presented in the form of tables, graphs, and diagrams.
- Analysis of the Data:
  - Analysis is required to get the right results. It is often carried out using measures of central tendencies, measures of dispersion, correlation, regression, and interpolation.
- Interpretation of Data:
  - In this last stage, conclusions are enacted. Use of comparisons is made. On this basis, forecasting is made.

Uses of Statistics

- Statistics helps to obtain appropriate quantitative Data.
- Statistics helps to present complex Data for the simple and consistent Interpretation of the Data in a suitable tabular, diagrammatic, and graphic form.
- Statistics help to explain the nature and pattern of variability through quantitative observations of a phenomenon.
- Statistics help to depict the Data in tabular form or in a graphical form in order to understand it properly.

1.14 Sampling Distributions

Sampling distribution is a statistic that determines the probability of an event based on data from a small group within a large population. Its primary purpose is to establish representative results of small samples of a comparatively larger population. Since the population is too large to analyze, the smaller group is selected and repeatedly sampled, or analyzed. The gathered data, or statistic, is used to calculate the likely occurrence, or probability, of an event.

Understanding sampling distribution

The idea behind a sampling distribution is that when you have a large amount of data (gathered from a large group, the value of a statistic from random samples of a small group will inform you of that statistic's value for the entire group. Once the data is plotted on a graph, the values of any given statistic in random samples will make a normal distribution from which you can draw inferences.

Each random sample selected will have a different value assigned to the statistic being studied. For example, if you randomly sample data three times and determine the mean, or the average, of each sample, all three means are likely to be different and fall somewhere along the graph. That's variability. You do that many times, and eventually the data you plot should look like a bell curve. That process is a sampling distribution.

Factors that influence sampling distribution

The sampling distribution's variability can be measured either by standard deviation, also called "standard error of the mean," or population variance, depending on the context and inferences you are trying to draw. They both are mathematical formulas that measure the spread of data points in relation to the mean.

There are three primary factors that influence the variability of a sampling distribution. They are:

- The number observed in a population: This variable is represented by "N." It is the measure of observed activity in a given group of data.
- The number observed in the sample: This variable is represented by "n." It is the measure of observed activity in a random sample of data that is part of the larger grouping.
- The method of choosing the sample: How the samples were chosen can account for variability in some cases.

Types of distributions

There are three standard types of sampling distributions in statistics.

- Sampling distribution of mean
    - The most common type of sampling distribution is of the mean. It focuses on calculating the mean of every sample group chosen from the population and plotting the data points. The graph shows a normal distribution where the center is the mean of the sampling distribution, which represents the mean of the entire population.
- Sampling distribution of proportion

    - This sampling distribution focuses on proportions in a population. Samples are selected and their proportions are calculated. The mean of the sample proportions from each group represent the proportion of the entire population
- T-distribution
    - A T-distribution is a sampling distribution that involves a small population or one where not much is known about it. It is used to estimate the mean of the population and other statistics such as confidence intervals, statistical differences and linear regression. The T-distribution uses a t-score to evaluate data that wouldn't be appropriate for a normal distribution.
    - The formula for t-score is: t = [x - μ] / [s / sqrt(n)]
    - In the formula, "x" is the sample mean and "μ" is the population mean and signifies standard deviation.

1.15 Re-Sampling

Once we have a data sample, it can be used to estimate the population parameter. The problem is that we only have a single estimate of the population parameter, with little idea of the variability or uncertainty in the estimate. One way to address this is by estimating the population parameter multiple times from our data sample. This is called Re-sampling.

Statistical Re-sampling methods are procedures that describe how to economically use available data to estimate a population parameter. The result can be both a more accurate estimate of the parameter (such as taking the mean of the estimates) and a quantification of the uncertainty of the estimate (such as adding a confidence interval).

Re-sampling methods are very easy to use, requiring little mathematical knowledge. They are methods that are easy to understand and implement compared to specialized statistical methods that may require deep technical skill in order to select and interpret.

Note: The key idea is to resample form the original data — either directly or via a fitted model — to create replicate datasets, from which the variability of the quantiles of interest can be assessed without long-winded and error-prone analytical calculation. Because this approach involves repeating the original data analysis procedure with many replicate sets of data, these are sometimes called computer-intensive methods.

Given the imbalanced dataset, the problem we often face is where most data falls into major class, whereas a few data falls into the minority class. To overcome the poor performance of model training for such imbalanced data, over-sampling and under-sampling techniques are primarily suggested to produce equally distributed data fall into each class.

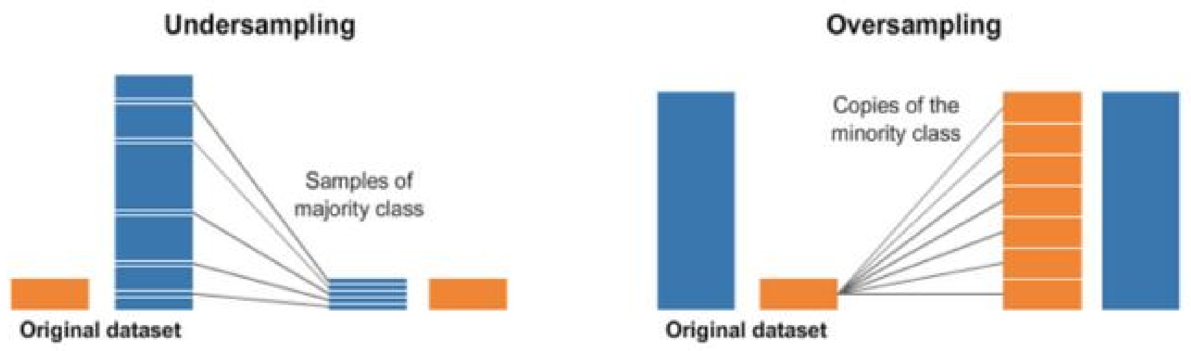


Figure: Re-sampling strategies for imbalanced datasets

Two commonly used Re-sampling methods that you may encounter are k-fold cross-validation and the bootstrap.

- **Bootstrap**. Samples are drawn from the dataset with replacement (allowing the same sample to appear more than once in the sample), where those instances not drawn into the data sample may be used for the test set.
- **k-fold Cross-Validation.** A dataset is partitioned into k groups, where each group is given the opportunity of being used as a held out test set leaving the remaining groups as the training set.

1.16 Statistical Inference

Statistical inference is the process of drawing conclusions about unknown population properties, using a sample drawn from the population. . Unknown population properties can be, for example, mean, proportion or variance. These are also called parameters.

Statistical inference is broadly divided into 2 parts: Estimation and Hypothesis Testing. Estimation is further divided into point estimation and interval estimation.

In point estimation, we estimate an unknown parameter using a single number that is calculated from the sample data. For example, the average salary of junior data scientists based on a sample is 55,000 euros

In Interval estimation, we find a range of values within which we believe the true population parameter lies with high probability. Here, the average salary of junior data scientists is between 52, 0000 and 58,000, with a 95% confidence level.

In hypothesis testing we need to decide whether a statement regarding a population parameter is true or false, based on sample data. For example, a claim that the average salary of junior data scientists is greater than 50, 0000 euros annually can be tested using sample data.

Parameters, Estimators and Estimates

Let's now look at the difference between parameters, estimators and estimates. A parameter is an unknown quantity such as population mean. It is estimated using a function of sample values, such as sample mean. So, a sample mean is an estimator. The value of the sample mean using sample values is called an estimate.

Hypothesis Testing

A hypothesis is an assertion about the distribution of one or more random variables.

A null hypothesis, often referred to as H0, is an assertion which is generally believed to be true until a researcher rejects it with evidence. An alternative hypothesis, H1, is where the researcher's claim contradicts the null hypothesis. In other words, hypothesis testing decides whether a statement regarding a population parameter is true or false, based on a set of sample data.

A test statistic is a random variable that is calculated from sample data and used in a hypothesis test. We can use a test statistic to determine whether to reject a null hypothesis. It compares our data with what is expected under a null hypothesis.

A critical region, also known as the rejection region, is a set of values for which the null hypothesis is rejected.

1.17 Introduction to Data Visualization

Data visualization is a graphic representation that expresses the significance of data. It reveals insights and patterns that are not immediately visible in the raw data. It is an art through which information, numbers, and measurements can be made more understandable.

The main goal of data visualization is to communicate information clearly and effectively through graphical means. It doesn't mean that data visualization needs to look boring to be functional or extremely sophisticated to look beautiful. To convey ideas effectively, both aesthetic form and functionality need to go hand in hand, providing insights into a rather sparse and complex data set by communicating its key-aspects in a more intuitive way.

Information visualization is the art of representing data so that it is easy to understand and manipulate, thus making the information useful. Visualization can make sense of information by helping to find relationships in the data and support (or disproving) ideas about the data.

1.17.1 Why data visualization is such a powerful tool

As the world is changing, the need for information is changing as well. Here are a few benefits of data visualization:

- Easily, graspable information – Data is increasing day-by-day, and it is not wise for anyone to scram through such quantity of data to understand it. Data visualization comes handy then.
- Establish relationships – Charts and graphs do not only show the data but also established co-relations between different data types and information.
- Share – Data visualization is also easy to share with others. You could share any important fact about a market trend using a chart and your team would be more receptive about it.
- Interactive visualization – today, when technological inventions are making waves in every market segment, regardless of big or small, you could also leverage interactive visualization to dig deeper and segment the different portions of charts and graphs to obtain a more detailed analysis of the information being presented.
- Intuitive, personalized, updatable – Data visualization is interactive. You could click on it and get another big picture of a particular information segment. They are also tailored according to the target audience and could be easily updated if the information modifies.

1.17.2 What are different Data Visualization Tools?

Data visualization tool helps in, well, visualizing data. Using these tools, data and information can be generated and read easily and quickly. Many data visualization tools range from simple to complex and from intuitive to obtuse.

- Tableau Desktop – A business intelligence tool which helps you in visualizing and understanding your data.
- Zoho Reports – Zoho Reports is a self-service business intelligence (BI) and analytics tool that enables you to design intuitive data visualizations.
- Microsoft Power BI – Developed by Microsoft, this is a suite of business analytics tools that allows you to transform information into visuals.
- MATLAB – A detailed data analysis tool that has an easy-to-use tool interface and graphical design options for visuals.
- Sisense – A BI platform that allows you to visualize the information to make better and more informed business decisions.

1.17.3 What are Data Visualization Techniques?

Here are a few data visualizations that you must know:

- Know the target audience – this shouldn't come as a surprise. Designing a chart of a graph should always be done based on the audience that will view it.
- Create a goal – or more like a logical narrative. Ensure to set clear goals that must be conveyed through the infographic. Also, the relevant content type is a must.
- Choose the chart type – A pie chart does not complement every information visually. Similarly, a bar graph does not show every statistic clearly. Choose the chart part accurately to put forth the information.
- Context – Use of colours is encouraged depending upon the context. A decrease in the profit growth could be marked red, whereas green could show the increasing parameter.
- Use tools – Yes, one of the easiest ways to create data visuals is using tools. Use them as they make the charts intuitive as well as easy to read.

SUMMARY

- Data Science is a blend of various tools, algorithms, and machine learning principles with the goal to discover hidden patterns from the raw data.
- Exploratory Data Analysis refers to the critical process of performing initial investigations on data so as to discover patterns, to spot anomalies, to test hypothesis and to check assumptions with the help of summary statistics and graphical representations.
- Big data is a term that describes large, hard-to-manage volumes of data – both structured and unstructured.
- A web crawler is an automated program, script or tool using that we can 'crawl' webPages to collect multiple information from websites.
- Data analytics is the science of analyzing raw datasets in order to derive a conclusion regarding the information they hold. It enables us to discover patterns in the raw data and draw valuable information from them.
- Big Data Analytics tools are very important for enterprises and large-scale industries because of the huge volume of data that will be generated and managed by modern organizational tools using Big Data tools. Big Data Analytics tools help businesses in saving time and money and also in gaining insights to make data-driven decisions.
- Reporting is the process of organizing data into informational summaries in order to monitor how different areas of a business are performing.
- A core analytics process also called as business intelligence is going to investigate what happened, when it happened, and what the impact was.
- Advanced analytics includes everything from complex ad hoc SQL, to forecasting, to data mining, to predictive modeling.

SELF-ASSESSMENT QUESTIONS

- Define big data. Why is big data required? How does traditional BI environment differ from big data environment?

- What are the challenges with big data?
- Describe the current analytical architecture for data scientists
- What are the key roles for the New Big Data Ecosystem?
- Write a short note on Classification of Analytics.
- Write a short note on data science and data science process.
- What are different phases of the Data Analytics Lifecycle? Explain each in detail.

FURTHER READINGS

- Bill Franks, "Taming the Big Data Tidal Wave: Finding Opportunities in Huge Data Streams with Advanced Analytics", John Wiley & sons, First Edition,2013.
- Rachel Schutt, Cathy O'Neil, "Doing Data Science", O'Reilly, First Edition,2013
- Foster Provost, Tom Fawcet, "Data Science for Business", O'Reilly, First Edition,2013.
- Bart Baesens, "Analytics in a Big Data World: The Essential Guide to Data Science and its Applications", Wiley, First Edition,2014.

2.1 Univariate Analysis

Univariate analysis is a basic kind of analysis technique for statistical data. Here the data contains just one variable and does not have to deal with the relationship of a cause and effect. Like for example consider a survey of a classroom. The analysts would want to count the number of boys and girls in the room. The data here simply talks about the number which is a single variable and the variable quantity. The main objective of the univariate analysis is to describe the data in order to find out the patterns in the data. This is done by looking at the mean, mode, median, standard deviation, dispersion, etc.

Univariate analysis is basically the simplest form to analyze data. Uni means one and this means that the data has only one kind of variable. The major reason for univariate analysis is to use the data to describe. The analysis will take data, summarise it, and then find some pattern in the data.

A variable is simply a condition or subset of your data in univariate analysis. It can be thought of as a “category.” For example, the analysis could look at a variable such as “age,” or it can look at “height,” or “weight.” However, it does not examine more than one variable at a time, nor would it look at their relationship. The analysis of two variables and their relationship is termed bivariate analysis. If three or more variables are considered simultaneously, it is multivariate analysis.

2.1.1 Frequency

Frequency means how often something takes place. The observation frequency tells the number of times for the occurrence of an event. The frequency distribution table may show categorical or qualitative and numeric or quantitative variables. The distribution gives a snapshot of the data to find out the patterns.

2.1.2 Mean

Mean is the average of the given numbers and is calculated by dividing the sum of given numbers by the total number of numbers.

Mean = (Sum of all the observations/Total number of observations)

Mean for Ungrouped Data

The example given below will help you in understanding how to find the mean of ungrouped data.

Example:

In a class there are 20 students and they have secured a percentage of 88, 82, 88, 85, 84, 80, 81, 82, 83, 85, 84, 74, 75, 76, 89, 90, 89, 80, 82, and 83.

Find the mean percentage obtained by the class.

Solution:

Mean = Total of percentage obtained by 20 students in class/Total number of students

= [88 + 82 + 88 + 85 + 84 + 80 + 81 + 82 + 83 + 85 + 84 + 74 + 75 + 76 + 89 + 90 + 89 + 80 + 82 + 83]/20

= 1660/20

= 83

Hence, the mean percentage of each student in the class is 83%.

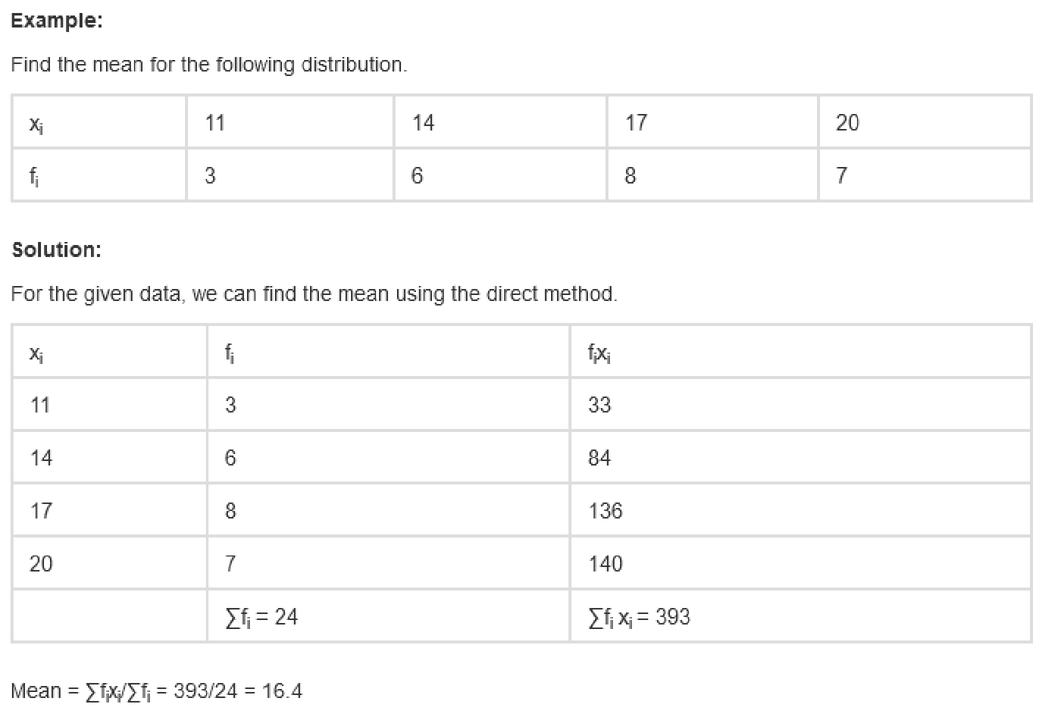
Mean for Grouped Data

For grouped data, we can find the mean using either of the following formulas.

Direct method:

Mean

$$\bar{x} = \frac{\sum_{i=1}^{n} f_i x_i}{\sum_{i=1}^{n} f_i}$$



Example:

Find the mean for the following distribution.

x_i	11	14	17	20
f_i	3	6	8	7

Solution:

For the given data, we can find the mean using the direct method.

x_i	f_i	f_ix_i
11	3	33
14	6	84
17	8	136
20	7	140
	$\sum f_i = 24$	$\sum f_i x_i = 393$

Mean = $\sum f_ix_i/\sum f_i$ = 393/24 = 16.4

2.1.3 Median

Median is a statistical measure that determines the middle value of a dataset listed in ascending order. The measure divides the lower half from the higher half of the dataset.

Example

- Consider the data: 4, 4, 6, 3, and 2. let’s arrange this data in ascending order: 2, 3, 4, 4, 6.
- Count the number of values. There are 5 values.
- Look for the middle value. The middle value is the median. Thus, median = 4.

2.1.4 Mode

A mode is defined as the value that has a higher frequency in a given set of values. It is the value that appears the most number of times.

How to Find the Mode

No calculations are necessary to find the mode. Simply follow the steps below:

- Collect and organize the data from a dataset.
- Determine all the distinct values in a dataset.
- Count the frequency of occurrence for each distinct value.
- The most frequent value(s) is the mode.

2.1.5 Variance

Variance is a statistical measurement that is used to determine the spread of numbers in a data set with respect to the average value or the mean.

There can be two types of variances in statistics, namely, sample variance and population variance. The symbol of variance is given by σ2. Variance is widely used in hypothesis testing, checking the goodness of fit, and Monte Carlo sampling.

Variance is a measure of dispersion. A measure of dispersion is a quantity that is used to check the variability of data about an average value. Data can be of two types - grouped and ungrouped. When data is expressed in the form of class intervals it is known as grouped data. On the other hand, if data consists of individual data points, it is called ungrouped data. The sample and population variance can be determined for both kinds of data.

Population Variance - All the members of a group are known as the population. When we want to find how each data point in a given population varies or is spread out then we use the population variance. It is used to give the squared distance of each data point from the population mean.

Sample Variance - If the size of the population is too large then it is difficult to take each data point into consideration. In such a case, a select number of data points are picked up from the population to form the sample that can describe the entire group. Thus, the sample variance can be defined as the average of the squared distances from the mean. The variance is always calculated with respect to the sample mean.

Formulas for Variance

Grouped Data Sample Variance = $\sum_{i=1}^{n} \frac{f(M_i - \bar{x})^2}{N-1}$

Grouped Data Population Variance = $\sum_{i=1}^{n} \frac{f(M_i - \bar{x})^2}{N}$

Ungrouped Data Sample Variance = $\sum_{i=1}^{n} \frac{(X_i - \bar{x})^2}{n-1}$

Ungrouped Data Population Variance =$\sum_{i=1}^{n} \frac{(X_i - \bar{x})^2}{n}$

where, $\bar{x}$ stands for mean, M_i is the midpoint of the ith interval, X_i is the ith data point, N is the summation of all frequencies and n is the number of observations.

The general formula for variance is given as,

Var (X) = $E[(X - \mu)^2]$

2.1.6 Standard Deviation

Standard deviation is the positive square root of the variance. Standard deviation is one of the basic methods of statistical analysis. Standard deviation is commonly abbreviated as SD and denoted by 'σ' and it tells about the value that how much it has deviated from the mean value. If we get a low standard deviation then it means that the values tend to be close to the mean whereas a high standard deviation tells us that the values are far from the mean value.

Standard deviation is the degree of dispersion or the scatter of the data points relative to its mean, in descriptive statistics. It tells how the values are spread across the data sample and it is the measure of the variation of the data points from the mean. The standard deviation of a sample, statistical population, random variable, data set, or probability distribution is the square root of its variance.

Formula for Calculating Standard Deviation

The population standard deviation formula is given as:

σ = $\sqrt{\frac{1}{N}\sum_{i=1}^{N}(X_i-\mu)^2}$

Here,

σ = Population standard deviation

μ = Assumed mean

Example: There are 39 plants in the garden. A few plants were selected randomly and their heights in cm were recorded as follows: 51, 38, 79, 46, 57. Calculate the standard deviation of their heights.

Solution:

N = 5

Mean($\bar{x}$) = (51+38+79+46+57)/5 = 54.2

Standard Deviation = $\sqrt{\frac{1}{N}\sum_{i=1}^{N}(X_i-\mu)^2}$

= √(51−54.2)2+(38−54.2)2+(79−54.2)2+(46−54.2)2+(57−54.2)2/4

= 15.5

Standard Deviation = 15.5

2.1.7 Skewness and Kurtosis

“Skewness essentially measures the symmetry of the distribution, while kurtosis determines the heaviness of the distribution tails.”

The understanding shape of data is a crucial action. It helps to understand where the most information is lying and analyze the outliers in a given data. Here we study about the shape of data, the importance of skewness and kurtosis.

Skewness

If the values of a specific independent variable (feature) are skewed, depending on the model, skewness may violate model assumptions or may reduce the interpretation of feature importance.

In statistics, skewness is a degree of asymmetry observed in a probability distribution that deviates from the symmetrical normal distribution (bell curve) in a given set of data.

The normal distribution helps to know a skewness. When we talk about normal distribution, data symmetrically distributed. The symmetrical distribution has zero skewness as all measures of a central tendency lies in the middle.

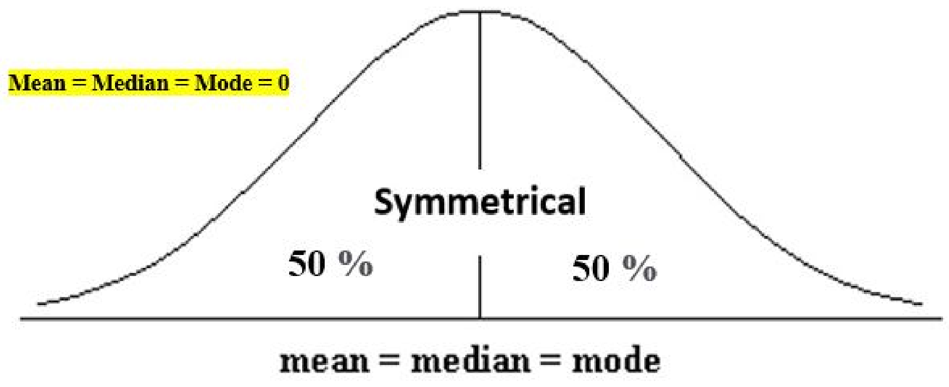


Figure 2.1: Zero skewness in symmetrical distribution

When data is symmetrically distributed, the left-hand side and right-hand side, contain the same number of observations. (If the dataset has 100 values, then the left-hand side has 50 observations, and the right-hand side has 50 observations.). But, what if not symmetrical distributed? That data is called asymmetrical data, and that time skewness comes into the picture.

The skewness can be on two types:

- **Positively Skewed:** In a distribution that is Positively Skewed, the values are more concentrated towards the right side, and the left tail is spread out. Hence, the statistical results are bent towards the left-hand side. Hence, that the mean, median, and mode are always positive. In this distribution, Mean > Median > Mode.
- **Negatively Skewed:** In a Negatively Skewed distribution, the data points are more concentrated towards the right-hand side of the distribution. This makes the mean, median, and mode bend towards the right. Hence these values are always negative. In this distribution, Mode > Median > Mean.

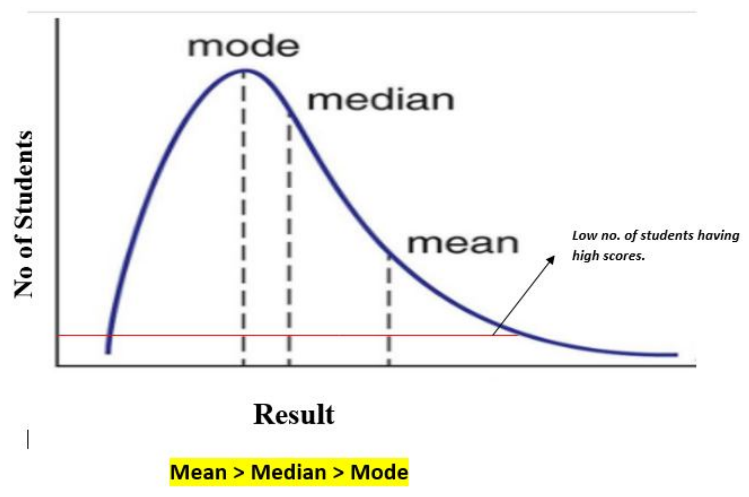


Figure 2.2: Positive skewed or right-skewed

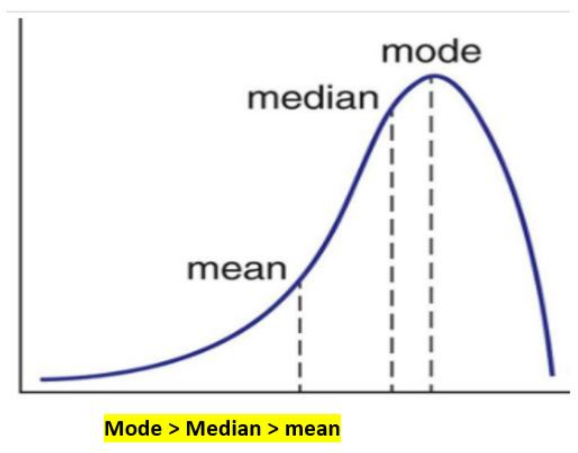


Figure 2.3: Negative skewed or left-skewed

Calculate the skewness coefficient of the sample

Pearson’s first coefficient of skewness

The median is always the middle value, and the mean and mode are the extremes, so you can derive a formula to capture the horizontal distance between mean and mode.

Pearson’s first coefficient = $\frac{Mean - Mode}{Standard\ Deviation}$

The above formula gives you Pearson's first coefficient. Division by the standard deviation will help you scale down the difference between mode and mean. This will scale down their values in a range of -1 to 1.

Pearson's first coefficient of skewness is helping if the data present high mode. But, if the data have low mode or various modes, Pearson's first coefficient is not preferred, and Pearson's second coefficient may be superior, as it does not rely on the mode.

Pearson's second coefficient of skewness

Pearson's second coefficient = $\frac{3(Mean - Median)}{Standard\ Deviation}$

- If the skewness is between -0.5 & 0.5, the data are nearly symmetrical.
- If the skewness is between -1 & -0.5 (negative skewed) or between 0.5 & 1(positive skewed), the data are slightly skewed.
- If the skewness is lower than -1 (negative skewed) or greater than 1 (positive skewed), the data are extremely skewed.

Kurtosis

Kurtosis is used to find the presence of outliers in our data. It gives us the total degree of outliers present.

Kurtosis is a statistical measure, whether the data is heavy-tailed or light-tailed in a normal distribution. In finance, kurtosis is used as a measure of financial risk. A large kurtosis is associated with a high level of risk for an investment because it indicates that there are high probabilities of extremely large and extremely small returns. On the other hand, a small kurtosis signals a moderate level of risk because the probabilities of extreme returns are relatively low.

Excess Kurtosis

- The data can be heavy-tailed, and the peak can be flatter, almost like punching the distribution or squishing it. This is called Negative Kurtosis (Platykurtic).
- If the distribution is light-tailed and the top curve steeper, like pulling up the distribution, it is called Positive Kurtosis (Leptokurtic)
- The expected value of kurtosis is 3. This is observed in a symmetric distribution.

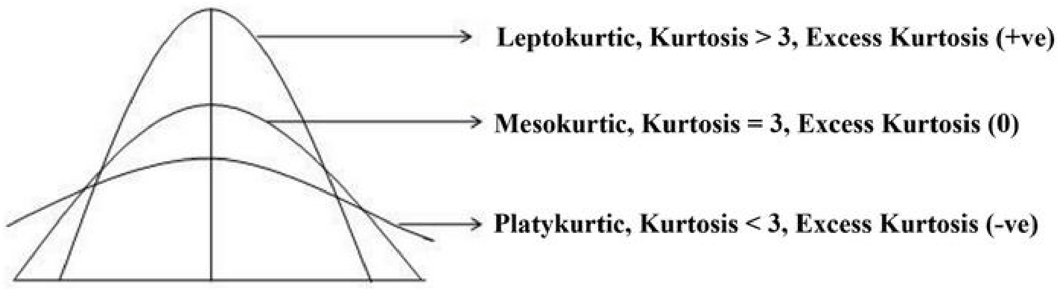

Leptokurtic, Kurtosis > 3, Excess Kurtosis (+ve)
Mesokurtic, Kurtosis = 3, Excess Kurtosis (0)
Platykurtic, Kurtosis < 3, Excess Kurtosis (-ve)

Figure 2.4: Excess Kurtosis

A kurtosis greater than three will indicate Positive Kurtosis. In this case, the value of kurtosis will range from 1 to infinity. Further, a kurtosis less than three will mean a negative kurtosis. The range of values for a negative kurtosis is from -2 to infinity. The greater the value of kurtosis, the higher the peak. Hence, you can say that Skewness and Kurtosis are used to describe the spread and height of your normal distribution. Skewness is used to denote the horizontal pull on the data. It tells you how spread out the data is, and Kurtosis is used to find the vertical pull or the peak's height.

2.2 Bivariate Analysis

Bivariate analysis is one of the statistical analysis where two variables are observed. One variable here is dependent while the other is independent. These variables are usually denoted by X and Y. So, here we analyze the changes occurred between the two variables and to what extent.

Bivariate analysis is stated to be an analysis of any concurrent relation between two variables or attributes. This study explores the relationship of two variables as well as the depth of this relationship to figure out if there are any discrepancies between two variables and any causes of this difference. Some of the examples are percentage table, scatter plot, etc.

For analysis, it is necessary to recognize bivariate data first. Usually, the data comprises two measurements such as X and Y. For each measurement, the bivariate data can be interpreted as the pair (X, Y). These variables are often called bivariate simple random sample (SRS). We can denote these variables as (X1,Y1), (X2,Y2),…..,(Xn,Yn).

The bivariate data can be represented in a table as shown below:

Observations	X-Variable	Y-Variable
1	12	6
2	10	5
3	8	3
4	7	1
5	5	-1

Types of Bivariate Analysis

The types of a bivariate analysis will depend upon the types of variables or attributes we will use for analysing. The variable could be numerical, categorical or ordinal. If the independent variable is categorical, like a particular brand of pen, then logit or probit regression can be used. If independent and dependent both the attributes are ordinal, which means they have position or ranking, then we can measure a rank correlation coefficient. If dependent attribute is ordinal, then ordered logit or ordered probit can be utilised. Also, if the dependent attribute is either ratio or interval, like temperature scale, then we can measure regression. So based on these data, we can mention the types of bivariate data analysis:

- Numerical and Numerical – In this type, both the variables of bivariate data, independent and dependent, are having numerical values.
- Categorical and Categorical – When both the variables are categorical.
- Numerical and Categorical – When one variable is numerical and one is categorical.

2.2.1 Correlation

Correlation is a statistical method used to assess a possible linear association between two continuous variables. It is simple both to calculate and to interpret. There are three possible results of a correlational study: a positive correlation, a negative correlation, and no correlation.

A positive correlation is a relationship between two variables in which both variables move in the same direction. Therefore, when one variable increases as the other variable increases, or one variable decreases while the other decreases. An example of positive correlation would be height and weight. Taller people tend to be heavier.

A negative correlation is a relationship between two variables in which an increase in one variable is associated with a decrease in the other. An example of negative correlation would be height above sea level and temperature. As you climb the mountain (increase in height) it gets colder (decrease in temperature).

A zero correlation exists when there is no relationship between two variables. For example there is no relationship between the amount of tea drunk and level of intelligence.

Types of correlation coefficient formulas

There are several types of correlation coefficient formulas.

One of the most commonly used formulas is Pearson's correlation coefficient formula.

$$r = \frac{n(\sum xy) - (\sum x)(\sum y)}{\sqrt{[n\sum x^2 - (\sum x)^2]\,[\,n\sum y^2 - (\sum y)^2]}}$$

The other two formulas commonly used are sample correlation coefficient and population correlation coefficient.

Sample correlation coefficient

$$r_{xy} = \frac{S_{xy}}{S_x S_y}$$

S_x and S_y are the sample standard deviations, and S_{xy} is the sample covariance.

Population correlation coefficient

$$\rho_{xy} = \frac{\sigma_{xy}}{\sigma_x \sigma_y}$$

The population correlation coefficient uses σ_x and σ_y as the population standard deviations, and σ_{xy} as the population covariance.

Some uses of Correlations

Prediction: If there is a relationship between two variables, we can make predictions about one from another.

Validity: Concurrent validity (correlation between a new measure and an established measure).

Reliability:

- Test-retest reliability (are measures consistent).
- Inter-rater reliability (are observers consistent).

Theory verification: Predictive validity.

2.3 Regression Modeling

A regression model provides a function that describes the relationship between one or more independent variables and a response, dependent, or target variable.

For example, the relationship between height and weight may be described by a linear regression model. A regression analysis is the basis for many types of prediction and for determining the effects on target variables. When you hear about studies on the news that talk about fuel efficiency, or the cause of pollution, or the effects of screen time on learning, there is often a regression model being used to support their claims.

Types of Regression Analysis

- Linear Regression
- Logistic Regression
- Polynomial Regression
- Ridge Regression
- Lasso Regression
- Quantile Regression
- Bayesian Linear Regression
- Principal Components Regression

- Partial Least Squares Regression
- Elastic Net Regression

There are numerous regression analysis approaches available for making predictions. Additionally, the choice of technique is determined by various parameters, including the number of independent variables, the form of the regression line, and the type of dependent variable.

2.3.1 Linear Regression

A linear regression is a model where the relationship between inputs and outputs is a straight line. This is the easiest to conceptualize and even observe in the real world. Even when a relationship isn't very linear, our brains try to see the pattern and attach a rudimentary linear model to that relationship.

One example may be around the number of responses to a marketing campaign. If we send 1,000 emails, we may get five responses. If this relationship can be modeled using a linear regression, we would expect to get ten responses when we send 2,000 emails. Your chart may vary, but the general idea is that we associate a predictor and a target, and we assume a relationship between the two.

Using a linear regression model, we want to estimate the correlation between the number of emails sent and response rates. In other words, if the linear model fits our observations well enough, then we can estimate that the more emails we send, the more responses we will get.

When making a claim like this, whether it is related to exercise, happiness, health, or any number of claims, there is usually a regression model behind the scenes to support the claim.

In addition, the model fit can be described using a mean squared error. This basically gives us a number to show exactly how well the linear model fits.

More serious examples of a linear regression would include predicting a patient's length of stay at a hospital, relationship between income and crime, education and birth rate, or sales and temperature.

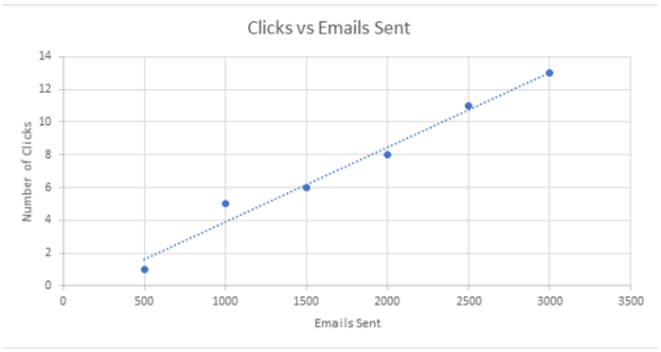


Figure 2.5: Example for linear regression

2.3.2 Logistic Regression

When the dependent variable is discrete, the logistic regression technique is applicable. In other words, this technique is used to compute the probability of mutually exclusive occurrences such as pass/fail, true/false, 0/1, and so forth. Thus, the target variable can take on only one of two values, and a sigmoid curve represents its connection to the independent variable, and probability has a value between 0 and 1.

For example,

- To predict whether an email is spam (1) or (0)
- Whether the tumor is malignant (1) or not (0)

Consider a scenario where we need to classify whether an email is spam or not. If we use linear regression for this problem, there is a need for setting up a threshold based on which classification can be done. Say if the actual class is malignant, predicted continuous value 0.4 and the threshold value is 0.5, the data point will be classified as not malignant which can lead to serious consequence in real time.

From this example, it can be inferred that linear regression is not suitable for classification problem. Linear regression is unbounded, and this brings logistic regression into picture. Their value strictly ranges from 0 to 1.

Types of Logistic Regression

- Binary Logistic Regression: The categorical response has only two 2 possible outcomes. Example: Spam or Not
- Multinomial Logistic Regression: Three or more categories without ordering. Example: Predicting which food is preferred more (Veg, Non-Veg, Vegan)
- Ordinal Logistic Regression: Three or more categories with ordering. Example: Movie rating from 1 to 5

Logistic regression applications in business

Organizations use insights from logistic regression outputs to enhance their business strategy for achieving business goals such as reducing expenses or losses and increasing ROI in marketing campaigns.

An e-commerce company that mails expensive promotional offers to customers, for example, would like to know whether a particular customer is likely to respond to the offers or not: i.e., whether that consumer will be a "responder" or a "non-responder." In marketing, this is called propensity to respond modeling.

Likewise, a credit card company will develop a model to help it predict if a customer is going to default on its credit card based on such characteristics as annual income, monthly credit card payments and the number of defaults. In banking parlance, this is known as default propensity modeling.

Why is logistic regression important?

Logistic regression is important because it transforms complex calculations around probability into a straightforward arithmetic problem. Admittedly, the calculation itself is a bit complex, but modern statistical applications automate much of this grunt work. This dramatically simplifies analyzing the impact of multiple variables and helps to minimize the effect of confounding factors.

Use cases of logistic regression

Logistic regression is commonly used for prediction and classification problems. Some of these use cases include:

- Fraud detection: Logistic regression models can help teams identify data anomalies, which are predictive of fraud. Certain behaviors or characteristics may have a higher association with fraudulent activities, which is particularly helpful to banking and other financial institutions in protecting their clients. SaaS-based companies have also started to adopt these practices to eliminate fake user accounts from their datasets when conducting data analysis around business performance.
- Disease prediction: In medicine, this analytics approach can be used to predict the likelihood of disease or illness for a given population. Healthcare organizations can set up preventative care for individuals that show higher propensity for specific illnesses.
- Churn prediction: Specific behaviors may be indicative of churn in different functions of an organization. For example, human resources and management teams may want to know if there are high performers within the company who are at risk of leaving the organization; this type of insight can prompt conversations to understand problem areas within the company, such as culture or compensation. Alternatively, the sales organization may want to learn which of their clients are at risk of taking their business elsewhere. This can prompt teams to set up a retention strategy to avoid lost revenue.

2.4 Multivariate Analysis

Multivariate Analysis is defined as a process of involving multiple dependent variables resulting in one outcome. This explains that the majority of the problems in the real world are Multivariate. For example, we cannot predict the weather of any year based on the season. There are multiple factors like pollution, humidity, precipitation, etc.

2.4.1 The History of Multivariate analysis

In 1928, Wishart presented his paper. The Precise distribution of the sample covariance matrix of the multivariate normal population, which is the initiation of MVA.

In the 1930s, R.A. Fischer, Hotelling, S.N. Roy, and B.L. Xu et al. made a lot of fundamental theoretical work on multivariate analysis. At that time, it was widely used in the fields of psychology, education, and biology.

In the middle of the 1950s, with the appearance and expansion of computers, multivariate analysis began to play a big role in geological, meteorological. Medical and social and science. From then on, new theories and new methods were proposed and tested constantly by practice and at the same time, more application fields were exploited. With the aids of modern computers, we can apply the methodology of multivariate analysis to do rather complex statistical analyses.

2.4.2 Multivariate data analysis techniques and examples

There are many different techniques for multivariate analysis, and they can be divided into two categories:

- Dependence techniques
- Interdependence techniques

Multivariate analysis techniques: Dependence vs. interdependence

When we use the terms "dependence" and "interdependence," we're referring to different types of relationships within the data. To give a brief explanation:

Dependence methods

Dependence methods are used when one or some of the variables are dependent on others. Dependence looks at cause and effect; in other words, can the values of two or more independent variables be used to explain, describe, or predict the value of another, dependent variable? To give a simple example, the dependent variable of "weight" might be predicted by independent variables such as "height" and "age."

In machine learning, dependence techniques are used to build predictive models. The analyst enters input data into the model, specifying which variables are independent and which ones are dependent—in other words, which variables they want the model to predict, and which variables they want the model to use to make those predictions.

Interdependence methods

Interdependence methods are used to understand the structural makeup and underlying patterns within a dataset. In this case, no variables are dependent on others, so you're not looking for causal relationships. Rather, interdependence methods seek to give meaning to a set of variables or to group them together in meaningful ways.

So: One is about the effect of certain variables on others, while the other is all about the structure of the dataset.

Some useful multivariate analysis techniques are:

- Multiple linear regression
- Multiple logistic regression
- Multivariate analysis of variance (MANOVA)
- Factor analysis
- Cluster analysis

2.4.3 Multiple linear regression

Multiple linear regression is a dependence method which looks at the relationship between one dependent variable and two or more independent variables. A multiple regression model will tell you the extent to which each independent variable has a linear relationship with the dependent variable. This is useful as it helps you to understand which factors are likely to influence a certain outcome, allowing you to estimate future outcomes.

Example: As a data analyst, you could use multiple regression to predict crop growth. In this example, crop growth is your dependent variable and you want to see how different factors affect it. Your independent variables could be rainfall, temperature, amount of sunlight, and amount of fertilizer added to the soil. A multiple regression model would show you the proportion of variance in crop growth that each independent variable accounts for.

2.4.4 Multiple logistic regression

Logistic regression analysis is used to calculate (and predict) the probability of a binary event occurring. A binary outcome is one where there are only two possible outcomes; either the event occurs (1) or it doesn’t (0). So, based on a set of independent variables, logistic regression can predict how likely it is that a certain scenario will arise. It is also used for classification.

Example: Let as consider you work as an analyst within the insurance sector and you need to predict how likely it is that each potential customer will make a claim. You might enter a range of independent variables into your model, such as age, whether or not they have a serious health condition, their occupation, and so on. Using these variables, a logistic regression analysis will calculate the probability of the event (making a claim) occurring.

2.4.5 Multivariate analysis of variance (MANOVA)

Multivariate analysis of variance (MANOVA) is used to measure the effect of multiple independent variables on two or more dependent variables. With MANOVA, it’s important to note that the independent variables are categorical, while the dependent variables are metric in nature. A categorical variable is a variable that belongs to a distinct category.

Example: The variable “employment status” could be categorized into certain units, such as “employed full-time,” “employed part-time,” “unemployed,” and so on. A metric variable is measured quantitatively and takes on a numerical value.

2.4.6 Factor analysis

Factor analysis is an interdependence technique which seeks to reduce the number of variables in a dataset. If you have too many variables, it can be difficult to find patterns in your data. At the same time, models created using datasets with too many variables are susceptible to overfitting. Overfitting is a modeling error that occurs when a model fits too closely and specifically to a certain dataset, making it less generalizable to future datasets, and thus potentially less accurate in the predictions it makes.

Factor analysis works by detecting sets of variables which correlate highly with each other. These variables may then be condensed into a single variable. Data analysts will often carry out factor analysis to prepare the data for subsequent analyses.

Example: Let’s imagine you have a dataset containing data pertaining to a person’s income, education level, and occupation. You might find a high degree of correlation among each of these variables, and thus reduce them to the single factor “socioeconomic status.” You might also have data on how happy they were with customer service, how much they like a certain product, and how likely they are to recommend the product to a friend. Each of these variables could be grouped into the single factor “customer satisfaction” (as long as they are found to correlate strongly with one another). Even though you’ve reduced several data points to just one factor, you’re not really losing any information—these factors adequately capture and represent the individual variables concerned. With your “streamlined” dataset, you’re now ready to carry out further analyses.

2.4.7 Cluster analysis

Another interdependence technique, cluster analysis is used to group similar items within a dataset into clusters. When grouping data into clusters, the aim is for the variables in one cluster to be more similar to each other than they are to variables in other clusters. This is measured in terms of intracluster and intercluster distance. Intracluster distance looks at the distance between data points within one cluster. This should be small. Intercluster distance looks at the distance between data points in different clusters. This should ideally be large. Cluster analysis helps you to understand how data in your sample is distributed, and to find patterns.

Example: A prime example of cluster analysis is audience segmentation. If you were working in marketing, you might use cluster analysis to define different customer groups which could benefit from more targeted campaigns. As a healthcare analyst, you might use cluster analysis to explore whether certain lifestyle factors or geographical locations are associated with higher or lower cases of certain illnesses. Because it's an interdependence technique, cluster analysis is often carried out in the early stages of data analysis.

2.4.8 Advantages and Disadvantages of Multivariate Analysis

Advantages

- The main advantage of multivariate analysis is that since it considers more than one factor of independent variables that influence the variability of dependent variables, the conclusion drawn is more accurate.
- The conclusions are more realistic and nearer to the real-life situation.

Disadvantages

- The main disadvantage of MVA includes that it requires rather complex computations to arrive at a satisfactory conclusion.
- Many observations for a large number of variables need to be collected and tabulated; it is a rather time-consuming process

2.5 Graphical representation of Univariate, Bivariate and Multivariate Analysis in R

2.5.1 Bar Plot

A bar chart represents data in rectangular bars with length of the bar proportional to the value of the variable. R uses the function barplot() to create bar charts. R can draw both vertical and Horizontal bars in the bar chart. In bar chart each of the bars can be given different colors. The bar graph is very convenient while comparing categories of data or different groups of data. It helps to track changes over time. It is best for visualizing discrete data.

Example Script to Create a Bar Plot

```
# Create the data for the chart
H <- c(7,12,28,3,41)
M <- c("Mar","Apr","May","Jun","Jul")

# Give the chart file a name
png(file = "barchart_months_revenue.png")

# Plot the bar chart
```

```
barplot(H,names.arg=M,xlab="Month",ylab="Revenue",col="blue",
main="Revenue chart",border="red")

# Save the file
dev.off()
```

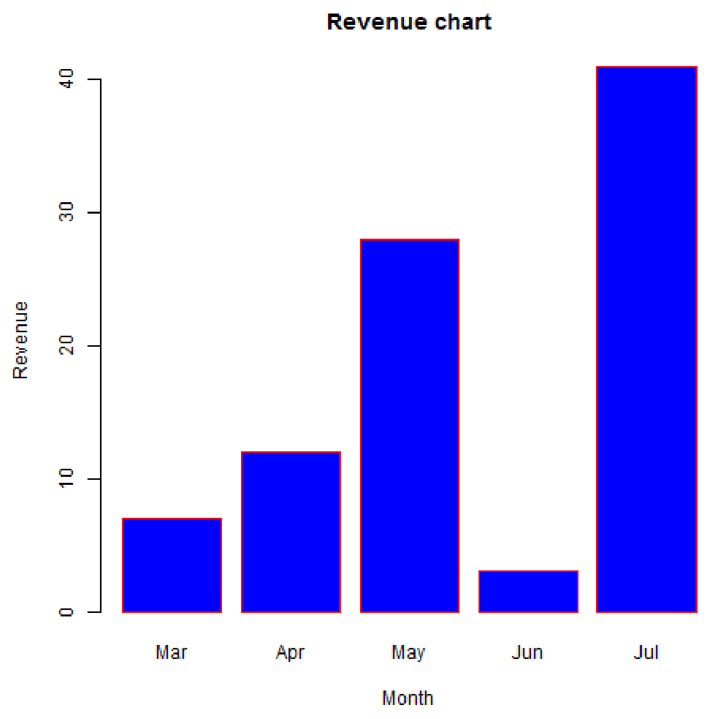


Figure 2.6: Bar Plot

2.5.2 Histogram

Histograms are similar to bar charts and display the same categorical variables against the category of data. Histograms display these categories as bins which indicate the number of data points in a range. It is best for visualizing continuous data.

R creates histogram using hist() function. This function takes a vector as an input and uses some more parameters to plot histograms.

Example Script to Create a Histograms

```
# Create data for the graph.
v <-  c(9,13,21,8,36,22,12,41,31,33,19)

# Give the chart file a name.
png(file = "histogram.png")
```

```
# Create the histogram.
hist(v,xlab = "Weight",col = "yellow",border = "blue")

# Save the file.
dev.off()
```

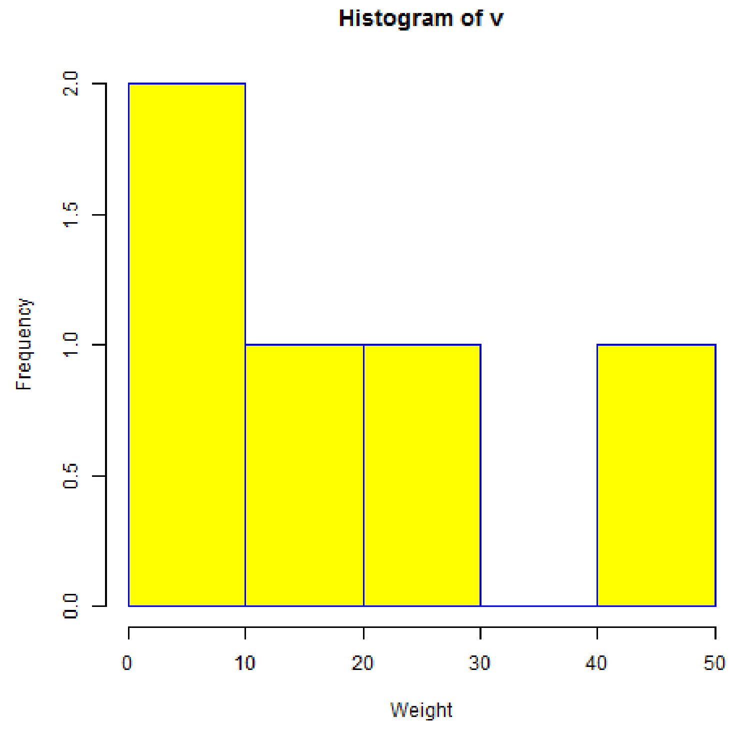


Figure 2.7: Histograms

2.5.3 Box Plot

A box graph is a chart that is used to display information in the form of distribution by drawing boxplots for each of them. This distribution of data based on five sets (minimum, first quartile, median, third quartile, maximum).It is also useful in comparing the distribution of data across data sets by drawing boxplots for each of them.

Boxplots are created in R by using the boxplot() function.

Example Script to Create a Boxplot

```
# Give the chart file a name.
png(file = "boxplot.png")

# Plot the chart.
boxplot(mpg ~ cyl, data = mtcars, xlab = "Number of Cylinders",
  ylab = "Miles Per Gallon", main = "Mileage Data")
```

```
# Save the file.
dev.off()
```

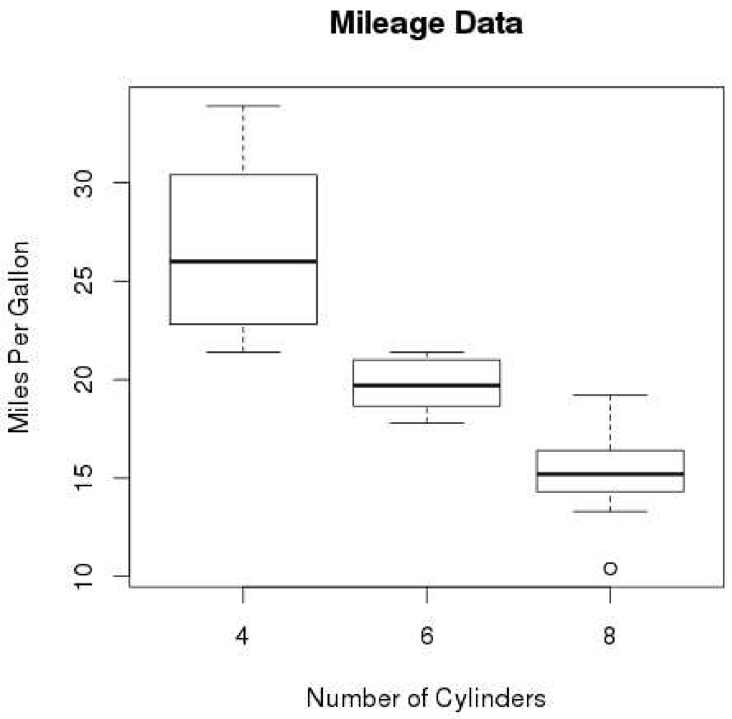


Figure 2.8: Boxplot

2.5.4 Line Plot

A line chart is a graph that connects a series of points by drawing line segments between them. These points are ordered in one of their coordinate (usually the x-coordinate) value. Line charts are usually used in identifying the trends in data.

The plot() function in R is used to create the line graph.

Example Script to Create a Line Plot

```
# Create the data for the chart.
v <- c(7,12,28,3,41)

# Give the chart file a name.
png(file = "line_chart.jpg")

# Plot the bar chart.
plot(v,type = "o")

# Save the file.
```

```
dev.off()
```

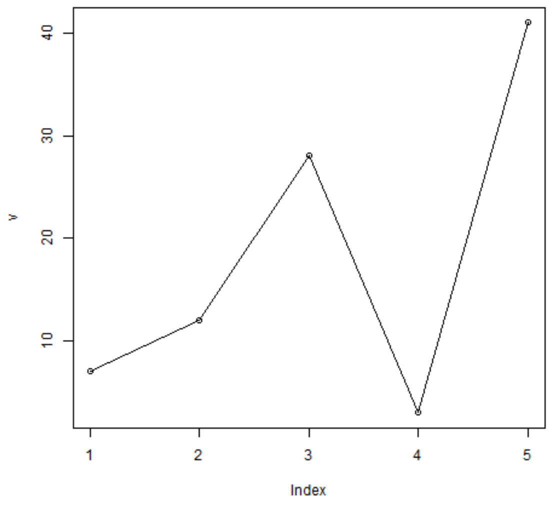


Figure 2.9: Line Plot

2.5.5 Scatter Plot

A scatter plot represents individual pieces of data using dots. These plots make it easier to see if two variables are related to each other. The resulting pattern indicates the type (linear or non-linear) and strength of the relationship between two variables.

The simple scatterplot is created using the plot() function.

Example Script to Create a Scatter Plot

```
# Get the input values.
input <- mtcars[,c('wt','mpg')]

# Give the chart file a name.
png(file = "scatterplot.png")

# Plot the chart for cars with weight between 2.5 to 5 and mileage between 15 and 30.
plot(x = input$wt,y = input$mpg,
   xlab = "Weight",
   ylab = "Milage",
   xlim = c(2.5,5),
   ylim = c(15,30),
   main = "Weight vs Milage"
)

# Save the file.
dev.off()
```

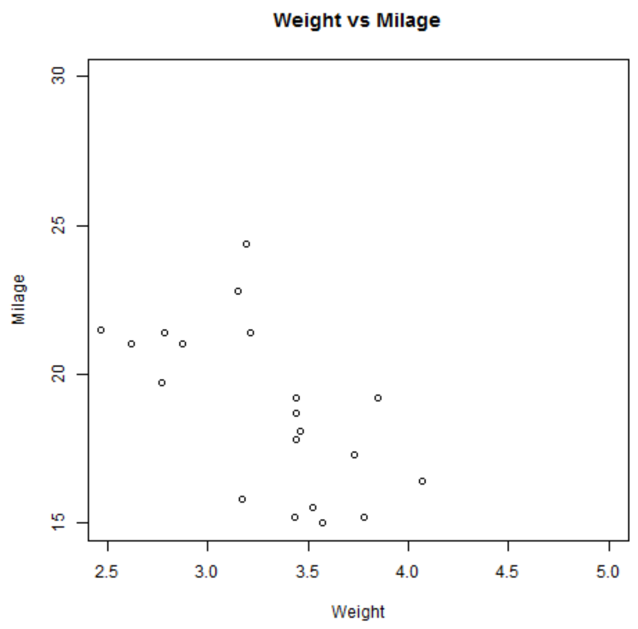


Figure 2.10: Scatter Plot

2.5.6 Lattice Plot

The lattice package is a graphics and data visualization package inspired by the trellis graphics package. The main focus of the package is multivariate data. It has a wide variety of functions that enable it to create basic plots of the base R package as well as enhance on them.

The xyplot() function can be used to create a scatter plot in R using the lattice package. The iris dataset is perfectly suited for this example.

Example Script to Create a Lattice Plot

```
xyplot(y ~ x, data)
my_data <- iris
head(my_data)
# Default plot
xyplot(Sepal.Length ~ Petal.Length, data = my_data)
```

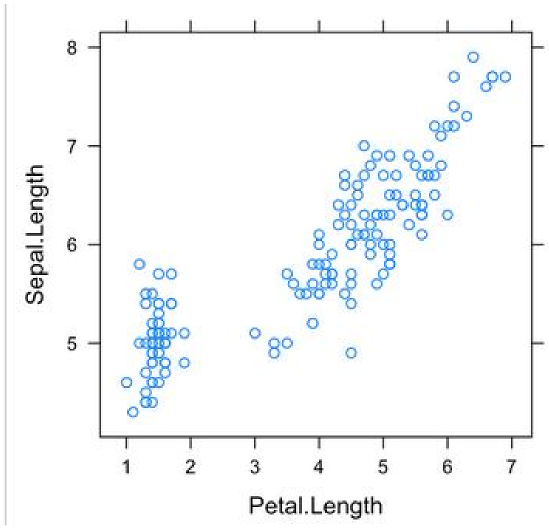


Figure 2.11: Lattice Plot

2.5.7 Regression Line

Regression analysis is a very widely used statistical tool to establish a relationship model between two variables. One of these variable is called predictor variable whose value is gathered through experiments. The other variable is called response variable whose value is derived from the predictor variable.

In Linear Regression these two variables are related through an equation, where exponent (power) of both these variables is 1. Mathematically a linear relationship represents a straight line when plotted as a graph. A non-linear relationship where the exponent of any variable is not equal to 1 creates a curve.

Steps to Establish a Regression

A simple example of regression is predicting weight of a person when his height is known. To do this we need to have the relationship between height and weight of a person.

The steps to create the relationship is

- Carry out the experiment of gathering a sample of observed values of height and corresponding weight.
- Create a relationship model using the lm() functions in R.
- Find the coefficients from the model created and create the mathematical equation using these
- Get a summary of the relationship model to know the average error in prediction. Also called residuals.
- To predict the weight of new persons, use the predict() function in R.

Script to create Relationship Model & get the Coefficients

```
x <- c(151, 174, 138, 186, 128, 136, 179, 163, 152, 131)
y <- c(63, 81, 56, 91, 47, 57, 76, 72, 62, 48)

# Apply the lm() function.
relation <- lm(y~x)

print(relation)
```

Script to get the Summary of the Relationship

```
x <- c(151, 174, 138, 186, 128, 136, 179, 163, 152, 131)
y <- c(63, 81, 56, 91, 47, 57, 76, 72, 62, 48)

# Apply the lm() function.
relation <- lm(y~x)

print(summary(relation))
```

Script to predict the weight of new persons

```
# The predictor vector.
x <- c(151, 174, 138, 186, 128, 136, 179, 163, 152, 131)

# The resposne vector.
y <- c(63, 81, 56, 91, 47, 57, 76, 72, 62, 48)

# Apply the lm() function.
relation <- lm(y~x)

# Find weight of a person with height 170.
a <- data.frame(x = 170)
result <-  predict(relation,a)
print(result)
```

Example Script to Create a Regression Line

```
# Create the predictor and response variable.
x <- c(151, 174, 138, 186, 128, 136, 179, 163, 152, 131)
y <- c(63, 81, 56, 91, 47, 57, 76, 72, 62, 48)
relation <- lm(y~x)
```

```
# Give the chart file a name.
png(file = "linearregression.png")

# Plot the chart.
plot(y,x,col = "blue",main = "Height & Weight Regression",
abline(lm(x~y)),cex = 1.3,pch = 16,xlab = "Weight in Kg",ylab = "Height in cm")

# Save the file.
dev.off()
```

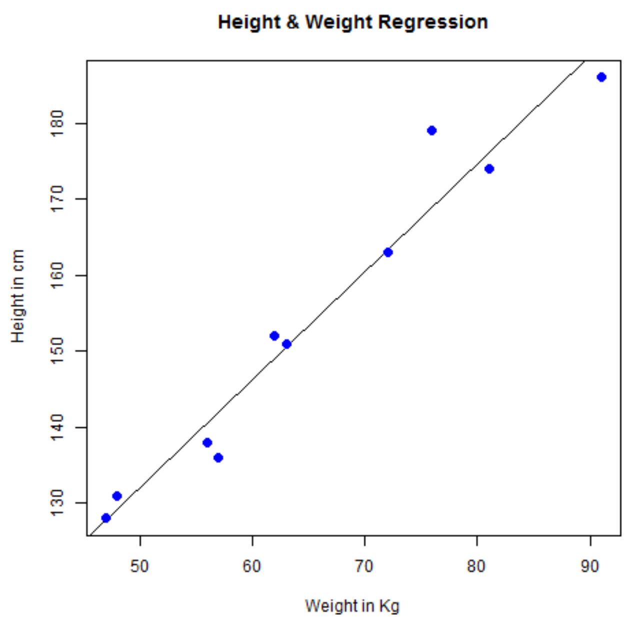


Figure 2.12: Regression Line

2.5.8 Two-Way cross Tabulation

A two-way table is used to display frequency for two categorical variables. The rows represent the categorical features and the column represents frequency. We can create two-way table using as.table() method. as.table() function in R Language is used to convert an object into a table.

Example Script to create a Two-Way cross Tabulation

```
# create dataframe with 2 columns
data = data.frame(col1=c("apple","mango","mango","guava","apple"),
            col2=c("fruit","veg","fruit","fruit","veg"))

# convert dataframe to table
data = table(data$col1,data$col2)

# display mosaicplot
```

```
mosaicplot(data, main='Eatables')
```

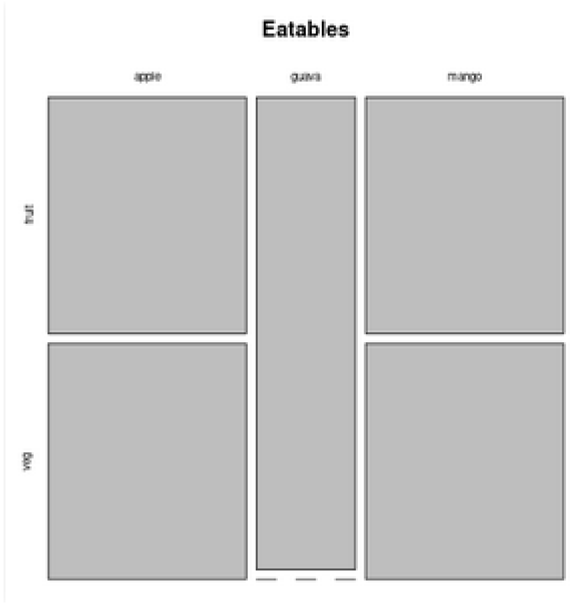


Figure 2.13: Two-Way cross Tabulation

SUMMARY

- Univariate analysis is basically the simplest form to analyze data. Uni means one and this means that the data has only one kind of variable. The major reason for univariate analysis is to use the data to describe. The analysis will take data, summarise it, and then find some pattern in the data.
- Mean is the average of the given numbers and is calculated by dividing the sum of given numbers by the total number of numbers.
- Median is a statistical measure that determines the middle value of a dataset listed in ascending order. The measure divides the lower half from the higher half of the dataset.
- A mode is defined as the value that has a higher frequency in a given set of values. It is the value that appears the most number of times.
- Variance is a statistical measurement that is used to determine the spread of numbers in a data set with respect to the average value or the mean.
- Standard deviation is the degree of dispersion or the scatter of the data points relative to its mean, in descriptive statistics. It tells how the values are spread across the data sample and it is the measure of the variation of the data points from the mean.
- Skewness essentially measures the symmetry of the distribution, while kurtosis determines the heaviness of the distribution tails.
- Bivariate analysis is one of the statistical analysis where two variables are observed. One variable here is dependent while the other is independent.
- Correlation is a statistical method used to assess a possible linear association between two continuous variables. It is simple both to calculate and to interpret.

- A regression model provides a function that describes the relationship between one or more independent variables and a response, dependent, or target variable.
- Multivariate Analysis is defined as a process of involving multiple dependent variables resulting in one outcome.

SELF-ASSESSMENT QUESTIONS

- Define Univariate analysis.
- What are the observation made on data to find out the patterns in Univariate analysis.
- Discuss about Skewness and Kurtosis.
- What is Positively Skewed and Negatively Skewed?
- Discuss on Excess Kurtosis.
- Explain Logistic Regression with an example.
- Discuss the data analysis techniques with examples in multivariate analysis.

FURTHER READINGS

- Bill Franks, “Taming the Big Data Tidal Wave: Finding Opportunities in Huge Data Streams with Advanced Analytics”, John Wiley & sons, First Edition,2013.
- Rachel Schutt, Cathy O'Neil, “Doing Data Science”, O'Reilly, First Edition,2013
- Foster Provost, Tom Fawcet, “Data Science for Business”, O'Reilly, First Edition,2013.
- Bart Baesens, “Analytics in a Big Data World: The Essential Guide to Data Science and its Applications”, Wiley, First Edition,2014.

3. Data Modelling

Data modelling is a very common terminology in software engineering and other IT disciplines. It has many interpretations and definitions depending on the field in discussion. In data science, data modelling is the process of finding the function by which data was generated. In this context, data modelling is the goal of any data analysis task. For instance if you have a 2d dataset, and finding the 2 variables are linearly correlated, This could be model using linear regression or if we visualize the data and find out it's non-linearly, it may modeled using nonlinear regression.

3.1 Bayesian Modeling

The Bayesian technique is an approach in statistics used in data analysis and parameter estimation. This approach is based on the Bayes theorem.

Bayesian Statistics follows a unique principle wherein it helps determine the joint probability distribution for observed and unobserved parameters using a statistical model. The knowledge of statistics is essential to tackle analytical problems in this scenario.

Ever since the introduction of the Bayes theorem in the 1770s by Thomas Bayes, it has remained an indispensable tool in statistics. Bayesian models are a classic replacement for frequentist models as recent innovations in statistics have helped breach milestones in a wide range of industries, including medical research, understanding web searches, and processing natural languages (Natural Language Processing).

Example: Alzheimer's is a disease known to pose a progressive risk as a person ages. However, with the help of the Bayes theorem, doctors can estimate the probability of a person having Alzheimer's in the future. It also applies to cancer and other age-related illnesses that a person becomes vulnerable to in the later years of his life.

3.1.1 Frequent Statistics Vs Bayesian Statistics

Frequent Statistics vs Bayesian statistics has consistently been a topic of controversy and nightmares for beginners, both of whom have difficulty choosing between the two. In the early 20th century, Bayesian statistics underwent its share of distrust and acceptance issues. With time, however, people realized the applicability of Bayesian models and the accurate solutions it yields.

3.1.2 Frequent Statistics

It is a widely used inferential methodology in the world of statistics. It analyzes whether or not an event (mentioned as a hypothesis) has taken place. It also estimates the probability of the event occurring during the span of the experiment. The experiment is repeated until the desired outcome is achieved.

Their distribution samples are of actual size, and the experiment is repeated infinite times theoretically.

Example: Showing how frequent statistics can be used to study the tossing of a coin.

- The possibility of getting a head on tossing the coin once is 0.5 (1/2).
- The number of heads denotes the actual number of leads obtained.
- The difference between the actual number of heads and the expected number of heads will increase as the number of tosses increases.

So here, the result depends on the number of times the experiment is repeated. It is a major drawback of frequent statistics.

3.1.3 Birth of Bayesian Statistics

Reverend Thomas Bayes first proposed the Bayesian approach to statistics in his essay written in 1763. This approach was published by Richard Price as a strategy in inverse probability to forecast future events based on the past.

Bayesian data modeling is to model your data using Bayes Theorem. Let us re-visit Bayes Rule again:

$$P(H|E) = \frac{P(E|H) * P(H)}{P(E)}$$

In the above equation, H is the hypothesis and E is the evidence. In the real world however, we understand Bayesian components differently! The evidence is usually expressed by data, and the hypothesis reflects the expert's prior estimation of the posterior. Therefore, we can re-write the Bayes Rule to be:

$$P(posterior) = \frac{P(data|\theta) * P(prior)}{P(data)}$$

In the above definition we learned about prior, posterior, and data, but what about θ parameter? θ is the set of coefficients that best define the data. You may think of θ as the set of slope and intercept of your linear regression equation, or the vector of coefficients ω in your polynomial regression function. As you see in the above equation, θ is the single missing parameter, and the goal of Bayesian modeling is to find it.

3.1.4 Bayesian Modeling & Probability Distributions

Bayes Rule is a probabilistic equation, where each term in it is expressed as a probability. Therefore, modeling the prior and the likelihood must be achieved using probabilistic functions. In this context arise probability distributions as a concrete tool in Bayesian modelling, as they provide a great variety of probabilistic functions that suits numerous styles of discrete and continuous variables.

In order to select the suitable distribution for your data, you should learn about the data domain and gather information from previous studies in it. You may also ask an expert to learn how data is developed over time. If you have big portions of data, you may visualize it and try to detect certain pattern(s) of its evolving over time, and select your probability distribution upon.

3.1.5 Maximum Likelihood Estimation (MLE)

Maximum Likelihood Estimation (MLE) is the simplest way of Bayesian data modelling, in which we ignore both prior and marginal probabilities (i.e. consider both quantities equal to 1). The formula of MLE is:

$$P(\text{posterior}) \propto P(\text{data}|\theta)$$

The steps of MLE are as follows:

- Select a probability distribution that best describes your data
- Estimate random value(s) of θ
- Tune θ value(s) and measure the corresponding likelihood
- Select θ that correspond to the maximum likelihood

Example:

"A company captures the blood glucose (BG) levels of diabetics, analyze these levels, and send its clients suitable diet preferences. After one week of inserting her BG levels, a client asked the company smart app whether she can consume desserts after her lunch or not? By checking her after-meal BG levels, it was {172,171,166,175,170,165,160}. Knowing that the client after-meal BG should not exceed 200 mg/dl, what should the app recommend to the client?"

The goal here is to estimate an average BG level for the client to use it in the company recommendation system. Having the above BG sample data, we assume these readings are drawn from a normal distribution, with a mean of 168.43 and standard deviation of 4.69. See figure 3.1

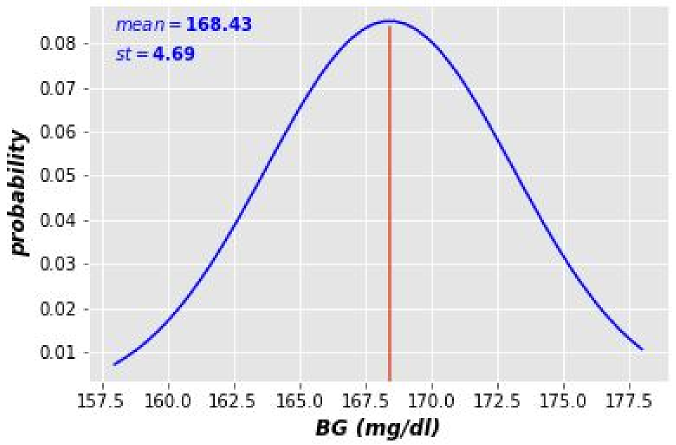


Figure 3.1: Example for MLE

Now we calculate the likelihood of this estimation as follows:

P(data|θ)=P({172,171,166,175,170,165,160}|168.43)=P(172|168.43)×P(171|168.43)×P(166|168.43)×P(175|168.43)×P(170|168.43)×P(165|168.43)×P(160|168.43)

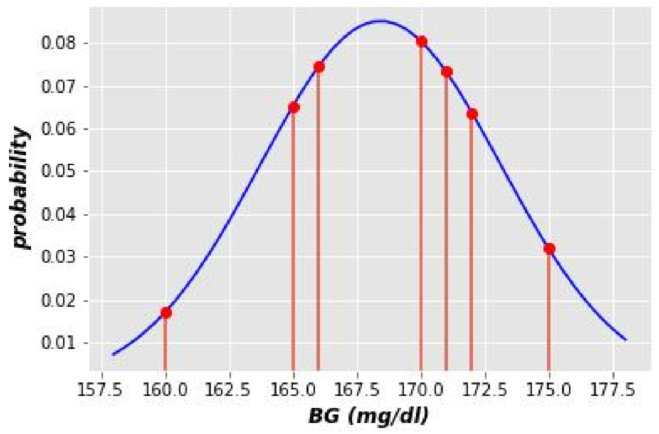


Figure 3.2: Example for MLE

Now we consider the multiplying small probability values (red dots in figure 3.2) generates very small value (very close to 0). Therefore, we replace multiplication with summation. In the above case, the sum of these probabilities equal 0.406:

P(data|θ)=P({172,171,166,175,170,165,160}|168.43)=P(172|168.43)+P(171|168.43)+P(166|168.43)+P(175|168.43)+P(170|168.43)+P(165|168.43)+P(160|168.43)=0.406

In order to find the maximum likelihood, we need to estimate different values of BG levels and calculate the corresponding likelihoods. The BG value with maximum likelihood is the most suitable estimation of the BG.

To automate this process, we generate 1000 random numbers in the same range of captured BG levels, and measure the corresponded likelihoods. The results are illustrated in the following plot figure 3.3.

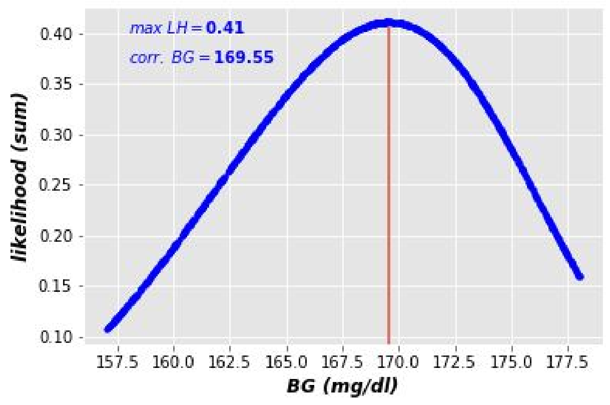


Figure 3.3: Example for MLE

As you can see, the maximum likelihood estimate of randomly generated BGs equals 169.55, which is very close to the average of captured BG levels (168.43). The difference between both values is due to the small size of captured data. The larger sample size you have, the smaller difference between both estimates you get.

Based on this estimate, the app can recommend it’s patient to consume a small piece of dessert after at least 3 hours of her lunch, with taking the suitable insulin dose.

3.1.6 Maximum A Posteriori (MAP)

Maximum A Posteriori (MAP) is the second approach of Bayesian modelling, where the posterior is calculated using both likelihood and prior probabilities.

$$P(\text{posterior}) \propto P(\text{data}|\theta) \times P(\text{Prior})$$

Consider MAP modeling as a generalization of MLE, where the likelihood is approximated with prior information. Technically speaking, we deal with our data samples as if it were generated by the prior distribution:

$$P(\text{posterior}) \propto P(\text{data}|\theta) \times P(\theta)$$

Example: ”Suppose that the client updated the app with her historical BG levels for the last month, which turned out to be 171 mg/dl on average, with standard deviation of 3, how the app can use this new information to update the patient’s BG level? “

In this case, we have two distributions of Blood Glucose, one of recent data (the likelihood), and the other of historical data (the prior). Each data source can be expressed as normal distribution (see the figure 3.4).

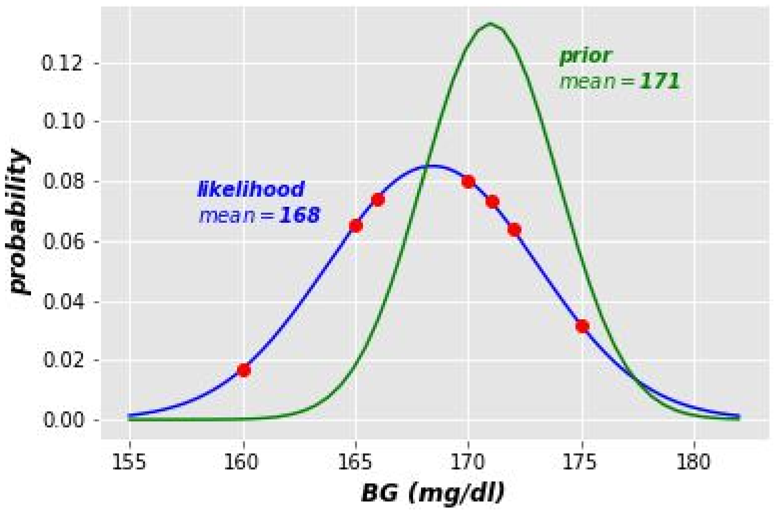


Figure 3.4: Example for MAP

The posterior here is updated by multiplying the prior marginal probability by each term of data probabilities:

$$P(data|\theta)=P(\{172,171,166,175,170,165,160\}|\theta)\times P(\theta)=P(172|\theta)\times P(\theta)\times P(171|\theta)\times P(\theta)\times P(166|\theta)\times P(\theta)\times P(175|\theta)\times P(\theta)\times P(170|\theta)\times P(\theta)\times P(165|\theta)\times P(\theta)\times P(160|\theta)\times P(\theta)$$

The prior marginal probability P(θ) is the summation of all data probabilities over the prior distribution:

$$P(\theta)=P(172|\theta)+P(171|\theta)+P(166|\theta)+P(175|\theta)+P(170|\theta)+P(165|\theta)+P(160|\theta)$$

As our goal is to maximize posterior estimation, we generate random values of θ and measure the corresponding estimations. By generating 500 guesses of θ, we obtain the following posterior distribution (in black). For visualization reasons, I raised the value of posterior probabilities to the power of 0.08.

Now we get more generalized estimation of the client's BG level. The MAP technique excludes two measures in the posterior distribution (the red data points outside the black curve), and generates more convenient estimate. As the below plot shows, the Standard Error (SE) of the posterior is less than that of the likelihood. Standard error is an indication of the reliability of a standard expectation (i.e. mean of predicted normal distribution). It is calculated as SE = $\sigma/\sqrt{N}$, where σ is the standard deviation and N is data size.

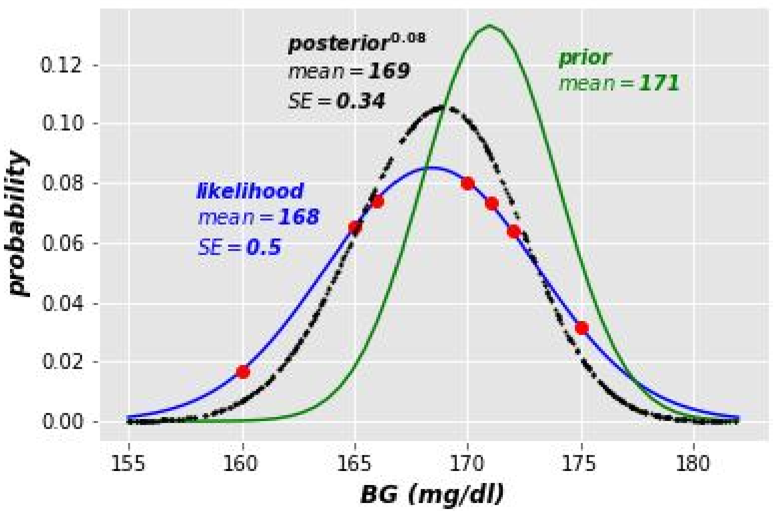


Figure 3.5: Example for MAP

The new MAP posterior estimation of the patient's BG level is higher than the one estimated using MLE show in figure 3.5. This may lead the app to prohibit the patient of consuming desserts after her lunch, until her BG levels become more stabilized.

3.1.7 Advantages of Bayesian Statistics

- It provides an organic and simple means to blend pre-conceived information with a solid framework with scientific evidence. The past information about a parameter can be used to form a prior distribution for future investigation. The inferences adhere to the Bayes theorem.
- The inferences from a Bayesian model are logical and mathematically accurate and not crude assumptions. The accuracy remains constant irrespective of the size of the sample.
- Bayesian statistics follow the likelihood principle. When two different samples have a common likelihood function for a belief θ, all inferences about the belief should be similar. Classical statistical techniques do not follow the likelihood principle.
- The solutions from a Bayesian analysis can be easily interpreted.
- It offers a conducive platform for various models like hierarchical models and incomplete data issues. The computations of all parametric models can be virtually tracked with the help of other numerical techniques.

3.2 Support Vector and Kernel Methods

Kernels or kernel methods (also called Kernel functions) are sets of different types of algorithms that are being used for pattern analysis. They are used to solve a non-linear problem by using a linear classifier. Kernels Methods are employed in SVM (Support Vector Machines) which are used in classification and regression problems. The SVM uses what is called a "Kernel Trick" where the data is transformed and an optimal boundary is found for the possible outputs.

3.2.1 The Need for Kernel Method and its Working

Before we get into the working of the Kernel Methods, it is more important to understand support vector machines or the SVMs because kernels are implemented in SVM models. So, Support Vector Machines are supervised machine learning algorithms that are used in classification and regression problems such as classifying an apple to class fruit while classifying a Lion to the class animal.

To demonstrate, below is what support vector machines look like:

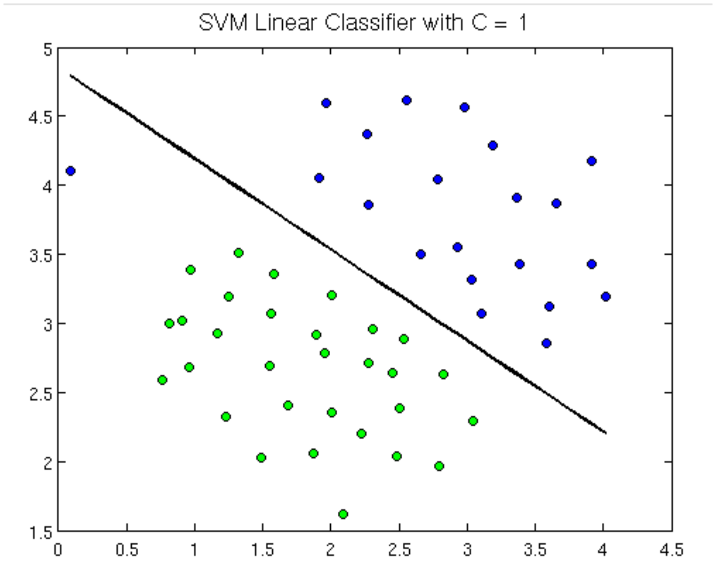


Figure 3.6: Support vector machines

Here we can see a hyper plane which is separating green dots from the blue ones. A hyper plane is one dimension less than the ambient plane. E.g. in the figure 3.6, we have 2 dimension which represents the ambient space but the lone which divides or classifies the space is one dimension less than the ambient space and is called hyper plane.

But what if we have input like this:

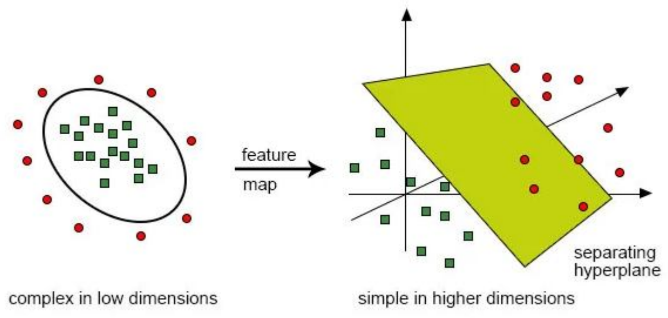


Figure 3.7: Separation in higher dimensions

It is very difficult to solve this classification using a linear classifier as there is no good linear line that should be able to classify the red and the green dots as the points are randomly distributed. Here comes the use of kernel function which takes the points to higher dimensions, solves the problem over there and returns the output. Think of this in this way, we can see that the green dots are enclosed in some perimeter area while the red one lies outside it, likewise, there could be other scenarios where green dots might be distributed in a trapezoid-shaped area.

So what we do is to convert the two-dimensional plane which was first classified by one-dimensional hyper plane ("or a straight line") to the three-dimensional area and here our classifier i.e. hyper plane will not be a straight line but a two-dimensional plane which will cut the area.

In order to get a mathematical understanding of kernel, let us understand the Lili Jiang's equation of kernel which is:

K(x, y)=<f(x), f(y)> where,
K is the kernel function,
X and Y are the dimensional inputs,
f is the map from n-dimensional to m-dimensional space and,
< x, y > is the dot product.

Example:
Let us say that we have two points, x= (2, 3, 4) and y= (3, 4, 5)
As we have seen, K(x, y) = < f(x), f(y) >.
Let us first calculate < f(x), f(y) >
f(x)=(x1x1, x1x2, x1x3, x2x1, x2x2, x2x3, x3x1, x3x2, x3x3)
f(y)=(y1y1, y1y2, y1y3, y2y1, y2y2, y2y3, y3y1, y3y2, y3y3)

so,
f(2, 3, 4)=(4, 6, 8, 6, 9, 12, 8, 12, 16)and
f(3 ,4, 5)=(9, 12, 15, 12, 16, 20, 15, 20, 25)

so the dot product,
f (x). f (y) = f(2,3,4) . f(3,4,5)=
(36 + 72 + 120 + 72 +144 + 240 + 120 + 240 + 400) = 1444
And,
K(x, y) = (2*3 + 3*4 + 4*5) ^2=(6 + 12 + 20)^2=38*38 = 1444.

This as we find out, f(x).f(y) and K(x, y) give us the same result, but the former method required a lot of calculations(because of projecting 3 dimensions into 9 dimensions) while using the kernel, it was much easier.

3.2.2 Types of Kernel and methods in SVM

1. Liner Kernel

Let us say that we have two vectors with name x1 and Y1, then the linear kernel is defined by the dot product of these two vectors: K(x1, x2) = x1 . x2

2. Polynomial Kernel

A polynomial kernel is defined by the following equation:

K(x1, x2) = (x1 . x2 + 1)d,

Where, d is the degree of the polynomial and x1 and x2 are vectors

3. Gaussian Kernel

This kernel is an example of a radial basis function kernel. Below is the equation for this:

$$K(x_i, x_j) = exp(-\sigma \left\|x_i - x_j\right\|^2)$$

The given sigma plays a very important role in the performance of the Gaussian kernel and should neither be overestimated and nor be underestimated, it should be carefully tuned according to the problem.

4. Exponential Kernel

This is in close relation with the previous kernel i.e. the Gaussian kernel with the only difference is the square of the norm is removed.

The function of the exponential function is:

$$K(x, y) = \exp\left(-\frac{\|x - y\|}{2\sigma^2}\right)$$

This is also a radial basis kernel function.

5. Laplacian Kernel

This type of kernel is less prone for changes and is totally equal to previously discussed exponential function kernel, the equation of Laplacian kernel is given as:

$$K(x,y) = \exp\left(-\frac{||x-y||}{\sigma}\right)$$

6. Hyperbolic or the Sigmoid Kernel

This kernel is used in neural network areas of machine learning. The activation function for the sigmoid kernel is the bipolar sigmoid function. The equation for the hyperbolic kernel function is:

$$K(x,y) = tanh(\alpha x^T y + c)$$

This kernel is very much used and popular among support vector machines.

7. Anova radial basis kernel

This kernel is known to perform very well in multidimensional regression problems just like the Gaussian and Laplacian kernels. This also comes under the category of radial basis kernel.

The equation for Anova kernel is:

$$K(x,y) = \Sigma_{k=1}^{n} \exp\left(-\sigma\left(x^k - y^k\right)^2\right)^d$$

There are a lot more types of Kernel Method and we have discussed the mostly used kernels. It purely depends on the type of problem which will decide the kernel function to be used.

3.3 Neuro–Fuzzy Modeling

A neuro-fuzzy system is a fuzzy system that uses a learning algorithm derived from or inspired by neural network theory to determine its parameters (fuzzy sets and fuzzy rules) by processing data samples.

3.3.1 Combining fuzzy systems with neural networks

Both neural networks and fuzzy systems have some things in common. They can be used for solving a problem (e.g. pattern recognition, regression or density estimation) if there does not exist any mathematical model of the given problem. They solely do have certain disadvantages and advantages which almost completely disappear by combining both concepts.

Neural networks can only come into play if the problem is expressed by a sufficient amount of observed examples. These observations are used to train the black box. On the one hand no prior knowledge about the problem needs to be given. On the other hand, however, it is not straightforward to extract comprehensible rules from the neural network's structure.

On the contrary, a fuzzy system demands linguistic rules instead of learning examples as prior knowledge. Furthermore the input and output variables have to be described linguistically. If the knowledge is incomplete, wrong or contradictory, then the fuzzy system must be tuned. Since there is not any formal approach for it, the tuning is performed in a heuristic way. This is usually very time consuming and error-prone.

3.3.2 Characteristics

Compared to a common neural network, connection weights and propagation and activation functions of fuzzy neural networks differ a lot. Although there are many different approaches to model a fuzzy neural network (Buckley and Hayashi, 1994, 1995; Nauck and Kruse, 1996), most of them agree on certain characteristics such as the following:

- A neuro-fuzzy system based on an underlying fuzzy system is trained by means of a data-driven learning method derived from neural network theory. This heuristic only takes into account local information to cause local changes in the fundamental fuzzy system.
- It can be represented as a set of fuzzy rules at any time of the learning process, i.e., before, during and after.
    - Thus the system might be initialized with or without prior knowledge in terms of fuzzy rules.
- The learning procedure is constrained to ensure the semantic properties of the underlying fuzzy system.
- A neuro-fuzzy system approximates a n-dimensional unknown function which is partly represented by training examples.
    - Fuzzy rules can thus be interpreted as vague prototypes of the training data.
- A neuro-fuzzy system is represented as special three-layer feedforward neural network.
    - The first layer corresponds to the input variables.
    - The second layer symbolizes the fuzzy rules.
    - The third layer represents the output variables.
    - The fuzzy sets are converted as (fuzzy) connection weights.
    - Some approaches also use five layers where the fuzzy sets are encoded in the units of the second and fourth layer, respectively. However, these models can be transformed into a three-layer architecture.

3.4 Principal Component Analysis

Principal Component Analysis is an unsupervised learning algorithm that is used for the dimensionality reduction in machine learning. It is a statistical process that converts the observations of correlated features into a set of linearly uncorrelated features with the help of orthogonal transformation. These new transformed features are called the Principal Components. It is one of the popular tools that is used for exploratory data analysis and predictive modeling. It is a technique to draw strong patterns from the given dataset by reducing the variances.

PCA generally tries to find the lower-dimensional surface to project the high-dimensional data.

PCA works by considering the variance of each attribute because the high attribute shows the good split between the classes, and hence it reduces the dimensionality. Some real-world applications of PCA are image processing, movie recommendation system, optimizing the power allocation in various communication channels. It is a feature extraction technique, so it contains the important variables and drops the least important variable.

The PCA algorithm is based on some mathematical concepts such as:

- Variance and Covariance
- Eigenvalues and Eigen factors

Some common terms used in PCA algorithm:

- **Dimensionality**: It is the number of features or variables present in the given dataset. More easily, it is the number of columns present in the dataset.
- **Correlation**: It signifies that how strongly two variables are related to each other. Such as if one changes, the other variable also gets changed. The correlation value ranges from -1 to +1. Here, -1 occurs if variables are inversely proportional to each other, and +1 indicates that variables are directly proportional to each other.
- **Orthogonal**: It defines that variables are not correlated to each other, and hence the correlation between the pair of variables is zero.
- **Eigenvectors**: If there is a square matrix M, and a non-zero vector v is given. Then v will be eigenvector if Av is the scalar multiple of v.
- **Covariance Matrix**: A matrix containing the covariance between the pair of variables is called the Covariance Matrix.

3.4.1 Principal Components in PCA

As described above, the transformed new features or the output of PCA are the Principal Components. The numbers of these PCs are either equal to or less than the original features present in the dataset. Some properties of these principal components are given below:

- The principal component must be the linear combination of the original features.
- These components are orthogonal, i.e., the correlation between a pair of variables is zero.
- The importance of each component decreases when going to 1 to n, it means the 1 PC has the most importance, and n PC will have the least importance.

3.4.2 Steps for PCA algorithm

1. **Getting the dataset**
   Firstly, we need to take the input dataset and divide it into two subparts X and Y, where X is the training set, and Y is the validation set.

2. **Representing data into a structure**
   Now we will represent our dataset into a structure. Such as we will represent the two-dimensional matrix of independent variable X. Here each row corresponds to the data items, and the column corresponds to the Features. The number of columns is the dimensions of the dataset.
3. **Standardizing the data**
   In this step, we will standardize our dataset. Such as in a particular column, the features with high variance are more important compared to the features with lower variance. If the importance of features is independent of the variance of the feature, then we will divide each data item in a column with the standard deviation of the column. Here we will name the matrix as Z.
4. **Calculating the Covariance of Z**
   To calculate the covariance of Z, we will take the matrix Z, and will transpose it. After transpose, we will multiply it by Z. The output matrix will be the Covariance matrix of Z.
5. **Calculating the Eigen Values and Eigen Vectors**
   Now we need to calculate the eigenvalues and eigenvectors for the resultant covariance matrix Z. Eigenvectors or the covariance matrix are the directions of the axes with high information. And the coefficients of these eigenvectors are defined as the eigenvalues.
6. **Sorting the Eigen Vectors**
   In this step, we will take all the eigenvalues and will sort them in decreasing order, which means from largest to smallest. And simultaneously sort the eigenvectors accordingly in matrix P of eigenvalues. The resultant matrix will be named as P*.
7. **Calculating the new features Or Principal Components**
   Here we will calculate the new features. To do this, we will multiply the P* matrix to the Z. In the resultant matrix Z*, each observation is the linear combination of original features. Each column of the Z* matrix is independent of each other.
8. **Remove less or unimportant features from the new dataset.**
   The new feature set has occurred, so we will decide here what to keep and what to remove. It means, we will only keep the relevant or important features in the new dataset, and unimportant features will be removed out.

3.4.3 Applications of Principal Component Analysis

- PCA is mainly used as the dimensionality reduction technique in various AI applications such as computer vision, image compression, etc.
- It can also be used for finding hidden patterns if data has high dimensions. Some fields where PCA is used are Finance, data mining, Psychology, etc.

3.5 Introduction to NoSQL

NoSQL Database is a non-relational Data Management System, that does not require a fixed schema. It avoids joins, and is easy to scale. The major purpose of using a NoSQL database is for distributed data stores with humongous data storage needs. NoSQL is used for Big data and real-time web apps. For example, companies like Twitter, Facebook and Google collect terabytes of user data every single day.

3.5.1 Why NoSQL?

The concept of NoSQL databases became popular with Internet giants like Google, Facebook, Amazon, etc. who deal with huge volumes of data. The system response time becomes slow when you use RDBMS for massive volumes of data.

To resolve this problem, we could "scale up" our systems by upgrading our existing hardware. This process is expensive. The alternative for this issue is to distribute database load on multiple hosts whenever the load increases. This method is known as "scaling out."

3.5.2 Brief History of NoSQL Databases

- 1998- Carlo Strozzi use the term NoSQL for his lightweight, open-source relational database
- 2000- Graph database Neo4j is launched
- 2004- Google BigTable is launched
- 2005- CouchDB is launched
- 2007- The research paper on Amazon Dynamo is released
- 2008- Facebooks open sources the Cassandra project
- 2009- The term NoSQL was reintroduced

3.5.3 Features of NoSQL

Non-relational

- NoSQL databases never follow the relational model
- Never provide tables with flat fixed-column records
- Work with self-contained aggregates or BLOBs
- Doesn't require object-relational mapping and data normalization
- No complex features like query languages, query planners, referential integrity joins, ACID

Schema-free

- NoSQL databases are either schema-free or have relaxed schemas

- Do not require any sort of definition of the schema of the data
- Offers heterogeneous structures of data in the same domain

Simple API

- Offers easy to use interfaces for storage and querying data provided
- APIs allow low-level data manipulation & selection methods
- Text-based protocols mostly used with HTTP REST with JSON
- Mostly used no standard based NoSQL query language
- Web-enabled databases running as internet-facing services

Distributed

- Multiple NoSQL databases can be executed in a distributed fashion
- Offers auto-scaling and fail-over capabilities
- Often ACID concept can be sacrificed for scalability and throughput
- Mostly no synchronous replication between distributed nodes Asynchronous Multi-Master Replication, peer-to-peer, HDFS Replication
- Only providing eventual consistency
- Shared Nothing Architecture. This enables less coordination and higher distribution.

3.5.4 Types of NoSQL Databases

NoSQL Databases are mainly categorized into four types: Key-value pair, Column-oriented, Graph-based and Document-oriented. Every category has its unique attributes and limitations. None of the above-specified database is better to solve all the problems. Users should select the database based on their product needs.

Key Value Pair Based

Data is stored in key/value pairs. It is designed in such a way to handle lots of data and heavy load. Key-value pair storage databases store data as a hash table where each key is unique, and the value can be a JSON, BLOB(Binary Large Objects), string, etc.

It is one of the most basic NoSQL database example. This kind of NoSQL database is used as a collection, dictionaries, associative arrays, etc. Key value stores help the developer to store schema-less data. They work best for shopping cart contents.

Column-based

Column-oriented databases work on columns and are based on BigTable paper by Google. Every column is treated separately. Values of single column databases are stored contiguously.

They deliver high performance on aggregation queries like SUM, COUNT, AVG, MIN etc. as the data is readily available in a column.

Column-based NoSQL databases are widely used to manage data warehouses, business intelligence, CRM, Library card catalogs,

HBase, Cassandra, HBase, Hypertable are NoSQL query examples of column based database.

Document-Oriented:

Document-Oriented NoSQL DB stores and retrieves data as a key value pair but the value part is stored as a document. The document is stored in JSON or XML formats. The value is understood by the DB and can be queried.

The document type is mostly used for CMS systems, blogging platforms, real-time analytics & e-commerce applications. It should not use for complex transactions which require multiple operations or queries against varying aggregate structures.

Amazon SimpleDB, CouchDB, MongoDB, Riak, Lotus Notes, MongoDB, are popular Document originated DBMS systems.

Graph-Based

A graph type database stores entities as well the relations amongst those entities. The entity is stored as a node with the relationship as edges. An edge gives a relationship between nodes. Every node and edge has a unique identifier.

Compared to a relational database where tables are loosely connected, a Graph database is a multi-relational in nature. Traversing relationship is fast as they are already captured into the DB, and there is no need to calculate them.

Graph base database mostly used for social networks, logistics, spatial data.

Neo4J, Infinite Graph, OrientDB, FlockDB are some popular graph-based databases.

3.5.5. CAP Theorem

CAP theorem is also called brewer’s theorem. It states that is impossible for a distributed data store to offer more than two out of three guarantees

- Consistency
- Availability
- Partition Tolerance

Consistency:

The data should remain consistent even after the execution of an operation. This means once data is written, any future read request should contain that data. For example, after updating the order status, all the clients should be able to see the same data.

Availability:

The database should always be available and responsive. It should not have any downtime.

Partition Tolerance:

Partition Tolerance means that the system should continue to function even if the communication among the servers is not stable. For example, the servers can be partitioned into multiple groups which may not communicate with each other. Here, if part of the database is unavailable, other parts are always unaffected.

3.5.6. MongoDB

MongoDB is a No SQL database. It is an open-source, cross-platform, document-oriented database written in C++.

Our MongoDB tutorial includes all topics of MongoDB database such as insert documents, update documents, delete documents, query documents, projection, sort() and limit() methods, create a collection, drop collection, etc. There are also given MongoDB interview questions to help you better understand the MongoDB database.

MongoDB is an open-source document database that provides high performance, high availability, and automatic scaling. In simple words, you can say that - Mongo DB is a document-oriented database. It is an open source product, developed and supported by a company named 10gen.

MongoDB is available under General Public license for free, and it is also available under Commercial license from the manufacturer.

History of MongoDB

The initial development of MongoDB began in 2007 when the company was building a platform as a service similar to window azure.

MongoDB was developed by a NewYork based organization named 10gen which is now known as MongoDB Inc. It was initially developed as a PAAS (Platform as a Service). Later in 2009, it is introduced in the market as an open source database server that was maintained and supported by MongoDB Inc.

The first ready production of MongoDB has been considered from version 1.4 which was released in March 2010.
MongoDB2.4.9 was the latest and stable version which was released on January 10, 2014.

Purpose of building MongoDB

It may be a very genuine question that - "what was the need of MongoDB although there were many databases in action?"

There is a simple answer: All the modern applications require big data, fast features development, flexible deployment, and the older database systems not competent enough, so the MongoDB was needed.

The primary purpose of building MongoDB is:

- Scalability
- Performance
- High Availability
- Scaling from single server deployments to large, complex multi-site architectures.
- Key points of MongoDB
- Develop Faster
- Deploy Easier
- Scale Bigger

3.5.7. RDBMS Vs MongoDB

Sr. No.	Key	RDBMS	MongoDB

1	Concept	RDBMS is a relational database management system and works on relational database.	MongoDB is a non-relational, document oriented database management system and works on document based database
2	Hiearchical	Difficult to store hiearchical data.	Have inbuilt support to store hiearchical data
3	Scalablity	RDBMS is vertically scalable. Performance increases with increase of RAM.	MongoDB is horizontally scalable as well. Its performance increases with addition of processor.
4	Schema	Schema need to be defined in RDBMS before using a database.	Schema can be dynamically created and accessed in MongoDB.
5	SQL Injection	Vulnerable to SQL Injection attack.	SQL injection is not possible.
6	Principle	Follows ACID principle, Atomicity, Consistency, Isolation, and Durability.	Follows CAP theorem, Consistency, Availability, and Partition tolerance.
7	Basis	Database uses Row.	Database uses Document.
8	Basis	Database uses Column.	Database uses Field.
9	Performance	RDBMS is slower in processing large hierachical data.	MongoDB is blazingly fast in processing large hierachical data.
10	Joins	RDBMS supports complex joins.	MongoDB has no support for complex joins.
11	JavaScript Client	RDBMS do not provide JavaScript based client to query database.	MongoDB provides Javascript based client to query database.
12	Query Language	RDBMS uses SQL to query database.	MongoDB uses BSON to query database.

3.5.8. Mongo DB Database Model

MongoDB provides two types of data models: Embedded data model and Normalized data model. Based on the requirement, you can use either of the models while preparing your document.

Embedded Data Model

In this model, you can embed all the related data in a single document, it is also known as de-normalized data model.

Example: Assume we are getting the details of employees in three different documents namely, Personal_details, Contact and, Address, you can embed all the three documents in a single one as shown below

```
{
	_id: ,
	Emp_ID: "10025AE336"
	Personal_details:{
		First_Name: "Radhika",
		Last_Name: "Sharma",
		Date_Of_Birth: "1995-09-26"
	},
	Contact: {
		e-mail: "radhika_sharma.123@gmail.com",
		phone: "9848022338"
	},
	Address: {
		city: "Hyderabad",
		Area: "Madapur",
		State: "Telangana"
	}
}
```

Normalized Data Model

In this model, you can refer the sub documents in the original document, using references.

Employee:

```
{
	_id: <ObjectId101>,
	Emp_ID: "10025AE336"
}
```

Personal_details:

```
{
	_id: <ObjectId102>,
	empDocID: " ObjectId101",
	First_Name: "Radhika",
	Last_Name: "Sharma",
	Date_Of_Birth: "1995-09-26"
}
```

Contact:

```
{
	_id: <ObjectId103>,
        empDocID: " ObjectId101",
        e-mail: "radhika_sharma.123@gmail.com",
        phone: "9848022338"
}
```

Address:

```
{
        _id: <ObjectId104>,
        empDocID: " ObjectId101",
        city: "Hyderabad",
        Area: "Madapur",
        State: "Telangana"
}
```

Considerations while designing Schema in MongoDB

- Design your schema according to user requirements.
- Combine objects into one document if you will use them together. Otherwise separate them (but make sure there should not be need of joins).
- Duplicate the data (but limited) because disk space is cheap as compare to compute time.
- Do joins while write, not on read.
- Optimize your schema for most frequent use cases.
- Do complex aggregation in the schema.

3.5.9 Data Types and Sharding

Following is a list of usable data types in MongoDB.

Data Types	**Description**
String	String is the most commonly used datatype. It is used to store data. A string must be UTF 8 valid in mongodb.
Integer	Integer is used to store the numeric value. It can be 32 bit or 64 bit depending on the server you are using.
Boolean	This datatype is used to store boolean values. It just shows YES/NO values.
Double	Double datatype stores floating point values.
Min/Max Keys	This datatype compare a value against the lowest and highest bson elements.
Arrays	This datatype is used to store a list or multiple values into a single key.
Object	Object datatype is used for embedded documents.
Null	It is used to store null values.
Symbol	It is generally used for languages that use a specific type.
Date	This datatype stores the current date or time in unix time format. It makes you possible to specify your own date time by creating object of date and pass the

	value of date, month, year into it.

Sharding

Sharding is the process of distributing data across multiple hosts. In MongoDB, sharding is achieved by splitting large data sets into small data sets across multiple MongoDB instances.

How sharding works

When dealing with high throughput applications or very large databases, the underlying hardware becomes the main limitation. High query rates can stress the CPU, RAM, and I/O capacity of disk drives resulting in a poor end-user experience.

To mitigate this problem, there are two types of scaling methods.

Vertical scaling

Vertical scaling is the traditional way of increasing the hardware capabilities of a single server. The process involves upgrading the CPU, RAM, and storage capacity. However, upgrading a single server is often challenged by technological limitations and cost constraints.

Horizontal scaling

This method divides the dataset into multiple servers and distributes the database load among each server instance. Distributing the load reduces the strain on the required hardware resources and provides redundancy in case of a failure.

However, horizontal scaling increases the complexity of underlying architecture. MongoDB supports horizontal scaling through sharding—one of its major benefits, as we'll see below.

MongoDB sharding basics

MongoDB sharding works by creating a cluster of MongoDB instances consisting of at least three servers. That means sharded clusters consist of three main components:

- The shard
- Mongos
- Config servers

Shard

A shard is a single MongoDB instance that holds a subset of the sharded data. Shards can be deployed as replica sets to increase availability and provide redundancy. The combination of multiple shards creates a complete data set. For example, a 2 TB data set can be broken down into four shards, each containing 500 GB of data from the original data set.

Mongos

Mongos act as the query router providing a stable interface between the application and the sharded cluster. This MongoDB instance is responsible for routing the client requests to the correct shard.

Config Servers

Configuration servers store the metadata and the configuration settings for the whole cluster.

Components illustrated

The following diagram from the official MongoDB docs explains the relationship between each component:

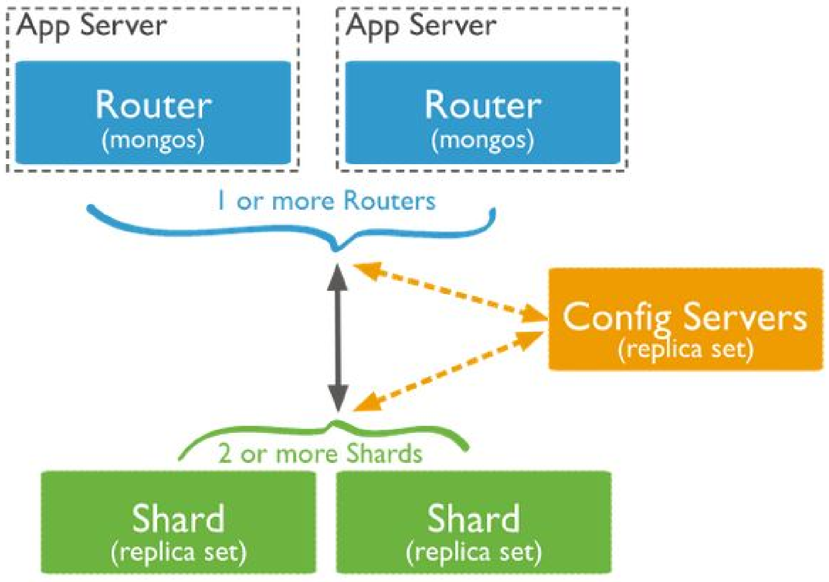


Figure 3.9: Components of MongoDB

- The application communicates with the routers (mongos) about the query to be executed.
- The mongos instance consults the config servers to check which shard contains the required data set to send the query to that shard.
- Finally, the result of the query will be returned to the application.

It's important to remember that the config servers also work as replica sets.

Shard Keys

When sharding a MongoDB collection, a shard key gets created as one of the initial steps. The “shard key” is used to distribute the MongoDB collection’s documents across all the shards. The key consists of a single field or multiple fields in every document. The sharded key is immutable and cannot be changed after sharding. A sharded collection only contains a single shard key.

When sharding a populated collection, the collection must have an index that starts with the shard key. For empty collections that don’t have an appropriate index, MongoDB will create an index for the specified shard key.

The shard key can directly have an impact on the performance of the cluster. Hence can lead to bottlenecks in applications associated with the cluster. To mitigate this, before sharding the collection, the shard key must be created based on:

- The schema of the data set
- How the data set is queried

Chunks

Chunks are subsets of shared data. MongoDB separates sharded data into chunks that are distributed across the shards in the shared cluster. Each chunk has an inclusive lower and exclusive upper range based on the shard key. A balancer specific for each cluster handles the chunk distribution.

The balancer runs as a background job and distributes the chunks as needed to achieve an even balance of chunks across all shards. This process is called even chuck distribution.

3.6. Data Modeling in HBase

HBase is a column-oriented database that’s an open-source implementation of Google’s Big Table storage architecture. It can manage structured and semi-structured data and has some built-in features such as scalability, versioning, compression and garbage collection.

Since its uses write-ahead logging and distributed configuration, it can provide fault-tolerance and quick recovery from individual server failures. HBase built on top of Hadoop / HDFS and the data stored in HBase can be manipulated using Hadoop’s MapReduce capabilities.

HBase Data Model

The Data Model in HBase is designed to accommodate semi-structured data that could vary in field size, data type and columns. Additionally, the layout of the data model makes it easier to partition the data and distribute it across the cluster. The Data Model in HBase is made of different logical components such as Tables, Rows, Column Families, Columns, Cells and Versions.

Tables – The HBase Tables are more like logical collection of rows stored in separate partitions called Regions. As shown above, every Region is then served by exactly one Region Server. The figure above shows a representation of a Table.

Rows – A row is one instance of data in a table and is identified by a rowkey. Rowkeys are unique in a Table and are always treated as a byte[].

Column Families – Data in a row are grouped together as Column Families. Each Column Family has one more Columns and these Columns in a family are stored together in a low level storage file known as HFile. Column Families form the basic unit of physical storage to which certain HBase features like compression are applied.

Columns – A Column Family is made of one or more columns. A Column is identified by a Column Qualifier that consists of the Column Family name concatenated with the Column name using a colon – example: column family:columnname. There can be multiple Columns within a Column Family and Rows within a table can have varied number of Columns.

Cell – A Cell stores data and is essentially a unique combination of rowkey, Column Family and the Column (Column Qualifier). The data stored in a Cell is called its value and the data type is always treated as byte[].

Version – The data stored in a cell is versioned and versions of data are identified by the timestamp. The number of versions of data retained in a column family is configurable and this value by default is 3.

3.6.1 Defining Schema

Schema design in NoSQL is very different from schema design in a RDBMS. Once you get something like HBase up and running, you may find yourself staring blankly at a shell, lost in the possibilities of creating your first table. HBase schemas can be created or updated using the Apache HBase Shell or by using Admin in the Java API. Tables must be disabled when making ColumnFamily modifications.

Example:

```
Configuration config = HBaseConfiguration.create();
Admin admin = new Admin(conf);
TableName table = TableName.valueOf(“myTable”);
admin.disableTable(table);
HColumnDescriptor cf1 = …;
admin.addColumn(table, cf1); // adding new ColumnFamily
HColumnDescriptor cf2 = …;
admin.modifyColumn(table, cf2);// modifying existing ColumnFamily
admin.enableTable(table);
```

Online schema changes are supported in the 0.92.x codebase, but the 0.90.x codebase requires the table to be disabled. When changes are made to either Tables or ColumnFamilies (e.g. region size, block size), these changes take effect the next time there is a major compaction and the StoreFiles get re-written.

Table Schema – There are many different data sets, with different access patterns and service-level expectations. Therefore, these rules of thumb are only an overview. Read the rest of this chapter to get more details after you have gone through this list.

- Aim to have regions sized between 10 and 50 GB.
- Aim to have cells no larger than 10 MB, or 50 MB if you use [mob]. Otherwise, consider storing your cell data in HDFS and store a pointer to the data in HBase.
- A typical schema has between 1 and 3 column families per table. HBase tables should not be designed to mimic RDBMS tables.
- Around 50-100 regions is a good number for a table with 1 or 2 column families. Remember that a region is a contiguous segment of a column family.
- Keep your column family names as short as possible. The column family names are stored for every value (ignoring prefix encoding). They should not be self-documenting and descriptive like in a typical RDBMS.
- If you are storing time-based machine data or logging information, and the row key is based on device ID or service ID plus time, you can end up with a pattern where older data regions never have additional writes beyond a certain age. In this type of situation, you end up with a small number of active regions and a large number of older regions which have no new writes. For these situations, you can tolerate a larger number of regions because your resource consumption is driven by the active regions only.
- If only one column family is busy with writes, only that column family accomulates memory. Be aware of write patterns when allocating resources.

5.6.2. CRUD Operations

In order to perform CRUD operations on HBase tables, we use Client API for HBase.

What is HBase Client API?

Basically, to perform CRUD operations on HBase tables we use Java client API for HBase. Since HBase has a Java Native API and it is written in Java thus it offers programmatic access to DML (Data Manipulation Language).

i. Class HBase Configuration

This class adds HBase configuration files to a Configuration. It belongs to the org.apache.hadoop.hbase package.

ii. Method

static org.apache.hadoop.conf.Configuration create()

Hbase CRUD Operations are

- Create
- Read
- Update
- Delete

Create

Let’s create an HBase table and insert data into the table. Now that we know, while creating a table user needs to create required Column Families.

Read

‘get’ and ‘scan’ command is used to read data from HBase.

Update

To update any record HBase uses ‘put’ command. To update any column value, users need to put new values and HBase will automatically update the new record with the latest timestamp.

Delete

‘delete‘ command is used to delete individual cells of a record.

SUMMARY

- Bayesian Statistics follows a unique principle wherein it helps determine the joint probability distribution for observed and unobserved parameters using a statistical model
- Kernel Function is a method used to take data as input and transform it into the required form of processing data. “Kernel” is used due to a set of mathematical functions used in Support Vector Machine providing the window to manipulate the data.
- A neuro-fuzzy system is a fuzzy system that uses a learning algorithm derived from or inspired by neural network theory to determine its parameters (fuzzy sets and fuzzy rules) by processing data samples.
- Principal Components Analysis, also known as PCA, is a technique commonly used for reducing the dimensionality of data while preserving as much as possible of the

information contained in the original data. PCA achieves this goal by projecting data onto a lower-dimensional subspace that retains most of the variance among the data points.

- NoSQL Database is a non-relational Data Management System, which does not require a fixed schema. It avoids joins, and is easy to scale. The major purpose of using a NoSQL database is for distributed data stores with humongous data storage needs. NoSQL is used for Big data and real-time web apps.
- Chunks are subsets of shared data. MongoDB separates sharded data into chunks that are distributed across the shards in the shared cluster. Each chunk has an inclusive lower and exclusive upper range based on the shard key. A balancer specific for each cluster handles the chunk distribution. The balancer runs as a background job and distributes the chunks as needed to achieve an even balance of chunks across all shards. This process is called even chuck distribution.
- HBase is a data model that is similar to Google's big table designed to provide quick random access to huge amounts of structured data. It leverages the fault tolerance provided by the Hadoop File System (HDFS).

SELF-ASSESSMENT QUESTIONS

- Discuss Maximum Likelihood Estimation and Maximum A Posteriori with an example.
- Discuss some type of the kernel function in SVM.
- Explain the need for Kernel Method and its Working process in SVM.
- What is Principal Components Analysis (PCA).
- How does Principal Components Analysis Work?
- Differentiate the difference between MongoDB and RDBMS.

FURTHER READINGS

- Bill Franks, "Taming the Big Data Tidal Wave: Finding Opportunities in Huge Data Streams with Advanced Analytics", John Wiley & sons, First Edition,2013.
- Rachel Schutt, Cathy O'Neil, "Doing Data Science", O'Reilly, First Edition,2013
- Foster Provost, Tom Fawcet, "Data Science for Business", O'Reilly, First Edition,2013.
- Bart Baesens, "Analytics in a Big Data World: The Essential Guide to Data Science and its Applications", Wiley, First Edition,2014.

3. Data Modelling

Data modelling is a very common terminology in software engineering and other IT disciplines. It has many interpretations and definitions depending on the field in discussion. In data science, data modelling is the process of finding the function by which data was generated. In this context, data modelling is the goal of any data analysis task. For instance if you have a 2d dataset, and finding the 2 variables are linearly correlated, This could be model using linear regression or if we visualize the data and find out it's non-linearly, it may modeled using nonlinear regression.

3.1 Bayesian Modeling

The Bayesian technique is an approach in statistics used in data analysis and parameter estimation. This approach is based on the Bayes theorem.

Bayesian Statistics follows a unique principle wherein it helps determine the joint probability distribution for observed and unobserved parameters using a statistical model. The knowledge of statistics is essential to tackle analytical problems in this scenario.

Ever since the introduction of the Bayes theorem in the 1770s by Thomas Bayes, it has remained an indispensable tool in statistics. Bayesian models are a classic replacement for frequentist models as recent innovations in statistics have helped breach milestones in a wide range of industries, including medical research, understanding web searches, and processing natural languages (Natural Language Processing).

Example: Alzheimer's is a disease known to pose a progressive risk as a person ages. However, with the help of the Bayes theorem, doctors can estimate the probability of a person having Alzheimer's in the future. It also applies to cancer and other age-related illnesses that a person becomes vulnerable to in the later years of his life.

3.1.1 Frequent Statistics Vs Bayesian Statistics

Frequent Statistics vs Bayesian statistics has consistently been a topic of controversy and nightmares for beginners, both of whom have difficulty choosing between the two. In the early 20th century, Bayesian statistics underwent its share of distrust and acceptance issues. With time, however, people realized the applicability of Bayesian models and the accurate solutions it yields.

3.1.2 Frequent Statistics

It is a widely used inferential methodology in the world of statistics. It analyzes whether or not an event (mentioned as a hypothesis) has taken place. It also estimates the probability of the event occurring during the span of the experiment. The experiment is repeated until the desired outcome is achieved.

Their distribution samples are of actual size, and the experiment is repeated infinite times theoretically.

Example: Showing how frequent statistics can be used to study the tossing of a coin.

- The possibility of getting a head on tossing the coin once is 0.5 (1/2).
- The number of heads denotes the actual number of leads obtained.
- The difference between the actual number of heads and the expected number of heads will increase as the number of tosses increases.

So here, the result depends on the number of times the experiment is repeated. It is a major drawback of frequent statistics.

3.1.3 Birth of Bayesian Statistics

Reverend Thomas Bayes first proposed the Bayesian approach to statistics in his essay written in 1763. This approach was published by Richard Price as a strategy in inverse probability to forecast future events based on the past.

Bayesian data modeling is to model your data using Bayes Theorem. Let us re-visit Bayes Rule again:

$$P(H|E) = \frac{P(E|H) * P(H)}{P(E)}$$

In the above equation, H is the hypothesis and E is the evidence. In the real world however, we understand Bayesian components differently! The evidence is usually expressed by data, and the hypothesis reflects the expert's prior estimation of the posterior. Therefore, we can re-write the Bayes Rule to be:

$$P(posterior) = \frac{P(data|\theta) * P(prior)}{P(data)}$$

In the above definition we learned about prior, posterior, and data, but what about θ parameter? θ is the set of coefficients that best define the data. You may think of θ as the set of slope and intercept of your linear regression equation, or the vector of coefficients ω in your polynomial regression function. As you see in the above equation, θ is the single missing parameter, and the goal of Bayesian modeling is to find it.

3.1.4 Bayesian Modeling & Probability Distributions

Bayes Rule is a probabilistic equation, where each term in it is expressed as a probability. Therefore, modeling the prior and the likelihood must be achieved using probabilistic functions. In this context arise probability distributions as a concrete tool in Bayesian modelling, as they provide a great variety of probabilistic functions that suits numerous styles of discrete and continuous variables.

In order to select the suitable distribution for your data, you should learn about the data domain and gather information from previous studies in it. You may also ask an expert to learn how data is developed over time. If you have big portions of data, you may visualize it and try to detect certain pattern(s) of its evolving over time, and select your probability distribution upon.

3.1.5 Maximum Likelihood Estimation (MLE)

Maximum Likelihood Estimation (MLE) is the simplest way of Bayesian data modelling, in which we ignore both prior and marginal probabilities (i.e. consider both quantities equal to 1). The formula of MLE is:

$$P(\text{posterior}) \propto P(\text{data}|\theta)$$

The steps of MLE are as follows:

- Select a probability distribution that best describes your data
- Estimate random value(s) of θ
- Tune θ value(s) and measure the corresponding likelihood
- Select θ that correspond to the maximum likelihood

Example:

“A company captures the blood glucose (BG) levels of diabetics, analyze these levels, and send its clients suitable diet preferences. After one week of inserting her BG levels, a client asked the company smart app whether she can consume desserts after her lunch or not? By checking her after-meal BG levels, it was {172,171,166,175,170,165,160}. Knowing that the client after-meal BG should not exceed 200 mg/dl, what should the app recommend to the client?”

The goal here is to estimate an average BG level for the client to use it in the company recommendation system. Having the above BG sample data, we assume these readings are drawn from a normal distribution, with a mean of 168.43 and standard deviation of 4.69. See figure 3.1

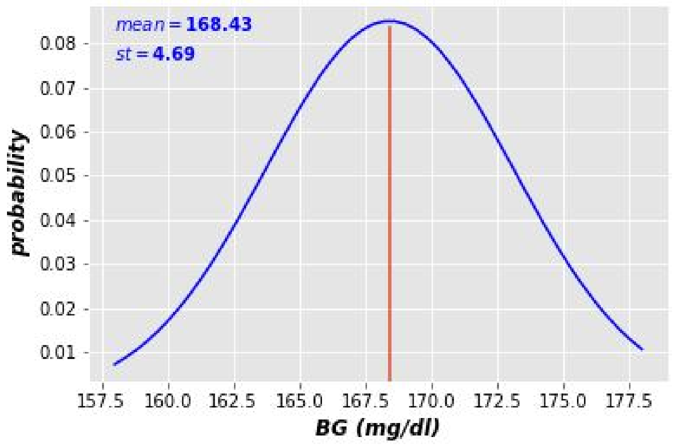


Figure 3.1: Example for MLE

Now we calculate the likelihood of this estimation as follows:

P(data|θ)=P({172,171,166,175,170,165,160}|168.43)=P(172|168.43)×P(171|168.43)×P(166|168.43)×P(175|168.43)×P(170|168.43)×P(165|168.43)×P(160|168.43)

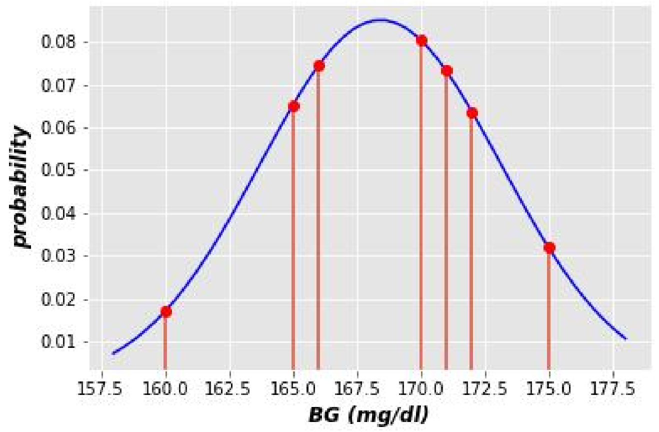


Figure 3.2: Example for MLE

Now we consider the multiplying small probability values (red dots in figure 3.2) generates very small value (very close to 0). Therefore, we replace multiplication with summation. In the above case, the sum of these probabilities equal 0.406:

P(data|θ)=P({172,171,166,175,170,165,160}|168.43)=P(172|168.43)+P(171|168.43)+P(166|168.43)+P(175|168.43)+P(170|168.43)+P(165|168.43)+P(160|168.43)=0.406

In order to find the maximum likelihood, we need to estimate different values of BG levels and calculate the corresponding likelihoods. The BG value with maximum likelihood is the most suitable estimation of the BG.

To automate this process, we generate 1000 random numbers in the same range of captured BG levels, and measure the corresponded likelihoods. The results are illustrated in the following plot figure 3.3.

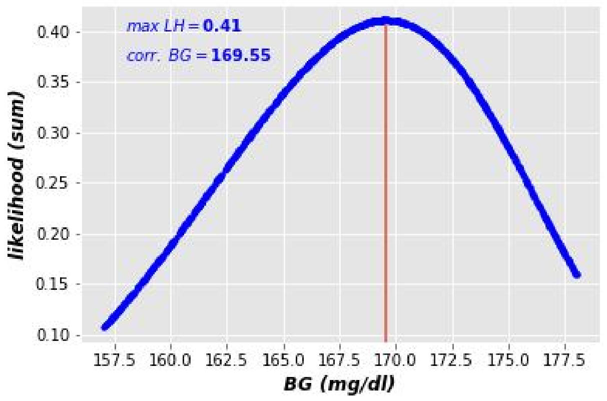


Figure 3.3: Example for MLE

As you can see, the maximum likelihood estimate of randomly generated BGs equals 169.55, which is very close to the average of captured BG levels (168.43). The difference between both values is due to the small size of captured data. The larger sample size you have, the smaller difference between both estimates you get.

Based on this estimate, the app can recommend it’s patient to consume a small piece of dessert after at least 3 hours of her lunch, with taking the suitable insulin dose.

3.1.6 Maximum A Posteriori (MAP)

Maximum A Posteriori (MAP) is the second approach of Bayesian modelling, where the posterior is calculated using both likelihood and prior probabilities.

$$P(\text{posterior}) \propto P(\text{data}|\theta) \times P(\text{Prior})$$

Consider MAP modeling as a generalization of MLE, where the likelihood is approximated with prior information. Technically speaking, we deal with our data samples as if it were generated by the prior distribution:

$$P(\text{posterior}) \propto P(\text{data}|\theta) \times P(\theta)$$

Example: ”Suppose that the client updated the app with her historical BG levels for the last month, which turned out to be 171 mg/dl on average, with standard deviation of 3, how the app can use this new information to update the patient’s BG level? “

In this case, we have two distributions of Blood Glucose, one of recent data (the likelihood), and the other of historical data (the prior). Each data source can be expressed as normal distribution (see the figure 3.4).

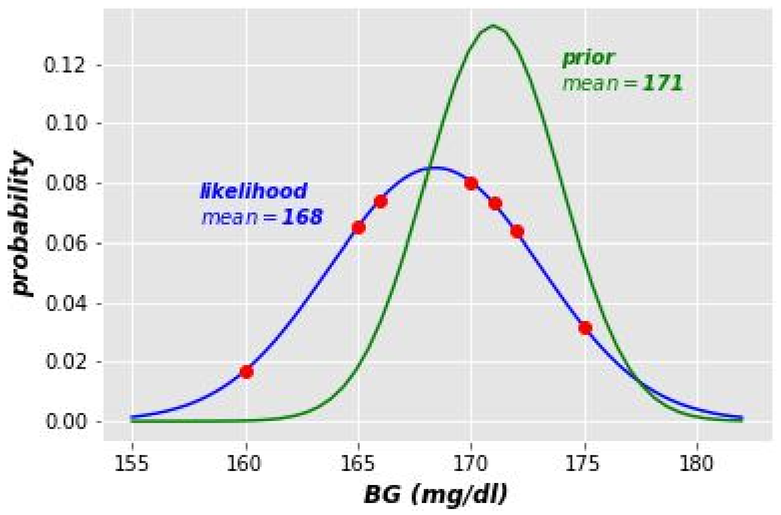


Figure 3.4: Example for MAP

The posterior here is updated by multiplying the prior marginal probability by each term of data probabilities:

$$P(data|\theta)=P(\{172,171,166,175,170,165,160\}|\theta)\times P(\theta)=P(172|\theta)\times P(\theta)\times P(171|\theta)\times P(\theta)\times P(166|\theta)\times P(\theta)\times P(175|\theta)\times P(\theta)\times P(170|\theta)\times P(\theta)\times P(165|\theta)\times P(\theta)\times P(160|\theta)\times P(\theta)$$

The prior marginal probability P(θ) is the summation of all data probabilities over the prior distribution:

$$P(\theta)=P(172|\theta)+P(171|\theta)+P(166|\theta)+P(175|\theta)+P(170|\theta)+P(165|\theta)+P(160|\theta)$$

As our goal is to maximize posterior estimation, we generate random values of θ and measure the corresponding estimations. By generating 500 guesses of θ, we obtain the following posterior distribution (in black). For visualization reasons, I raised the value of posterior probabilities to the power of 0.08.

Now we get more generalized estimation of the client's BG level. The MAP technique excludes two measures in the posterior distribution (the red data points outside the black curve), and generates more convenient estimate. As the below plot shows, the Standard Error (SE) of the posterior is less than that of the likelihood. Standard error is an indication of the reliability of a standard expectation (i.e. mean of predicted normal distribution). It is calculated as SE = $\sigma/\sqrt{N}$, where σ is the standard deviation and N is data size.

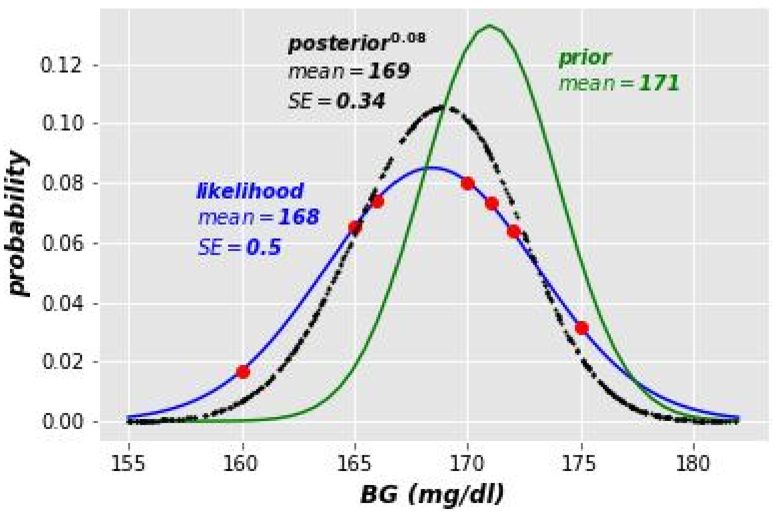


Figure 3.5: Example for MAP

The new MAP posterior estimation of the patient's BG level is higher than the one estimated using MLE show in figure 3.5. This may lead the app to prohibit the patient of consuming desserts after her lunch, until her BG levels become more stabilized.

3.1.7 Advantages of Bayesian Statistics

- It provides an organic and simple means to blend pre-conceived information with a solid framework with scientific evidence. The past information about a parameter can be used to form a prior distribution for future investigation. The inferences adhere to the Bayes theorem.
- The inferences from a Bayesian model are logical and mathematically accurate and not crude assumptions. The accuracy remains constant irrespective of the size of the sample.
- Bayesian statistics follow the likelihood principle. When two different samples have a common likelihood function for a belief θ, all inferences about the belief should be similar. Classical statistical techniques do not follow the likelihood principle.
- The solutions from a Bayesian analysis can be easily interpreted.
- It offers a conducive platform for various models like hierarchical models and incomplete data issues. The computations of all parametric models can be virtually tracked with the help of other numerical techniques.

3.2 Support Vector and Kernel Methods

Kernels or kernel methods (also called Kernel functions) are sets of different types of algorithms that are being used for pattern analysis. They are used to solve a non-linear problem by using a linear classifier. Kernels Methods are employed in SVM (Support Vector Machines) which are used in classification and regression problems. The SVM uses what is called a "Kernel Trick" where the data is transformed and an optimal boundary is found for the possible outputs.

3.2.1 The Need for Kernel Method and its Working

Before we get into the working of the Kernel Methods, it is more important to understand support vector machines or the SVMs because kernels are implemented in SVM models. So, Support Vector Machines are supervised machine learning algorithms that are used in classification and regression problems such as classifying an apple to class fruit while classifying a Lion to the class animal.

To demonstrate, below is what support vector machines look like:

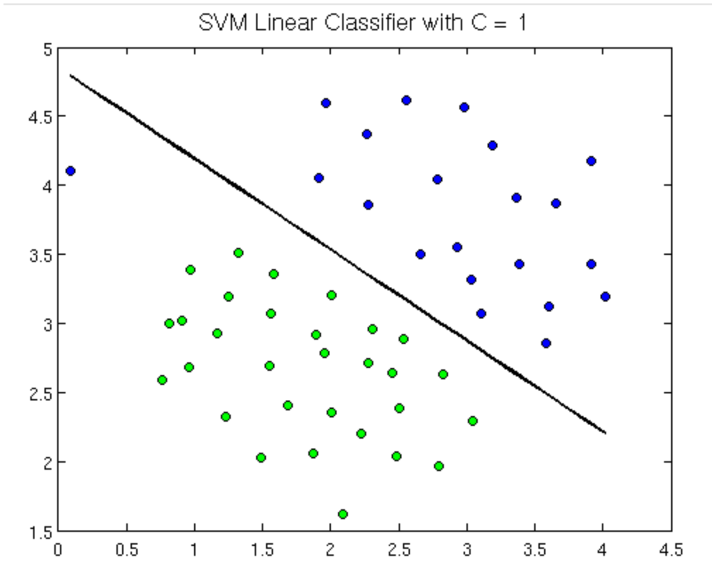


Figure 3.6: Support vector machines

Here we can see a hyper plane which is separating green dots from the blue ones. A hyper plane is one dimension less than the ambient plane. E.g. in the figure 3.6, we have 2 dimension which represents the ambient space but the lone which divides or classifies the space is one dimension less than the ambient space and is called hyper plane.

But what if we have input like this:

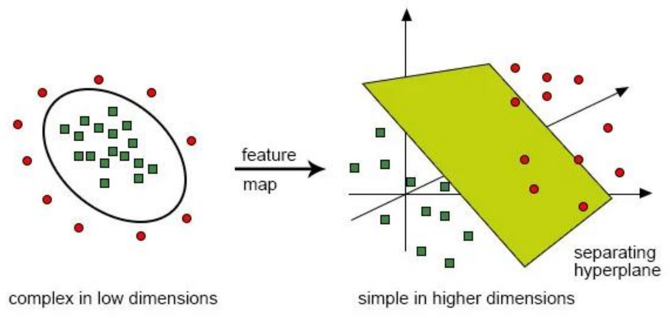


Figure 3.7: Separation in higher dimensions

It is very difficult to solve this classification using a linear classifier as there is no good linear line that should be able to classify the red and the green dots as the points are randomly distributed. Here comes the use of kernel function which takes the points to higher dimensions, solves the problem over there and returns the output. Think of this in this way, we can see that the green dots are enclosed in some perimeter area while the red one lies outside it, likewise, there could be other scenarios where green dots might be distributed in a trapezoid-shaped area.

So what we do is to convert the two-dimensional plane which was first classified by one-dimensional hyper plane ("or a straight line") to the three-dimensional area and here our classifier i.e. hyper plane will not be a straight line but a two-dimensional plane which will cut the area.

In order to get a mathematical understanding of kernel, let us understand the Lili Jiang's equation of kernel which is:

K(x, y)=<f(x), f(y)> where,
K is the kernel function,
X and Y are the dimensional inputs,
f is the map from n-dimensional to m-dimensional space and,
< x, y > is the dot product.

Example:
Let us say that we have two points, x= (2, 3, 4) and y= (3, 4, 5)
As we have seen, K(x, y) = < f(x), f(y) >.
Let us first calculate < f(x), f(y) >
f(x)=(x1x1, x1x2, x1x3, x2x1, x2x2, x2x3, x3x1, x3x2, x3x3)
f(y)=(y1y1, y1y2, y1y3, y2y1, y2y2, y2y3, y3y1, y3y2, y3y3)

so,
f(2, 3, 4)=(4, 6, 8, 6, 9, 12, 8, 12, 16)and
f(3 ,4, 5)=(9, 12, 15, 12, 16, 20, 15, 20, 25)

so the dot product,
f (x). f (y) = f(2,3,4) . f(3,4,5)=
(36 + 72 + 120 + 72 +144 + 240 + 120 + 240 + 400) = 1444
And,
K(x, y) = (2*3 + 3*4 + 4*5) ^2=(6 + 12 + 20)^2=38*38 = 1444.

This as we find out, f(x).f(y) and K(x, y) give us the same result, but the former method required a lot of calculations(because of projecting 3 dimensions into 9 dimensions) while using the kernel, it was much easier.

3.2.2 Types of Kernel and methods in SVM

1. Liner Kernel

Let us say that we have two vectors with name x1 and Y1, then the linear kernel is defined by the dot product of these two vectors: K(x1, x2) = x1 . x2

2. Polynomial Kernel

A polynomial kernel is defined by the following equation:

K(x1, x2) = (x1 . x2 + 1)d,

Where, d is the degree of the polynomial and x1 and x2 are vectors

3. Gaussian Kernel

This kernel is an example of a radial basis function kernel. Below is the equation for this:

$$K(x_i, x_j) = exp(-\sigma \left|\left|x_i - x_j\right|\right|^2)$$

The given sigma plays a very important role in the performance of the Gaussian kernel and should neither be overestimated and nor be underestimated, it should be carefully tuned according to the problem.

4. Exponential Kernel

This is in close relation with the previous kernel i.e. the Gaussian kernel with the only difference is the square of the norm is removed.

The function of the exponential function is:

$$K(x, y) = \exp\,(-\frac{||x - y||}{2\sigma^2})$$

This is also a radial basis kernel function.

5. Laplacian Kernel

This type of kernel is less prone for changes and is totally equal to previously discussed exponential function kernel, the equation of Laplacian kernel is given as:

$$K(x,y) = \exp\left(-\frac{||x-y||}{\sigma}\right)$$

6. Hyperbolic or the Sigmoid Kernel

This kernel is used in neural network areas of machine learning. The activation function for the sigmoid kernel is the bipolar sigmoid function. The equation for the hyperbolic kernel function is:

$$K(x,y) = tanh(\alpha x^T y + c)$$

This kernel is very much used and popular among support vector machines.

7. Anova radial basis kernel

This kernel is known to perform very well in multidimensional regression problems just like the Gaussian and Laplacian kernels. This also comes under the category of radial basis kernel.

The equation for Anova kernel is:

$$K(x,y) = \Sigma_{k=1}^{n} \exp\left(-\sigma\left(x^k - y^k\right)^2\right)^d$$

There are a lot more types of Kernel Method and we have discussed the mostly used kernels. It purely depends on the type of problem which will decide the kernel function to be used.

3.3 Neuro–Fuzzy Modeling

A neuro-fuzzy system is a fuzzy system that uses a learning algorithm derived from or inspired by neural network theory to determine its parameters (fuzzy sets and fuzzy rules) by processing data samples.

3.3.1 Combining fuzzy systems with neural networks

Both neural networks and fuzzy systems have some things in common. They can be used for solving a problem (e.g. pattern recognition, regression or density estimation) if there does not exist any mathematical model of the given problem. They solely do have certain disadvantages and advantages which almost completely disappear by combining both concepts.

Neural networks can only come into play if the problem is expressed by a sufficient amount of observed examples. These observations are used to train the black box. On the one hand no prior knowledge about the problem needs to be given. On the other hand, however, it is not straightforward to extract comprehensible rules from the neural network's structure.

On the contrary, a fuzzy system demands linguistic rules instead of learning examples as prior knowledge. Furthermore the input and output variables have to be described linguistically. If the knowledge is incomplete, wrong or contradictory, then the fuzzy system must be tuned. Since there is not any formal approach for it, the tuning is performed in a heuristic way. This is usually very time consuming and error-prone.

3.3.2 Characteristics

Compared to a common neural network, connection weights and propagation and activation functions of fuzzy neural networks differ a lot. Although there are many different approaches to model a fuzzy neural network (Buckley and Hayashi, 1994, 1995; Nauck and Kruse, 1996), most of them agree on certain characteristics such as the following:

- A neuro-fuzzy system based on an underlying fuzzy system is trained by means of a data-driven learning method derived from neural network theory. This heuristic only takes into account local information to cause local changes in the fundamental fuzzy system.
- It can be represented as a set of fuzzy rules at any time of the learning process, i.e., before, during and after.
    - Thus the system might be initialized with or without prior knowledge in terms of fuzzy rules.
- The learning procedure is constrained to ensure the semantic properties of the underlying fuzzy system.
- A neuro-fuzzy system approximates a n-dimensional unknown function which is partly represented by training examples.
    - Fuzzy rules can thus be interpreted as vague prototypes of the training data.
- A neuro-fuzzy system is represented as special three-layer feedforward neural network.
    - The first layer corresponds to the input variables.
    - The second layer symbolizes the fuzzy rules.
    - The third layer represents the output variables.
    - The fuzzy sets are converted as (fuzzy) connection weights.
    - Some approaches also use five layers where the fuzzy sets are encoded in the units of the second and fourth layer, respectively. However, these models can be transformed into a three-layer architecture.

3.4 Principal Component Analysis

Principal Component Analysis is an unsupervised learning algorithm that is used for the dimensionality reduction in machine learning. It is a statistical process that converts the observations of correlated features into a set of linearly uncorrelated features with the help of orthogonal transformation. These new transformed features are called the Principal Components. It is one of the popular tools that is used for exploratory data analysis and predictive modeling. It is a technique to draw strong patterns from the given dataset by reducing the variances.

PCA generally tries to find the lower-dimensional surface to project the high-dimensional data.

PCA works by considering the variance of each attribute because the high attribute shows the good split between the classes, and hence it reduces the dimensionality. Some real-world applications of PCA are image processing, movie recommendation system, optimizing the power allocation in various communication channels. It is a feature extraction technique, so it contains the important variables and drops the least important variable.

The PCA algorithm is based on some mathematical concepts such as:

- Variance and Covariance
- Eigenvalues and Eigen factors

Some common terms used in PCA algorithm:

- **Dimensionality**: It is the number of features or variables present in the given dataset. More easily, it is the number of columns present in the dataset.
- **Correlation**: It signifies that how strongly two variables are related to each other. Such as if one changes, the other variable also gets changed. The correlation value ranges from -1 to +1. Here, -1 occurs if variables are inversely proportional to each other, and +1 indicates that variables are directly proportional to each other.
- **Orthogonal**: It defines that variables are not correlated to each other, and hence the correlation between the pair of variables is zero.
- **Eigenvectors**: If there is a square matrix M, and a non-zero vector v is given. Then v will be eigenvector if Av is the scalar multiple of v.
- **Covariance Matrix**: A matrix containing the covariance between the pair of variables is called the Covariance Matrix.

3.4.1 Principal Components in PCA

As described above, the transformed new features or the output of PCA are the Principal Components. The numbers of these PCs are either equal to or less than the original features present in the dataset. Some properties of these principal components are given below:

- The principal component must be the linear combination of the original features.
- These components are orthogonal, i.e., the correlation between a pair of variables is zero.
- The importance of each component decreases when going to 1 to n, it means the 1 PC has the most importance, and n PC will have the least importance.

3.4.2 Steps for PCA algorithm

1. **Getting the dataset**
   Firstly, we need to take the input dataset and divide it into two subparts X and Y, where X is the training set, and Y is the validation set.

2. **Representing data into a structure**
   Now we will represent our dataset into a structure. Such as we will represent the two-dimensional matrix of independent variable X. Here each row corresponds to the data items, and the column corresponds to the Features. The number of columns is the dimensions of the dataset.
3. **Standardizing the data**
   In this step, we will standardize our dataset. Such as in a particular column, the features with high variance are more important compared to the features with lower variance. If the importance of features is independent of the variance of the feature, then we will divide each data item in a column with the standard deviation of the column. Here we will name the matrix as Z.
4. **Calculating the Covariance of Z**
   To calculate the covariance of Z, we will take the matrix Z, and will transpose it. After transpose, we will multiply it by Z. The output matrix will be the Covariance matrix of Z.
5. **Calculating the Eigen Values and Eigen Vectors**
   Now we need to calculate the eigenvalues and eigenvectors for the resultant covariance matrix Z. Eigenvectors or the covariance matrix are the directions of the axes with high information. And the coefficients of these eigenvectors are defined as the eigenvalues.
6. **Sorting the Eigen Vectors**
   In this step, we will take all the eigenvalues and will sort them in decreasing order, which means from largest to smallest. And simultaneously sort the eigenvectors accordingly in matrix P of eigenvalues. The resultant matrix will be named as P*.
7. **Calculating the new features Or Principal Components**
   Here we will calculate the new features. To do this, we will multiply the P* matrix to the Z. In the resultant matrix Z*, each observation is the linear combination of original features. Each column of the Z* matrix is independent of each other.
8. **Remove less or unimportant features from the new dataset.**
   The new feature set has occurred, so we will decide here what to keep and what to remove. It means, we will only keep the relevant or important features in the new dataset, and unimportant features will be removed out.

3.4.3 Applications of Principal Component Analysis

- PCA is mainly used as the dimensionality reduction technique in various AI applications such as computer vision, image compression, etc.
- It can also be used for finding hidden patterns if data has high dimensions. Some fields where PCA is used are Finance, data mining, Psychology, etc.

3.5 Introduction to NoSQL

NoSQL Database is a non-relational Data Management System, that does not require a fixed schema. It avoids joins, and is easy to scale. The major purpose of using a NoSQL database is for distributed data stores with humongous data storage needs. NoSQL is used for Big data and real-time web apps. For example, companies like Twitter, Facebook and Google collect terabytes of user data every single day.

3.5.1 Why NoSQL?

The concept of NoSQL databases became popular with Internet giants like Google, Facebook, Amazon, etc. who deal with huge volumes of data. The system response time becomes slow when you use RDBMS for massive volumes of data.

To resolve this problem, we could "scale up" our systems by upgrading our existing hardware. This process is expensive. The alternative for this issue is to distribute database load on multiple hosts whenever the load increases. This method is known as "scaling out."

3.5.2 Brief History of NoSQL Databases

- 1998- Carlo Strozzi use the term NoSQL for his lightweight, open-source relational database
- 2000- Graph database Neo4j is launched
- 2004- Google BigTable is launched
- 2005- CouchDB is launched
- 2007- The research paper on Amazon Dynamo is released
- 2008- Facebooks open sources the Cassandra project
- 2009- The term NoSQL was reintroduced

3.5.3 Features of NoSQL

Non-relational

- NoSQL databases never follow the relational model
- Never provide tables with flat fixed-column records
- Work with self-contained aggregates or BLOBs
- Doesn't require object-relational mapping and data normalization
- No complex features like query languages, query planners, referential integrity joins, ACID

Schema-free

- NoSQL databases are either schema-free or have relaxed schemas

- Do not require any sort of definition of the schema of the data
- Offers heterogeneous structures of data in the same domain

Simple API

- Offers easy to use interfaces for storage and querying data provided
- APIs allow low-level data manipulation & selection methods
- Text-based protocols mostly used with HTTP REST with JSON
- Mostly used no standard based NoSQL query language
- Web-enabled databases running as internet-facing services

Distributed

- Multiple NoSQL databases can be executed in a distributed fashion
- Offers auto-scaling and fail-over capabilities
- Often ACID concept can be sacrificed for scalability and throughput
- Mostly no synchronous replication between distributed nodes Asynchronous Multi-Master Replication, peer-to-peer, HDFS Replication
- Only providing eventual consistency
- Shared Nothing Architecture. This enables less coordination and higher distribution.

3.5.4 Types of NoSQL Databases

NoSQL Databases are mainly categorized into four types: Key-value pair, Column-oriented, Graph-based and Document-oriented. Every category has its unique attributes and limitations. None of the above-specified database is better to solve all the problems. Users should select the database based on their product needs.

Key Value Pair Based

Data is stored in key/value pairs. It is designed in such a way to handle lots of data and heavy load. Key-value pair storage databases store data as a hash table where each key is unique, and the value can be a JSON, BLOB(Binary Large Objects), string, etc.

It is one of the most basic NoSQL database example. This kind of NoSQL database is used as a collection, dictionaries, associative arrays, etc. Key value stores help the developer to store schema-less data. They work best for shopping cart contents.

Column-based

Column-oriented databases work on columns and are based on BigTable paper by Google. Every column is treated separately. Values of single column databases are stored contiguously.

They deliver high performance on aggregation queries like SUM, COUNT, AVG, MIN etc. as the data is readily available in a column.

Column-based NoSQL databases are widely used to manage data warehouses, business intelligence, CRM, Library card catalogs,

HBase, Cassandra, HBase, Hypertable are NoSQL query examples of column based database.

Document-Oriented:

Document-Oriented NoSQL DB stores and retrieves data as a key value pair but the value part is stored as a document. The document is stored in JSON or XML formats. The value is understood by the DB and can be queried.

The document type is mostly used for CMS systems, blogging platforms, real-time analytics & e-commerce applications. It should not use for complex transactions which require multiple operations or queries against varying aggregate structures.

Amazon SimpleDB, CouchDB, MongoDB, Riak, Lotus Notes, MongoDB, are popular Document originated DBMS systems.

Graph-Based

A graph type database stores entities as well the relations amongst those entities. The entity is stored as a node with the relationship as edges. An edge gives a relationship between nodes. Every node and edge has a unique identifier.

Compared to a relational database where tables are loosely connected, a Graph database is a multi-relational in nature. Traversing relationship is fast as they are already captured into the DB, and there is no need to calculate them.

Graph base database mostly used for social networks, logistics, spatial data.

Neo4J, Infinite Graph, OrientDB, FlockDB are some popular graph-based databases.

3.5.5. CAP Theorem

CAP theorem is also called brewer’s theorem. It states that is impossible for a distributed data store to offer more than two out of three guarantees

- Consistency
- Availability
- Partition Tolerance

Consistency:

The data should remain consistent even after the execution of an operation. This means once data is written, any future read request should contain that data. For example, after updating the order status, all the clients should be able to see the same data.

Availability:

The database should always be available and responsive. It should not have any downtime.

Partition Tolerance:

Partition Tolerance means that the system should continue to function even if the communication among the servers is not stable. For example, the servers can be partitioned into multiple groups which may not communicate with each other. Here, if part of the database is unavailable, other parts are always unaffected.

3.5.6. MongoDB

MongoDB is a No SQL database. It is an open-source, cross-platform, document-oriented database written in C++.

Our MongoDB tutorial includes all topics of MongoDB database such as insert documents, update documents, delete documents, query documents, projection, sort() and limit() methods, create a collection, drop collection, etc. There are also given MongoDB interview questions to help you better understand the MongoDB database.

MongoDB is an open-source document database that provides high performance, high availability, and automatic scaling. In simple words, you can say that - Mongo DB is a document-oriented database. It is an open source product, developed and supported by a company named 10gen.

MongoDB is available under General Public license for free, and it is also available under Commercial license from the manufacturer.

History of MongoDB

The initial development of MongoDB began in 2007 when the company was building a platform as a service similar to window azure.

MongoDB was developed by a NewYork based organization named 10gen which is now known as MongoDB Inc. It was initially developed as a PAAS (Platform as a Service). Later in 2009, it is introduced in the market as an open source database server that was maintained and supported by MongoDB Inc.

The first ready production of MongoDB has been considered from version 1.4 which was released in March 2010.
MongoDB2.4.9 was the latest and stable version which was released on January 10, 2014.

Purpose of building MongoDB

It may be a very genuine question that - "what was the need of MongoDB although there were many databases in action?"

There is a simple answer: All the modern applications require big data, fast features development, flexible deployment, and the older database systems not competent enough, so the MongoDB was needed.

The primary purpose of building MongoDB is:

- Scalability
- Performance
- High Availability
- Scaling from single server deployments to large, complex multi-site architectures.
- Key points of MongoDB
- Develop Faster
- Deploy Easier
- Scale Bigger

3.5.7. RDBMS Vs MongoDB

Sr. No.	Key	RDBMS	MongoDB

1	Concept	RDBMS is a relational database management system and works on relational database.	MongoDB is a non-relational, document oriented database management system and works on document based database
2	Hiearchical	Difficult to store hiearchical data.	Have inbuilt support to store hiearchical data
3	Scalablity	RDBMS is vertically scalable. Performance increases with increase of RAM.	MongoDB is horizontally scalable as well. Its performance increases with addition of processor.
4	Schema	Schema need to be defined in RDBMS before using a database.	Schema can be dynamically created and accessed in MongoDB.
5	SQL Injection	Vulnerable to SQL Injection attack.	SQL injection is not possible.
6	Principle	Follows ACID principle, Atomicity, Consistency, Isolation, and Durability.	Follows CAP theorem, Consistency, Availability, and Partition tolerance.
7	Basis	Database uses Row.	Database uses Document.
8	Basis	Database uses Column.	Database uses Field.
9	Performance	RDBMS is slower in processing large hierachical data.	MongoDB is blazingly fast in processing large hierachical data.
10	Joins	RDBMS supports complex joins.	MongoDB has no support for complex joins.
11	JavaScript Client	RDBMS do not provide JavaScript based client to query database.	MongoDB provides Javascript based client to query database.
12	Query Language	RDBMS uses SQL to query database.	MongoDB uses BSON to query database.

3.5.8. Mongo DB Database Model

MongoDB provides two types of data models: Embedded data model and Normalized data model. Based on the requirement, you can use either of the models while preparing your document.

Embedded Data Model

In this model, you can embed all the related data in a single document, it is also known as de-normalized data model.

Example: Assume we are getting the details of employees in three different documents namely, Personal_details, Contact and, Address, you can embed all the three documents in a single one as shown below

```
{
	_id: ,
	Emp_ID: "10025AE336"
	Personal_details:{
		First_Name: "Radhika",
		Last_Name: "Sharma",
		Date_Of_Birth: "1995-09-26"
	},
	Contact: {
		e-mail: "radhika_sharma.123@gmail.com",
		phone: "9848022338"
	},
	Address: {
		city: "Hyderabad",
		Area: "Madapur",
		State: "Telangana"
	}
}
```

Normalized Data Model

In this model, you can refer the sub documents in the original document, using references.

Employee:

```
{
	_id: <ObjectId101>,
	Emp_ID: "10025AE336"
}
```

Personal_details:

```
{
	_id: <ObjectId102>,
	empDocID: " ObjectId101",
	First_Name: "Radhika",
	Last_Name: "Sharma",
	Date_Of_Birth: "1995-09-26"
}
```

Contact:

```
{
	_id: <ObjectId103>,
        empDocID: " ObjectId101",
        e-mail: "radhika_sharma.123@gmail.com",
        phone: "9848022338"
}
```
Address:
```
{
        _id: <ObjectId104>,
        empDocID: " ObjectId101",
        city: "Hyderabad",
        Area: "Madapur",
        State: "Telangana"
}
```

Considerations while designing Schema in MongoDB

- Design your schema according to user requirements.
- Combine objects into one document if you will use them together. Otherwise separate them (but make sure there should not be need of joins).
- Duplicate the data (but limited) because disk space is cheap as compare to compute time.
- Do joins while write, not on read.
- Optimize your schema for most frequent use cases.
- Do complex aggregation in the schema.

3.5.9 Data Types and Sharding

Following is a list of usable data types in MongoDB.

Data Types	**Description**
String	String is the most commonly used datatype. It is used to store data. A string must be UTF 8 valid in mongodb.
Integer	Integer is used to store the numeric value. It can be 32 bit or 64 bit depending on the server you are using.
Boolean	This datatype is used to store boolean values. It just shows YES/NO values.
Double	Double datatype stores floating point values.
Min/Max Keys	This datatype compare a value against the lowest and highest bson elements.
Arrays	This datatype is used to store a list or multiple values into a single key.
Object	Object datatype is used for embedded documents.
Null	It is used to store null values.
Symbol	It is generally used for languages that use a specific type.
Date	This datatype stores the current date or time in unix time format. It makes you possible to specify your own date time by creating object of date and pass the

	value of date, month, year into it.

Sharding

Sharding is the process of distributing data across multiple hosts. In MongoDB, sharding is achieved by splitting large data sets into small data sets across multiple MongoDB instances.

How sharding works

When dealing with high throughput applications or very large databases, the underlying hardware becomes the main limitation. High query rates can stress the CPU, RAM, and I/O capacity of disk drives resulting in a poor end-user experience.

To mitigate this problem, there are two types of scaling methods.

Vertical scaling

Vertical scaling is the traditional way of increasing the hardware capabilities of a single server. The process involves upgrading the CPU, RAM, and storage capacity. However, upgrading a single server is often challenged by technological limitations and cost constraints.

Horizontal scaling

This method divides the dataset into multiple servers and distributes the database load among each server instance. Distributing the load reduces the strain on the required hardware resources and provides redundancy in case of a failure.

However, horizontal scaling increases the complexity of underlying architecture. MongoDB supports horizontal scaling through sharding—one of its major benefits, as we’ll see below.

MongoDB sharding basics

MongoDB sharding works by creating a cluster of MongoDB instances consisting of at least three servers. That means sharded clusters consist of three main components:

- The shard
- Mongos
- Config servers

Shard

A shard is a single MongoDB instance that holds a subset of the sharded data. Shards can be deployed as replica sets to increase availability and provide redundancy. The combination of multiple shards creates a complete data set. For example, a 2 TB data set can be broken down into four shards, each containing 500 GB of data from the original data set.

Mongos

Mongos act as the query router providing a stable interface between the application and the sharded cluster. This MongoDB instance is responsible for routing the client requests to the correct shard.

Config Servers

Configuration servers store the metadata and the configuration settings for the whole cluster.

Components illustrated

The following diagram from the official MongoDB docs explains the relationship between each component:

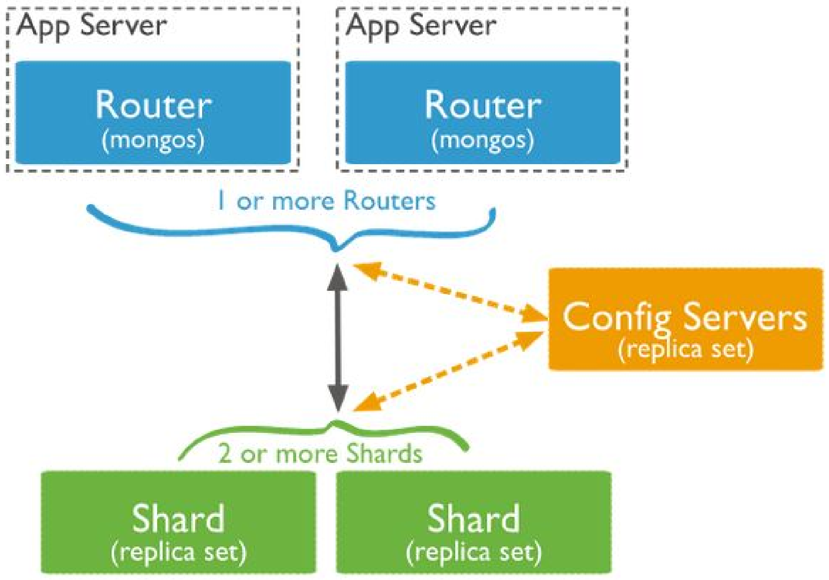


Figure 3.9: Components of MongoDB

- The application communicates with the routers (mongos) about the query to be executed.
- The mongos instance consults the config servers to check which shard contains the required data set to send the query to that shard.
- Finally, the result of the query will be returned to the application.

It's important to remember that the config servers also work as replica sets.

Shard Keys

When sharding a MongoDB collection, a shard key gets created as one of the initial steps. The “shard key” is used to distribute the MongoDB collection’s documents across all the shards. The key consists of a single field or multiple fields in every document. The sharded key is immutable and cannot be changed after sharding. A sharded collection only contains a single shard key.

When sharding a populated collection, the collection must have an index that starts with the shard key. For empty collections that don’t have an appropriate index, MongoDB will create an index for the specified shard key.

The shard key can directly have an impact on the performance of the cluster. Hence can lead to bottlenecks in applications associated with the cluster. To mitigate this, before sharding the collection, the shard key must be created based on:

- The schema of the data set
- How the data set is queried

Chunks

Chunks are subsets of shared data. MongoDB separates sharded data into chunks that are distributed across the shards in the shared cluster. Each chunk has an inclusive lower and exclusive upper range based on the shard key. A balancer specific for each cluster handles the chunk distribution.

The balancer runs as a background job and distributes the chunks as needed to achieve an even balance of chunks across all shards. This process is called even chuck distribution.

3.6. Data Modeling in HBase

HBase is a column-oriented database that’s an open-source implementation of Google’s Big Table storage architecture. It can manage structured and semi-structured data and has some built-in features such as scalability, versioning, compression and garbage collection.

Since its uses write-ahead logging and distributed configuration, it can provide fault-tolerance and quick recovery from individual server failures. HBase built on top of Hadoop / HDFS and the data stored in HBase can be manipulated using Hadoop’s MapReduce capabilities.

HBase Data Model

The Data Model in HBase is designed to accommodate semi-structured data that could vary in field size, data type and columns. Additionally, the layout of the data model makes it easier to partition the data and distribute it across the cluster. The Data Model in HBase is made of different logical components such as Tables, Rows, Column Families, Columns, Cells and Versions.

Tables – The HBase Tables are more like logical collection of rows stored in separate partitions called Regions. As shown above, every Region is then served by exactly one Region Server. The figure above shows a representation of a Table.

Rows – A row is one instance of data in a table and is identified by a rowkey. Rowkeys are unique in a Table and are always treated as a byte[].

Column Families – Data in a row are grouped together as Column Families. Each Column Family has one more Columns and these Columns in a family are stored together in a low level storage file known as HFile. Column Families form the basic unit of physical storage to which certain HBase features like compression are applied.

Columns – A Column Family is made of one or more columns. A Column is identified by a Column Qualifier that consists of the Column Family name concatenated with the Column name using a colon – example: column family:columnname. There can be multiple Columns within a Column Family and Rows within a table can have varied number of Columns.

Cell – A Cell stores data and is essentially a unique combination of rowkey, Column Family and the Column (Column Qualifier). The data stored in a Cell is called its value and the data type is always treated as byte[].

Version – The data stored in a cell is versioned and versions of data are identified by the timestamp. The number of versions of data retained in a column family is configurable and this value by default is 3.

3.6.1 Defining Schema

Schema design in NoSQL is very different from schema design in a RDBMS. Once you get something like HBase up and running, you may find yourself staring blankly at a shell, lost in the possibilities of creating your first table. HBase schemas can be created or updated using the Apache HBase Shell or by using Admin in the Java API. Tables must be disabled when making ColumnFamily modifications.

Example:

```
Configuration config = HBaseConfiguration.create();
Admin admin = new Admin(conf);
TableName table = TableName.valueOf(“myTable”);
admin.disableTable(table);
HColumnDescriptor cf1 = …;
admin.addColumn(table, cf1); // adding new ColumnFamily
HColumnDescriptor cf2 = …;
admin.modifyColumn(table, cf2);// modifying existing ColumnFamily
admin.enableTable(table);
```

Online schema changes are supported in the 0.92.x codebase, but the 0.90.x codebase requires the table to be disabled. When changes are made to either Tables or ColumnFamilies (e.g. region size, block size), these changes take effect the next time there is a major compaction and the StoreFiles get re-written.

Table Schema – There are many different data sets, with different access patterns and service-level expectations. Therefore, these rules of thumb are only an overview. Read the rest of this chapter to get more details after you have gone through this list.

- Aim to have regions sized between 10 and 50 GB.
- Aim to have cells no larger than 10 MB, or 50 MB if you use [mob]. Otherwise, consider storing your cell data in HDFS and store a pointer to the data in HBase.
- A typical schema has between 1 and 3 column families per table. HBase tables should not be designed to mimic RDBMS tables.
- Around 50-100 regions is a good number for a table with 1 or 2 column families. Remember that a region is a contiguous segment of a column family.
- Keep your column family names as short as possible. The column family names are stored for every value (ignoring prefix encoding). They should not be self-documenting and descriptive like in a typical RDBMS.
- If you are storing time-based machine data or logging information, and the row key is based on device ID or service ID plus time, you can end up with a pattern where older data regions never have additional writes beyond a certain age. In this type of situation, you end up with a small number of active regions and a large number of older regions which have no new writes. For these situations, you can tolerate a larger number of regions because your resource consumption is driven by the active regions only.
- If only one column family is busy with writes, only that column family accomulates memory. Be aware of write patterns when allocating resources.

5.6.2. CRUD Operations

In order to perform CRUD operations on HBase tables, we use Client API for HBase.

What is HBase Client API?

Basically, to perform CRUD operations on HBase tables we use Java client API for HBase. Since HBase has a Java Native API and it is written in Java thus it offers programmatic access to DML (Data Manipulation Language).

i. Class HBase Configuration

This class adds HBase configuration files to a Configuration. It belongs to the org.apache.hadoop.hbase package.

ii. Method

static org.apache.hadoop.conf.Configuration create()

Hbase CRUD Operations are

- Create
- Read
- Update
- Delete

Create

Let's create an HBase table and insert data into the table. Now that we know, while creating a table user needs to create required Column Families.

Read

'get' and 'scan' command is used to read data from HBase.

Update

To update any record HBase uses 'put' command. To update any column value, users need to put new values and HBase will automatically update the new record with the latest timestamp.

Delete

'delete' command is used to delete individual cells of a record.

SUMMARY

- Bayesian Statistics follows a unique principle wherein it helps determine the joint probability distribution for observed and unobserved parameters using a statistical model
- Kernel Function is a method used to take data as input and transform it into the required form of processing data. "Kernel" is used due to a set of mathematical functions used in Support Vector Machine providing the window to manipulate the data.
- A neuro-fuzzy system is a fuzzy system that uses a learning algorithm derived from or inspired by neural network theory to determine its parameters (fuzzy sets and fuzzy rules) by processing data samples.
- Principal Components Analysis, also known as PCA, is a technique commonly used for reducing the dimensionality of data while preserving as much as possible of the

information contained in the original data. PCA achieves this goal by projecting data onto a lower-dimensional subspace that retains most of the variance among the data points.

- NoSQL Database is a non-relational Data Management System, which does not require a fixed schema. It avoids joins, and is easy to scale. The major purpose of using a NoSQL database is for distributed data stores with humongous data storage needs. NoSQL is used for Big data and real-time web apps.
- Chunks are subsets of shared data. MongoDB separates sharded data into chunks that are distributed across the shards in the shared cluster. Each chunk has an inclusive lower and exclusive upper range based on the shard key. A balancer specific for each cluster handles the chunk distribution. The balancer runs as a background job and distributes the chunks as needed to achieve an even balance of chunks across all shards. This process is called even chuck distribution.
- HBase is a data model that is similar to Google's big table designed to provide quick random access to huge amounts of structured data. It leverages the fault tolerance provided by the Hadoop File System (HDFS).

SELF-ASSESSMENT QUESTIONS

- Discuss Maximum Likelihood Estimation and Maximum A Posteriori with an example.
- Discuss some type of the kernel function in SVM.
- Explain the need for Kernel Method and its Working process in SVM.
- What is Principal Components Analysis (PCA).
- How does Principal Components Analysis Work?
- Differentiate the difference between MongoDB and RDBMS.

FURTHER READINGS

- Bill Franks, "Taming the Big Data Tidal Wave: Finding Opportunities in Huge Data Streams with Advanced Analytics", John Wiley & sons, First Edition,2013.
- Rachel Schutt, Cathy O'Neil, "Doing Data Science", O'Reilly, First Edition,2013
- Foster Provost, Tom Fawcet, "Data Science for Business", O'Reilly, First Edition,2013.
- Bart Baesens, "Analytics in a Big Data World: The Essential Guide to Data Science and its Applications", Wiley, First Edition,2014.

4.1 Introduction to Hadoop

Hadoop is an Apache open source framework written in java that allows distributed processing of large datasets across clusters of computers using simple programming models. The Hadoop framework application works in an environment that provides distributed storage and computation across clusters of computers. Hadoop is designed to scale up from single server to thousands of machines, each offering local computation and storage.

4.1.1. History of Hadoop

Apache Software Foundation is the developers of Hadoop, and it's co-founders are Doug Cutting and Mike Cafarella.

It's co-founder Doug Cutting named it on his son's toy elephant. In October 2003 the first paper release was Google File System. In January 2006, MapReduce development started on the Apache Nutch which consisted of around 6000 lines coding for it and around 5000 lines coding for HDFS. In April 2006 Hadoop 0.1.0 was released.

4.1.2. Hadoop Overview

Hadoop is a framework that allows you to first store Big Data in a distributed environment, so that, you can process it parallely. There are basically two components in Hadoop:

The first one is HDFS for storage (Hadoop distributed File System), that allows you to store data of various formats across a cluster. The second one is YARN, for resource management in Hadoop. It allows parallel processing over the data, i.e. stored across HDFS.

HDFS

HDFS creates an abstraction, let me simplify it for you. Similar as virtualization, you can see HDFS logically as a single unit for storing Big Data, but actually you are storing your data across multiple nodes in a distributed fashion. HDFS follows master-slave architecture.

In HDFS, Namenode is the master node and Datanodes are the slaves. Namenode contains the metadata about the data stored in Data nodes, such as which data block is stored in which data node, where are the replications of the data block kept etc. The actual data is stored in Data Nodes.

I also want to add, we actually replicate the data blocks present in Data Nodes, and the default replication factor is 3. Since we are using commodity hardware and we know the failure rate of these hardwares are pretty high, so if one of the DataNodes fails, HDFS will still have the copy of those lost data blocks. You can also configure replication factor based on your requirements.

Hadoop-as-a-Solution

Let's understand how Hadoop provided the solution to the Big Data problems that we just discussed.

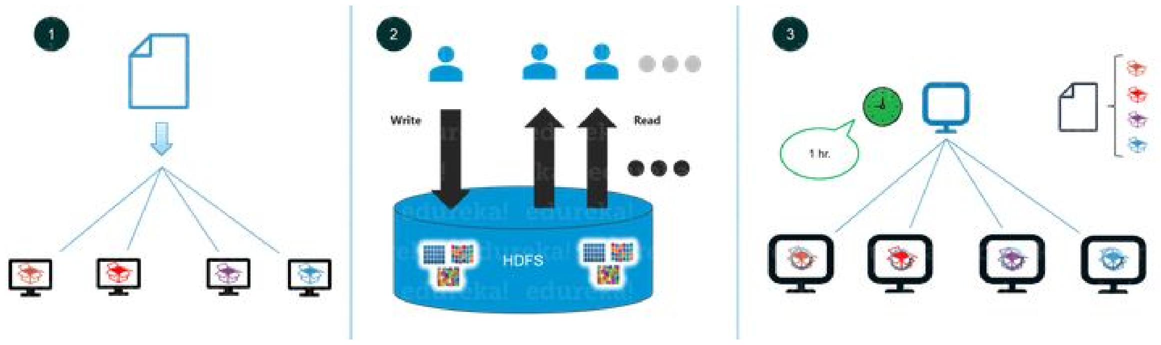


Figure 4.1: Hadoop-as-a-Solution

The first problem is storing Big data.

HDFS provides a distributed way to store Big data. Your data is stored in blocks across the DataNodes and you can specify the size of blocks. Basically, if you have 512MB of data and you have configured HDFS such that, it will create 128 MB of data blocks. So HDFS will divide data into 4 blocks as 512/128=4 and store it across different DataNodes, it will also replicate the data blocks on different DataNodes. Now, as we are using commodity hardware, hence storing is not a challenge.

It also solves the scaling problem. It focuses on horizontal scaling instead of vertical scaling. You can always add some extra data nodes to HDFS cluster as and when required, instead of scaling up the resources of your DataNodes. Let me summarize it for you basically for storing 1 TB of data, you don't need a 1TB system. You can instead do it on multiple 128GB systems or even less.

Next problem was storing the variety of data.

With HDFS you can store all kinds of data whether it is structured, semi-structured or unstructured. Since in HDFS, there is no pre-dumping schema validation. And it also follows write once and read many model. Due to this, you can just write the data once and you can read it multiple times for finding insights.

Third challenge was accessing & processing the data faster.

Yes, this is one of the major challenges with Big Data. In order to solve it, we move processing to data and not data to processing. What does it mean? Instead of moving data to the master node and then processing it. In MapReduce, the processing logic is sent to the various slave nodes & then data is processed parallel across different slave nodes. Then the processed results are sent to the master node where the result is merged and the response is sent back to the client.

In YARN architecture, we have ResourceManager and NodeManager. ResourceManager might or might not be configured on the same machine as NameNode. But, NodeManagers should be configured on the same machine where DataNodes are present.

YARN

YARN performs all your processing activities by allocating resources and scheduling tasks.

It has two major components, i.e. ResourceManager and NodeManager.

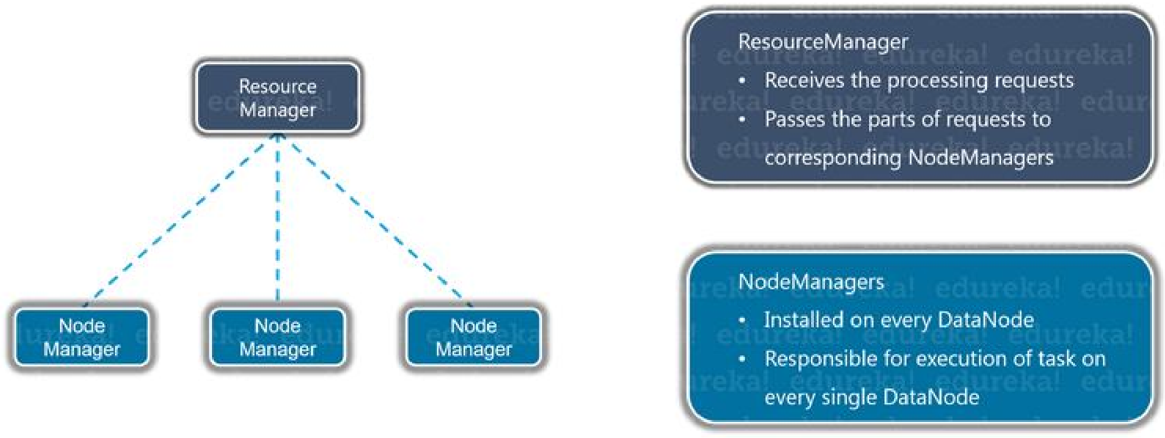

Resource Manager
Node Manager
Node Manager
Node Manager
ResourceManager
• Receives the processing requests
• Passes the parts of requests to corresponding NodeManagers
NodeManagers
• Installed on every DataNode
• Responsible for execution of task on every single DataNode

Figure 4.2: YARN

ResourceManager is again a master node. It receives the processing requests and then passes the parts of requests to corresponding NodeManagers accordingly, where the actual processing takes place. NodeManagers are installed on every DataNode. It is responsible for the execution of the task on every single DataNode.

4.1.3 RDBMS versus Hadoop

RDBMS(Relational Database Management System): RDBMS is an information management system, which is based on a data model.In RDBMS tables are used for information storage. Each row of the table represents a record and column represents an attribute of data. Organization of data and their manipulation processes are different in RDBMS from other databases. RDBMS ensures ACID (atomicity, consistency, integrity, durability) properties required for designing a database. The purpose of RDBMS is to store, manage, and retrieve data as quickly and reliably as possible.

Hadoop: It is an open-source software framework used for storing data and running applications on a group of commodity hardware. It has large storage capacity and high processing power. It can manage multiple concurrent processes at the same time. It is used in predictive analysis, data mining and machine learning. It can handle both structured and unstructured form of data. It is more flexible in storing, processing, and managing data than traditional RDBMS. Unlike traditional systems, Hadoop enables multiple analytical processes on the same data at the same time. It supports scalability very flexibly.

4.2 HDFS (Hadoop Distributed File System)

HDFS (Hadoop Distributed File System) is the primary storage system used by Hadoop applications. This open source framework works by rapidly transferring data between nodes. It's often used by companies who need to handle and store big data. HDFS is a key component of many Hadoop systems, as it provides a means for managing big data, as well as supporting big data analytics.

HDFS operates as a distributed file system designed to run on commodity hardware. HDFS is fault-tolerant and designed to be deployed on low-cost, commodity hardware. HDFS provides high throughput data access to application data and is suitable for applications that have large data sets and enables streaming access to file system data in Apache Hadoop.

HDFS has a master/slave architecture. An HDFS cluster consists of a single NameNode, a master server that manages the file system namespace and regulates access to files by clients. In addition, there are a number of DataNodes, usually one per node in the cluster, which manage storage attached to the nodes that they run on. HDFS exposes a file system namespace and allows user data to be stored in files. Internally, a file is split into one or more blocks and these blocks are stored in a set of DataNodes.

- The NameNode executes file system namespace operations like opening, closing, and renaming files and directories. It also determines the mapping of blocks to DataNodes.
- The DataNodes are responsible for serving read and write requests from the file system's clients. The DataNodes also perform block creation, deletion, and replication upon instruction from the NameNode.

The NameNode and DataNode are pieces of software designed to run on commodity machines. These machines typically run a GNU/Linux operating system (OS). HDFS is built using the Java language; any machine that supports Java can run the NameNode or the DataNode software. Usage of the highly portable Java language means that HDFS can be deployed on a wide range of machines. A typical deployment has a dedicated machine that runs only the NameNode software. Each of the other machines in the cluster runs one instance of the DataNode software. The architecture does not preclude running multiple DataNodes on the same machine but in a real deployment that is rarely the case.

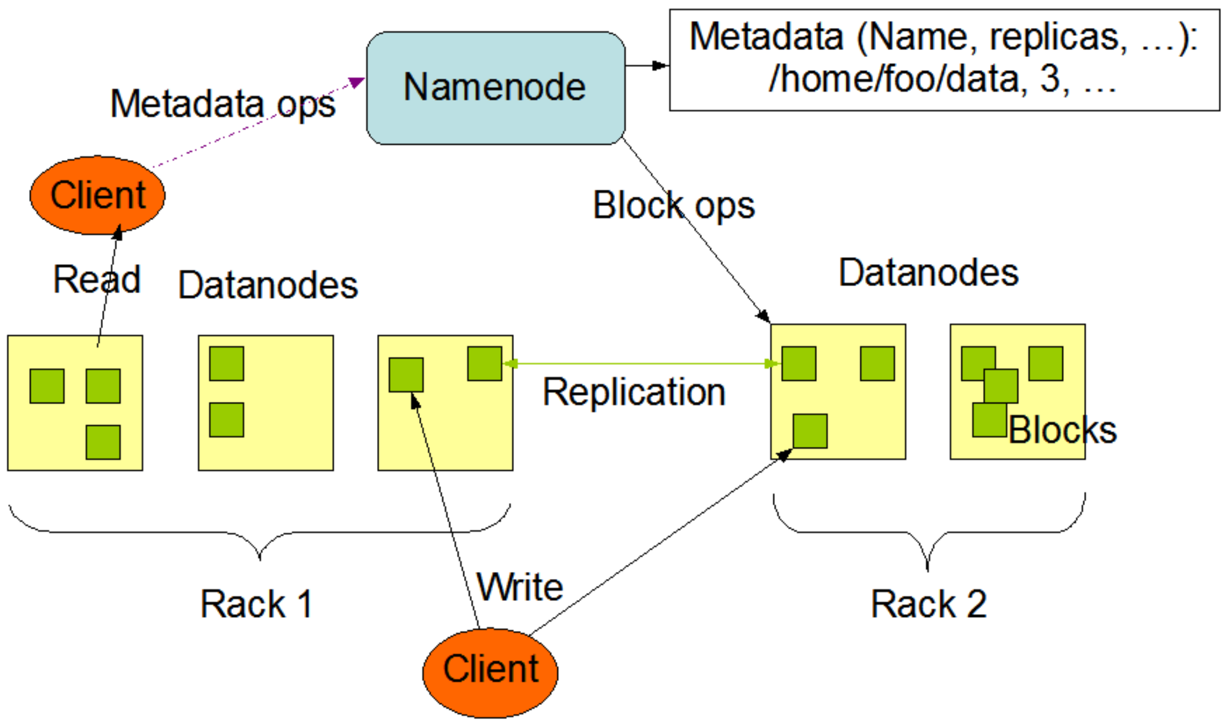


Figure 4.3: HDFS Architecture

The existence of a single NameNode in a cluster greatly simplifies the architecture of the system. The NameNode is the arbitrator and repository for all HDFS metadata. The system is designed in such a way that user data never flows through the NameNode.

The HDFS meaning and purpose is to achieve the following goals:

- Manage large datasets - Organizing and storing datasets can be a hard talk to handle. HDFS is used to manage the applications that have to deal with huge datasets. To do this, HDFS should have hundreds of nodes per cluster.
- Detecting faults - HDFS should have technology in place to scan and detect faults quickly and effectively as it includes a large number of commodity hardware. Failure of components is a common issue.
- Hardware efficiency - When large datasets are involved it can reduce the network traffic and increase the processing speed.

4.2.1. HDFS components

It’s important to know that there are three main components of Hadoop. Hadoop HDFS, Hadoop MapReduce, and Hadoop YARN.

- Hadoop HDFS - Hadoop Distributed File System (HDFS) is the storage unit of Hadoop.

- Hadoop MapReduce - Hadoop MapReduce is the processing unit of Hadoop. This software framework is used to write applications to process vast amounts of data.
- Hadoop YARN - Hadoop YARN is a resource management component of Hadoop. It processes and runs data for batch, stream, interactive, and graph processing - all of which are stored in HDFS.

4.2.2 Block Replication

HDFS is a reliable storage component of Hadoop. This is because every block stored in the file system is replicated on different Data Nodes in the cluster. This makes HDFS fault-tolerant.

HDFS stores each file as a sequence of blocks. The blocks of a file are replicated for fault tolerance.

- The NameNode makes all decisions regarding replication of blocks. It periodically receives a Blockreport from each of the DataNodes in the cluster. A Blockreport contains a list of all blocks on a DataNode.
- DataNode death may cause the replication factor of some blocks to fall below their specified value.
- The NameNode constantly tracks which blocks need to be replicated and initiates replication whenever necessary.

The default replication factor in HDFS is 3. This means that every block will have two more copies of it, each stored on separate DataNodes in the cluster. However, this number is configurable.

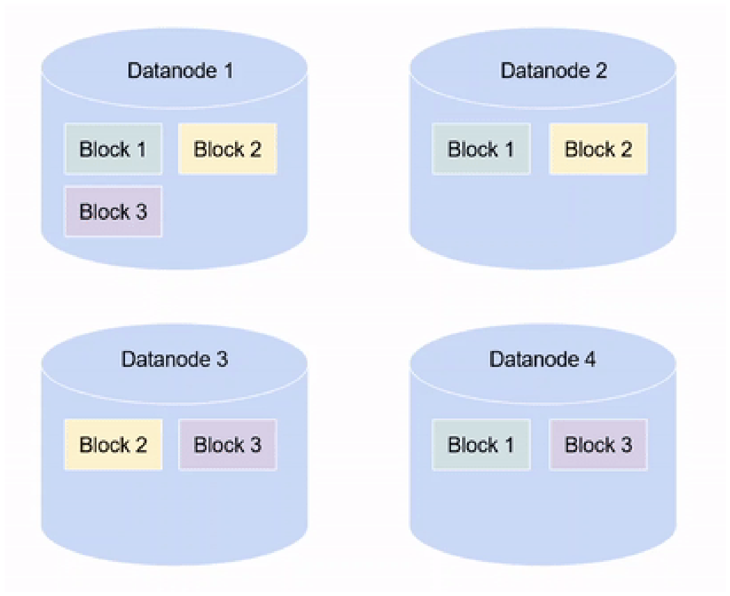


Figure 4.4: HDFS Cluster

But you must be wondering doesn’t that mean that we are taking up too much storage. For instance, if we have 5 blocks of 128MB each, that amounts to 5*128*3 = 1920 MB. True. But then these nodes are commodity hardware. We can easily scale the cluster to add more of these machines. The cost of buying machines is much lower than the cost of losing the data!

Now to understand, how does Namenode decide which Datanode to store the replicas on? This is done by Rack in Hadoop.

Rack in Hadoop

A Rack is a collection of machines (30-40 in Hadoop) that are stored in the same physical location. There are multiple racks in a Hadoop cluster, all connected through switches.

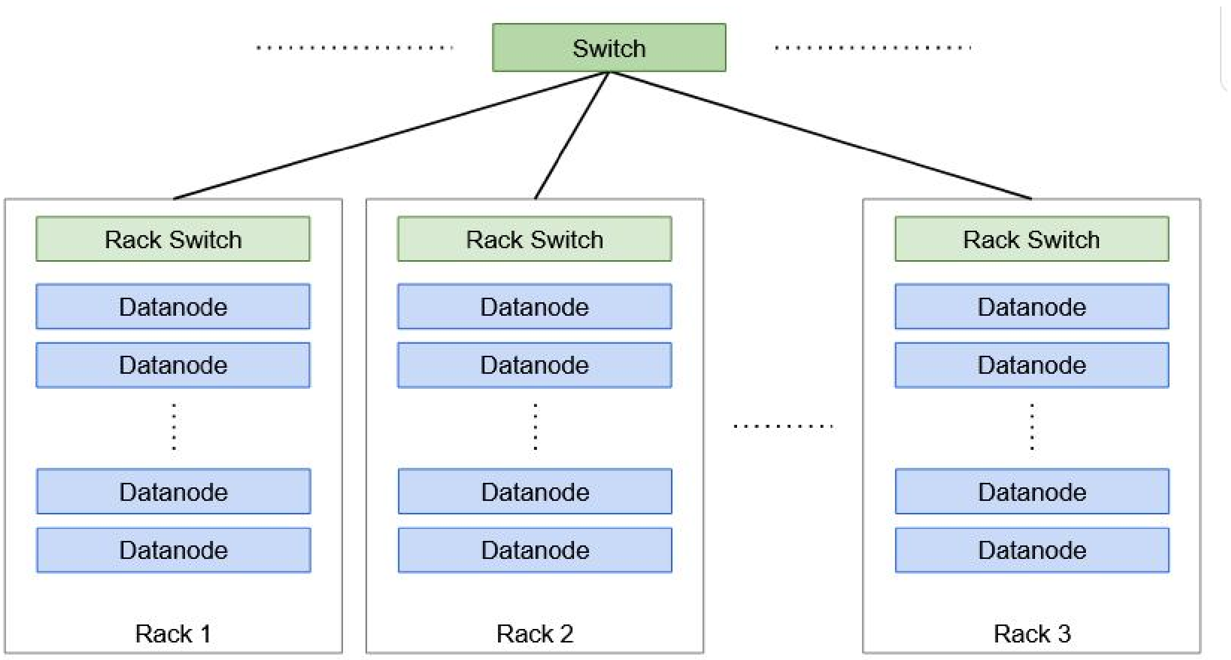


Figure 4.5: Rack in Hadoop

Rack awareness

Replica storage is a tradeoff between reliability and read/write bandwidth. To increase reliability, we need to store block replicas on different racks and Datanodes to increase fault tolerance. While the write bandwidth is lowest when replicas are stored on the same node. Therefore, Hadoop has a default strategy to deal with this conundrum, also known as the Rack Awareness algorithm.

For example, if the replication factor for a block is 3, then the first replica is stored on the same Datanode on which the client writes. The second replica is stored on a different Datanode but on a different rack, chosen randomly. While the third replica is stored on the same rack as the second but on a different Datanode, again chosen randomly. If, however, the replication factor was higher, then the subsequent replicas would be stored on random Data Nodes in the cluster.

4.3 Introduction to MapReduce

MapReduce is a software framework and programming model used for processing huge amounts of data. MapReduce is a hugely parallel processing framework that can be easily scaled over massive amounts of commodity hardware to meet the increased need for processing larger amounts of data. Once you get the mapping and reducing tasks right all it needs a change in the configuration in order to make it work on a larger set of data. This kind of extreme scalability from a single node to hundreds and even thousands of nodes is what makes MapReduce a top favorite among Big Data professionals worldwide.

The MapReduce is a paradigm which has two phases, the mapper phase, and the reducer phase. In the Mapper, the input is given in the form of a key-value pair. The output of the Mapper is fed to the reducer as input. The reducer runs only after the Mapper is over. The reducer too takes input in key-value format, and the output of reducer is the final output.

4.3.1 MapReduce Architecture

The whole process goes through four phases of execution namely, splitting, mapping, shuffling, and reducing. Figure 4.6 shows MapReduce Architecture.

The data goes through the following phases of MapReduce in Big Data

Input Splits: An input to a MapReduce in Big Data job is divided into fixed-size pieces called input splits Input split is a chunk of the input that is consumed by a single map

Mapping: This is the very first phase in the execution of map-reduce program. In this phase data in each split is passed to a mapping function to produce output values. In our example, a job of mapping phase is to count a number of occurrences of each word from input splits (more details about input-split is given below) and prepare a list in the form of <word, frequency>

Shuffling: This phase consumes the output of Mapping phase. Its task is to consolidate the relevant records from Mapping phase output. In our example, the same words are clubed together along with their respective frequency.

Reducing: In this phase, output values from the Shuffling phase are aggregated. This phase combines values from Shuffling phase and returns a single output value. In short, this phase summarizes the complete dataset.

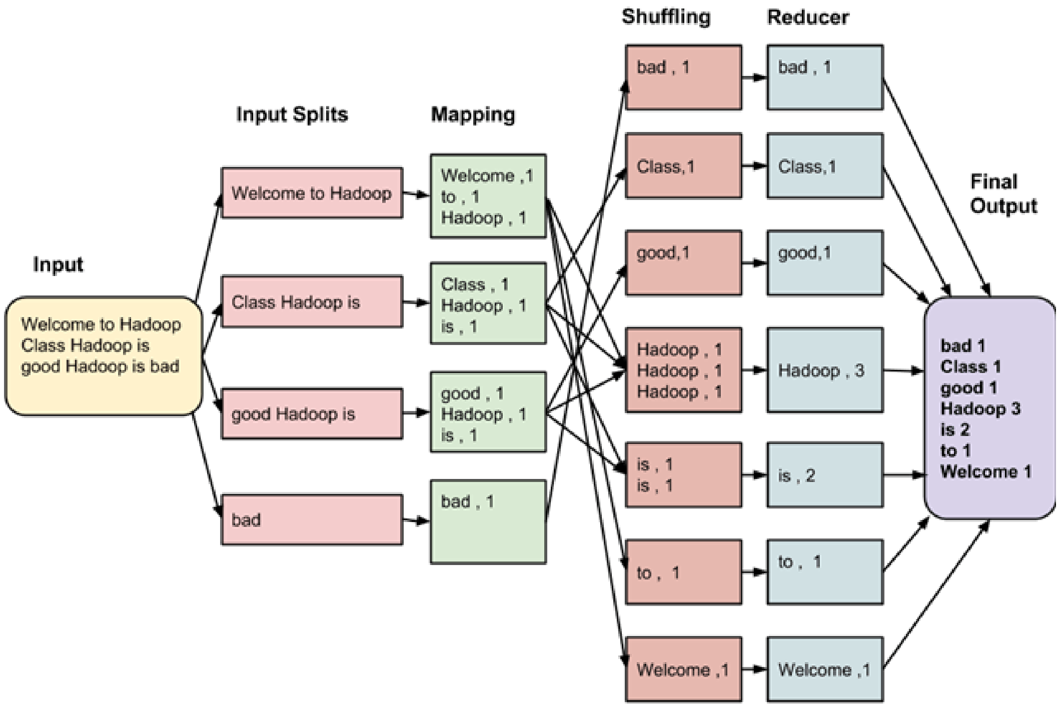

Shuffling
Reducer
Input Splits
Mapping
Final Output
Input
Welcome to Hadoop Class Hadoop is good Hadoop is bad
Welcome to Hadoop
Class Hadoop is
good Hadoop is
bad
Welcome ,1 to , 1 Hadoop , 1
Class , 1 Hadoop , 1 is , 1
good , 1 Hadoop , 1 is , 1
bad , 1
bad , 1
Class,1
good,1
Hadoop , 1 Hadoop , 1 Hadoop , 1
is , 1 is , 1
to , 1
Welcome ,1
bad , 1
Class,1
good,1
Hadoop , 3
is , 2
to , 1
Welcome ,1
bad 1 Class 1 good 1 Hadoop 3 is 2 to 1 Welcome 1

Figure 4.6: MapReduce Architecture

Here we discuss how the MapReduce works to get a better understanding of its architecture:

Hadoop divides the job into tasks. There are two types of tasks:

- Map tasks (Splits & Mapping)
- Reduce tasks (Shuffling, Reducing)

The complete execution process (execution of Map and Reduce tasks, both) is controlled by two types of entities called a

- Jobtracker: Acts like a master (responsible for complete execution of submitted job)
- Multiple Task Trackers: Acts like slaves, each of them performing the job

For every job submitted for execution in the system, there is one Jobtracker that resides on Namenode and there are multiple tasktrackers which reside on Datanode.

- A job is divided into multiple tasks which are then run onto multiple data nodes in a cluster.
- It is the responsibility of job tracker to coordinate the activity by scheduling tasks to run on different data nodes.
- Execution of individual task is then to look after by task tracker, which resides on every data node executing part of the job.

- Task tracker's responsibility is to send the progress report to the job tracker.
- In addition, task tracker periodically sends 'heartbeat' signal to the Jobtracker so as to notify him of the current state of the system.
- Thus job tracker keeps track of the overall progress of each job. In the event of task failure, the job tracker can reschedule it on a different task tracker.

4.4 Introduction to HBase

HBase is defined as an Open Source, distributed, NoSQL, Scalable database system, written in Java. It was developed by Apache software foundation for supporting Apache Hadoop, and it runs on top of HDFS (Hadoop distributed file system). HBase provides capabilities like Bigtable which contains billions of rows and millions of columns to store the vast amounts of data. It allows users to save billions of records and retrieves the information instantly. For example, the HBase consists of 5 billion records, and in that, if you wish to find 20 large items, HBase does it for you immediately: that's how it works.

HBase tables act as an input and output device for MapReduce jobs that run on Hadoop. It is a column-oriented key-value data store. HBase is well suited for faster read and write tasks on large data sets. HBase is not a direct replacement to the SQL database, but Apache Phoenix provides an SQL layer and JDBC (Java Database Connectivity) Driver that allows in integrating with analytics and business intelligence applications.

HBase Storage Mechanism

HBase is a column-oriented NoSQL database in which the data is stored in a table. The HBase table schema defines only column families. The HBase table contains multiple families, and each family can have unlimited columns. The column values are stored in a sequential manner on a disk. Every cell of the table has its timestamp (which means a digital record of time or occurrence of a particular event at a future time.)

HBase Table consists of the following components:

- **Table**: It's a collection of rows.
- **Row**: It is nothing but a collection of family columns.
- **Column family**: It's a collection of families.
- **Column**: It is a group of key-value pairs.
- **Timestamp**: It is a record of digital time and date.

Features of HBase:

Below mentioned are some of the essential features of the HBase:

- **Atomic and Consistent reads and writes**: HBase continuously works on reads and writes to fulfil the high-speed requirements of data processing.

- **Linear and modular scalability**: It is highly scalable, which means, we can add more machines to its cluster. By using fly, we can add more clusters to the network. When a new RegionServer is up, the cluster automatically begins rebalancing, it starts the RegionServer on the new node and scales up.
- **Automatic and configurable sharding of tables**: An HBase table is made up of regions and is hosted by the RegionServers. All the regions are distributed across all region servers on various DataNodes. HBase automatically splits these regions into small subregions till it reaches to a threshold size to reduce I/O time and overhead.
- **Easy to use Java API for client access**: HBase has been developed with the robust Java API support (client/server) which is simple to create and easy to execute.
- **Thrift gateway and RESTful Web services**: To support the front end apart from Java programing language, it supports Thrift and REST API.

Applications of HBase

Below mentioned are the areas where HBase is being used widely for supporting data processing.

Medical: The medical industry uses HBase to store the data related to patients such as patient diseases, information such as age, gender, etc., to run MapReduce on it.

Sports: Sports industry uses HBase to store the information related to the matches. This information would help perform analytics and in predicting the outcomes in future matches.

Web: The web is using the HBase services to store the history searches of the customers. This search information helps the companies to target the customer directly with the product or service that they had searched for.

Oil and petroleum: HBase is used to store the exploration data which helps in analysing and predicting the areas where oil can be found.

E-commerce: E-commerce is using HBase to record the customer logs and the products they are searching for. It enables the organizations in targeting the customer with the ads to induce him to buy their products or services.

4.4.1 HBase Architecture

In Hbase, the tables are divided into regions and served by region servers. Figure 4.7 shows HBase Architecture Diagram.

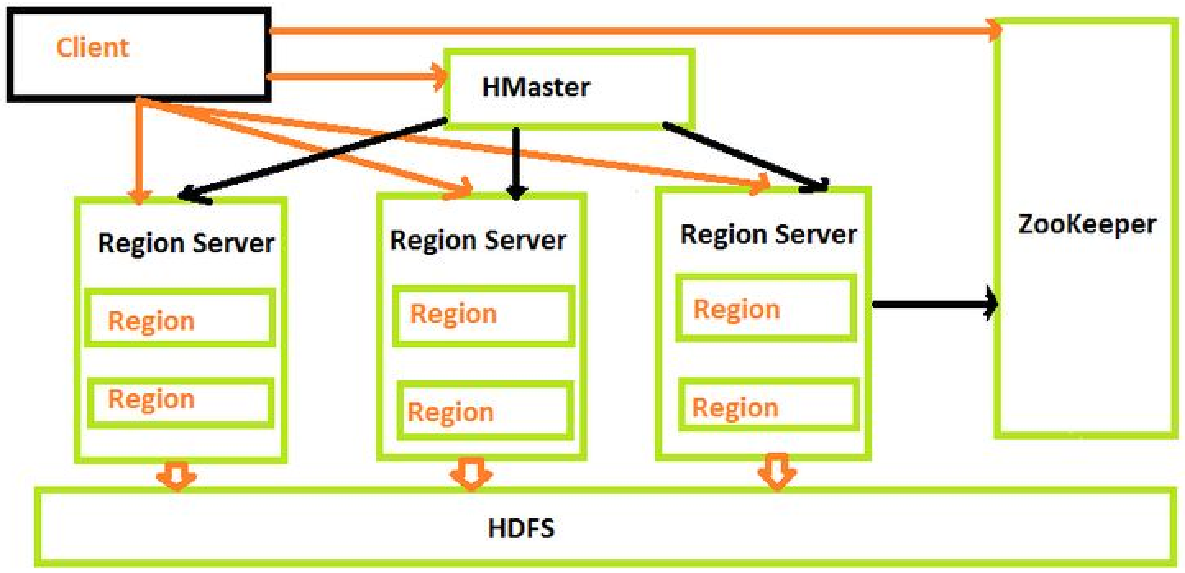


Figure 4.7 : HBase Architecture Diagram

The main component of Hbase are

- HMaster
- HRegionserver
- HRegions
- Zookeeper
- HDFS

HMaster

HMaster in HBase is the implementation of a Master server in HBase architecture. It acts as a monitoring agent to monitor all Region Server instances present in the cluster and acts as an interface for all the metadata changes. In a distributed cluster environment, Master runs on NameNode. Master runs several background threads.

The following are important roles performed by HMaster in HBase.

- Plays a vital role in terms of performance and maintaining nodes in the cluster.
- HMaster provides admin performance and distributes services to different region servers.
- HMaster assigns regions to region servers.
- HMaster has the features like controlling load balancing and failover to handle the load over nodes present in the cluster.
- When a client wants to change any schema and to change any Metadata operations, HMaster takes responsibility for these operations.

Some of the methods exposed by HMaster Interface are primarily Metadata oriented methods.

- Table (createTable, removeTable, enable, disable)
- ColumnFamily (add Column, modify Column)
- Region (move, assign)

The client communicates in a bi-directional way with both HMaster and ZooKeeper. For read and write operations, it directly contacts with HRegion servers. HMaster assigns regions to region servers and in turn, check the health status of region servers.

In entire architecture, we have multiple region servers. Hlog present in region servers which are going to store all the log files.

HBase Region Servers

When HBase Region Server receives writes and read requests from the client, it assigns the request to a specific region, where the actual column family resides. However, the client can directly contact with HRegion servers, there is no need of HMaster mandatory permission to the client regarding communication with HRegion servers. The client requires HMaster help when operations related to metadata and schema changes are required.

HRegionServer is the Region Server implementation. It is responsible for serving and managing regions or data that is present in a distributed cluster. The region servers run on Data Nodes present in the Hadoop cluster.

HMaster can get into contact with multiple HRegion servers and performs the following functions.

- Hosting and managing regions
- Splitting regions automatically
- Handling read and writes requests
- Communicating with the client directly

HBase Regions

HRegions are the basic building elements of HBase cluster that consists of the distribution of tables and are comprised of Column families. It contains multiple stores, one for each column family. It consists of mainly two components, which are Memstore and Hfile.

ZooKeeper

HBase Zookeeper is a centralized monitoring server which maintains configuration information and provides distributed synchronization. Distributed synchronization is to access the distributed applications running across the cluster with the responsibility of providing coordination services between nodes. If the client wants to communicate with regions, the server’s client has to approach ZooKeeper first.

It is an open source project, and it provides so many important services. Services provided by ZooKeeper

- Maintains Configuration information
- Provides distributed synchronization
- Client Communication establishment with region servers
- Provides ephemeral nodes for which represent different region servers
- Master servers usability of ephemeral nodes for discovering available servers in the cluster
- To track server failure and network partitions

Master and HBase slave nodes (region servers) registered themselves with ZooKeeper. The client needs access to ZK(zookeeper) quorum configuration to connect with master and region servers.

During a failure of nodes that present in HBase cluster, ZKquoram will trigger error messages, and it starts to repair the failed nodes.

HDFS

HDFS is a Hadoop distributed File System, as the name implies it provides a distributed environment for the storage and it is a file system designed in a way to run on commodity hardware. It stores each file in multiple blocks and to maintain fault tolerance, the blocks are replicated across a Hadoop cluster.

HDFS provides a high degree of fault –tolerance and runs on cheap commodity hardware. By adding nodes to the cluster and performing processing & storing by using the cheap commodity hardware, it will give the client better results as compared to the existing one.

In here, the data stored in each block replicates into 3 nodes any in a case when any node goes down there will be no loss of data, it will have a proper backup recovery mechanism.

HDFS get in contact with the HBase components and stores a large amount of data in a distributed manner.

4.4.2 HLog and HFile

HLog

HLog is used for disaster recovery. Why do you say that? Assuming that there is no HLog, we make a write request, which will first write to the MemStore and wait until the Memstore reaches a certain capacity before flushing to the storefile. But what if the host is powered off before then? The data of this part of the operation is all lost. This is obviously not the result we thought of, so with HLog, when a write request is initiated, it will first write to HLog and then write to MemStore. After success (at this time it has not been saved in the sotrefile), it will write to the client Enter the successful response.

Writing HLog process

The HLog file is an ordinary Hadoop Sequence File (also in the form of KeyValue, analogous to the KeyValue in the previous data block).

The Key of the Sequence File is the HLogKey object. The HLogKey records the ownership information of the written data, except for the table and region names. , Also includes sequence number and timestamp, timestamp is "write time", the starting value of sequence number is 0, or it is the last sequence number stored in the file system.

The Value of HLog Sequece File is the KeyValue object of HBase, which corresponds to the KeyValue in HFile. Including: row, column family, qualifier, timestamp, value, and "Key Type" (such as PUT or DELETE)

HFile

HFile is the storage format of KeyValue data in HBase (Do not think of KeyValue in the form of Map here, it will be better to understand). HFile is a binary format file of Hadoop. In fact, StoreFile is a lightweight packaging for HFile. That is, the bottom layer of StoreFile is HFile.

HFile consists of six parts:

- **Data (data block):** save the data in the table (in the form of KeyValue), this part can be compressed.
- **Meta (metadata block):** store user-defined KeyValue
- **File Info**: fixed length; some meta information of the file is recorded, for example: AVG_KEY_LEN, AVG_VALUE_LEN, LAST_KEY, COMPARATOR, MAX_SEQ_ID_KEY, etc.
- **Data Index (data block index):** records the starting index of each Data block
- **Meta Index (metadata block index):** records the starting index of each Meta block
- **Trailer**: fixed length; used to point to the starting point of other data blocks.

4.4.3 Data Replication

Data replication refers to copying data from one cluster to another cluster by replicating the writes when the first cluster receives it. In Apache HBase, Intercluster replication is achieved by log shipping asynchronously. Data replication is a disaster recovery solution and this can be implemented in Apache HBase. It can also serve more practically; for example, as a way to easily copy edits from a web-facing cluster to a "MapReduce" cluster which will process old and new data and ship back the results automatically.

The basic architecture pattern used for HBase replication is (HBase cluster) master-push; it is much easier to keep track of what's currently being replicated since each region server has its own write-ahead-log (aka WAL or HLog), just like other well known solutions like MySQL master/slave replication where there's only one bin log to keep track of. One master cluster can replicate to any number of slave clusters, and each region server will participate to replicate their own stream of edits. For more information on the different properties of master/slave replication and other types of replication, please consult How Google Serves Data From Multiple Datacenters.

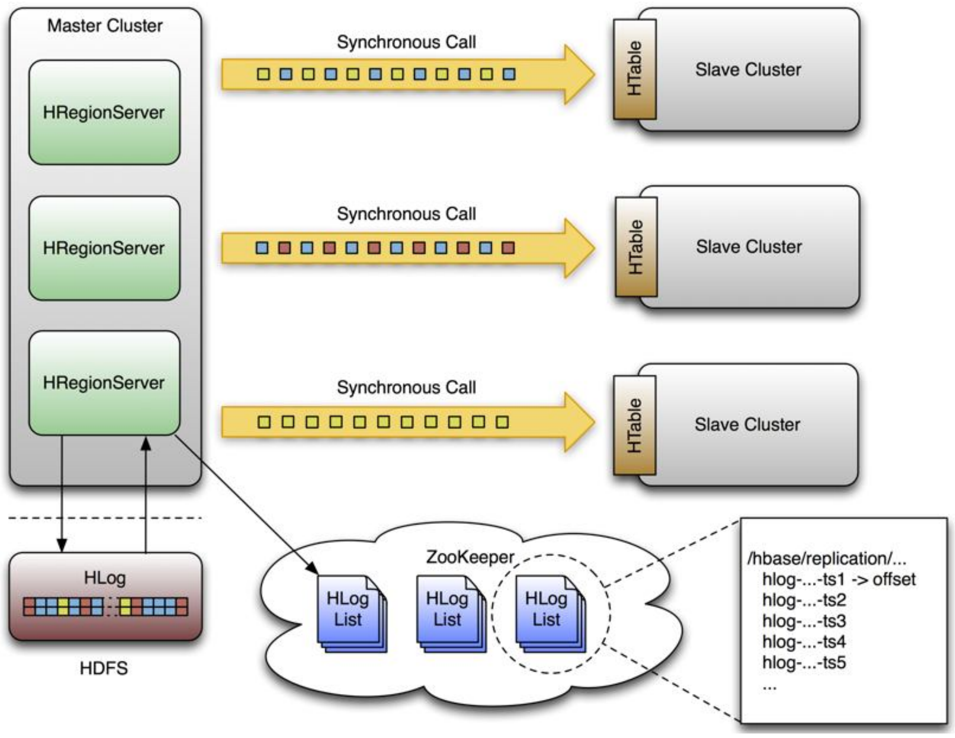


Figure 4.8: Data Replication Architecture

The replication is done asynchronously, meaning that the clusters can be geographically distant, the links between them can be offline for some time, and rows inserted on the master cluster won't be available at the same time on the slave clusters (eventual consistency).

The replication format used in this design is conceptually the same as MySQL's statement-based replication . Instead of SQL statements, whole WALEdits (consisting of multiple cell inserts coming from the clients' Put and Delete) are replicated in order to maintain atomicity.

The HLogs from each region server are the basis of HBase replication, and must be kept in HDFS as long as they are needed to replicate data to any slave cluster. Each RS reads from the oldest log it needs to replicate and keeps the current position inside ZooKeeper to simplify failure recovery. That position can be different for every slave cluster, same for the queue of HLogs to process.

The clusters participating in replication can be of asymmetric sizes and the master cluster will do its "best effort" to balance the stream of replication on the slave clusters by relying on randomization.

Some use cases for cluster replication include.

- Backup and disaster recovery.
- Geographic data distribution.
- Data aggregation.

Life of a WAL Edit

Apache HBase WAL logs go through many steps which are mentioned with the following steps.

1. Apache HBase data is modified using either PUT or DELETE operation.
2. Now the Apache HBase region server writes the request to the WAL. In case of failure, it will be rewritten.
3. If the modification cell is part of the replication column family then the edit log generated for the PUT or DELETE will go in queue for further replication.
4. After that, as a part of the batch method, the edit will be scanned from the log.
5. Now edit will be assigned a master UUID.
6. Now edits will be read by the Apache HBase region server in sequential order and order in buffers for example a table is assigned one buffer.
7. After that, the WAL that is going to replicate will be registered with ZooKeeper.
8. In the initial three-step, the edit is inserted and define as unique.
9. Now the Apache HBase Master will try the same process to send logs.
10. In case the Slave region server is not available then the Master selects a sub-set of region server and tries to send logs.
11. The WAL which are not able to deliver is stored in the ZooKeeper queue.
12. Once the Slave node gets online the normal buffer applies process is started.

4.5 Introduction to Hive

Hive is an ETL and Data warehousing tool developed on top of Hadoop Distributed File System (HDFS). Hive makes job easy for performing operations like

- Data encapsulation
- Ad-hoc queries
- Analysis of huge datasets

Hive provides the functionality of reading, writing, and managing large datasets residing in distributed storage. It runs SQL like queries called HQL (Hive query language) which gets internally converted to MapReduce jobs.

Using Hive, we can skip the requirement of the traditional approach of writing complex MapReduce programs. Hive supports Data Definition Language (DDL), Data Manipulation Language (DML), and User Defined Functions (UDF).

Hive Architecture

Figure 4.9 explains the Apache Hive architecture in detail

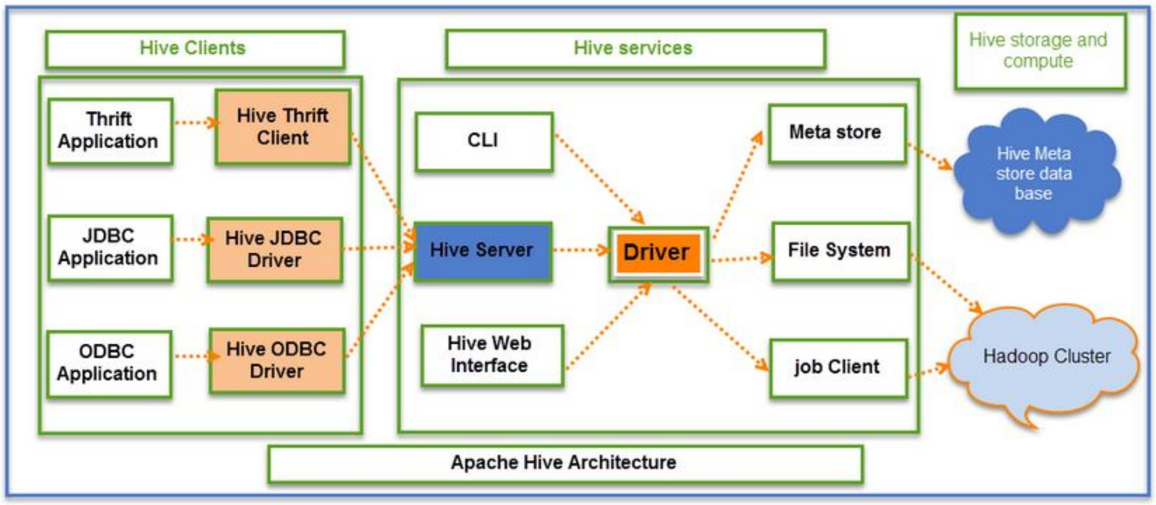


Figure 4.9: Apache Hive architecture

Hive chiefly consists of three core parts:

- Hive Clients: Hive offers a variety of drivers designed for communication with different applications. For example, Hive provides Thrift clients for Thrift-based applications. These clients and drivers then communicate with the Hive server, which falls under Hive services.

- Hive Services: Hive services perform client interactions with Hive. For example, if a client wants to perform a query, it must talk with Hive services.
- Hive Storage and Computing: Hive services such as file system, job client, and meta store then communicates with Hive storage and stores things like metadata table information and query results.

Job Execution Flow

Figure 4.10 shows the Job execution flow in Hive with Hadoop

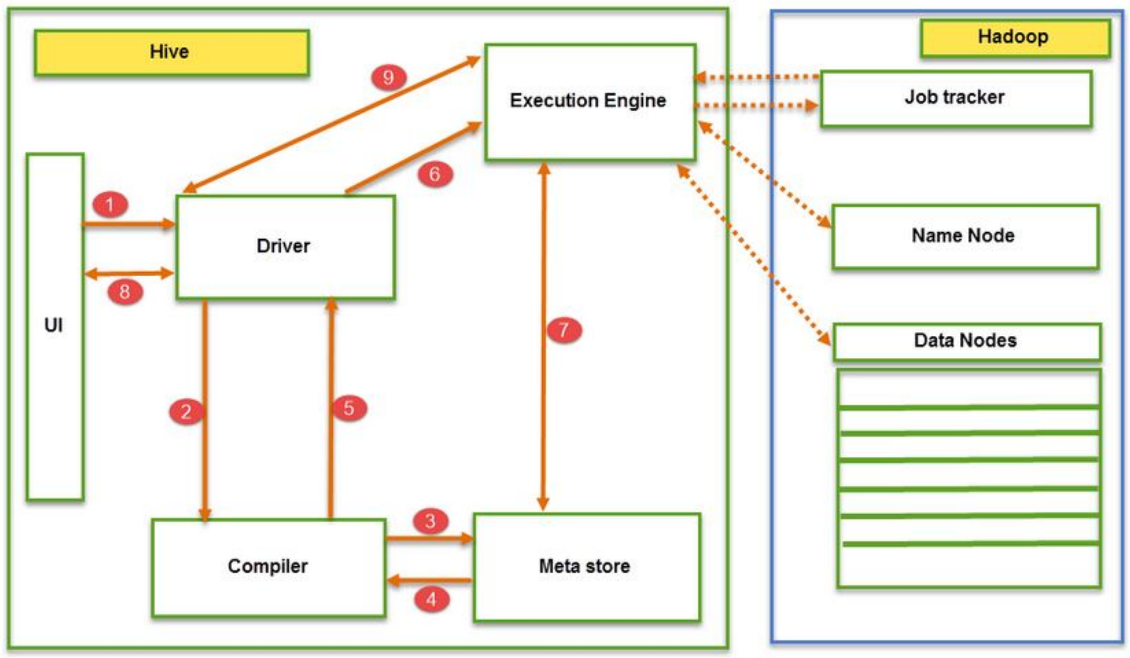


Figure 4.10: Job execution flow

The data flow in Hive behaves in the following pattern;

1. Executing Query from the UI(User Interface)
2. The driver is interacting with Compiler for getting the plan. (Here plan refers to query execution) process and its related metadata information gathering
3. The compiler creates the plan for a job to be executed. Compiler communicating with Meta store for getting metadata request
4. Meta store sends metadata information back to compiler
5. Compiler communicating with Driver with the proposed plan to execute the query
6. Driver Sending execution plans to Execution engine
7. Execution Engine (EE) acts as a bridge between Hive and Hadoop to process the query.
8. Fetching results from driver
9. Sending results to Execution engine. Once the results fetched from data nodes to the EE, it will send results back to driver and to UI (front end)

Different modes of Hive

Hive can operate in two modes depending on the size of data nodes in Hadoop.

These modes are,

- Local mode
- Map reduce mode

When to use Local mode:

- If the Hadoop installed under pseudo mode with having one data node we use Hive in this mode
- If the data size is smaller in term of limited to single local machine, we can use this mode
- Processing will be very fast on smaller data sets present in the local machine

When to use Map reduce mode:

- If Hadoop is having multiple data nodes and data is distributed across different node we use Hive in this mode
- It will perform on large amount of data sets and query going to execute in parallel way
- Processing of large data sets with better performance can be achieved through this mode

Hive's Features

These are Hive's chief characteristics:

- Hive is designed for querying and managing only structured data stored in tables
- Hive is scalable, fast, and uses familiar concepts
- Schema gets stored in a database, while processed data goes into a Hadoop Distributed File System (HDFS)
- Tables and databases get created first; then data gets loaded into the proper tables
- Hive supports four file formats: ORC, SEQUENCEFILE, RCFILE (Record Columnar File), and TEXTFILE
- Hive uses an SQL-inspired language, sparing the user from dealing with the complexity of MapReduce programming. It makes learning more accessible by utilizing familiar concepts found in relational databases, such as columns, tables, rows, and schema, etc.
- The most significant difference between the Hive Query Language (HQL) and SQL is that Hive executes queries on Hadoop's infrastructure instead of on a traditional database
- Since Hadoop's programming works on flat files, Hive uses directory structures to "partition" data, improving performance on specific queries
- Hive supports partition and buckets for fast and simple data retrieval

- Hive supports custom user-defined functions (UDF) for tasks like data cleansing and filtering. Hive UDFs can be defined according to programmers' requirements

Limitations of Hive

Hive has some limitations. They are:

- Hive doesn’t support OLTP. Hive supports Online Analytical Processing (OLAP), but not Online Transaction Processing (OLTP).
- It doesn’t support subqueries.
- It has a high latency.
- Hive tables don’t support delete or update operations.

4.6. Introduction to Spark

Apache Spark is a general-purpose & lightning fast cluster computing system. It provides a high-level API. For example, Java, Scala, Python, and R. Apache Spark is a tool for Running Spark Applications. Spark is 100 times faster than Bigdata Hadoop and 10 times faster than accessing data from disk.

Spark is written in Scala but provides rich APIs in Scala, Java, Python, and R. It can be integrated with Hadoop and can process existing Hadoop HDFS data.

Spark was built on the top of the Hadoop MapReduce. It was optimized to run in memory whereas alternative approaches like Hadoop's MapReduce writes data to and from computer hard drives. So, Spark process the data much quicker than other alternatives.

There are three ways of Spark deployment as explained below.

- Standalone − Spark Standalone deployment means Spark occupies the place on top of HDFS(Hadoop Distributed File System) and space is allocated for HDFS, explicitly. Here, Spark and MapReduce will run side by side to cover all spark jobs on cluster.
- Hadoop Yarn − Hadoop Yarn deployment means, simply, spark runs on Yarn without any pre-installation or root access required. It helps to integrate Spark into Hadoop ecosystem or Hadoop stack. It allows other components to run on top of stack.
- Spark in MapReduce (SIMR) − Spark in MapReduce is used to launch spark job in addition to standalone deployment. With SIMR, user can start Spark and uses its shell without any administrative access.

Components of Spark

The following illustration depicts the different components of Spark.

- **Apache Spark Core:** Spark Core is the underlying general execution engine for spark platform that all other functionality is built upon. It provides In-Memory computing and referencing datasets in external storage systems.
- **Spark SQL:** Spark SQL is a component on top of Spark Core that introduces a new data abstraction called SchemaRDD, which provides support for structured and semi-structured data.
- **Spark Streaming**: Spark Streaming leverages Spark Core's fast scheduling capability to perform streaming analytics. It ingests data in mini-batches and performs RDD (Resilient Distributed Datasets) transformations on those mini-batches of data.
- **MLlib (Machine Learning Library):** MLlib is a distributed machine learning framework above Spark because of the distributed memory-based Spark architecture. It is, according to benchmarks, done by the MLlib developers against the Alternating Least Squares (ALS) implementations. Spark MLlib is nine times as fast as the Hadoop disk-based version of Apache Mahout (before Mahout gained a Spark interface).
- **GraphX**: GraphX is a distributed graph-processing framework on top of Spark. It provides an API for expressing graph computation that can model the user-defined graphs by using Pregel abstraction API. It also provides an optimized runtime for this abstraction.

Resilient Distributed Dataset – RDD

The key abstraction of Spark knows as RDD. Resilient Distributed Dataset (RDD) is the fundamental unit of data in Apache Spark, which is a distributed collection of elements across cluster nodes and can perform parallel operations. Spark RDDs are immutable but can generate new RDD by transforming existing RDD.

There are three ways to create RDDs in Spark:

- Parallelized collections – We can create parallelized collections by invoking parallelize method in the driver program.
- External datasets – By calling a textFile method one can create RDDs. This method takes URL of the file and reads it as a collection of lines.
- Existing RDDs – By applying transformation operation on existing RDDs we can create new RDD.

Apache Spark RDDs support two types of operations:

- Transformation – Creates a new RDD from the existing one. It passes the dataset to the function and returns new dataset.
- Action – Spark Action returns final result to driver program or write it to the external data store.

Spark Shell

Apache Spark provides an interactive spark-shell. It helps Spark applications to easily run on the command line of the system. Using the Spark shell we can run/test our application code interactively. Spark can read from many types of data sources so that it can access and process a large amount of data.

Features of Apache Spark

- Fast - It provides high performance for both batch and streaming data, using a state-of-the-art DAG scheduler, a query optimizer, and a physical execution engine.
- Easy to Use - It facilitates to write the application in Java, Scala, Python, R, and SQL. It also provides more than 80 high-level operators.
- Generality - It provides a collection of libraries including SQL and DataFrames, MLlib for machine learning, GraphX, and Spark Streaming.
- Lightweight - It is a light unified analytics engine which is used for large scale data processing.
- Runs Everywhere - It can easily run on Hadoop, Apache Mesos, Kubernetes, standalone, or in the cloud.

Uses of Spark

- Data integration: The data generated by systems are not consistent enough to combine for analysis. To fetch consistent data from systems we can use processes like Extract, transform, and load (ETL). Spark is used to reduce the cost and time required for this ETL process.
- Stream processing: It is always difficult to handle the real-time generated data such as log files. Spark is capable enough to operate streams of data and refuses potentially fraudulent operations.
- Machine learning: Machine learning approaches become more feasible and increasingly accurate due to enhancement in the volume of data. As spark is capable of storing data in memory and can run repeated queries quickly, it makes it easy to work on machine learning algorithms.
- Interactive analytics: Spark is able to generate the respond rapidly. So, instead of running pre-defined queries, we can handle the data interactively.

4.7 Introduction to Apache Sqoop

Apache Sqoop is a tool designed for data transfer between the Hadoop Distributed File System and the relational databases or mainframes.

We can use Apache Sqoop for importing data from the RDBMS, that is, relational database management systems such as Oracle or MySQL or a mainframe into the HDFS (Hadoop Distributed File System).

We can use Sqoop for transforming data in Hadoop MapReduce and then exporting it back into the RDBMS.

Initially, Sqoop was developed and managed by Cloudera. Later on, on 23 July 2011, Sqoop was incubated by Apache Software Foundation. In April 2012, Sqoop was promoted as Apache's top-level project.

Apache Sqoop relies on the relational database to describe the schema for data to be imported. It uses the Hadoop MapReduce model for importing and exporting the data. This provides the capability of fault tolerance as well as parallel operation.

Sqoop can easily integrate with the Hadoop and dump structured data from RDBMS on HDFS, thus complementing the Hadoop's power.

Why do we need Sqoop?

Analytical processing using Hadoop requires loading of huge amounts of data from diverse sources into Hadoop clusters. This process of bulk data load into Hadoop, from heterogeneous sources and then processing it, comes with a certain set of challenges. Maintaining and ensuring data consistency and ensuring efficient utilization of resources, are some factors to consider before selecting the right approach for data load.

Major Issues:

1. Data load using Scripts: The traditional approach of using scripts to load data is not suitable for bulk data load into Hadoop; this approach is inefficient and very time-consuming.
2. Direct access to external data via Map-Reduce application: Providing direct access to the data residing at external systems (without loading into Hadoop) for map-reduce applications complicates these applications. So, this approach is not feasible.
3. In addition to having the ability to work with enormous data, Hadoop can work with data in several different forms. So, to load such heterogeneous data into Hadoop, different tools have been developed. Sqoop and Flume are two such data loading tools.

Sqoop Architecture

All the existing Database Management Systems are designed with SQL standard in mind. However, each DBMS differs with respect to dialect to some extent. So, this difference poses challenges when it comes to data transfers across the systems. Sqoop Connectors are components which help overcome these challenges.

Data transfer between Sqoop Hadoop and external storage system is made possible with the help of Sqoop's connectors.

Sqoop has connectors for working with a range of popular relational databases, including MySQL, PostgreSQL, Oracle, SQL Server, and DB2. Each of these connectors knows how to interact with its associated DBMS. There is also a generic JDBC connector for connecting to any database that supports Java's JDBC protocol. In addition, Sqoop Big data provides optimized MySQL and PostgreSQL connectors that use database-specific APIs to perform bulk transfers efficiently.

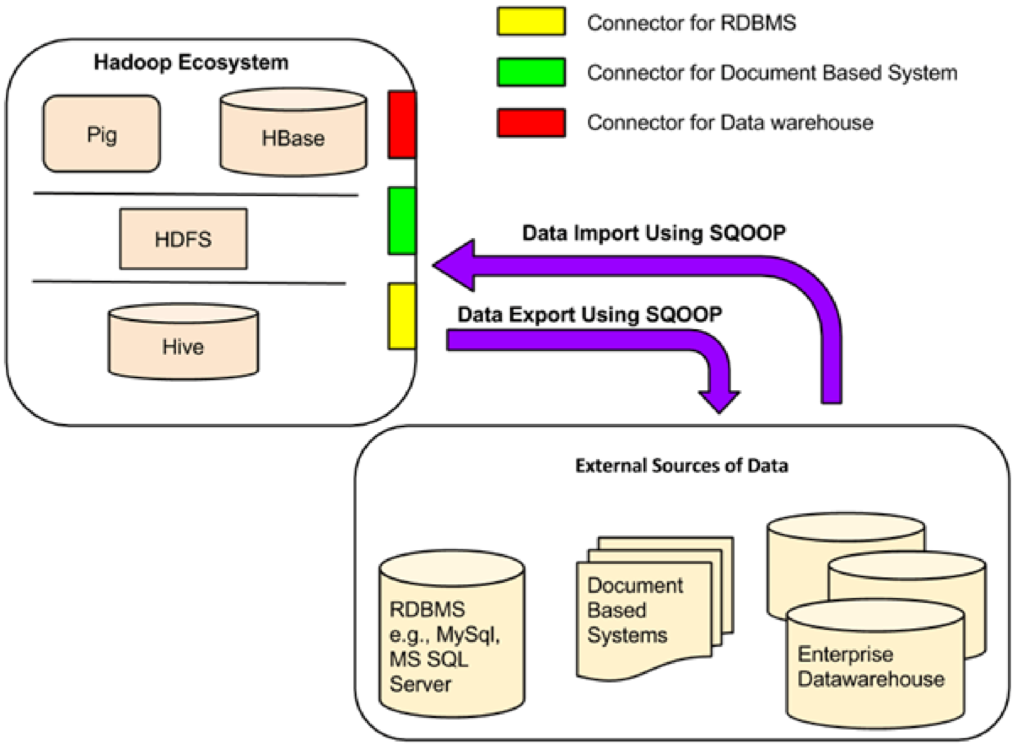


Figure 4.11: Sqoop Architecture

In addition to this, Sqoop in big data has various third-party connectors for data stores, ranging from enterprise data warehouses (including Netezza, Teradata, and Oracle) to NoSQL stores (such as Couchbase). However, these connectors do not come with Sqoop bundle; those need to be downloaded separately and can be added easily to an existing Sqoop installation.

Sqoop Import

The import tool imports individual tables from RDBMS to HDFS. Each row in a table is treated as a record in HDFS. All records are stored as text data in text files or as binary data in Avro and Sequence files.

Sqoop Export

The export tool exports a set of files from HDFS back to an RDBMS. The files given as input to Sqoop contain records, which are called as rows in table. Those are read and parsed into a set of records and delimited with user-specified delimiter.

Features of Sqoop

Sqoop provides many salient features like:

- Full Load
- Incremental Load
- Parallel import/export
- Import results of SQL query
- Compression
- Connectors for all major RDBMS Databases
- Kerberos Security Integration
- Load data directly into Hive/Hbase
- Support for Accumulo

SUMMARY

- Hadoop is an Apache open source framework written in java that allows distributed processing of large datasets across clusters of computers using simple programming models. The Hadoop framework application works in an environment that provides distributed storage and computation across clusters of computers.
- YARN performs all your processing activities by allocating resources and scheduling tasks.
- RDBMS is an information management system, which is based on a data model.In RDBMS tables are used for information storage. Each row of the table represents a record and column represents an attribute of data. Organization of data and their manipulation processes are different in RDBMS from other databases. RDBMS ensures ACID (atomicity, consistency, integrity, durability) properties required for designing a database.
- HDFS (Hadoop Distributed File System) is the primary storage system used by Hadoop applications. This open source framework works by rapidly transferring data between nodes. It's often used by companies who need to handle and store big data.
- The NameNode executes file system namespace operations like opening, closing, and renaming files and directories. It also determines the mapping of blocks to DataNodes.

- The DataNodes are responsible for serving read and write requests from the file system's clients. The DataNodes also perform block creation, deletion, and replication upon instruction from the NameNode
- Hadoop MapReduce is the processing unit of Hadoop. This software framework is used to write applications to process vast amounts of data.
- HBase is a column-oriented NoSQL database in which the data is stored in a table. The HBase table schema defines only column families. The HBase table contains multiple families, and each family can have unlimited columns. The column values are stored in a sequential manner on a disk. Every cell of the table has its timestamp.
- HMaster in HBase is the implementation of a Master server in HBase architecture. It acts as a monitoring agent to monitor all Region Server instances present in the cluster and acts as an interface for all the metadata changes. In a distributed cluster environment, Master runs on NameNode. Master runs several background threads.
- HRegions are the basic building elements of HBase cluster that consists of the distribution of tables and are comprised of Column families.
- Hive is an ETL and Data warehousing tool developed on top of Hadoop Distributed File System (HDFS).
- Apache Spark provides an interactive spark-shell. It helps Spark applications to easily run on the command line of the system. Using the Spark shell we can run/test our application code interactively. Spark can read from many types of data sources so that it can access and process a large amount of data.

SELF-ASSESSMENT QUESTIONS

- Define Hadoop
- Discuss the various issues faced in Big data that was addressed by Hadoop.
- Discuss the difference between RDBMS versus Hadoop
- Explain the architecture of HDFS.
- What are the components of HDFS.
- Define MapReduce.
- Define HBase.
- Explain the storage Mechanism of HBase.
- List the parts of HFile.
- Define Data replication.
- Explain the architecture of Hive.
- List the characteristics of Hive.
- Define Spark.
- Explain the key abstract of Resilient Distributed Dataset.

FURTHER READINGS

- Bill Franks, “Taming the Big Data Tidal Wave: Finding Opportunities in Huge Data Streams with Advanced Analytics”, John Wiley & sons, First Edition,2013.
- Rachel Schutt, Cathy O'Neil, “Doing Data Science”, O'Reilly, First Edition,2013
- Foster Provost, Tom Fawcet, “Data Science for Business”, O'Reilly, First Edition,2013.
- Bart Baesens, “Analytics in a Big Data World: The Essential Guide to Data Science and its Applications”, Wiley, First Edition,2014.

5. 1 Introduction to Streams Concepts

Streaming data is the continuous flow of data generated by various sources. By using stream processing technology, data streams can be processed, stored, analyzed, and acted upon as it's generated in real-time. Streaming data is data that is generated continuously by thousands of data sources, which typically send in the data records simultaneously, and in small sizes (order of Kilobytes). Streaming data includes a wide variety of data such as log files generated by customers using your mobile or web applications, ecommerce purchases, in-game player activity, information from social networks, financial trading floors, or geospatial services, and telemetry from connected devices or instrumentation in data centers.

This data needs to be processed sequentially and incrementally on a record-by-record basis or over sliding time windows, and used for a wide variety of analytics including correlations, aggregations, filtering, and sampling. Information derived from such analysis gives companies visibility into many aspects of their business and customer activity such as –service usage (for metering/billing), server activity, website clicks, and geo-location of devices, people, and physical goods –and enables them to respond promptly to emerging situations. For example, businesses can track changes in public sentiment on their brands and products by continuously analyzing social media streams, and respond in a timely fashion as the necessity arises.

5.1.1 Types of Data Streams:

- Data stream – A data stream is a(possibly unchained) sequence of tuples. Each tuple comprised of a set of attributes, similar to a row in a database table.
- Transactional data stream – It is a log interconnection between entities
  1. Credit card – purchases by consumers from producer
  2. Telecommunications – phone calls by callers to the dialed parties
  3. Web – accesses by clients of information at servers
- Measurement data streams –
  1. Sensor Networks – a physical natural phenomenon, road traffic
  2. IP Network – traffic at router interfaces
  3. Earth climate – temperature, humidity level at weather stations

5.1.2 Examples of streaming data

- Sensors in transportation vehicles, industrial equipment, and farm machinery send data to a streaming application. The application monitors performance, detects any potential defects in advance, and places a spare part order automatically preventing equipment down time.

- A financial institution tracks changes in the stock market in real time, computes value-at-risk, and automatically rebalances portfolios based on stock price movements.
- A real-estate website tracks a subset of data from consumers' mobile devices and makes real-time property recommendations of properties to visit based on their geo-location.
- A solar power company has to maintain power throughput for its customers, or pay penalties. It implemented a streaming data application that monitors of all of panels in the field, and schedules service in real time, thereby minimizing the periods of low throughput from each panel and the associated penalty payouts.
- A media publisher streams billions of clickstream records from its online properties, aggregates and enriches the data with demographic information about users, and optimizes content placement on its site, delivering relevancy and better experience to its audience.
- An online gaming company collects streaming data about player-game interactions, and feeds the data into its gaming platform. It then analyzes the data in real-time, offers incentives and dynamic experiences to engage its players.

5.1.3 Benefits of Streaming Data

Data collection is only one piece of the puzzle. Today's enterprise businesses simply cannot wait for data to be processed in batch form. Instead, everything from fraud detection and stock market platforms, to ride share apps and e-commerce websites rely on real-time data streams.

Paired with streaming data, applications evolve to not only integrate data, but process, filter, analyze, and react to that data in real-time, as it's received. This opens a new plethora of use cases such as real-time fraud detection, Netflix recommendations, or a seamless shopping experience across multiple devices that updates as you shop.

In short, any industry that deals with big data, can benefit from continuous, real-time data will benefit from this technology.

5.2 Stream Data Model

In the data stream model, some or all of the input is represented as a finite sequence of integers (from some finite domain) which is generally not available for random access, but instead arrives one at a time in a "stream". Consider a data stream S composed of individual items s_1,s_2,..., ordered by arrival time. Let A be a signal described by S. Assume that A is a function from a discrete and ordered domain to the range of reals.

There are four models for representing A using individual stream items:

1. In the aggregate model, each stream item s_i corresponds to a range value for some domain value.

2. In the cash register model, each stream item si represents a domain value and a partial range value r_i, such that $r_i >= 0$. Reconstructing the signal A involves aggregating all the r_i values corresponding to each domain value.

3. The turnstile model generalizes the cash register model by allowing any r_i to be negative. Thus, reconstructing the signal A involves adding/subtracting the contributions of stream items having positive/negative range values.

4. In the reset model, each stream item s_i corresponds to a range value and is understood to replace all previously reported range values for the given domain value.

Each of the four models defined above has an ordered and an unordered version. In the ordered version, stream items arrive over time in increasing order of the domain values. In the unordered version, the ordering of the domain does not correspond to the arrival order of stream items.

5.3 Streaming Data Architecture

Streaming data architecture is a framework of software components built to ingest and process large volumes of streaming data from multiple sources. While traditional data solutions focused on writing and reading data in batches, streaming data architecture consumes data immediately as it is generated, persists it to storage, and may include various additional components per use case – such as tools for real-time processing, data manipulation, and analytics.

Streaming architectures must account for the unique characteristics of data streams, which tend to generate massive amounts of data (terabytes to petabytes) that it is at best semi-structured and requires significant pre-processing and ETL to become useful.

5.3.1 Four Key Components of a Streaming Data Architecture

A streaming data architecture is a set of software components designed to handle large streams of raw data from various sources:

Message Broker (Stream Processor)

The stream processor collects data from its source, converts it to a standard message format, and then streams it continuously for consumption by other components. A storing streaming data component, such as a data warehouse/data lake, an ETL tool, or another type of component are examples of such components. Stream processors have a high throughput, but they don't do any data transformation or task scheduling.

Message Broker can act as a proxy between two applications where communication between them is achieved using queues. In such case we refer to it as point-to-point broker.

If one application is broadcasting a single message to multiple applications, we say that broker acts in Publish/Subscribe model.

Popular stream processing tools:

- Apache Kafka
- RabbitMQ

Batch processing and real-time ETL tools

In data-intensive organizations, process streaming data is an essential component of the big data architecture. There are many fully managed frameworks to choose from that all set up an end-to-end streaming data pipeline in the cloud to enable real-time analytics.

Example managed tools:

- Amazon Kinesis Data Streams
- Azure Event Hub
- Google Cloud PubSub

Streaming Data Storage

Organizations typically store their streaming event data in cloud object stores to serve as operational data lake due to the sheer volume and multi-structured nature of event streams. They offer a low-cost and long-term solution for storing large amounts of event data. They're also a flexible integration point, allowing tools from outside your streaming ecosystem to access data.

Examples:

- Amazon Redshift
- Microsoft Azure Data Lake Storage
- Google Cloud Storage

Data Analytics / Serverless Query Engine

With data processed and stored in a data warehouse/data lake, you will now need data analytics tools.

Examples (not exhaustive):

- Query engines – Athena, Presto, Hive, Redshift Spectrum, Pig
- Text search engines – Elasticsearch, OpenSearch, Solr, Kusto
- Streaming data analytics – Amazon Kinesis, Google Cloud DataFlow, Azure Stream Analytics

5.4 Stream Computing

- Stream computing is a computing paradigm that reads data from collections of software or hardware sensors in stream form and computes continuous data streams.
- Stream computing uses software programs that compute continuous data streams.
- Stream computing uses software algorithm that analyzes the data in real time.
- Stream computing is one effective way to support Big Data by providing extremely low-latency velocities with massively parallel processing architectures.
- It is becoming the fastest and most efficient way to obtain useful knowledge from Big Data.

There are three main technical approaches to stream computing at the present time.

Custom code: Traditionally, stream processing applications have been hand-coded in a low-level programming language such as C or C++. The current trend is toward using one of the other two technologies to achieve lower development and maintenance cost.

Stream-oriented SQL: Recent research activity has extended SQL with primitives for real-time operation. The main additions are the notion of real-time windows, over which SQL aggregates can be computed, facilities to perform pattern matching on sequences of messages, and primitives to deal with out-of-order data. There are now high performance Stream-oriented SQL engines from several vendors.

Rule engines: The final approach is to utilize a high performance implementation of a rule engine for stream processing. These systems are descendent of the rule engines found in expert systems in the 1980's and originally specified by the Artificial Intelligence community in pioneering work in the 1970s. Such systems contain rich pattern matching capabilities, but must be extended with aggregation and windowing constructs. Currently, there are a variety of commercial rule engines addressing the stream processing market.

5.5 Sampling Data in a Stream

Stream sampling is the process of collecting a representative sample of the elements of a data stream. The sample is usually much smaller than the entire stream, but can be designed to retain many important characteristics of the stream, and can be used to estimate many important aggregates on the stream.

Reservoir Sampling

This technique, due to Vitter allows the maintenance of a random sample of the stream of a particular target size in an online fashion. A reservoir R is maintained such that at time $t > n$ the probability of accepting point $x(t)$ in the reservoir is equal to n/t.

The algorithm is as follows:

- Fill the reservoir R with the first n points of the stream.
- At time t > n replace a randomly chosen (equal probability) entry in the reservoir R with acceptance probability n/t.

This leads to a reservoir R(t) such that each point x(1)…x(t) is contained in R(t) with equal property n/t.

Advantages

- The reservoir contains data points from all history of the stream with equal probability.
- Very simple implementation; adding a point requires only O(1)

Drawbacks

- A concept drift cannot be compensated; the oldest data point x(1) is equal important in this sampling technique as the latest data point x(t).

5.6 Filtering Streams

- The randomized algorithms and data structures we have seen so far always produce the correct answer but have a small probability of being slow.
- We will consider randomized algorithms that are always fast, but have a small probability of returning the wrong answer.
- More generally, we are interested in tradeoffs between the (likely) efficiency of the algorithm and the (likely) quality of its output.

Bloom Filters

Whenever a list or set is used, and space is consideration, a Bloom filter should be considered. When using a Bloom filter, consider the potential effects of false positives."

- It is a randomized data structure that is used to represent a set.
- It answers membership queries
- It can give FALSE POSITIVE while answering membership queries (very less %).
- But can't return FALSE NEGATIVE
  - POSSIBLY IN SET
  - DEFINITELY NOT IN SET
- Space efficient
- Bloom filters are a natural variant of hashing proposed by Burton Bloom in 1970 as a mechanism for supporting membership queries in sets.
- Example: Email spam filtering
  - We know 1 billion “good” email addresses
  - If an email comes from one of these, it is NOT spam.

Filtering Stream Content

- To motivate the Bloom-filter idea, consider a web crawler.
- It keeps, centrally, a list of all the URL's it has found so far.
- It assigns these URL's to any of a number of parallel tasks; these tasks stream back the URL's they find in the links they discover on a page.
- It needs to filter out those URL's it has seen before.

Role of the Bloom Filter

- A Bloom filter placed on the stream of URL's will declare that certain URL's have been seen before.
- Others will be declared new, and will be added to the list of URL's that need to be crawled.
- Unfortunately, the Bloom filter can have false positives.
- It can declare a URL has been seen before when it hasn't.
- But if it says "never seen", then it is truly new.

How a Bloom Filter Works?

- A Bloom filter is an array of bits, together with a number of hash functions.
- The argument of each hash function is a stream element and it returns a position in the array.
- Initially, all bits are 0.
- When input x arrives, we set to 1 the bits h(x). for each hash function h

Operations that a Bloom Filter supports

- Insert(x): To insert an element in the Bloom Filter.
- Lookup(x): to check whether an element is already present in Bloom Filter with a positive false probability.

5.7 Counting Distinct Elements in a Stream

Distinct Counting (also referred to as Count Distinct) is a commonly used analyzing function for Big Data analysis. It refers to the number of unique values in a column or array of data – in SQL the function is count(distinct col). The difference between the function count(distinct col) and count(col) is the distinct descriptor. Its role is to remove the duplicate values, therefore earning its name "Distinct Count."

Distinct Counting has a variety of uses. A common use case is with websites and apps that are counting values. Here, PV/UV is the most commonly used index, where UV (Unique Visitor) is the de-duplicated value, causing each unique visitor to be counted as one. For the owner of a website or app, PV (Page View) represents the frequency or time of uses, UV represents the number of users, and both values are important. Combining these two numbers, we can more accurately understand the users and any changes in the frequency of PV/UV.

Calculations for Distinct Counting with Big Data

Many researchers have already realized there is room for optimization here and have developed a variety of formulas and data structures in response. The most popular two being HyperLogLog and Bitmap.

The similarity of the two algorithms is that both of them use extremely refined structures to store a set of distinct values (or complete set). Not only will this return the distinct value, but this structure can also perform follow up calculations (for example yesterday's and today's Distinct Count). Compared to de-duplicating at the origin value every single time, the efficiency of storage and calculations are greatly improved in both of these algorithms.

However, these two algorithms have very obvious differences:

- HyperLogLog (HLL) has a low complexity level in terms of storage space (log(log(n)), giving it the name HLL), where it changes regardless of the dataset. Depending on the level of accuracy, one HLL takes up between 1KB to 64KB of space. On the other hand, since Bitmap uses one bit to represent each ID, as the dataset size increases, the space required will also increase. Storing 100,000,000 values with raw Bitmap will require around 12MB of space. As we can see here, Bitmap requires a lot more storage space than HLL by an order of one or two.
- HLL supports multiple types of data entries as input making it very easy to use; Bitmap only supports int/long types of values as input. Therefore, if the original value is a string, then the user needs to map it into int/long before we can use Bitmap.
- The reason why HLL supports multiple data types is because it uses Hash Functions. This function maps inputted values into binary bytes, then the binary byte undergoes bucketing until the first 1 appears in the final position, ultimately estimating how many multiple values are within this bucket. Since HLL uses Hash Functions and probability estimations, HLL calculated results are destined to be inaccurate. Even though HLL has multiple correction algorithms to reduce the margin of error, the fact of its inaccuracy remains. As accurate as it can be, its theoretical margin of error is still over 1%.
- Bitmap faithfully uses a bit, (1) or (0), for every ID. So, as long as it can guarantee the fact that every unique user has a unique ID value, Bitmap's results are usually very accurate.

Both of these calculations have their pros and cons: overall, HLL is very good but it lacks accuracy; Bitmap may take up a lot more space than HLL, but it does guarantee accuracy.

5.8 Estimating Moments

Estimating moments is a generalization of the problem of counting distinct elements in a stream. The problem, called computing "moments," involves the distribution of frequencies of different elements in the stream.

Suppose a stream consists of elements chosen from a universal set. Assume the universal set is ordered so we can speak of the ith element for any i.

Let m_i be the number of occurrences of the ith element for any i. Then the kth-order moment of the stream is the sum over all i of $(m_i)^k$

For example:-

The 0th moment is the sum of 1 of each mi that is greater than 0 i.e., 0th moment is a count of the number of distinct element in the stream.

The 1st moment is the sum of the m_i's, which must be the length of the stream. Thus, first moments are especially easy to compute i.e., just count the length of the stream seen so far.

The second moment is the sum of the squares of the m_i's. It is sometimes called the surprise number, since it measures how uneven the distribution of elements in the stream is.

To see the distinction, suppose we have a stream of length 100, in which eleven different elements appear. The most even distribution of these eleven elements would have one appearing 10 times and the other ten appearing 9 times each.

In this case, the surprise number is $10^2 + 10 \times 9^2 = 910$. At the other extreme, one of the eleven elements could appear 90 times and the other ten appear 1 time each. Then, the surprise number would be $90^2 + 10 \times 12 = 8110$.

5.9 Counting Oneness in a Window

We have a window of length N on a binary stream. We want at all times to be able to answer queries of the form "how many 1's are there in the last k bits?" for any $k \leq N$. For this purpose we use the DGIM algorithm.

The basic version of the algorithm uses O(log2 N) bits to represent a window of N bits, and allows us to estimate the number of 1's in the window with an error of no more than 50%.

To begin, each bit of the stream has a timestamp, the position in which it arrives. The first bit has timestamp 1, the second has timestamp 2, and so on.

Since we only need to distinguish positions within the window of length N, we shall represent timestamps modulo N, so they can be represented by log2 N bits. If we also store the total number of bits ever seen in the stream (i.e., the most recent timestamp) modulo N, then we can determine from a timestamp modulo N where in the current window the bit with that timestamp is.

We divide the window into buckets, 5 consisting of:

- The timestamp of its right (most recent) end.
- The number of 1's in the bucket. This number must be a power of 2, and we refer to the number of 1's as the size of the bucket.

To represent a bucket, we need log2 N bits to represent the timestamp (modulo N) of its right end. To represent the number of 1's we only need log2 log2 N bits. The reason is that we know this number i is a power of 2, say 2j , so we can represent i by coding j in binary. Since j is at most log2 N, it requires log2 log2 N bits. Thus, O(logN) bits suffice to represent a bucket. There are six rules that must be followed when representing a stream by buckets.

- The right end of a bucket is always a position with a 1.
- Every position with a 1 is in some bucket.
- No position is in more than one bucket.
- There are one or two buckets of any given size, up to some maximum size.
- All sizes must be a power of 2.
- Buckets cannot decrease in size as we move to the left (back in time).

5.10 Decaying Window

This algorithm allows you to identify the most popular elements (trending, in other words) in an incoming data stream.

The decaying window algorithm not only tracks the most recurring elements in an incoming data stream, but also discounts any random spikes or spam requests that might have boosted an element's frequency.

In a decaying window, you assign a score or weight to every element of the incoming data stream. Further, you need to calculate the aggregate sum for each distinct element by adding all the weights assigned to that element. The element with the highest total score is listed as trending or the most popular.

- Assign each element with a weight/score.
- Calculate aggregate sum for each distinct element by adding all the weights assigned to that element.

Advantages of Decaying Window Algorithm:-

- Sudden spikes or spam data is taken care.
- New element is given more weight by this mechanism, to achieve right trending output.

SUMMARY

- Streaming data is the continuous flow of data generated by various sources. By using stream processing technology, data streams can be processed, stored, analyzed, and acted upon as it's generated in real-time.
- Streaming data architecture is a framework of software components built to ingest and process large volumes of streaming data from multiple sources.
- The stream processor collects data from its source, converts it to a standard message format, and then streams it continuously for consumption by other components.
- Stream computing is a computing paradigm that reads data from collections of software or hardware sensors in stream form and computes continuous data streams.
- Stream sampling is the process of collecting a representative sample of the elements of a data stream. The sample is usually much smaller than the entire stream, but can be designed to retain many important characteristics of the stream, and can be used to estimate many important aggregates on the stream.
- Distinct Counting (also referred to as Count Distinct) is a commonly used analyzing function for Big Data analysis. It refers to the number of unique values in a column or array of data – in SQL the function is count(distinct col).
- Estimating moments is a generalization of the problem of counting distinct elements in a stream. The problem, called computing "moments," involves the distribution of frequencies of different elements in the stream.

SELF-ASSESSMENT QUESTIONS

- Define Streaming data.
- Explain the types of Data Streams.
- Discuss the model of data streams.
- What is message broket?
- Discuss the technical approached of stream computing.
- Explain the algorithm of Reservoir Sampling.
- What is filtering Streams.
- Define Estimating Moments.
- Explain Decaying Window.

FURTHER READINGS

- Bill Franks, “Taming the Big Data Tidal Wave: Finding Opportunities in Huge Data Streams with Advanced Analytics”, John Wiley & sons, First Edition,2013.
- Rachel Schutt, Cathy O'Neil, “Doing Data Science”, O'Reilly, First Edition,2013
- Foster Provost, Tom Fawcet, “Data Science for Business”, O'Reilly, First Edition,2013.
- Bart Baesens, “Analytics in a Big Data World: The Essential Guide to Data Science and its Applications”, Wiley, First Edition,2014.