

1. Introduction 2
2. Managing Large-scale Data 49
3. Big Data 162
4. Hadoop
 - i) Hadoop 209
 - ii) Hadoop Operations 254
5. Hadoop MapReduce
 - i) Hadoop MapReduce 273
 - ii) MapReduce 297
6. Hive and Pig Framework, Distributed Database and Distributed Hash Table 308
7. Text Analysis 393
8. Locality Sensitive Hashing
 - i) Locality Sensitive Hashing (LSH) 501
 - ii) LSH Method 2 550
9. Mining Data Stream
 - i) Data Stream Mining 551
 - ii) DGIM 3 photos 594
 - iii) Data Stream Mining Algorithms 597
10. Data Visualization
 - i) Data Visualization 608
 - ii) Visualization 3 other charts 611
 - iii) Visualization 4.1 scatter and Bubble plots 622
 - iv) Visualization 4.2 Box Plot 632
11. ML Model and Clustering Methods
 - i) Machine Learning Models 651
 - ii) Clustering 667
 - iii) Clustering Algorithms 673
 - iv) K-means Clustering 679

INTRODUCTORY CONCEPTS

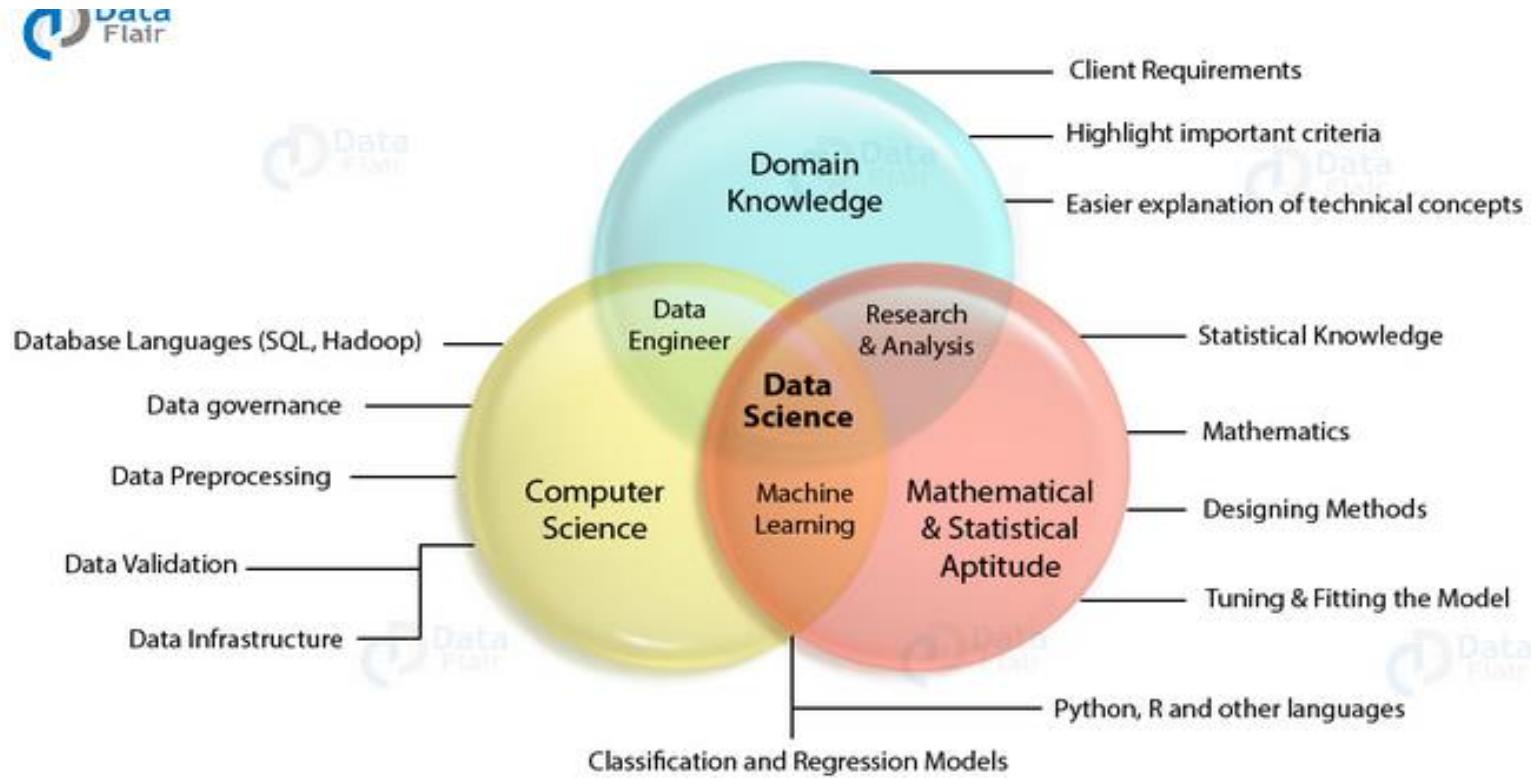
Introduction to Data Science

- Data science is the domain of study that deals with vast volumes of data using modern tools and techniques to find unseen patterns, derive meaningful information, and make business decisions
- Data Science is a comprehensive process that involves preprocessing, analysis, visualization and prediction.
- Data science is about finding hidden patterns in the data.



Introduction to Data Science

- Data science involves various underlying fields like statistics, mathematics, computer science, predictive analytics, machine learning algorithm development, and new technologies to gain insights from big data.



Why Data Science ?

- Traditionally, the data that we had was mostly **structured** and **small** in size, which could be analyzed by using simple **BI tools**. Unlike data in the traditional systems which was mostly structured, today most of the data is **unstructured or semi-structured**.
- Year 2020 survey shows that, **more than 80 % of the data will be unstructured**. This data is generated from different sources like financial logs, text files, multimedia forms, sensors, and instruments. Simple BI tools are not capable of processing this huge volume and variety of data.
- This is why we need more complex and advanced analytical tools and algorithms for processing, analyzing and drawing meaningful insights out of it.

Why Data Science ?

- Data science or data-driven science enables better decision making, predictive analysis, and pattern discovery.
- Allows companies to increase efficiencies, manage costs, identify new market opportunities, and boost their market advantage.
- Data Science has helped to create smarter systems that can take autonomous decisions based on historical datasets.

BI vs Data Science

- Difference between BI and Data science

Features	Business Intelligence (BI)	Data Science
Data Sources	Structured (Usually SQL, often Data Warehouse)	Both Structured and Unstructured (logs, cloud data, SQL, NoSQL, text)
Approach	Statistics and Visualization	Statistics, Machine Learning, Graph Analysis, Neuro-linguistic Programming (NLP)
Focus	Past and Present	Present and Future
Tools	Pentaho, Microsoft BI, QlikView, R	RapidMiner, BigML, Weka, R

Types of data scientists

- Type A Data Scientists
 - The A here stands for **Analysis**. This is a more static approach towards analysis of data or gaining insights from it. The work of a Type A data scientist is more closely related to that of a statistician.
 - A Type A Data Scientist is well versed with data cleaning, working with large data-sets, data visualization, domain knowledge, etc.

Types of data scientists

- Type B Data Scientists
 - The B here stands **Building**. While Type B Data Scientists share their background in statistics with Type A Data Scientists, they are well versed in coding and fundamentals of software engineering. They are responsible for building data products that directly interact with the user.
 - This helps them to craft products that provide recommendations and other forms of interactive results to the user.

Application

- Examples
- Gmail filters your emails in the spam and non-spam categories
 - Before Google/Gmail decides to segregate the emails into spam or not spam category, before it arrives to your mailbox, hundreds of rules apply to those email in the data centres. These rules describe the properties of a spam email.
 - There are common types of spam filters which are used by Gmail/Google —Blatant Blocking, Bulk Email Filter, Category Filters, Null Sender Disposition, Null Sender Header Tag Validation.
 - There are ways to avoid spam filtering and send your emails straight to the inbox.

Application

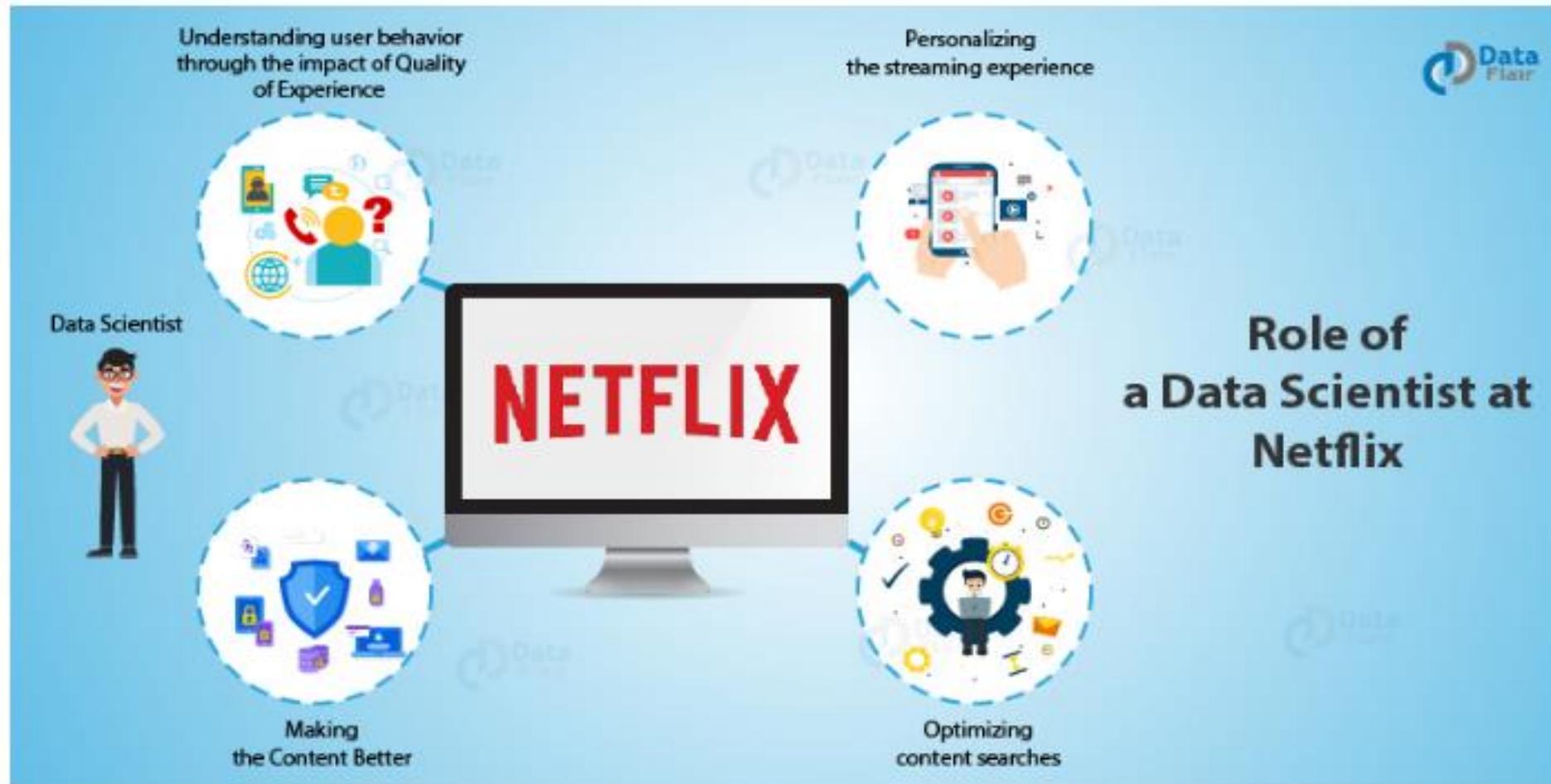
- Examples
 - Gmail filters your emails in the spam and non-spam categories



- Spam detection is a supervised machine learning problem. This means you must provide your machine learning model with a set of examples of spam and ham messages and let it find the relevant patterns that separate the two different categories. Most email providers have their own vast data sets of labelled emails.

Case Study

- How Netflix Used Data Science to Improve its Recommendation System?



Case Study

- How Netflix Used Data Science to Improve its Recommendation System?
 - In order to make this happen, Netflix invested in a lot of algorithms to provide a flawless movie experience to its users. One of such algorithms is the **recommendation system** that is used by Netflix to provide suggestions to the users.
 - A recommendation system **understands the needs of the users** and provides suggestions of the various cinematographic products. System takes the information about the user as an input.
 - This information can be in the form of the **past usage of product or the ratings** that were provided to the product. It then processes this information to predict how much the user would rate or prefer the product. A recommendation system makes use of a variety of [machine learning algorithms](#).

Case Study

- How Netflix Used Data Science to Improve its Recommendation System?
 - Another important role that a recommendation system plays today is to **search for similarity** between different products. In the case of Netflix, the recommendation system searches for movies that are similar to the ones you have watched or have liked previously.
 - Therefore, based on the movies that are watched, Netflix provides recommendations of the films that share a degree of similarity.
 - Now a days Netflix uses Hybrid Recommendation System (Content + Collaborative filtering) for suggesting content to its users.

Recommendation System

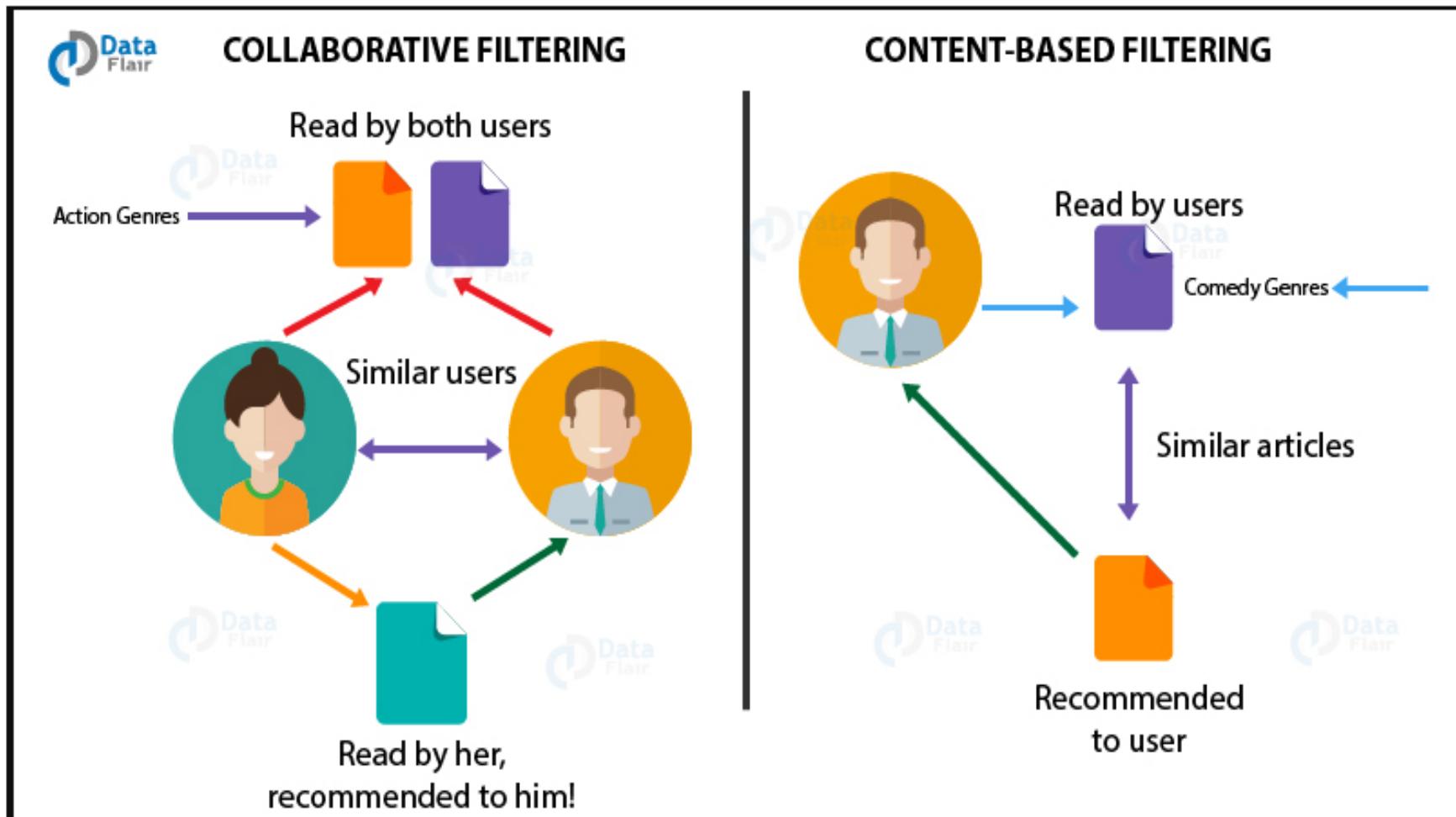
- There are two main types of Recommendation Systems –
- **Content-based recommendation systems**
 - In a content-based recommendation system, the **background knowledge of the products and customer information** are taken into consideration. Based on the content that you have viewed on Netflix, it provides you with similar suggestions.
 - **For example**, if you have watched a film that has a sci-fi genre, the content-based recommendation system will provide you with suggestions for similar films that have the same genre.

Recommendation System

- There are two main types of Recommendation Systems –
- **Collaborative filtering recommendation systems**
 - Unlike the content based filtering that provided recommendations of similar products, Collaborative Filtering provides recommendations based on the **similar profiles of its users**. One key advantage of collaborative filtering is that it is independent of the product knowledge.
 - Rather, it relies on the users with a basic assumption that what the users liked in the past will also like in the future. **For example**, if a person A watches crime, sci-fi and thriller genres and B watches sci-fi, thriller and action genres then A will also like action and B will like crime genre.

Recommendation System

- There are two main types of Recommendation Systems –



Applications

- Examples
 - Companies like Google and Amazon are using Data Science to develop powerful recommendation systems for their users.
 - Various financial companies are using predictive analytics and forecasting methods to predict stock prices.
 - Asking a personal assistant like Alexa or Siri for a recommendation

Applications

- Data science is all about using data to solve problems.
- **Decision making:** The problem could be decision making such as identifying which email is spam and which is not. OR understand the precise requirements of your customers from the existing data like the customer's past browsing history, purchase history, age and income.
- **Product recommendation:** product recommendation such as which movie to watch?
- **predictive analytics:** Predicting the outcome such as who will be the next President of the USA?
- So, the core job of a data scientist is to **understand** the data, **extract** useful information out of it and **apply** this in solving the problems.

Applications

- **Self-driving car:** self-driving cars collect live data from sensors, including radars, cameras, and lasers to create a map of its surroundings. Based on this data, it takes decisions like when to speed up, when to speed down, when to overtake, where to take a turn – making use of advanced machine learning algorithms
- **Weather forecasting:** Data from ships, aircraft, radars, satellites can be collected and analyzed to build models. These models will not only forecast the weather but also help in predicting the occurrence of any natural calamities. It will help you to take appropriate measures beforehand and save many precious lives.
- **Interest based search engine**
- **Talking to a Chabot for customer service**

Applications

- With the help of data science, Airlines can optimize operations in many ways, including:
 - Plan routes and decide whether to schedule direct or connecting flights
 - Build predictive analytics models to forecast flight delays
 - Offer personalized promotional offers based on customers booking patterns
 - Decide which class of planes to purchase for better overall performance
- To choose insurance plan
- To find restaurant
- To select internet plan
- To create marketing campaign
- To choose next destination for vacation

Applications

- All the domains where Data Science is creating its impression.



Applications

- **Data Science in Healthcare**
 - With the help of classification algorithms, doctors are able to detect cancer and tumors at an early stage using Image Recognition software.
 - Genetic Industries use Data Science for analyzing and classifying patterns of genomic sequences. Various virtual assistants are also helping patients to resolve their physical and mental ailments.
 - Drug Discovery with Data Science
 - Predictive Analytics in Healthcare: Finds various correlations and association of symptoms, finds habits, diseases and then makes meaningful predictions.
 - Monitoring Patient Health

Applications

- **Data Science in E-commerce**
 - Amazon uses a recommendation system that recommends users various products based on their historical purchase. Data Scientists have developed recommendation systems predict user preferences using Machine Learning.
- **Data Science in Manufacturing**
 - Industrial robots have made taken over mundane and repetitive roles required in the manufacturing unit. These industrial robots are autonomous in nature and use Data Science technologies such as Reinforcement Learning and Image Recognition.

Applications

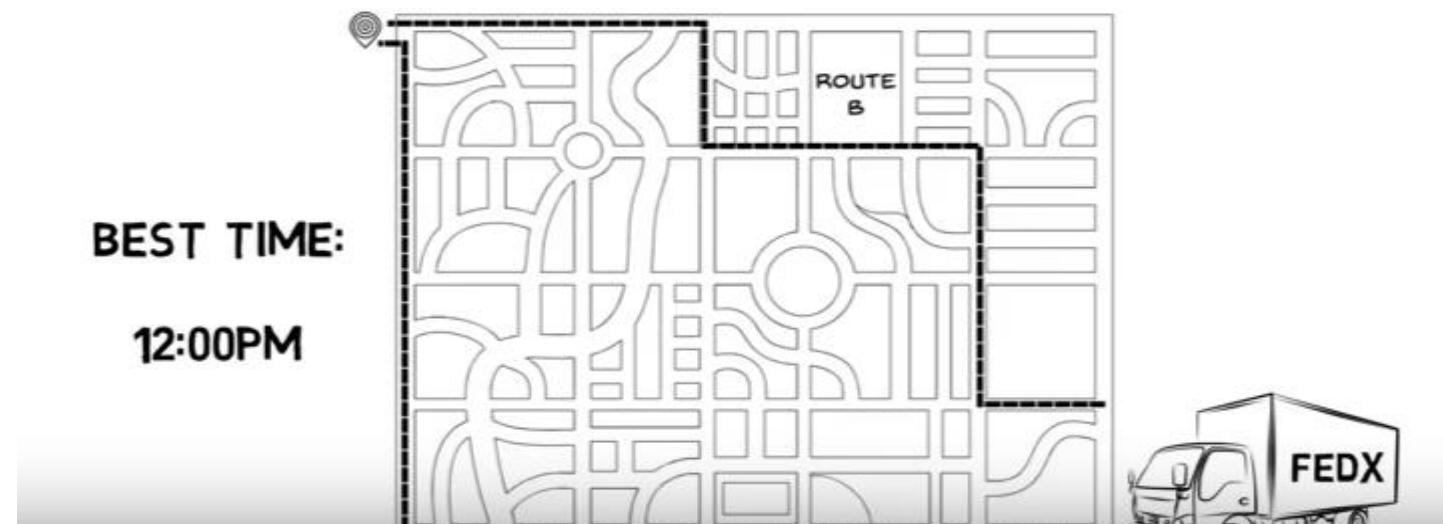
- **Data Science as Conversational Agents**
 - Amazon's Alexa and Siri by Apple use Speech Recognition to understand users. Data Scientists develop this speech recognition system, that converts human speech into textual data. Also, it uses various Machine Learning algorithms to classify user queries and provide an appropriate response.

Applications

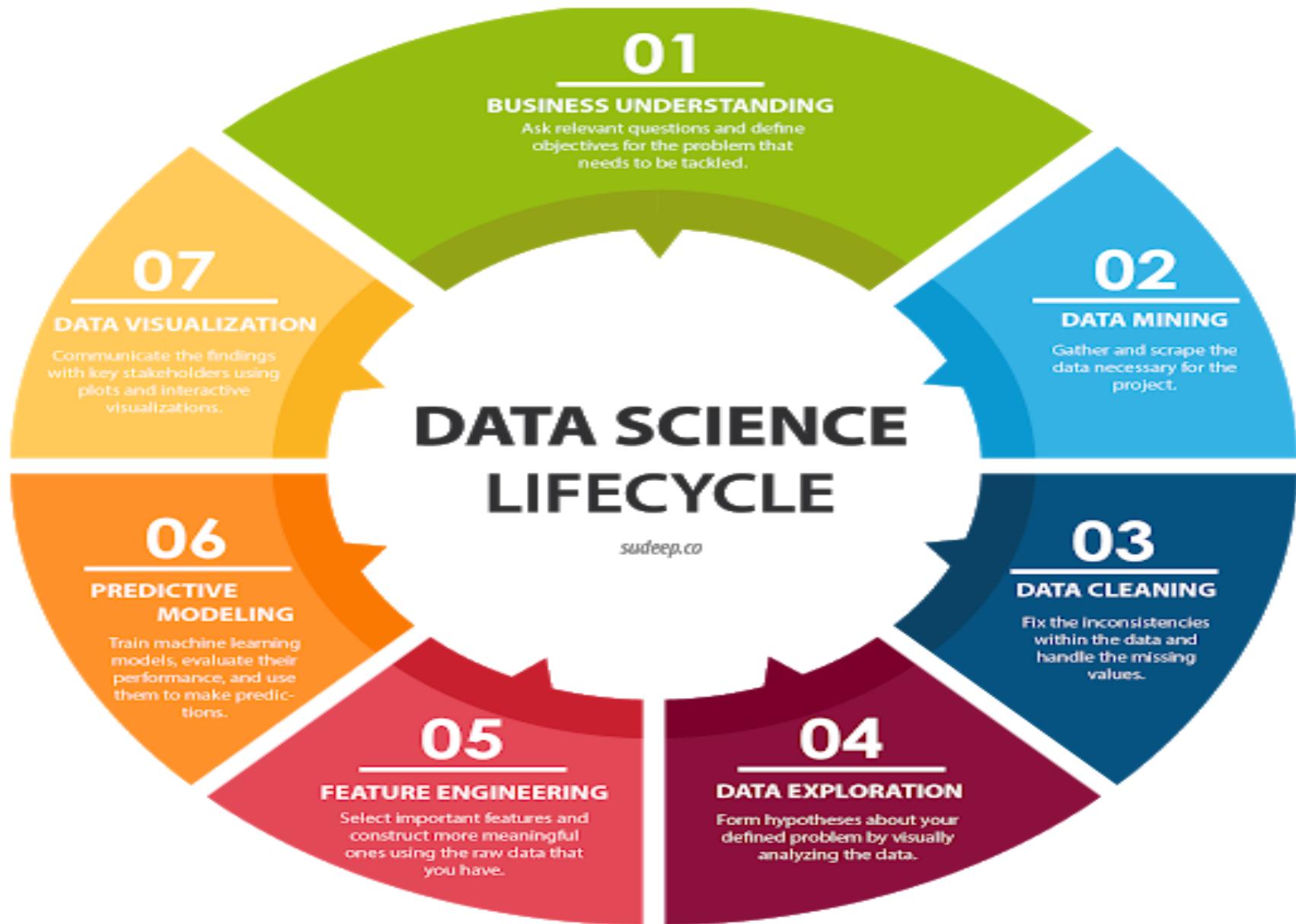
- **Data Science in Transport**

➤ Self Driving Cars use autonomous agents that utilize Reinforcement Learning and Detection algorithms. Self-Driving Cars are no longer fiction due to advancements in Data Science.

**LOGISTICS COMPANIES LIKE DHL, FEDEX HAVE
DISCOVERED THE BEST TIME AND ROUTES TO SHIP**



Data Science Life Cycle



Data Science Life Cycle

- **Step 1: Define Problem Statement**
 - Creating a well-defined problem statement is a first and critical step in data science. It is a brief description of the problem that you are going to solve.
why do we need a well-defined problem statement?
 - Most of the times, these initial set of a problem shared with you is vague and ambiguous.
 - For example, the problem statement: “I want to increase the revenue”, doesn’t tell you how much to increase the revenue such as 20% or 30%, for which products to increase revenue and what is the time frame to increase the revenue.
 - You have to make the problem statement clear, goal-oriented and measurable. This can be achieved by asking the right set of questions.

Data Science Life Cycle

- **Step 2: Data Collection**
 - Data collection is a systematic approach to gather relevant information from a variety of sources.
 - Depending on the problem statement, the data collection method is broadly classified into two categories.
 - Primary data collection
 - Secondary data collection
 - **Primary data collection:** When we have some unique problem and no related research is done on the subject. Then, we need to collect new data. This method is called as **primary data collection**.
 - For **example**, we want information on the average time that employees spend in a cafeteria across companies. There is no public data available of these. But we can collect the data through various methods such as surveys, interviews of employees and by monitoring the time spent by employees in cafeteria. This method is time-consuming.

Data Science Life Cycle

- **Step 2: Data Collection**
 - **Secondary data collection:** To use the data which is readily available or collected by someone else. These data can be found on the internet, news articles, government census, magazines and so on. This method is called as **secondary data collection**.
 - This method is less time-consuming than the primary method.

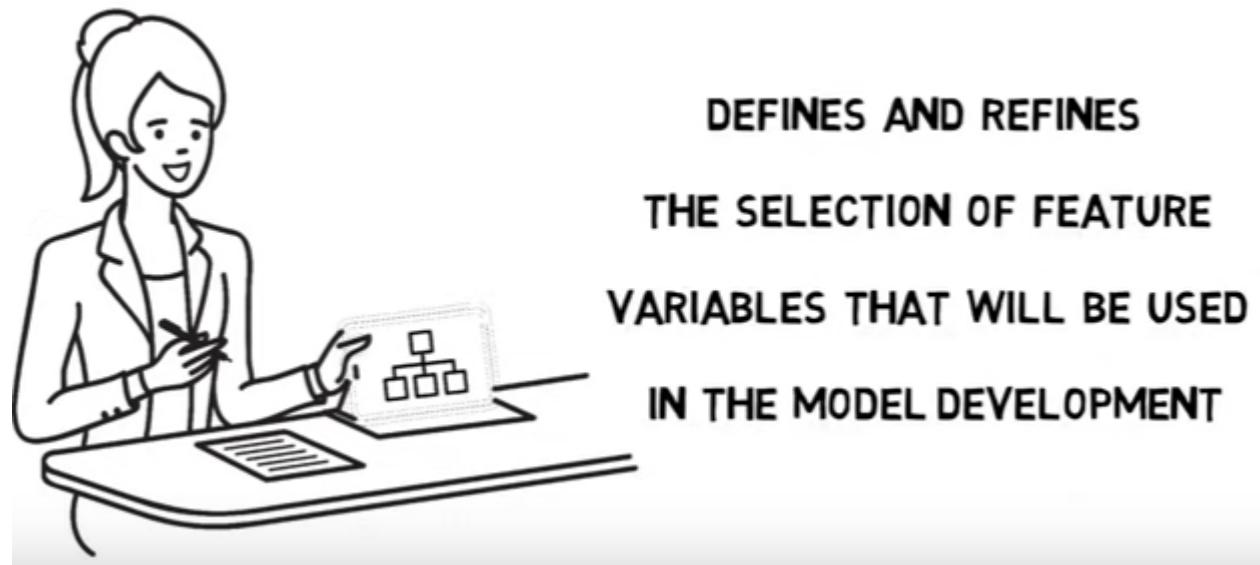
Data Science Life Cycle

- **Step 3: Data Preparation**
 - Since raw data may not be usable, data preparation is the most crucial aspect of the data science lifecycle.
 - A data scientist must first examine the data to identify any gaps or data that do not add any value. During this process, you must go through several steps, including:
 - **Data Integration:** Resolve any conflicts in the dataset and eliminate redundancies
 - **Data Transformation:** Normalize, transform and aggregate data using ETL (extract, transform, load) methods
 - **Data Reduction:** Using various strategies, reduce the size of data without impacting the quality or outcome
 - **Data Cleaning:** Correct inconsistent data by filling out missing values and smoothing out noisy data

Data Science Life Cycle

- **Step 4: Exploratory Data Analysis**

- it's important to analyze the data. It is the most exciting step as it helps you to build familiarity with the data and extract useful insights.
- If you skip this step then you might end up generating inaccurate models and choosing the insignificant variables in your model.



Data Science Life Cycle

- **Step 4: Exploratory Data Analysis**
 - A Data Scientist analyzes the data through various statistical procedures. In particular, two types of procedures used are:
 - Descriptive Statistics
 - Inferential Statistics
 - Assume that you are a Data Scientist working for a company that manufactures cell phones. You have to analyze customers using the mobile phones of your company. In order to do so, you will first take a thorough look at the data and understand various trends and patterns involved.
 - In the end, you will summarize the data and present it in the form of a graph or a chart. You therefore, apply **Descriptive Statistics** to solve the problem.

Data Science Life Cycle

- **Step 4: Exploratory Data Analysis**
 - You will then draw ‘inferences’ or conclusions from the data. To understand inferential statistics through the following example – Assume that you wish to find out a number of defects that occurred during manufacturing.
 - However, individual testing of mobile phones can take time. Therefore, you will consider a sample of the given phones and make a generalization about the number of defective phones in the total sample.

Data Science Life Cycle

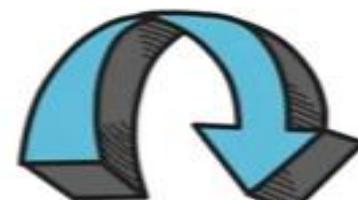
- **Step 5: Data Modeling**

- Modeling means formulating every step and gather the techniques required to achieve the solution.
- The important factor is how to perform the calculations. There are various techniques under Statistics and Machine Learning that you can choose based on the requirement.



IDENTIFY THE MODEL
THAT BEST FITS THE
BUSINESS REQUIREMENT

TRAIN THE MODELS ON THE
TRAINING DATASET AND TEST



SELECT THE BEST
PERFORMING MODEL



python™



Data Science Life Cycle

- **Step 6: Communicate Results:** Data reporting, Data visualization, Business Intelligence, Decision making
 - This is the final step where you present the results from your analysis to the stakeholders. You explain to them how you came to a specific conclusion and your critical findings.
 - Uses tools like tableau, Power BI, QlikView to create powerful reports and dashboards

Data Science Life Cycle

- **Step 7: Deploys and maintains**

- This is the final step where you present the results from your analysis to the stakeholders. You explain to them how you came to a specific conclusion and your critical findings.
- Test the selective model in a pre production environment before deploying in production environment.
- After successfully deployment, uses reports and dashboards to get real time analytics.
- Further monitor and maintain project performance

Data Science Use Cases

- Big companies are using data science for different purposes.



Data Science Use Cases

- **Facebook – Using Data to Revolutionize Social Networking & Advertising**
 - Facebook using advanced techniques like deep learning in data science to study user behaviour and gain insights to improve their product.
 - Using deep learning, Facebook makes use of facial recognition and text analysis.
 - In facial recognition, Facebook uses powerful neural networks to classify faces in the photographs. It uses its own text understanding engine called “DeepText” to understand user sentences.
 - It also uses Deep Text to understand people’s interest and aligning photographs with texts.

Data Science Use Cases

- **Facebook – Using Data to Revolutionize Social Networking & Advertising**
 - Facebook uses deep learning for targeted advertising. Using this, it decides what kind of advertisements the users should view.
 - It uses the insights gained from the data to cluster users based on their preferences and provides them with the advertisements that appeal to them.

Data Science Use Cases

- **Uber – Using Data to Make Rides Better**

- Uber is a popular smartphone application that allows you to book a cab. Uber makes extensive use of Big Data because Uber has to maintain a large database of drivers, customers, and several other records.
- Uber contains a database of drivers. Therefore, whenever you hail for a cab, Uber matches your profile with the most suitable driver. What differentiates Uber from other cab companies is that Uber charges you based on the time it takes to cover the distance and not the distance itself.
- It calculates the time taken through various algorithms that also make use of data related to traffic density and weather conditions.

Data Science Use Cases

- **Uber – Using Data to Make Rides Better**
 - Uber makes the best use of data science to calculate its surge pricing. When there are less drivers available to more riders, the price of the ride goes up. This happens only during the scarcity of drivers in any given area.
 - However, if the demand for Uber rides is less, then Uber charges a lower rate. This dynamic pricing is rooted in Big Data and makes excellent usage of data science to calculate the fares based on the parameters.

Data Science Use Cases

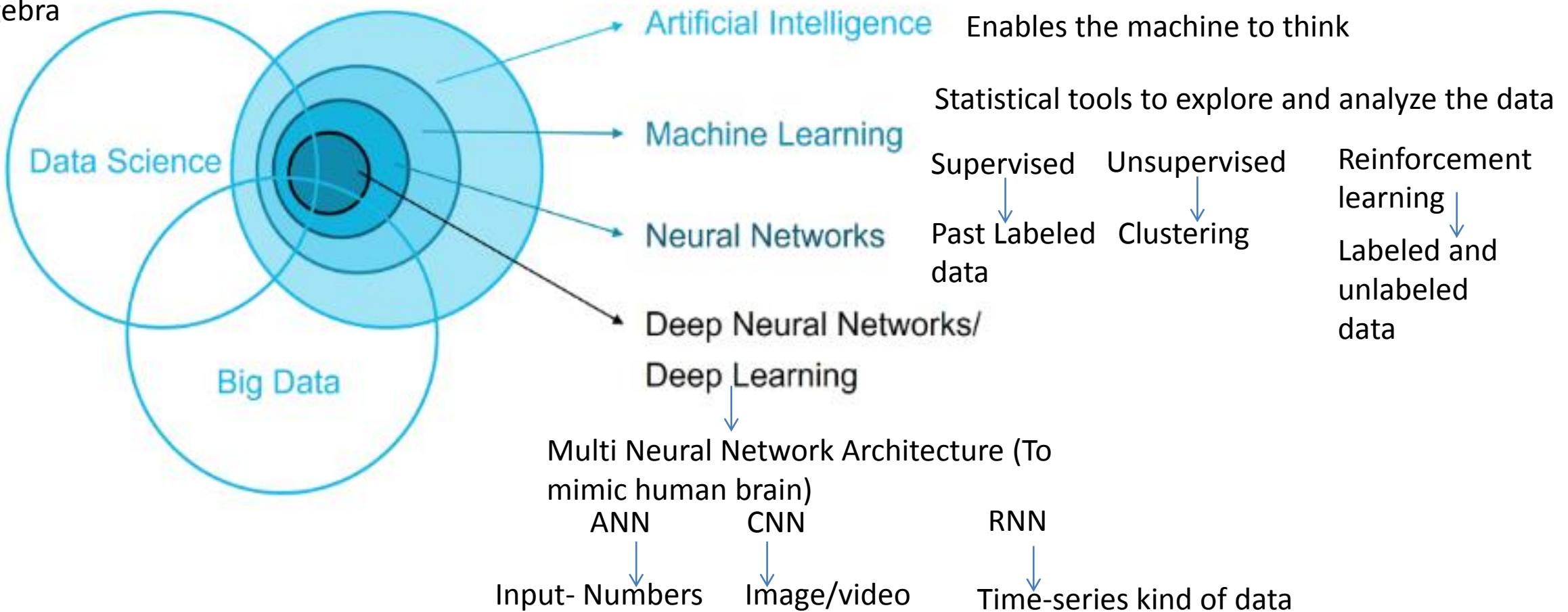
- **Bank of America – Using Data to Leverage Customer Experience**
 - Using data science, banking industries are able to detect frauds in payments and customer information. It also prevents frauds regarding insurances, credit cards, and accounting.
 - In order to minimize the losses, a bank needs to detect fraud sooner. In order to carry this out, banks employ data scientists to use their quantitative knowledge where they apply algorithms like association, clustering, forecasting, and classification.

Data Science Use Cases

- **Spotify – Revolutionizing Music Streaming**
 - It is an online music streaming giant that uses Data Science for providing personalized music recommendations
 - Spotify is a data-driven company that leverages big data to provide personalized playlists to its users.
 - In the year 2017, Spotify used data science to gain insights about which universities had the highest percentage of party playlists and which ones spent the most time on it. It publishes its findings on their page “Spotify Insights” to provide information about the on-going trends in the music.

AI vs ML vs DL vs Data science

Uses tools like Statistics, probability,
linier algebra



Tools and Libraries

- Programming language
 - R
 - Python
 - Java
- Machine learning Algorithms
 - Classification and clustering
 - Regression
 - Reinforcement
 - Deep learning
 - Dimensionality reduction
- IDE (Integrated development environment)
 - Pycharm
 - Jupyter
 - Spyder
 - R Studio

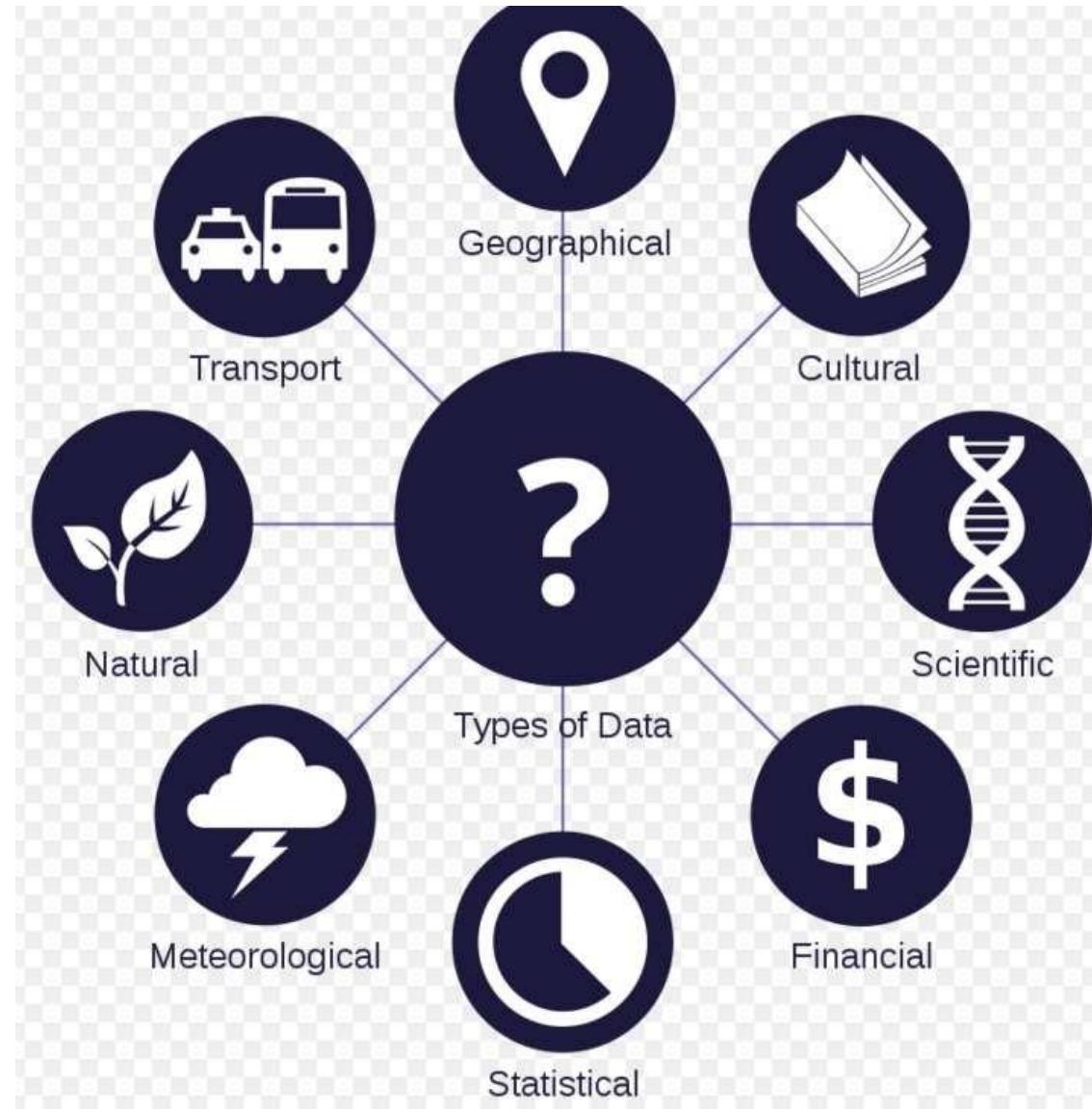
Tools and Libraries

- Web Scraping
 - Beautiful soup library
 - Scrapy tool
 - URLLib library
- Math
 - Statistics
 - Linear Algebra
 - Differential Calculus
- Data Visualization tools
 - Tableau
 - Power BI
 - Matplotlib library
- Data analysis
 - Feature engineering
 - Data Wrangling
 - Exploratory data analysis

Thank you.

Data and Types of Data in Data Science

Data



What is Data?

- Data are the actual pieces of information that you collect through your study and used for the purpose of analysis.
- Data refers to a set of values, which are usually organized by variables
- It is the raw information from which statistics are created.
- For example, if you ask five of your friends how many pets they own, they might give you the following data: 0, 2, 1, 4, 18.
- Not all data are numbers; let's say you also record the gender of each of your friends, getting the following data: male, male, female, male, female.
- Most data fall into one of two groups OR Variables are of different types and can be classified in many ways: numerical or categorical

Data Objects

- Data sets are made up of data objects.
- A **data object** represents an entity.
- Examples:
 - sales database: customers, store items, sales
 - medical database: patients, treatments
 - university database: students, professors, courses
- Also called *samples* , *examples*, *instances*, *data points*, *objects*, *tuples*.
- Data objects are described by **attributes**.
- Database rows -> data objects; columns -> attributes.

Attributes

- **Attribute (or dimensions, features, variables)**: a data field, representing a characteristic or feature of a data object.
 - *E.g., customer_ID, name, address*
- Types:
 - Numerical
 - Categorical

Numerical Data

- Numerical data is information that is measurable, and it is, of course, data represented as numbers and not words or text.
- These data have meaning as a measurement, such as a person's height, weight, IQ, or blood pressure; or they're a count, such as the number of stock shares a person owns, how many teeth a dog has, or how many pages you can read of your favourite book before you fall asleep.
- Numerical data can be further broken into two types:
 - Discrete
 - Continuous.

Numerical Data

- **Discrete Data:** Discrete data is a numerical type of data that includes whole, concrete numbers with specific and fixed data values determined by counting.
 - They have a logical end to them.
 - Discrete data represent items that can be counted
 - Finite number of possible values
 - They take on possible values that can be listed out. The list of possible values may be fixed (also called finite); or it may go from 0, 1, 2, on to infinity.
 - Examples: number of people in a room, number of items in a basket, number of hours in a day, money

Numerical Data

- **Continuous Data:** Continuous data includes complex numbers and varying data values that are measured over a specific time interval.
 - Numbers that don't have a logical end to them.
 - Continuous data represent measurements; their possible values cannot be counted and can only be described using intervals on the real number line.
 - Infinite number of possible values.
 - For example, the exact amount of gas purchased at the pump for cars with 20-gallon tanks would be continuous data from 0 gallons to 20 gallons, represented by the interval $[0, 20]$, inclusive. You might pump 8.40 gallons, or 8.41, or 8.414863 gallons, or any possible number from 0 to 20.

Categorical Data

- Categorical data, this is any data that isn't a number, which can mean a string of text or date.
- It describes an event using a string of words rather than numbers.
- Categorical data represent characteristics such as a person's gender, marital status, hometown, or the types of movies they like. Categorical data can take on numerical values (such as "1" indicating male and "2" indicating female), but those numbers don't have mathematical meaning.
- These variables can be broken down into nominal, ordinal values.

Nominal Data

- There is no order at all: each category has its unique meaning
- We can only count but can not order or measure nominal data
- A nominal scale describes a variable with categories that do not have a natural order or ranking
- Examples: Names of cars, book titles in a library, marital status



Ordinal Data

- If there is a sense of order there, the variables are called ordinal.
- An ordinal scale is one where the **order matters** but **not the difference between values**.
- Examples of ordinal variables include:
 - socio economic status (“low income”, “middle income”, “high income”),
 - education level (“high school”, “BS”, “MS”, “PhD”),
 - income level (“less than 50K”, “50K-100K”, “over 100K”),
 - satisfaction rating (“extremely dislike”, “dislike”, “neutral”, “like”, “extremely like”).

Ordinal Data

- Note the differences between adjacent categories do not necessarily have the same meaning. For example, the difference between the two income levels “less than 50K” and “50K-100K” does not have the same meaning as the difference between the two income levels “50K-100K” and “over 100K”.

		MEDALS		
	COUNTRY	GOLD	SILVER	BRONZE
1	GER	13	11	9
2	NOR	11	5	10
3	CAN	10	10	5
4	USA	9	7	12
5	GER	8	6	5
6	NED	8	7	9
7	SUI	6	3	2
8	BLR	3	0	1
9	AUT	4	8	5
10	FRA	4	4	7

Binary Data

- Binary data has only two possible states:
 - Yes or No
 - 0 or 1
 - True or False
- Binary data is used heavily for classification machine learning models.
- Examples of binary variables can include whether a person has stopped their subscription service or not, or if a person bought a car or not, Toss of a coin, Switch On or Off

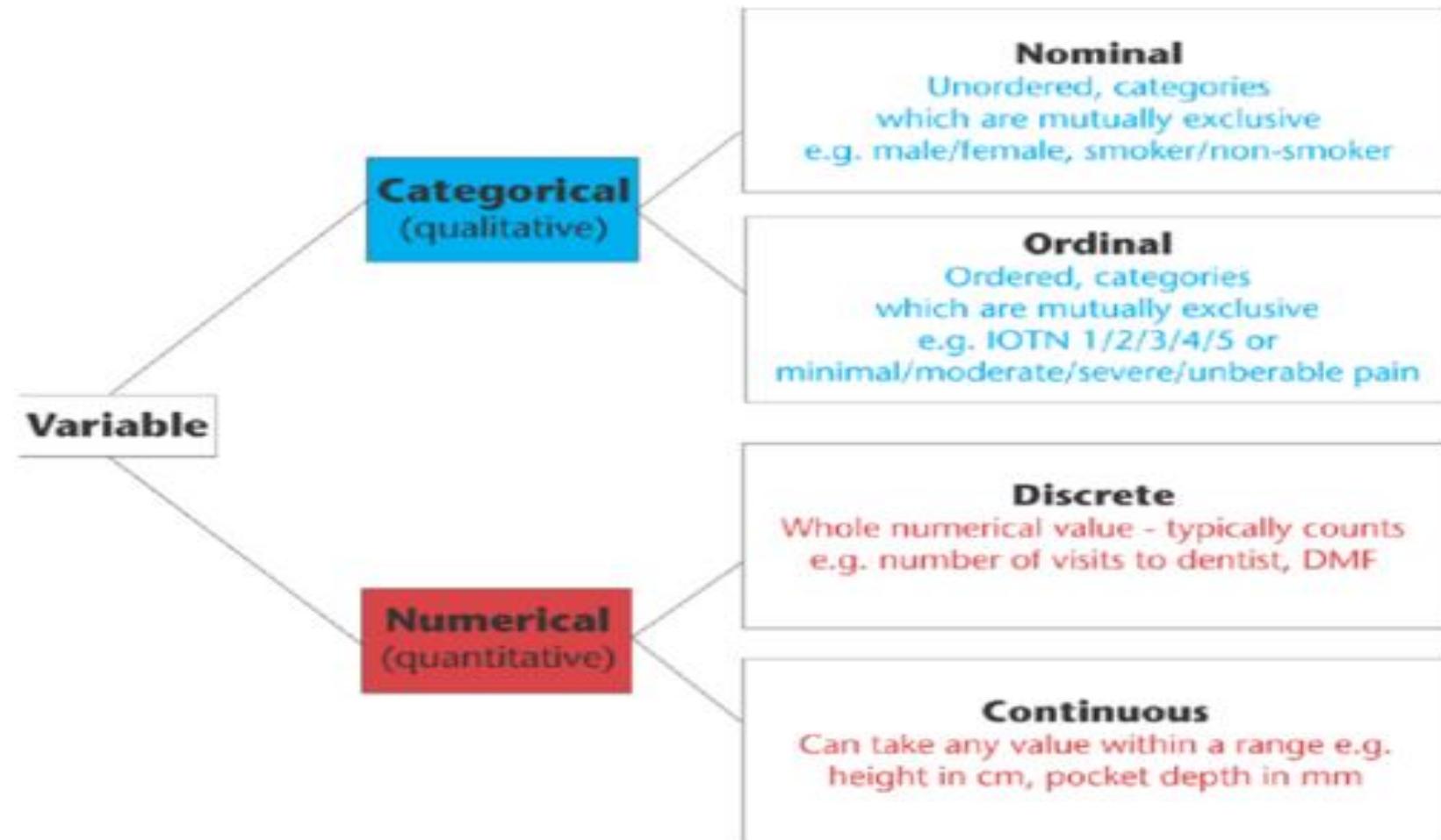
Interval Data

- An interval scale is one where there is **order and the difference between two values is meaningful.**
- Examples of interval variables include:
 - The temperature measured in Celsius, the difference in temperature between 50-60 degrees is the same as the difference in temperature between 80-90 degrees.
 - pH, SAT score (200-800), credit score (300-850), year.
- Interval data doesn't have 'absolute zero' value. The zero points, in this case, are arbitrary as there can be a temperature below zero degrees Celsius.
- The mathematical operations such as addition, subtraction, mean, median, mode and standard deviation can be calculated.

Ratio Variable

- A ratio variable, has all the properties of an interval variable, and also has a clear definition of 0.0. When the variable equals 0.0, there is none of that variable.
- It has an absolute zero value.
- For example, if you measure temperature in degrees Kelvin then it is considered ratio data. This is because zero points are absolute as there can't be temperatures below zero degrees Kelvin. Ratio data doesn't have any negative numerical value. For example, height can't be negative.
- Other Examples: enzyme activity, dose amount, reaction rate, flow rate, concentration, pulse, weight, length, survival time.

Qualitative (Categorical) vs Quantitative (Numerical)



Qualitative (Categorical) vs Quantitative (Numerical)

- There are other ways of classifying variables that are common in statistics. One is qualitative vs. quantitative.
- Qualitative variables are descriptive/categorical. Qualitative data is data concerned with descriptions, which can be observed but cannot be computed. Many statistics, such as mean and standard deviation, do not make sense to compute with qualitative variables.
- Quantitative variables have numeric meaning, so statistics like means and standard deviations make sense.

Qualitative Data

- Qualitative data is defined as the data that approximates and characterizes.
- Qualitative data can be observed and recorded.
- This data type is non-numerical in nature.
- This type of data is collected through methods of observations, one-to-one interviews, conducting focus groups, and similar methods.
- Qualitative data in statistics is also known as categorical data – data that can be arranged categorically based on the attributes and properties of a thing or a phenomenon.

Qualitative Data

- For example, think of a student reading a paragraph from a book during one of the class sessions. A teacher who is listening to the reading gives feedback on how the child read that paragraph. If the teacher gives feedback based on fluency, intonation, throw of words, clarity in pronunciation without giving a grade to the child, this is considered as an example of qualitative data.
- The cake is orange, blue, and black in colour
- Females have brown, black, and red hair
- Qualitative data does not include numbers in its definition of traits

Quantitative Data

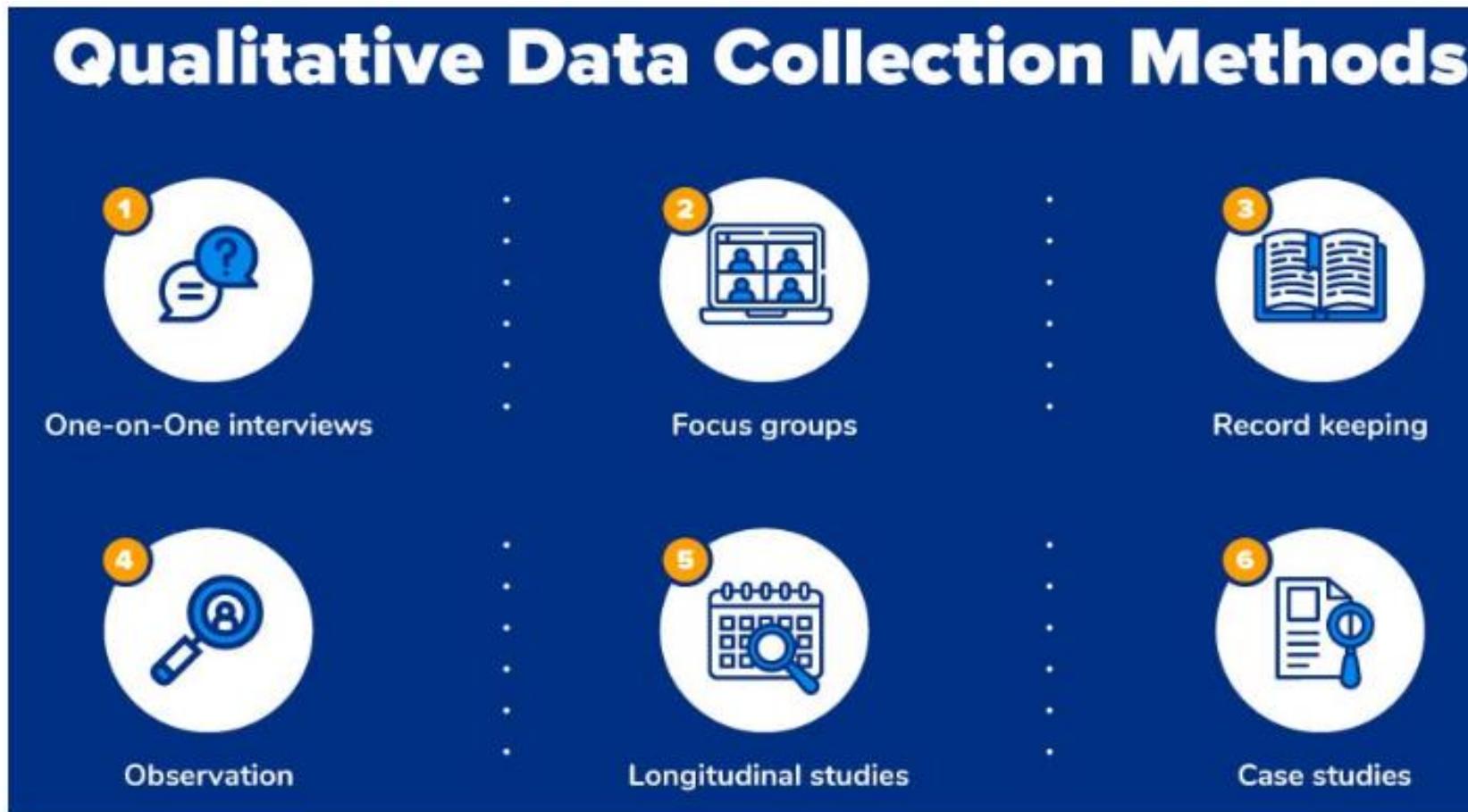
- Quantitative data is all about numbers.
 - Quantitative data is any quantifiable information that can be used for mathematical calculation or statistical analysis.
 - This form of data helps in making real-life decisions based on mathematical derivations.
 - Quantitative data is used to answer questions like how many? How often? How much? This data can be validated and verified.
- Example: There are four cakes and three muffins kept in the basket
- Example: One glass of fizzy drink has 97.5 calories

Key Differences (Quantitative vs Qualitative Data)

Quantitative Data	Qualitative Data
These are data that deal with quantities, values, or numbers.	These data, on the other hand, deals with quality.
Measurable.	They are generally not measurable.
Expressed in numerical form.	They are descriptive rather than numerical in nature.
Conclusive	Exploratory
Measures quantities such as length, size, amount, price, and even duration.	Narratives often make use of adjectives and other descriptive words to refer to data on appearance, color, texture, and other qualities.
Approach is Objective	Approach is Subjective
Determines Level of occurrence	Determines Depth of understanding
Data Collection Techniques: Quantitative surveys, Interviews, Experiments	Data Collection Techniques: Qualitative Survey, Focus group méthode, Documental revision, etc.
Data Structure: Structured	Data Structure: Unstructured

Qualitative Data Collection Methods

- Qualitative Data Collection Methods



Qualitative Data Collection Methods

- Exploratory in nature, these methods are mainly concerned at gaining insights and understanding of underlying reasons and motivations, so they tend to dig deeper.
- Since they cannot be quantified, measurability becomes an issue.
- This lack of measurability leads to the preference for methods or tools that are largely unstructured or, in some cases, maybe structured but only to a very small, limited extent.
- Generally, qualitative methods are **time-consuming and expensive** to conduct, and so researchers try to lower the costs incurred by decreasing the sample size or number of respondents.

Quantitative Data Collection Methods

- Qualitative Data Collection Methods



Quantitative Data Collection Methods

- Data can be readily quantified and generated into numerical form, which will then be converted and processed into useful information mathematically.
- The result is often in the form of statistics that is meaningful and, therefore, useful.
- Unlike qualitative methods, these quantitative techniques usually make use of larger sample sizes because its measurable nature makes that possible and easier.
- Example: Administrative data collection->financial data, performance data, resource allocation etc.

Forms of Data

Structured

- Data that has been **organized into a formatted** repository,
- Eg: database, table ,etc.

Semi- Structured

- They have **some organizational properties** that make it easier to analyze (self describing data)
- Eg. XML, JSON, NoSQL

Un Structured

- Data with no structure
- Eg. Images, videos, etc.

Structured Data

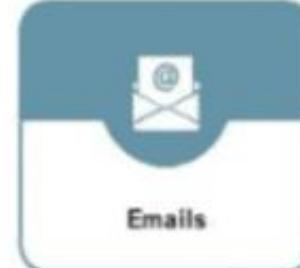
- **Structured data** is generally tabular data that is represented by columns and rows in a database.
- Databases that hold tables in this form are called *relational databases*.
- The mathematical term “*relation*” specify to a formed set of data held as a table.
- In structured data, all row in a table has the same set of columns.
- SQL (Structured Query Language) programming language used for structured data.

Semi-structured Data

- Semi-structured data is information that doesn't consist of Structured data (relational database) but still has some structure to it.
- Semi-structured data consist of documents held in JavaScript Object Notation (JSON) format. It also includes key-value stores and graph databases.

Unstructured Data

- **Unstructured data** is information that either does not organize in a pre-defined manner or not have a pre-defined data model.
- Unstructured information is a set of text-heavy but may contain data such as numbers, dates, and facts as well.
- **Videos, audio, and binary** data files might not have a specific structure. They're assigned to as **unstructured data**.



Acquire Data

Data Acquisition

- What is data acquisition or data collection?
- What kind of data do we need?
- Where you can find these data sets?
- What are some common methods of accessing this data?

Data Acquisition

- Data collection is a systematic approach to gather relevant information from a variety of sources.
- Depending on the problem statements, the data collection method is broadly classified into two categories.

- Primary data collection
- Secondary data collection



Sample Data

- For Example, we have collected and aggregated the data from various open-source websites such as Github, Kaggle, and datahub. A snapshot of the data collected is shown on the screen.

Player Name	Age	Club	Height	Weight	Foot	Joined
Pierre-Emerick Aubameyang	29	Arsenal	6'2"	176lbs	Right	Jan 31, 2018
Alexandre Lacazette	27	Arsenal	5'9"	161lbs	Right	Jul 5, 2017
Bernd Leno		Arsenal	6'3"	183lbs	Right	Jul 1, 2018
Henrikh Mkhitaryan	29	Arsenal	5'10"	165lbs	Right	Jan 22, 2018
Granit Xhaka	25	Arsenal	6'1"	181lbs	Left	Jul 1, 2016
Shkodran Mustafi	26	Arsenal	6'0"	181lbs	Right	Aug 30, 2016
Jack Grealish	22	Aston Villa	5'9"	150lbs	Right	Mar 1, 2012
John McGinn	23	Aston Villa	5'10"	150lbs	Left	Aug 8, 2018
Anwar El Ghazi	23	Aston Villa	6'2"	550lbs	Right	Jan 31, 2017
Conor Hourihane	27	Aston Villa	5'11"	137lbs	Left	Jan 26, 2017
James Chester	29	Aston Villa	5'11"	174lbs		Aug 12, 2016
James Chester	29	Aston Villa	5'11"	174lbs		Aug 12, 2016
James Chester	29	Aston Villa	5'11"	174lbs		Aug 12, 2016
James Chester	29	Aston Villa	5'11"	174lbs		Aug 12, 2016
Jonathan Kodjia	2	Aston Villa	6'2"	170lbs	Right	Aug 30, 2016
Callum Wilson	26		5'11"	146lbs	Right	Jul 4, 2014

 Quantitative Data

 Qualitative Data

Sources of Data

User generated

- Blogs
- Documents

System/Application generated

- Web logs
- Network event logs

Device generated

- Surveillance cameras capturing traffic patterns
- Point Of Sale (POS) system

Internal

- Generated internally in an organization across business processes; Sales data, logistics data, finance data, HR data, and so on

External

- Generated by external bodies or data aggregators or credit bureaus

Freely available data sources

- There are many sources of data available at no cost
 - Some are public domain and some are copyrighted
 - Be sure to check the license to verify that your use is allowed

U.S. Census Bureau	http://factfinder2.census.gov/
U.S. Executive Branch	http://www.data.gov/
U.K. Government	http://data.gov.uk/
E.U. Government	http://publicdata.eu/
The World Bank	http://data.worldbank.org/
Freebase	http://www.freebase.com/
Wikidata	http://meta.wikimedia.org/wiki/Wikidata
Amazon Web Services	http://aws.amazon.com/datasets
InfoChimps *	http://www.infochimps.com/marketplace

Commercial data sources

- Many companies also offer data
 - Usually for a fee, but sometimes available at no cost
 - Always be sure to check the license terms

Gnip	Social Media	http://gnip.com/
AC Nielsen	Media Usage	http://www.nielsen.com/
Rapleaf	Demographic	http://www.rapleaf.com/
ESRI	Geographic (GIS)	http://www.esri.com/
eBay	Auction	https://developer.ebay.com/
D&B	Business Entity	http://www.dnb.com/

Acquisition techniques

- Data, comes from, many places, local and remote, in many varieties, structured and un-structured. And, with different velocities.
- There are many techniques and technologies to access these different types of data.
- For examples: A lot of data exists in conventional **relational databases**, like structure big data from organizations.
- The **tool** of choice to access data from databases is structured query language or **SQL**, which is supported by all relational databases management systems.
- Data can also exist in files such as text files and Excel spread sheets. **Scripting languages** are generally used to get data from files.

Acquisition techniques

- For extracting data from websites; we can use web scraping tools like Scrapy, Scraper API, ParseHub, Webhouse.io, Content Grabber, Common crawl etc.
- NoSQL storage systems are increasingly used to manage a variety of data types in big data. These data stores are databases that do not represent data in a table format with columns and rows. NoSQL data stores provide APIs to allow users to access data. These APIs can be used directly or in an application that needs to access the data

Data Acquisition

- Example: WIFIRE case study as a real project that acquires data using several different mechanisms. The WIFIRE project stores sensor data from weather stations in a relational database. We use SQL to retrieve this data from the database to create models to identify weather patterns associated with Santa Ana conditions.
- To determine whether a particular weather station is currently experiencing Santa Ana conditions, we access real time data using a **websocket** service. Once we start listening to this service, we **receive weather station measurements** as they occur. This data is then processed and compared to patterns found by our models to determine if a weather station is experiencing Santa Ana conditions.

Data Acquisition

- At the same time Tweets are retrieved using hashtags related to any fire that is occurring in the region. The **Tweet messages** are retrieves using the **Twitter REST service**. The idea is to **determine the sentiment of these tweets** to see if people are expressing fear, anger or are simply nonchalant about the nearby fire.
- The combination of **sensor data and tweet sentiments** helps to give us a sense of the **urgency of the fire situation**. As a summary, big data and data science comes from many places. Finding and evaluating data useful to your big data analytics is important before you start acquiring data. Depending on the source and structure of data, there are alternative ways to access it.

Data Pre-processing

Data Pre processing

- Data Pre processing is a technique that is used to convert the raw data into a clean data set. In other words, whenever the data is gathered from different sources it is collected in raw format which is not feasible for the analysis.
- Therefore, certain steps are executed to convert the data into a small clean data set. This technique is performed before the execution of the Iterative Analysis. The set of steps is known as Data Pre processing. It includes –
 - Data Cleaning
 - Data Integration
 - Data Transformation
 - Data Reduction

Data Pre processing

- **Tasks of Data Preparation**

1. **Data Cleaning**

➤ This is the first step which is implemented in Data Preprocessing. In this step, the primary focus is on handling missing data, noisy data, detection, and removal of outliers minimizing duplication, and computed biases within the data.

2. **Data Integration**

➤ This process is used when data is gathered from various data sources and data are combined to form consistent data. This consistent data after performing data cleaning is used for Data Preparation and analysis.

Data Pre processing

- **Tasks of Data Preparation**

3. Data Transformation

- This step is used to convert the raw data into a specified format according to the need of the model. The options for the transformation of data are given below -
- **Normalization** - In this method, numerical data is converted into the specified range, i.e., between 0 and one so that scaling of data can be performed.
- **Aggregation** - The concept can be derived from the word itself, this method is used to combine the features into one. For example, combining two categories can be used to form a new group.

Data Pre processing

- **Tasks of Data Preparation**
- **Generalization** - In this case, lower level attributes are converted to a higher standard.

4. Data Reduction

- After the transformation and scaling of data duplication, i.e., redundancy within the data is removed and efficiently organize the data during Data Preparation.

Data Quality

- Importance of good quality data
- Factors that cause data quality issues
 - Data Quality Remediation

Importance of good quality data

- After collecting the data, most people start the analysis on it. Often, they forget to do a sanity check on the data. If the data is of bad quality, it can give misleading information.
- For example, if you start the analysis without ensuring data quality then you might get unexpected results such as the Crystal Palace club will win the next EPL. However, your domain knowledge on EPL says that the result looks inaccurate as Crystal Palace has never even finished in the top 4.
- A surprising fact is that a professional data scientist spends approximately 60% of his time ensuring that data is of high quality.

Importance of good quality data

- Example

Player Name	Age	Club	Height	Weight	Foot	Joined
Pierre-Emerick Aubameyang	29	Arsenal	6'2"	176lbs	Right	Jan 31, 2018
Alexandre Lacazette	27	Arsenal	5'9"	161lbs	Right	Jul 5, 2017
Bernd Leno		Arsenal	6'3"	183lbs	Right	Jul 1, 2018
Henrikh Mkhitaryan	29	Arsenal	5'10"	165lbs	Right	Jan 22, 2018
Granit Xhaka	25	Arsenal	6'1"	181lbs	Left	Jul 1, 2016
Shkodran Mustafi	26	Arsenal	6'0"	181lbs	Right	Aug 30, 2016
Jack Grealish	22	Aston Villa	5'9"	150lbs	Right	Mar 1, 2012
John McGinn	23	Aston Villa	5'10"	150lbs	Left	Aug 8, 2018
Anwar El Ghazi	23	Aston Villa	5'2"	550lbs	Right	Jan 31, 2017
Conor Hourihane	27	Aston Villa	5'11"	137lbs	Left	Jan 26, 2017
James Chester	29	Aston Villa	5'7"	174lbs		Aug 12, 2016
James Chester	29	Aston Villa	5'7"	174lbs		Aug 12, 2016
James Chester	29	Aston Villa	5'11"	174lbs		Aug 12, 2016
James Chester	29	Aston Villa	5'7"	174lbs		Aug 12, 2016
Jonathan Kodjia	2	Aston Villa	5'6"	170lbs	Right	Aug 30, 2016
Callum Wilson	26	Crystal Palace	5'11"	146lbs	Right	Jul 4, 2014

Bad quality data



Misleading information

Raw Data

- We collect data from the source like twitter, blogs, websites etc.
- Raw data may:
 - Have errors
 - Not validated
 - Multiple forms
 - Unformatted
 - Requiring confirmation or citation
- Input to the data processing process
- Raw data Example: If the correct format is not specified in an application form, the date of birth data can take many forms, such as “ 31st July 2021”, “31/07/2021”, “31/07/21” and “31 July 21”,
- This raw data needs to be processed to a common format for further use by systems/ humans.

Why does the data quality issue occur?

- **Need of Data Preparation Process and Pre processing**
 - Some specified Machine Learning and Deep Learning model need information in a specified format, for example, Random Forest algorithm does not support null values, therefore to execute random forest algorithm null values has to be managed from the original raw data set.
 - Another aspect of Data Preparation and analysis is that the data set should be formatted in such a way that more than one Machine Learning and Deep Learning algorithms are executed in one data set, and the best out of them is chosen.

Why does the data quality issue occur?

- Improper Data Collection

Company	Employee Name	Age	Time Spent (hours)
Apple	John S.	23	100
Apple	Evan B.	27	8
Apple	Emily B.	31	12
Google	Ava W.		7
Google	Noah A.	34	9



Incorrect
measurement



Incorrect
time



Incomplete
data

Why does the data quality issue occur?

- Improper Data Integration

Player Name	Team	Weight (lbs.)
P. Bardsley	Chelsea	150
D. McNeil	Chelsea	198
Adam Legzdins	Chelsea	170
Dan Agyei	Chelsea	168
David Luiz	Chelsea	192

Source: X (in lbs.)

Player Name	Team	Weight (kgs.)
Jamal Blackman	Chelsea	72
Ethan Ampadu	Chelsea	68
Billy Gilmour	Chelsea	73
Ike Ugbo	Chelsea	64.5
George McEachran	Chelsea	75

Source: Y (in kgs.)

Player Name	Team	Weight (lbs.)
P. Bardsley	Chelsea	150
D. McNeil	Chelsea	198
Adam Legzdins	Chelsea	170
Dan Agyei	Chelsea	168
David Luiz	Chelsea	192
Jamal Blackman	Chelsea	72
Ethan Ampadu	Chelsea	68
Billy Gilmour	Chelsea	73
Ike Ugbo	Chelsea	64.5
George McEachran	Chelsea	75

Factors that cause data quality issues

- It can occur during data collection or data integration. Consider that you are recording the average time that employees spend in a cafeteria weekly across companies. You recorded 100 hours instead of 10 hours, or the unit of **measurement recorded incorrectly**. Also, if you are interviewing and someone may choose not to respond to certain questions which leads to **missing values**
- Another cause is when data is **collected from different sources and merged**. For example, you require the weight of all players of EPL in a single file. You extract the player's weight data from source X. But the weight data of some new players is not available, so you get it from source Y. The unit of weight measurement in source X is in pounds, and source y is in kilograms. If data collected from both sources are combined as it is, then there will be inconsistency in the data, and it will result in **inaccurate data**.

Factors that cause data quality issues

- Inconsistent data is when data fails to match. Let's say, the user entered birthday to be May 07, 1993 and the age attribute displays 50. Or over time the ratings of a movie have changed from the numeric rating 1, 2, 3 to alphabets — A, B, C. Thus, in the same column data is not consistent.

Student ID	Student Name	Age	GPA	Classification
100122014	Joseph	21	3.5	Junior
100232015	Patrick	200	3.2	Sophomore
100122012	Seller	24	3.0	Senior
100342013	Roger	23	234	Senior
100942012	Davis	2.8	3.7	Sophomore
	Travis	23	3.4	Sr
100982015	Alex	27		Sophomore
100982013	Trevor	-22	4.0	Senior
AUC2016XC	Aman	30	3.5	Jr

Missing Data

Inconsistent Data

Noisy Data

Types of data quality issues

- The common data quality issues that are easy to spot are **missing values, duplicate values, and inconsistent data**.
- However, some issues are difficult to spot. For example, If you follow EPL, then there is no club with the name of Real Madrid in EPL. From this example, you can infer that, detecting and handling such problems would require domain knowledge.

Player Name	Age	Club	Height	Weight	Foot	Joined
Eden Hazard	27	Chelsea	5'6"	159lbs	Right	Jul 16, 2016
N'Golo Kanté	28	Chelsea	5'10"	168lbs	Right	Aug 24, 2012
César Azpilicueta	23	Chelsea	6'1"	187lbs	Right	Aug 8, 2018
Kepa Arrizabalaga	29	Chelsea	5'9"	172lbs	Right	Aug 28, 2013
Willian	31	Chelsea	6'2"	190lbs	Right	Aug 31, 2016
David Luiz	27	Chelsea	6'2"	192lbs	Left	Aug 31, 2016
Ferland Mendy	23	Real Madrid	5'9"	161lbs	Left	Jun 8, 1995

Types of data quality issues

- **Summary**
- Data Pre processing is necessary because of the presence of unformatted real-world data. Mostly real-world data is composed of –
 - **Inaccurate data (missing data)** - There are many reasons for missing data such as data is not continuously collected, a mistake in data entry, technical problems with biometrics
 - **The presence of noisy data (erroneous data and outliers)** - The reasons for the existence of noisy data could be a technological problem of gadget that gathers data, a human mistake during data entry and much more.
 - **Inconsistent data** - The presence of inconsistencies are due to the reasons such that existence of duplication within data, human data entry, containing mistakes in codes or names, i.e., violation of data constraints and much more necessitate Data Preparation and analysis.

Data Quality Remediation

- Therefore as a data scientist, you should develop a good **understanding** of the **domain**, and the **problem** you are solving.
- **How to fix these data quality issues?**
- Once you identify the inaccurate and missing data, you can use the alternate source of data, if available.
 - For example, if Bernd Leno's age is missing, then you can get it from the **alternative source**, such as Wikipedia. However, this approach is not always possible, as you can't find an alternative source for every data set, or data point.
 - In that case, a simple approach is to **remove the inaccurate data**. This can work well if you have a few inaccurate data points.
 - But, if there are many records with data quality problems, then this approach can reduce the data size, resulting in a poor analysis.

Data Quality Remediation

- A better approach, would be to **replace incorrect or missing values by mean, mode, or median.**
- For example, in this dataset, you can impute the missing weight of Joe Hart, by the **mode of 185**, or **mean of 178.3**, or **median of 178.5**. But, there are chances that the **imputed values are inaccurate**.

Player Name	Age	Club	Height	Weight
Joe Hart	30	Burnley	5'9"	
Steven Defour	26	Burnley	6'2"	203lbs
Chris Wood	28	Burnley	6'1"	185lbs
Ashley Barnes	29	Burnley	5'11"	172lbs
Matthew Lowton	30	Burnley	5'9"	171lbs
Robert Brady	24	Burnley	6'1"	154lbs
Charlie Taylor	26	Burnley	6'0"	185lbs

Data Quality Remediation

- Another approach, is to **estimate the missing weight value**, based on the player whose height and age is similar to Joe Hart.
- For example, Matthew Lowton has the same age and height as Joe Hart, so you can assign 171.

Player Name	Age	Club	Height	Weight
Joe Hart	30	Burnley	5'9"	171lbs
Steven Defour	26	Burnley	6'2"	203lbs
Chris Wood	28	Burnley	6'1"	185lbs
Ashley Barnes	29	Burnley	5'11"	172lbs
Matthew Lowton	30	Burnley	5'9"	171lbs
Robert Brady	24	Burnley	6'1"	154lbs
Charlie Taylor	26	Burnley	6'0"	185lbs

- However, not all values can be estimated from the values of other attributes. Thus, the **approach to remediate the data quality issues depends**, on the **type of data** you are dealing with, and the **domain understanding** of the data.

How is Data Pre processing performed?

- Data Pre processing is carried out to remove the cause of unformatted real-world data which we discussed. Therefore by using these three different steps we can **handled missing data** during Data Preparation–
 - **Ignoring the missing record** - It is the simplest and efficient method for handling the missing data. But, this method should not be performed at the time when the number of missing values is immense or when the pattern of data is related to the unrecognized primary root of the cause of the statement problem.
 - **Filling the missing values manually** - This is one of the best-chosen methods of Data Preparation process. But there is one limitation that when there are large data set, and missing values are significant then, this approach is not efficient as it becomes a time-consuming task.

How is Data Pre processing performed?

- **Filling using computed values** - The missing values can also be occupied by **computing mean, mode or median** of the observed given values. Another method could be the predictive values in Data Preprocessing is that are computed by using any **Machine Learning or Deep learning tools and algorithms**. But one drawback of this approach is that it can generate bias within the data as the **calculated values are not accurate** concerning the observed values.
- **Constant:** You may replace the missing values of a column by using a constant such as “Unknown” or “∞”.

Imputation

- Imputation is the process of replacing unknown values with the best guess.
- Usually, we use the statistical parameters such as mean, median or mode to impute the missing values.
- When to use which parameter is decided by the parameters ability to provide the best central location for the data.

Imputation

- **Mean**
 - Mean is defined as the average value of all the data points.
 - The mean can be used to impute any missing data if the distribution of data is even. If you impute the missing data with the mean, the overall mean of the data remains the same but the standard deviation will reduce.
 - For example, if the data is skewed. By imputing with mean, the distribution of the actual dataset will change completely. This may result in erroneous conclusions.
 - The mean has one more disadvantage. It is very susceptible to the influence of outliers. Outliers are unusually large or small value compared to the rest of the dataset. These values shift the mean towards one side and make it unrepresentative of the dataset. In such cases, it is better to use the median to impute values

Imputation

- **Median**
 - Median can be defined as the central value of a dataset. Unlike the mean, the median value is less affected by outliers or skewed data. In such case a median imputation would be more appropriate than mean imputation.
- **Mode**
 - The mode can be defined as the most frequent value in a dataset. Unlike the mean, the mode is less affected by outliers or skewed data. We can use the mode value to impute the missing data.
 - Since it is the most likely value in a dataset, the distribution pattern doesn't affect the mode value much. The mode is the natural choice for imputation in most cases.

Imputation

- **Mode**
 - In case where there are multiple modes, you have to decide which mode value should be used for imputation.
 - In situations where the data contains a lot of outliers in a particular range then the mode of the data is not close to the most of the data and cannot be treated a representative of the data. In such cases using the median would be a better option.

How is Data Pre processing performed?

- Meaning of **noisy data**
- Noisy Data: Noisy data is a **meaningless data** that can't be interpreted by machines. It can be generated due to faulty data collection, data entry errors etc.
- For example, out of range values like a person filling out the numeric value -679 in the salary field or some negative four digit random number in the age field. Impossible data combinations like — Gender: Male, Pregnant: Yes adds to the noise in the data.
- Noisy data is used interchangeably with the term **corrupt data**.
- The process of removing noise from a data set is termed as **data smoothing**.

How is Data Pre processing performed?

- How we can deal with **noisy data**?
- **Data Binning**
 - Binning is a technique where we sort the data and then partition the data into equal frequency bins. Then you may either replace the noisy data with the bin mean, bin median or the bin boundary.
 - This is a simple example of data binning.

Sorted data for Age: 3, 7, 8, 13, 22, 22, 22, 26, 26, 28, 30, 37

Smoothing the data by equal frequency bins

- **Bin 1:** 3, 7, 8, 13
- **Bin 2:** 22, 22, 22, 26
- **Bin 3:** 26, 28, 30, 37

How is Data Pre processing performed?

- Smoothing by bin means
 - **Bin 1:** 8, 8, 8, 8
 - **Bin 2:** 23, 23, 23, 23
 - **Bin 3:** 30, 30, 30, 30
 - In smoothing by bin means, each value in a bin is replaced by the mean value of the bin.
- Smooth the data by bin boundaries: pick the minimum and maximum value, Put the minimum on the left side and maximum on the right side and Middle values in bin boundaries move to its closest neighbour value with less distance.
 - **Bin 1:** 3, 3, 3, 13
 - **Bin 2:** 22, 22, 22, 26
 - **Bin 3:** 26, 26, 26, 37
 - In smoothing by bin boundaries, each bin value is replaced by the closest boundary value.

How is Data Pre processing performed?

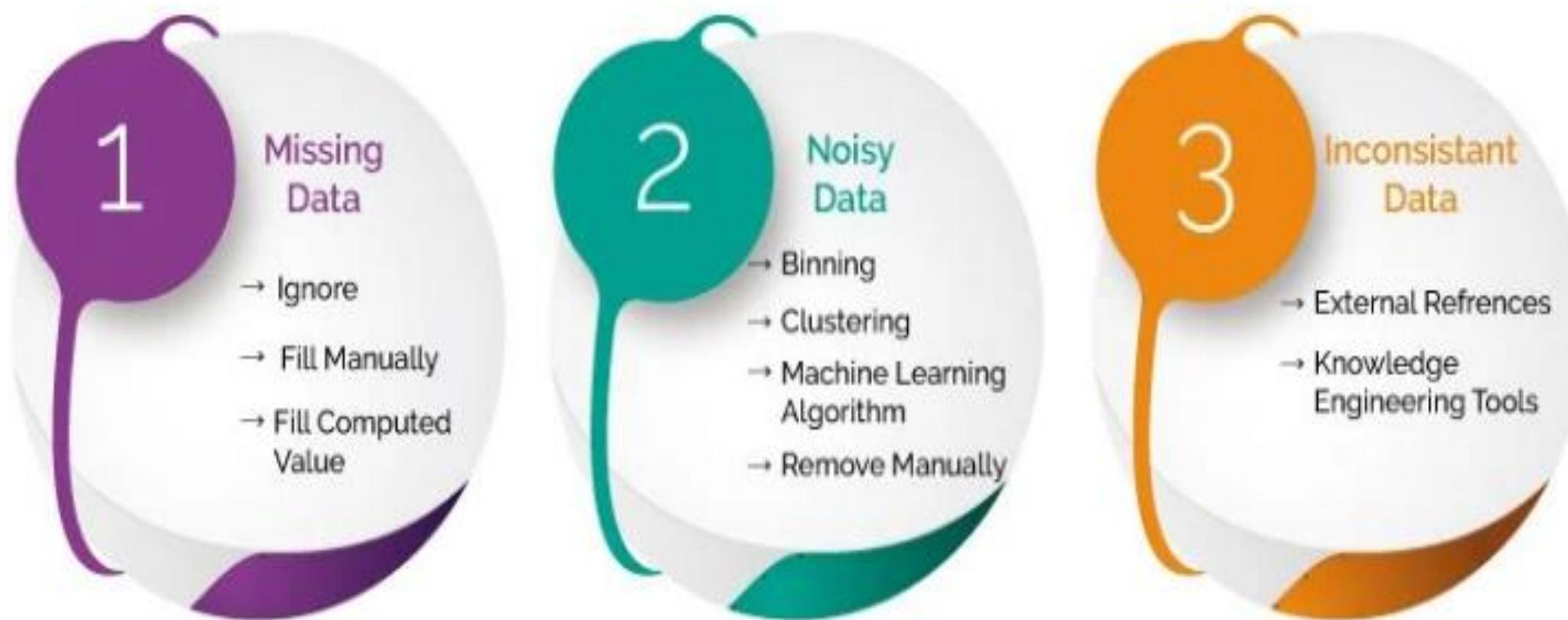
- How we can deal with noisy data?
- **Pre processing in Clustering**
 - In the approach, the outliers may be detected by grouping similar data in the same group, i.e., in the same cluster. Thus, values that fall far apart from the cluster may be considered noise or outliers.
- **Machine Learning**
 - A Machine Learning algorithm can be executed for the smoothing of data during Data Preprocessing . For example, **Regression Algorithm** can be used for the smoothing of data using a specified linear function.

How is Data Pre processing performed?

- How we can deal with noisy data?
- **Removing manually**
 - The noisy data can be deleted manually by the human being, but it is a time-consuming Data Preparation process, so mostly this method is not given priority. To deal with the inconsistent data manually and perform Data Preparation and analysis properly, the data is managed using external references and knowledge engineering tools like the knowledge engineering process.

How is Data Pre processing performed?

- Data Cleaning steps for data pre-processing



Data Integration

- There are numerous tools available in the market that would help us query the data effectively since our data will not integrate itself.
- We have some **Open Source** Data Integration Tools, **Cloud-based** Tools, and also the **On-premises** Data Integration tools.
- The best tool to choose depends on the **requirements, platform, and type of data** that particular business organizations are likely to use.
- List of Common Open-source Tools
 - Apache Airflow
 - CloverETL
 - Talend Open Studio
 - Karma
 - Pentaho
 - Dell Bhoomi AtomSphere

Data Transformation

- Turning the data into an appropriate format for the computer to learn from.
- Example: For research about smog around the globe, you have data about **wind speeds**. However, the data got mixed, and we have three **variants** of figures: **meters per second**, **miles per second**, and **kilometers per hour**. We need to transform these data to the same scale for ML modeling.
- Here are the techniques for data transformation or data scaling:
 - Aggregation
 - Normalization
 - Discretization
 - Concept hierarchy generation
 - Generalization

Data Transformation

- **Aggregation**

- In the case of data aggregation, the data is pooled together and presented in a unified format for data analysis.
- Working with a large amount of high-quality data allows for getting more reliable results from the ML model.
- Example: If we want to build a neural network algorithm that simulates the style of Vincent Van Gogh, we need to provide as many paintings by this famous artist as we can to provide enough material for training. The **images** need to have the **same digital format**, and we will use data transformation techniques to achieve that.

Data Transformation

- **Normalization**

- It helps you to scale the data within a range to avoid building incorrect ML models while training and/or executing data analysis. If the data range is very wide, it will be hard to compare the figures. With various normalization techniques, you can transform the original data linearly, perform decimal scaling or Z-score normalization.
- For example, to compare the population growth of city X (1+ million citizens) to 1 thousand new citizens in city Y, we need to normalize.

Data Transformation

- **Discretization**
 - During discretization, a programmer transforms the data into sets of small intervals. For example, putting people in categories “young”, “middle age”, “senior” rather than working with continuous age values. Discretization helps to improve efficiency.
- **Concept hierarchy generation**
 - If you use the concept hierarchy generation method, you can generate a hierarchy between the attributes where it was not specified. For example, if you have the location information that includes a street, city, province, and country but they have no hierarchical order, this method can help you transform the data.

Data Transformation

- **Generalization**

- With the help of generalization, it is possible to convert low-level data features to high-level data features. For example, house addresses can be generalized to higher-level definitions, such as town or country.

Data Reduction

- When we work with large amounts of data, it becomes harder to come up with reliable solutions. Data reduction can be used to **reduce the amount of data and decrease the costs** of analysis.
- Researchers really need data reduction when working with **verbal speech** datasets. Massive arrays contain individual features of the speakers, for example, interjections and filling words. In this case, huge databases can be decreased to a representative sampling for the analysis.
 - Attribute feature selection
 - Dimensionality reduction
 - Numerosity reduction
 - Data compression

Data Reduction

- **Attribute feature selection**
 - If we construct a **new feature combining the given features** in order to make the data mining process more efficient, it is called an attribute selection.
 - For example, the features male/female and student can be constructed into male student/female student. This can be useful if we conduct research about how many men and/or women are students but their study field doesn't interest us.

Data Reduction

- **Dimensionality reduction**
 - Datasets that are used to solve real-life tasks have a huge number of features. Computer vision, speech generation, translation, and many other tasks cannot sacrifice the speed of operation for the sake of quality. It's possible to use dimensionality reduction to cut the number of features used.
- **Numerosity reduction**
 - Numerosity reduction is a method of data reduction that replaces the original data by a smaller form of data representation. There are two types of numerosity reduction methods –
 - Parametric
 - Non-Parametric.

Data Reduction

- **Parametric Methods**

- Parametric methods use models to represent data. Commonly, regression is used to build such models.

- **Non-parametric methods**

- These techniques allow for storing reduced representations of the data through histograms, data sampling, and data cube aggregation.

- **Data compression**

- Example: Audio/video compression

Data Wrangling

Data Wrangling

- Data Wrangling is a technique that is executed at the time of making an interactive model. In other words, it is used to convert the raw data into the format that is convenient for the consumption of data.
- This technique is also known as **Data Munging**.
- This method also follows certain steps such as after extracting the data from different data sources, **sorting** of data using the certain algorithms are performed, **decompose** the data into a different structured format and finally **store** the data into another database.
- Transforming data into a form that is most appropriate for learning algorithms.

Data Wrangling

- **The need of Data Wrangling**
 - Data Wrangling is an important aspect of implementing the model. Therefore, data is **converted to the proper feasible format** before applying any model to it. By performing filtering, grouping, and selecting appropriate data; accuracy and performance of the model could be increased.
 - Another concept is that when time-series data has to be handled every algorithm is executed with different aspects. Therefore Data Wrangling is used to **convert the time series data into the required format** of the applied model.
 - In simple words, the **complex data is transformed into a usable format** for performing analysis on it.

Data Wrangling

- **Why is Data Wrangling Important?**
- Data Wrangling is used to handle the issue of **Data Leakage** while implementing Machine Learning and Deep Learning.

Data Leakage

- Data Leakage is responsible for the cause of an invalid Machine Learning/Deep Learning model due to the over-optimization of the applied model.
- Data Leakage is the term used when the data from outside, i.e., not part of the training dataset is used for the learning process of the model. This additional learning of information by the applied model will disapprove of the computed estimated performance of the model.
- For example when we want to use the particular feature for performing Predictive Analysis, but that **specific feature is not present** at the time of **training** of dataset then data leakage will be introduced within the model.

Data Leakage

- Data Leakage can be demonstrated in many ways that are given below -
 - The Leakage of data from test dataset to the training data set.
 - Leakage of computed correct prediction to the training dataset.
 - Usage of data outside the scope of the applied algorithm
- In general, the leakage of data is observed from two primary sources of Machine Learning/Deep Learning algorithms such as **feature attributes** (variables) and **training data set**.

Data Leakage

- **Checking the presence of Data Leakage within the applied model**
- Data Leakage is observed at the time of usage of complex datasets. They are described below –
 - At the time of dividing the time series dataset into training and test, the dataset is a complex problem.
 - The implementation of sampling in a graphical problem is a complex task.
 - Storage of analog observations in the form of audios and images in separate files having a defined size and timestamp.

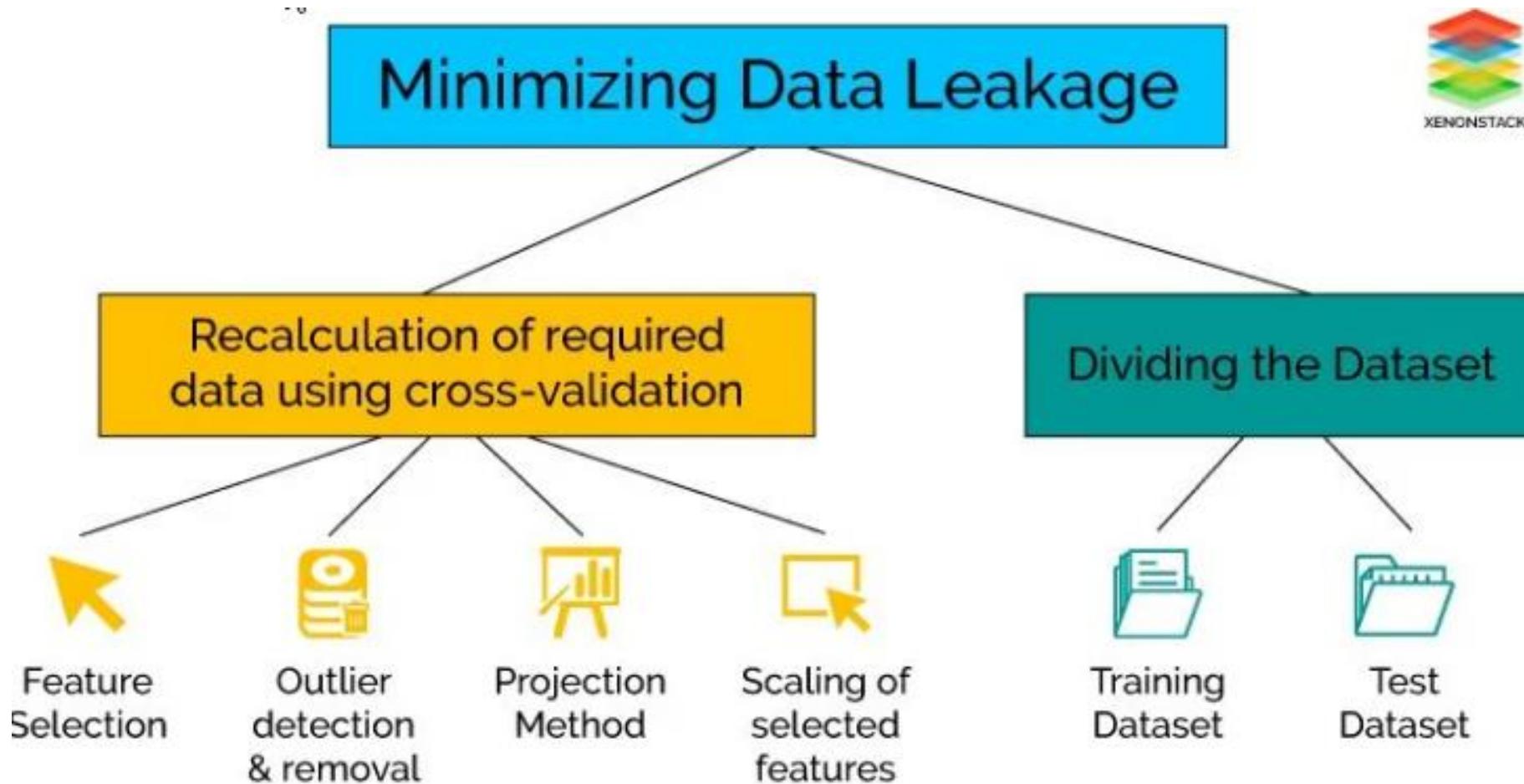
How is Data Wrangling performed?

- Data Wrangling is conducted to minimize the effect of Data Leakage while executing the model.
- The effect of Data Leakage could be minimized by recalculating for the required Data Preparation during the **cross-validation process** that includes **feature selection, outliers detection, and removal, projection methods, scaling of selected features** and much more.
- Another solution is that dividing the complete dataset into **training dataset** that is used to train the model and **validation dataset** which is used to evaluate the performance and accuracy of the applied model.

How is Data Wrangling performed?

- The selection of the model is made by looking at the results of the test data set in the cross-validation process. This conclusion will not always be valid as the sample of the test data set could vary.
- The performance of different models is evaluated for the particular type of test dataset. Therefore, while selecting the best model, test error is overfitting. The test error variance is determined by using different samples of the test dataset . Choosing of suitable model happens in this way.

How is Data Wrangling performed?



Data Preparation vs Data Wrangling

- Data Preprocessing steps are performed before the Data Wrangling.
- In this case, Data Preprocessing data is prepared **exactly after receiving the data from the data source**. In this initial transformations, Data Cleaning or any aggregation of data is performed. It is executed once.
- For example, we have data where one attribute has three variables, and we have to convert them into three attributes and delete the special characters from them. The concept of Data Preparation steps **performed before applying any iterative model** and will be executed once in the project.

Data Preparation vs Data Wrangling

- On the other hand, Data Wrangling is **performed during the iterative analysis** and **model building**. This concept at the time of feature engineering. The conceptual view of the dataset changes as different models are applied to achieve a good analytic model.
- For example, we have data containing 30 attributes where two attributes are used to compute another attribute, and that computed feature is used for further analysis. In this way, the **data** could be **changed according to the requirement of the applied model**, and Data Preparation can be effective.

Data Preparation vs Data Wrangling

- Summary
- Data Preprocessing: Preparation of data directly after accessing it from a data source. Typically realized by a developer or data scientist for initial transformations, aggregations and data cleansing. This step is done before the interactive analysis of data begins. It is executed once.
- Data Wrangling: Preparation of data during the interactive data analysis and model building. Typically done by a data scientist or business analyst to change views on a dataset and for features engineering. This step iteratively changes the shape of a dataset until it works well for finding insights or building a good analytic model.

Tasks of Data Wrangling

- **Discovering**
 - Firstly, data should be understood thoroughly and examine which approach will best suit. For example: You can liken it to looking in your refrigerator before cooking a meal to see what ingredients you have at your disposal.
- **Structuring**
 - As the data is gathered from different sources, the data will be present in various shapes and sizes. Therefore, there is a need for structuring the data in a proper format.
- **Cleaning**
 - Cleaning or removing of data should be performed that can degrade the performance of the analysis.

Tasks of Data Wrangling

- **Enrichment**
 - Extract new features or data from the given data set to optimize the performance of the applied model.
- **Validating**
 - This approach is used for improving the quality of data and consistency rules so that transformations that are applied to the data could be verified.
- **Publishing**
 - After completing the steps of Data Wrangling, the steps can be documented so that similar steps can be performed for the same kind of data to save time.

Data Wrangling vs ETL

- Data Wrangling is used to analyze the data that was gathered from different data sources. It is **designed specially to handle diverse and complex data of any scale**. But in the case of ETL, it can **handle structured data** that was originated from different databases or operating systems.
- Data Wrangling technology is used by **business analysts**, users engaged in business, and managers. On the other hand, ETL (Extract, Transform, and Load) is employed by **IT Professionals**. They receive the requirements from business people and then they use ETL tools to deliver the data in a required format.
- The primary task of the Data Wrangling method is to **manage the newly generated data** from various sources **for the analysis process** whereas the goal of ETL is to extract, transform and load the data into the central enterprise Data Warehouse for performing **analysis process using business applications**.

Tools

- **Data Preprocessing Tools**

- Data Preprocessing in R
- Data Preprocessing in Python
- Data Preprocessing in Weka

- **Data Wrangling Tools**

- Data Wrangling in Tabula
- Data Wrangling in R
- Data Wrangling in CSVKit
- Data Wrangling using Python with Pandas
- Data Wrangling using Mr. Data Converter

Encoding the categorical data

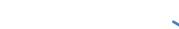
- Categorical data refers to the information that has specific categories within the dataset. Machine Learning models are primarily based on mathematical equations. Thus, we can intuitively understand that keeping the categorical data in the equation will cause certain issues since we would only need numbers in the equations.

Index	Country	Age	Salary	Purchased
0	India	38	68000	No
1	France	43	45000	Yes
2	Germany	30	54000	No
3	France	48	65000	No
4	Germany	40	nan	Yes
5	India	35	58000	Yes
6	Germany	nan	53000	No
7	France	49	79000	Yes
8	India	50	88000	No
9	France	37	77000	Yes

Encoding the categorical data

- **Label Encoding or Ordinal Encoding**

- We use this categorical data encoding technique when the categorical feature is ordinal. In this case, retaining the order is important. Hence encoding should reflect the sequence.
- In Label encoding, each **label is converted into an integer** value. We will create a variable that contains the categories representing the education qualification of a person.



Degree	Degree
0 High school	1
1 Masters	4
2 Diploma	2
3 Bachelors	3
4 Bachelors	3
5 Masters	4
6 Phd	5
7 High school	1
8 High school	1

Encoding the categorical data

- **Label Encoding or Ordinal Encoding**

- In this process, we assign a discrete number to each unique category using some defined process. For example, we can sort the variables in order of the **number of occurrences** and **number them in increasing order**.

Day of Month	Day of Week
2	Monday
4	Wednesday
8	Sunday
10	Tuesday
12	Thursday
13	Friday
14	Saturday

{ Monday -> 0
Wednesday -> 1
Sunday -> 2
Tuesday -> 3
Thursday -> 4
Friday -> 5
Saturday -> 6 }

Day of Month	Day of Week
2	0
4	1
8	2
10	3
12	4
13	5
14	6

Encoding the categorical data

- **Label Encoding or Ordinal Encoding**
- In this example, the days are labelled in the order of their appearance in the data. The major problems here are:-
- **Natural ordering is lost**
- Common relationships between categories are not captured. (For example, Saturday and Sunday together make a weekend and hence should be closer to each other)

Day of Month	Day of Week
2	Monday
4	Wednesday
8	Sunday
10	Tuesday
12	Thursday
13	Friday
14	Saturday

{ Monday -> 0
Wednesday -> 1
Sunday -> 2
Tuesday -> 3
Thursday -> 4
Friday -> 5
Saturday -> 6 }

Day of Month	Day of Week
2	0
4	1
8	2
10	3
12	4
13	5
14	6

Encoding the categorical data

- **One Hot Encoding**
 - We use this categorical data encoding technique when the features are nominal(do not have any order). In one hot encoding, for each level of a categorical feature, we create a new variable. Each category is mapped with a binary variable containing either 0 or 1. Here, **0 represents the absence**, and **1 represents the presence** of that category.
 - These newly created binary features are known as **Dummy variables**. The number of dummy variables depends on the levels present in the categorical variable. This might sound complicated. Let us take an example to understand this better.
 - Suppose we have a dataset with a category animal, having different animals like Dog, Cat, Sheep, Cow, Lion. Now we have to one-hot encode this data.

Encoding the categorical data

- **One Hot Encoding**
 - After encoding, we have dummy variables each representing a category in the feature Animal. Now for each category that is present, we have 1 in the column of that category and 0 for the others.

The diagram illustrates the One-Hot encoding process. On the left, a table shows the original data with an 'Index' column and an 'Animal' column. The 'Animal' column contains five categories: Dog, Cat, Sheep, Horse, and Lion. An arrow labeled 'One-Hot code' points from this table to the right, where another table shows the encoded data. This second table has an 'Index' column and six columns for different animals: Dog, Cat, Sheep, Lion, and Horse. The rows correspond to the indices in the first table. In the second table, the 'Dog' column has a value of 1 at index 0 and 0 elsewhere; the 'Cat' column has a value of 1 at index 1 and 0 elsewhere; the 'Sheep' column has a value of 1 at index 2 and 0 elsewhere; the 'Lion' column has a value of 1 at index 4 and 0 elsewhere; and the 'Horse' column has a value of 1 at index 3 and 0 elsewhere. All other entries in the second table are 0.

Index	Animal	Index	Dog	Cat	Sheep	Lion	Horse
0	Dog	0	1	0	0	0	0
1	Cat	1	0	1	0	0	0
2	Sheep	2	0	0	1	0	0
3	Horse	3	0	0	0	0	1
4	Lion	4	0	0	0	1	0

Encoding the categorical data

- **Dummy Encoding**
 - Dummy coding scheme is similar to one-hot encoding. This categorical data encoding method transforms the categorical variable into a set of binary variables (also known as dummy variables). In the case of one-hot encoding, for N categories in a variable, it uses N binary variables. The dummy encoding is a small improvement over one-hot-encoding. Dummy encoding uses **$N-1$ features** to represent N labels/categories.
 - To understand this better here we are coding the same data using both one-hot encoding and dummy encoding techniques. While one-hot uses 3 variables to represent the data whereas dummy encoding uses 2 variables to code 3 categories.

Encoding the categorical data

- **Dummy Encoding**

Column	Code
A	100
B	010
C	001

One- Hot Coding

Column	Code
A	10
B	01
C	00

Dummy Code

Encoding the categorical data

- **Drawbacks of One-Hot and Dummy Encoding**
 - One hot encoder and dummy encoder are two powerful and effective encoding schemes.
 - They are also very popular among the data scientists, But may not be as effective when- If there are multiple categories in a feature variable in such a case we need a similar number of dummy variables to encode the data. For example, a column with 30 different values will require 30 new variables for coding.
 - Due to the massive increase in the dataset, coding slows down the learning of the model along with deteriorating the overall performance that ultimately makes the model computationally expensive.

Encoding the categorical data

- **Count or frequency encoding**

- Replace the categories by the count of the observations that show that category in the dataset. Similarly, we can replace the category by the frequency -or percentage- of observations in the dataset. That is, if 10 of our 100 observations show the colour blue, we would replace blue by 10 if doing count encoding, or by 0.1 if replacing by the frequency.
- Limitation: If two different categories appear the same amount of times in the dataset, that is, they appear in the same number of observations, they will be replaced by the same number, hence, may lose valuable information.

Encoding the categorical data

- **Summary**
 - As handling categorical variables in any dataset is crucial step in feature engineering, any of the above techniques can be applied depending upon type of model.
 - If there are **lesser categories** and it is **nominal** categorical data, then **one-hot encoding** works just fine. If the relationship between any categorical column as independent variable and dependent variable (Target Variable) is important, then **Ordered Integer Encoding** can be applied. For **ordinal** categorical data, simply **Label Encoding** can be used.
 - Traditional techniques for handling categorical variables kind of works but limits the capabilities of algorithms.

Encoding the categorical data

- **One hot vectors vs Word embedding(word2vec, GloVe)**
 - One-hot vectors are high-dimensional and sparse, while word embedding's are low-dimensional and dense. When we use one-hot vectors as a feature in a classifier, your feature vector grows with the vocabulary size; word embedding's are more computationally efficient.
 - Word embedding's have the ability to generalize, due to semantically similar words having similar vectors, which is not the case in one-hot vectors (each pair of such vectors w_i, w_j has cosine similarity $\cos(w_i, w_j)=0$).
 - If feature vector contains one-hot vectors of the documents' words, we will only be able to consider features we've seen during training; when we use embedding, semantically similar words will create similar features, and will lead to similar classification.

Encoding the categorical data

- **One hot vectors vs Word embedding(word2vec, GloVe)**
 - Example. Let's say that your classifier works with the bag-of-words approach, i.e. the feature vector is the sum of all the document's word vectors (which is equivalent to a vector that counts the number of occurrences of each word in the vocabulary in the one-hot representation).
 - Suppose that in your training data you have a document with the single word *school*, and your test data contains a document with the single word *education*, that wasn't previously observed during training. In the one-hot vector representation the feature vectors of these two instances will be completely different, possibly leading to different class prediction, while in word embedding representation, they will be similar, hopefully leading to the same classification.

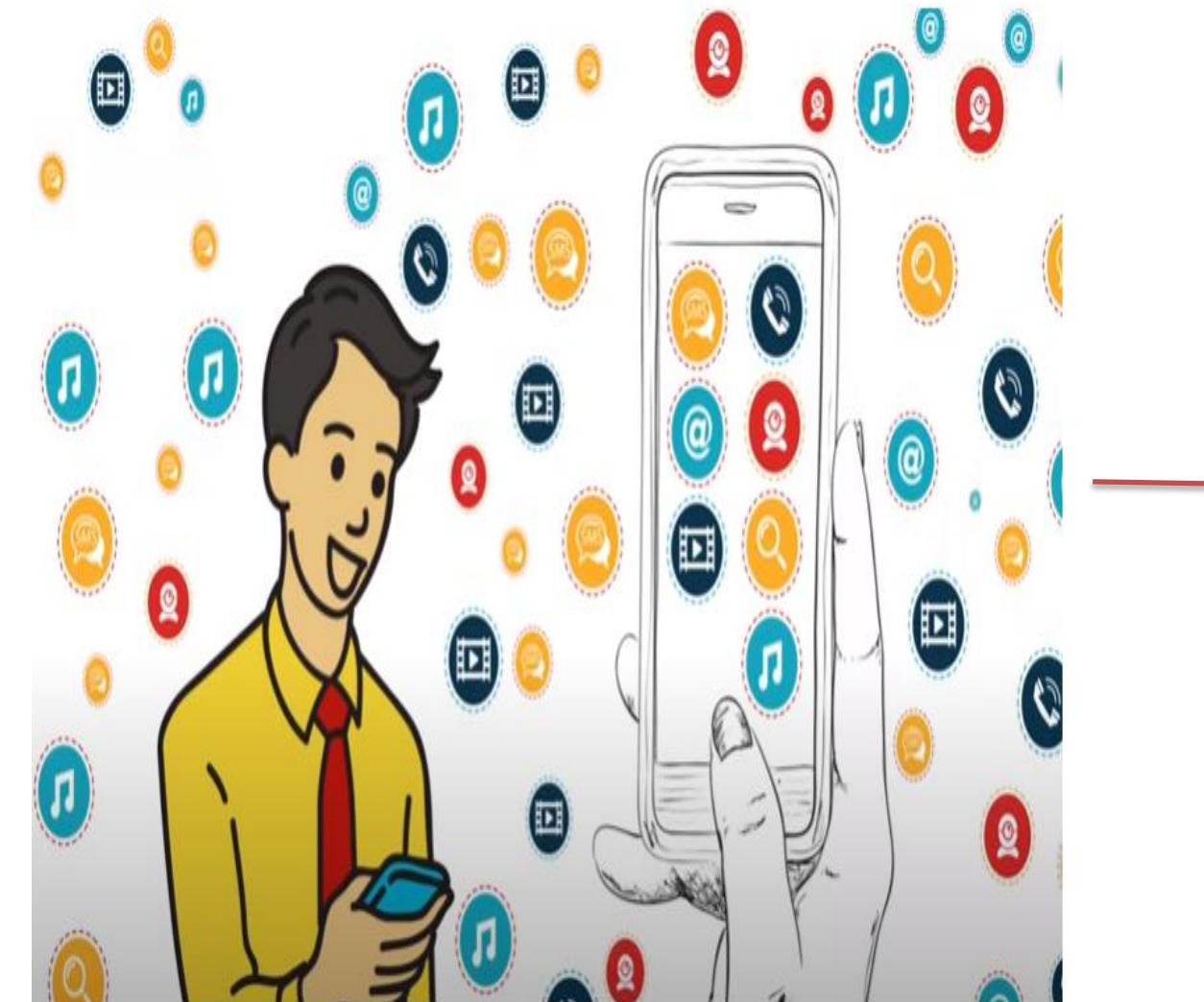
References

- Data Integration tool- Talend Open Studio
 - <https://www.talend.com/resources/introduction-talend-open-studio-data-integration/>
 - https://info.talend.com/rs/talend/images/WP_EN_DI_Talend_Definitive_Guide_DataIntegration.pdf
- Data wrangling tool
 - <http://vis.stanford.edu/wrangler/>
- One hot encoding
 - https://colab.research.google.com/github/alzayats/Google_Colab/blob/master/6_1_one_hot_encoding_of_words_or_characters.ipynb#scrollTo=xaiRJYibT-u3

Thank you.

Big Data

Introduction to Big data



40 Exabytes x 5,000,000,000



Introduction to Big data

Let's have a look at the data generated per minute
on the internet



"2.1Million"



"3.8Million"



"1.0Million"



"4.5Million"



"188Million"

That's a lot of data

Introduction to Big data

- Facebook: Facebook is collecting a huge amount of data. Every time whenever you are clicking a notification, visiting a page, uploading a photo, or checking out a friend's link, you're generating data for the company to track various records.
- Users shared 2.5 billion content items daily (status updates + wall posts + photos + videos + comments). ~~300 million photos are uploaded by users per day. 105 terabytes of data scanned via Hive, Facebook's Hadoop query language in every 30 minutes. 70,000 queries executed on these databases per day. 500+terabytes of new data ingested into the databases every day.~~

Digital World



Social media and networks
(all of us are generating data)



Scientific instruments
(collecting all sorts of data)



Mobile devices
(tracking all objects all the time)



Sensor technology and networks
(measuring all kinds of data)

Introduction to Big data

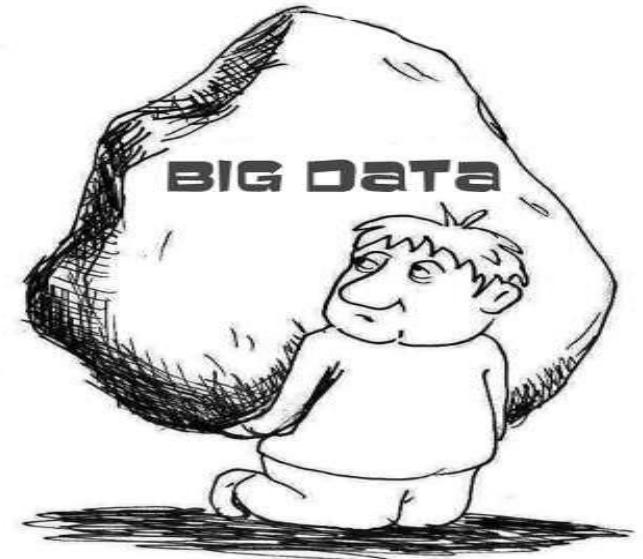
- A massive amount of data that **cannot** be stored, processed, and analysed using the **traditional ways** (Why? => Storage capacity, Processing power)
- The term Big data refers to a huge volume of data that can not be stored processed by any traditional data storage or processing units, that is generated at a very large scale and it is being used to process and analyze in order to uncover insights.
- Big Data is a collection of data that is huge in volume, yet **growing exponentially** with time.

Introduction to Big data

Big Data is the amount of data just beyond technology's capability to store, manage and process efficiently.

“Big data is high-volume, high-velocity and high-variety information assets that demand cost-effective, innovative forms of information processing for enhanced insight and decision making”

“Big data is a term that describes large volumes of high velocity, complex and variable data that require advanced techniques and technologies to enable the capture, storage, distribution, management, and analysis of the information”



History of big data

- Although the concept of big data itself is relatively new, the origins of large data sets go back to the 1960s and '70s when the world of data was just getting started with the first data centres and the development of the relational database.
- Around 2005, people began to realize just **how much data users generated** through Facebook, YouTube, and other online services. Hadoop (an open-source framework created specifically to store and analyse big data sets) was developed that same year. NoSQL also began to gain popularity during this time.

History of big data

- The development of open-source frameworks, such as Hadoop (and more recently, Spark) was essential for the growth of big data because they make big data easier to work with and **cheaper to store**. In the years since then, the volume of big data has skyrocketed. Users are still generating huge amounts of data—but it's not just humans who are doing it.
- With the advent of the Internet of Things (IoT), more objects and devices are connected to the internet, gathering data on customer usage patterns and product performance. The emergence of machine learning has produced still more data.

History of big data

- While big data has come far, its usefulness is only just beginning. Cloud computing has expanded big data possibilities even further. The cloud offers truly elastic scalability, where developers can simply spin up ad hoc clusters to test a subset of data. And **graph databases** are becoming increasingly important as well, with their ability to **display massive amounts of data** in a way that makes analytics fast and comprehensive.

Small Vs Big Data

Small / Traditional Data	Big Data
Mostly Structured	Structured, Unstructured and Semi-structured
Data store in MB, GB, TB	Data store in PB, EB
Data Increase Gradually	Data Increases Exponentially
Locally Present, Save in Centralized manner Example: Universities / Colleges data (Attendance, Library)	Globally Present, Distributed Example: Facebook, Google
Software: SQL Server, Oracle	Software: Hadoop, Spark, Big Query
To handle traffic -> Single node	To handle traffic -> Multinode cluster

Types Of Big Data

- **Structured:** Structured data owns a dedicated data model, is also has a **well defined structure**, it follows a consistent order and it is designed in such a way that it can be easily accessed and used by a person or a computer. Structured data is usually stored in well defined columns and also databases.
Example: DBMS
- **Unstructured:** Different type od data which neither has a **structure** nor obeys to follow the **formal structure** rules of data models. It does not even have a consistent format and it found to be varying all the time. But rarely it may have information related to data and time.
Example: audio, video, images etc.
- **Semi-Structured:** can be considered as another form of structure data. It inherits a few properties of structured data, but the major part of this kind of data **fails to have a definite structure** and also it does not obey the formal structure of data models such as RDBMS. **Example: CSV File**

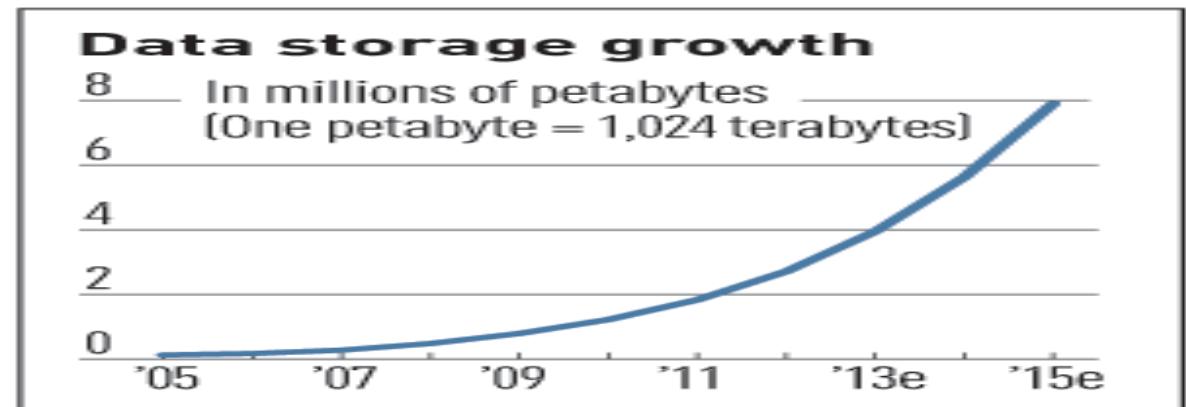
Characteristics Of Big Data

- Big data can be described by the following characteristics:
 - Volume
 - Variety
 - Velocity
 - Veracity
 - Value



Characteristics Of Big Data

- **Volume** – The name Big Data itself is related to a **size** which is enormous. Size of data plays a very crucial role in determining value out of data. Also, whether a particular data can actually be considered as a Big Data or not, is dependent upon the volume of data. Hence, '**Volume**' is one characteristic which needs to be considered while dealing with Big Data solutions.
- Example: Facebook alone can generate about billion messages, 4.5 billion times that the “like” button is recorded and over 350 million new posts are uploaded each day.



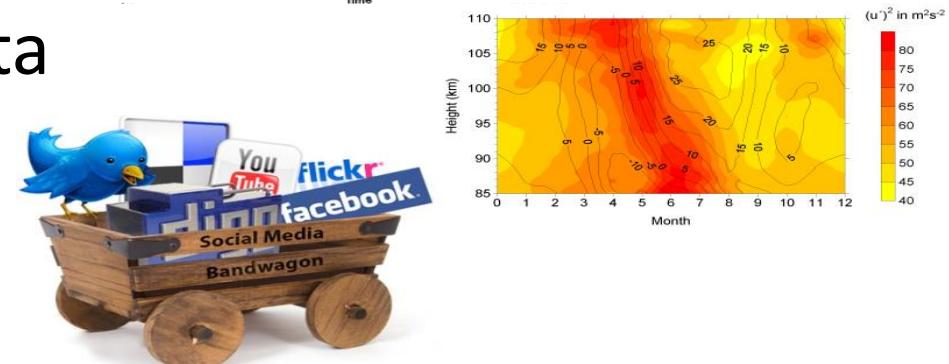
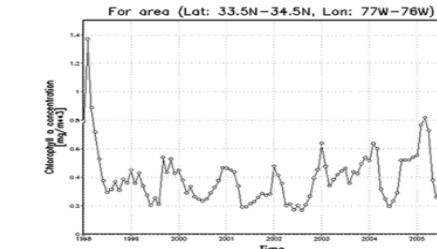
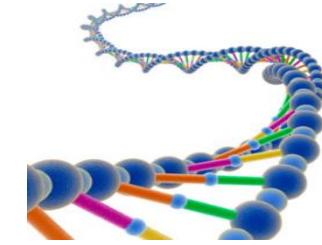
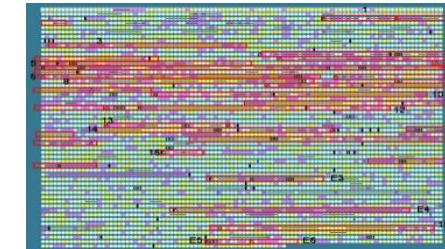
Characteristics Of Big Data

- **Variety** – The next aspect of Big Data is its variety.
 - Variety refers to **heterogeneous sources and the nature of data**, both structured and unstructured. During earlier days, spread sheets and databases were the only sources of data considered by most of the applications. Nowadays, data in the form of **emails, photos, videos, monitoring devices, PDFs, audio, etc.** are also being considered in the analysis applications. This variety of unstructured data poses certain issues for storage, mining and analyzing data.

Characteristics Of Big Data

- Various formats, types, and structures
- Text, numerical, images, audio, video, sequences, time series, social media data, multi-dim arrays, etc...
- A single application can be generating/collecting many types of data

To extract knowledge → all these types of data need to linked together



Characteristics Of Big Data

- **Velocity** – The term 'velocity' refers to the **speed of generation of data**. How fast the data is generated and processed to meet the demands, determines real potential in the data.
 - Big Data Velocity deals with the speed at which data flows in from sources like business processes, application logs, networks, and social media sites, sensors, Mobile devices, etc. The flow of data is massive and continuous.

Characteristics Of Big Data

- Data is begin generated fast and need to be processed fast
- Online Data Analytics
- Late decisions → missing opportunities
- **Examples**
 - **E-Promotions:** Based on your current location, your purchase history, what you like → send promotions right now for store next to you
 - **Healthcare monitoring:** sensors monitoring your activities and body → any abnormal measurements require immediate reaction

Characteristics Of Big Data

- **Veracity** – Veracity basically means the degree of reliability that the data has to offer. Since a major part of the data is **unstructured** and **irrelevant**, Big data needs to find an alternate way to filter them or to translate them out as the data is crucial in business developments
- **Value** – The value of the data is that they are actionable, responsible for the companies to make a decisions. Value is a major issue that we need to concentrate on. It is not just the amount of data that we store or process. It is the amount of valuable, reliable and trustworthy data that needs to stored, processed and analyzed to find insights.

Characteristics Of Big Data

The 5 V's of Big Data



01. Volume

The amount of data



Big data involves a huge volume of data. The volume refers to the amount of data generated every second, minute and days.

02. Velocity

An unprecedented speed



Speed is the Big V that represents how fast data is being received and processed.

03. Variety

The types of data that are available



When working with so much data, a lot of it is unstructured and need to be further processed to structure it properly.

04. Veracity

The degree to which Big Data can be trusted



When you have a lot of data, you can actually use it for very different purposes and format it in different ways.

05. Value

The final output from data management



The value of the data is that they are actionable, responsible for the companies to make a decision.



Characteristics Of Big Data

- **Additional V's of Big Data**
- Other characteristics and properties are as follows:
 - **Visualization** means collecting and analyzing a huge amount of information using Real time analytics to make it understandable and easy to read. Without this, it is impossible to maximize and leverage the raw information.
 - **Validity:** It means how clean, accurate, and correct the information is to use. The benefit of analytics is only as good as its underlying information, so good data governance practices should be adopted to ensure consistent data quality, common definitions, and metadata.

Characteristics Of Big Data

- **Additional V's of Big Data**
 - **Volatility:** How long is data valid and how long should it be stored. In this world of real time data we need to determine at what point is data no longer relevant to the current analysis.
 - **Vulnerability:** A huge volume of data comes up with many new security concerns since there have been many big data breaches.
 - **Variability:** Some data streams can have peaks and seasonality, periodicity. Managing a large amount of unstructured information is difficult and requires powerful processing techniques.

Applications Of Big Data

- **Entertainment:** Netflix and amazon use big data to make shows and movie recommendation to their users
- **Insurance:** uses big data to predict illness, accidents and price their products accordingly.
- **Driverless cars:** Google's driverless cars collect about one gigabyte data per second. These experiments require more and more data for their successful execution.
- **Education:** Opting for big data part technology as a learning tool instead of traditional lecture methods which enhance the learning of students as well as aided the teacher to track the performance better.
- **Automobiles:** Rolls Royce has embraced big data by fitting hundreds of sensors into its engines, Which record every tiny detail about their operation.

Applications Of Big Data

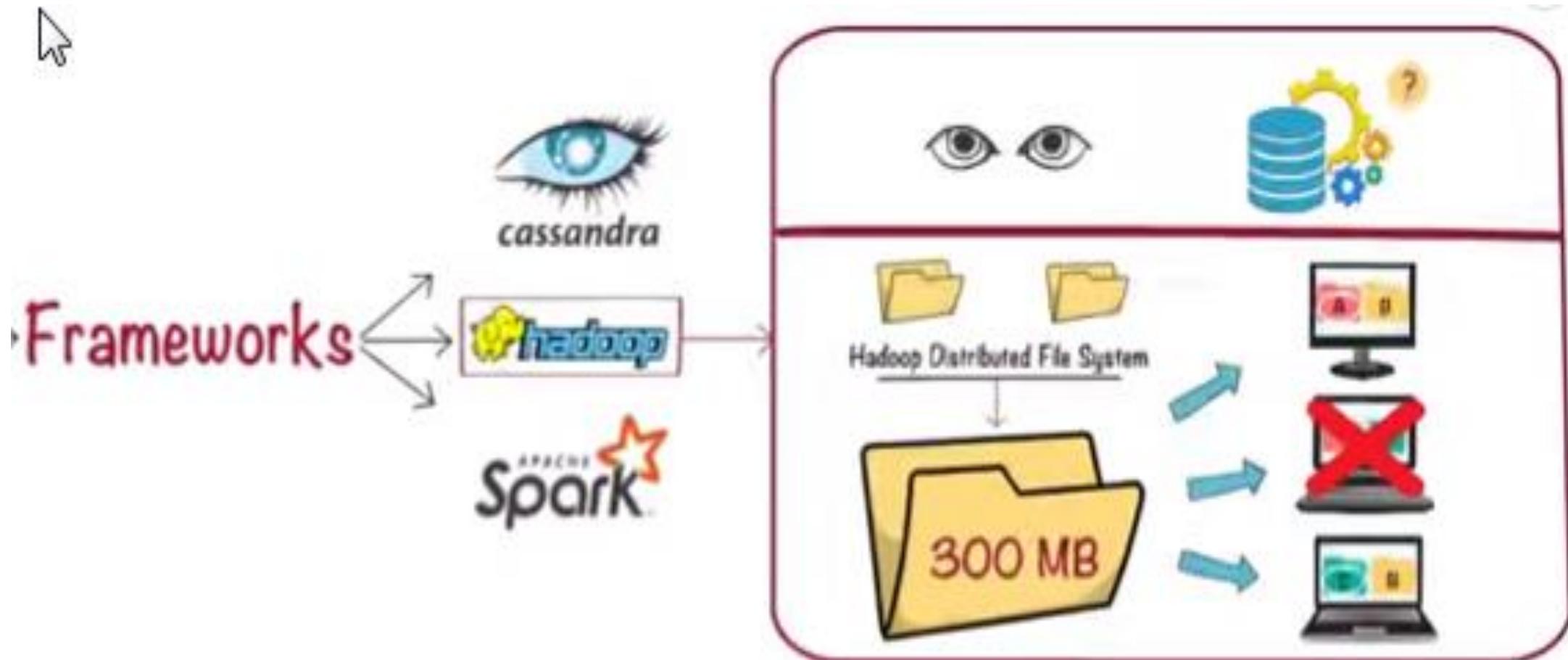
- **Travel and tourism:** Enabled us to predict requirements of travel facilities in many places improving the business through dynamic pricing and many more.
- **Financial and banking sector:** Big data analytics can aid banks in understanding customer behavior based on inputs received from the investments patterns, shopping trends motivation to invent and personal or financial backgrounds.
- **Healthcare Sector:** Medical professionals and healthcare persons are now capable to provide personalized healthcare services to individual patients.
- **Telecommunication and media sector:** Zettabytes of data getting generated every day and to handle such huge data we need big data technologies.
- **Politics:** To analyze patters and influence election results.

Benefits Of Big Data

- Better decision making
- Greater innovations
- Improvement in education sector
- Product price optimization
- Recommendation engines
- Life-Saving application in the healthcare industry
- Predictive analysis which can serve organizations from operational risk.
- Organizations grow business by analyzing customer needs.
- Enabled many multimedia platforms to share data Ex. YouTube, Instagram

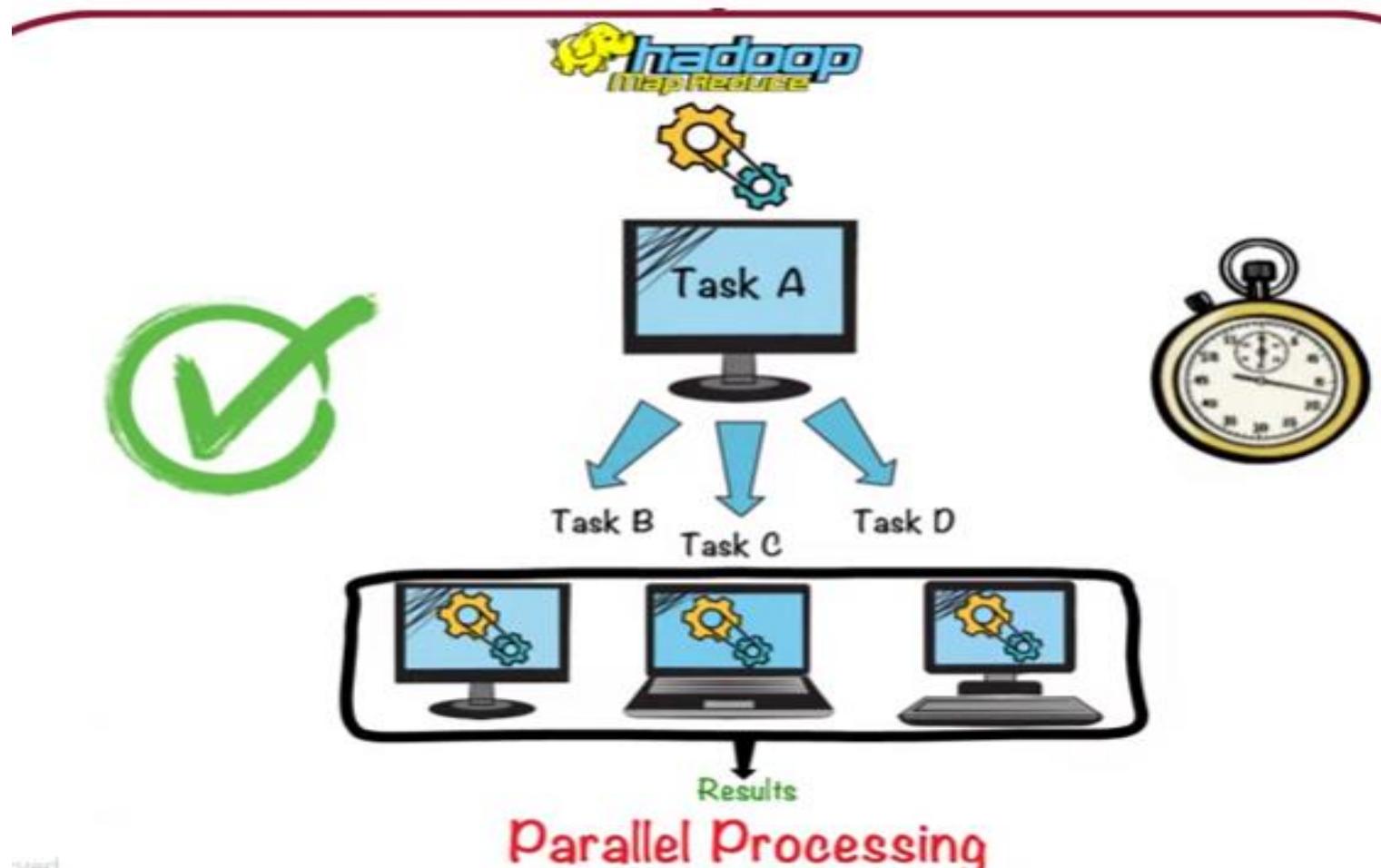
Frameworks

- How do we store this big data?



Frameworks

- How do we process this big data?



Big Data Frameworks/Tools

- Here is the list of top 10 big data tools –

- Apache Hadoop
- Apache Spark
- Kafka
- Flink
- Apache Storm
- Apache Cassandra
- MongoDB
- Tableau
- RapidMiner
- R Programming

Big Data Frameworks/Tools

- These big data tool not only helps you in **storing large data** but also helps in **processing the stored data in a faster way** and provides you better results and new ideas for the growth of your business.
- There are a vast number of Big Data tools available in the market. You just need to choose the right tool according to the requirements of your project.
- Remember, “If you choose the right tool and use it properly, you will create something extraordinary; If used wrong, it makes a mess.”

Big Data Case studies

- **Walmart**
 - Walmart leverages Big Data and Data Mining to create personalized product recommendations for its customers. With the help of these two emerging technologies, Walmart can uncover valuable patterns showing the **most frequently bought products**, **most popular products**, and even the **most popular product bundles** (products that complement each other and are usually purchased together).
 - Based on these insights, Walmart creates attractive and customized recommendations for individual users. By effectively implementing Data Mining techniques, the retail giant has successfully increased the conversion rates and improved its customer service substantially. Furthermore, Walmart uses Hadoop and NoSQL technologies to allow customers to access real-time data accumulated from disparate sources.

Big Data Case studies

- **Uber**
 - Uber is one of the major cab service providers in the world. It leverages customer data to **track and identify the most popular and most used services by the users**. Once this data is collected, Uber uses data analytics to analyze the usage patterns of customers and determine which services should be given more emphasis and importance.
 - Apart from this, Uber uses Big Data in another unique way. Uber closely studies the demand and supply of its services and changes the cab fares accordingly. It is the surge pricing mechanism that works something like this – suppose when you are in a hurry, and you have to book a cab from a crowded location, Uber will charge you double the normal amount!

Big Data Case studies

- **Starbucks:** Starbucks use big data to analyze the **preferences** of their customers to enhance and personalize their experience. They analyze their members **coffee buying habits** along with their **prefer drinks**.
 - So even when people visit a new starbucks location that stores point-of-scale system is able to identify customer through their smartphone and give their buy star as their preferred order in addition based on ordering preferences their app will suggest the new products that the customers might be interested in trying.
- This is what we call big data analytics.

What is Big Data Analytics

- Big data analytics is largely used by companies to facilitate their growth and development. This majorly involves **applying** various **data mining algorithms** on the given set of data, which will then aid them in better decision making.

"Big data analytics examines large and different types of data to uncover hidden patterns, correlations and other insights"



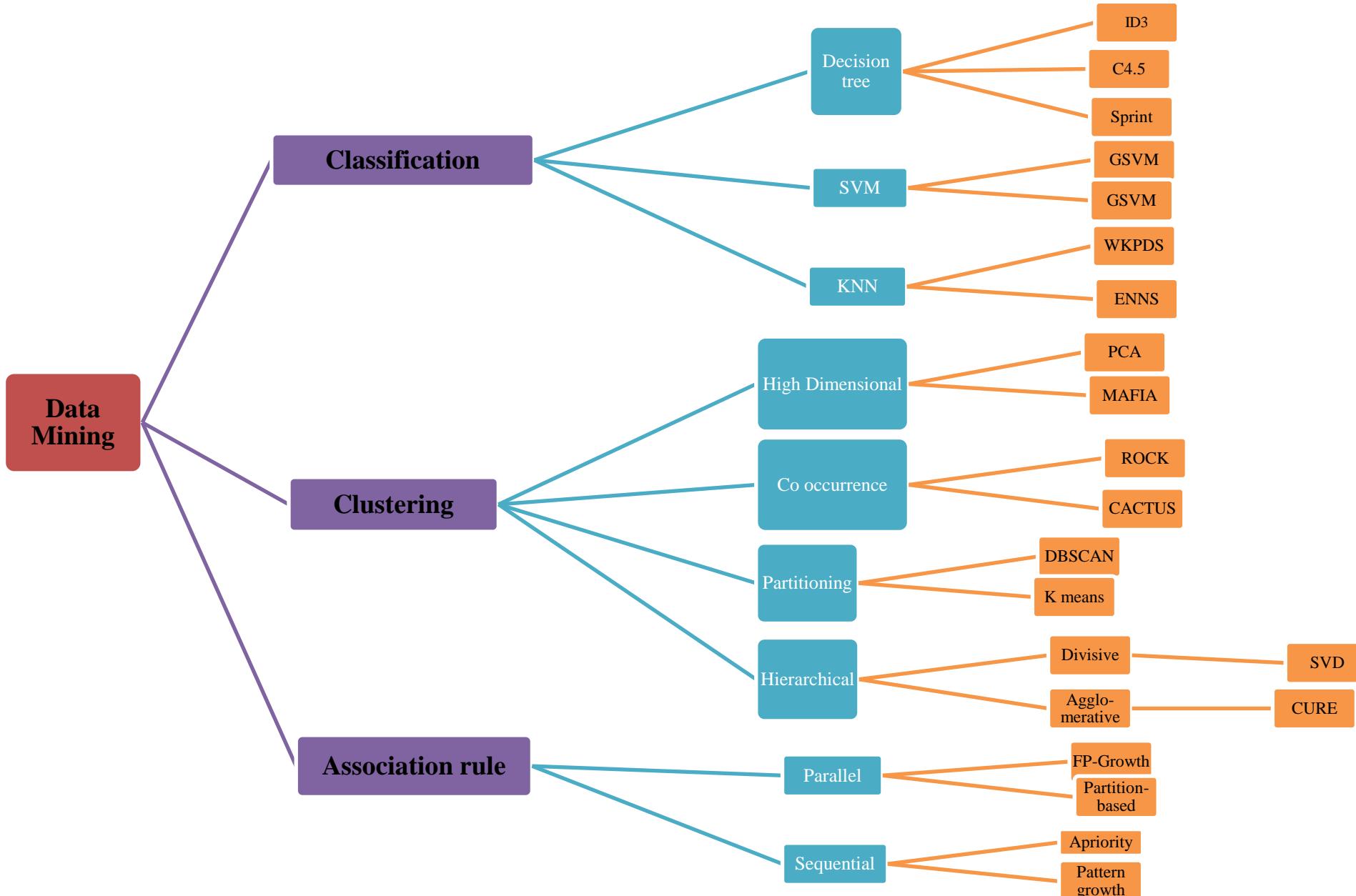


Figure: Taxonomy of Data mining algorithms

Need of Big Data Analytics

1. Making Smarter and more efficient organization

- Big data analytics is basically highly contributing to these factors and organizations are adopting this to basically lead them to faster decision making.
- Example: New York Police Department (NYPD)
 - Big data and analytics helping the NYPD and the other large police departments to **anticipate and identify the criminal activity** before it occurs. so what they do is that they analyze the big data technology to geo locate and then analyze the historical patterns and they map these historical patterns with sporting events pea days, rain falls, traffic flows and federal holidays.

Need of Big Data Analytics

- Example: New York Police Department (NYPD)
 - So essentially What the NYPD is doing that they utilizing these data patterns, scientific analytics, technological tools to do their job and they are ensuring that by using these different tools they are doing their job to the best of their ability.
 - So by using a **big data and analytic strategy**; the NYPD is able to **identify** something called **crime hotspot** so basically where crime occurrence was more. so they were able to identify these hotspot and then from there they deployed their local officers so that they could reach there on time before it was actually committed.

Need of Big Data Analytics

2

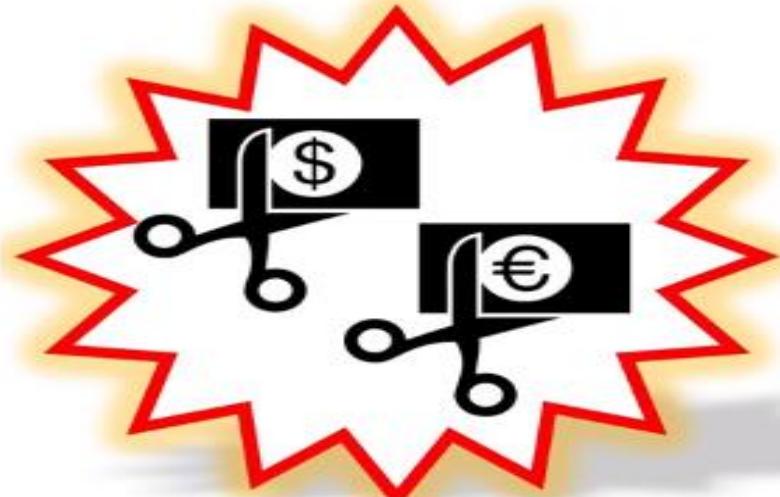
Optimize Business Operations by analysing customer behaviour



- **Example:** Amazon uses customer click stream data and historical purchase data of more than 300 million customer and each user is shown customized results on customized web pages.

Need of Big Data Analytics

③ Cost Reduction



Parkland Hospital uses analytics and predictive modelling to identify high-risk patients and predict likely outcomes once patients are sent home. As a result, Parkland reduced 30-day readmissions for patients with heart failure, by 31 percent, saving \$500,000 annually.



- Reduce cost by using Hadoop technology. Hadoop stores big data in distributed fashion so that we can process it parallel. So it reduces our cost a lot. So by using commodity hardware they are reducing their cost significantly.

Need of Big Data Analytics

4

Next Generation Products

Big Data tools are used to operate Google's Self Driving Cars. The Toyota Prius is fitted with cameras, GPS as well as powerful computers and sensors to safely drive on the road without the intervention of human beings.



Netflix launched the seasons of its TV show House of Cards based on the user reviews, ratings and viewership.



A smart yoga mat has sensors embedded in the mat will be able to provide feedback on your postures, score your practice, and even guide you through an at-home practice.



How big data analytics works

- Big data analytics refers to collecting, processing, cleaning, and analyzing large datasets to help organizations operationalize their big data.
- **Collect Data**
 - Data collection looks different for every organization. With today's technology, organizations can gather both structured and unstructured data from a variety of sources — from **cloud storage to mobile applications to in-store IoT sensors and beyond**. Some data will be stored in data warehouses where business intelligence tools and solutions can access it easily. Raw or unstructured data that is too diverse or complex for a warehouse may be assigned metadata and stored in a data lake.

How big data analytics works

- **Process Data**
 - Once data is collected and stored, it must be organized properly to get accurate results on analytical queries, especially when it's large and unstructured. Available data is growing exponentially, making data processing a challenge for organizations. One processing option is **batch processing**, which looks at large data blocks over time. Batch processing is useful when there is a longer turnaround time between collecting and analyzing data. **Stream processing** looks at small batches of data at once, shortening the delay time between collection and analysis for quicker decision-making. Stream processing is more complex and often more expensive.

How big data analytics works

- **Clean Data**
 - Data big or small requires scrubbing to improve data quality and get stronger results; all data must be formatted correctly, and any duplicative or irrelevant data must be eliminated or accounted for. Dirty data can obscure and mislead, creating flawed insights.

How big data analytics works

- **Analyze Data**
 - Getting big data into a usable state takes time. Once it's ready, advanced analytics processes can turn big data into big insights. Some of these big data analysis methods include:
 - **Data mining** sorts through large datasets to identify patterns and relationships by identifying anomalies and creating data clusters.
 - **Predictive analytics** uses an organization's historical data to make predictions about the future, identifying upcoming risks and opportunities.
 - **Deep learning** imitates human learning patterns by using artificial intelligence and machine learning to layer algorithms and find patterns in the most complex and abstract data.

Aspects of Big data

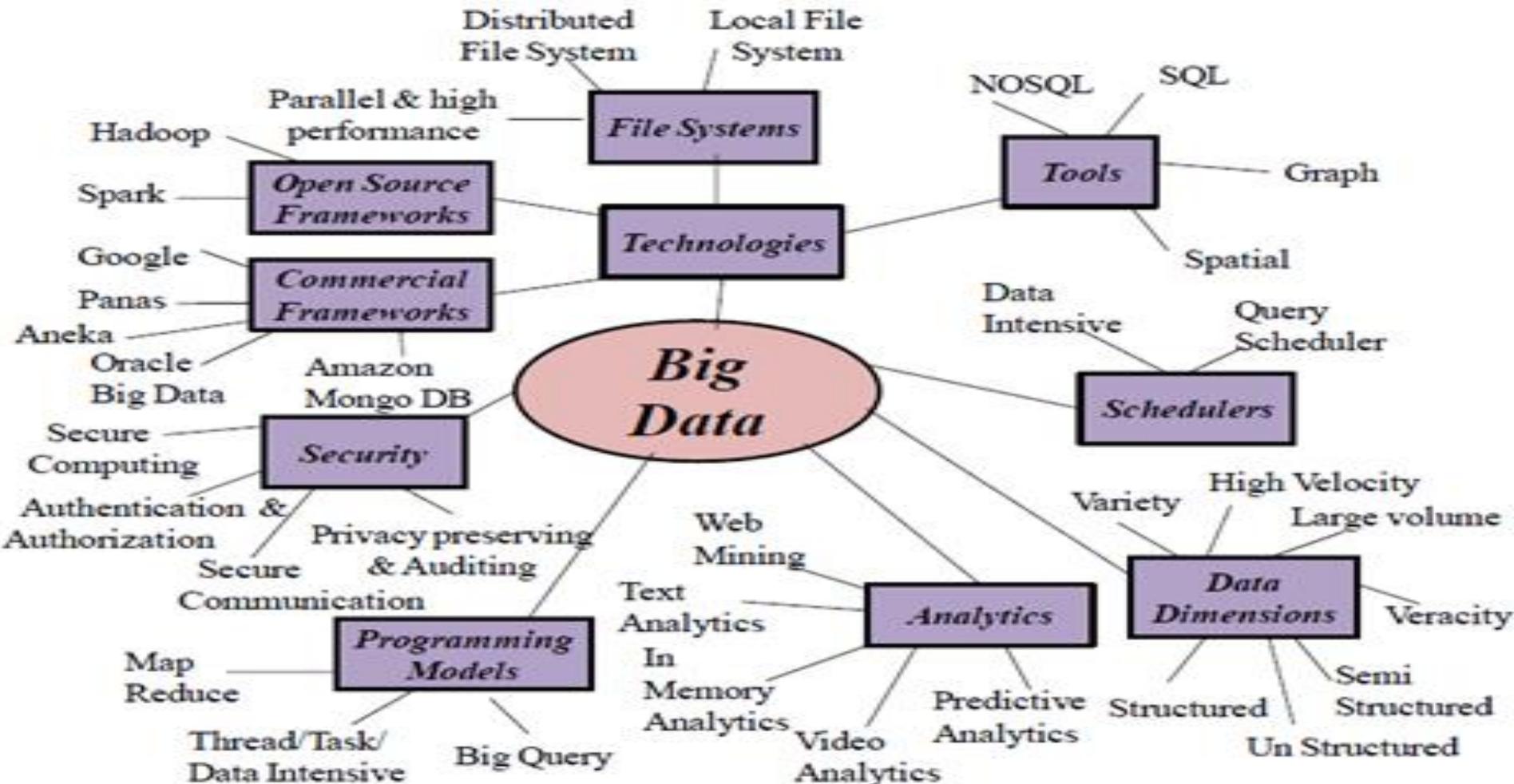


Figure:Overview of different aspects of Big data [6]

The big challenges of big data

- Big data brings big benefits, but it also brings big challenges such new privacy and security concerns, accessibility for business users, and choosing the right solutions for your business needs. To capitalize on incoming data, organizations will have to address the following:
- **Making big data accessible.** Collecting and processing data becomes more difficult as the amount of data grows. Organizations must make data easy and convenient for data owners of all skill levels to use.
- **Maintaining quality data.** With so much data to maintain, organizations are spending more time than ever before scrubbing for duplicates, errors, absences, conflicts, and inconsistencies.

The big challenges of big data

- **Keeping data secure.** As the amount of data grows, so do privacy and security concerns. Organizations will need to strive for compliance and put tight data processes in place before they take advantage of big data.
- **Finding the right tools and platforms.** New technologies for processing and analyzing big data are developed all the time. Organizations must find the right technology to work within their established ecosystems and address their particular needs. Often, the right solution is also a flexible solution that can accommodate future infrastructure changes.

Thank you.



Why Hadoop is Invented?

- **Shortcomings of the traditional approach** which led to the invention of Hadoop –
- **Storage for Large Datasets**
 - The conventional RDBMS is incapable of storing huge amounts of Data. The cost of data storage in available RDBMS is very high. As it incurs the cost of hardware and software both.
- **Handling data in different formats**
 - The RDBMS is capable of storing and manipulating data in a structured format. But in the real world we have to deal with data in a structured, unstructured and semi-structured format.

Why Hadoop is Invented?

- **Data getting generated with high speed:**
 - The data is oozing out in the order of tera to peta bytes daily. Hence we need a system to process data in real-time within a few seconds. The traditional RDBMS fail to provide real-time processing at great speeds.

Why Hadoop?

- Apache Hadoop is not only a storage system but is a platform for data storage as well as processing. It is **scalable** (as we can add more nodes on the fly), **Fault-tolerant** (Even if nodes go down, data processed by another node).
- Following characteristics of Hadoop make it a unique platform:
 - Flexibility to store and mine any type of data whether it is structured, semi-structured or unstructured. It is not bounded by a single schema.
 - Excels at processing data of complex nature. Its scale-out architecture divides workloads across many nodes. Another added advantage is that its flexible file-system eliminates ETL bottlenecks.
 - Scales economically, as discussed it can deploy on commodity hardware. Apart from this its open-source nature guards against vendor lock.

Fault Tolerance

- ▶ Failures are detected by the master program which reassigns the work to a different node
- ▶ Restarting a task does not affect the nodes working on other portions of the data
- ▶ If a failed node restarts, it is added back to the system and assigned new tasks
- ▶ The master can redundantly execute the same task to avoid slow running nodes

What is Hadoop?

- Hadoop is the solution to above Big Data problems. It is the technology to **store massive datasets** on a cluster of cheap machines in a **distributed manner**. Not only this it provides Big Data analytics through distributed computing framework.
- It is an **open-source software** developed as a project by Apache Software Foundation. **Doug Cutting** created Hadoop. In the year 2008 Yahoo gave Hadoop to Apache Software Foundation. Since then three versions of Hadoop has come. Version 1.0 in the year 2011, version 2.0.6 in the year 2013 and Version 3.1.x – released on 21 October 2019. Hadoop comes in various flavors like Cloudera, IBM BigInsight, MapR and Hortonworks.

Hadoop Advantages

- ▶ Unlimited data storage
 1. Server Scaling Mode
 - a) Vertical Scale
 - b) Horizontal Scale
- ▶ High speed processing system
- ▶ All varieties of data processing
 1. Structural
 2. Unstructural
 3. semi-structural

Uses for Hadoop

- ▶ Data-intensive text processing
- ▶ Graph mining
- ▶ Machine learning and data mining
- ▶ Large scale social network analysis

Who Uses Hadoop?



eHarmony®



facebook

IBM

twitter

amazon.com

SAMSUNG

The New York Times

JPMorganChase

intel

NETFLIX

VISA

YAHOO!

Prerequisites to Learn Hadoop

- **Familiarity with some basic Linux Command** – Hadoop is set up over Linux Operating System preferable Ubuntu. So one must know certain ***basic Linux commands***. These commands are for uploading the file in HDFS, downloading the file from HDFS and so on.
- **Basic Java concepts** – Folks want to learn Hadoop can get started in Hadoop while simultaneously grasping basic concepts of JAVA. We can write **map and reduce functions** in Hadoop using other languages too. And these are **Python, Perl, C, Ruby**, etc. This is possible via streaming API. It supports reading from standard input and writing to standard output. Hadoop also has high-level abstractions tools like Pig and Hive which do not require familiarity with Java.

Hadoop

- Hadoop is Open Source Framework and Distributed Under Apache
- When should we go for Hadoop?
 - Data is too huge
 - Processes are independent
 - Better scalability
 - Parallelism
 - Unstructured data
- Hadoop Roles
 - Hadoop Admin
 - Hadoop Developer
- Hadoop Support
 - Java, Scala, Python, C++, C, Ruby

What is Hadoop Architecture?

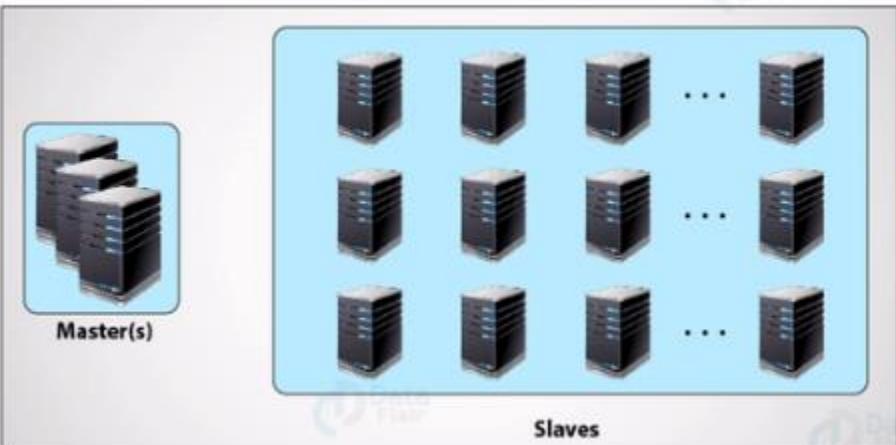
- Hadoop works in master-slave fashion. There is a **master node** and there are **n numbers of slave nodes** where n can be 1000s. Master manages, maintains and monitors the slaves while slaves are the actual worker nodes. In Hadoop architecture, the Master should deploy on good configuration hardware, not just commodity hardware. As it is the centrepiece of Hadoop cluster.
- Master stores the metadata (data about data) while slaves are the nodes which store the data. Distributedly data stores in the cluster. The client connects with the master node to perform any task.

What is Hadoop Architecture?

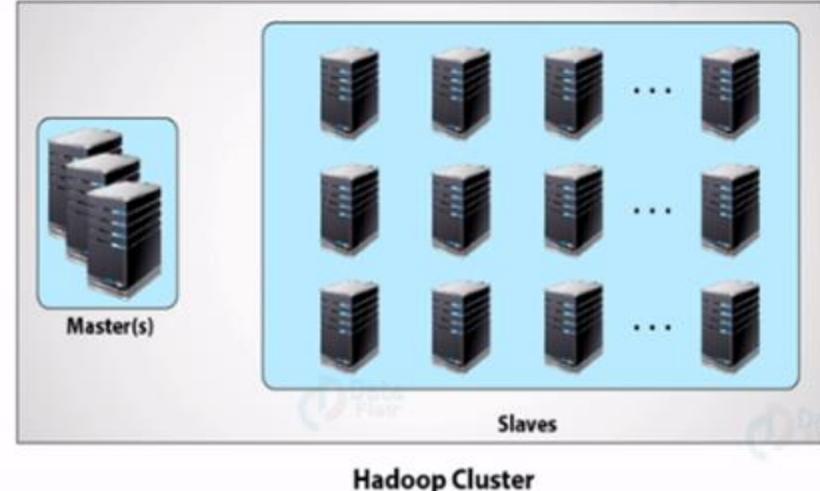
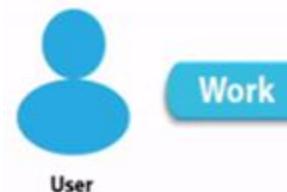
Develops the work



User



User submits work
on master



Hadoop Cluster

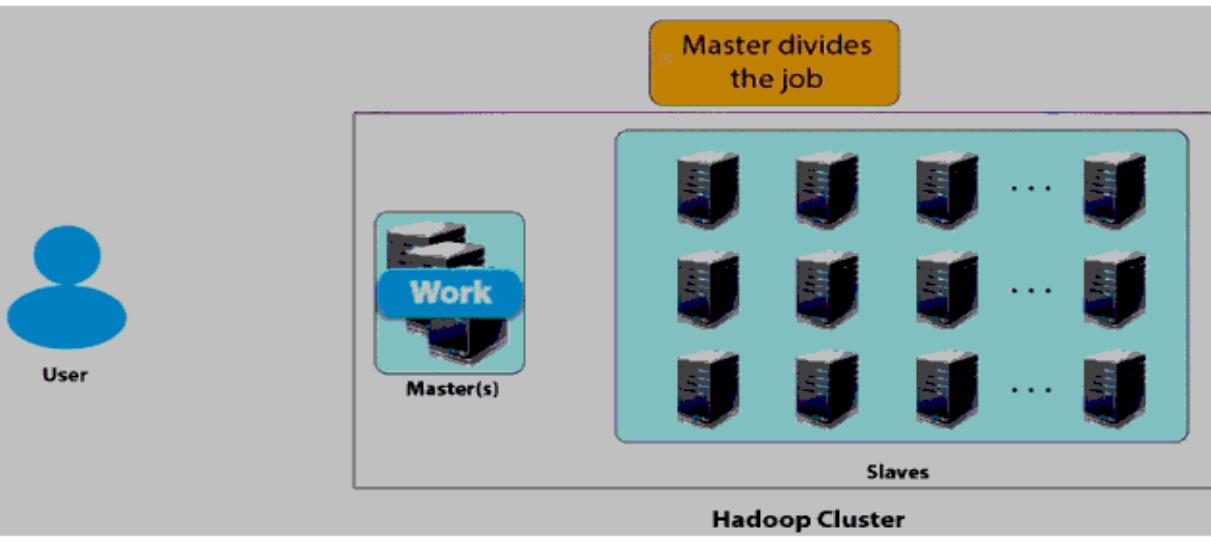
Master assigns
sub-work to slaves



User



Master divides
the job



Hadoop Cluster

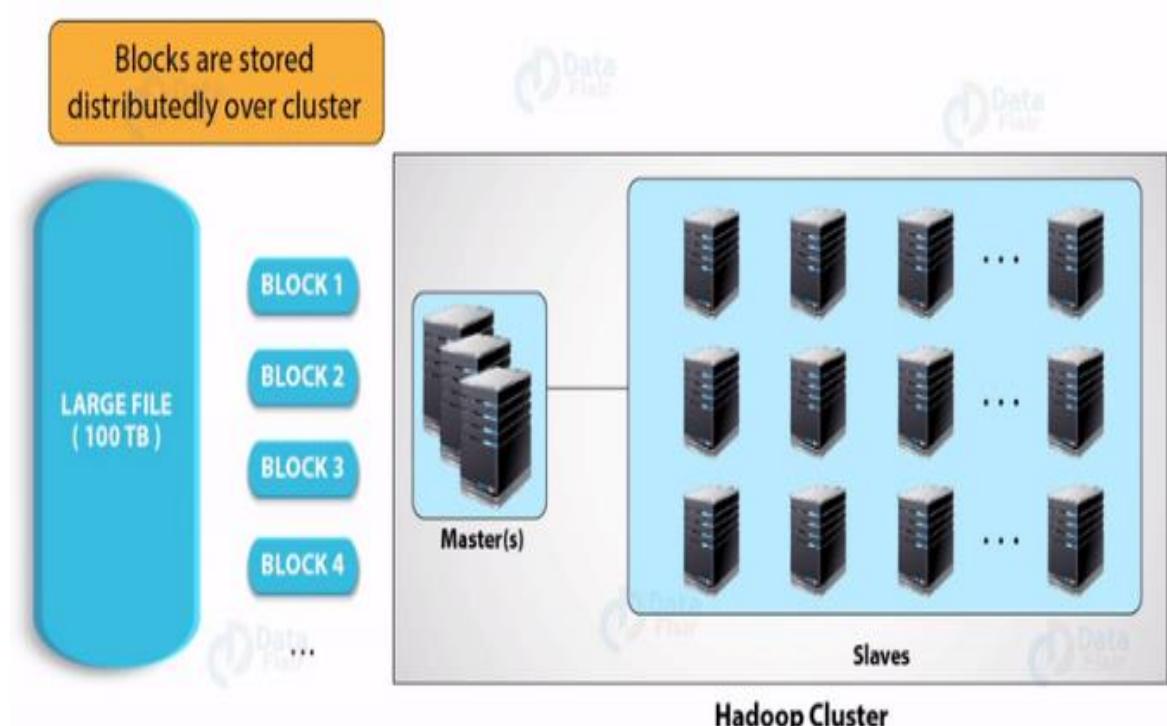
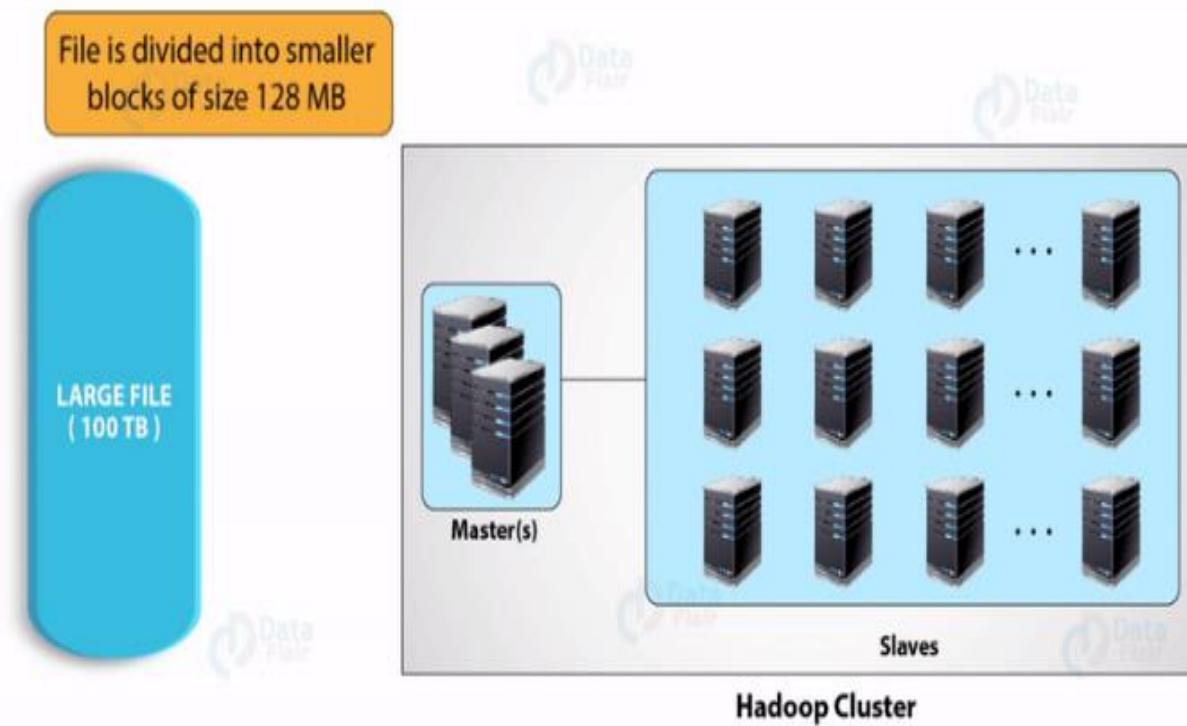
Core Components of Hadoop

- Hadoop consists of three core components –
 - **Hadoop Distributed File System (HDFS)** – It is the storage layer of Hadoop.
 - **Map-Reduce** – It is the data processing layer of Hadoop.
 - **YARN** – It is the resource management layer of Hadoop.

Core Components of Hadoop

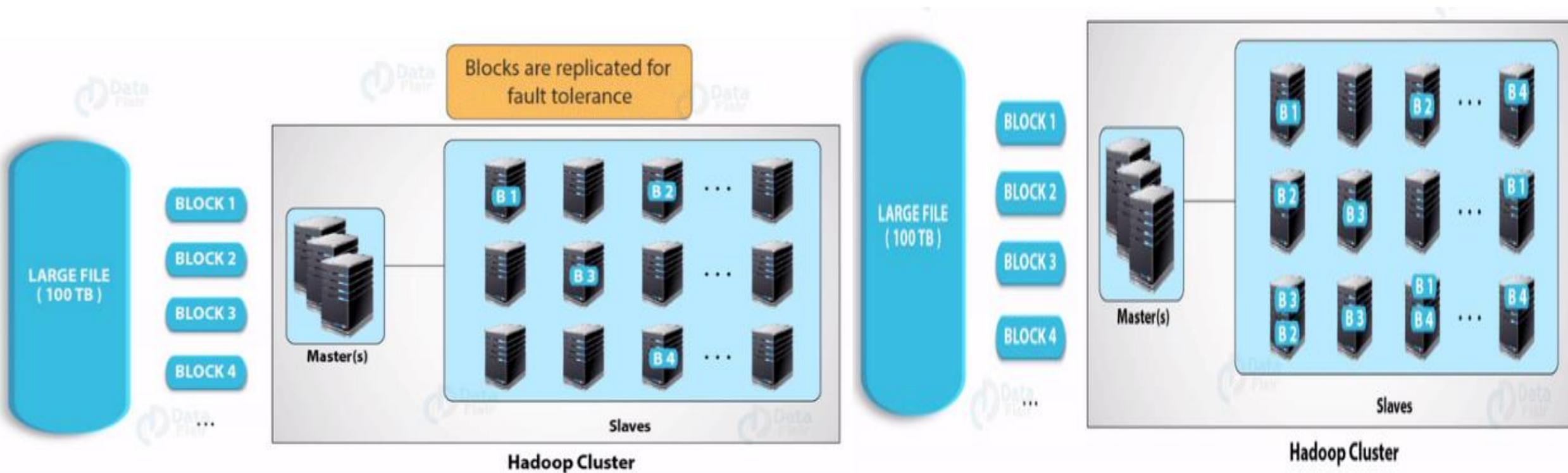
1. HDFS

- Short for Hadoop Distributed File System provides for distributed storage for Hadoop. HDFS has a master-slave topology.



Core Components of Hadoop

- Master is a high-end machine whereas slaves are inexpensive computers. The Big Data files get divided into the number of **blocks**. Hadoop stores these blocks in a **distributed fashion** on the cluster of **slave nodes**. On the master, we have metadata stored.



Core Components of Hadoop

- HDFS has two daemons running for it. They are : NameNode and DataNode
- **NameNode** : Name Node which is used to store metadata about the Data node, placed with the Master Node. They contain details like the details about the slave note, indexing and their respective locations along with timestamps for timelining.
- NameNode is nothing but the master node. The NameNode is responsible for managing file system namespace, controlling the client's access to files. Also, it executes tasks such as opening, closing and naming files and directories. NameNode has two major files – FSImage and Edits log

Core Components of Hadoop

- **FSImage** – FSImage is a point-in-time snapshot of HDFS's metadata. It contains information like file permission, disk quota, modification timestamp, access time, location of the data on the Data Blocks and which blocks are stored on which node, etc.
- **Edits log** – It contains modifications on FSImage. It records incremental changes like renaming the file, appending data to the file, addition of a new block, replication, deletion etc. In short, it records the changes since the last FslImage was created.
- Whenever the NameNode restarts it applies Edits log to FSImage and the new FSImage gets loaded on the NameNode.

Core Components of Hadoop

- Every time the NameNode restarts, EditLogs are applied to FslImage to get the latest snapshot of the file system. But NameNode restarts are rare in production clusters. Because of this, you may encounter the following issues: .
 - EditLog grows unwieldy in size, particularly where the NameNode runs for a long period of time without a restart;
 - NameNode restart takes longer, as too many changes now have to be merged
 - If the NameNode fails to restart (i.e., crashes), there will be significant data loss, as the FslImage used at the time of the restart is very old

Core Components of Hadoop

- **Secondary NameNode**
- Secondary Namenode helps to overcome the above issues by taking over the responsibility of merging EditLogs with FslImage from the NameNode.
- The Secondary NameNode obtains the FslImage and EditLogs from the NameNode at regular intervals.
- Secondary NameNoide loads both the FslImage and EditLogs to main memory and applies each operation from the EditLogs to the FslImage.
- Once a new FslImage is created, Secondary NameNode copies the image back to the NameNode.
- Namenode will use the new FslImage for the next restart, thus reducing startup time.

Core Components of Hadoop

- However, this seemingly fail-proof process is not without issues. Delays in the aforesaid process can cause a NameNode to startup without the latest FslImage at its disposal. Such delays can occur if:
 - The Secondary NameNode takes too long to download the EditLogs from the NameNode;
 - The NameNode is slow in uploading FslImages to the Secondary NameNode and/or in downloading the updated FslImages from the Secondary NameNode
- To avoid such delays, administrators will have to closely monitor the communication between the NameNode and Secondary NameNode, proactively detect any slowness in the upload and/or download of FslImages / EditLogs, and promptly initiate measures to isolate and remove the source of the slowness. This is where the Hadoop FS Image EditLogs test helps!
- This test monitors the following:
 - How quickly the Secondary NameNode downloads EditLogs from the NameNode;
 - How quickly the NameNode uploads and downloads FslImages from the Secondary NameNode

Core Components of Hadoop

- NameNode performs following functions –
 - NameNode Daemon runs on the master machine.
 - It is responsible for maintaining, monitoring and managing DataNodes.
 - It records the metadata of the files like the location of blocks, file size, permission, hierarchy etc.
 - Namenode captures all the changes to the metadata like deletion, creation and renaming of the file in edit logs.
 - It regularly receives heartbeat and block reports from the DataNodes.

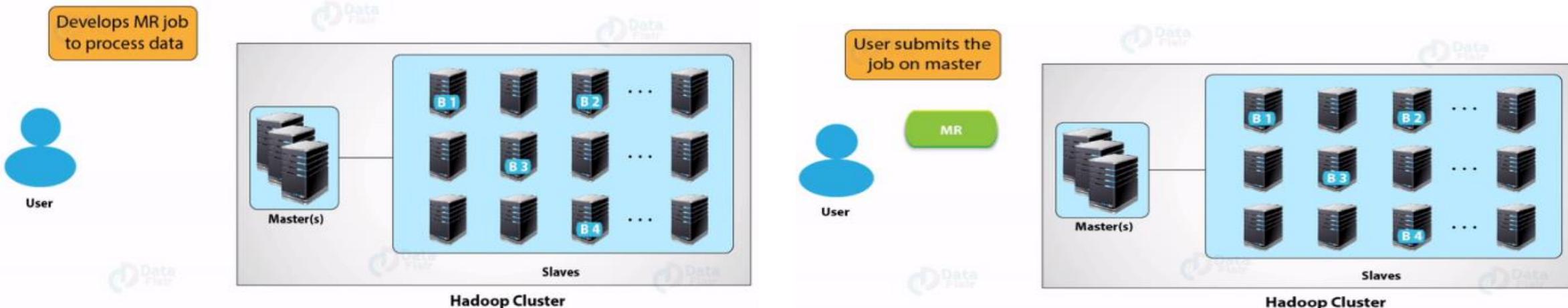
Core Components of Hadoop

- **DataNode:** Data Nodes used for storage of data related to the applications in use placed in the Slave Nodes.
- The various functions of DataNode are as follows –
 - DataNode runs on the slave machine.
 - It stores the actual business data.
 - It serves the read-write request from the user.
 - DataNode does the ground work of creating, replicating and deleting the blocks on the command of NameNode.
 - After every 3 seconds, by default, it sends heartbeat to NameNode reporting the health of HDFS.
- In other words, a node which knows where the files are to be found in hdfs are Namenode, and the node which have the data of the files are Datanodes.

Core Components of Hadoop

2. MapReduce

- It is the data processing layer of Hadoop. It processes data in two phases. They are:-
- **Map Phase-** This phase applies business logic to the data. The input data gets converted into key-value pairs.
- **Reduce Phase-** The Reduce phase takes as input the output of Map Phase. It applies aggregation based on the key of the key-value pairs.

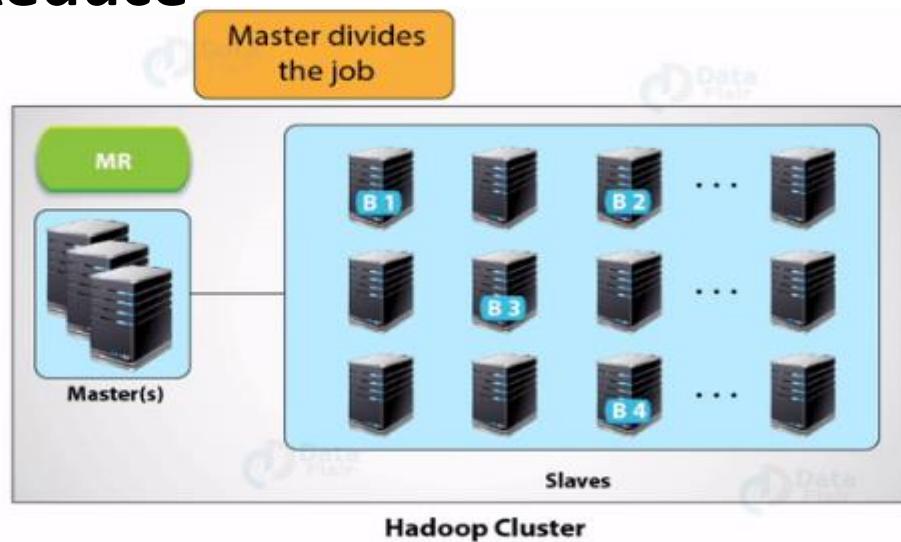


Core Components of Hadoop

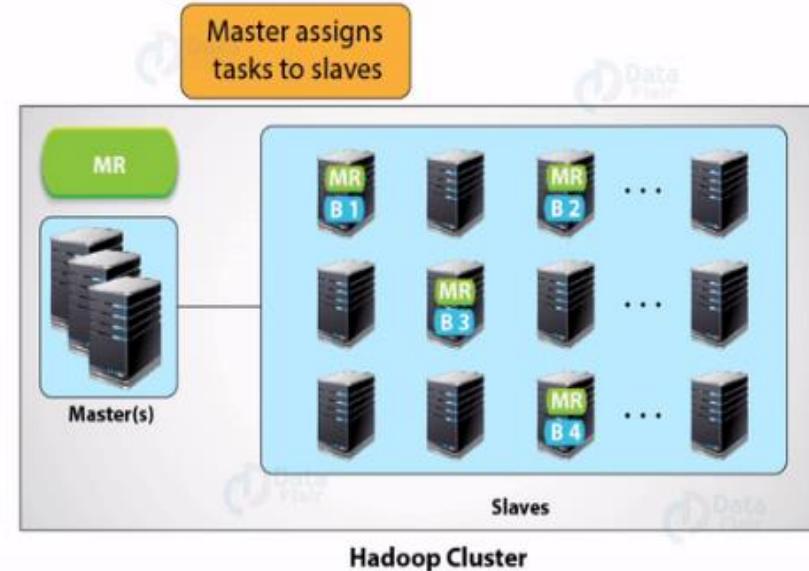
- **MapReduce**



Data
Flair

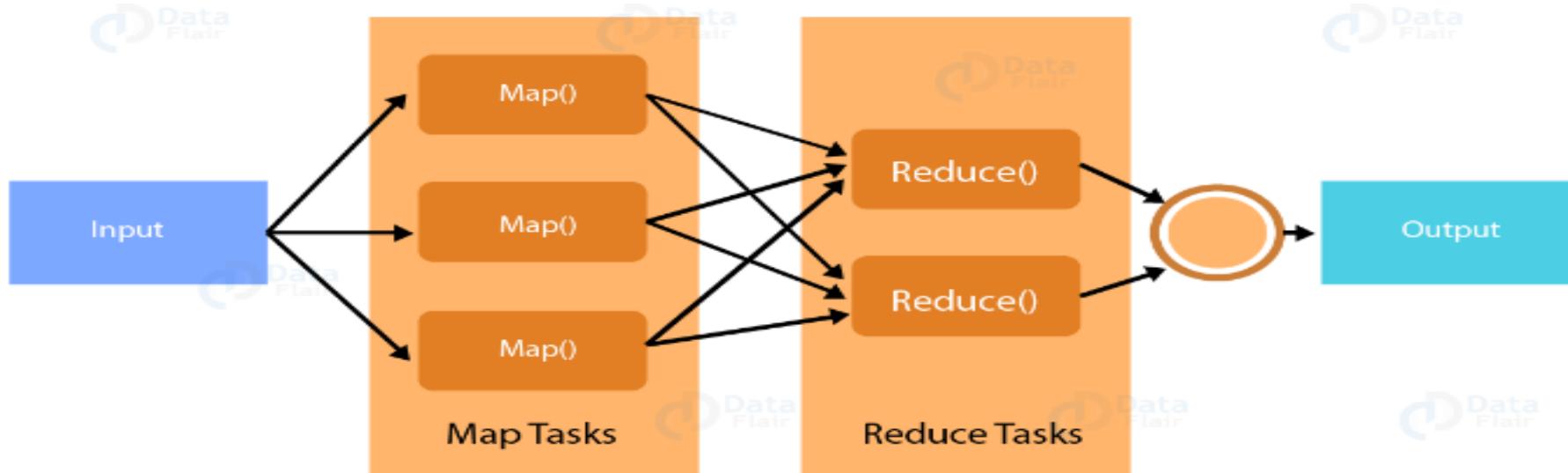


Data
Flair



Core Components of Hadoop

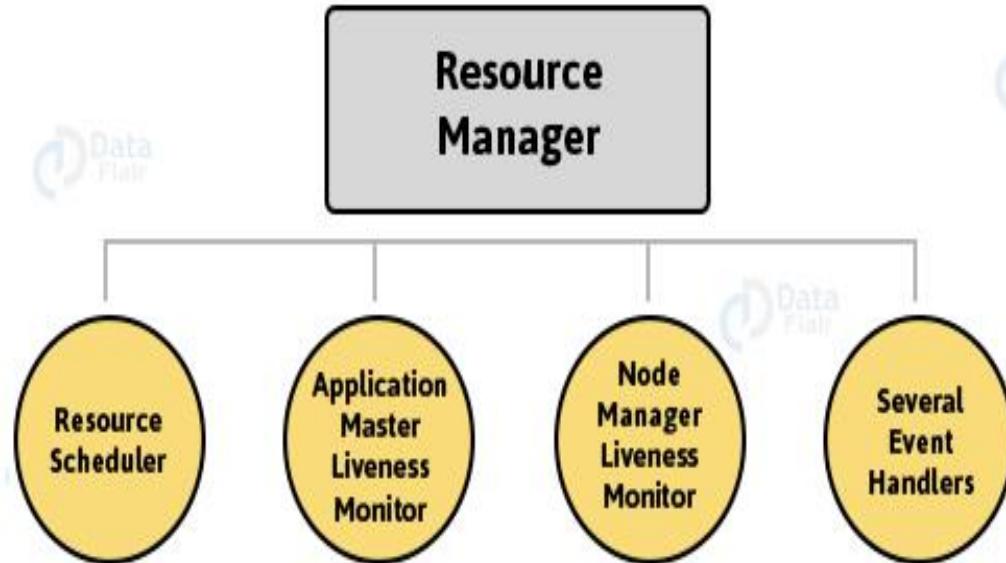
- **MapReduce**



- Mapper reads the block of data and converts it into key-value pairs.
- Now, these key-value pairs are input to the reducer.
- The reducer receives data tuples from multiple mappers.
- Reducer applies aggregation to these tuples based on the key.
- The final output from reducer gets written to HDFS.
- MapReduce framework takes care of the failure. It recovers data from another node in an event where one node goes down.

Core Components of Hadoop

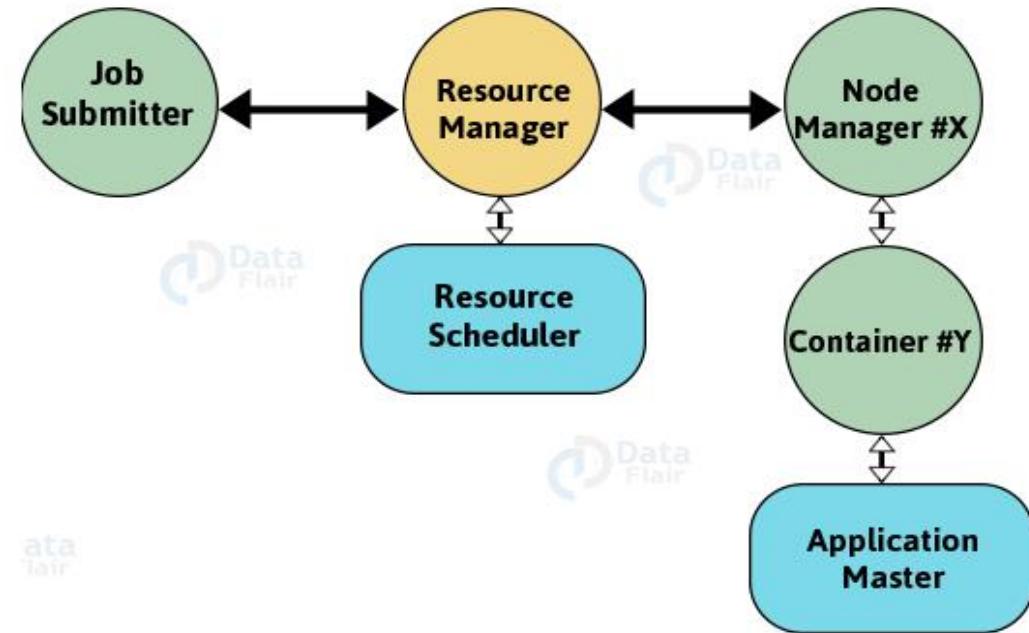
3. YARN (Yet Another Resource Negotiator)



- **Resource Manager** runs on the **master node**.
- It knows where the location of slaves.
- It is aware about how much resources each slave have.
- **Resource Scheduler** and **Application Manager** is one of the important **service** run by the **Resource Manager**.
- Resource Scheduler decides how the resources get assigned to various tasks.
- Application Manager negotiates the first container for an application.
- Resource Manager keeps track of the heart beats from the Node Manager.

Core Components of Hadoop

3. YARN (Yet Another Resource Negotiator)



- The application start up process is as follows:-
 - The client submits the **job** to **Resource Manager**.
 - Resource Manager contacts **Resource Scheduler** and allocates **container**.
 - Now Resource Manager contacts the relevant **Node Manager** to launch the **container**.
 - Container runs Application Master.

Core Components of Hadoop

3. YARN (Yet Another Resource Negotiator)

- The basic idea of YARN was to **split the task** of resource management and job scheduling. It has one global Resource Manager and per-application Application Master. An application can be either one job or DAG of jobs.
- The Resource Manager's job is to assign resources to various competing applications. **Node Manager** runs on the **slave** nodes. It is responsible for **containers, monitoring resource utilization and informing** about the same to Resource Manager.
- The job of Application master is to negotiate resources from the Resource Manager. It also works with NodeManager to execute and monitor the tasks.

Core Components of Hadoop

- Daemons are the processes that run in the background. The Hadoop Daemons are:-
 - a) Namenode – It runs on master node for HDFS.
 - b) Datanode – It runs on slave nodes for HDFS.
 - c) Resource Manager – It runs on YARN master node for MapReduce.
 - d) Node Manager – It runs on YARN slave node for MapReduce.

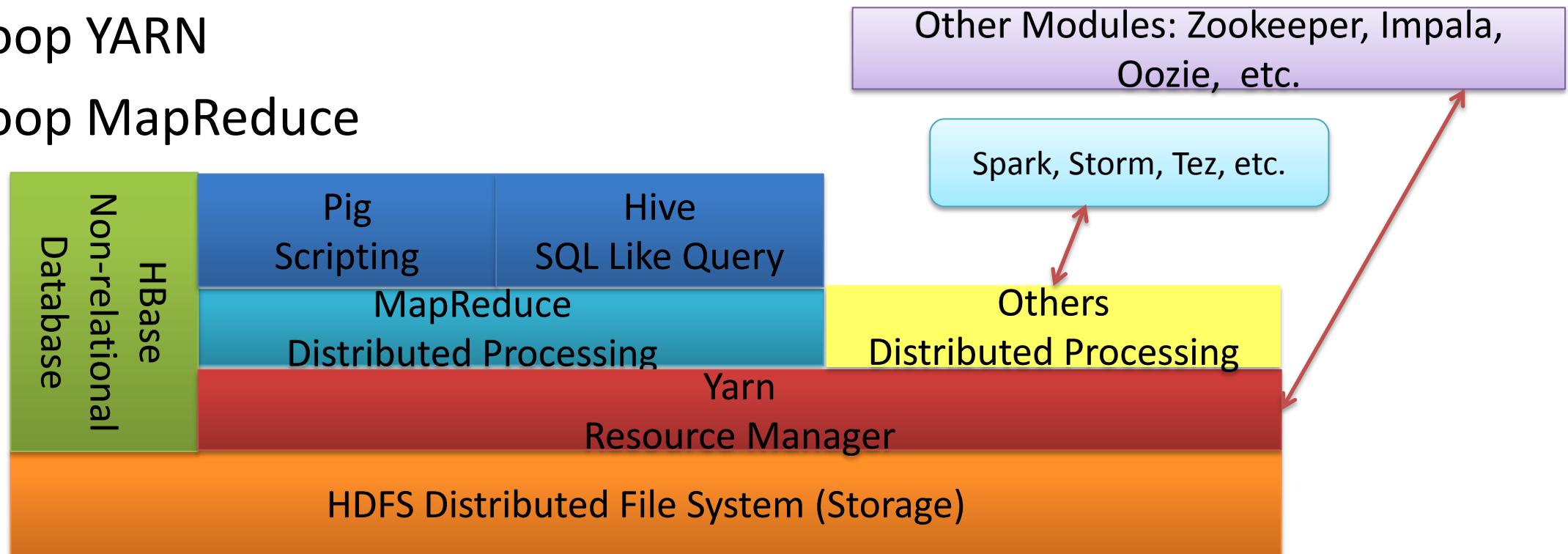
These 4 daemons run for Hadoop to be functional.

How Hadoop Works?

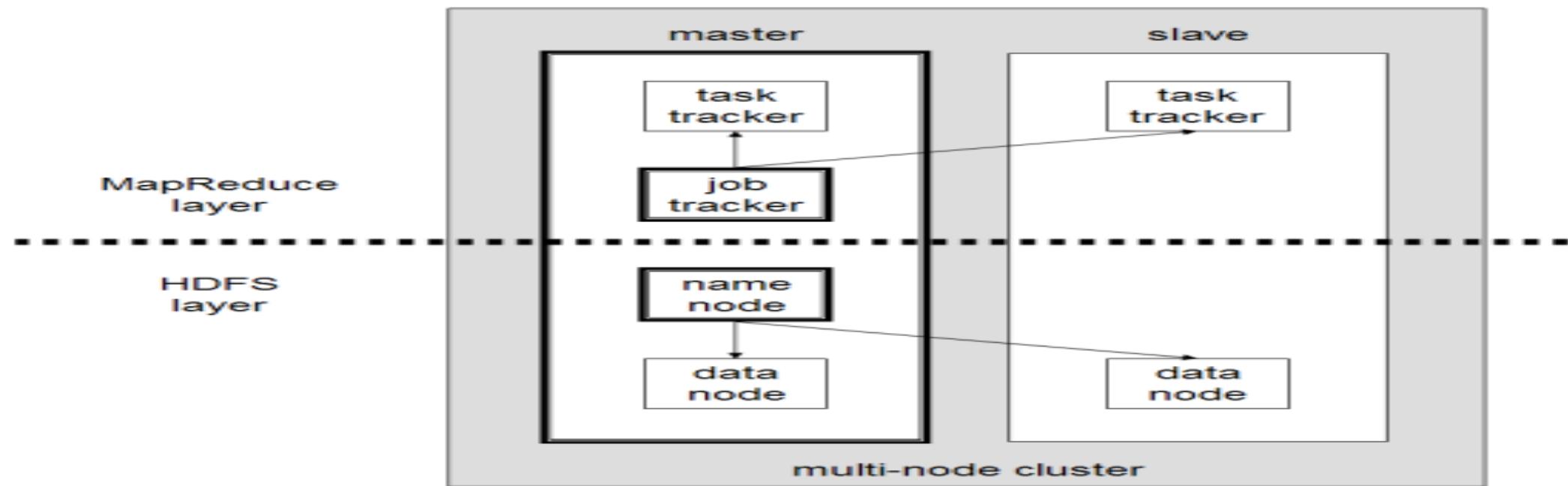
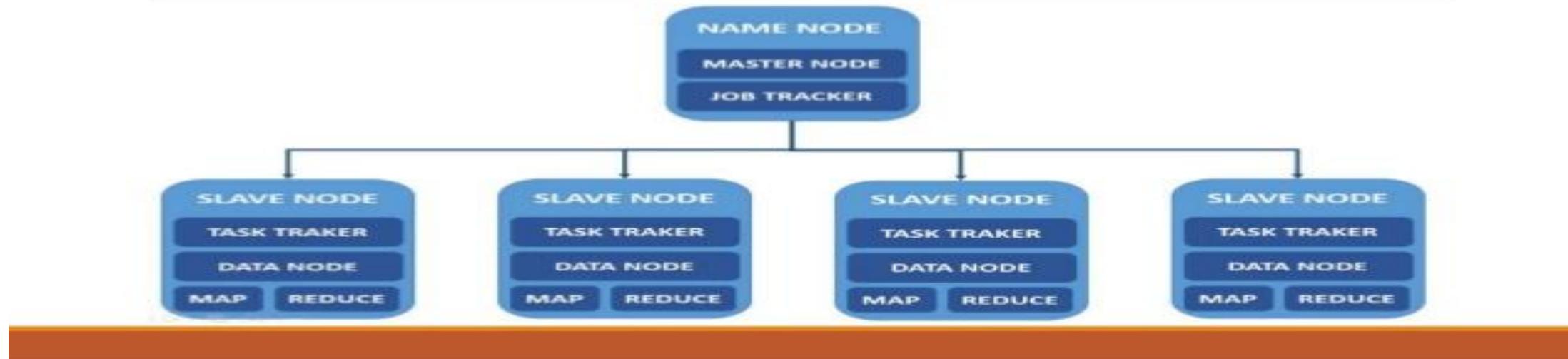
- Summarize how Hadoop works step by step:
 - Input data is broken into blocks of size 128 Mb and then blocks are moved to different nodes.
 - Once all the blocks of the data are stored on data-nodes, the user can process the data.
 - Resource Manager then schedules the program (submitted by the user) on individual nodes.
 - Once all the nodes process the data, the output is written back to HDFS.

Apache Hadoop Basic Modules

- Hadoop Common
- Hadoop Distributed File System (HDFS)
- Hadoop YARN
- Hadoop MapReduce



HADOOP MASTER/SLAVE ARCHITECTURE





Hadoop Core Components:
HDFS
MapReduce
YARN (MapReduce 2.0)

Hadoop Essential
Pig (Pig Latin + Pig Runtime) Hive or HQL, HBase (Google's Big table), Cassandra (Amazon's Dynamo) Avro, Zookeeper, Sqoop, Mahout (machine Learning) ...

Hadoop Incubator
Chukwa, Ambari, HDT Hcatalog, Knox, Spark, Kafka (Data Ingestion layer) Storm (Data analytic layer) Samza, Hama, Nutch ...

Training Data Sets

Algorithms

Dashboard

Administration

Platform ETL Tools

Platform Functions

Platform Core Units

Data Processing Modules

Platform Admin Modules



③ Hadoop Framework, V1 and V2
Map Reduce Hadoop File Distributed System (HFDS)
YARN (Map Reduce 2.0)



Sqoop & Hiho
(Structured Data)

Flume & Scribe
(Unstructured data)

Chukwa
(log)

Pentaho, Talend

ETL

Knox
Security, AAA

Data Integration

Map Reduce programming
Data Storage, Data Library Meta Data Store
Data Interaction Visualization Execution Development
Data Serialization
Data Intelligence
Data Analysis with SQL



② Pig, Hive, Java, Perl

Parquet HBase, Cassandra DataFu, Whirr (cloud), Sentry, Nutch, Solr, Gora MongoDB

Hcatalog, Lucene Hama, Crunch

Avro, Thrift

Drill (Dremel), Mahout

Impala, Spark, Hive

Oozie (workflow)

Zookeeper (configure)

Ambari, CDH (provisioning)

Hue (User Experiences)

Nagios, Ganglia Monitoring

Splunk, Talend Report

HDT, development

Management, Monitoring, Orchestration

Operation Systems (Windows, Linux)

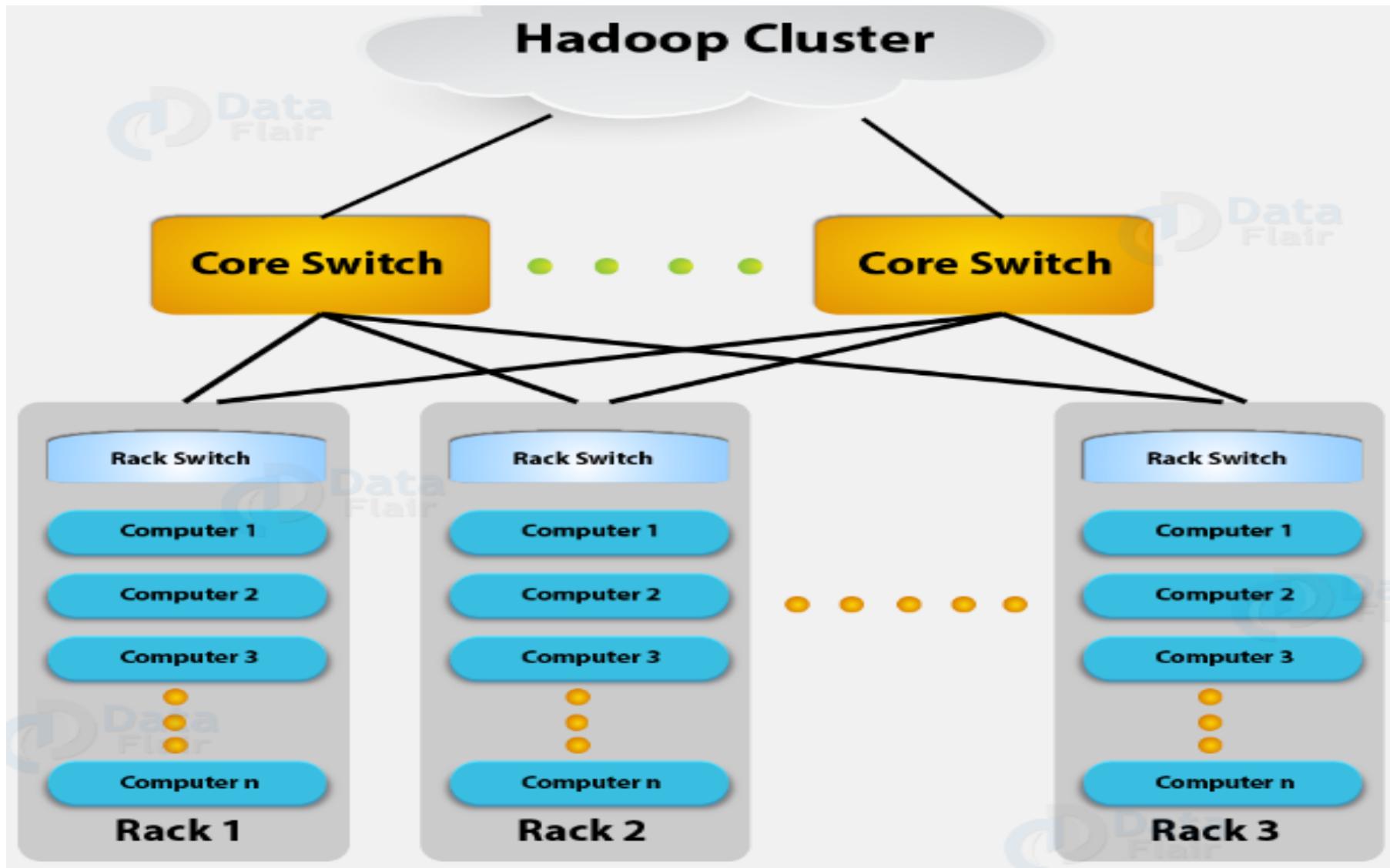
Virtualized Infrastructure (Virtual Machine, J2EE container)



Hadoop Cluster

- Cluster is a set of connected computers which work together as a single system. Similarly, the Hadoop cluster is just a computer cluster which we use for Handling huge volume of data distributedly.
- **Hadoop Cluster Architecture**
- Basically, for the purpose of **storing** as well as **analyzing huge amounts of unstructured data** in a **distributed computing environment**, a special type of computational cluster is designed that what we call as Hadoop Clusters.
- Hadoop Clusters: two main terms come up, they are cluster and node, so on defining them:
 - A collection of nodes is what we call the cluster.
 - A node is a point of intersection/connection within a network, ie a server

Hadoop Cluster Architecture



Hadoop Cluster

- There is nothing shared between the nodes in a Hadoop cluster except for the network which connects them (Hadoop follows shared-nothing architecture). This feature **decreases the processing latency** so the cluster-wide latency is minimized when there is a need to process queries on huge amounts of data.
- In addition, Hadoop clusters have two types of machines, such as Master and Slave, where:
 - **Master: HDFS NameNode, YARN ResourceManager.**
 - **Slaves: HDFS DataNodes, YARN NodeManagers.**

Hadoop Cluster

- However, it is recommended to separate the master and slave node, because:
 - Task/application workloads on the slave nodes should be isolated from the masters.
 - Slaves nodes are frequently decommissioned for maintenance.
- Moreover, it is possible to scale out a Hadoop cluster. Here, Scaling means to add more nodes. That's why we also call it **linearly scalable**. Hence, we get a corresponding boost in throughput, for every node we add.

Hadoop Cluster

- **The Communication Protocols**

- For inter-node communication, Hadoop uses **tcp ip**. Basically, on top of the TCP/IP protocol, all HDFS communication protocols are layered and on the NameNode machine, a client establishes a connection to a configurable TCP port. Moreover, with the NameNode it talks the ClientProtocol. And, by using the DataNode Protocol, the DataNodes talk to the NameNode.
- In addition, the abstraction Remote Procedure Call (RPC) wraps both the DataNode Protocol as well as the Client Protocol. Although, the NameNode never starts any RPCs, by design. Rather than that it only responds to RPC requests which are issued by DataNodes or clients.

Hadoop Cluster

- **Hadoop Nodes Configuration**
- However, by two types of important configuration files, Hadoop's Java configuration is driven:
- Read-only default configuration : core-default.xml, hdfs-default.xml, yarn-default.xml and mapred-default.xml.
- Site-specific configuration : etc/hadoop/core-site.xml, etc/hadoop/hdfs-site.xml, etc/hadoop/yarn-site.xml and etc/hadoop/mapred-site.xml.
- Moreover, by setting site-specific values via above files and etc/hadoop/hadoop-env.sh and etc/hadoop/yarn-env.sh, we can control the Hadoop scripts found in the bin/ directory of the distribution.

Hadoop Cluster

- **Advantages of a Hadoop Cluster**

- The cluster helps in increasing the speed of the analysis process.
- It is inexpensive.
- These clusters are failure resilient.
- One more benefit to Hadoop clusters is “scalability”, it means Hadoop offers Scalable and flexible Data Storage. Here Scalability means, we can scale a Hadoop cluster by adding new servers to the cluster if needed.
- Hadoop Clusters deal with data from many sources and formats in a very quick, easy manner.
- It is possible to deploy Hadoop using a single-node installation, for evaluation purposes.

Hadoop Features

- **Reliability**
 - In the Hadoop cluster, if any node goes down, it will not disable the whole cluster. Instead, another node will take the place of the failed node. Hadoop cluster will continue functioning as nothing has happened. Hadoop has built-in fault tolerance feature.
- **Scalable**
 - Hadoop gets integrated with cloud-based service. If you are installing Hadoop on the cloud you need not worry about scalability. You can easily procure more hardware and expand your Hadoop cluster within minutes.

Hadoop Features

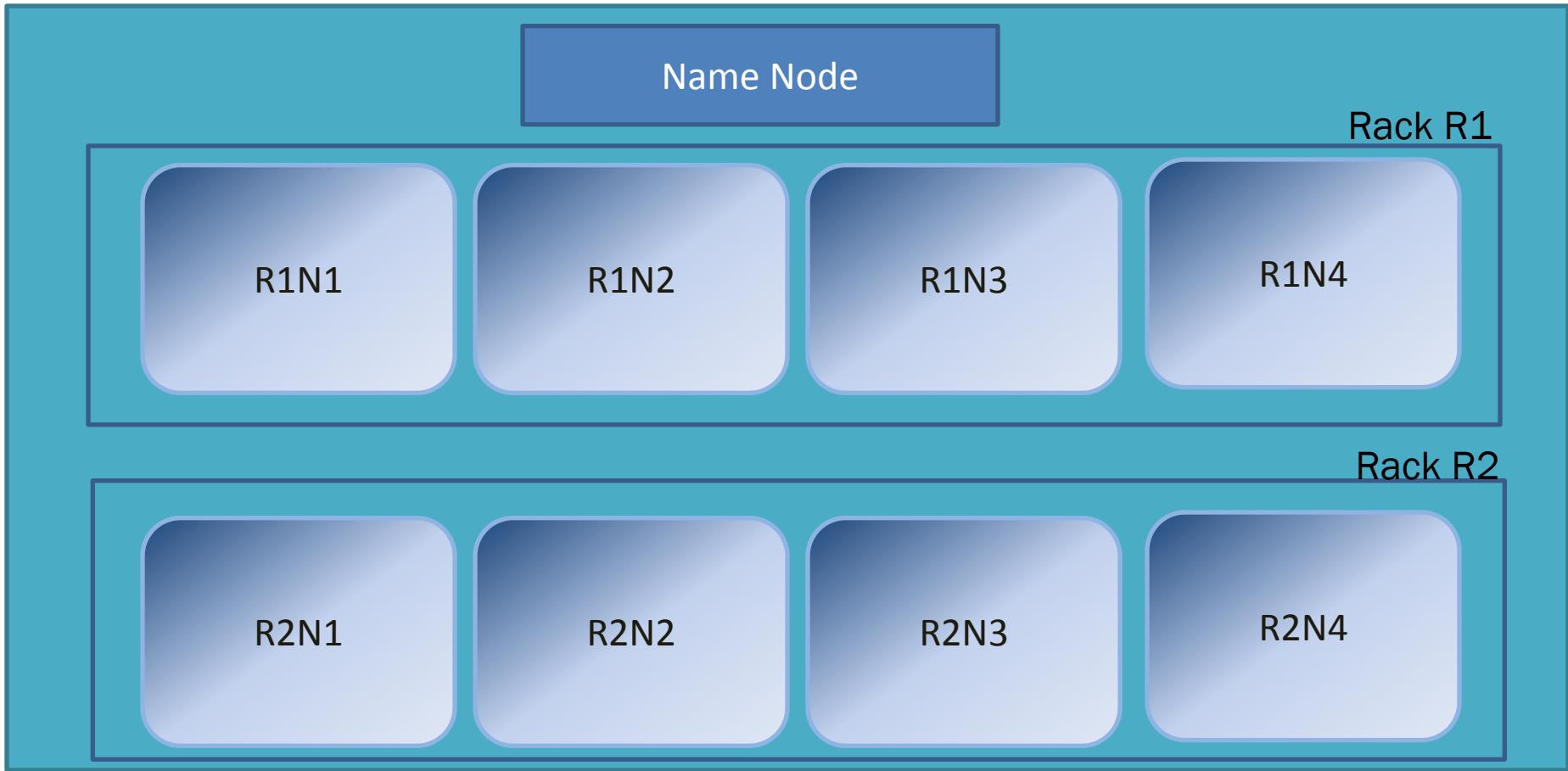
- **Economical**
 - Hadoop gets deployed on commodity hardware which is cheap machines. This makes Hadoop very economical. Also as Hadoop is an open system software there is no cost of license too.
- **Distributed Processing**
 - In Hadoop, any job submitted by the client gets divided into the number of sub-tasks. These sub-tasks are independent of each other. Hence they execute in parallel giving high throughput.

Hadoop Features

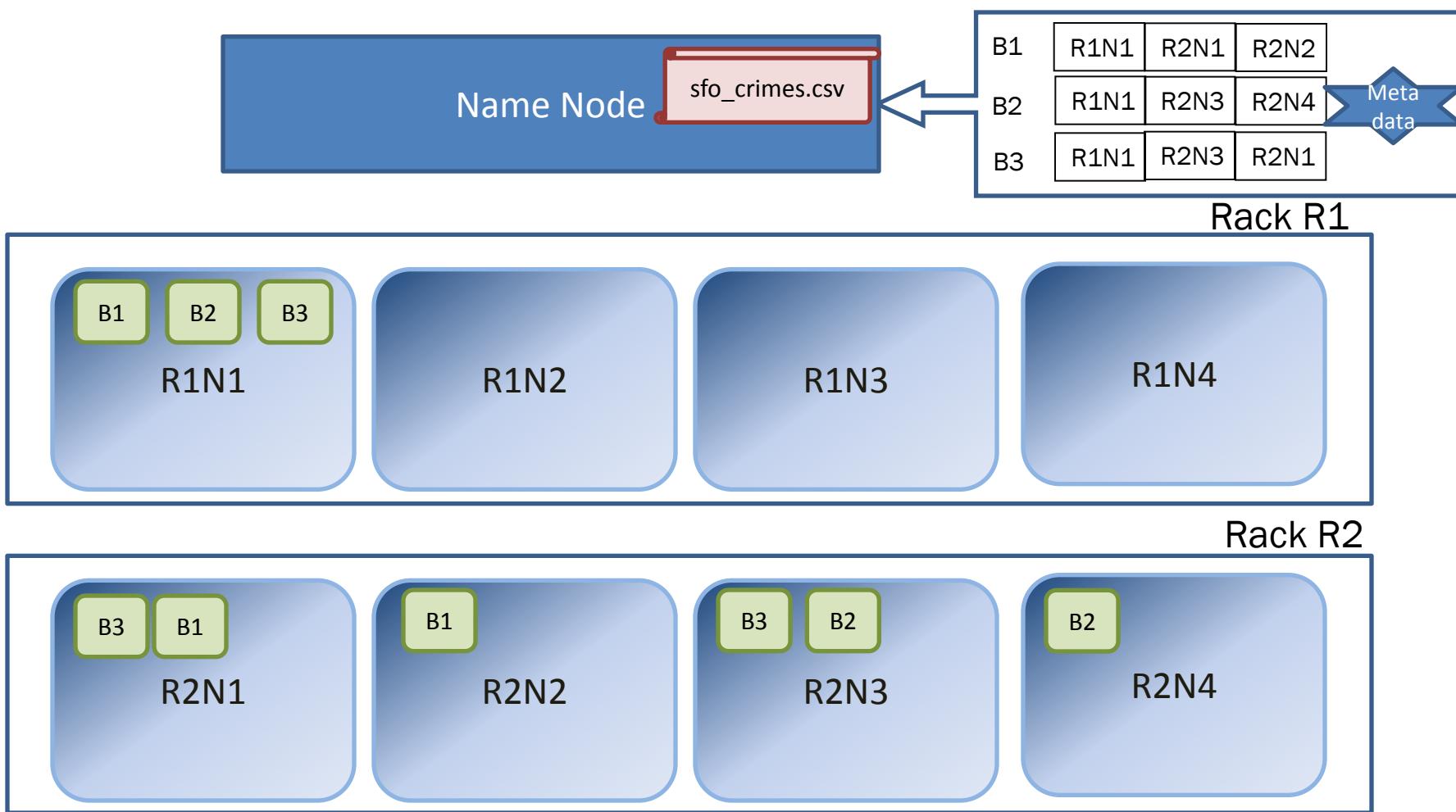
- **Distributed Storage**
 - Hadoop splits each file into the number of blocks. These blocks get stored distributedly on the cluster of machines.
- **Fault Tolerance**
 - Hadoop replicates every block of file many times depending on the replication factor. Replication factor is 3 by default. In Hadoop suppose any node goes down then the data on that node gets recovered. This is because this copy of the data would be available on other nodes due to replication. Hadoop is fault tolerant.

Thank you.

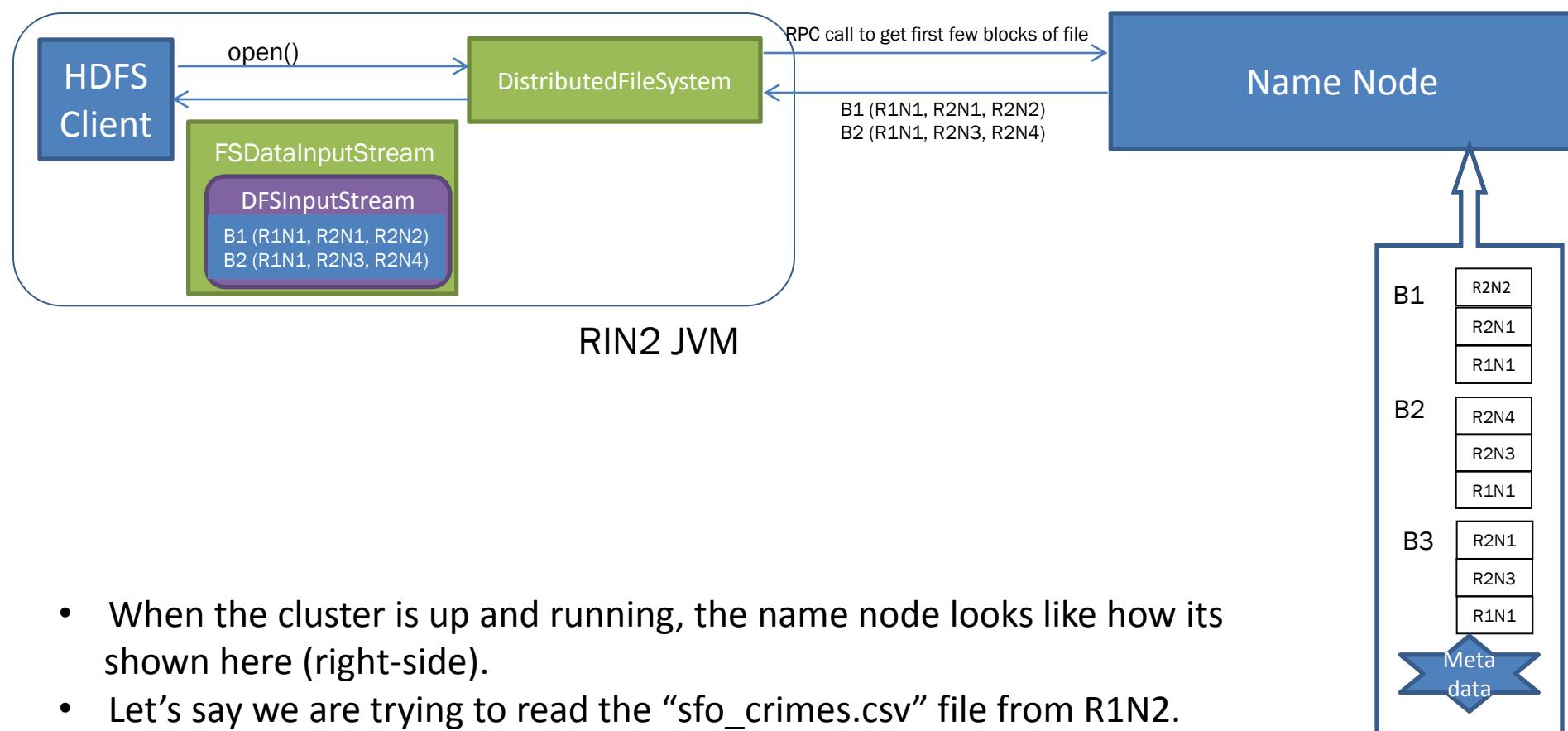
Anatomy of file read in Hadoop



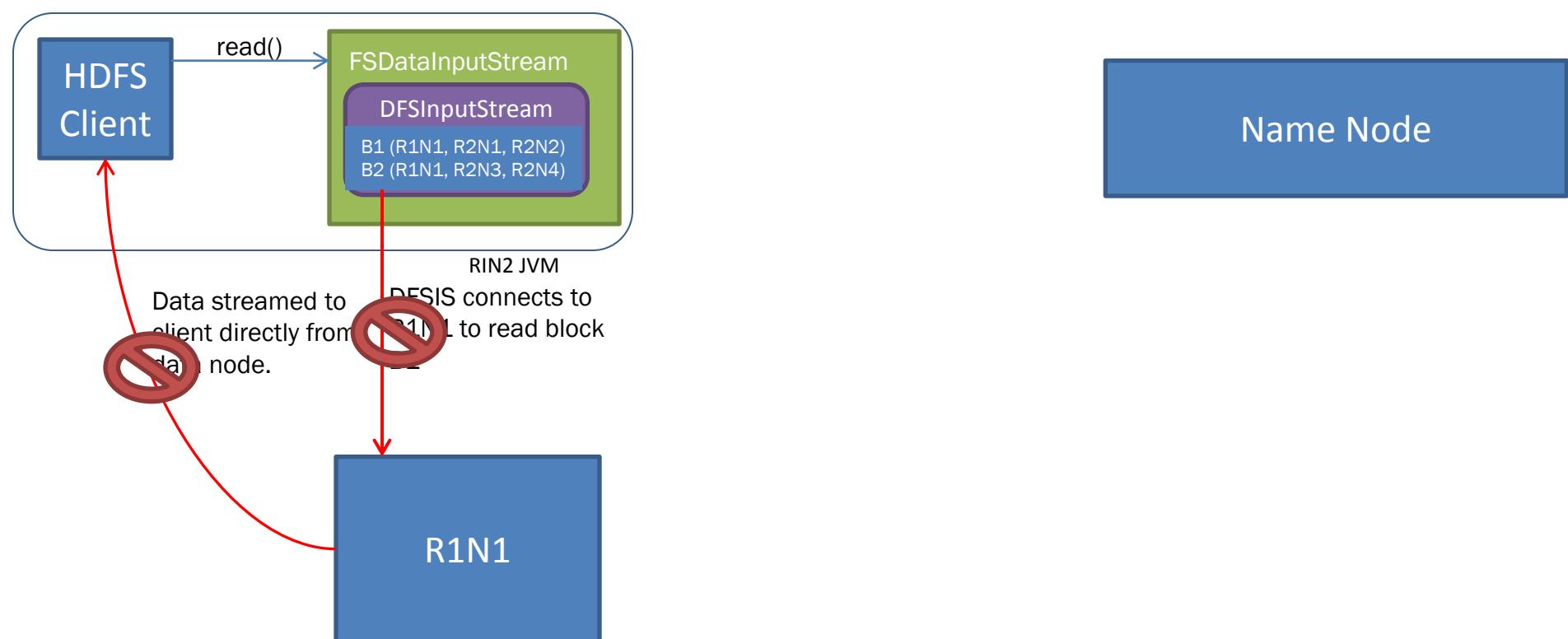
1. This is a Hadoop cluster with one name node and two racks named R1 and R2 in a data center D1. Each rack has 4 nodes and they are uniquely identified as R1N1, R1N2 and so on.
2. Replication factor is 3.
3. HDFS block size is 64 MB.
4. This cluster is used as an example to explain the concepts.



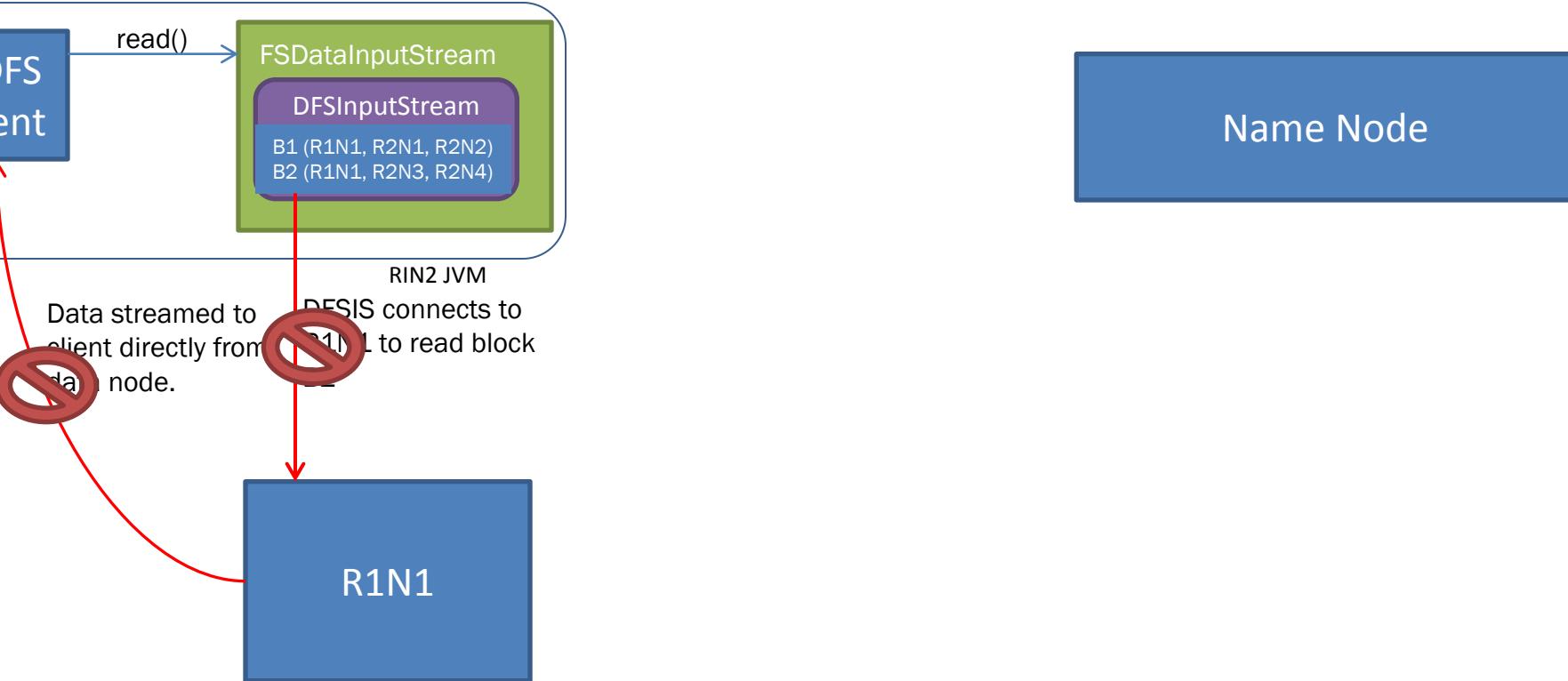
- Let's assume a file named "sfo_crime.csv" of size 192 MB is saved in this cluster.
- Also assume that the file was written from node R1N1.
- Metadata is written in name node.
- The file is split into 3 blocks each of size 64 MB. And each block is copied 3 times in the cluster.
- Along with data, a checksum will be saved in each block. This is used to ensure the data read from the block is read without error.
- When cluster is started, the metadata will look as shown on top right corner.



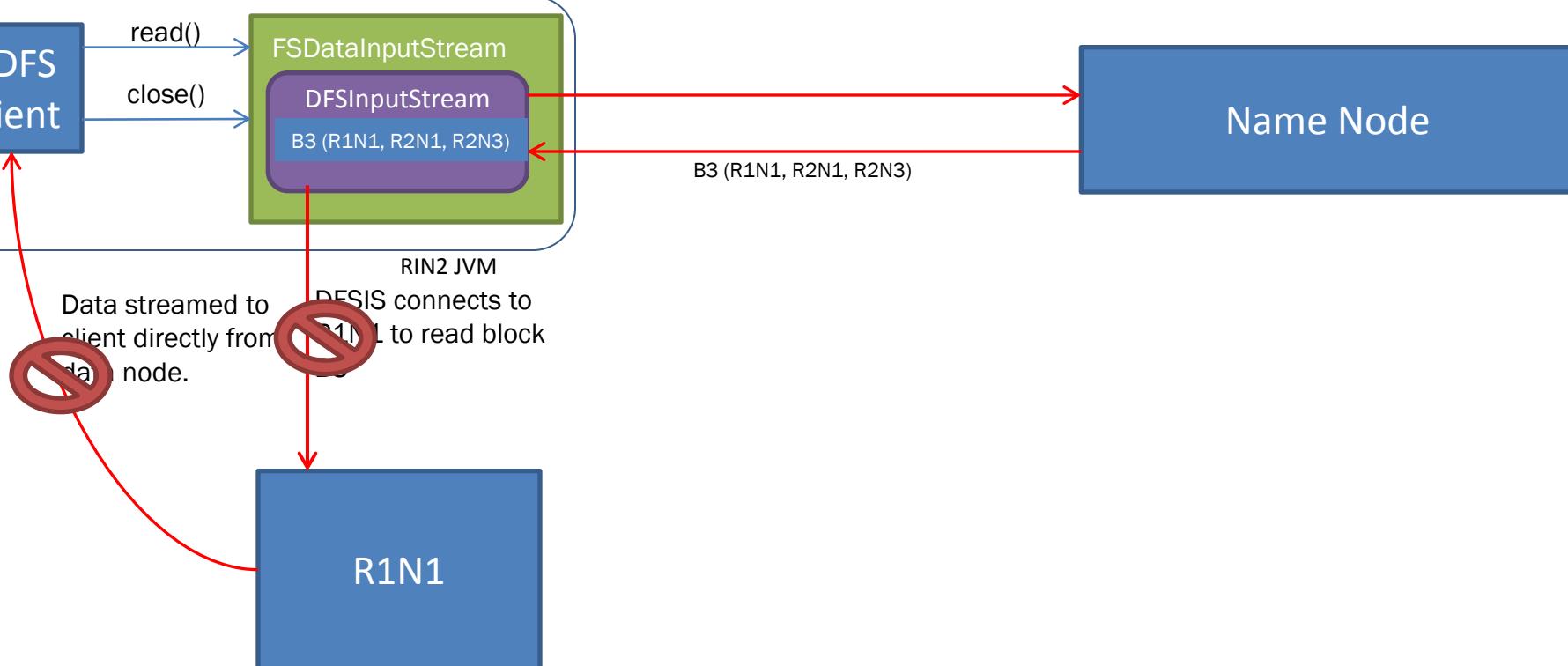
- When the cluster is up and running, the name node looks like how its shown here (right-side).
- Let's say we are trying to read the “sfo_crimes.csv” file from R1N2.
- So a HDFS Client program will run on R1N2’s JVM.
- First the HDFS client program calls the method `open()` on a Java class `DistributedFileSystem` (subclass of `FileSystem`).
- DFS makes a RPC call returns first few blocks on the file. NN returns the address of the DN ORDERED with respect to the node from where the read is performed.
- The block information is saved in `DFSInputStream` which is wrapped in `FSDataInputStream`.
- In response to ‘`FileSystem.open()`’, HDFS Client receives this `FSDataInputStream`.



- From now on HDFS Client deals with **FSDataInputStream** (FSDIS).
- HDFS Client invokes `read()` on the stream.
- Blocks are read in order. DFSIS connects to the closest node (**R1N1**) to read block **B1**.
- DFSIS connects to data node and streams data to client, which calls `read()` repeatedly on the stream. DFSIS verifies checksums for the data transferred to client.
- When the block is read completely, DFSIS closes the connection.

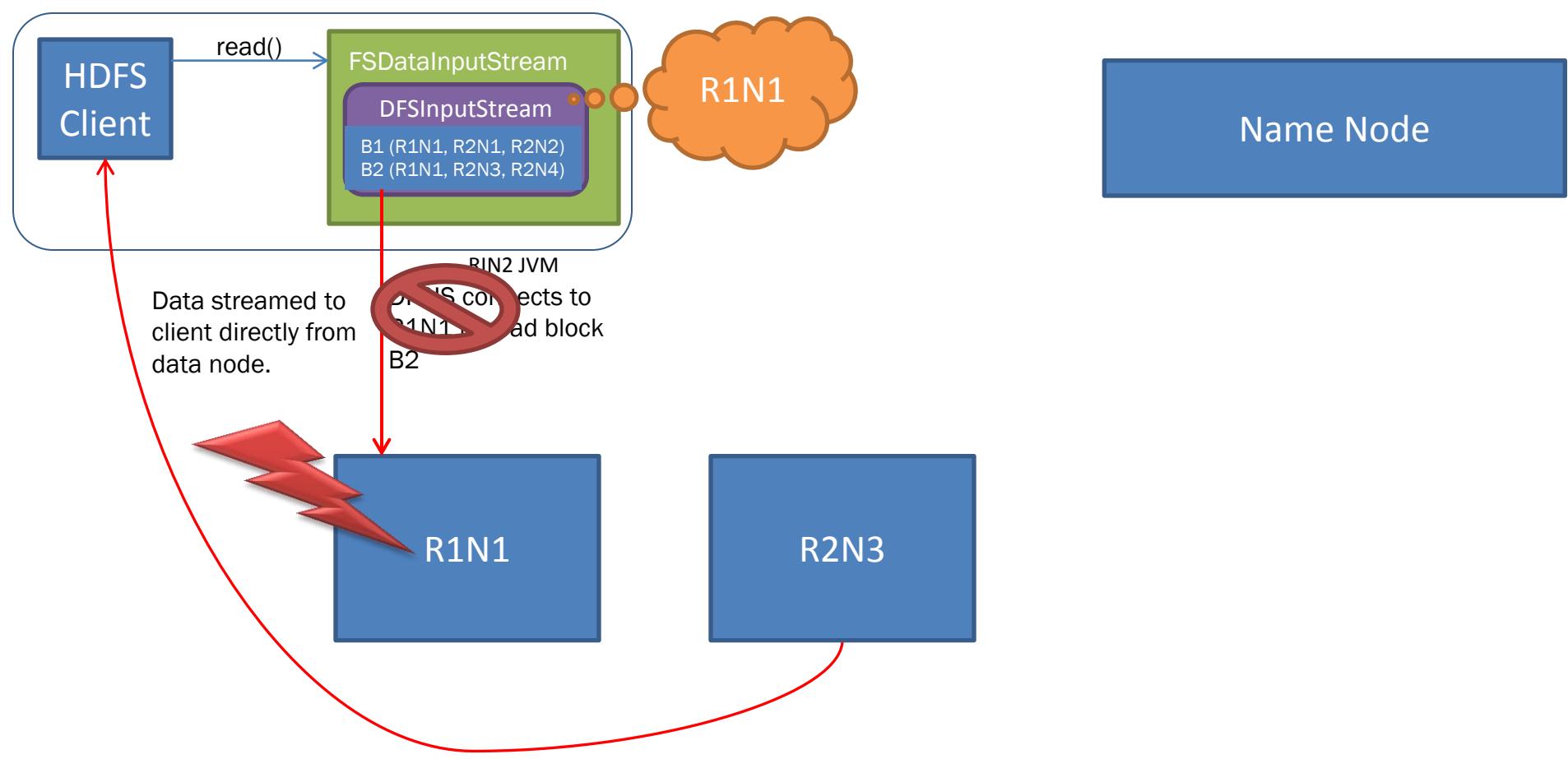


- Next DFSIS attempts to read block B2. As mentioned earlier, the previous connection is closed and a fresh connection is made to the closest node (R1N1) of block B2.



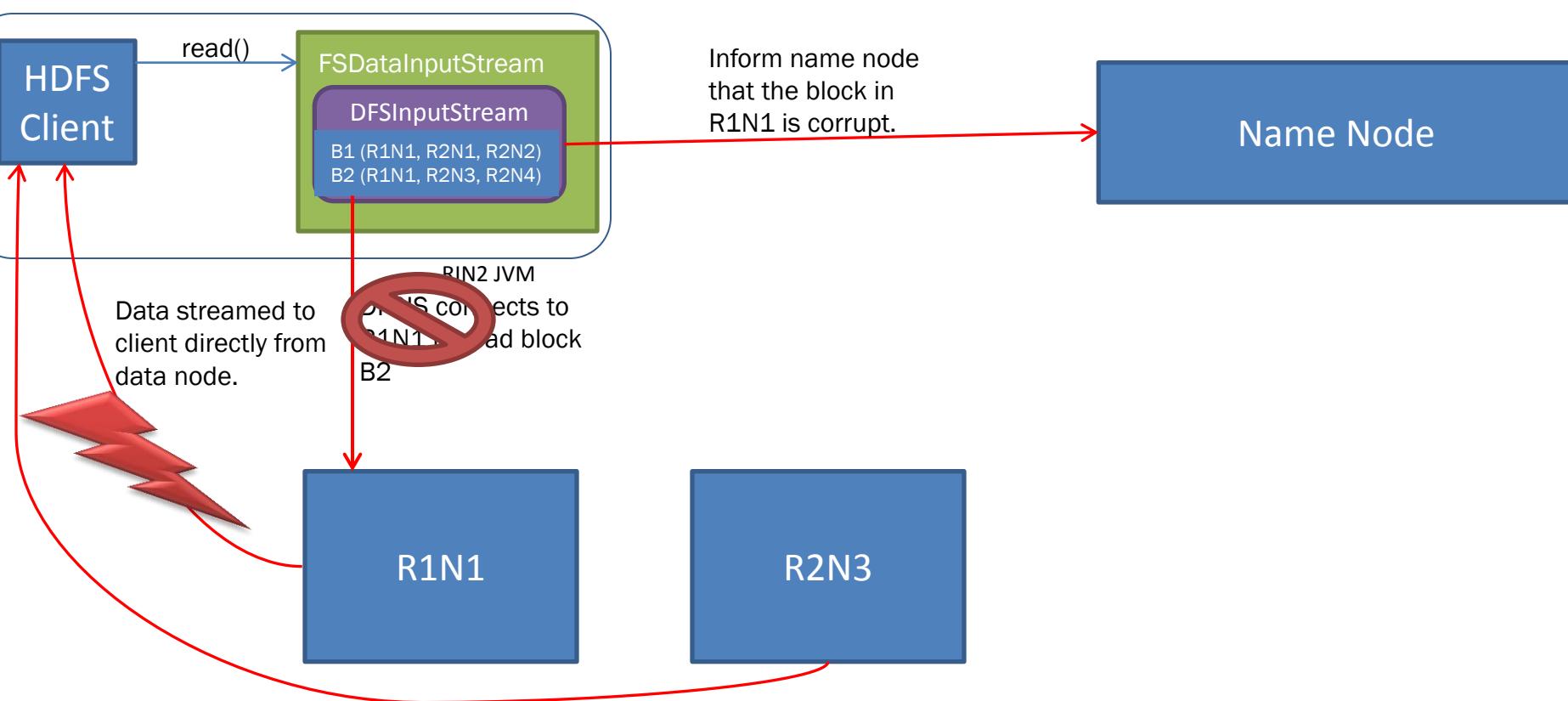
- Now DFSIS has read all blocks returned by the first RPC call (B1 & B2). But the file is not read completely. In our case there is one more block to read.
- DFSIS calls name node to get data node locations for next batch of blocks as needed.
- After the complete file is read for the HDFS client call `close()`.

Anatomy of file read – Data Node Connection Error



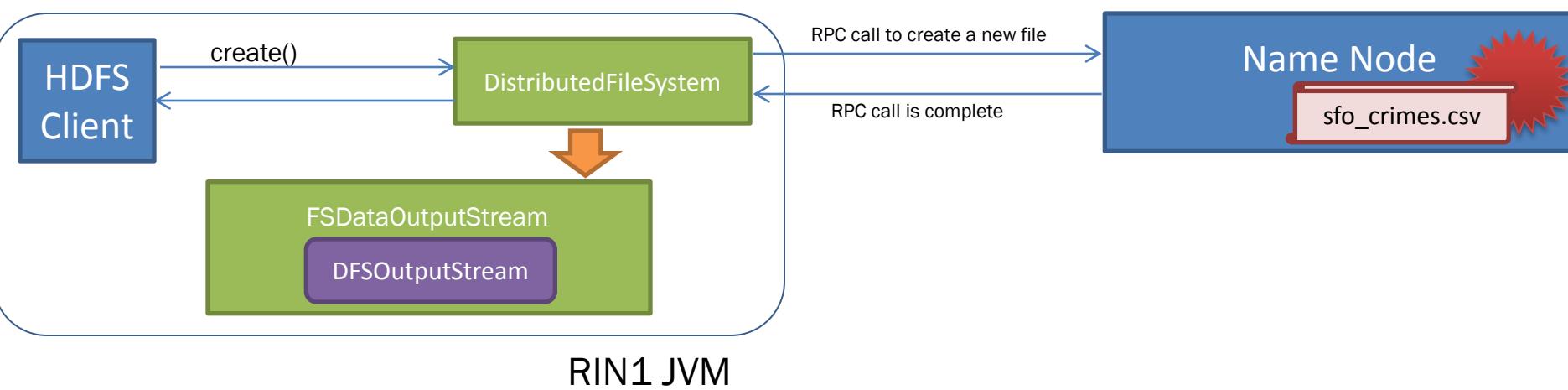
- Let's say there is some error while connecting to R1N1.
- DFSIS remembers this info, so it won't try to read from R1N1 for future blocks. Then it tries to connect to next closest node (R2N3).

Anatomy of file read – Data Node Checksum Error

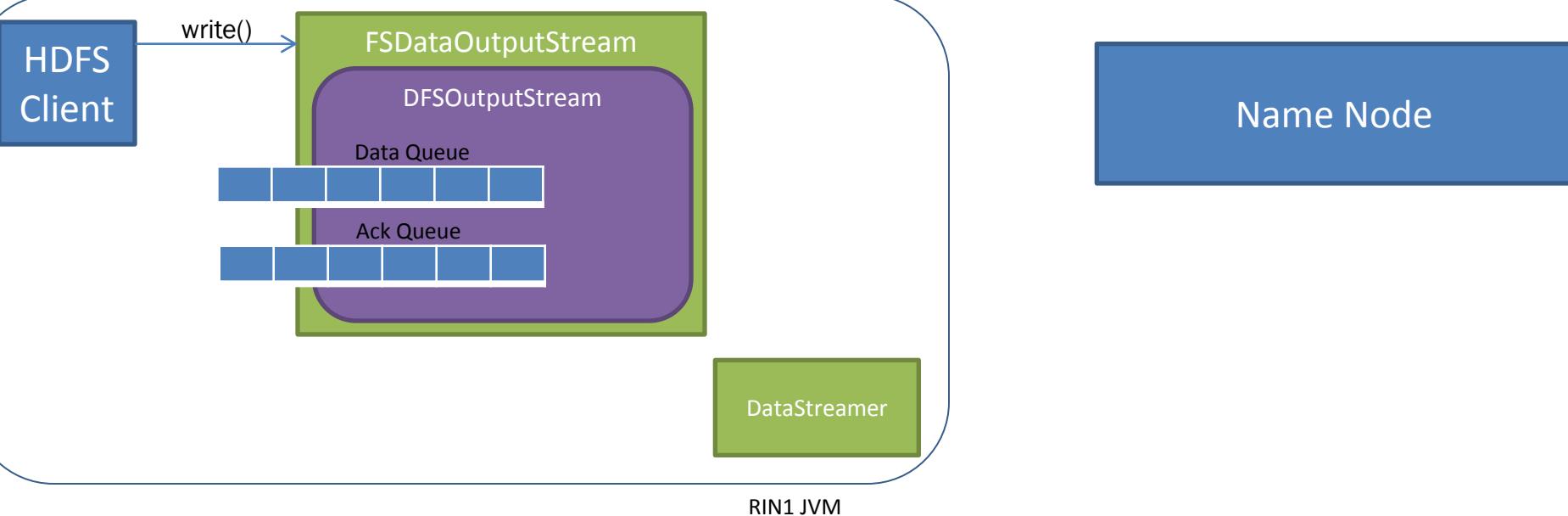


- Let's say there is a checksum error. This means the block is corrupt.
- Information about this corrupt block is sent to name node. Then DFSIS tries to connect to next closest node (R2N3).

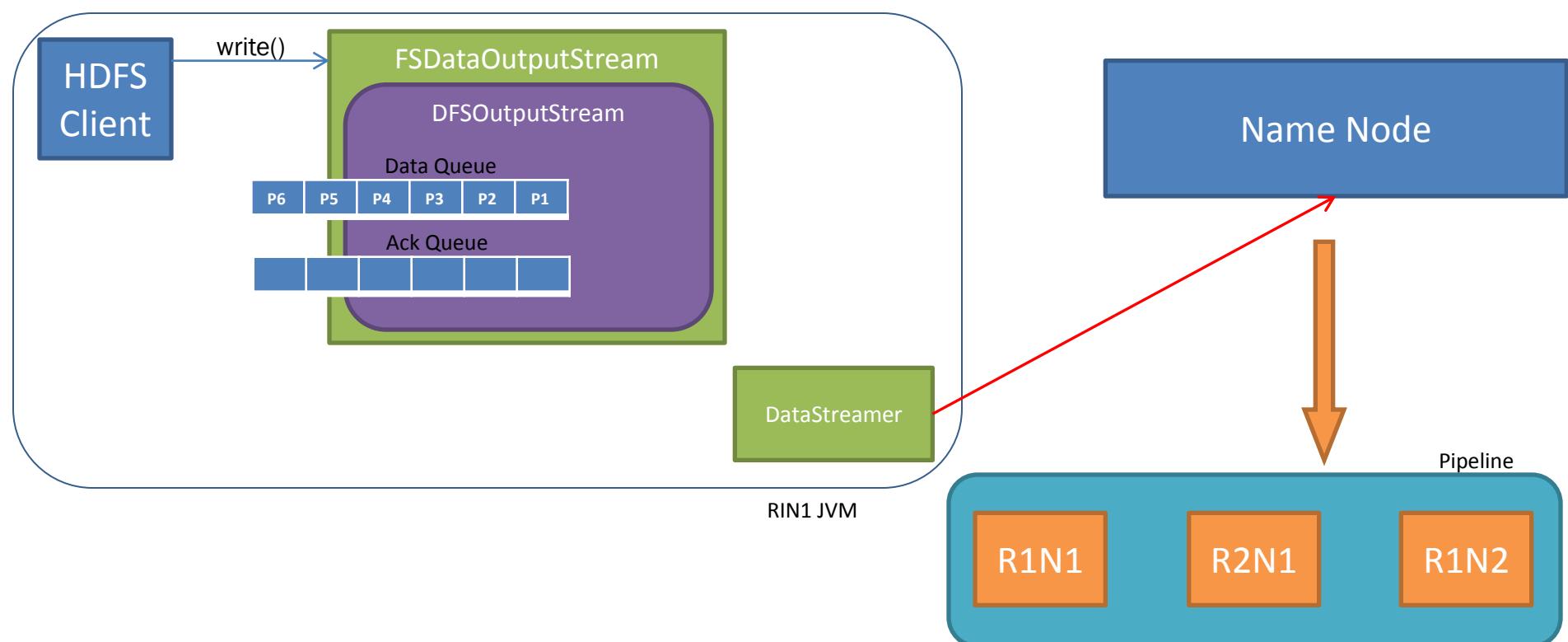
Anatomy of file write



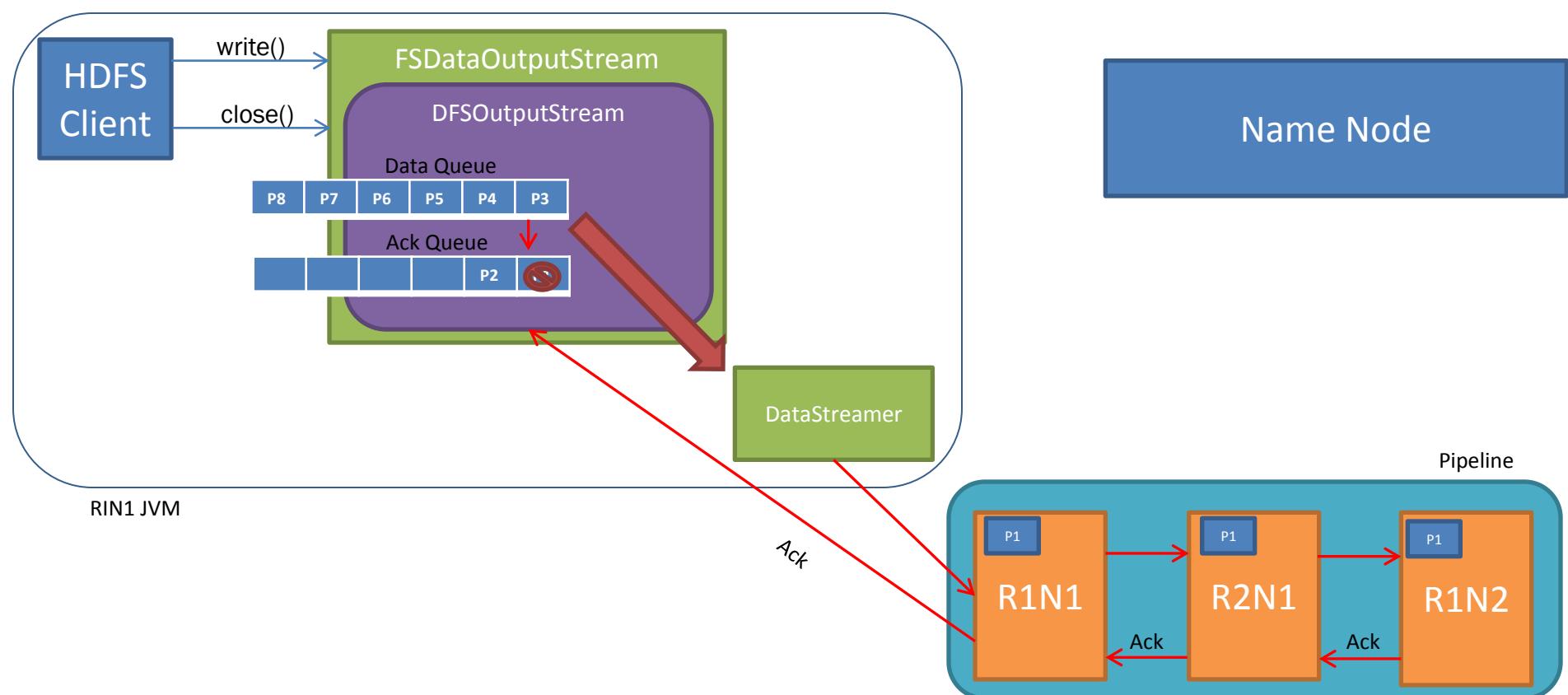
- Let's say we are trying to write the “sfo_crimes.csv” file from R1N1.
- So a HDFS Client program will run on R1N1’s JVM.
- First the HDFS client program calls the method `create()` on a Java class `DistributedFileSystem` (subclass of `FileSystem`).
- DFS makes a RPC call to name node to create a new file in the file system's namespace. No blocks are associated to the file at this stage.
- Name node performs various checks; ensures the file doesn't exists, the user has the right permissions to create the file. Then name node creates a record for the new file.
- Then DFS creates a `FSDataOutputStream` for the client to write data to. `FSDOS` wraps a `DFSOutputStream`, which handles communication with DN and NN.
- In response to ‘`FileSystem.create()`’, HDFS Client receives this `FSDataOutputStream`.



- From now on HDFS Client deals with FSDataOutputStream.
- HDFS Client invokes write() on the stream.
- Following are the important components involved in a file write;
 - **Data Queue**: When client writes data, DFDOS splits into packets and writes into this internal queue.
 - **DataStreamer**: The data queue is consumed by this component, which also communicates with name node for block allocation.
 - **Ack Queue**: Packets consumed by DataStreamer are temporarily saved in this internal queue.

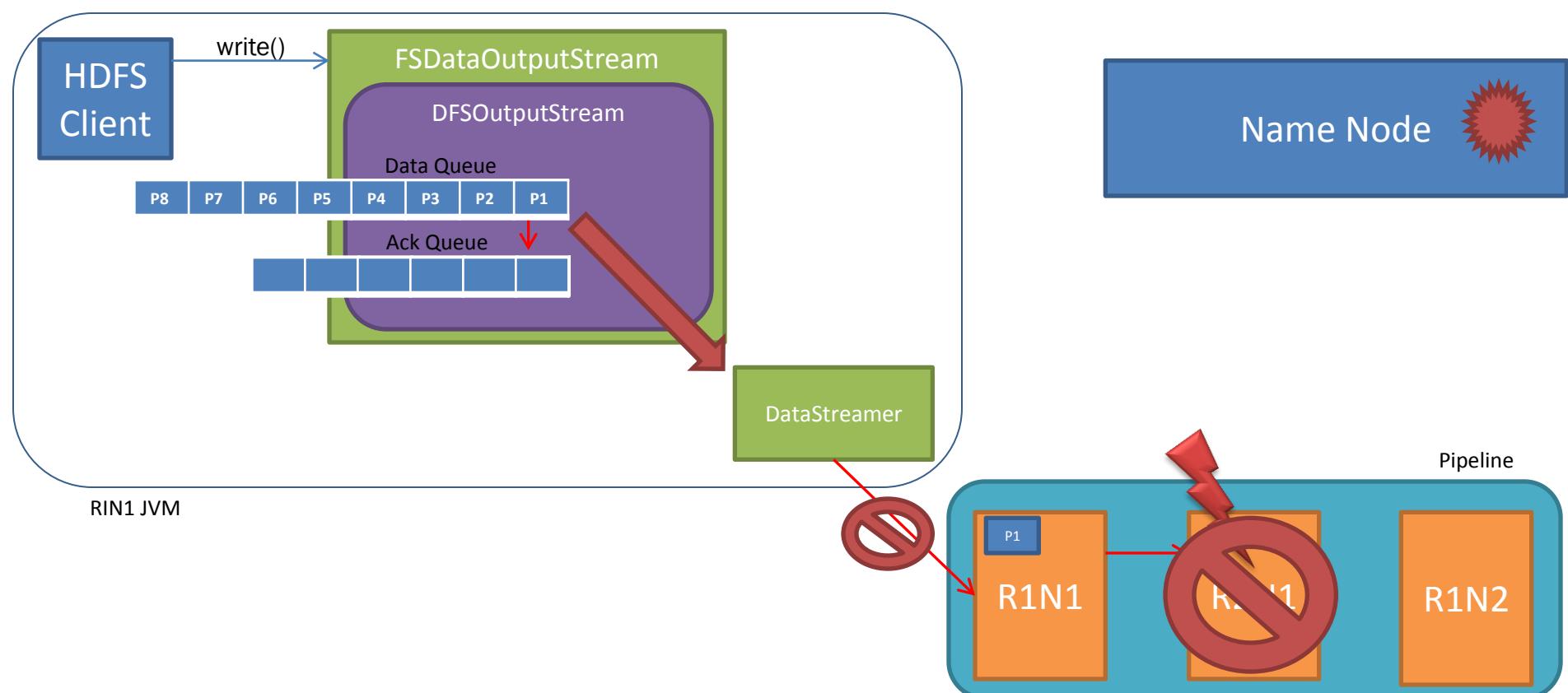


- As said, data written by client will be converted into packets and stored in data queue.
- DataStreamer communicates with NN to allocate new blocks by picking a list of suitable DNs to store the replicas. NN uses ‘Replica Placement’ as a strategy to pick DNs for a block.
- The list of DNs form a pipeline. Since the replication factor is assumed as 3, there are 3 nodes picked by NN.



- DataStreamer consumes few packets from data queue. A copy of the consumed data is stored in ‘ack queue’.
- DataStreamer streams the packet to first node in pipeline. Once the data is written in DN1, the data is forwarded to next DN. This repeats till last DN.
- Once the packet is written to the last DN, an acknowledgement is sent from each DN to DFSOS. The packet P1 is removed from Ack Queue.
- The whole process continues till a block is filled. After that, the pipeline is closed and DataStreamer asks NN for fresh set of DNs for next block. And the cycle repeats.
- HDFS Client calls the close() method once the write is finished. This would flush all the remaining packets to the pipeline & waits for ack before informing the NN that the write is complete.

Anatomy of file WRITE – Data Node WRITE Error



- A normal write begins with a `write()` method call from HDFS client on the stream. And let's say an error occurred while writing to `R2N1`.
- The pipeline will be closed.
- Packets in ack queue are moved to front data queue.
- The current block on good DNs are given a new identity and its communicated to NN, so the partial block on the failed DN will be deleted if the failed DN recovers later.
- The failed data node is removed from pipeline and the remaining data is written to the remaining two DNs.
- NN notices that the block is under-replicated, and it arranges for further replica to be created on another node.

Thank you



Hadoop MapReduce

- **What is MapReduce?**
- MapReduce is the processing layer of Hadoop. MapReduce programming model is designed for **processing large volumes of data in parallel** by dividing the work into a set of independent tasks. You need to put **business logic** in the way MapReduce works and rest things will be taken care by the framework. Work (complete job) which is submitted by the user to master is divided into small works (tasks) and assigned to slaves.
- MapReduce programs are written in a particular style influenced by functional programming constructs, specifical idioms for processing lists of data. In MapReduce, we get inputs from a list and it converts it into output which is again a list. It is the heart of Hadoop.

Hadoop MapReduce

- Hadoop is so much powerful and efficient due to MapReduce as here **parallel processing** is done.
- Map-Reduce divides the work into small parts, each of which can be done in parallel on the cluster of servers. A problem is divided into a large number of smaller problems each of which is processed to give individual outputs. These individual outputs are further processed to give final output.
- Hadoop Map-Reduce is scalable and can also be used across many computers. Many small machines can be used to process jobs that could not be processed by a large machine.

Hadoop MapReduce

- **Apache MapReduce Terminologies**
 - Map-Reduce is the data processing component of Hadoop. Map-Reduce programs transform **lists of input data elements** into **lists of output data elements**. A Map-Reduce program will do this twice, using two different list processing idioms-
 - Map
 - Reduce
 - In between Map and Reduce, there is small phase called **Shuffle and Sort** in MapReduce.

Hadoop MapReduce

- **What is a MapReduce Job?**

- MapReduce Job or a “full program” is an **execution of a Mapper and Reducer** across a data set. It is an execution of 2 processing layers i.e mapper and reducer. A MapReduce job consists of the **input data, the MapReduce Program, and configuration info.**
- So client needs to submit input data, he needs to write Map Reduce program and set the configuration info (These were provided during Hadoop setup in the configuration file and also we specify some configurations in our program itself which will be specific to our map reduce job).

Hadoop MapReduce

- **What is Task in Map Reduce?**

➤ A task in MapReduce is an **execution of a Mapper or a Reducer** on a slice of data. It is also called Task-In-Progress (TIP). It means processing of data is in progress either on mapper or reducer.

- **What is Task Attempt?**

➤ Task Attempt is a particular instance of an attempt to execute a task on a node. There is a possibility that anytime any machine can go down. For example, while processing data if any node goes down, framework reschedules the task to some other node. This **rescheduling of the task cannot be infinite**. There is an **upper limit** for that as well. **The default value of task attempt is 4.** If a task (Mapper or reducer) fails 4 times, then the **job is considered as a failed job**. For high priority job or huge job, the value of this task attempt can also be increased.

Hadoop MapReduce

- **Map Abstraction**

- The first phase of MapReduce paradigm, what is a map/mapper, what is the input to the mapper, how it processes the data, what is output from the mapper?
- The map takes key/value pair as input. Whether data is in structured or unstructured format, framework converts the incoming data into key and value.
 - Key is a reference to the input value.
 - Value is the data set on which to operate.

- **Map Processing:**

- A function defined by user – user can write custom business logic according to his need to process the data.
- Applies to every value in value input.

Hadoop MapReduce

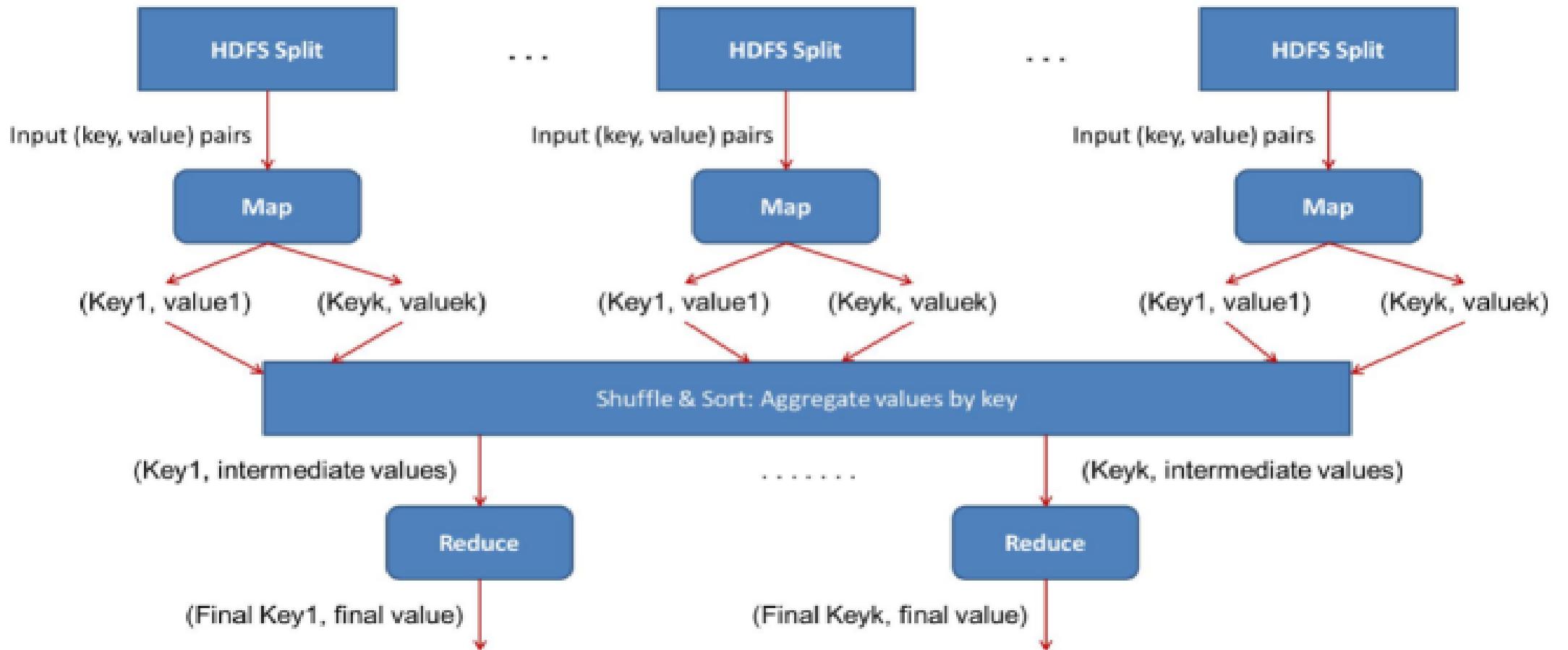
- Map produces a new list of key/value pairs:
 - An output of Map is called intermediate output.
 - Can be the different type from input pair.
 - An output of map is stored on the **local disk** from where it is shuffled to reduce nodes.
- **Reduce Abstraction**
 - The second phase of MapReduce – **Reducer** , what is the input to the reducer, what work reducer does, where reducer writes output?
 - **Reduce** takes intermediate Key / Value pairs as input and processes the output of the mapper. Usually, in the reducer, we do aggregation or summation sort of computation.
 - Input given to reducer is generated by Map (intermediate output)
 - Key / Value pairs provided to reduce are sorted by key

Hadoop MapReduce

- **Reduce processing:**
 - A function defined by user – Here also user can write custom business logic and get the final output.
 - Iterator supplies the values for a given key to the Reduce function.
 - Reduce produces a final list of key/value pairs:
 - An output of Reduce is called Final output.
 - It can be a different type from input pair.
 - An output of Reduce is stored in HDFS.

Hadoop MapReduce

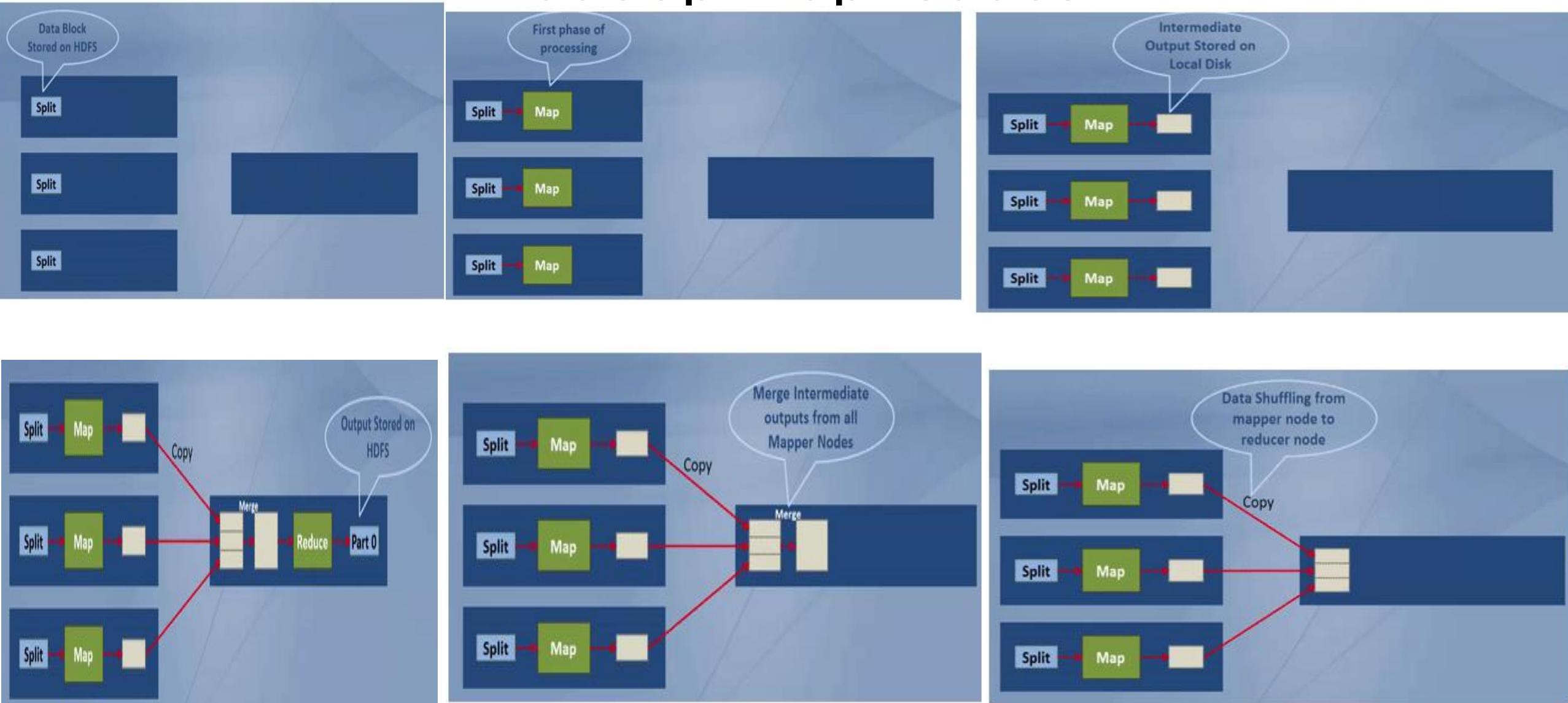
- How Map and Reduce work Together?



Hadoop MapReduce

- How Map and Reduce work Together?
 - **Input data** given to mapper is processed through **user defined function** written at **mapper**. All the required complex **business logic** is implemented at the mapper level so that heavy processing is done by the mapper in parallel as the number of mappers is much more than the number of reducers. Mapper generates an output which is intermediate data and this output goes as input to reducer.
 - This **intermediate result** is then processed by user defined function written at **reducer** and final output is generated. Usually, in reducer very light processing is done. This final output is stored in HDFS and replication is done as usual.

Hadoop MapReduce



Hadoop MapReduce

- As seen from the diagram of mapreduce workflow in Hadoop, the **square block** is a **slave**. There are 3 slaves in the figure. On all 3 slaves mappers will run, and then a reducer will run on any 1 of the slave. For simplicity of the figure, the reducer is shown on a different machine but it will run on mapper node only.
- Let us now discuss the map phase: An input to a mapper is 1 block at a time. (Split = block by default)
- An **output of mapper** is written to a **local disk** of the machine on which **mapper** is running. Once the map finishes, this intermediate output travels to reducer nodes (node where reducer will run).
- **Reducer** is the second phase of processing where the user can again write his custom business logic. Hence, an output of reducer is the final **output written to HDFS**.

Hadoop MapReduce

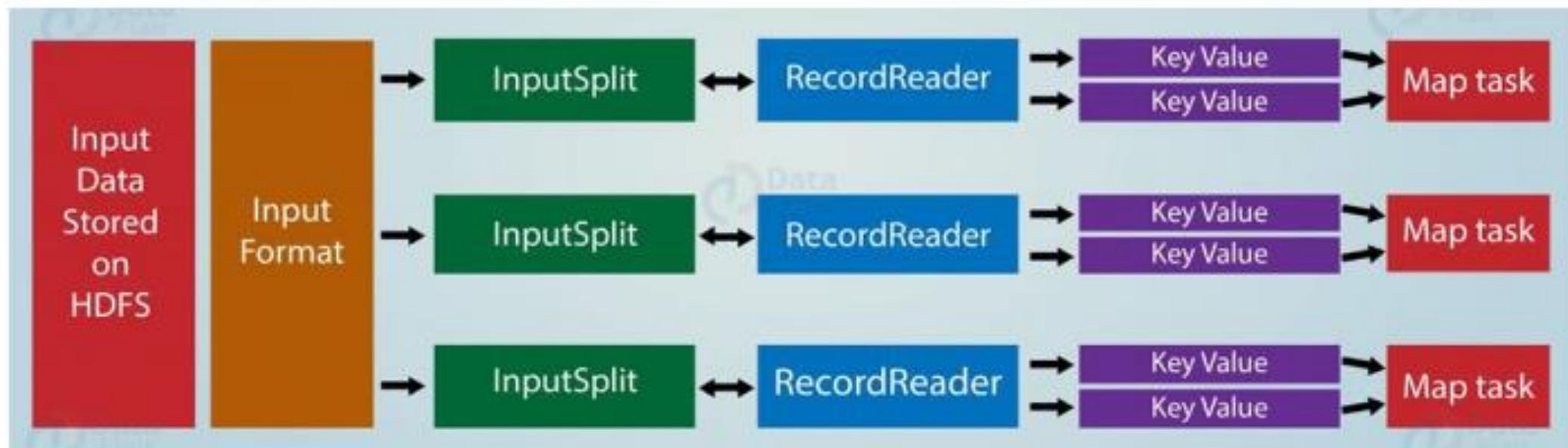
- By default on a slave, 2 mappers run at a time which can also be increased as per the requirements. It depends again on factors like data node hardware, block size, machine configuration etc. We should not increase the number of mappers beyond the certain limit because it will decrease the performance.
- **Mapper** in Hadoop Mapreduce writes the output to the local disk of the machine it is working. This is the temporary data. An output of mapper is also called intermediate output. All mappers are writing the output to the local disk. As First mapper finishes, data (output of the mapper) is traveling from mapper node to reducer node. Hence, this **movement of output from mapper node to reducer node** is called **shuffle**.

Hadoop MapReduce

- **Reducer** is also deployed on any one of the datanode only. An output from all the mappers goes to the reducer. All these **outputs** from different **mappers** are **merged** to form input for the reducer. This input is also on local disk. Reducer is another processor where you can write custom business logic. It is the second stage of the processing. Usually to reducer we write **aggregation**, **summation** etc. type of functionalities. Hence, Reducer gives the final output which it writes on HDFS.
- Map and reduce are the stages of processing. They run one after other. After all, mappers complete the processing, then only reducer starts processing.

Hadoop MapReduce

- Concept of Key-Value Pair in Hadoop MapReduce



- In MapReduce process, before passing the data to the mapper, data should be first converted into key-value pairs as mapper only understands key-value pairs of data.

Hadoop MapReduce

- key-value pairs in Hadoop MapReduce is generated as follows:
 - **InputSplit** – It is the logical representation of data. The data to be processed by an individual Mapper is presented by the InputSplit.
 - **RecordReader** – It communicates with the InputSplit and it converts the Split into records which are in form of key-value pairs that are suitable for reading by the mapper. By default, RecordReader uses TextInputFormat for converting data into a key-value pair. RecordReader communicates with the InputSplit until the file reading is not completed.

Hadoop MapReduce

- In MapReduce, map function processes a certain key-value pair and emits a certain number of key-value pairs and the Reduce function processes values grouped by the same key and emits another set of key-value pairs as output. The output types of the Map should match the input types of the Reduce as shown below:
- Map: $(K_1, V_1) \rightarrow \text{list } (K_2, V_2)$
- Reduce: $\{(K_2, \text{list } (V_2))\} \rightarrow \text{list } (K_3, V_3)$

Hadoop MapReduce

- **On what basis is a key-value pair generated in Hadoop?**

➤ Generation of a key-value pair in Hadoop depends on the data set and the required output. In general, the key-value pair is specified in 4 places: Map input, Map output, Reduce input and Reduce output.

1. Map Input

➤ Map-input by default will take the line offset as the key and the content of the line will be the value as Text. By using custom InputFormat we can modify them.

2. Map Output

➤ Map basic responsibility is to filter the data and provide the environment for grouping of data based on the key.

➤ Key – It will be the field/ text/ object on which the data has to be grouped and aggregated on the reducer side.

➤ Value – It will be the field/ text/ object which is to be handled by each individual reduce method.

Hadoop MapReduce

3. Reduce Input

- The output of Map is the input for reduce, so it is same as Map-Output.

4. Reduce Output

- It depends on the required output.

- **MapReduce key-value pair Example**

- Suppose, the content of the file which is stored in HDFS is **John is Mark Joey is John**. Using InputFormat, we will define how this file will split and read. By default, RecordReader uses TextInputFormat to convert this file into a key-value pair.

- **Key** – It is offset of the beginning of the line within the file.

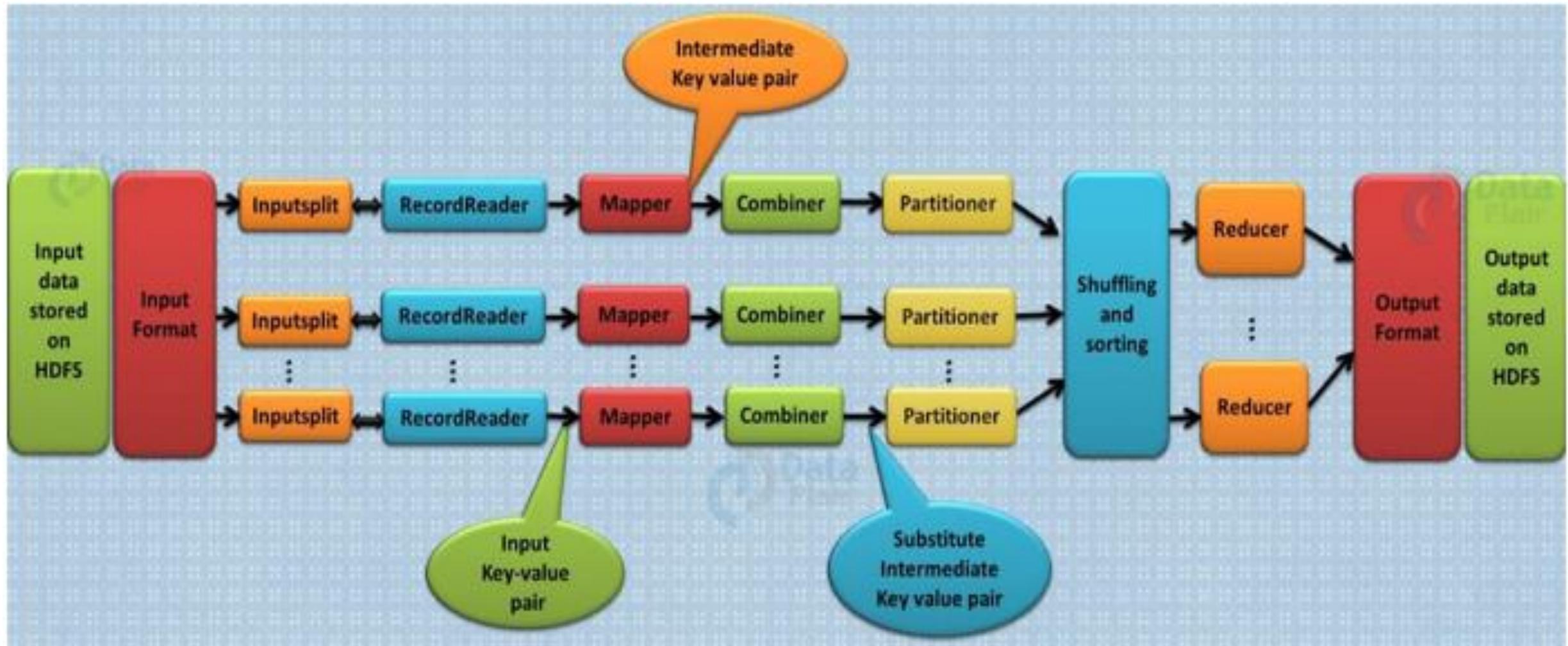
- **Value** – It is the content of the line, excluding line terminators.

- From the above content of the file-

- **Key** is 0

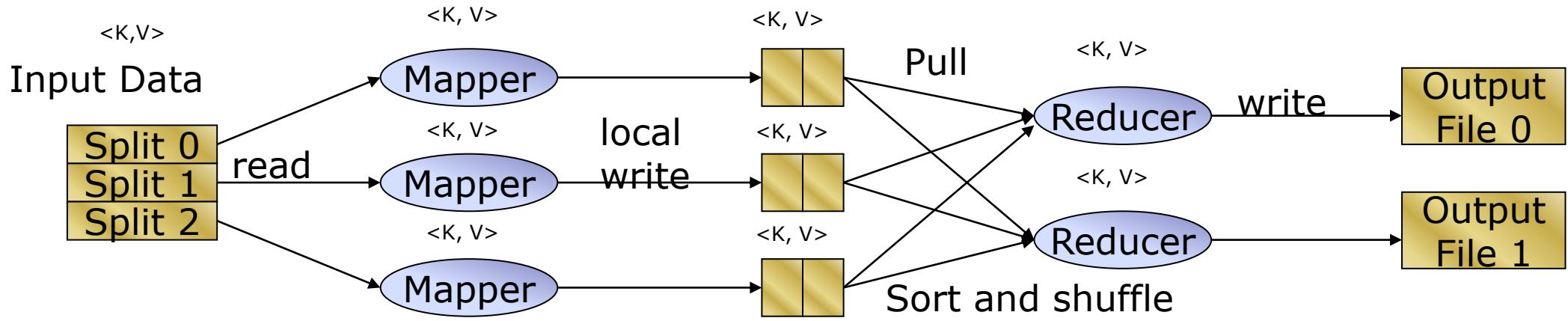
- **Value** is John is Mark Joey is John.

Hadoop MapReduce

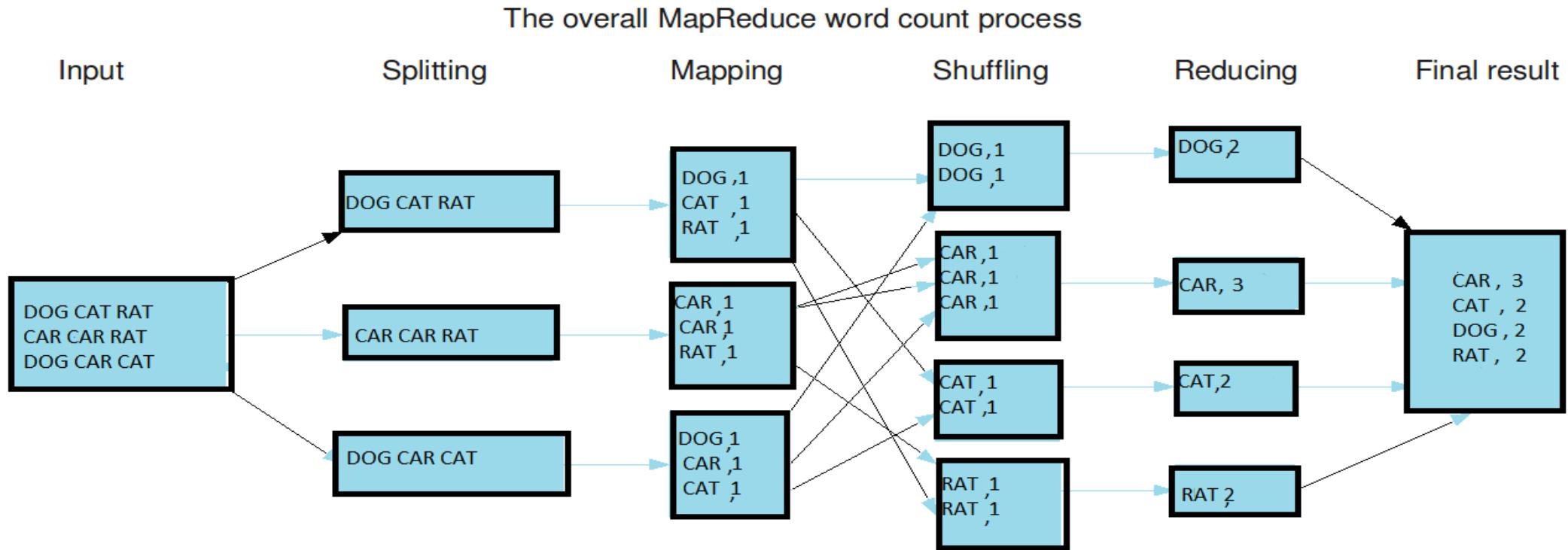


MapReduce

- MapReduce

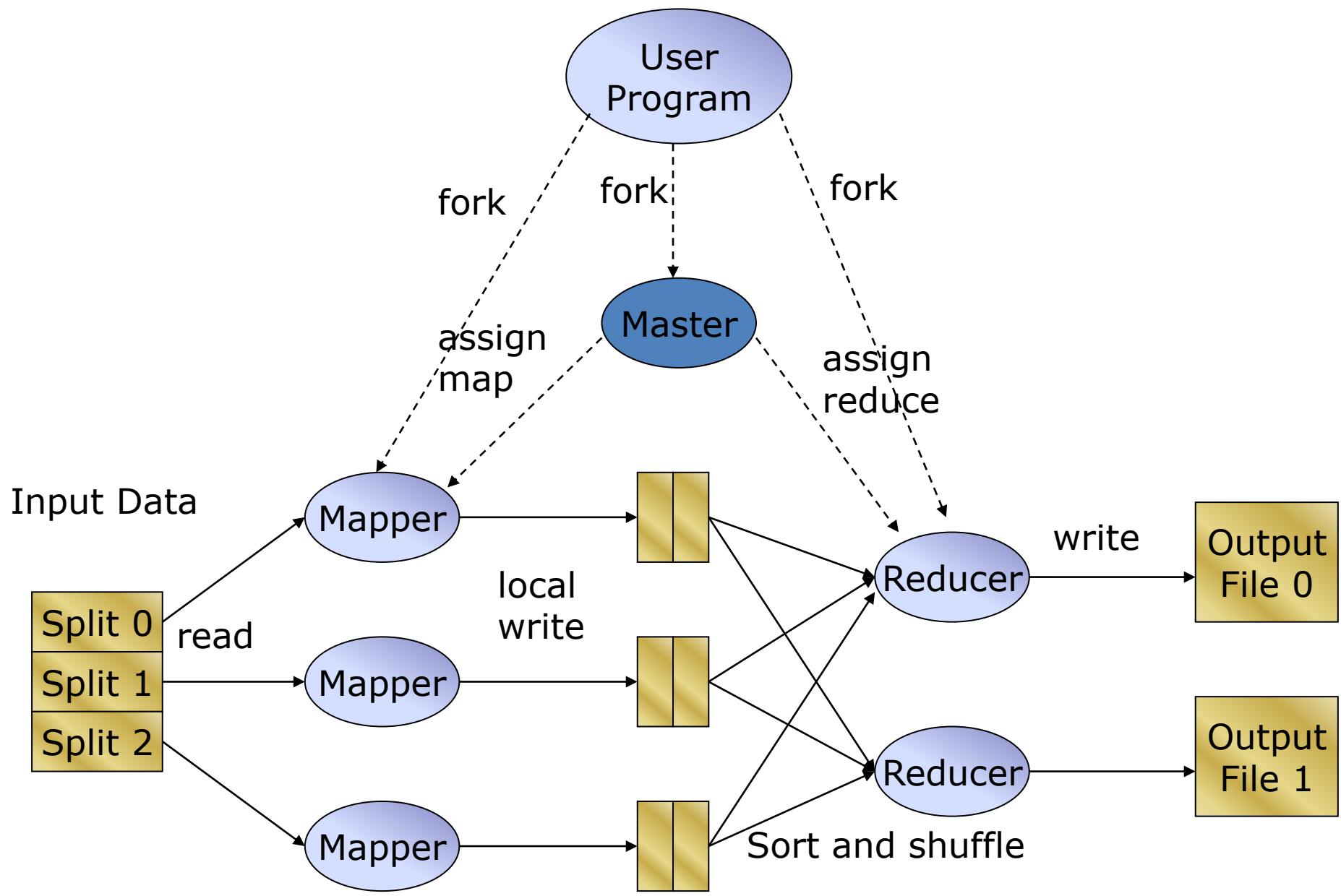


WordCount



Thank you.

MapReduce



Text Data



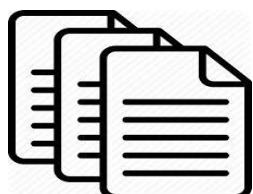
256 MBs



64 MBs



64 MBs



64 MBs



64 MBs

Block



**Computing
Node**



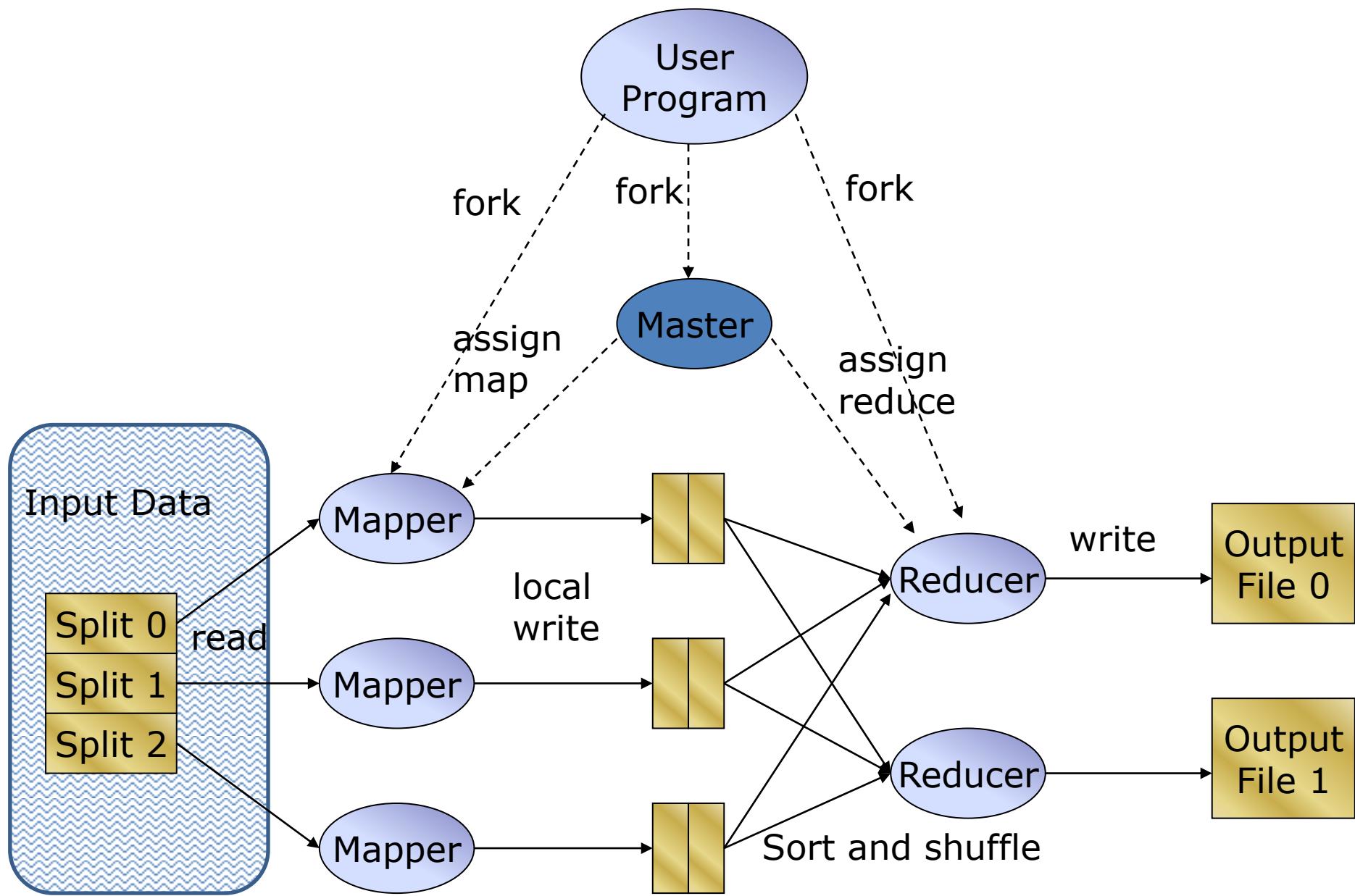
**Computing
Node**



**Computing
Node**



**Computing
Node**



Mapper

Computing Node

hello wordcount
MapReduce
Hadoop program \n
.....

Computing Node

this is my first
MapReduce
program\n
.....

Computing Node

I am working on
MapReduce and
Spark \n
.....

Computing Node

MapReduce is
efficient
framework\n
.....

Computing Node Mapper → I/P

<Key, Value>

<1, “hello
wordcount
MapReduce
Hadoop program”>
<98, Next Line>
....

Computing Node Mapper → I/P

<Key, Value>

<1, “hello this is my
first MapReduce
program”>
<106, Next Line>
....

Computing Node Mapper → I/P

<Key, Value>

<1, “I am working
on MapReduce and
Spark ”>
<59, Next Line>
....

Computing Node Mapper → I/P

<Key, Value>

<1, “MapReduce is
efficient
framework”>
<33, Next Line>
....

Tokenization

Tokenization

Tokenization

Tokenization

Computing Node Mapper → O/P

<Key, Value>
<hello, 1>
<wordcount, 1>
<MapReduce, 1>
<Hadoop, 1>
<program, 1>
....

Computing Node Mapper → O/P

<Key, Value>
<hello, 1>
<this, 1>
<is, 1>
<my, 1>
<first, 1>
<MapReduce, 1>
<program, 1>
....

Computing Node Mapper → O/P

<Key, Value>
<i, 1>
<am, 1>
<working, 1>
<on, 1>
<MapReduce, 1>
<and, 1>
<Spark, 1>....

Computing Node Mapper → O/P

<Key, Value>
<MapReduce, 1>
<is, 1>
<efficient, 1>
<framework, 1>
....

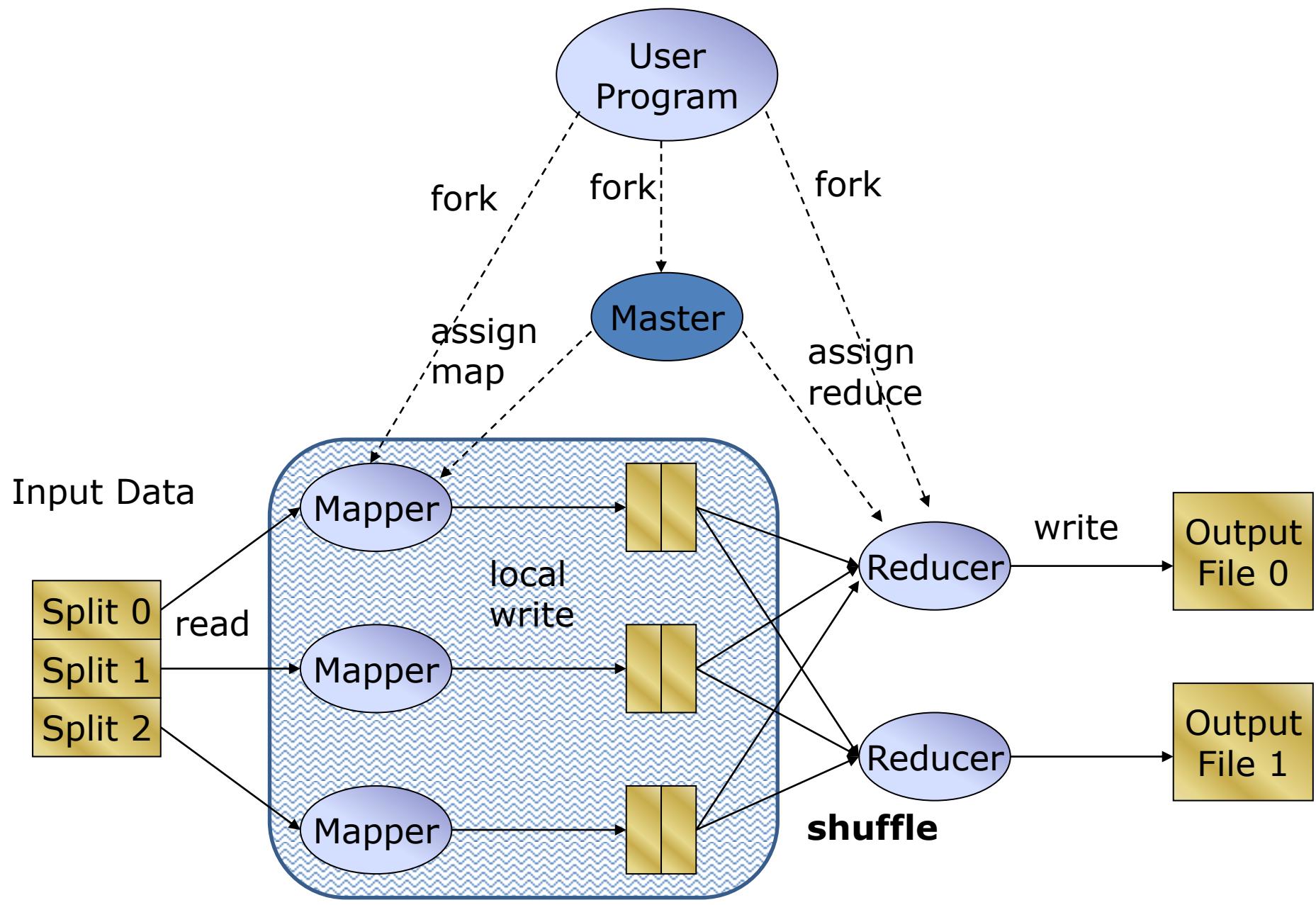
Spilling Sorting

<Key, Value>
<Hadoop, 1>
<MapReduce, 1>
<hello, 1>
<program, 1>
<wordcount, 1>
....

<Key, Value>
<MapReduce, 1>
<first, 1>
<hello, 1>
<is, 1>
<my, 1>
<program, 1>
<this, 1>
....

<Key, Value>
<MapReduce, 1>
<Spark, 1>
<and, 1>
<am, 1>
<i, 1>
<on, 1>
<working, 1>
....

<Key, Value>
<MapReduce, 1>
<efficient, 1>
<framework, 1>
<is, 1>
....



Computing Node Mapper → O/P

<Key, Value>
<Hadoop, 1>
<MapReduce, 1>
<hello, 1>
<program, 1>
<wordcount, 1>
....

Computing Node Mapper → O/P

<Key, Value>
<MapReduce, 1>
<first, 1>
<hello, 1>
<is, 1>
<my, 1>
<program, 1>
<this, 1>
....

Computing Node Mapper → O/P

<Key, Value>
<MapReduce, 1>
<Spark, 1>
<and, 1>
<am, 1>
<i, 1>
<on, 1>
<working, 1>
....

Computing Node Mapper → O/P

<Key, Value>
<MapReduce, 1>
<efficient, 1>
<framework, 1>
<is, 1>
....

Computing Node Reducer → O/P

<Key, Value>
<MapReduce, 1>
<MapReduce, 1>
<MapReduce, 1>
<MapReduce, 1>
<Hadoop, 1>
<Spark, 1>
<and, 1>
<am, 1>
<efficient, 1>
<framework, 1>
....

Computing Node Reducer → O/P

<Key, Value>
<hello, 1>
<hello, 1>
<i, 1>
<is, 1>
<is, 1>
<my, 1>
<on, 1>
<program, 1>
<program, 1>
<wordcount, 1>
<working, 1>
....

Shuffling

Computing Node Reducer → O/P

<Key, Value>
<MapReduce, 1>
<MapReduce, 1>
<MapReduce, 1>
<MapReduce, 1>
<Hadoop, 1>
<Spark, 1>
<and, 1>
<am, 1>
<efficient, 1>
<framework, 1>

....

Computing Node Reducer → O/P

<Key, Value>
<hello, 1>
<hello, 1>
<i, 1>
<is, 1>
<is, 1>
<my, 1>
<on, 1>
<program, 1>
<program, 1>
<wordcount, 1>
<working, 1>

....

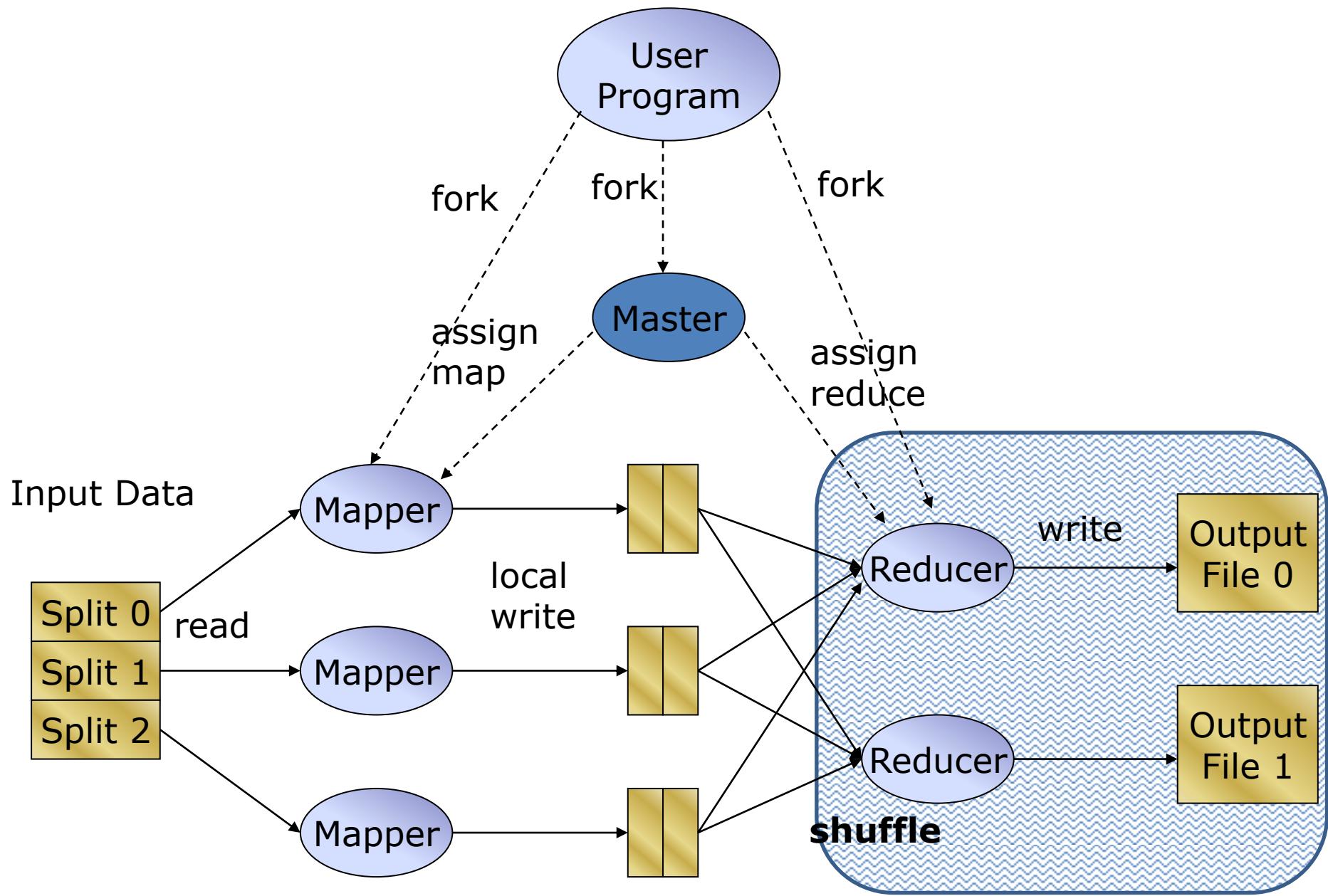
<Key, Value>
<MapReduce, 4>
<Hadoop, 1>
<Spark, 1>
<and, 1>
<am, 1>
<efficient, 1>
<framework, 1>

....

<Key, Value>
<hello, 2>
<i, 1>
<is, 2>
<my, 1>
<on, 1>
<program, 2>
<wordcount, 1>
<working, 1>

....

HDFS



- No of Mapper and Reducer
- Key Parameter in MapReduce Design
 - Key-Value pair
 - Load Balance b/t Mapper and Reducer



Query large datasets in near real time with
Hive and Pig

Hive

- **Challenges at Facebook: Exponential Growth of Data**

➤ Before 2008, all the data processing infrastructure in Facebook was built around a data warehouse based on commercial RDBMS. These infrastructures were capable enough to suffice the needs of Facebook at that time. But, as the data started growing very fast, it became a huge challenge to manage and process this huge dataset. According to a Facebook article, the data scaled from a 15 TB data set in 2007 to a 2 PB data in 2009. Also, many Facebook products involve analysis of the data like Audience Insights, Facebook Lexicon, Facebook Ads, etc. So, they needed a scalable and economical solution to cope up with this very problem and, therefore started using the Hadoop framework.

Hive

- **Birth of Hive**
 - Facebook played an active role in the birth of Hive as Facebook uses Hadoop to handle Big Data. Hadoop uses MapReduce to process data. Previously, users needed to write lengthy, complex codes to process and analyze data. Not everyone was well-versed in **Java and other complex programming languages**. Also, for performing simple analysis one has to write a hundred lines of MapReduce code. On the other hand, many individuals were **comfortable with** writing queries in **SQL**. For this reason, there was a need to develop a language similar to SQL, which was well-known to all users. This is how the Hive Query Language, also known as HiveQL, came to be.

Hive

- **What is Hive in Hadoop?**
 - Hive is a data warehouse system used to **query and analyze large datasets stored in HDFS**. Hive uses a query language called HiveQL, which is similar to SQL.



- The image above demonstrates a user writing queries in the HiveQL language, which is then converted into MapReduce tasks. Next, the data is processed and analyzed. **HiveQL works on structured and semi-structured data**, such as numbers, addresses, dates, names, and so on. HiveQL allows **multiple users to query data simultaneously**.

Hive

- **What is Hive in Hadoop?**

- Hive is query engine because hive does not has it's own storage to store the data.
- Require knowledge of simple SQL query (Create, update, delete, insert, select, sub queries and join). No need of complex query like stored procedure, trigger these kind of sequel knowledge is not required.
- Hive act as a vehicle and runs on a engine map-reduce. Therefore we say hive is abstraction of map-reduce.
- Hive internally used map reduce engine to process the query. So instead of Java; communicate with sequels. (Bypassing this with sequel via hive)
- Hive is just replacing Java part; Hive is not replacing map reduce part.

Hive

- **What is Hive in Hadoop?**
- Hive **reads** data from **HDFS** and then **process** it in **hive** with the help of **map reduce** and then the **output** again stored back to **HDFS**.
- When someone say we are not using map reduce then what I understood is that You are using some other engine to run hive.
- Similar to map reduce We have one more engine which is **Spark**. Spark is replacement of map reduce. But it's not replacement of whole hadoop framework.
- Hive can also run on Spark engine instead of map reduce. So hive is a vehicle and map reduce/spark engine which operates vehicle.
- Question: We can do everything with sequel itself; no need to write code for every situation?

Hive

- No, We can't do everything with sequel itself. There will be some complex transformation (like require extra optimization, extra performance, extra complex logic etc.) then we have to go for spark or map reduce and to write programs. But Hive is more matured in recent days and we can do so many things with hive.
- Amazon use Hive with S3 (Database). So Hive can also be use with S3 also instead of HDFS

Hive

- **Why Hive?**
- Apache Hive saves developers from **writing complex** Hadoop MapReduce jobs for ad-hoc requirements. Hence, hive provides summarization, analysis, and query of data.
- Hive is very **fast and scalable**. It is highly extensible. Since Apache Hive is similar to SQL, hence it becomes very easy for the SQL developers to learn and implement Hive Queries.
- Hive reduces the complexity of MapReduce by providing an interface where the user can submit SQL queries. So, now business analysts can play with **Big Data** using Apache Hive and generate insights.
- It also provides file access on various data stores like HDFS and HBase. The most important feature of Apache Hive is that to learn Hive we don't have to learn Java.

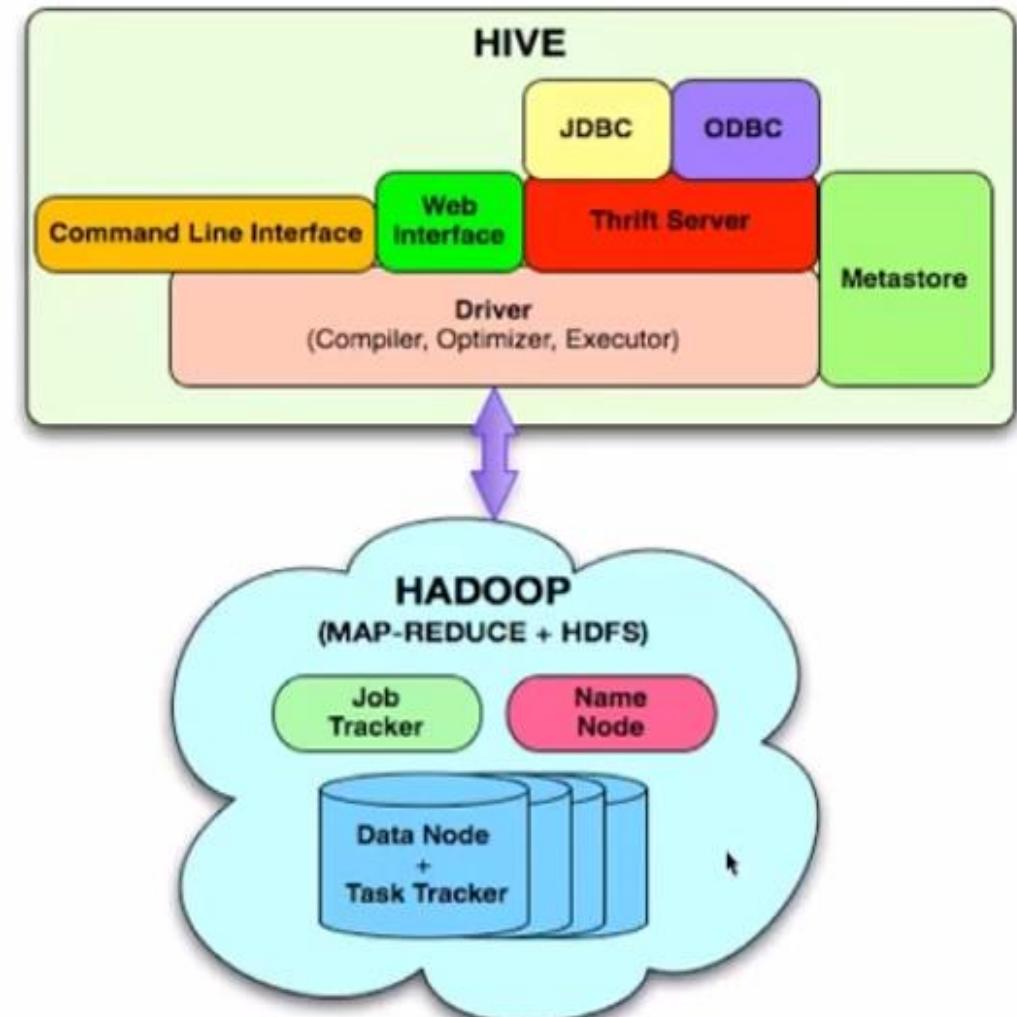
Architecture

External Interfaces- CLI, WebUI, JDBC, ODBC
programming interfaces

Thrift Server – Cross Language service framework .

Metastore - Meta data about the Hive tables, partitions

Driver - Brain of Hive! Compiler, Optimizer and Execution engine



Hive components

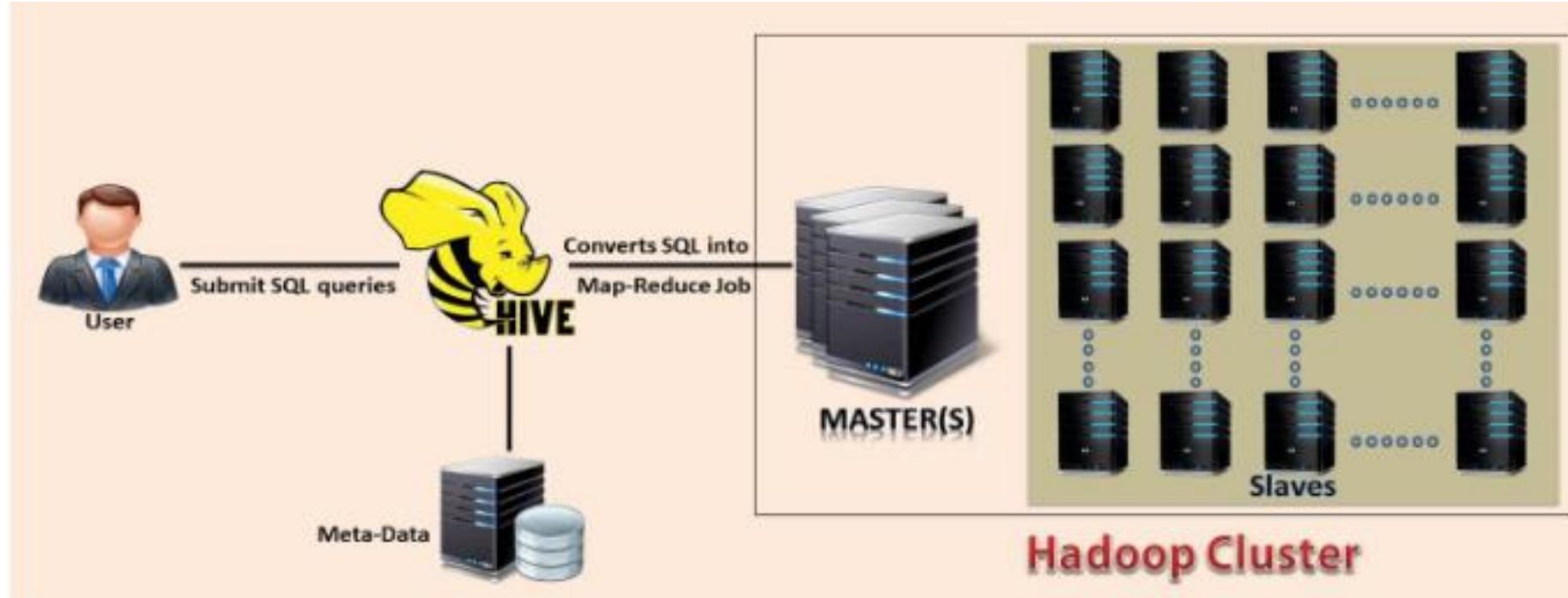
- **Metastore** – It stores metadata for each of the tables like their schema and location. Hive also includes the partition metadata. This helps the driver to track the progress of various data sets distributed over the cluster. It stores the data in a traditional RDBMS format. Hive **metadata helps the driver to keep a track of the data** and it is highly crucial. Backup server regularly replicates the data which it can retrieve in case of data loss.
- **Driver** – It acts like a controller which **receives the HiveQL statements**. The driver starts the execution of the statement by creating sessions. It monitors the life cycle and progress of the execution. Driver stores the necessary metadata generated during the execution of a HiveQL statement. It also acts as a collection point of data or query result obtained after the Reduce operation.

Hive components

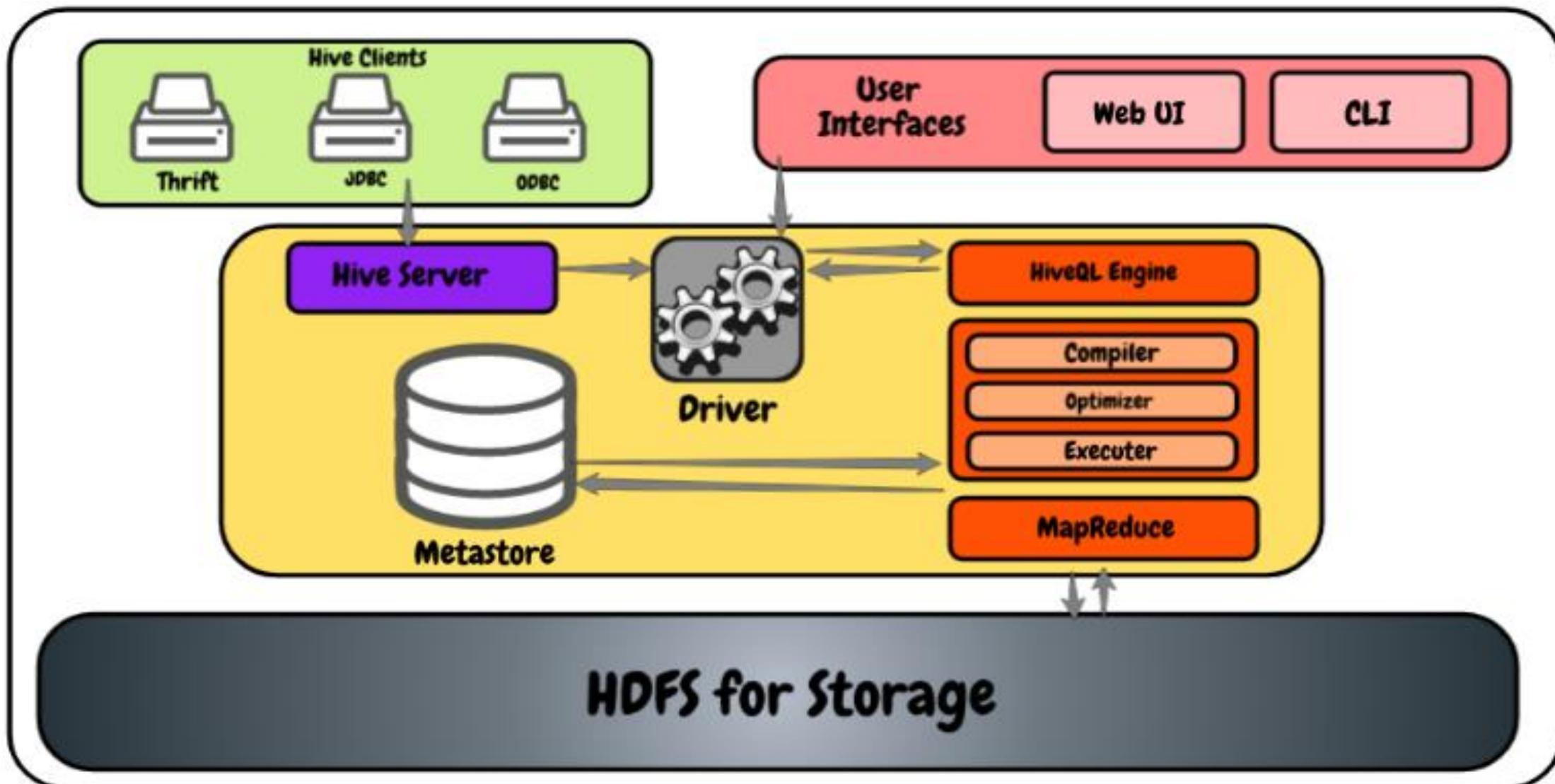
- **Compiler** – It performs the compilation of the HiveQL query. This converts the query to an execution plan. The plan contains the tasks. It also contains steps needed to be performed by the MapReduce to get the output as translated by the query. The compiler in Hive converts the query to an **Abstract Syntax Tree** (AST). First, check for compatibility and compile-time errors, then converts the AST to a **Directed Acyclic Graph (DAG)**.
- **Optimizer** – It performs various transformations on the execution plan to provide optimized DAG. It **aggregates** the transformations together, such as converting a pipeline of joins to a single join, for better performance. The optimizer can also split the tasks, such as applying a transformation on data before a reduce operation, to provide better performance.

Hive components

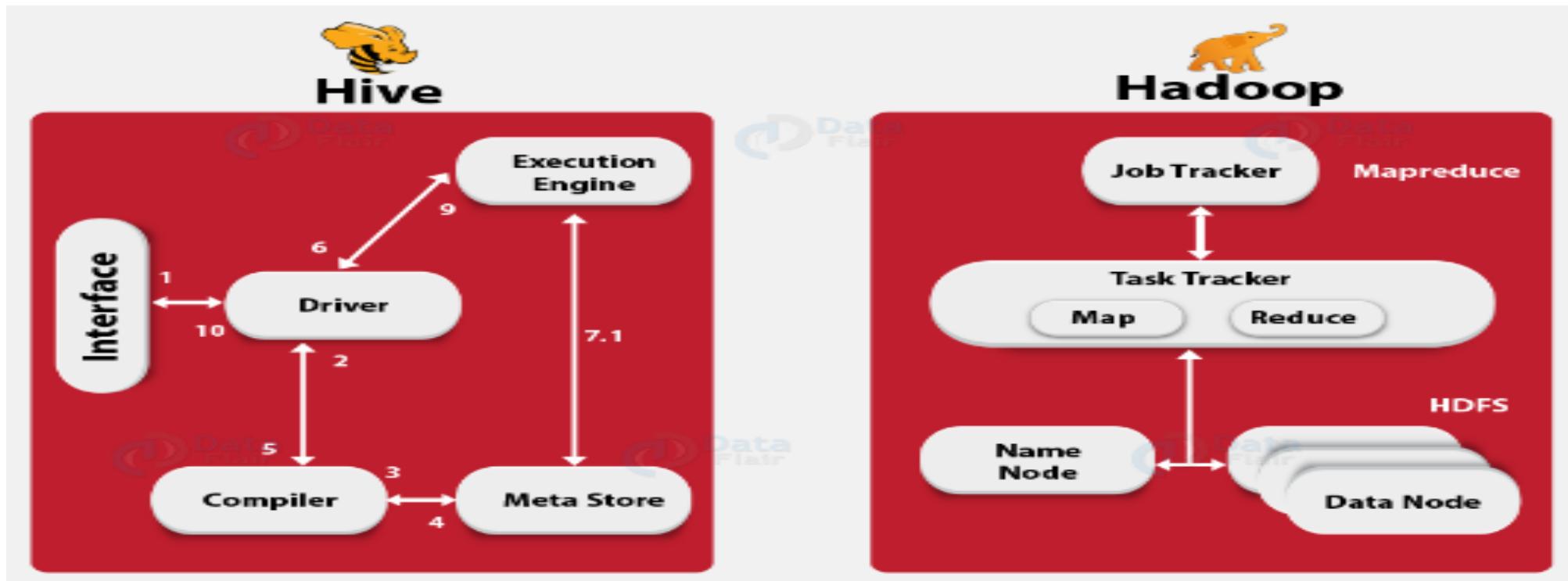
- **Executor** – Once compilation and optimization complete, the executor executes the tasks. Executor takes care of pipelining the tasks.
- **CLI, UI, and Thrift Server** – CLI (command-line interface) provides a user interface for an external user to interact with Hive. Thrift server in Hive allows external clients to interact with Hive over a network, similar to the JDBC or ODBC protocols.



Hive Architecture



How Does Hive Works?



- **Step-1 Execute Query:** In the first step, we write down the query using the web interface or the command-line interface of the hive. It sends it to the driver to execute the query.

How Does Hive Works?

- **Step-2 Get Plan:** In the next step, the driver sends the received query to the compiler where the compiler verifies the syntax.
- **Step-3 Get Metadata:** And once the syntax verification is done, it requests metadata from the meta store.
- **Step-4 Send Metadata:** Now, the metadata provides information like the database, tables, data types of the column in response to the query back to the compiler.
- **Step-5 Send Plan:** The compiler again checks all the requirements received from the meta store and sends the execution plan to the driver.
- **Step-6 Execute Plan:** Now, the driver sends the execution plan to the HiveQL process engine where the engine converts the query into the map-reduce job.

How Does Hive Works?

- **Step-7 Execute Job:** After the query is converted into the map-reduce job, it sends the task information to the Hadoop where the processing of the query begins and at the same time it updates the metadata about the map-reduce job in the meta store.
- **Step-8 Fetch Result:** Once the processing is done, the execution engine receives the results of the query.
- **Step-9 Send Results:** The execution engine transfers the results back to the driver and at last, the driver sends the results to Hive user interfaces from where we can see results.

Hive Shell

- The shell is the primary way with the help of which we interact with the Hive; we can issue our commands or queries in HiveQL inside the Hive shell. Hive Shell is almost similar to MySQL Shell.
- It is the **command line interface** for Hive. In Hive Shell users can run HQL queries. HiveQL is also **case-insensitive** (except for string comparisons) same as SQL.
- We can run the Hive Shell in two modes which are: **Non-Interactive mode and Interactive mode**
- **Hive in Non-Interactive mode** – Hive Shell can be run in the non-interactive mode, with **-f option** we can specify the location of a file which contains HQL queries. For example- `hive -f my-script.q`

Hive Shell

- **Hive in Interactive mode** – Hive Shell can also be run in the interactive mode. In this mode, we directly need to go to the hive shell and run the queries there. In hive shell, we can submit required queries manually and get the result. For example- \$bin/hive, go to hive shell.

HiveQL

DDL :

- CREATE DATABASE
- CREATE TABLE
- ALTER TABLE
- SHOW TABLE
- DESCRIBE

DML:

- LOAD TABLE
- INSERT

QUERY:

- SELECT
- GROUP BY
- JOIN
- MULTI TABLE INSERT

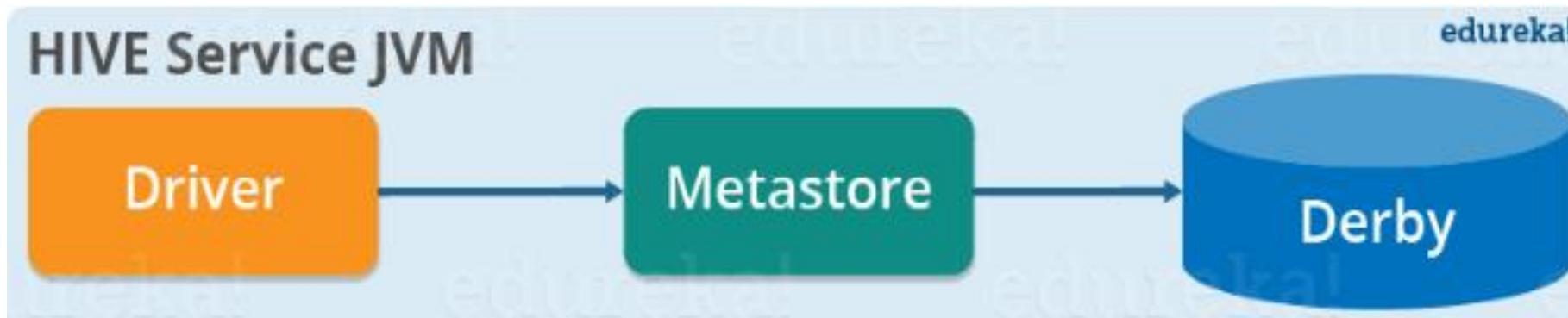
Metastore Configuration

- **Metastore Configuration**
- Metastore stores the meta data information using RDBMS and an open source ORM (Object Relational Model) layer called Data Nucleus which converts the object representation into relational schema and vice versa. The reason for choosing RDBMS instead of HDFS is to achieve low latency.
- We can implement metastore in following two configurations:
 - Embedded Metastore
 - Remote Metastore

Metastore Configuration

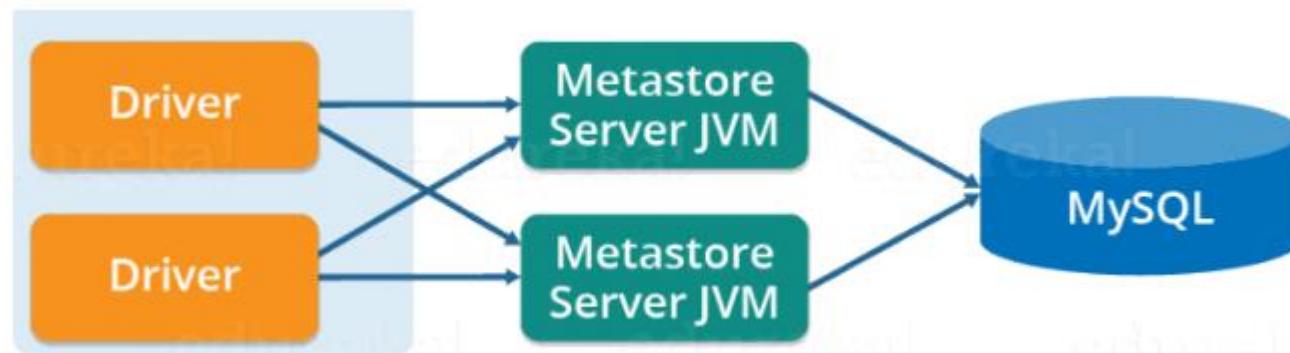
- **Embedded Metastore:**

- Both the metastore service and the Hive service runs in the same JVM by default using an embedded Derby Database instance where metadata is stored in the local disk. This is called embedded metastore configuration. In this case, only one user can connect to metastore database at a time. If you start a second instance of Hive driver, you will get an error. This is good for unit testing, but not for the practical solutions.



Metastore Configuration

- **Remote Metastore:**
- In the remote metastore configuration, the metastore service runs on its own separate JVM and not in the Hive service JVM. Other processes communicate with the metastore server using Thrift Network APIs. You can have one or more metastore servers in this case to provide more availability. The main advantage of using remote metastore is you do not need to share JDBC login credential with each Hive user to access the metastore database.



Hive Data Model

- Data in Hive can be categorized into three types on the granular level:
 - Table
 - Partition
 - Bucket
- **Tables:**
- Tables in Hive are the same as the tables present in a Relational Database. You can perform filter, project, join and union operations on them. There are two types of tables in Hive:

1. Managed Table:

- *Command:*
- CREATE TABLE <table_name> (column1 data_type, column2 data_type);
- LOAD DATA INPATH <HDFS_file_location> INTO table managed_table;

Hive Data Model

- As the name suggests (managed table), Hive is responsible for managing the data of a managed table. In other words, “Hive manages the data”, is that if you load the data from a file present in HDFS into a Hive Managed Table and issue a DROP command on it, the table along with its metadata will be deleted. So, the data belonging to the dropped managed_table no longer exist anywhere in HDFS and you can't retrieve it by any means. Basically, you are moving the data when you issue the LOAD command from the HDFS file location to the Hive warehouse directory.
- Note: The default path of the warehouse directory is set to /user/hive/warehouse. The data of a Hive table resides in warehouse_directory/table_name (HDFS). You can also specify the path of the warehouse directory in the hive.metastore.warehouse.dir configuration parameter present in the hive-site.xml.

Hive Data Model

2. External Table:

- CREATE EXTERNAL TABLE <table_name> (column1 data_type, column2 data_type) LOCATION '<table_hive_location>';
- LOAD DATA INPATH '<HDFS_file_location>' INTO TABLE <table_name>;
- For external table, Hive is not responsible for managing the data. In this case, when you issue the LOAD command, Hive moves the data into its warehouse directory. Then, Hive creates the metadata information for the external table. Now, if you issue a DROP command on the external table, only metadata information regarding the external table will be deleted. Therefore, you can still retrieve the data of that very external table from the warehouse directory using HDFS commands.

Hive Data Model

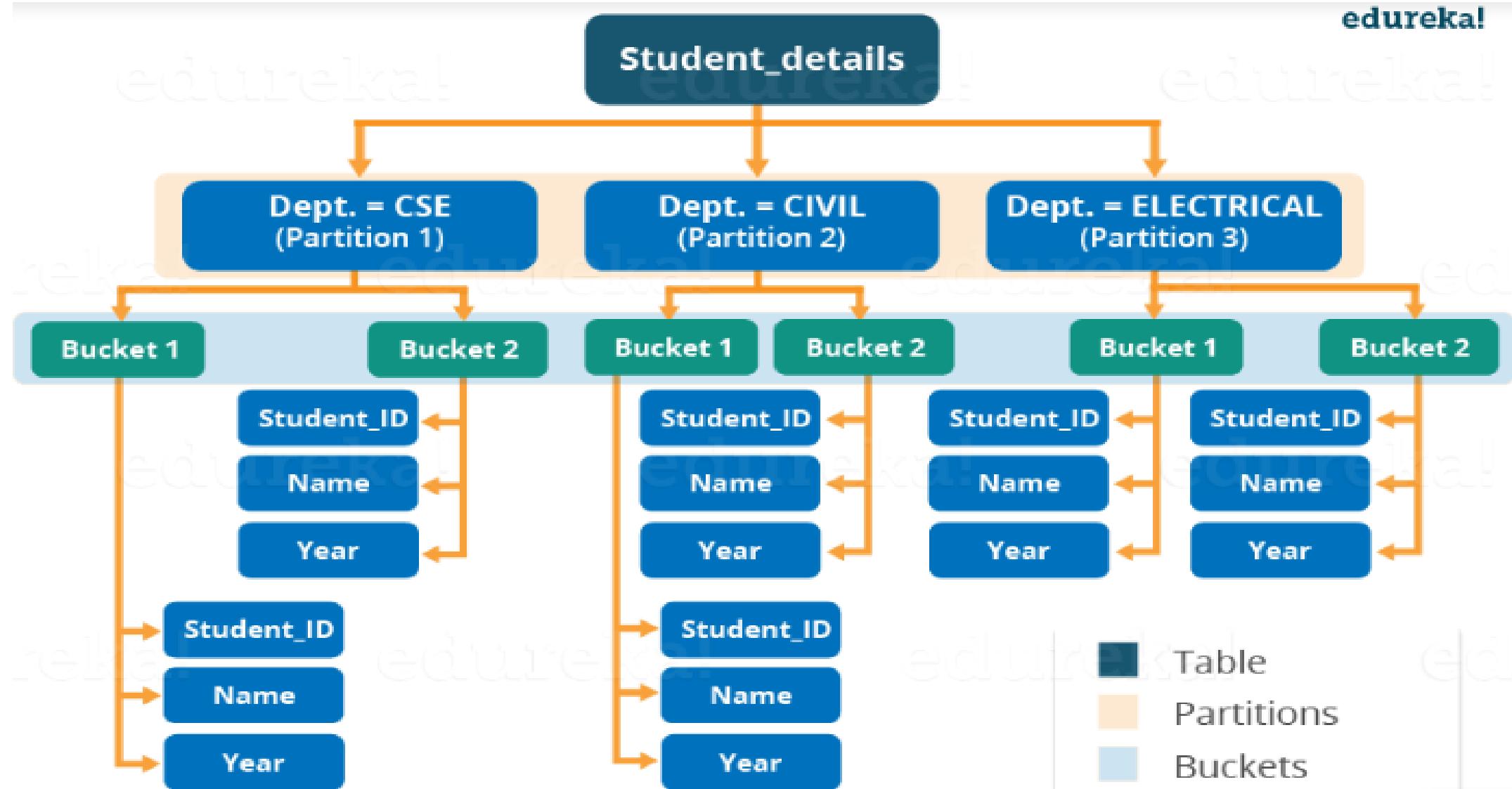
- **Partitions:**
- `CREATE TABLE table_name (column1 data_type, column2 data_type)
PARTITIONED BY (partition1 data_type, partition2 data_type,...);`
- Hive organizes tables into partitions for grouping similar type of data together based on a column or partition key. Each Table can have one or more partition keys to identify a particular partition. This allows us to have a faster query on slices of the data.
- **Note:** Remember, the most common mistake made while creating partitions is to specify an existing column name as a partition column. While doing so, you will receive an error – “Error in semantic analysis: Column repeated in partitioning columns”.

Hive Data Model

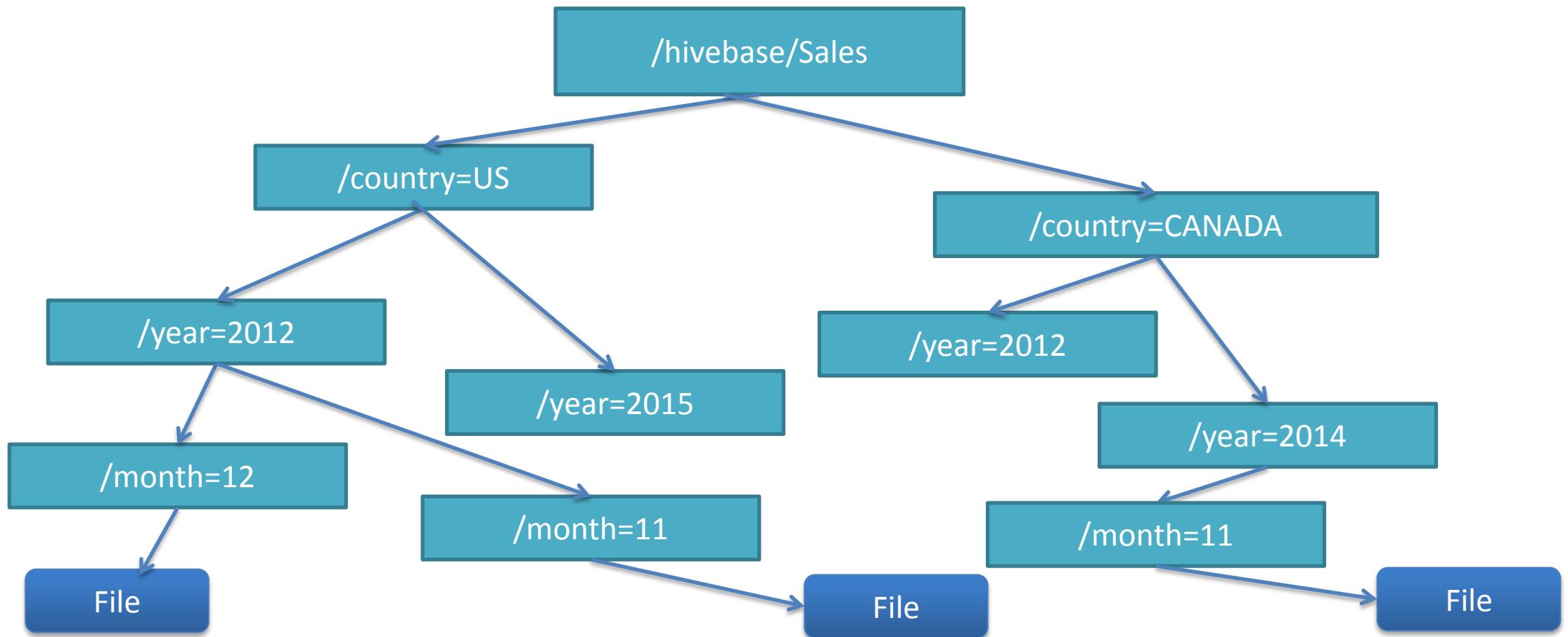
- Let us understand partition by taking an example where I have a table student_details containing the student information of some engineering college like student_id, name, department, year, etc. Now, if I perform partitioning based on department column, the information of all the students belonging to a particular department will be stored together in that very partition. Physically, a partition is nothing but a sub-directory in the table directory.
- Let's say we have data for three departments in our student_details table – CSE, ECE and Civil. Therefore, we will have three partitions in total for each of the departments as shown in the image below. And, for each department we will have all the data regarding that very department residing in a separate sub-directory under the Hive table directory. For example, all the student data regarding CSE departments will be stored in user/hive/warehouse/student_details/dept.=CSE. So, the queries regarding CSE students would only have to look through the data present in the CSE partition. This makes partitioning very useful as it reduces the query latency by scanning only **relevant** partitioned data instead of the whole data set. In fact, in real world implementations, you will be dealing with hundreds of TBs of data. So, imagine scanning this huge amount of data for some query where **95%** data scanned by you was un-relevant to your query.

Hive Data Model

edureka!



Hierarchy of Hive Partitions



Hive Data Model

- **Buckets:**
- CREATE TABLE table_name PARTITIONED BY (partition1 data_type, partition2 data_type,...) CLUSTERED BY (column_name1, column_name2, ...) SORTED BY (column_name [ASC|DESC], ...)] INTO num_buckets BUCKETS;
- Now, you may divide each partition or the unpartitioned table into Buckets based on the hash function of a column in the table. Actually, each bucket is just a file in the partition directory or the table directory (unpartitioned table). Therefore, if you have chosen to divide the partitions into n buckets, you will have n files in each of your partition directory. For example, you can see the above image where we have bucketed each partition into 2 buckets. So, each partition, say CSE, will have two files where each of them will be storing the CSE student's data.

Hive Data Model

- **How Hive distributes the rows into buckets?**
 - Hive determines the bucket number for a row by using the formula: **hash_function (bucketing_column) modulo (num_of_buckets)**. Here, hash_function depends on the column data type. For example, if you are bucketing the table on the basis of some column, let's say user_id, of INT datatype, the hash_function will be – **hash_function (user_id)= integer value of user_id**. And, suppose you have created two buckets, then Hive will determine the rows going to bucket 1 in each partition by calculating: (value of user_id) modulo (2). Therefore, in this case, rows having user_id ending with an even integer digit will reside in a same bucket corresponding to each partition. The hash_function for other data types is a bit complex to calculate and in fact, for a string it is not even humanly recognizable.

Hive Data Model

- **Why do we need buckets?**
- There are two main reasons for performing bucketing to a partition:
 - A map side join requires the data belonging to a unique join key to be present in the same partition. But what about those cases where your partition key differs from join? Therefore, in these cases you can perform a map side join by bucketing the table using the join key.
 - Bucketing makes the sampling process more efficient and therefore, allows us to decrease the query time.

Data Types in Apache Hive

- Hive data types are divided into the following 5 different categories:
 - Numeric Type: TINYINT, SMALLINT, INT, BIGINT
 - Date/Time Types: TIMESTAMP, DATE, INTERVAL
 - String Types: STRING, VARCHAR, CHAR
 - Complex Types: STRUCT, MAP, UNION, ARRAY
 - Misc Types: BOOLEAN, BINARY

Data Type	Description
TINYINT	It is a 1-byte signed integer ranges from -128 to 127
SMALLINT	It is a 2-byte signed integer ranges from -10 ¹⁵ to 10 ¹⁵ - 1
INT	It is a 4-byte signed integer ranges from -10 ³¹ to 10 ³¹ - 1
BIGINT	It is a 8-byte signed integer ranges from -10 ⁶³ to 10 ⁶³ - 1
FLOAT	It is a 4-byte single precision floating point number.
DOUBLE	It is a 8-byte double precision floating point number.
TIMESTAMP	It follows the format "YYYY-MM-DD HH:MM:SS.fffffffff" with 9 decimal place precision and also has the option of UNIX timestamp with nanosecond precision.
STRING	String literals can be expressed with either single quotes ('') or double quotes ("").
VARCHAR	It is a variable length type that ranges 1 and 65535, which specifies the maximum number of characters.
CHAR	It is a fixed-length type whose maximum length is fixed at 255.
MAP	It contains the key-value tuples where the fields are accessed using array notation.
ARRAY	It is a collection of similar types of values that are indexable using zero-based integers.

Features of Apache Hive

- Hive is designed for querying and managing mostly 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 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

Features of Apache Hive

- 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

Advantages/Disadvantages of Apache Hive

- Uses SQL like query language which is already familiar to most of the developers so makes it easy to use.
- It is highly scalable, you can use it to process any size of data.
- Supports multiple databases like MySQL, derby, Postgres, and Oracle for its metastore.
- Supports multiple data formats also allows indexing, partitioning, and bucketing for query optimization.
- Can only deal with cold data and is useless when it comes to processing real-time data.
- It is comparatively slower than some of its competitors. If your use-case is mostly about batch processing then Hive is well and fine.

REFERENCES

- <https://hive.apache.org/>
- <http://www.qubole.com/blog/big-data/hive-best-practices/>

Pig

- Pig came into existence to solve issues with MapReduce.
- **Birth of Pig**
 - Although MapReduce helped process and analyze Big Data faster, it had its flaws. Individuals who were unfamiliar with programming often found it challenging to write lengthy Java codes. Eventually, it became a difficult task to maintain and optimize the code, and as a result, the processing time increased.
 - This was the reason Yahoo faced problems when it came to processing and analyzing large datasets. Apache Pig was developed to analyze large datasets without using time-consuming and complex Java codes. Pig was explicitly developed for non-programmers.

Pig

- **What is Pig in Hadoop?**
- Pig is a scripting platform that runs on Hadoop clusters, designed to process and analyze large datasets. Pig uses a language called **Pig Latin**, which is similar to SQL. This language does not require as much code in order to analyze data. Although it is similar to SQL, it does have significant differences. In Pig Latin, **10 lines of code is equivalent to 200 lines in Java**. This, in turn, results in shorter development times.



Pig

- What stands out about Pig is that it operates on various types of data, including **structured, semi-structured, and unstructured** data. Whether you're working with structured, semi-structured, or unstructured data, Pig takes care of it all.
- To analyze data using Apache Pig, programmers need to write scripts using Pig Latin language.
- All these scripts are internally converted to Map and Reduce tasks.
- Apache Pig has a component known as Pig Engine that accepts the Pig Latin scripts as input and converts those scripts into MapReduce jobs.

Why do we need Apache Pig?

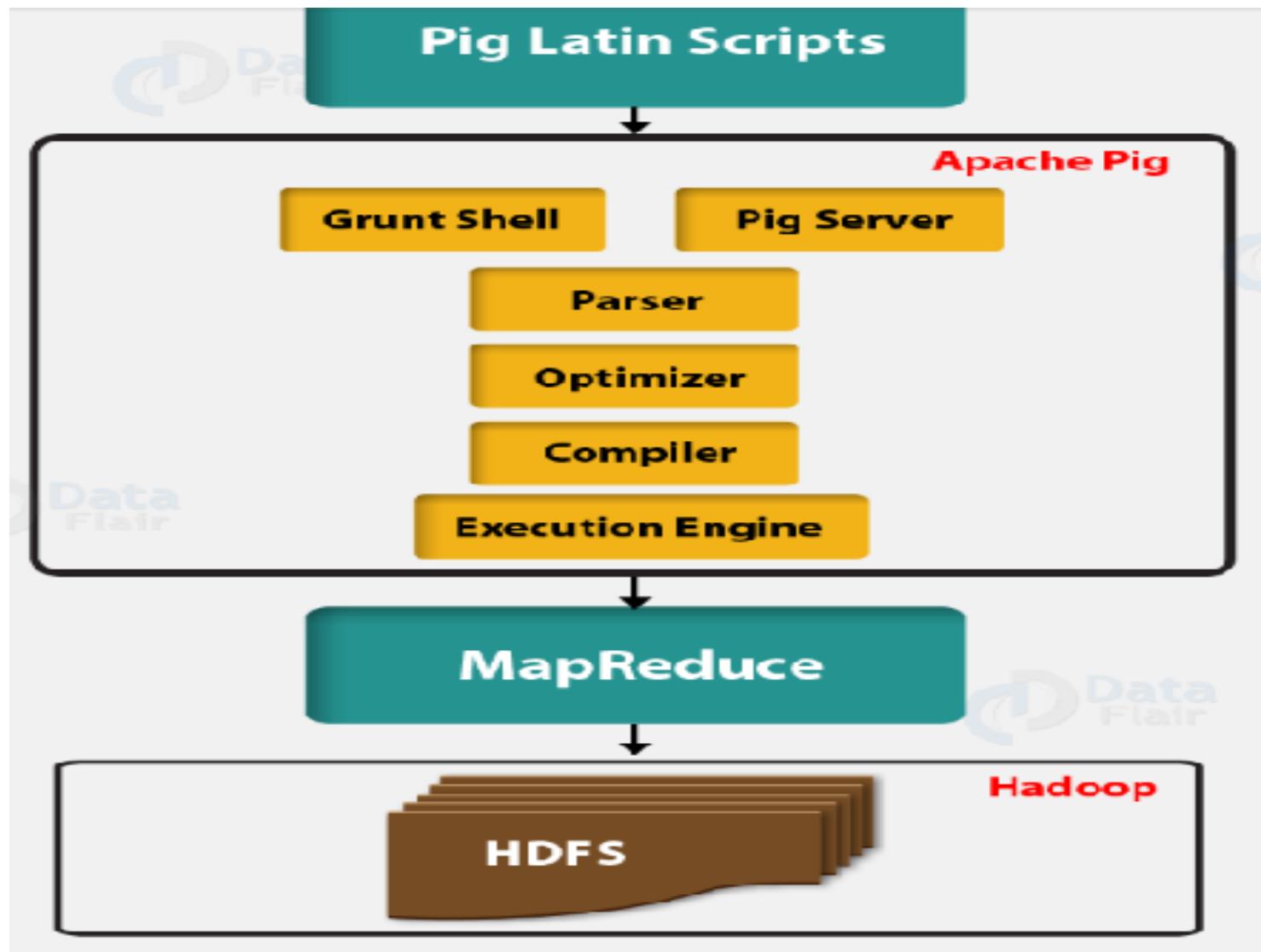
- Using Pig Latin, programmers can perform MapReduce tasks easily without having to type complex codes in Java.
- Apache Pig uses multi-query approach, thereby reducing the length of codes. For example, an operation that would require you to type 200 lines of code (LoC) in Java can be easily done by typing as less as just 10 LoC in Apache Pig. Ultimately, Apache Pig reduces the development time by almost 16 times.
- Pig Latin is SQL-like language and it is easy to learn Apache Pig when you are familiar with SQL.
- Apache Pig provides many built-in operators to support data operations like joins, filters, ordering, etc. In addition, it also provides nested data types like tuples, bags, and maps that are missing from MapReduce.

Pig Architecture

- In Pig, there is a language we use to analyze data in **Hadoop**. That is what we call Pig Latin. Also, it is a high-level data processing language that offers a rich set of data types and operators to perform several operations on the data.
- Moreover, in order to perform a particular task, programmers need to write a **Pig script using the Pig Latin language** and execute them using any of the execution mechanisms (Grunt Shell, UDFs, Embedded) using Pig.
- To produce the desired output, these scripts will go through a series of transformations applied by the Pig Framework, after execution.
- Further, Pig converts these scripts into a series of **MapReduce** jobs internally. Therefore it makes the programmer's job easy. Here, is the architecture of Apache Pig.

Pig Architecture

- Architecture

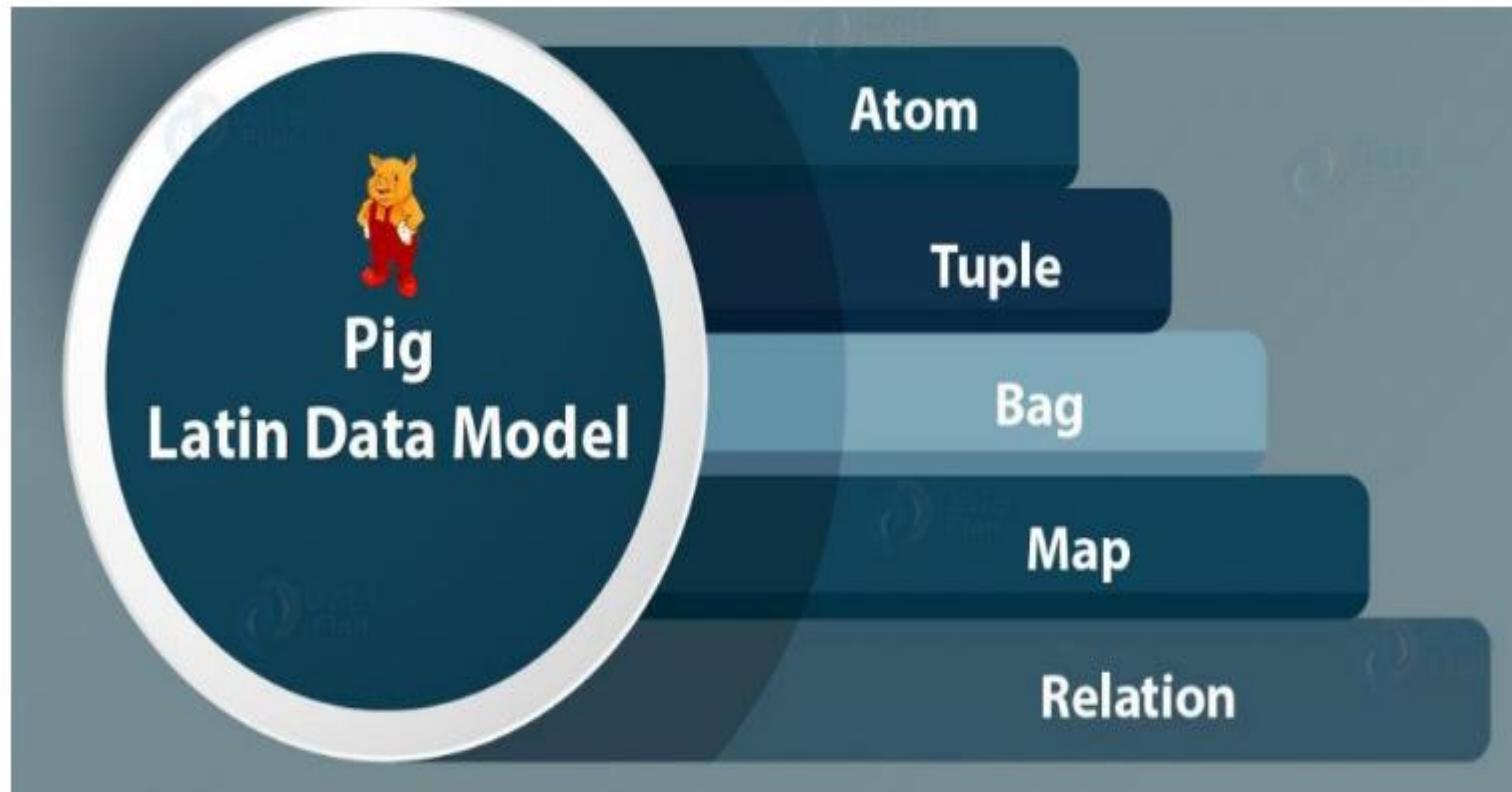


Apache Pig Components

- **Parser:** At first, all the Pig Scripts are handled by the Parser. Parser basically checks the syntax of the script, does type checking, and other miscellaneous checks. Afterwards, Parser's output will be a DAG (directed acyclic graph) that represents the Pig Latin statements as well as logical operators.
 - The logical operators of the script are represented as the nodes and the data flows are represented as edges in DAG (the logical plan)
- **Optimizer:** Afterwards, the logical plan (DAG) is passed to the logical optimizer. It carries out the logical optimizations further such as projection and push down.
- **Compiler:** Then compiler compiles the optimized logical plan into a series of MapReduce jobs.
- **Execution engine:** Eventually, all the MapReduce jobs are submitted to Hadoop in a sorted order. Ultimately, it produces the desired results while these MapReduce jobs are executed on Hadoop.

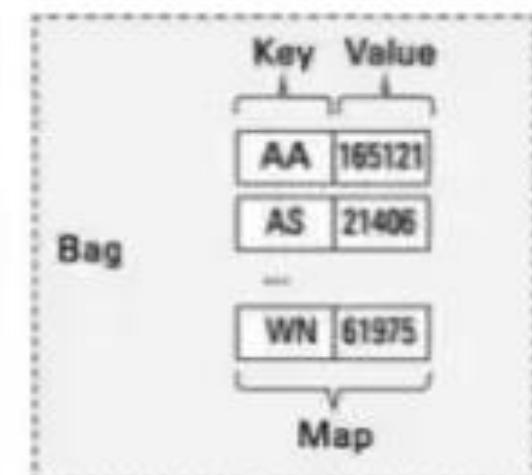
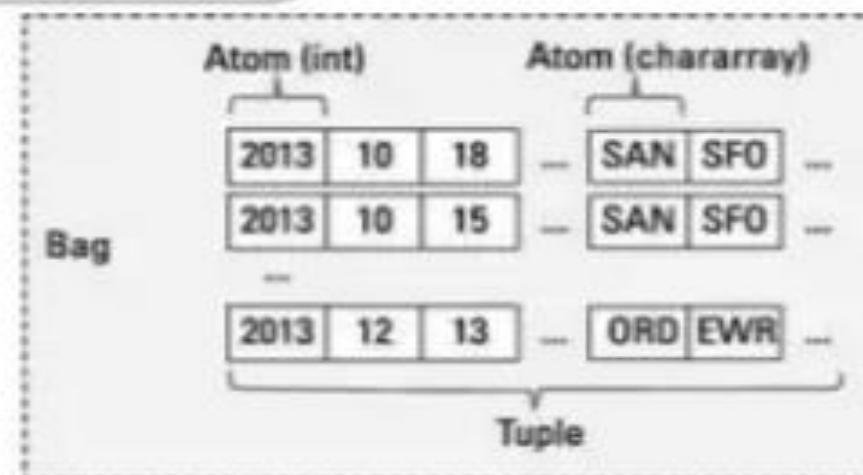
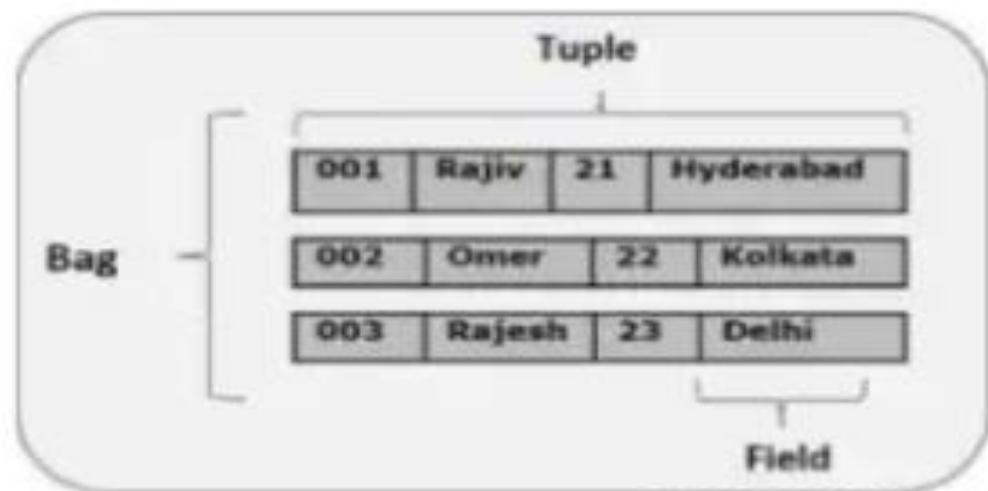
Pig Latin Data Model

- Pig Latin data model is fully nested. Also, it allows complex non-atomic data types like map and tuple.



Pig Latin Data Model

- Apache Pig – Elements



Pig Latin Data Model

- **Atom**

- Atom is defined as any single value in Pig Latin, irrespective of their data. Basically, we can use it as string and number and store it as the string. Atomic values of Pig are int, long, float, double, char array, and byte array. Moreover, a field is a piece of data or a simple atomic value in Pig.

- For Example – ‘Shubham’ or ‘25’

- **Tuple**

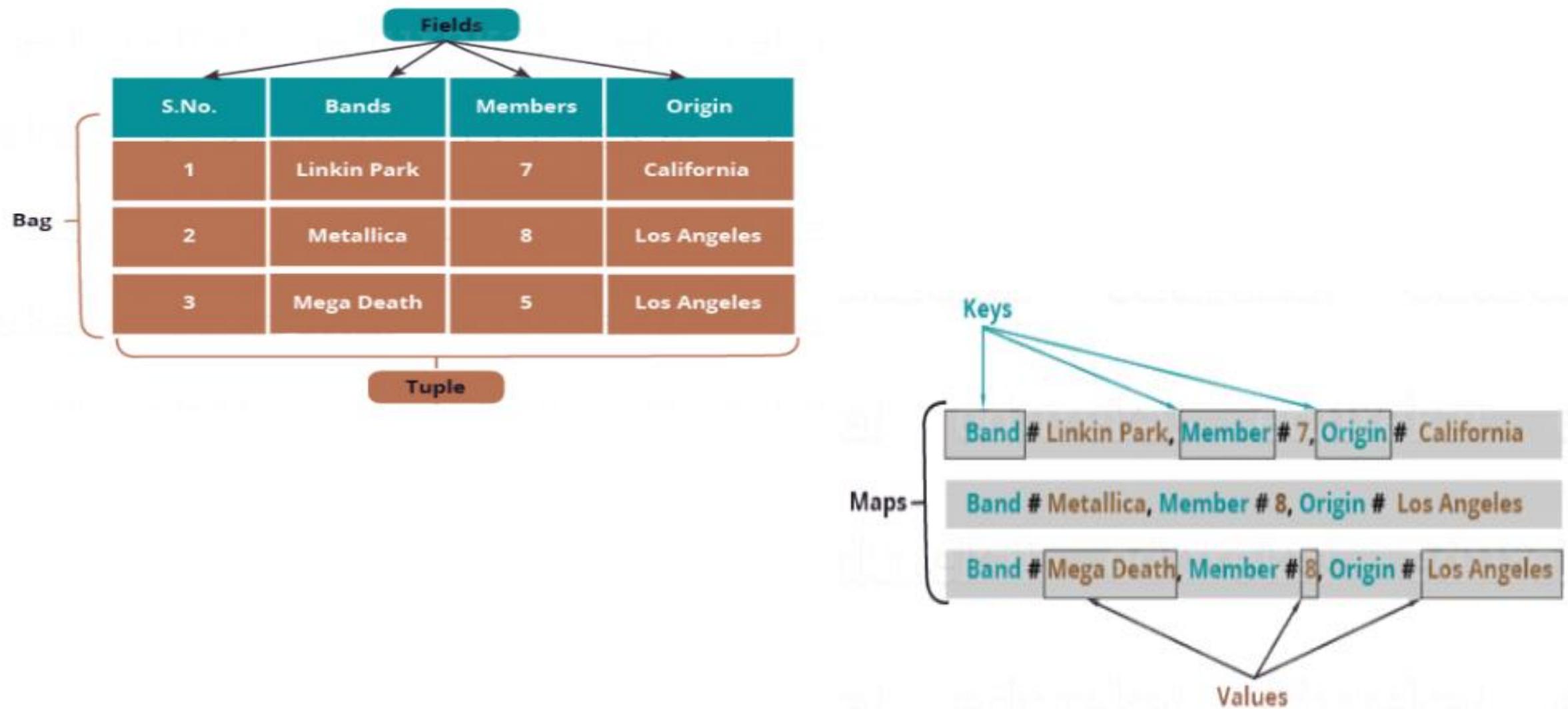
- Tuple is a record that is formed by an ordered set of fields. However, the fields can be of any type. In addition, a tuple is similar to a row in a table of RDBMS.

For Example – (Shubham, 25)

Pig Latin Data Model

- **Bag**
 - An unordered set of tuples is what we call Bag. To be more specific, a Bag is a collection of tuples (non-unique). Moreover, each tuple can have any number of fields (flexible schema). Generally, we represent a bag by '{}'.
 - For Example – {(Shubham, 25), (Pulkit, 35)}
 - In addition, when a bag is a field in a relation, in that way it is known as the **inner bag**.
 - Example – {Shubham, 25, {9826022258, Shubham@gmail.com,}}
- **Map**
 - A set of key-value pairs is what we call a map (or data map). Basically, the key needs to be of type char array and should be unique. Also, the value might be of any type. And, we represent it by '[]'
 - For Example – [name#Shubham, age#25]

Pig Latin Data Model



Pig Latin Scripts

- **Pig Latin Scripts**
- We submit Pig scripts to the Apache Pig execution environment which can be written in Pig Latin using built-in operators.
- There are three ways to execute the Pig script:
 - **Script File:** Write all the Pig commands in a script file and execute the Pig script file. This is executed by the Pig Server.
 - **Grunt Shell:** This is Pig's interactive shell provided to execute all Pig Scripts.
 - **Embedded Script:** If some functions are unavailable in built-in operators, we can programmatically create User Defined Functions to bring that functionalities using other languages like Java, Python, Ruby, etc. and embed it in Pig Latin Script file. Then, execute that script file.

Job Execution Flow

- The developer creates the scripts, and then it goes to the local file system as functions. Moreover, when the developers submit Pig Script, it contacts with Pig Latin Compiler.
- The compiler then splits the task and run a series of MR jobs. Meanwhile, Pig Compiler retrieves data from the **HDFS**. The output file again goes to the HDFS after running MR jobs.

Pig Execution Modes

- We can run Pig in two execution modes. These modes depend upon where the Pig script is going to run. It also depends on where the data is residing. We can thus store data on a single machine or in a distributed environment like Clusters.
- **Pig Local mode**
 - In this mode, pig implements on **single JVM** and access the file system. This mode is better for dealing with the **small data sets**.
 - Start the pig in local mode:
 - **pig -x local**
 - While the user can provide **-x local** to get into Pig local mode of execution. Therefore, Pig always looks for the **local file system path while loading data**.

Pig Execution Modes

- **Pig Map Reduce Mode**
 - In this mode, a user could have proper **Hadoop cluster setup** and installations on it. By default, Apache Pig installs as in MR mode. The Pig also translates the queries into Map reduce jobs and runs on top of Hadoop cluster. Hence, this mode as a Map reduce runs on a **distributed cluster**.
 - Start the pig in mapreduce mode (needs hadoop datanode started):
 - **pig -x mapreduce**
 - The statements like LOAD, STORE read the data from the HDFS file system and to show output. These Statements are also used to process data.

Features of Pig

- **Rich set of operators:** It provides many operators to perform operations like join, sort, filer, etc.
- **Ease of programming:** Pig Latin is similar to SQL and it is easy to write a Pig script if you are good at SQL.
- **Optimization opportunities:** The tasks in Apache Pig optimize their execution automatically, so the programmers need to focus only on semantics of the language.
- **Extensibility:** Using the existing operators, users can develop their own functions to read, process, and write data.
- **UDF's:** Pig provides the facility to create User-defined Functions in other programming languages such as Java and invoke or embed them in Pig Scripts.
- **Handles all kinds of data:** Apache Pig analyzes all kinds of data, both structured as well as unstructured. It stores the results in HDFS.

Applications of Apache Pig

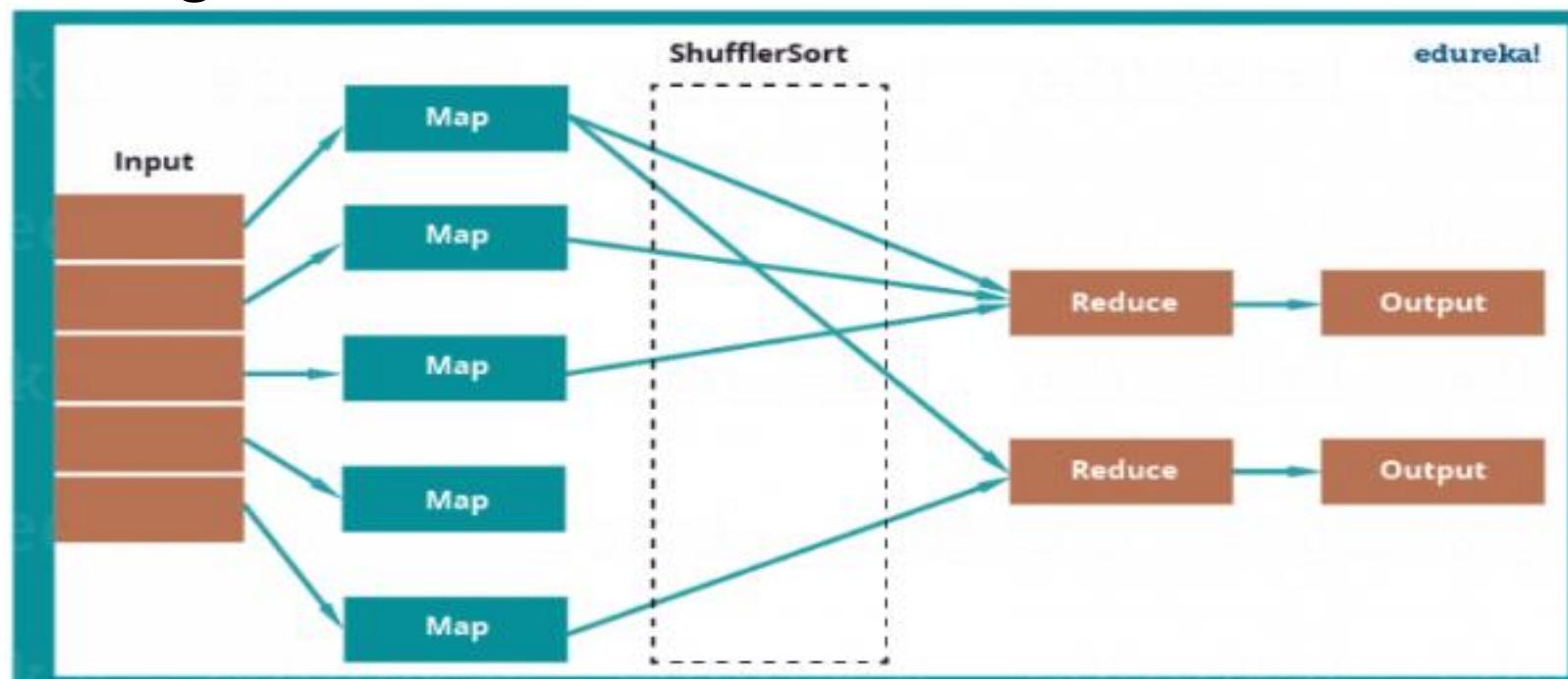
- To process huge data sources such as web logs, streaming online data, etc.
- To perform data processing for search platforms (different types of data needs to be processed) like ***Yahoo uses Pig for 40% of their jobs including news feeds and search engine.***
- To process time sensitive data loads. Here, data needs to be extracted and analyzed quickly. E.g. machine learning algorithms require time sensitive data loads, like twitter needs to quickly extract data of customer activities (i.e. tweets, re-tweets and likes) and analyze the data to find patterns in customer behaviors, and make recommendations immediately like trending tweets.

Twitter Case Study

- Their major aim was to analyse data stored in Hadoop to come up with the following insights on a daily, weekly or monthly basis.
- **Counting operations:**
 - How many requests twitter serve in a day?
 - What is the average latency of the requests?
 - How many searches happens each day on Twitter?
 - How many unique queries are received?
 - How many unique users come to visit?
- **Correlating Big Data:**
 - Cohort analysis: analyzing data by categorizing user, based on their behavior.
 - What goes wrong while site problem occurs?
 - Search correction and search suggestions.
- **Research on Big Data & produce better outcomes like:**
 - What is the ratio of the follower to following?

Twitter Case Study

- So, for analyzing data, Twitter used MapReduce initially, which is parallel computing over HDFS. For example, they wanted to analyse how many tweets are stored per user, in the given tweet table?
- Using MapReduce, this problem will be solved sequentially as shown in the below image:

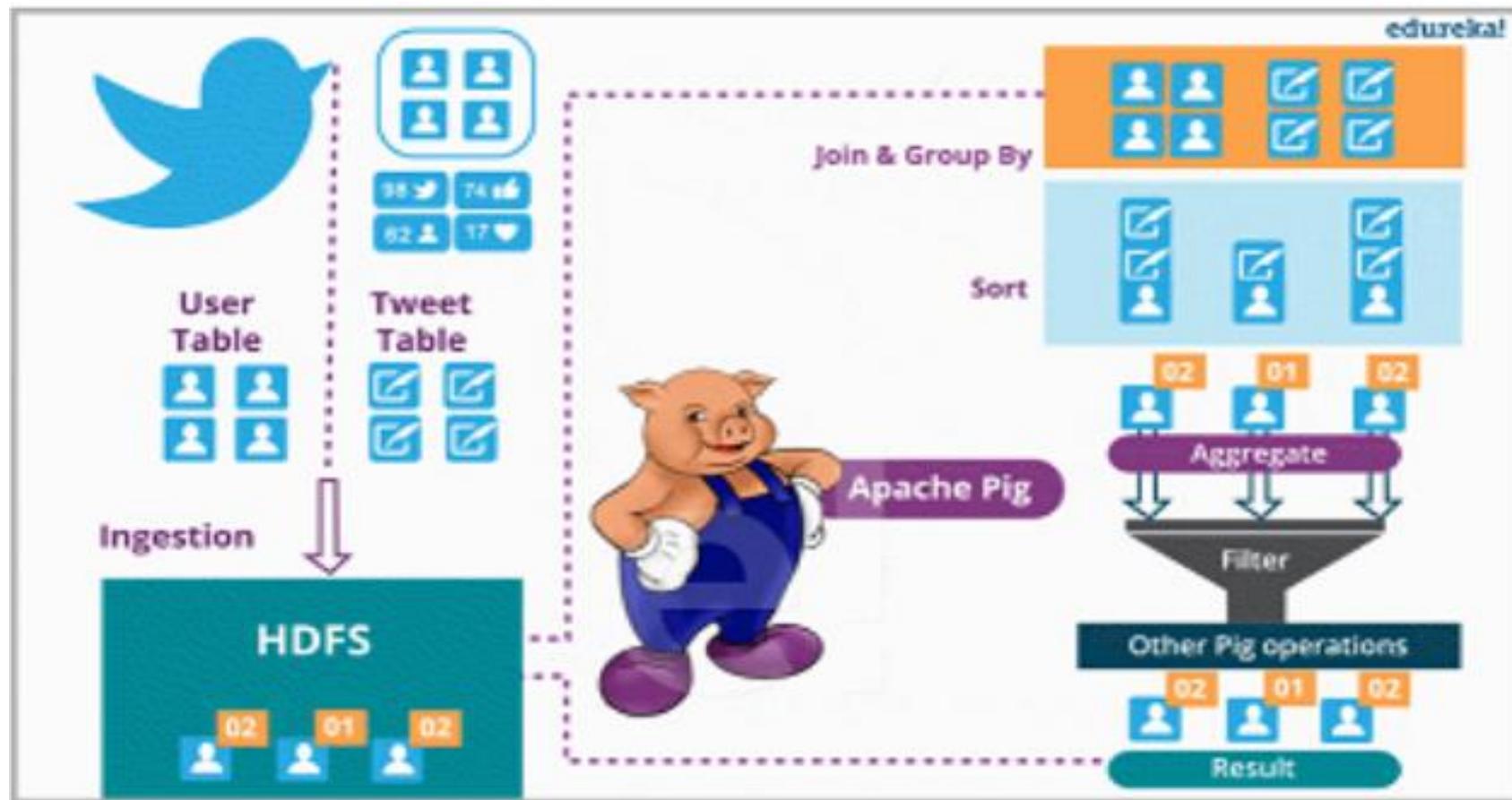


Twitter Case Study

- MapReduce program first inputs the key as rows and sends the tweet table information to mapper function. Then the Mapper function will select the user id and associate unit value (i.e. 1) to every user id. The Shuffle function will sort same user ids together. At last, Reduce function will add all the number of tweets together belonging to same user. The output will be user id, combined with user name and the number of tweets per user.
- But while using MapReduce, they faced some limitations:
 - Analysis needs to be typically done in Java.
 - Joins, that are performed, needs to be written in Java, which makes it longer and more error-prone.
 - For projection and filters, custom code needs to be written which makes the whole process slower.

Twitter Case Study

- So, Twitter moved to Apache Pig for analysis. Now, joining data sets, grouping them, sorting them and retrieving data becomes easier and simpler.

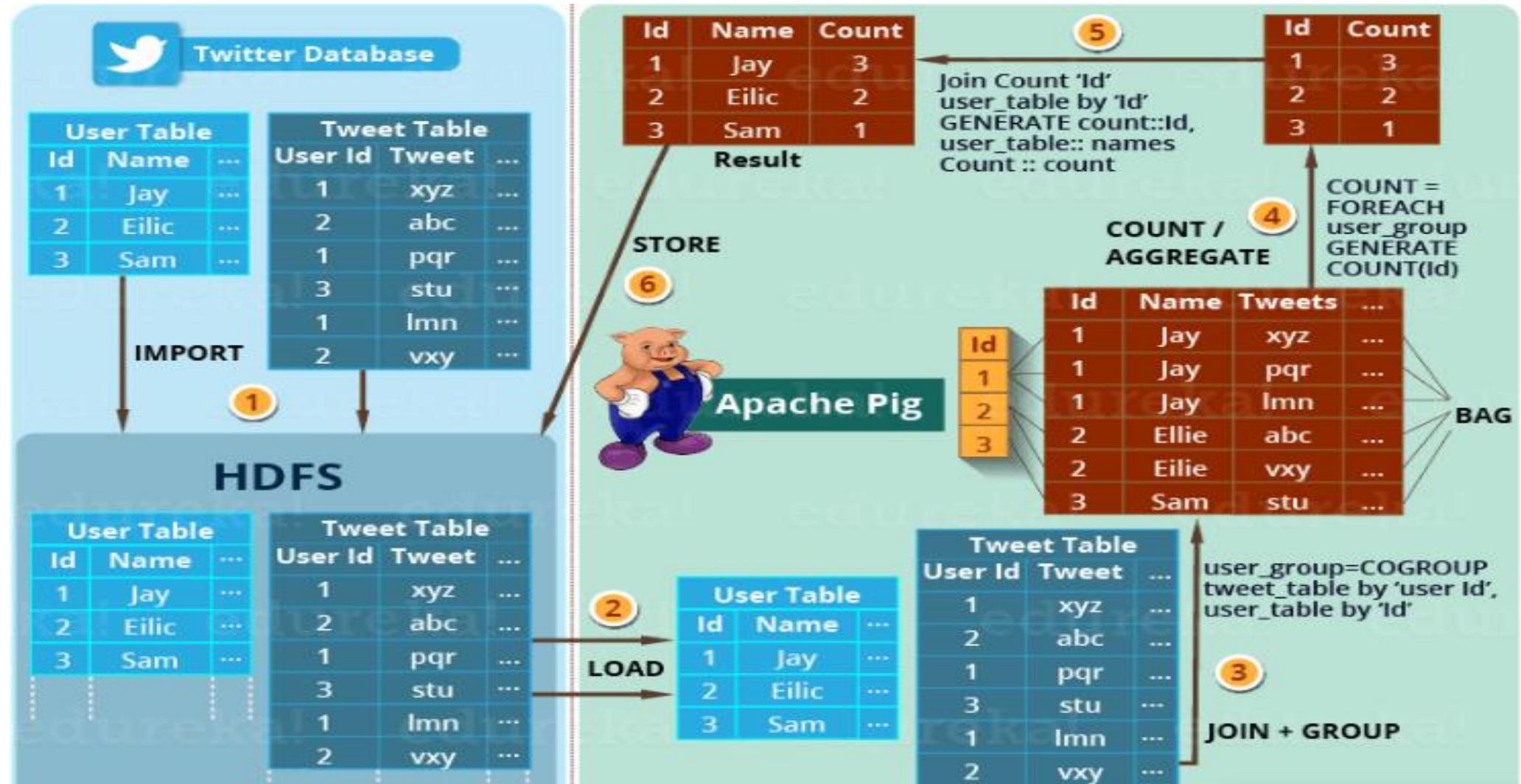


Twitter Case Study

- Twitter had both semi-structured data like Twitter Apache logs, Twitter search logs, Twitter MySQL query logs, application logs and structured data like tweets, users, block notifications, phones, favorites, saved searches, re-tweets, authentications, SMS usage, user followings, etc. which can be easily processed by Apache Pig.
- Twitter dumps all its archived data on HDFS. It has two tables i.e. user data and tweets data. User data contains information about the users like username, followers, followings, number of tweets etc. While Tweet data contains tweet, its owner, number of re-tweets, number of likes etc. Now, twitter uses this data to analyse their customer's behaviors and improve their past experiences.

Twitter Case Study

- Q: Analyzing how many tweets are stored per user, in the given tweet tables?



Twitter Case Study

- **STEP 1**– First of all, twitter imports the twitter tables (i.e. user table and tweet table) into the HDFS.
- **STEP 2**– Then Apache Pig loads (**LOAD**) the tables into Apache Pig framework.
- **STEP 3**– Then it joins and groups the tweet tables and user table using **COGROUP** command. This results in the inner Bag Data type.
- Example of Inner bags produced–
 - (1,{{(1,Jay,xyz),(1,Jay,pqr),(1,Jay,lmn)}})
 - (2,{{(2,Ellie,abc),(2,Ellie,vxy)}})
 - (3, {{(3,Sam,stu)}})
- **STEP 4**– Then the tweets are counted according to the users using **COUNT** command. So, that the total number of tweets per user can be easily calculated.

Twitter Case Study

- Example of tuple produced as (id, tweet count) (refer to the above image) –
 - (1, 3)
 - (2, 2)
 - (3, 1)
- **STEP 5**– At last the result is joined with user table to extract the user name with produced result.
- Example of tuple produced as (id, name, tweet count)
 - (1, **Jay**, 3)
 - (2, **Ellie**, 2)
 - (3, **Sam**, 1)
- **STEP 6**– Finally, this result is stored back in the HDFS.

Hive vs. Pig

Features	Hive	Pig
Language	Hive uses a declarative language called HiveQL	With Pig Latin, a procedural data flow language is used
Data Processing	Hive is used for batch processing	Pig is a high-level data-flow language
Partitions	Yes	No. Pig does not support partitions although there is an option for filtering
Web interface	Hive has a web interface	Pig does not support web interface
User Specification	Data analysts are the primary users	Programmers and researchers use Pig
Used for	Reporting	Programming

Hive vs. Pig

Features	Hive	Pig
Type of data	Hive works on structured data. Does not work on other types of data	Pig works on structured, semi-structured and unstructured data
Operates on	Works on the server-side of the cluster	Works on the client-side of the cluster
Avro File Format	Hive does not support Avro	Pig supports Avro
Loading Speed	Hive takes time to load but executes quickly	Pig loads data quickly
JDBC/ ODBC	Supported, but limited	Unsupported



Hive v/s Pig



Similarities:

- Both High level Languages which work on top of map reduce framework
- Can coexist since both use the under lying HDFS and map reduce

Differences:

◆ **Language**

- Pig is a procedural ; (A = load 'mydata'; dump A)
- Hive is Declarative (select * from A)

◆ **Work Type**

- Pig more suited for adhoc analysis (on demand analysis of click stream search logs)
- Hive a reporting tool (e.g. weekly BI reporting)



Hive v/s Pig



Differences:

◆ **Users**

- Pig – Researchers, Programmers (build complex data pipelines, machine learning)
- Hive – Business Analysts

◆ **Integration**

- Pig - Doesn't have a thrift server(i.e no/limited cross language support)
- Hive - Thrift server

◆ **User's need**

- Pig – Better dev environments, debuggers expected
- Hive - Better integration with technologies expected(e.g JDBC, ODBC)

Hive vs. Pig vs. Hadoop MapReduce

	Hive	Pig	Hadoop MapReduce
Language	It has SQL like Query language.	It has the scripting language.	compiled language.
Abstraction	It has a Low level of Abstraction.	has the High level of Abstraction.	has the High level of Abstraction.
Line of codes	Comparatively less no. of the line of codes from both MapReduce and Pig.	Comparatively less no. of the line of codes from MapReduce.	It has More line of codes.
Development Efforts	Comparatively fewer development efforts from both MapReduce and Pig.	Comparatively less development effort.	More development effort is involved.
Code Efficiency	Code efficiency is relatively less.	Code efficiency is relatively less.	It has high Code efficiency.

Distributed Database

- Distributed databases are a collection of data stored at different sites of a computer network.
- A distributed database can be created by splitting and scattering the data of an existing database over different sites or by federating together multiple existing databases.
- Distributed database may be required when a particular database needs to be accessed by various users globally.
- It needs to be managed such that for the users it looks like one single database.

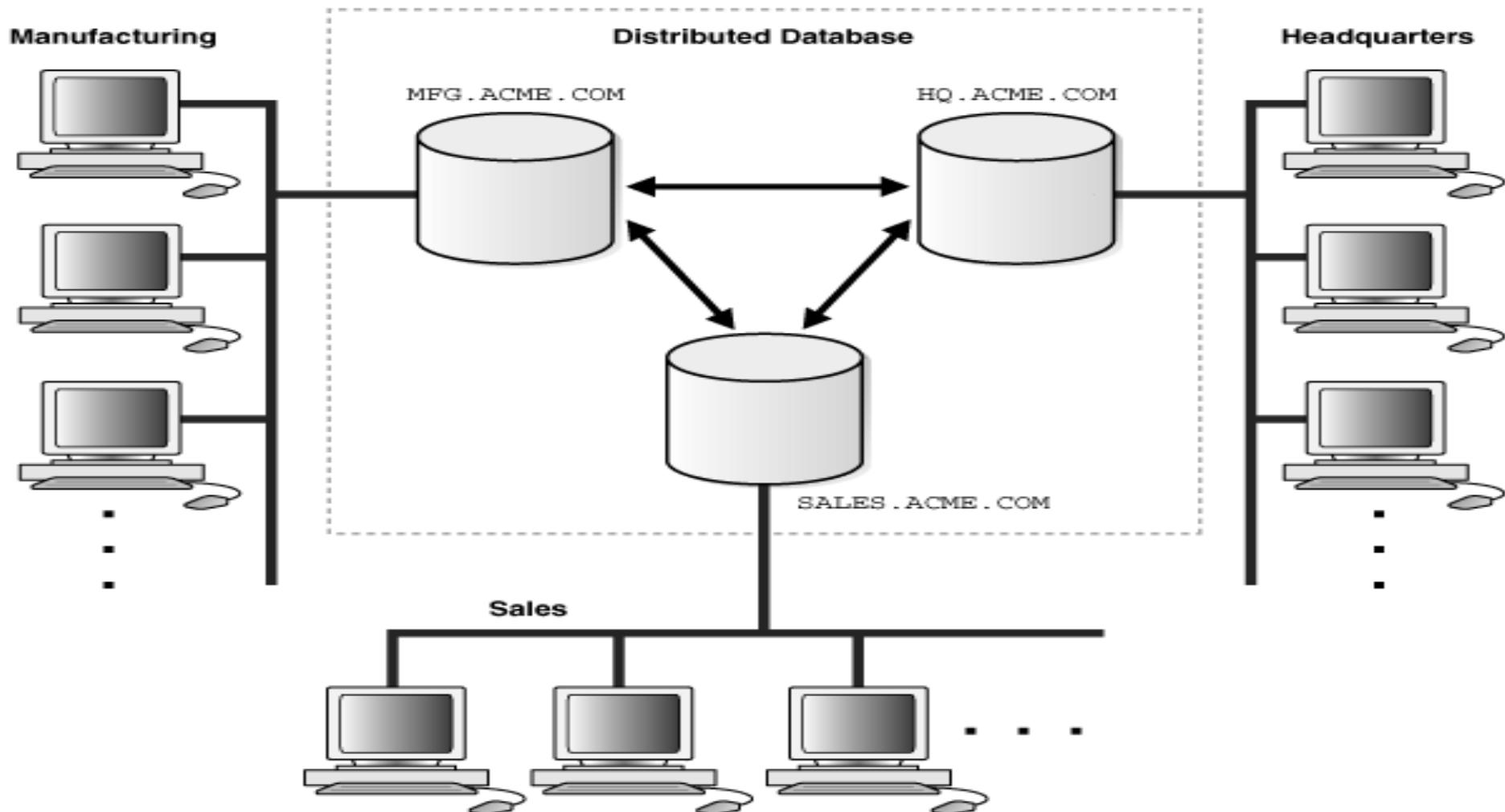
Types of Distributed Database

1. Homogeneous Database:

- In a homogeneous database, all different sites store database identically. The operating system, database management system and the data structures used – all are same at all sites. Hence, they're easy to manage.
- Types of Homogeneous Distributed Database
- **Autonomous** – Each database is independent that functions on its own. They are integrated by a controlling application and use message passing to share data updates.
- **Non-autonomous** – Data is distributed across the homogeneous nodes and a central or master DBMS co-ordinates data updates across the sites.

Types of Distributed Database

1. Homogeneous Database:



Types of Distributed Database

2. Heterogeneous Database:

- In a heterogeneous distributed database, different sites can use different schema and software that can lead to problems in query processing and transactions. Also, a particular site might be completely unaware of the other sites. Different computers may use a different operating system, different database application. They may even use different data models for the database.
- Hence, translations are required for different sites to communicate.

Distributed Data Storage

- There are 2 ways in which data can be stored on different sites. These are:

1. Replication –

- In this approach, the entire relation is stored redundantly at 2 or more sites. If the entire database is available at all sites, it is a fully redundant database. Hence, in replication, systems maintain copies of data.
- This is advantageous as it increases the availability of data at different sites. Also, now query requests can be processed in parallel. However, it has certain disadvantages as well. Data needs to be constantly updated. Any change made at one site needs to be recorded at every site that relation is stored or else it may lead to inconsistency. This is a lot of overhead. Also, concurrency control becomes way more complex as concurrent access now needs to be checked over a number of sites.

Distributed Data Storage

2. Fragmentation –

- In this approach, the relations are fragmented (i.e., they're divided into smaller parts) and each of the fragments is stored in different sites where they're required. It must be made sure that the fragments are such that they can be used to reconstruct the original relation (i.e, there isn't any loss of data).
- Fragmentation is advantageous as it doesn't create copies of data, consistency is not a problem.

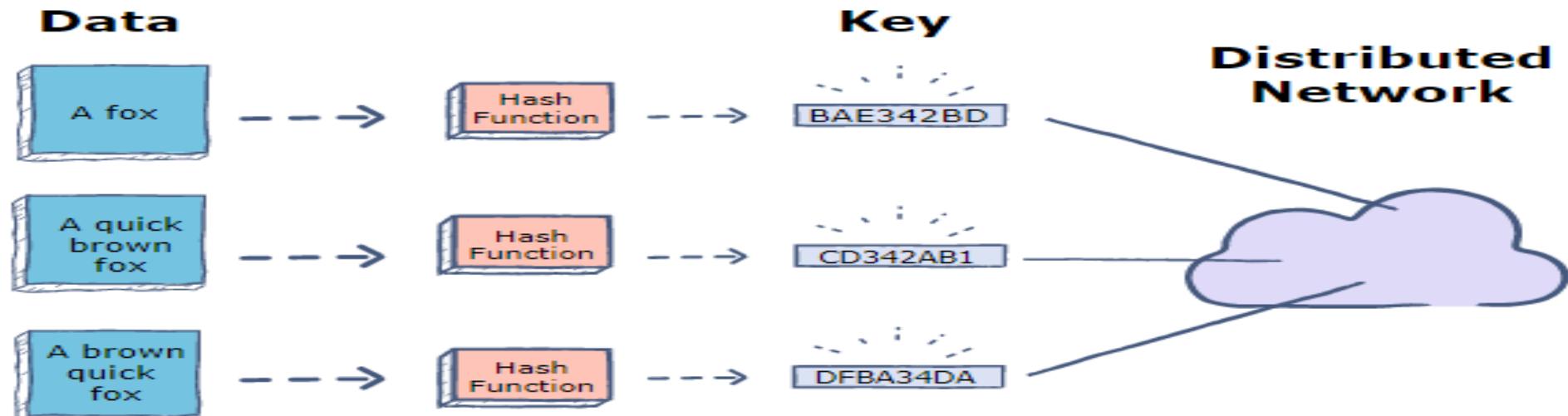
Distributed Hash Tables

- A distributed hash table (DHT) is a **decentralized** storage system that provides lookup and storage schemes similar to a hash table, storing key-value pairs.
- Each node in a DHT is responsible for keys along with the mapped values. Any node can efficiently retrieve the value associated with a given key. Just like in hash tables, values mapped against keys in a DHT can be any arbitrary form of data.
- Distributed hash tables are decentralized, so all nodes form the collective system without any centralized coordination.
- They are generally fault-tolerant because data is replicated across multiple nodes. Distributed hash tables can scale for large volumes of data across many nodes.

Distributed Hash Table

- Distributed Hash Tables are a form of a distributed database that can store and retrieve information associated with a key in a network of peer nodes that can join and leave the network at any time.
- The nodes coordinate among themselves to balance and store data in the network without any central coordinating party.
- DHT's require that information to be evenly distributed across the network. To achieve this goal, the concept of consistent hashing is used. A key is passed through a hash algorithm that serves as a randomization function. This ensures that each node in the network has an equal chance of being chosen to store the key/value pair.

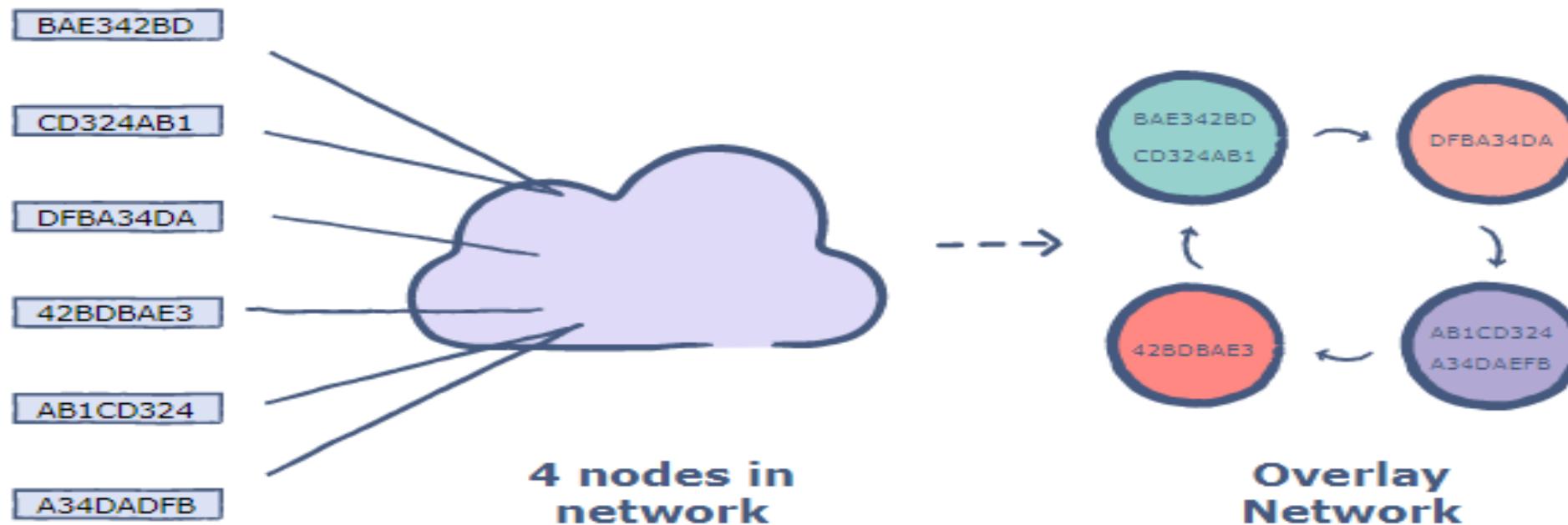
Distributed Hash Tables



- DHTs have the following properties:
- **Decentralised & Autonomous:** Nodes collectively form the system without any central authority.
- **Fault Tolerant:** System is reliable with lots of nodes joining, leaving, and failing at all times.
- **Scalable:** System should function efficiently with even thousands or millions of nodes.

Distributed Hash Tables

- Just like hash tables, DHTs support the following 2 functions:
 - put (key, value)
 - get (key)
- The nodes in a DHT are connected together through an **overlay network** in which neighboring nodes are connected. This network allows the nodes to find any given key in the key-space.



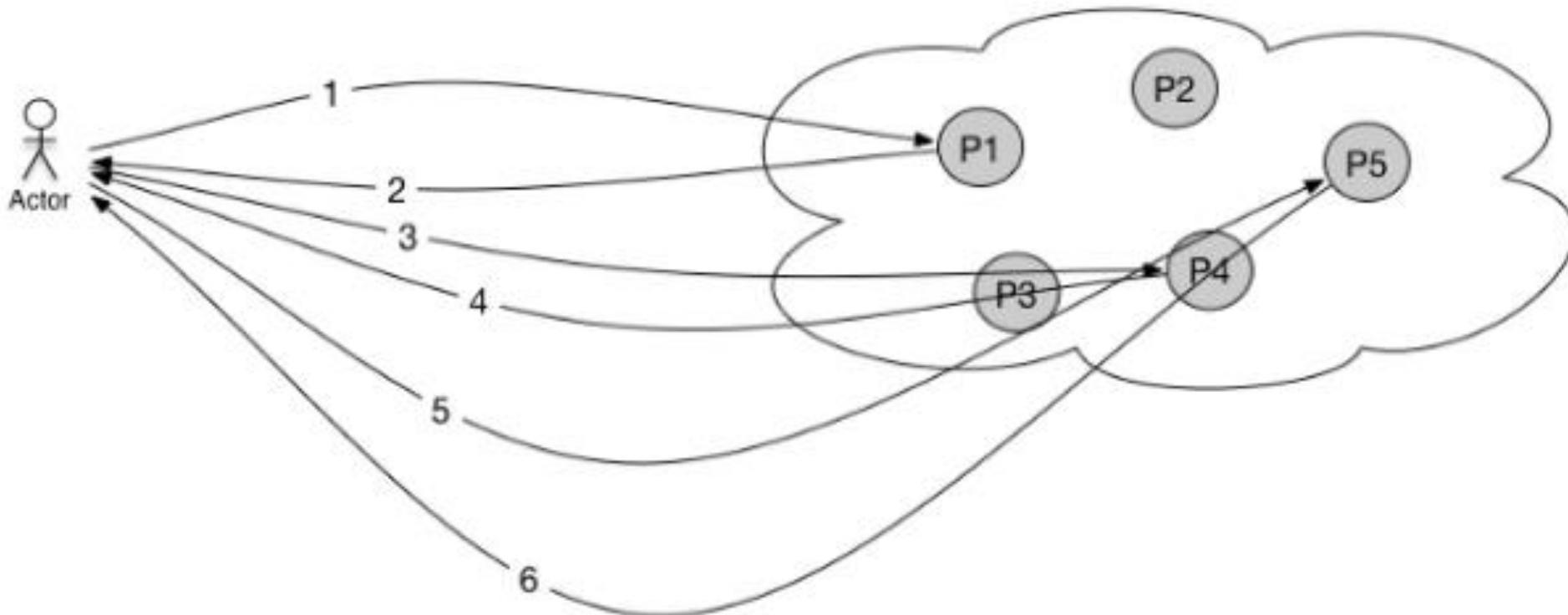
Why Is a Distributed Hash Table Used?

- Distributed hash tables provide an easy way to find information in a large collection of data because all keys are in a consistent format, and the entire set of keys can be partitioned in a way that allows fast identification on where the key/value pair resides.
- The nodes participating in a distributed hash table act as peers to find specific data values, as each node stores the key partitioning scheme so that if it receives a request to access a given key, it can quickly map the key to the node that stores the data. It then sends the request to that node.
- Also, nodes in a distributed hash table can be easily added or removed without forcing a significant amount of re-balancing of the data in the cluster. Cluster rebalancing, especially for large data sets, can often be a time-consuming task that also impacts performance. Having a quick and easy means for growing or shrinking a cluster ensures that changes in data size does not disrupt the operation of the applications that access data in the distributed hash table.

Iterative and Recursive Lookup

- There are two common styles of lookup operations.
- In an iterative lookup, a requesting node will query another node asking for a key/value pair. If that node does not have that node, it will return one or more nodes that are “closer”. The requesting node will then query the next closest node. This process repeats until either the key/value is found and returned, or the last queried node returns an error saying that the key simply cannot be found.
- With recursive lookups, the first requesting node will query the next closes node, which will then query the next closest node until the data is found. The data is then passed back along the chain of requests back to the requestor.

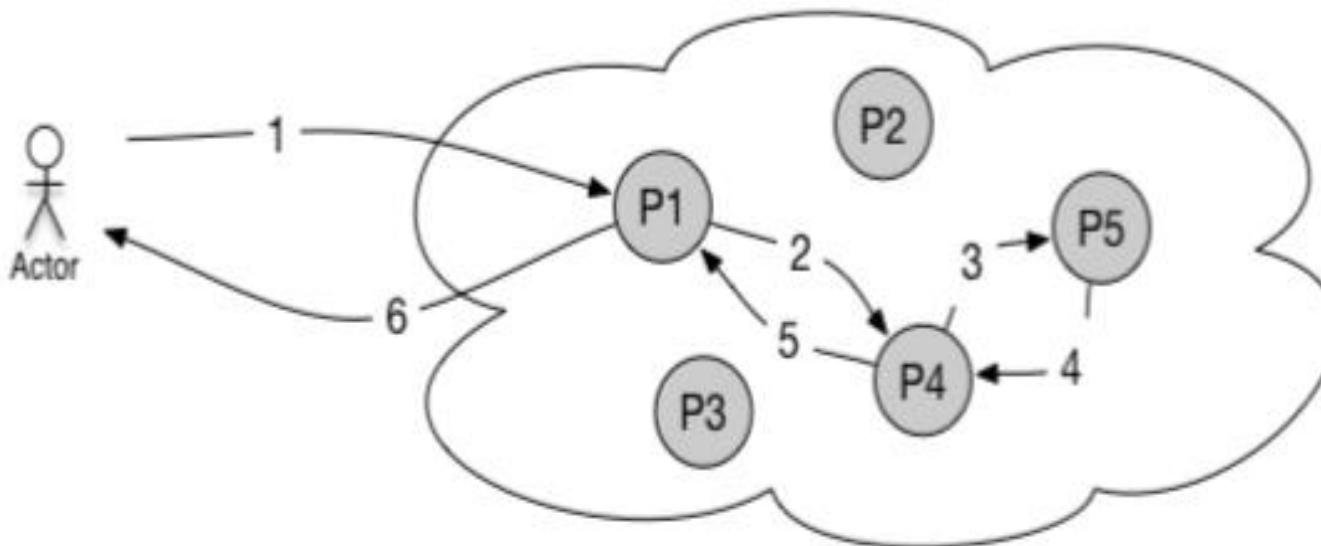
Iterative Lookup



1. STORE "MyKey" / "My Value"
2. I'm not responsible for "MyKey" - but P4 is closer
3. STORE "MyKey" / "My Value"
4. I'm not responsible for "MyKey" - but P5 is closer
5. STORE "MyKey" / "My Value"
6. OK - value is stored.

1. GET "MyKey"
2. I'm not responsible for "MyKey" - but P4 is closer
3. GET "MyKey"
4. I'm not responsible for "MyKey" - but P5 is closer
5. GET "MyKey"
6. OK - here is "My Value"

Recursive Lookup



1. STORE "MyKey" / "My Value"
2. I'm not responsible for "MyKey", P4 is closer, so ask P4
3. I'm not responsible for "MyKey", P5 is closer, so ask P5
4. OK, value is stored
5. OK, value is stored
6. OK, value is stored

1. GET "MyKey"
2. I'm not responsible for "MyKey", P4 is closer, so ask P4
3. I'm not responsible for "MyKey", P5 is closer, so ask P5
4. OK, here is "My Value"
5. OK, passing back "My Value"
6. OK, passing back "My Value"

Routing Table

- DHT nodes don't store all the data, there needs to be a routing layer so that any node can locate the node that stores a particular key. The mechanics of how the routing table works, and how the table is updated as nodes join and leave the network is a key differentiator between **different DHT algorithms**.
- In general, DHT nodes will only store a portion of the routing information in the network. This means that when a node joins the network, routing information only needs to be disseminated to a small number of peers.
- **Because each node only contains a portion of the routing table, the process of finding or storing a key/value pair requires contacting multiple nodes.** The number of nodes that needs to be contacted is related to the amount of routing information each node stores. Common lookup complexity is $O(\log n)$ where n is the number of nodes in the network.

I want to find my key of 66

1. I query node 1 for the key of 66
1. Node 1 is not responsible for 66, but Node 58 is the closest so ask 58
3. Query node 58 for the key of 66
4. Node 58 is not responsible for the key 66, but node 63 is the closes, so ask 63
5. Query node 63 for key of 66
6. Node 66 has the key of 66, return the value

Routing Table

Node / ID	Keyspace Range	Routing Table Node ID:Range	Database
1	1-10	Node 22: 21-30 Node 58: 51-60	"5":"some value"
22	21-30	Node 36: 21-30 Node 63: 61-70	"23": "some value" "27": "some other value"
36	31-40	Node 41: 41-50 Node 77: 71-80	Nothing
41	41-50	Node 58: 51-60 Node 89: 81-90	"48": "some value"
58	51-60	Node 63: 61-70 Node 95: 91-100	"54": "some value" "56": "some value"
63	61-70	Node 77: 71-80 Node 1: 1-10	"66": "some value"
77	71-80	Node 89: 81-90 Node 22: 21-30	"71": "some value"
89	81-90	Node 95: 91-100 Node 36: 31-40	"89": "some value"
95	91-100	Node 1: 1-10 Node 41: 41-50	"97": "some value"

Thank you.

Text Analysis

Text Analysis

- Text analysis, also known as text mining, is the process of automatically classifying and extracting meaningful information from unstructured text. It involves detecting and interpreting trends and patterns to obtain relevant insights from data in just seconds.
- Let's say your team needs to analyze hundreds of online reviews to learn about the things that your clients like or dislike about your product. **Text analysis**, however, can help you automatically tag reviews by topic and classify each opinion as positive, negative, or neutral, saving your team valuable hours.
- Another term you may have heard is **text analytics**. While it's closely related to text analysis, text analytics uses data visualization tools to transform insights into quantitative data while text analysis obtains qualitative insights from unstructured text.

Text Analysis

- Text mining or text analytics is a discipline that combines language science and computer science with statistical and machine learning techniques.
- Text mining is used for analyzing texts and turning them into a more structured form. Then it takes this structured form and tries to derive insights from it.
- Example: When analyzing crime from police reports, for example, text mining helps you recognize persons, places, and types of crimes from the reports. Then this new structure is used to gain insight into the evolution of crimes

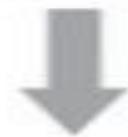
Text Analysis Example

Police reports of 10 June 2015

Danny W stole a watch in Chelsea Market

A person punched me, Xi Li, in the face in Orlando

During the Chelsea soccer game, my car was stolen



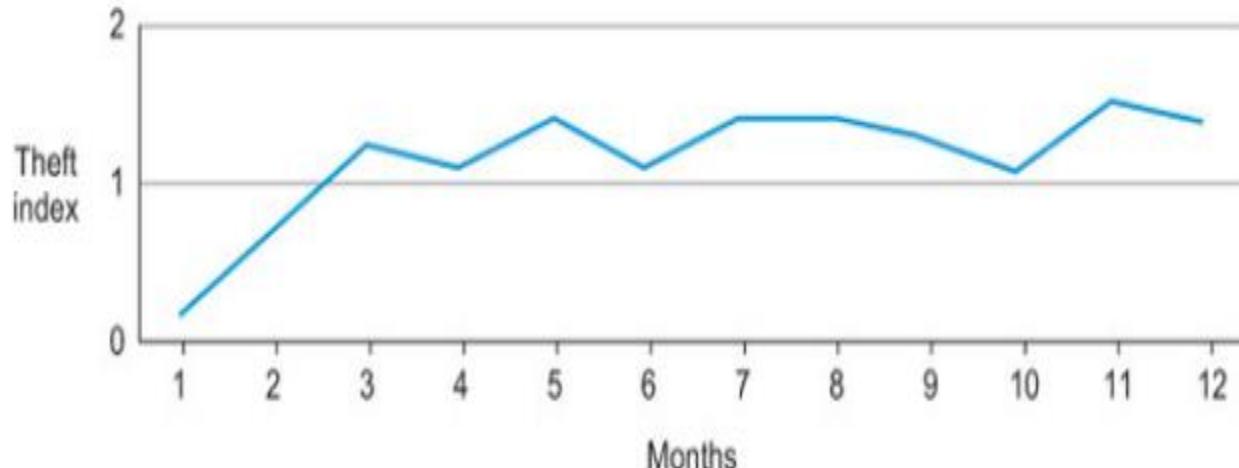
Add structure

Person	Place	Crime	Victim	Date
Danny W	Chelsea Market, New York	Theft	Unknown	10th June 2015
Unknown	Orlando	Violence	Xi Li	10th June 2015
Unknown	Chelsea, London	Theft	Bart Smith	10th June 2015



Analyze and visualize

Evolution of the theft in Chelsea Market



Text Pre-processing using NLP

Text Mining in the real world

- In your day-to-day life you've already come across text mining and natural language applications. Autocomplete and spelling correctors are constantly analyzing the text you type before sending an email or text message. When Facebook autocompletes your status with the name of a friend, it does this with the help of a technique called **named entity recognition**, although this would be only one component of their repertoire. The goal isn't only to detect that you're typing a noun, but also to guess you're referring to a person and recognize who it might be.
- Another example of named entity recognition is shown in figure 8.2. Google knows Chelsea is a football club but responds differently when asked for a person.

Text Mining in the real world

Figure 8.2. The different answers to the queries “Who is Chelsea?” and “What is Chelsea?” imply that Google uses text mining techniques to answer these queries.

The screenshot shows a Google search results page for the query "what is chelsea". The search bar at the top contains the query. Below it, the "Web" tab is selected, followed by "Images", "Maps", "News", "Videos", "More", and "Search tools". A search count of "About 202 000 000 results (0.48 seconds)" is displayed. The first result is a link to the Wikipedia page for Chelsea F.C., which includes a snippet about the club's history and seasons. The second result is a link to the Wikipedia page for Chelsea, London, with a snippet about the area's location. To the right of the search results, there is a summary box for "Chelsea F.C." featuring the club's logo, its name, its status as a football club, a brief description, and a link to the Wikipedia page. Below this summary, the manager's name, José Mourinho, is mentioned.

what is chelsea

Web Images Maps News Videos More Search tools

About 202 000 000 results (0.48 seconds)

[Chelsea F.C. - Wikipedia, the free encyclopedia](#)
en.wikipedia.org/wiki/Chelsea_F.C. ▾
Chelsea Football Club / tʃɛlə / are a professional football club based in Fulham, London, who play in the Premier League, the highest level of English football. Founded in 1905, the club have spent most of their history in the top tier of English football.
2014–15 Chelsea FC season - Roman Abramovich - Stamford Bridge - Eden Hazard

[Chelsea, London - Wikipedia, the free encyclopedia](#)
en.wikipedia.org/wiki/Chelsea_London ▾
Chelsea is an affluent area in central London, bounded to the south by the River Thames. Its frontage runs from Chelsea Bridge along the Chelsea Embankment.

Chelsea F.C.
Football club
Chelsea Football Club are a professional football club based in Fulham, London, who play in the Premier League, the highest level of English football. Founded in 1905, the club have spent most of their history in the top tier of English football. [Wikipedia](#)

Manager: José Mourinho

Text Mining in the real world

- Google uses many types of text mining when presenting you with the results of a query. What pops up in your own mind when someone says “Chelsea”? Chelsea could be many things: a person; a soccer club; a neighborhood in Manhattan, New York or London; a food market; a flower show; and so on. Google knows this and returns different answers to the question “Who is Chelsea?” versus “What is Chelsea?” To provide the most relevant answer, Google must do (among other things) all of the following:
- Preprocess all the documents it collects for named entities
- Perform language identification
- Detect what type of entity you’re referring to
- Match a query to a result
- Detect the type of content to return (PDF, adult-sensitive)

Text Mining in the real world

- Google uses text mining for much more than answering queries like shielding its Gmail users from spam, it also divides the emails into different categories such as social, updates, and forums.
- Text mining has many applications, including, but not limited to, the following:
 - Entity identification
 - Plagiarism detection
 - Topic identification
 - Text clustering
 - Translation
 - Automatic text summarization
 - Fraud detection
 - Spam filtering
 - Sentiment analysis

Text Mining in the real world

- When looking at the examples, we might have gotten the impression that text mining is easy. Sadly, no. In reality text mining is a complicated task and even many seemingly simple things can't be done satisfactorily.
- **Ambiguity** is one of the most important problem that we face in a text mining application.
- Another problem is **spelling mistakes** and different (correct) spelling forms of a word. Take the following three references to New York: “NY,” “Neww York,” and “New York.” For a human, it’s easy to see they all refer to the city of New York. Because of the way our brain interprets text, understanding text with spelling mistakes comes naturally to us; people may not even notice them. But for a computer these are unrelated strings unless we use algorithms to tell it that they’re referring to the same entity.
- Related problems are synonyms and the use of pronouns. **Try assigning the right person to the pronoun “she” in the next sentences:** “John gave flowers to Marleen’s parents when he met her parents for the first time. She was so happy with this gesture.” Easy enough, right? Not for a computer.

Example of a text data

- Emma knocked on the door. No answer. She knocked again and waited. There was a large maple tree next to the house. Emma looked up the tree and saw a giant raven perched at the tree top. Under the afternoon sun, the raven gleamed magnificently. Its beak was hard and pointed, its claws sharp and strong. It looked regal and imposing. It reigned the tree it stood on. The raven was looking straight at Emma with its beady black eyes. Emma felt slightly intimidated. She took a step back from the door and tentatively said, “Hello?”
- The paragraph contains a **lot of information**. Which parts of this paragraph of information are **salient (important) features**?
- So, we can apply **feature processing** on this piece of text

Feature

- **Feature** is numeric representation of **data**.
- Raw data can be changed to numeric measurements by many ways.
- Feature engineering is process of formulating the **most appropriate features given the data**.
- If there are **not enough informative features**, then the model will be **unable to perform the ultimate task**.
- If there are **too many features**, or if most of them are **irrelevant**, then the model will be **more expensive** and tricky to **train**.
- Features and models sit between raw data and the desired insights.

Feature

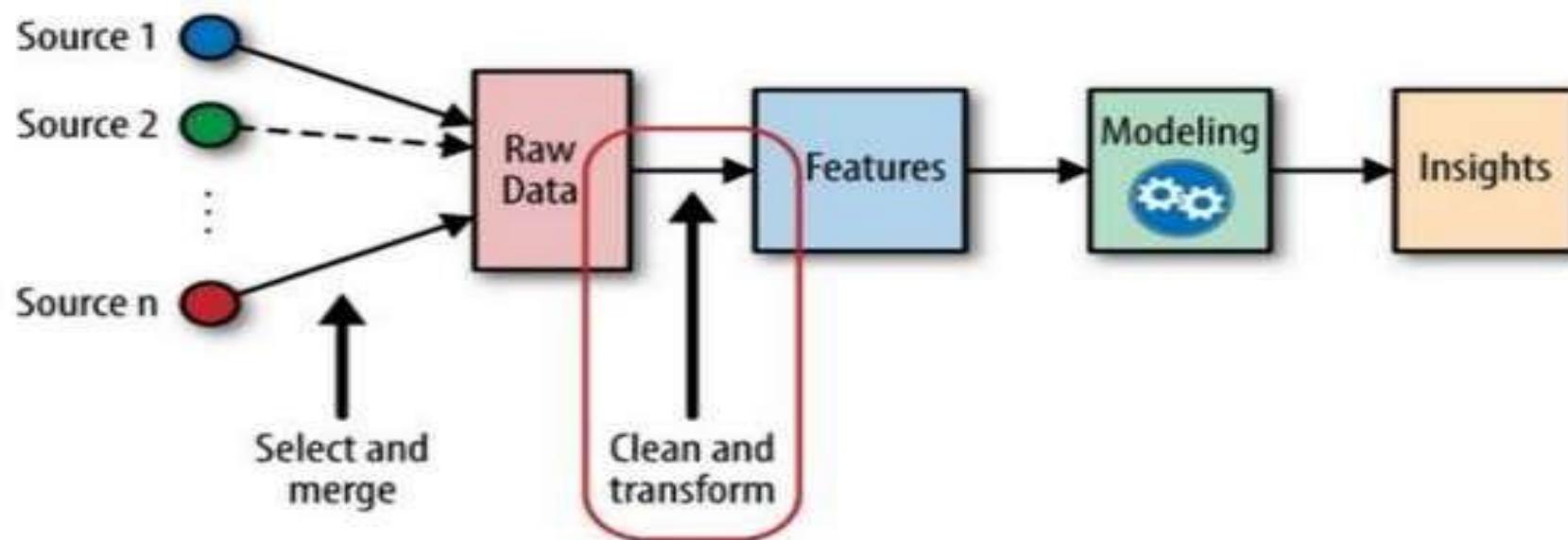


Fig. The place of feature engineering in the machine learning workflow

Feature

- In a machine learning workflow, both **model and features** are to be picked.
- It can be considered as a **double-jointed lever**, and the choice of one affects the other.
- **Good features** make the subsequent modeling step **easy** and the resulting model **more capable of completing the desired task**.
- Good features should not only represent salient aspects of the data, but also conform to the assumptions of the model.
- **Bad features** may require a **much more complicated** model to achieve the same level of performance.
- For selecting good features, feature engineering is required.

Feature Selection

- Feature selection techniques prune away non-useful features in order to reduce the complexity of the resulting model.
- The goal is to make a model that is quicker to compute, with little or no degradation in predictive accuracy.
- **Objectives of Feature Selection**
 - It eliminates irrelevant and noisy features by keeping the ones with minimum redundancy and maximum relevance to the target variable.
 - It reduces the computational time and complexity of training and testing a classifier, so it results in more cost-effective models.
 - It improves learning algorithms' performance, avoids overfitting, and helps to create better general models.

Feature Selection Methods

- There are three categories of feature selection methods, depending on how they interact with the classifier, namely, **filter**, **wrapper**, and **embedded** methods.
- **Filter methods**
 - Filtering techniques pre-process features to remove the features that are unlikely to be useful for the model.
 - For example, The correlation or mutual information between each feature and the response variable can be computed, and filter out the features that fall below a threshold.
 - Filtering techniques are much cheaper than the wrapper
 - They may not be able to select the right features for the model.

Feature Selection Methods

- **Wrapper methods** evaluate multiple models using procedures that add and/or remove predictors to find the optimal combination that maximizes model performance.
 - These procedures are normally built after the concept of Greedy Search technique (or algorithm).
 - A greedy algorithm is any algorithm that follows the problem-solving heuristic of making the locally optimal choice at each stage.
 - Generally, three directions of procedures are possible:
 - **Forward selection** — starts with one predictor and adds more iteratively. At each subsequent iteration, the best of the remaining original predictors are added based on performance criteria.

Feature Selection Methods

- **Backward elimination** — starts with all predictors and eliminates one-by-one iteratively. One of the most popular algorithms is Recursive Feature Elimination (RFE) which eliminates less important predictors based on feature importance ranking.
- **Step-wise selection** — bi-directional, based on a combination of forward selection and backward elimination. It is considered less greedy than the previous two procedures since it does reconsider adding predictors back into the model that has been removed (and vice versa). Nonetheless, the considerations are still made based on local optimisation at any given iteration.

Feature Selection Methods

- **Embedded methods** bridge the gap between filters and wrappers.
 - To begin with, they fuse measurable and statistical criteria like a filter to choose some features, and then using a machine learning algorithm, they pick the subset with the best classification performance.
 - They reduce the computational complexity of wrappers without re-classifying the subsets in each iteration and can model feature dependencies. They do not perform iterations.
 - Feature selection is performed in the learning phase, meaning that these methods achieve both model fitting and feature selection at the same time.
 - One disadvantage is their dependency on the classifier.

Natural Language Processing (NLP)

- It is an area of computer science and artificial intelligence that is concerned with the **interaction between computers and humans in natural language**.
- NLP is all about enabling**computers to understand and to generate human language**.
- Applications of NLP techniques are
 - **Voice Assistants like Siri**
 - **Speech recognition,**
 - **Sentiment analysis,**
 - **Automatic text summarization,**
 - **Machine translation**

Natural Language Processing (NLP)

Information Retrieval

Doc A



Doc 1

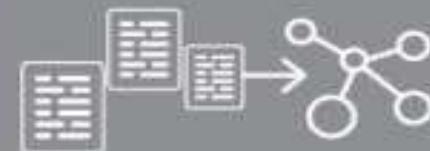
Doc 2

Doc 3

Sentiment Analysis



Information Extraction



Machine Translation



Natural
Language
Processing

Question Answering



Human: When was Apollo sent to space?

Machine: First flight – AS-201, February 26, 1966

Text mining techniques

- Tackle the problem of text classification: automatically classifying uncategorized texts into specific categories. To get from raw textual data to our final destination we'll need a few data mining techniques that require background information for us to use them effectively. The first important concept in text mining is the “bag of words.”
- **Bag of words**
 - Bag of words is the simplest way of structuring textual data: every document is turned into a word vector. If a certain word is present in the vector it's labelled “True”; the others are labelled “False”.

Bag of words

- Figure shows a simplified example of this, in case there are only two documents. The two word vectors together form the *document-term matrix*. The document-term matrix holds a column for every term and a row for every document. The values are yours to decide upon. In this chapter we'll use binary: term is present? True or False.

Figure 8.7. A text is transformed into a bag of words by labeling each word (term) with “True” if it is present in the document and “False” if not.

Game of Thrones is a great television series but the books are better.

Doing data science is more fun than watching television.

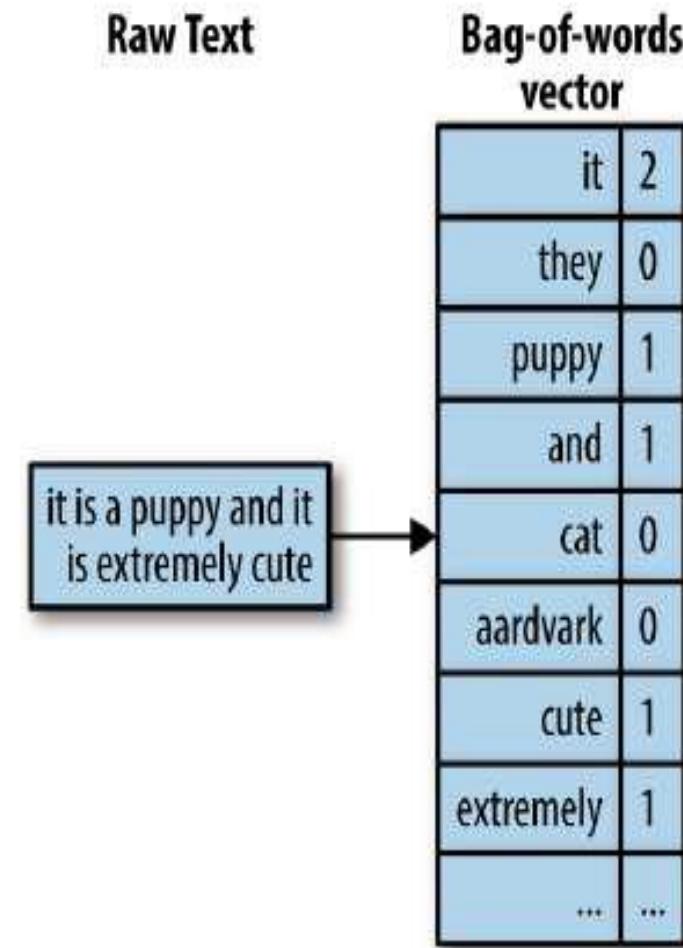
```
[({'game':True,'of':True,'thrones':True,'is':True,'a':True,
'great':True,'television':True,'series':True,'but':True,
'the':True,'books':True,'are':True,'better':True,'doing':
False,'data':False,'science':False,'more':False,'fun':False,
'than':False,'watching':False},
{'gameofthrones'},
({'doing':True,'data':True,'science':True,'is':True,'more':
True,'fun':True,'than':True,'watching':True,'television':True,
'game':False,'of':False,'thrones':False,'a':False,'great':
False,'series':False,'but':False,'the':False,'books':False,
'are':False,'better':False},
{'datascience'})]
```

Bag-of-words

- A bag-of-words is a representation of text that describes the **occurrence of words within a document**.
- It involves two things:
 - A **vocabulary** of known **words**.
 - A **measure of the presence of known words**.
- In **bag-of-words (BoW) featurization**, a text document is converted **into a vector of counts**. (A vector is just a collection of n numbers.)
- *The vector contains an entry for every possible word in the vocabulary.*
- If the word—say, “is”—appears **three times** in the document, then the feature **vector has a count of 3** in the position corresponding to that word. If a word in the **vocabulary doesn’t appear in the document**, then it gets a **count of 0**.

Bag of words

- Example



- Fig. Turning raw text into a bag-of-words representation

Bag of words

- Example: **Data:** “It was the best of times, it was the worst of times, it was the age of wisdom, it was the age of foolishness”,

Design the Vocabulary

- The unique words here (ignoring case and punctuation) are:

- “it”
- “was”
- “the”
- “best”
- “of”
- “times”
- “worst”
- “age”
- “wisdom”
- “foolishness”

Bag-of-Words

- **Create Document Vectors**
 - The objective is to turn each document of free text into a vector that we can use as input or output for a machine learning model.
 - **Vector for first line:** “*It was the best of times*”
 - “it” = 1
 - “was” = 1
 - “the” = 1
 - “best” = 1
 - “of” = 1
 - “times” = 1
 - “worst” = 0
 - “age” = 0
 - “wisdom” = 0
 - “foolishness” = 0

Bag-of-Words

- EG:
 - Similarly for other lines:
 - "it was the worst of times" = [1, 1, 1, 0, 1, 1, 1, 0, 0, 0]
 - "it was the age of wisdom" = [1, 1, 1, 0, 1, 0, 0, 1, 1, 0]
 - "it was the age of foolishness" = [1, 1, 1, 0, 1, 0, 0, 1, 0, 1]
 - These docs can also be represented as **document-term** matrix:

	it	was	the	best	of	times	worst	age	wisdom	foolishness
Doc 1	1	1	1	0	1	1	1	0	0	0
Doc 2	1	1	1	0	1	0	0	1	1	0
Doc 3	1	1	1	0	1	0	0	1	0	1

Table 1 : An example document-term matrix

Bag-of-words

- Bag-of-words converts a text document into a flat vector.
- It is “flat” because it doesn’t contain any of the original textual structures.
- The original text is a sequence of words.
- But a bag-of-words has no sequence; it just remembers how many times each word appears in the text.

Bag-of-words

- **Drawbacks:**
 - Bag-of-words **is not perfect.**
 - Breaking down a sentence into single words can **destroy the semantic meaning.**
 - For instance, “not bad” semantically means “decent” or “good”
 - But “not” and “bad” constitute a **negation** plus a **negative sentiment.**
 - “**toy dog**” and “**dog toy**” could be very different things and the meaning is lost with the singleton words “toy” and “dog.”

Bag-of-n-Grams

- **Bag-of-n-Grams** is a extension of bag-of-words.
- Each word or token is called a “gram”.
- An **n-gram is a sequence of n tokens**.
- A word is a **1-gram**, also known as a **unigram**.
- An N-gram is an **N-token sequence of words**:
 - A 2-gram (bigram) is a two-word sequence of words like “please turn”, “turn your”, or “your homework”,
 - A 3-gram (trigram) is a three-word sequence of words like “please turn your”, or “turn your homework”
- After tokenization, the counting mechanism can **collate individual tokens into word counts**, or **count overlapping sequences as n-grams**.
- For example, the sentence “Emma knocked on the door” generates the n-grams “Emma knocked,” “knocked on,” “on the,” and “the door.”

Bag-of-n-Grams

- n-grams **retain more of the original sequence structure** of the text,
- Hence, the **bag-of-n-grams representation can be more informative.**
- **However, n-grams are more expensive to compute, store, and model.**
- **The larger n is, the richer the information, and the greater the cost.**

Frequency based Filtering

- the **occurrence of words** in example documents needs to be **scored** after vocabulary has been made
- **Counts:** Count the number of times each word appears in a document.
- **Frequencies:** Calculate the frequency that each word appears in a document out of all the words in the document.
- **Rare words: need to filter out rare words.** To a statistical model, a word that appears in only one or two documents is more like noise than useful information.
 - Rare words can be easily identified and trimmed based on word count statistics.
 - **Their counts can be aggregated into a special garbage bin, which can serve as an additional feature**

Frequency based Filtering

- Frequency based Filtering
 - Rare words:

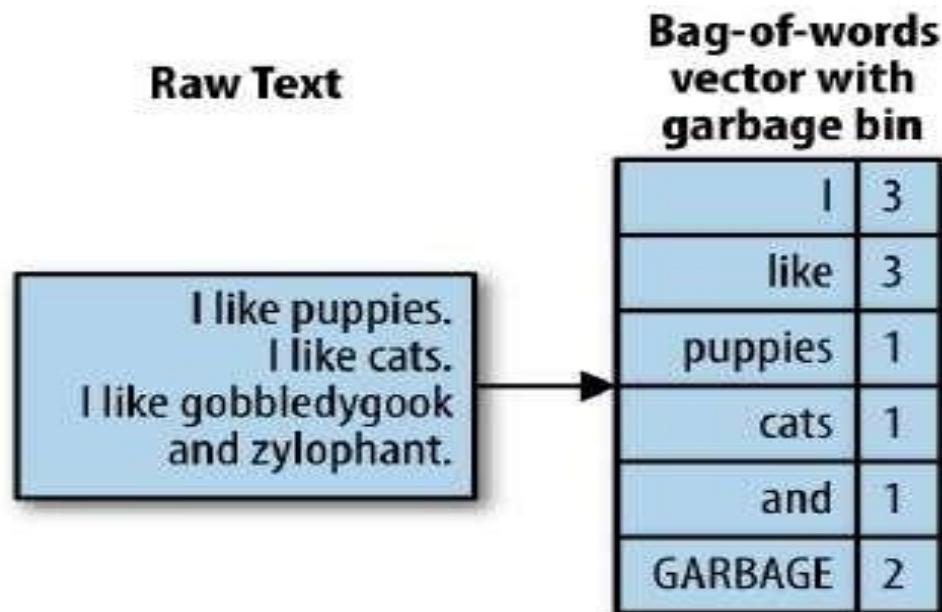


Fig. Bag-of-words feature vector with a garbage bin

Tf-Idf

- Tf-idf is a simple twist on the bag-of-words approach. It stands for **term frequency– inverse document frequency**.
- Bag of Words just creates a set of vectors containing the **count of word occurrences** in the document , while the TF-IDF model contains information on the **more important words and the less important ones** as well.
- Bag of Words vectors are easy to interpret. However, TF-IDF usually performs better in machine learning models. Hence TF-IDF is preferred.
- TF-IDF is a statistical measure **used to evaluate how important a word is to a document** in a collection or corpus.
- The tf-idf weight is composed by two terms:
 - **TF: Term Frequency**
 - **IDF: Inverse Document Frequency,**

Tf-Idf

– TF: Term Frequency

- How frequently a term occurs in a document.
- $TF(t) = (\text{Number of times term } t \text{ appears in a document}) / (\text{Total number of terms in the document})$.

– IDF: Inverse Document Frequency

- measures how important a term is
- While computing TF, all terms are considered equally important.
- However it is known that certain terms, such as "is", "of", and "that", may appear a lot of times but have little importance.
- Thus we need to weigh down the frequent terms while scale up the rare ones, by computing the following:
- $IDF(t) = \log(\text{Total number of documents} / \text{Number of documents with term } t \text{ in it})$.

Tf-Idf

- Hence, **Tf-Idf = tf*idf**
- Eg:
 - Consider a document containing 100 words where in the word *cat* appears 3 times.
 - **tf for *cat* = $(3 / 100) = 0.03$.**
 - Now, assume we have **10 million documents** and the word *cat* **appears in one thousand of these.**
 - **idf = $\log(10,000,000 / 1,000) = 4$.**
 - **Tf-idf :** $0.03 * 4 = 0.12$.

Tf-Idf

- Common words like ‘is’, ‘the’, ‘a’ etc. tend to appear quite frequently in comparison to the words which are important to a document.
- For example, a document on Lionel Messi is going to contain more occurrences of the word “Messi” in comparison to other documents.
- But common words like “the” etc. are also going to be present in higher frequency in almost every document
- TF-IDF works by penalising these common words by assigning them lower weights while giving importance to words like Messi in a particular document.

Tf-Idf

Document 1

Term	Count
This	1
is	1
about	2
Messi	4

Document 2

Term	Count
This	1
is	2
about	1
Tf-idf	1

Compute TF-IDF for **THIS**:

TF = (Number of times term t appears in a document)/(Number of terms in the document)

So, TF(This, Document1) = 1/8

TF(This, Document2)=1/5

TF denotes the contribution of the word to the document i.e words relevant to the document should be frequent.

eg: A document about Messi should contain the word 'Messi' in large number.

Tf-Idf

- $\text{IDF} = \log(N/n)$,
 - where, **N is the number of documents**
 - **n is the number of documents a term t has appeared in.**
- So, $\text{IDF}(\text{This}) = \log(2/2) = 0$.
- Why IDF?
- Ideally, if a word has appeared in all the documents, then probably that word is not relevant to a particular document.
- But if word has appeared in a subset of documents then probably the word is of some relevance to the documents it is present in.

Tf-Idf

Document 1

Term	Count
This	1
is	1
about	2
Messi	4

Document 2

Term	Count
This	1
is	2
about	1
Tf-idf	1

- Compute IDF for the word ‘Messi’:

$$\text{IDF} = \log(N/n),$$

where, **N** is the number of documents

n is the number of documents a term **t** has appeared in.

$$\text{IDF}(\text{Messi}) = \log(2/1) = 0.301.$$

Tf-Idf

Document 1

Term	Count
This	1
is	1
about	2
Messi	4

Document 2

Term	Count
This	1
is	2
about	1
Tf-idf	1

- Compare the TF-IDF for a common word ‘This’ and a word ‘Messi’ which seems to be of relevance to Document 1.
- $\text{TF-IDF}(\text{This}, \text{Document1}) = \text{TF} * \text{IDF} = (1/8) * (0) = 0$
- $\text{TF-IDF}(\text{Messi}, \text{Document1}) = (4/8) * 0.301 = 0.15$
- TF-IDF method heavily penalizes the word ‘This’ but assigns greater weight to ‘Messi’.
- So, this may be understood as ‘Messi’ is an important word for Document1 from the context of the entire corpus.

Tokenization

- Tokenization is the process breaking complex data like paragraphs into simple units called tokens.
- **Sentence tokenization** : split a paragraph into list of sentences using `sent_tokenize()` method
- **Word tokenization** : split a sentence into list of words using `word_tokenize()` method
- Some other important terms related to word Tokenization are:
- **Bigrams**: Tokens consist of two consecutive words known as bigrams.
- **Trigrams**: Tokens consist of three consecutive words known as trigrams.
- **Ngrams**: Tokens consist of 'N' number of consecutive words known as ngrams

Tokenization

- These tokens are the most basic unit of information you'll use for your model. The terms are often words but this isn't a necessity. Entire sentences can be used for analysis.
- We'll use unigrams: terms consisting of one word. Often, however, it's useful to include bigrams (two words per token) or trigrams (three words per token) to capture extra meaning and increase the performance of your models. This does come at a cost, though, because you're building bigger term-vectors by including bigrams and/or trigrams in the equation.

Processing Text Features

- **Text cleaning techniques :**
 - Ignoring case
 - Ignoring punctuation
 - Ignoring frequent words that don't contain much information, called **stop words**, like “a,” “of,” etc.
 - Reducing words **to their stem** (e.g. “play” from “playing”).

Processing Text Features

- **Stopwords:**
 - **Stopwords** refers to the most common words in a language (such as “the”, “a”, “an”, “in”) which helps in formation of sentence to make sense, but these words does not provide any significance in language processing so remove it .
 - In the sentence “**Emma knocked on the door**,” the words “**on**” and “**the**” don’t change the fact that this sentence is about a person and a door
 - For **coarse-grained tasks** such as classification, the pronouns, articles, and prepositions may not add much value.

Stemming

- Stemming is a normalization technique where list of tokenized words are converted into shorten root words to remove redundancy. Stemming is the process of reducing inflected (or sometimes derived) words to their word stem, base or root form.
- A computer program that stems word may be called a stemmer.
- A stemmer reduce the words like *fishing*, *fished*, and *fisher* to the stem *fish*. The stem need not be a word, for example the Porter algorithm reduces, *argue*, *argued*, *argues*, *arguing*, and *argus* to the stem *argu* .

Stemming

- Over-stemming: Over-stemming is when two words with different stems are stemmed to the same root. This is also known as a false positive.
- **Example: universal, university, universe**
- All the above 3 words are stemmed to **univers** which is wrong behavior. Though these three words are etymologically related, their modern meanings are in widely different domains, so treating them as synonyms in NLP/NLU will likely reduce the relevance of the search results
- Under-stemming is when two words that should be stemmed to the same root are not. This is also known as a false negative. Below is the example for the same.
- **Example: alumnus, alumni, alumnae**

Lemmatization

- Major drawback of stemming is it produces **Intermediate representation** of word. **Stemmer may or may not return meaningful word.**
- To overcome this problem Lemmatization comes into picture.
- Stemming algorithm works by cutting suffix or prefix from the word. On the contrary Lemmatization consider morphological analysis of the words and **returns meaningful word** in proper form.
- Hence, **lemmatization is preferred.**

Processing Text Features

- **Parsing and Tokenization**
 - how does a computer know what a word is?
 - **Parsing is necessary when the string contains more than plain text**
 - **For eg, if the raw data is a web page, an email, or a log of some sort, then it contains additional structure.**
 - If the document is a **web page, then the parser needs to handle URLs.**
 - If it is an **email, then fields like From, To, and Subject may require special handling—otherwise these headers will end up as normal words in the final count, which may not be useful.**

Processing Text Features

- **Parsing and Tokenization**
 - After parsing, **the plain-text portion of the document can go through tokenization.**
 - This turns **the string—a sequence of characters—into** a sequence of tokens.
 - **Each token can then be counted as a word.**
 - **The tokenizer needs to know what characters indicate that one token has ended and another is beginning.**
 - **Space characters are usually good separators, as are punctuation characters.**
 - If the text contains **tweets**, then hash marks (#) **should not be used as separators (*delimiters*)**

Processing Text Features

- Parsing and Tokenization
 - Complex text featurization methods like **word2vec** also work with sentences or paragraphs
 - Here, first document are parsed into sentences, then further tokenize each sentence into words

POS (Parts of Speech)

- POS (Parts of Speech) tell us about grammatical information of words of the sentence by assigning specific token (Determiner, noun, adjective , adverb ,verb,Personal Pronoun etc.) as tag (DT,NN ,JJ,RB,VB,PRP etc) to each words.
- Word can have more than one POS depending upon context where it is used. we can use POS tags as statistical NLP tasks it distinguishes sense of word which is very helpful in text realization and infer semantic information from gives text for sentiment analysis.

POS (Parts of Speech)

```
import nltk
from nltk.tokenize import word_tokenize
data="The pink sweater fit her perfectly"
words=word_tokenize(data)
for word in words:
    print(nltk.pos_tag([word]))
```

Output:

```
[('The', 'DT')]
[('pink', 'NN')]
[('sweater', 'NN')]
[('fit', 'NN')]
[('her', 'PRP$')]
[('perfectly', 'RB')]
```

Named Entity Recognition

- The process of recognizing named entity such as person name, location name , organization name , designation of person ,quantities or values.

```
import nltk
from nltk.tokenize import word_tokenize
from nltk import ne_chunk
sentence="The US president stays in WHITE HOUSE"
sent_tokens=word_tokenize(sentence)
tags=nltk.pos_tag(sent_tokens)
NER=ne_chunk(tags)
print(NER)
```

output:

```
(S
  The/DT
  (GSP US/NNP)
  president/NN
  stays/NNS
  in/IN
  (FACILITY WHITE/NNP HOUSE/NNP))
```

Processing Text Features

- Collocation Extraction for Phrase Detection
 - A collocation is an expression consisting of **two or more words that correspond to some conventional way of saying things.**
 - The words together can mean more than their sum of parts (*The Times of India, disk drive*)
 - **Techniques for finding the collocation:**
 - Frequency-based methods
 - Chunking and part-of-speech tagging

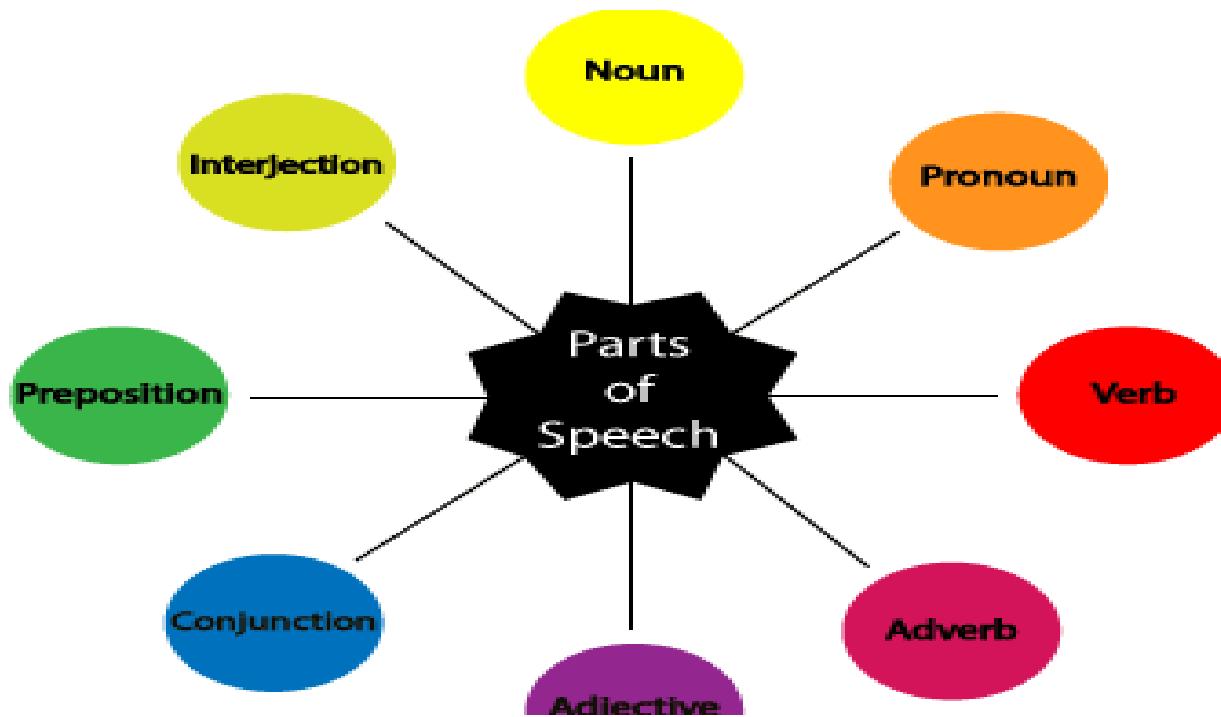
Processing Text Features

- Techniques for finding the **collocation**:
 - Frequency-based methods
 - Find collocations by **counting the number of occurrences**.
 - Need also to define a maximum size window
 - generate longer phrases, there are other methods such as chunking or combining with part-of-speech (PoS) tagging.

Count	Word 1	Word 2
11657	New	York
24	previous	games
46	minus	points
131	hundreds	dollars

Processing Text Features

- Techniques for finding the collocation
 - Chunking and part-of-speech tagging
 - PoS Tagging: The **part of speech** explains how a word is used in a sentence. There are eight main parts of speech



Chunking

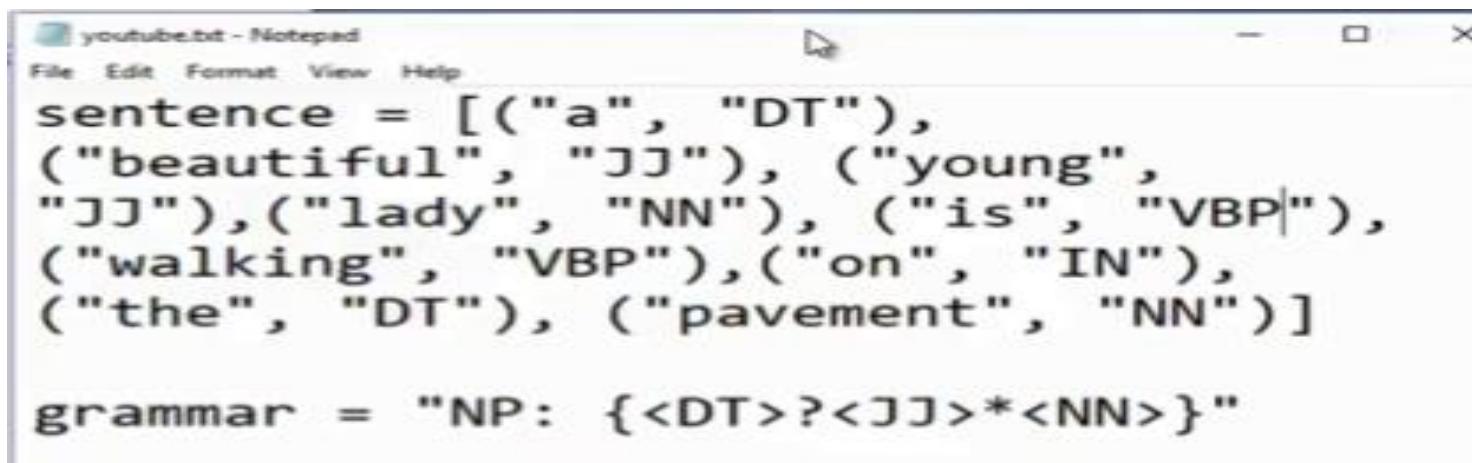
- The identification of parts of speech and short phrases (like noun phrases).
- Part of speech tagging tells you whether words are noun, verbs, adjective etc. but it does not give you any clue about the structure of the sentence or phrases in the sentence.
- Also like tokenization, the pieces produced by chunker do not overlap in source text.
- Used in Named Entity Recognition.
- Labeling of tokens present in raw text.

Chunking

- NLP Chunking can be applied to any area and on any level.
- **Chunking Up**
 - Moving to more general or abstract pieces of information / Zoom out / Overview of the situation.
- **Chunking Down**
 - Moving to more specific and detailed information / Zoom in / Getting details and distinction.
- **Noun phrase chunking:** chunks corresponding to Noun-phrases.
 - Example: **[John P.] saw [a beautiful woman] with a [telescope]**

Chunking

- Steps to produce noun phrase chunker through NLTK module.
 - Defining a **chunker grammar**
 - Creating a **chunk parser**
 - Output will be a **tree** which shows all chunks present in the sentence.



A screenshot of a Windows Notepad window titled "youtube.txt - Notepad". The window contains the following Python code:

```
youtube.txt - Notepad
File Edit Format View Help
sentence = [("a", "DT"),
("beautiful", "JJ"), ("young",
"JJ"), ("lady", "NN"), ("is", "VBP|"),
("walking", "VBP"), ("on", "IN"),
("the", "DT"), ("pavement", "NN")]
grammar = "NP: {<DT>?<JJ>*<NN>}"
```

Chunking

```
Python 3.4.4 (v3.4.4:737efcadf5a6, Dec 20 2015, 19:28:18) [MSC v.1600 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> import nltk
>>> sentence = [("a", "DT"), ("beautiful", "JJ"), ("young", "JJ"), ("lady", "NN"),
   ("is", "VBP"), ("walking", "VBP"), ("on", "IN"), ("the", "DT"), ("pavement", "NN")]
>>> grammar = "NP: ((DT)?<JJ>*<NN>)"
```

```
>>> parser=nltk.RegexpParser(grammar)
>>> output=parser.parse(sentence)
>>> output.draw()
```



Chunking

Python 3.4 Shell

File Edit Shell Debug Options Window Help

Python 3.4.4 (v3.4.4:737efcadf5a6, Dec 20 2015, 19:28:10) [MSC v.1600 32 bit (Intel)] on win32

Type "copyright", "credits" or "license()" for more information.

```
>>> import nltk  
>>> sentence = [("a", "DT"), ("beautiful", "JJ"), ("young", "JJ"), ("lady", "NN")  
, ("is", "VBP"), ("walking", "VBP"), ("on", "IN"), ("the", "DT"), ("pavement", "NN")]  
>>> grammar = "NP: ((DT)?(JJ)*NN)"  
>>> parser=nltk.RegexpParser(grammar)  
>>> output=parser.parse(sentence)  
>>> output.draw()
```

```
>>> grammar = "NP: ((DT)?(JJ)*NN)"  
>>> parser=nltk.RegexpParser(grammar)  
>>> output=parser.parse(sentence)  
>>> output.draw()
```



Chunking

- Chunking is the process of making a group of tokens into chunks. In simple words chunking is used as selecting the subsets of tokens. The parts of speech are combined with *regular expressions*.
- Chunking is used for entity detection. An entity is that part of the sentence by which machine get the value for any intention
- Chunking is used to categorize different tokens into the same chunk. The result will depend on grammar which has been selected. Further chunking is used to tag patterns and to explore text corpora.

Feature Scaling

- In many machine learning algorithms, **to bring all features in the same standing**, we need to do scaling so that one significant number doesn't impact the model just because of their large magnitude.
- The machine learning algorithm works on numbers and does not know what that number represents. A weight of 10 grams and a price of 10 dollars represents completely two different things — which is a no brainer for humans, but for a model as a feature, it treats both as same.
- The “Weight” cannot have a meaningful comparison with the “Price.” So the assumption algorithm makes that since “Weight” > “Price,” thus “Weight,” is more important than “Price.”

Feature Scaling

- So these more significant number starts playing a more decisive role while training the model. Thus feature scaling is needed **to bring every feature in the same footing without any upfront importance.**
- One more reason is **saturation**, like in the case of sigmoid activation in Neural Network, scaling would help not to saturate too fast.
- Feature scaling in machine learning is one of the most critical steps during the pre-processing of data before creating a machine learning model. Scaling can make a difference between a weak machine learning model and a better one

Feature Scaling

- The most common techniques of feature scaling are Normalization and Standardization.
- **Normalization** is used when we want to bound our values between two numbers, typically, between $[0,1]$ or $[-1,1]$. While **Standardization** transforms the data to have zero mean and a variance of 1, they make our data **unitless**.
- Algorithms that do not require normalization/scaling are the ones that **rely on rules**. They would not be affected by any monotonic transformations of the variables. Scaling is a monotonic transformation. Examples of algorithms in this category are all the tree-based algorithms
 - **CART, Random Forests, Gradient Boosted Decision Trees**. These algorithms utilize rules and **do not require normalization**.

Feature Scaling

- We must have always faced this question of whether to Normalize or to Standardize. While there is no obvious answer to this question, it really depends on the application
- Normalization is good to use when,
 - In Neural Networks algorithm that require data on a 0–1 scale, normalization is an essential pre-processing step. Another popular example of data normalization is image processing, where pixel intensities have to be normalized to fit within a certain range (i.e., 0 to 255 for the RGB color range).
- Standardization can be helpful in cases where the data follows a Gaussian distribution. Though this does not have to be necessarily true. Since standardization does not have a bounding range, so, even if there are outliers in the data, they will not be affected by standardization.

Feature Scaling

- There are some points which can be considered while deciding whether we need Standardization or Normalization
 - Standardization may be used when data represent Gaussian Distribution, while Normalization is great with Non-Gaussian Distribution
 - Impact of Outliers is very high in Normalization

Dimensionality Reduction

- As we can imagine, when we have real-world documents the number of the dimensions would increase dramatically like a few thousand because the **number of the dimensions equals to the number of unique terms** in the entire set of documents. We call this type of data ‘high-dimensionality’ data.
- Another thing you would notice by looking at the table is that there are many cells that are left empty because all of the terms don’t necessarily exist in all the documents. We call this type of data ‘**sparse**’ data or ‘**sparse matrix**’.
- Calculating the distance among all the documents from this type of high-dimensional and sparse data with algorithms like ‘K-means’ would have a problem called ‘[Curse of dimensionality](#)’.

Dimensionality Reduction

- Problems with Bag-of-words or N-gram
 - Curse of Dimensionality (too many features)
 - Example: Find sentiment of tweets (tweet has 280 characters or 50+ words). So when we are working with 10,000 tweets, then we have huge matrix of features.

Illinois is also known as Land of Lincoln. Abraham Lincoln, Statesman and Lawyer, served as the 16th President of the United States.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
Illinois	is	also	known	as	land	of	Lincoln	Abraham	statesman	and	lawyer	served	the	16th	president	United	States
1	1	1	1	2	1	2	2	1	1	1	1	1	2	1	1	1	1

Dimensionality Reduction

- Example: When we have only 4 sentences but when we plot them with bag-of-word then it transforms into matrix of 12 feature though we have max. 4 words in each sentence.

Sentences	John	likes	movies	Liza	watching	Frans	loves	playing	football	merry	enjoys	action
John likes movies	1	1	1	0	0	0	0	0	0	0	0	0
Liza likes watching movies	0	1	1	1	1	0	0	0	0	0	0	0
Frans loves playing football	0	0	0	0	0	1	1	1	1	0	0	0
Merry enjoys action movies	0	0	1	0	0	0	0	0	0	1	1	1

Dimensionality Reduction

- **Need:** As the number of features increase, the number of samples also increases proportionally. The more features we have, the more number of samples we will need to have all combinations of feature values well represented in our sample.
 - As the number of features increases, the model becomes more complex. The more the number of features, the more the chances of overfitting. resulting in poor performance on real data, beating the purpose.

Dimensionality Reduction

- Avoiding overfitting is a major motivation for performing dimensionality reduction. The fewer features our training data has, the lesser assumptions our model makes and the simpler it will be. But that is not all and dimensionality reduction has a lot more advantages to offer, like
 - Less misleading data means model accuracy improves.
 - Less dimensions mean less computing. Less data means that algorithms train faster.
 - Less data means less storage space required.
 - Less dimensions allow usage of algorithms unfit for a large number of dimensions
 - Removes redundant features and noise.

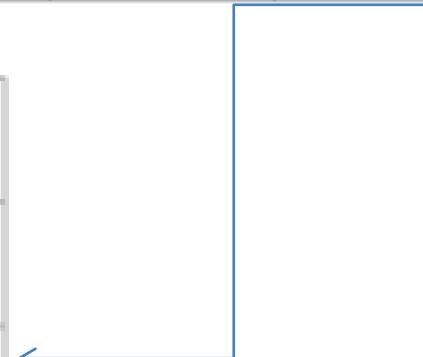
Dimensionality Reduction

- Can we compact this high dimensional and sparse data into a data that is represented by only a **few dimensions without losing** much of the original **information**? If we can, then maybe we can feed such reduced data into the clustering or any algorithm.
- Dimensionality reduction is, the process of reducing the dimension of your feature set. Your feature set could be a dataset with a hundred columns (i.e features) or it could be an array of points that make up a large sphere in the three-dimensional space.
- Dimensionality reduction is bringing the number of columns down to say, twenty or converting the sphere to a circle in the two-dimensional space.

Dimensionality Reduction

	a	an	apple	ate	banana	eat	i	today	will	yesterday
Doc 1		0.0811	0.0811	0.0811			0			0.0811
Doc 2		0.0676	0.0676			0.1831	0	0.1831	0.1831	
Doc 3	0.2197			0.0811	0.2197		0			0.0811

	1	2	3
Doc 1	0.0144	0.8164	0.5774
Doc 2	-0.7142	-0.3957	0.5774
Doc 3	0.6998	-0.4207	0.5774

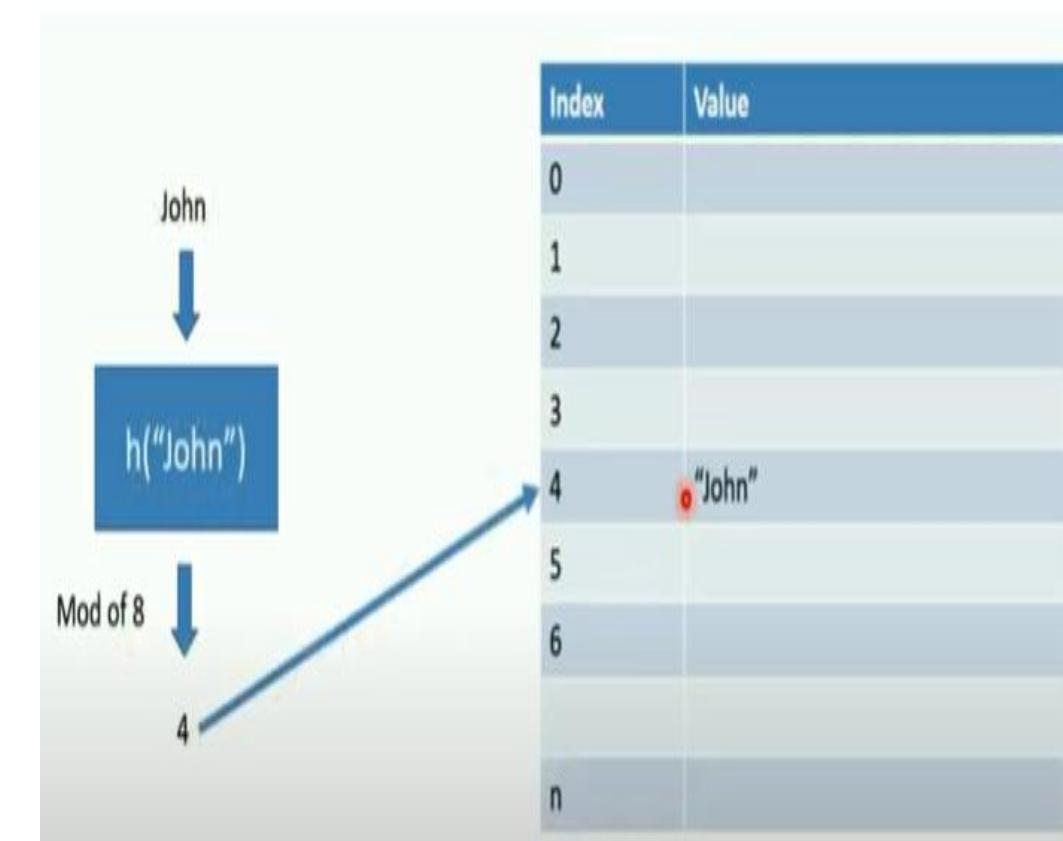
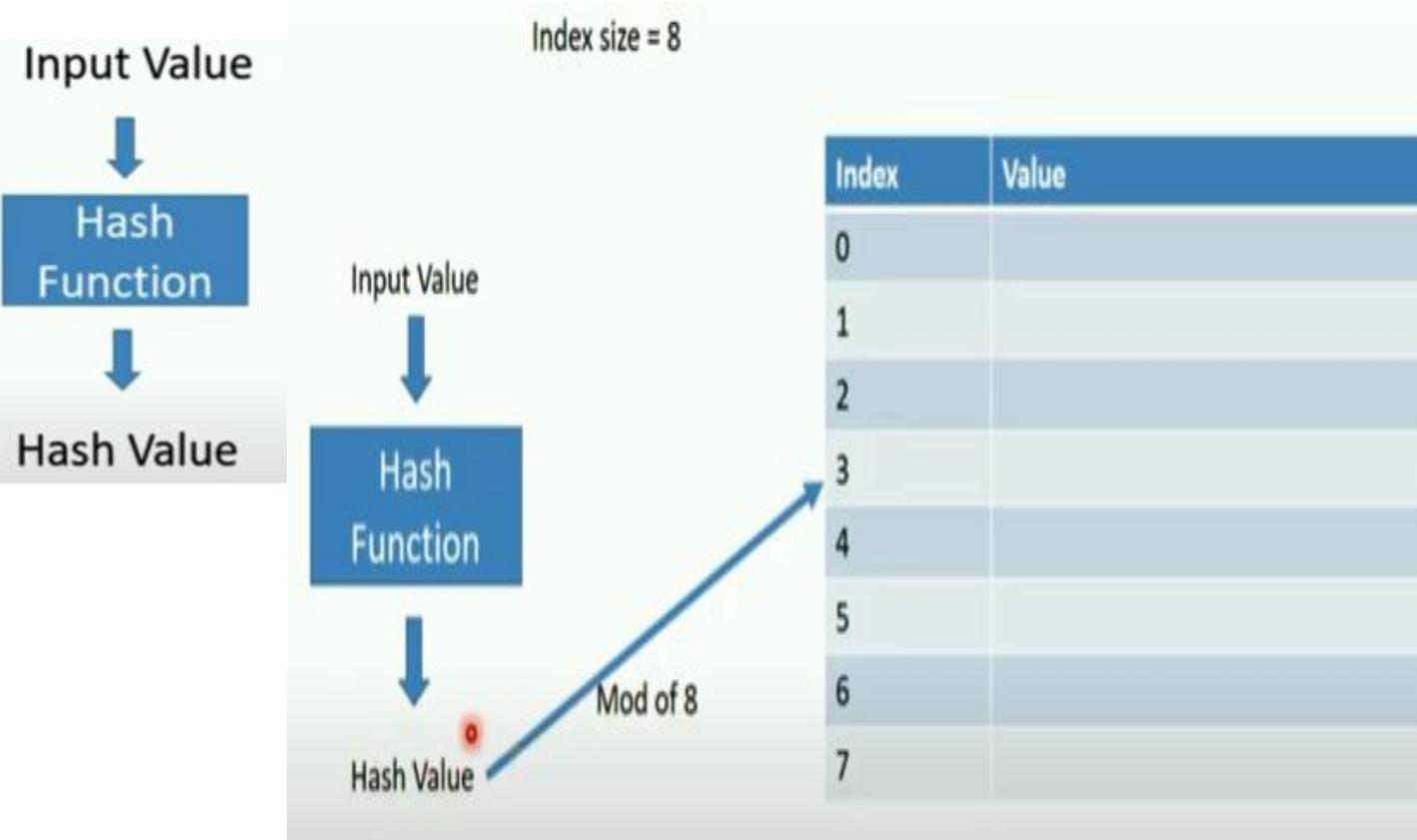


Dimensionality Reduction

- Dimensionality reduction could be done by both feature selection methods as well as feature engineering methods.
- Feature selection is the process of identifying and selecting relevant features for your sample. Feature engineering is manually generating new features from existing features, by applying some transformation or performing some operation on them.

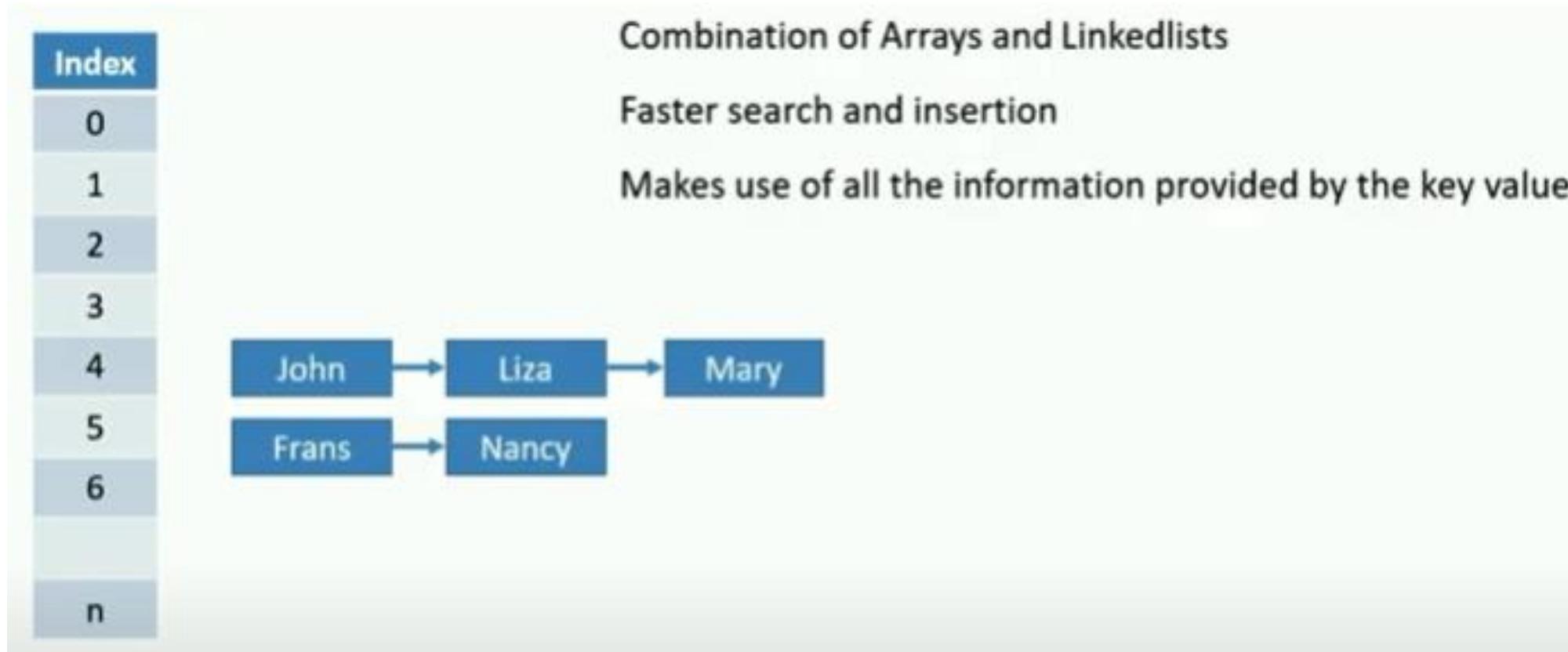
Feature Hashing

- Hashing: Hash is simply fixed length number or string that is generated by the hashing algorithm because we are trying to generate indices of table or array; we use hashing algorithms.



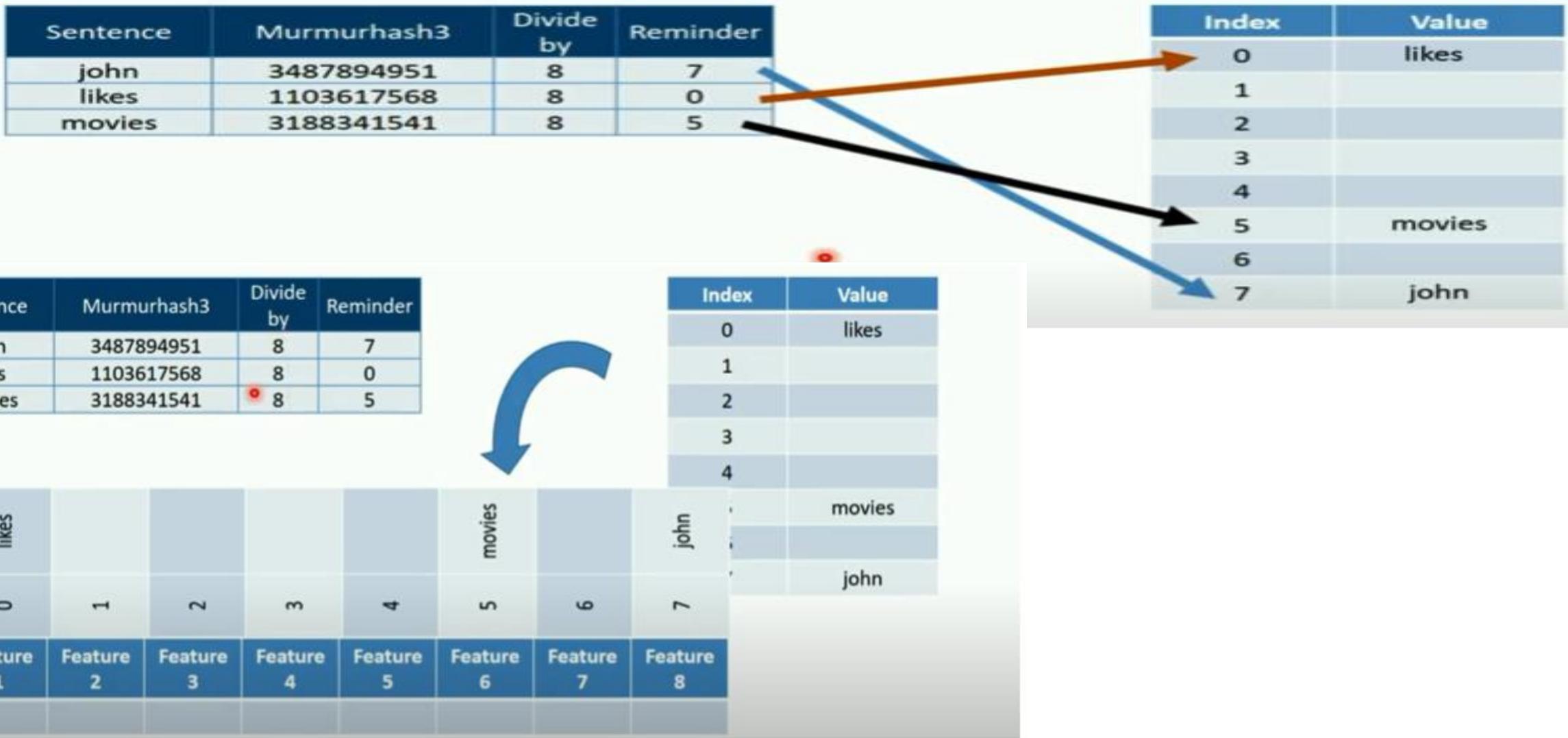
Feature Hashing

- Hash table with separate chaining



Feature Hashing

- No. of extracted features = mod of number



Comparing bag-of-word with feature Hashing Model

- No. of extracted features = mod of number

Bag Of Words

	John	likes	movies	Liza	watching	Frans	loves	playing	football	merry	enjoys	action
John likes movies	1	1	1	0	0	0	0	0	0	0	0	0
Liza likes watching movies	0	1	1	1	1	0	0	0	0	0	0	0
Frans loves playing football	0	0	0	0	0	1	1	1	1	0	0	0
Merry enjoys action movies	0	0	1	0	0	0	0	0	0	1	1	1

Feature Hashing

	Feature 1	Feature 2	Feature 3	Feature 4	Feature 5	Feature 6	Feature 7	Feature 8
John likes movies	1	0	0	0	0	2	0	0
Liza likes watching movies	2	0	1	0	0	1	0	0
Frans loves playing football	0	1	1	0	1	0	0	1
Merry enjoys action movies	0	0	0	0	0	3	0	1

Feature Hashing

- Also known as **hashing trick** and Used for **vectorizing features**
- It works by applying a **hash function** to the features and using their **hash values as indices directly.**
- A **hash function** is a **deterministic** function that **maps** a **potentially unbounded integer** to a **finite integer range** $[1, m]$.

Feature Hashing

- Need
 - Many domains have **very high feature dimension**.
 - Some feature domains are **naturally dense** (for eg, **images and video**).
 - This creates a challenge at **scale** because even simple models can become very large, **Consume more memory and resources** during training and when making predictions in production systems.
 - Instead of maintaining a **one-to-one mapping of categorical feature values to locations in the feature vector**, we use a **hash function** to determine **the feature's location in a vector dimension**.

Feature Hashing

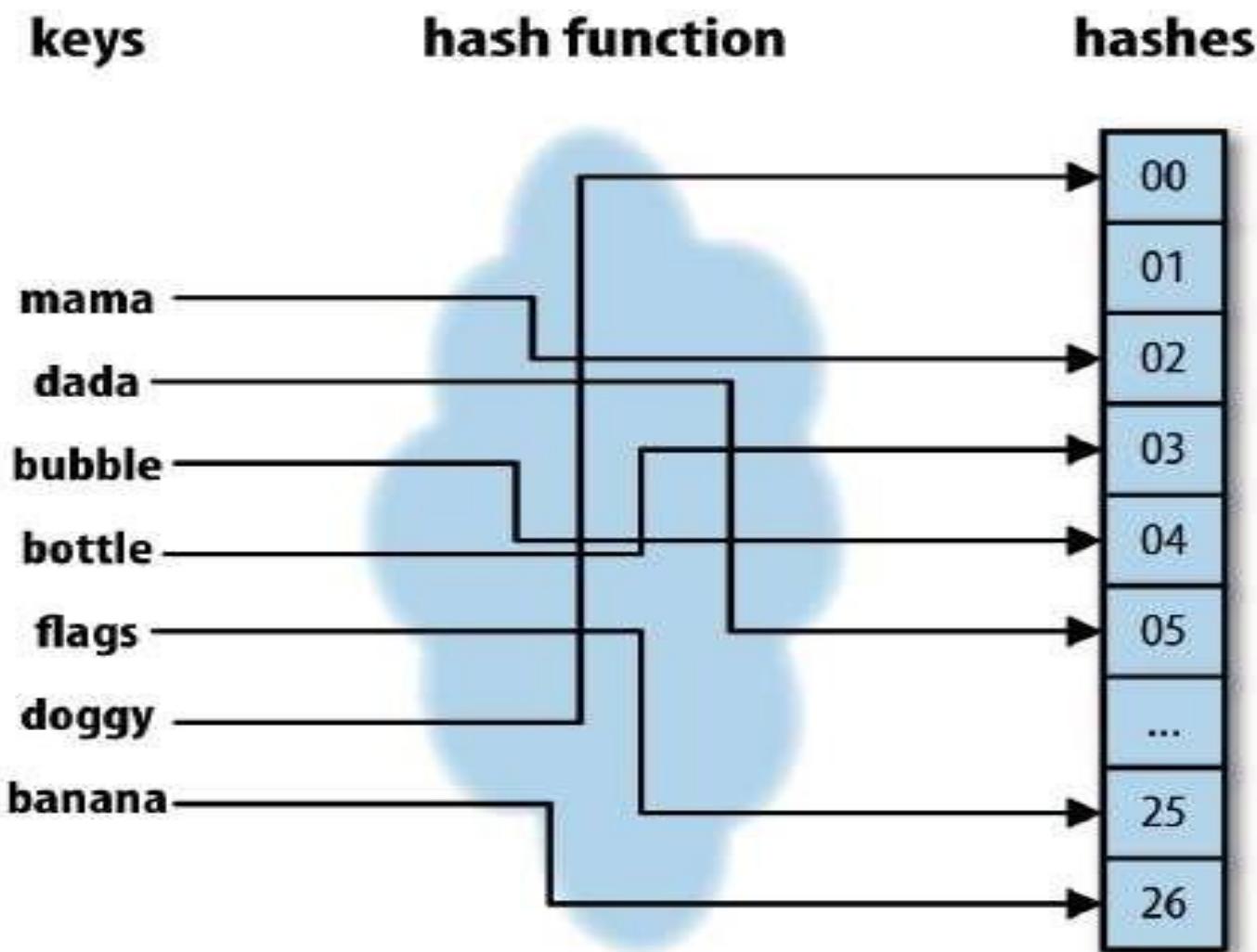


Fig. Hash functions mapping

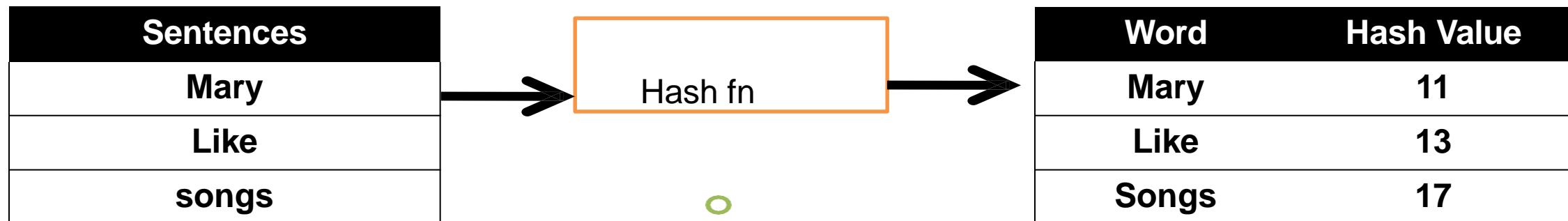
Feature Hashing

```
def hash_features(word_list, m):
    for word in word_list: index =
        hash_fn(word) % m  output[index]
        += 1
    return output
```

- Here **m = m-dimensional vector**
- **hash_fn** is any hashing function

Feature Hashing

- EG:



Here any
dummy Hash
function has
been used

Feature Hashing

- EG: let $m = 4$ i.e. four dimensional vector

Word	Hash Value(h)	Index (h %m)
Mary	11	3
Like	13	1
Songs	10	2

- Feature Vector

Sentences	Feature 1 (Remainder -0)	Feature 2 (Remainder -1)	Feature 3 (Remainder -2)	Feature 4 (Remainder -3)
Mary	0	0	0	1
Like	0	1	0	0
songs	0	0	1	0

Feature Hashing

- Advantages:
 - Significant advantage can be gained in **memory usage**
 - Feature hashing is well-suited to online learning scenarios, systems with **limited resources**, or **when speed and simplicity are important.**
- Disadvantages:
 - The ability to perform **the inverse mapping from feature indices back to feature values is lost**

Distance Measurement

- In NLP, we also want to find the similarity among sentence or document. Text is not like number and coordination that we cannot compare the different between “Apple” and “Orange” but similarity score can be calculated.
- **Why?**
 - Since we cannot simply subtract between “Apple is fruit” and “Orange is fruit” so that we have to find a way to convert text to numeric in order to calculate it. Having the score, we can understand how similar among two objects.
- **When?**
 - Compare whether 2 article are describing same news
 - Identifying similar documents
 - Classifying the category by giving product description

Distance Measurement

- How?
 - Euclidean Distance
 - Cosine Distance
 - Jaccard Similarity
- Before any distance measurement, text have to be tokenized.
- **Euclidean Distance**
- Comparing the shortest distance among two objects. It uses Pythagorean Theorem.
- Score means the distance between two objects. If it is 0, it means that both objects are identical. The following example shows score when comparing the first sentence.

Distance Measurement

- Euclidean Distance

Euclidean
$$\sqrt{\sum_{i=1}^k (x_i - y_i)^2}$$

```
Master Sentence: Elon Musk's Boring Co to build high-speed airport  
link in Chicago
```

```
Score: 0.00, Comparing Sentence: Elon Musk's Boring Co to build high-  
speed airport link in Chicago
```

```
Score: 1.73, Comparing Sentence: Elon Musk's Boring Company to build  
high-speed Chicago airport link
```

```
Score: 4.36, Comparing Sentence: Elon Musk's Boring Company approved  
to build high-speed transit between downtown Chicago and O'Hare  
Airport
```

```
Score: 4.24, Comparing Sentence: Both apple and orange are fruit
```

Distance Measurement

- **Cosine Similarity**
- Determine the angle between two objects is the calculation method to find similarity. The range of score is 0 to 1. If score is 1, it means that they are same in orientation (not magnitude). The following example shows score when comparing the first sentence.
- The measure computes the cosine of the angle between vectors **x and y**.
 - **A cosine value of 0 means that the two vectors are at 90 degrees to each other (orthogonal) and have no match.**
 - **The closer the cosine value to 1, the smaller the angle and the greater the match between vectors.**

Distance Measures

- **Cosine Similarity**
 - Let x and y be two vectors for comparison

$$sim(x, y) = \frac{\mathbf{x} \cdot \mathbf{y}}{\|\mathbf{x}\| \|\mathbf{y}\|},$$

- $\|\mathbf{x}\|$ is Euclidean norm of vector $x=(x_1, x_2, \dots, x_p)$
- **Can be Calculated using:**

$$\sqrt{x_1^2 + x_2^2 + \dots + x_p^2}.$$

Distance Measures

- **Cosine Similarity**

- Document Vector or Term-Frequency Vector

Document	team	coach	hockey	baseball	soccer	penalty	score	win	loss	season
<i>Document1</i>	5	0	3	0	2	0	0	2	0	0
<i>Document2</i>	3	0	2	0	1	1	0	1	0	1
<i>Document3</i>	0	7	0	2	1	0	0	3	0	0
<i>Document4</i>	0	1	0	0	1	2	2	0	3	0

Distance Measures

- **Cosine similarity between two term-frequency vectors.**
- first two term-frequency vectors in Table. That is, $\mathbf{x} = (5, 0, 3, 0, 2, 0, 0, 2, 0, 0)$ and $\mathbf{y} = (3, 0, 2, 0, 1, 1, 0, 1, 0, 1)$.

$$\begin{aligned}\mathbf{x}^t \cdot \mathbf{y} &= 5 \times 3 + 0 \times 0 + 3 \times 2 + 0 \times 0 + 2 \times 1 + 0 \times 1 + 0 \times 0 + 2 \times 1 \\ &\quad + 0 \times 0 + 0 \times 1 = 25\end{aligned}$$

$$\|\mathbf{x}\| = \sqrt{5^2 + 0^2 + 3^2 + 0^2 + 2^2 + 0^2 + 0^2 + 2^2 + 0^2 + 0^2} = 6.48$$

$$\|\mathbf{y}\| = \sqrt{3^2 + 0^2 + 2^2 + 0^2 + 1^2 + 1^2 + 0^2 + 1^2 + 0^2 + 1^2} = 4.12$$

$$sim(\mathbf{x}, \mathbf{y}) = 0.94$$

- if we were using the cosine similarity measure to compare these documents, they would be considered quite similar

Distance Measures

- Cosine Similarity Applications:

Similarity between 2 Docs:

Sentence 1: AI is our friend and it has been friendly

Sentence 2: AI and humans have always been friendly

Soln:

- lemmatization to reduce words to the same root word.
- “friend” and “friendly” will both become “friend”, “has” and “have” will both become “has”.

Term Frequencies:												
Sentence	AI	IS	FRIEND	HUMAN	ALWAYS	AND	BEEN	OUR	IT	HAS		
1	1	1	2	0	0	1	1	1	1	1		
2	1	0	1	1	1	1	1	0	0	1		

Distance Measures

- Cosine Similarity Applications:

Term Frequencies:												
Sentence	AI	IS	FRIEND	HUMAN	ALWAYS	AND	BEEN	OUR	IT	HAS		
1	1	1	2	0	0	1	1	1	1	1	1	1
2	1	0	1	1	1	1	1	1	0	0	0	1

– Cosine: $(1+2+1+1+1)/ \sqrt{11} * \sqrt{7} = 0.684$

Distance Measures

- Cosine Similarity Applications:

Term Frequencies:												
Sentence	AI	IS	FRIEND	HUMAN	ALWAYS	AND	BEEN	OUR	IT	HAS		
1		2	1	2	0	0	1	1	1	1	1	1
2		1	0	1	1	1	1	1	0	0	0	1

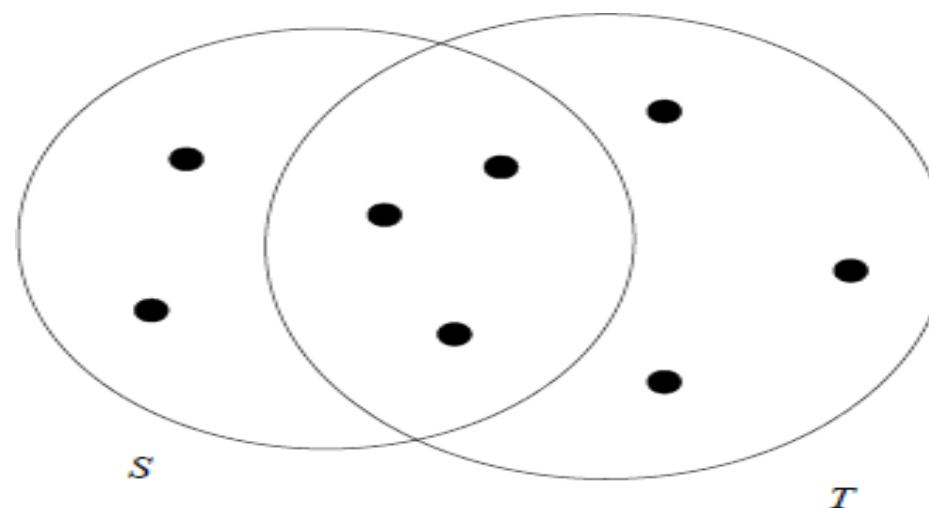
- If AI appears 2 times in sen:1
- $(2+2+1+1+1)/\sqrt{14} * \sqrt{7} = 0.7071$

Distance Measurement

- **Jaccard Similarity**
- The measurement is refer to number of common words over all words. More commons mean both objects should be similarity.
- Jaccard Similarity = $(\text{Intersection of A and B}) / (\text{Union of A and B})$
- The range is 0 to 1. If score is 1, it means that they are identical. There is no any common word between the first sentence and the last sentence so the score is 0. The following example shows score when comparing the first sentence.

Distance Measures

- **Jaccard similarity :**
 - The Jaccard similarity of sets S and T
 - is $|S \cap T| / |S \cup T|$,
 - that is, the ratio of the size of the intersection of S and T to the size of their union.



$$\text{sim}(s,t) = 3/8$$

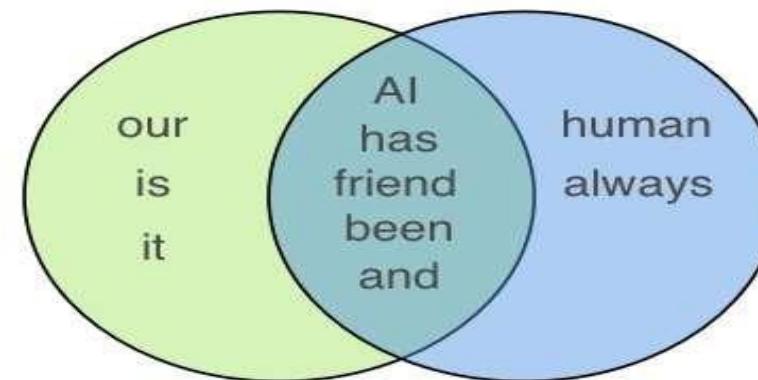
Distance Measures

- Jaccard Similarity Applications:

Similarity between 2 Docs:

Sentence 1: AI is our friend and it has been friendly
Sentence 2: AI and humans have always been friendly

Soln:



- lemmatization to reduce words to the same root word.
- “friend” and “friendly” will both become “friend”, “has” and “have” will both become “has”.
- Jaccard similarity of $5/(5+3+2) = 0.5$

Distance Measures

- Jaccard Similarity Applications:
- **Recommend Movies /Recommend Products**
 - If a customer rated a movie highly, put “liked” for that movie in the customer’s set.
 - If they gave a low rating to a movie, put “hated” for that movie in their set. Then, we can look for high Jaccard similarity among these sets.
 - Eg: If ratings are **1-to-5-stars**, *put a movie in a customer’s set n times if they rated the movie n-stars.*
- Then, use Jaccard similarity for bags when measuring the similarity of customers.
- The Jaccard **similarity for bags B and C is defined by counting an element n times in the intersection if n is the minimum of the number of times the element appears in B and C.**
 - In the union, we count the element the sum of the number of times it appears in B and in C.
 - The bag-similarity of bags **{a, a, a, b}** and **{a, a, b, b, c}** is **1/3**.
 - The intersection counts a twice and b once, so its size is 3. The size of the union of two bags is always the sum of the sizes of the two bags, or 9 in this case.

Distance Measures

- Jaccard vs. Cosine Similarity Applications:
 - Jaccard similarity takes only **unique set of words** for each sentence / document while cosine similarity takes **total length of the vectors**.

Distance Measures

- **Hamming Distance**
- Hamming distance calculates the distance between two binary vectors, also referred to as binary strings or bit strings for short.
- You are most likely going to encounter bit strings when you one-hot encode categorical columns of data.
- For example, if a column had the categories ‘red,’ ‘green,’ and ‘blue,’ you might one hot encode each example as a bit string with one bit for each column.
- red = [1, 0, 0]
- green = [0, 1, 0]
- blue = [0, 0, 1]

Distance Measures

- **Hamming Distance**
- The distance between red and green could be calculated as the sum or the average number of bit differences between the two bitstrings. This is the Hamming distance.
- For a one-hot encoded string, it might make more sense to summarize to the sum of the bit differences between the strings, which will always be a 0 or 1.
- Hamming Distance = sum for i to N $\text{abs}(v1[i] - v2[i])$
- For bitstrings that may have many 1 bits, it is more common to calculate the average number of bit differences to give a hamming distance score between 0 (identical) and 1 (all different).
- Hamming Distance = $(\text{sum for } i \text{ to } N \text{ abs}(v1[i] - v2[i])) / N$

Distance Measures

- **Hamming Distance**
 - Given two vectors to be the number of components in **which they differ**
 - The Hamming distance allows only substitution, hence, it only applies to strings of the same length.
 - Most commonly, Hamming distance is used when the vectors are Boolean; they consist of 0's and 1's only.
 - However, in principle, the vectors can have components from any set.
 - Eg:
 - The Hamming distance between the vectors 10101 and 11110 is 3.
 - That is, these vectors differ in the second, fourth, and fifth components, while they agree in the first and third components.

Distance Measures

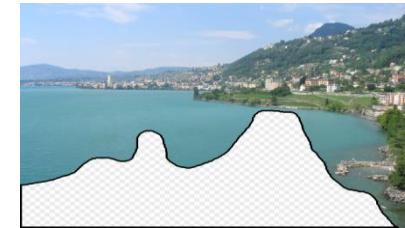
- **Hamming Distance**
 - Applications:
 - Information theory, coding theory, and cryptography.
 - *Error detecting and error correcting codes*
 - telecommunication
 - to count the number of flipped bits in a fixed-length binary word as an estimate of error, and therefore is sometimes called the **signal distance**

Thank you.

Locality Sensitive Hashing (LSH)

A Common Metaphor

- Many problems can be expressed as finding “similar” sets:
 - Find near-neighbors in high-dimensional space
- Examples:
 - Pages with similar words
 - For duplicate detection, classification by topic
 - Customers who purchased similar products
 - Products with similar customer sets
 - Images with similar features
 - Users who visited similar websites



Locality Sensitive Hashing (LSH)

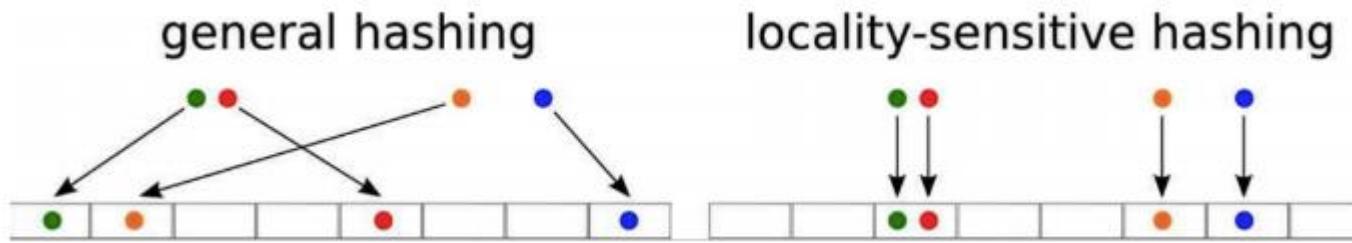
- The task of finding nearest neighbours is very common.
- You can think of applications **like finding duplicate or similar documents, audio/video search**. Although using brute force to check for all possible combinations will give you the exact nearest neighbour but it's not scalable at all.
- Approximate algorithms to accomplish this task has been an area of active research. Although these algorithms don't guarantee to give you the exact answer, more often than not they'll be provide a good approximation. These algorithms are faster and scalable.
- Locality sensitive hashing (LSH) is a procedure for finding similar pairs in a large dataset. For a dataset of size N , the brute force method of comparing every possible pair would take $N!/(2!(N-2)!) \sim N^2/2 = O(N^2)$ time. The LSH method aims to cut this down to $O(N)$ time.

Locality Sensitive Hashing (LSH)

- Locality sensitive hashing (LSH) is one such algorithm. LSH has many applications, including:
 - Near-duplicate detection: LSH is commonly used to deduplicate large quantities of documents, webpages, and other files.
 - Genome-wide association study: Biologists often use LSH to identify similar gene expressions in genome databases.
 - Large-scale image search: Google used LSH along with PageRank to build their image search technology VisualRank.
 - Audio/video fingerprinting: In multimedia technologies, LSH is widely used as a fingerprinting technique A/V data.

Locality Sensitive Hashing (LSH)

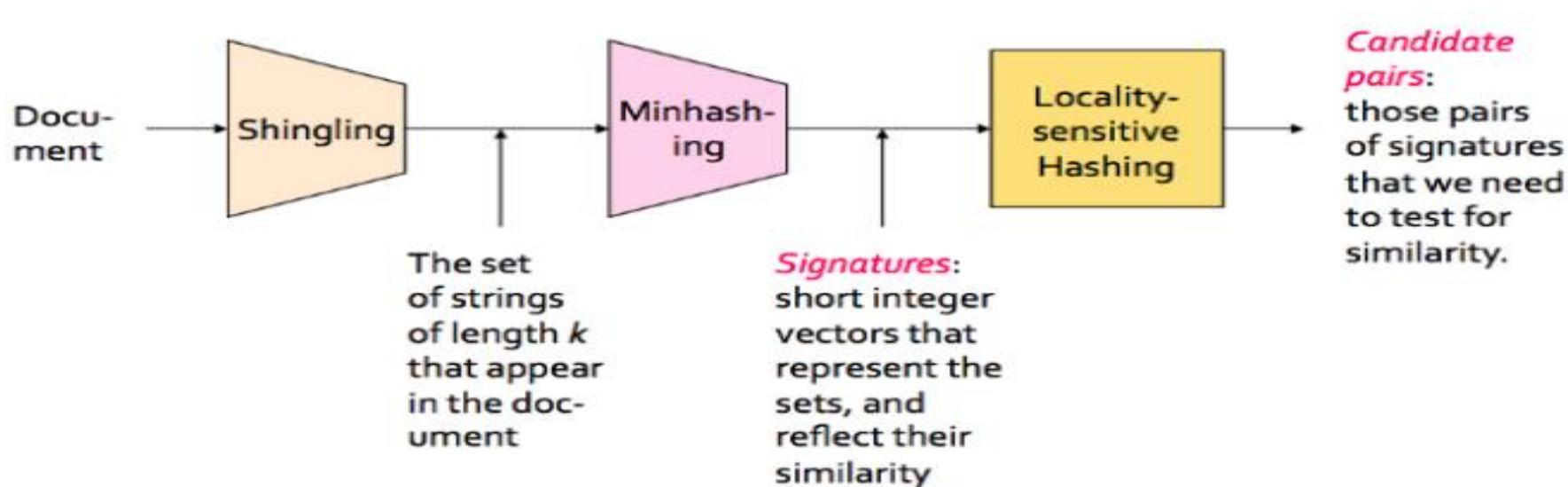
- LSH refers to a family of functions (known as LSH families) to hash data points into buckets so that **data points near each other are located in the same buckets with high probability**, while data points far from each other are likely to be in different buckets. This makes it easier to identify observations with various degrees of similarity.

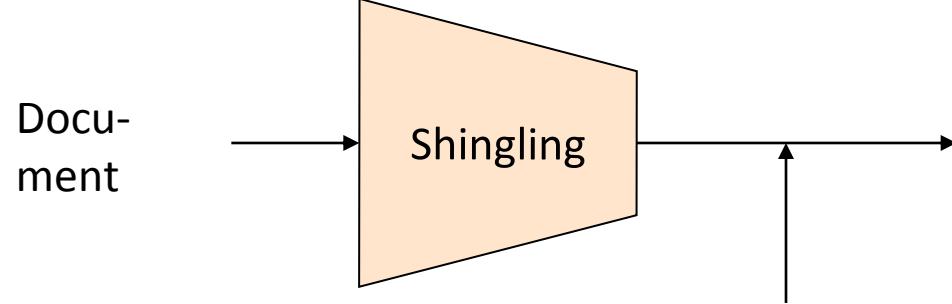


- Finding similar documents
 - how we can leverage LSH in solving an actual problem. The problem that we're trying to solve: **Goal:** You have been given a large collection of documents. You want to find “near duplicate” pairs.

Locality Sensitive Hashing (LSH)

- In the context of this problem, we can break down the LSH algorithm into 3 broad steps:
- **Shingling** - Convert documents to sets
- **Min hashing** – Convert large sets to short signatures while preserving similarity. Signature estimate the Jaccard similarity of sets.
- **Locality-sensitive hashing** - Focus on pairs of signatures likely to be from similar documents





The set
of strings
of length k
that appear
in the doc-
ument

Shingling

Step 1: *Shingling*: Convert documents to sets

In order to quantify a document, we need to vectorize it. One method for doing this, is to enumerate all of its **k -shingles**. A **k -shingle** is just any k -consecutive characters occurring in the document.

Locality Sensitive Hashing (LSH)

- **Shingling**
 - In this step, we convert each document into a set of characters of length k (also known as k -shingles or k -grams). The key idea is to represent each document in our collection as a set of k -shingles.
 - For ex: One of your document (D): “Nadal”. Now if we’re interested in 2-shingles, then our set: {Na, ad, da, al}. Similarly set of 3-shingles: {Nad, ada, dal}.
 - Similar documents are more likely to share more shingles
 - k value of 8–10 is generally used in practice. A small value will result in many shingles which are present in most of the documents (bad for differentiating documents)

Finding Similar Items

- **Simple approaches:**
 - Document = set of words appearing in document
 - Document = set of “important” words
- A k-shingle (or k-gram) for a document is a sequence of k tokens that appears in the doc
 - Tokens can be characters, words or something else, depending on the application
 - Assume tokens = characters for examples
 - **Example:** $k=2$; document $D_1 = \text{abcab}$
Set of 2-shingles: $S(D_1) = \{\text{ab}, \text{bc}, \text{ca}\}$

Finding Similar Items

- **Finding Similar Documents (3 – Steps)**
- **Step 1: Shingling: Convert documents to sets**
- "a rose is a rose is a rose"
 - The set of all contiguous sequences of 4 tokens (Thus $4=n$, thus 4-grams) is
 - { (a,rose,is,a), (rose,is,a,rose), (is,a,rose,is),
(a,rose,is,a), (rose,is,a,rose) }
 - reduced to { (a,rose,is,a), (rose,is,a,rose),
(is,a,rose,is) }.

Finding Similar Items

- Choosing the Shingle Size
 - k should be picked large enough that the probability of any given shingle appearing in any given document is low.
 - Effect of k : k = 1, most Web pages will have most of the common characters and few other characters, so almost all Web pages will have high similarity.
 - Short (E-mail): k = 5
 - large documents, such as research articles: choice k = 9

Choosing k

- Let $k = 5$
- Suppose that only letters and general white space characters appear in emails.
- If so, there would be $27^5 = 14,348,907$ possible shingles.
- Since the typical email is much smaller than 14 million characters long, we would expect $k = 5$ to work well and it does.
- A rule of thumb is to imagine there are 20 characters and estimate the number of shingles as 20^k . For large documents (research articles), a choice of $k = 9$ is considered to be safe

Finding Similar Items

- Matrix Representation of Sets
 - To construct small signatures from large sets,
 - First : visualize a collection of sets as their characteristic matrix
 - The **columns of the matrix** correspond to the sets,
 - **The rows correspond to elements** of the universal set from which elements of the sets are drawn.

Finding Similar Items

- Matrix Representation of Sets

– Here, $S_1 = \{a, d\}$, $S_2 = \{c\}$, $S_3 = \{b, d, e\}$, and $S_4 = \{a, c, d\}$.

rows correspond to elements of the universal set

columns of the matrix correspond to the sets ..like $S_1 = \{a, b\}$

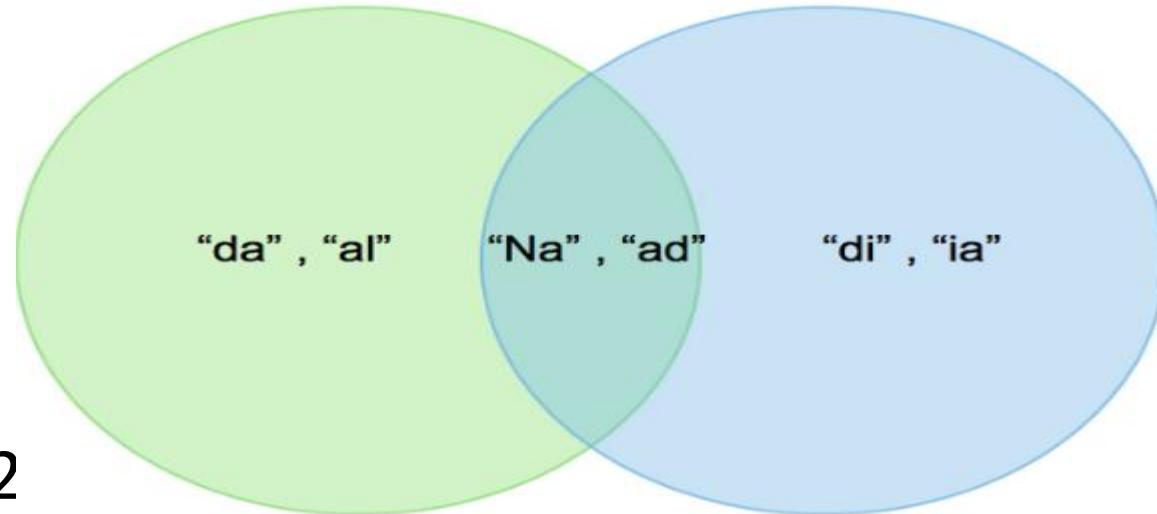
Element	S_1	S_2	S_3	S_4
a	1	0	0	1
b	0	0	1	0
c	0	1	0	1
d	1	0	1	1
e	0	0	1	0

Fig 1: Matrix Representation of sets

Locality Sensitive Hashing (LSH)

- Jaccard Index
 - The **Jaccard similarity** of two **sets** is the size of their intersection divided by the size of their union:
$$sim(C_1, C_2) = |C_1 \cap C_2| / |C_1 \cup C_2|$$
 - **Jaccard distance:** $d(C_1, C_2) = 1 - |C_1 \cap C_2| / |C_1 \cup C_2|$
 - Suppose A: “Nadal” and B: “Nadia”, then 2-shingles representation will be: A: {Na, ad, da, al} and B: {Na, ad, di, ia}.

Locality Sensitive Hashing (LSH)



- Jaccard Index = 2
- More number of common shingles will result in bigger Jaccard Index and hence more likely that the documents are similar.
- Let's discuss 2 big issues that we need to tackle:
 - Time complexity
 - Space complexity

Locality Sensitive Hashing (LSH)

- **Time complexity**
 - Now you may be thinking that we can stop here. But if you think about the scalability, doing just this won't work. For a collection of n documents, you need to do $n*(n-1)/2$ comparison, basically $O(n^2)$. Imagine you have 1 million documents, then the number of comparison will be $5*10^{11}$.
- **Space complexity**
 - The document matrix is a sparse matrix and storing it as it is will be a big memory overhead.
 - One way to solve this is hashing.

Locality Sensitive Hashing (LSH)

- **Hashing**
- The idea of hashing is to convert each document to a small signature using a hashing function H . Suppose a document in our corpus is denoted by d . Then:
- $H(d)$ is the signature and it's small enough to fit in memory
- If $\text{similarity}(d_1, d_2)$ is high then $\text{Probability}(H(d_1) == H(d_2))$ is high
- If $\text{similarity}(d_1, d_2)$ is low then $\text{Probability}(H(d_1) == H(d_2))$ is low
- Choice of hashing function is tightly linked to the similarity metric we're using. For Jaccard similarity the appropriate hashing function is min-hashing.

Locality Sensitive Hashing (LSH)

- Computing Minhash values
 - Rows are permuted randomly
 - Minhasfunc $h(C) =$ the number of the first (n the permuted order) row in which column C has 1.
 - Independent hash functions to create signature of each column.

Minhashing

- Let the order of rows **beadc** be the random permutation for the matrix (fig 1)
- It defines a minhash function h that maps sets to rows.

<i>Element</i>	S_1	S_2	S_3	S_4
<i>b</i>	0	0	1	0
<i>e</i>	0	0	1	0
<i>a</i>	1	0	0	1
<i>d</i>	1	0	1	1
<i>c</i>	0	1	0	1

Minhashing

- To compute *the minhash value of set S_1 according to h .*
- Check for first ‘1’ in the column:
 - The first column, which is the column for set S_1 , has 0 in row b,
 - So we proceed to row e, the second in the permuted order. There is again a 0 in the column for S_1 ,
 - So we proceed to row a, where we find a 1.
 - $h(S_1) = a$.
 - **(Obtained similarly) $h(S_2) = c$, $h(S_3) = b$, and $h(S_4) = a$**

<i>Element</i>	S_1	S_2	S_3	S_4
<i>b</i>	0	0	1	0
<i>e</i>	0	0	1	0
<i>a</i>	1	0	0	1
<i>d</i>	1	0	1	1
<i>c</i>	0	1	0	1

Minhashing

- Let **Random number n** (say, 100s or 1000s) be the
- Let minhash functions determined by these permutations h_1, h_2, \dots, h_n .
- From the column representing set S ,
 - construct the minhash signature for S , the vector $[h_1(S), h_2(S), \dots, h_n(S)]$. List hash-values as columns

<i>Element</i>	S_1	S_2	S_3	S_4	
b	0	0	1	0	
e	0	0	1	0	
a	1	0	0	1	
d	1	0	1	1	
c	0	1	0	1	

a	c	b	a
---	---	---	---

Minhashing

- **Minhashing and Jaccard Similarity**
- The probability that the minhash function for a random permutation of rows produces the same value for two sets equals the jaccard similarity of those sets.
- X rows (1's in both cols)
- Y rows (1 in one of the columns, 0 in the other)
- Z rows (0 in both cols)

<u>C₁</u>	<u>C₂</u>	
0	1	*
1	0	*
1	1	* *
0	0	
1	1	* *
0	1	*

Sim(C₁, C₂) =
2/5 = 0.4

Minhashing

- Min hashing (Example-2)

➤ **Step 1:** Random permutation (π) of *row index* of document shingle matrix.

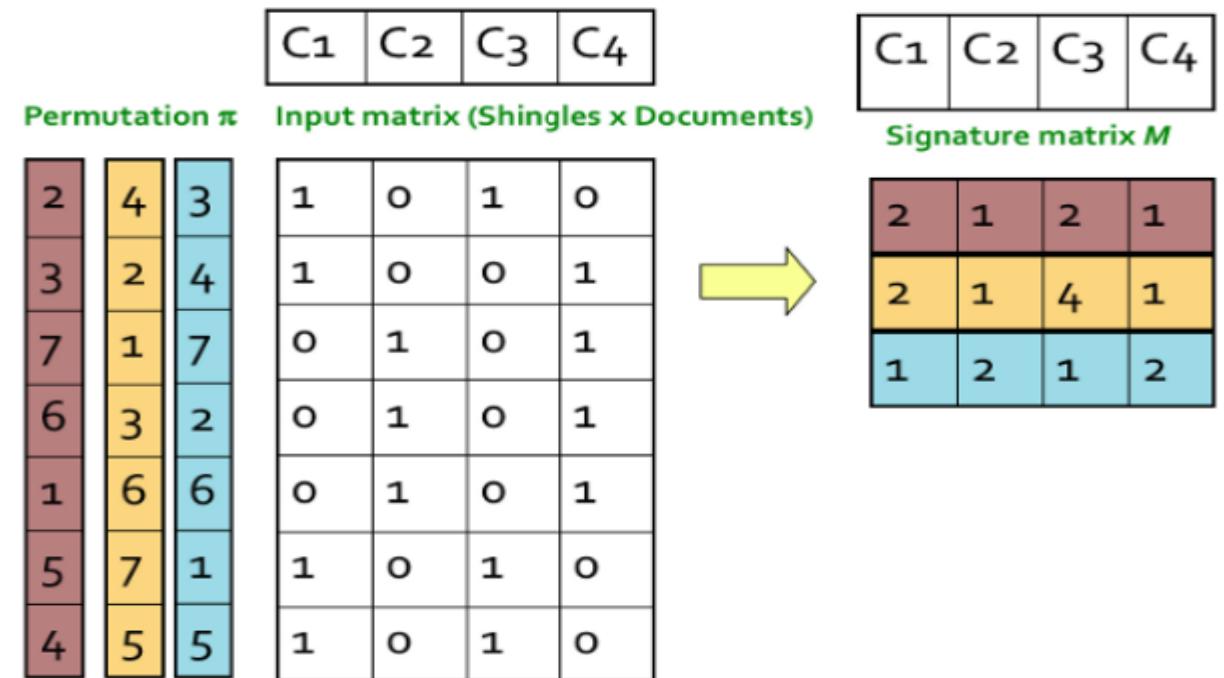
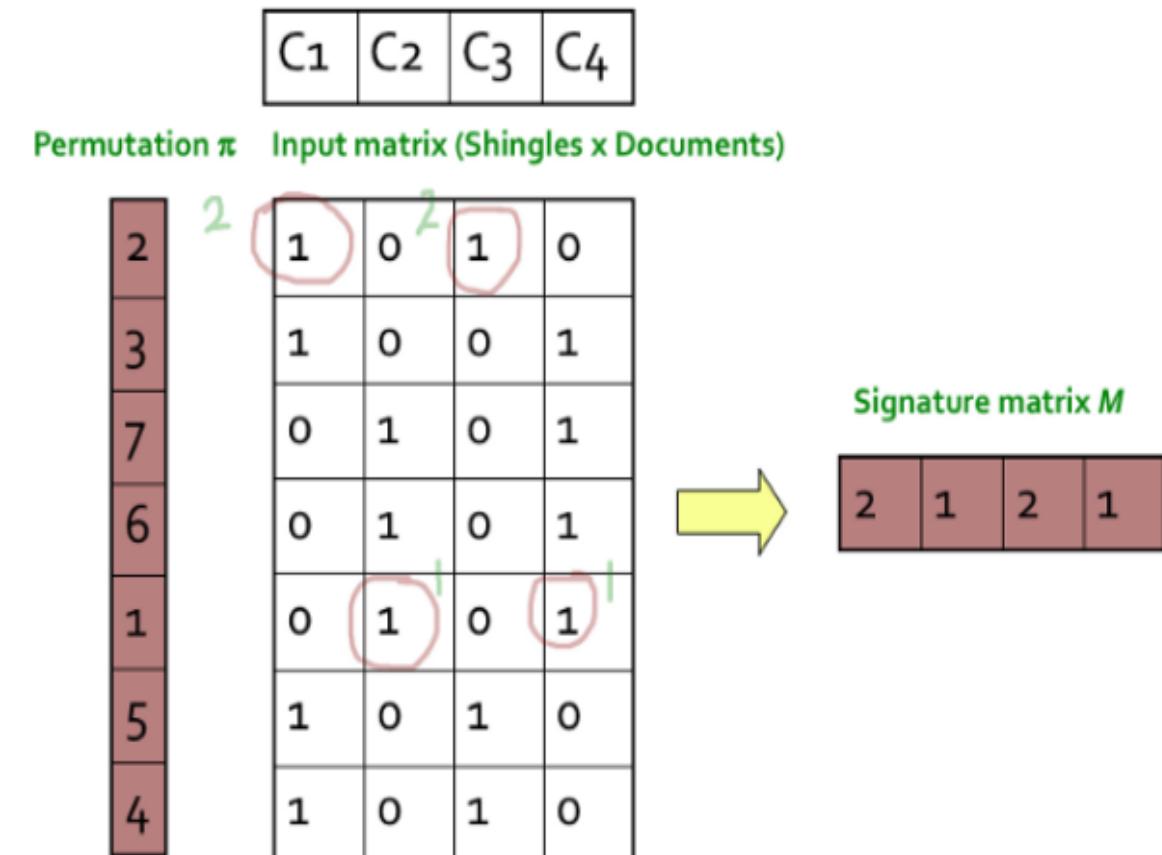
Permutation π	Input matrix (Shingles x Documents)			
2	1	0	1	0
3	1	0	0	1
7	0	1	0	1
6	0	1	0	1
1	0	1	0	1
5	1	0	1	0
4	1	0	1	0

- **Step 2:** Hash function is the index of the first (in the permuted order) row in which column C has value 1. Do this several time (use different permutations) to create signature of a column.

$$h_{\pi}(C) = \min_{\pi} \pi(C)$$

Minhashing

- Min hashing



Minhashing

- Algorithm

- Let $SIG(i, c)$ be the element of the signature matrix for the i th hash function and column c . Initially, set $SIG(i, c)$ to ∞ for all i and c

For each row r do begin

 For each hash function h_i do:

 Compute $h_i(r)$;

 For each column c

 If c has 1 in row r

 For each hash function h_i do:



 If $h_i(r) < SIG(i, c)$ then

$SIG(i, c) = h_i(r)$

Minhashing

- Algorithm

	S_1	S_2	S_3	S_4		
h_1	∞	∞	∞	∞		
h_2	∞	∞	∞	∞		

Row	S_1	S_2	S_3	S_4	$x + 1 \bmod 5$	$3x + 1 \bmod 5$
0	1	0	0	1	1	1
1	0	0	1	0	2	4
2	0	1	0	1	3	2
3	1	0	1	1	4	0
4	0	0	1	0	0	3

	S_1	S_2	S_3	S_4		
h_1	1	∞	2	1		
h_2	1	∞	4	1		

Row	S_1	S_2	S_3	S_4	$x + 1 \bmod 5$	$3x + 1 \bmod 5$
0	1	0	0	1	1	1
1	0	0	1	0	2	4
2	0	1	0	1	3	2
3	1	0	1	1	4	0
4	0	0	1	0	0	3

Row	S_1	S_2	S_3	S_4	$x + 1 \bmod 5$	$3x + 1 \bmod 5$
0	1	0	0	1	1	1
1	0	0	1	0	2	4
2	0	1	0	1	3	2
3	1	0	1	1	4	0
4	0	0	1	0	0	3

	S_1	S_2	S_3	S_4
h_1	∞	∞	∞	∞
h_2	∞	∞	∞	∞

	S_1	S_2	S_3	S_4
h_1	1	3	2	1
h_2	1	2	4	1

Row	S_1	S_2	S_3	S_4	$x + 1 \bmod 5$	$3x + 1 \bmod 5$
0	1	0	0	1	1	1
1	0	0	1	0	2	4
2	0	1	0	1	3	2
3	1	0	1	1	4	0
4	0	0	1	0	0	3

	S_1	S_2	S_3	S_4
h_1	1	3	2	1
h_2	1	2	0	0

	S_1	S_2	S_3	S_4
h_1	1	3	2	1
h_2	0	2	0	0

Row	S_1	S_2	S_3	S_4	$x + 1 \bmod 5$	$3x + 1 \bmod 5$
0	1	0	0	1	1	1
1	0	0	1	0	2	4
2	0	1	0	1	3	2
3	1	0	1	1	4	0
4	0	0	1	0	0	3

Minhashing

- Algorithm $P[h_\pi(C_1) = h_\pi(C_2)] = \text{sim}(C_1, C_2)$

	S_1	S_2	S_3	S_4
h_1	1	3	0	1
h_2	0	2	0	0

$$\text{sim}(\varsigma_1, \varsigma_2) = 1.0$$

Row	S_1	S_2	S_3	S_4	$x + 1 \bmod 5$	$3x + 1 \bmod 5$
0	1	0	0	1	1	1
1	0	0	1	0	2	4
2	0	1	0	1	3	2
3	1	0	1	1	4	0
4	0	0	1	0	0	3

$$\text{Jaccard sim} = \frac{2}{5}$$

	S_1	S_2	S_3	S_4
h_1	1	3	0	1
h_2	0	2	0	0

$$\text{sim}(\varsigma_1, \varsigma_2) = \frac{1}{2}$$

Row	S_1	S_2	S_3	S_4	$x + 1 \bmod 5$	$3x + 1 \bmod 5$
0	1	0	0	1	1	1
1	0	0	1	0	2	4
2	0	1	0	1	3	2
3	1	0	1	1	4	0
4	0	0	1	0	0	3

$$\text{sim}(\varsigma_1, \varsigma_3) = \frac{1}{4}$$

Minhashing

- Algorithm $P[h_\pi(C_1) = h_\pi(C_2)] = \text{sim}(C_1, C_2)$

	S_1	S_2	S_3	S_4
h_1	1	3	0	1
h_2	0	2	0	0

$$\text{sim}\{S_1, S_2\} = 0$$

Row	S_1	S_2	S_3	S_4	$x + 1 \bmod 5$	$3x + 1 \bmod 5$
0	1	0	0	1	1	1
1	0	0	1	0	2	4
2	0	1	0	1	3	2
3	1	0	1	1	4	0
4	0	0	1	0	0	3

$$\text{sim}\{S_1, S_2\} = 0$$

Min-Hashing Example

Permutation π

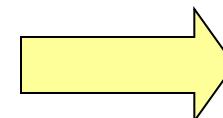
2	4	3
3	2	4
7	1	7
6	3	2
1	6	6
5	7	1
4	5	5

Input matrix (Shingles x Documents)

1	0	1	0
1	0	0	1
0	1	0	1
0	1	0	1
0	1	0	1
1	0	1	0
1	0	1	0

Signature matrix M

2	1	2	1
2	1	4	1
1	2	1	2

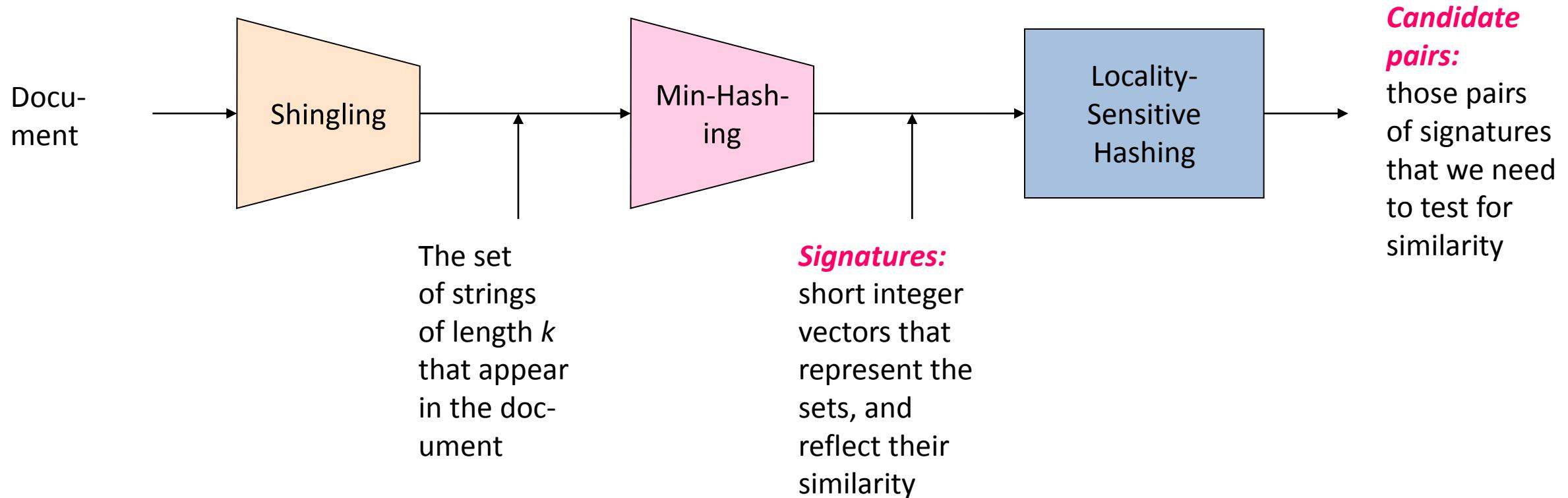


Similarities:

	1-3	2-4	1-2	3-4
Col/Col	0.75	0.75	0	0
Sig/Sig	0.67	1.00	0	0

Locality Sensitive Hashing (LSH)

- **Locality-sensitive hashing- The band structure procedure.**
- Goal: Find documents with Jaccard similarity of at least t
- The general idea of LSH is to find a algorithm such that if we input signatures of 2 documents, it tells us that those 2 documents from a candidate pair or not i.e. their similarity is greater than a threshold t . Remember that we are taking similarity of signatures as a proxy for Jaccard similarity between the original documents.
- The intuition is this: instead of comparing every pair of elements, what if we could just hash them into buckets, and hopefully elements that map to the same buckets will be the right level of “close” to each other. The level of “close” is precisely the desired similarity threshold



Locality Sensitive Hashing

Step 3: *Locality-Sensitive Hashing:*

Focus on pairs of signatures likely to be from similar documents

2	1	4	1
1	2	1	2
2	1	2	1

LSH: First Cut

- Goal: Find documents with Jaccard similarity at least s (for some similarity threshold, e.g., $s=0.8$)
- LSH – General idea: Use a function $f(x,y)$ that tells whether x and y is a *candidate pair*: a pair of elements whose similarity must be evaluated
- For Min-Hash matrices:
 - Hash columns of *signature matrix M* to many buckets
 - Each pair of documents that hashes into the same bucket is a *candidate pair*

Candidates from Min-Hash

2	1	4	1
1	2	1	2
2	1	2	1

- Pick a similarity threshold s ($0 < s < 1$)
- Columns x and y of M are a **candidate pair** if their signatures agree on at least fraction s of their rows:
 $M(i, x) = M(i, y)$ for at least frac. s values of i
 - We expect documents x and y to have the same (Jaccard) similarity as their signatures

LSH for Min-Hash

2	1	4	1
1	2	1	2
2	1	2	1

- **Big idea: Hash columns of signature matrix M several times**
- Arrange that (only) **similar columns** are likely to **hash to the same bucket**, with high probability
- **Candidate pairs are those that hash to the same bucket**

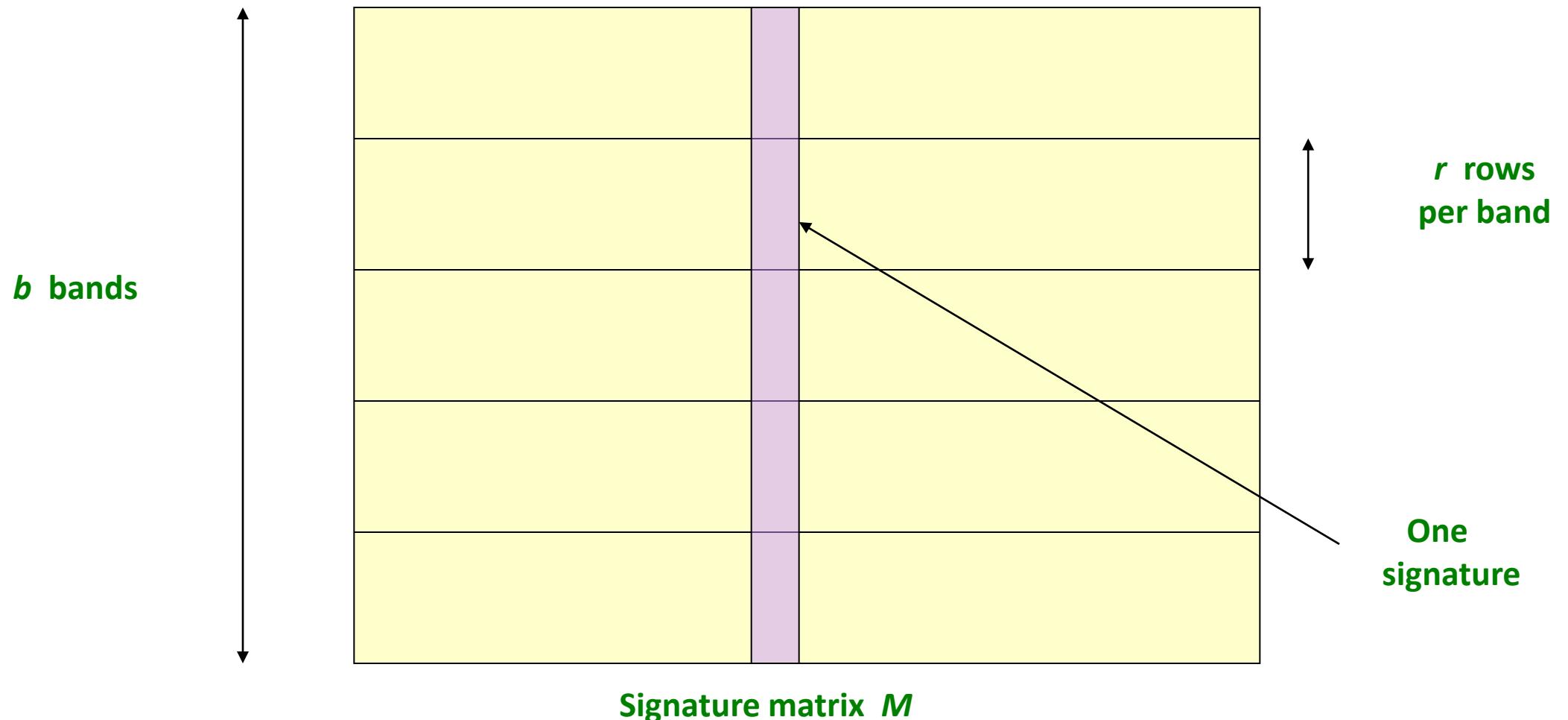
LSH for Min-Hash

2	1	4	1
1	2	1	2
2	1	2	1

- **Band Method**
- n is the number of hash functions, i.e. the dimension K of the signature matrix. Also define:
 - b as the number of bands
 - r as the number of rows per band
 - and it's apparent that $n=b \cdot r$.
- Two rows are considered **candidate pairs** (meaning they have the possibility to have a similarity score above the desired threshold), if the two rows have **at least one** band in which all the rows are identical
- Hopefully the intuition behind b and r should make sense here; **increasing b gives rows more chances to match**, so it let's in candidate pairs **with lower similarity scores**. **Increasing r makes the match criteria stricter**, restricting to **higher similarity scores**. r and b do opposite things

2	1	4	1
1	2	1	2
2	1	2	1

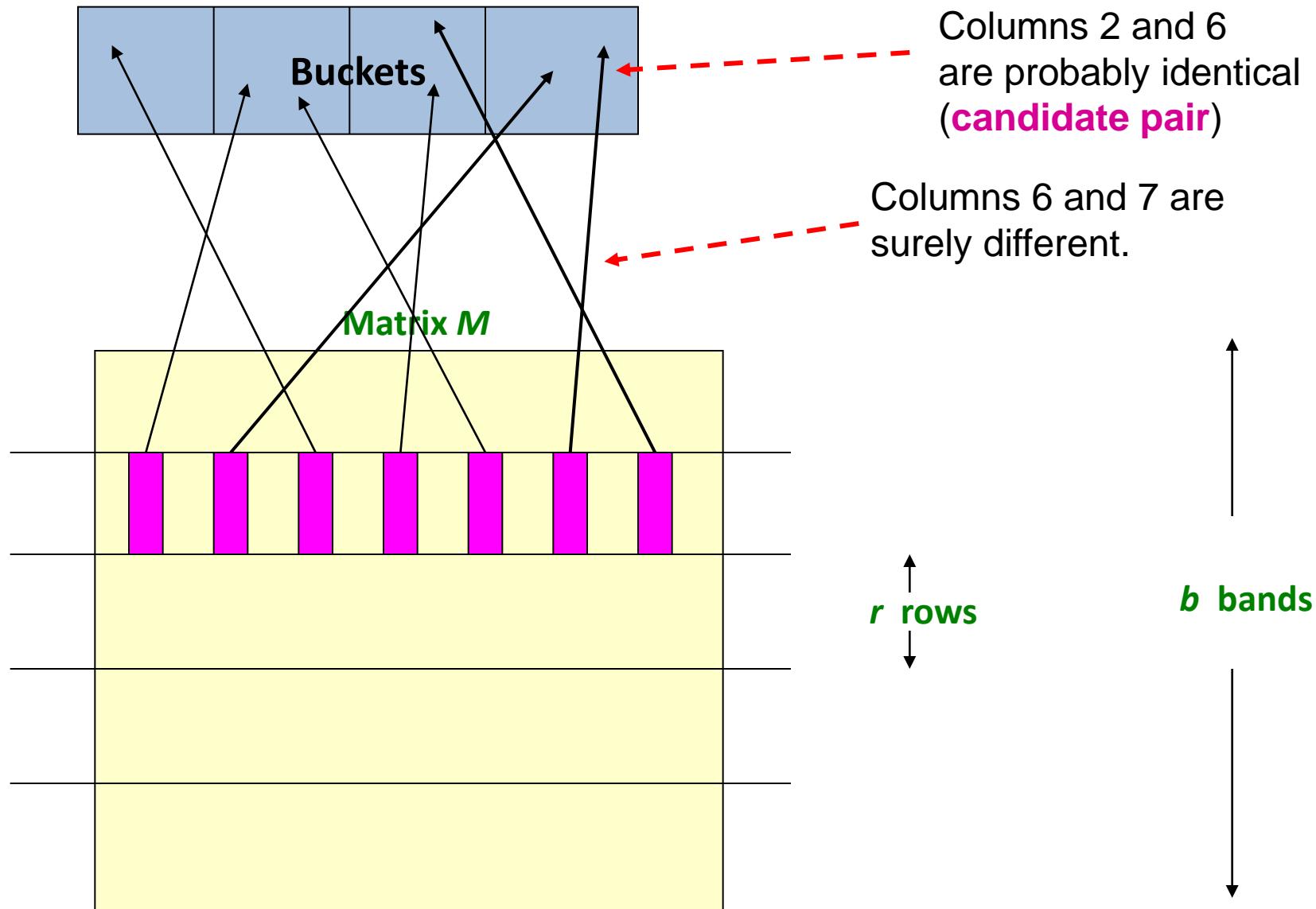
Partition M into b Bands



Partition M into Bands

- Divide matrix M into b bands of r rows
- For each band, hash its portion of each column to a hash table with k buckets
 - Make k as large as possible
- **Candidate** column pairs are those that hash to the same bucket for ≥ 1 band
- Tune b and r to catch most similar pairs, but few non-similar pairs

Hashing Bands



2	1	4	1
1	2	1	2
2	1	2	1

Example of Bands

Assume the following case:

- Suppose 100,000 columns of M (100k docs)
- Signatures of 100 integers (rows)
- Therefore, signatures take 40Mb
- Choose $b = 20$ bands of $r = 5$ integers/band
- **Goal:** Find pairs of documents that are at least $s = 0.8$ similar

2	1	4	1
1	2	1	2
2	1	2	1

C_1, C_2 are 80% Similar

- **Find pairs of $\geq s=0.8$ similarity, set $b=20, r=5$**
- **Assume:** $\text{sim}(C_1, C_2) = 0.8$
 - Since $\text{sim}(C_1, C_2) \geq s$, we want C_1, C_2 to be a **candidate pair**: We want them to hash to at least 1 common bucket (at least one band is identical)
- **Probability C_1, C_2 identical in one particular band:** $(0.8)^5 = 0.328$
- Probability C_1, C_2 are **not** similar in all of the 20 bands: $(1-0.328)^{20} = 0.00035$
 - i.e., about 1/3000th of the 80%-similar column pairs are **false negatives** (we miss them)
 - **We would find 99.965% pairs of truly similar documents**

2	1	4	1
1	2	1	2
2	1	2	1

C_1, C_2 are 30% Similar

- Find pairs of $\geq s=0.8$ similarity, set $b=20$, $r=5$
- Assume: $\text{sim}(C_1, C_2) = 0.3$
 - Since $\text{sim}(C_1, C_2) < s$ we want C_1, C_2 to hash to **NO common buckets** (all bands should be different)
- Probability C_1, C_2 identical in one particular band: $(0.3)^5 = 0.00243$
- Probability C_1, C_2 identical in at least 1 of 20 bands: $1 - (1 - 0.00243)^{20} = 0.0474$
 - In other words, approximately 4.74% pairs of docs with similarity 0.3% end up becoming **candidate pairs**
 - They are **false positives** since we will have to examine them (they are candidate pairs) but then it will turn out their similarity is below threshold s

2	1	4	1
1	2	1	2
2	1	2	1

LSH Involves a Tradeoff

- **Pick:**
 - The number of Min-Hashes (rows of M)
 - The number of bands b , and
 - The number of rows r per band
to balance false positives/negatives
- **Example:** If we had only 15 bands of 5 rows, the number of false positives would go down, but the number of false negatives would go up

b bands, r rows/band

- Columns C_1 and C_2 have similarity t
- Pick any band (r rows)
 - Prob. that all rows in band equal = t^r
 - Prob. that some row in band unequal = $1 - t^r$
- Prob. that no band identical = $(1 - t^r)^b$
- Prob. that at least 1 band identical = $1 - (1 - t^r)^b$

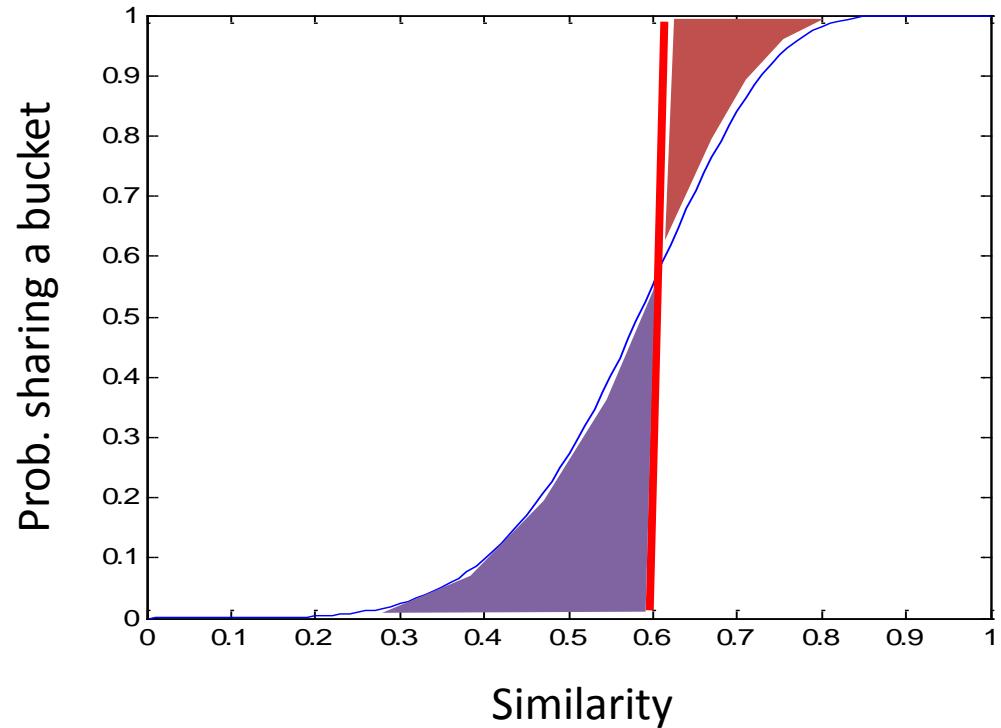
Example: $b = 20$; $r = 5$

- **Similarity threshold s**
- **Prob. that at least 1 band is identical:**

s	$1-(1-s^r)^b$
.2	.006
.3	.047
.4	.186
.5	.470
.6	.802
.7	.975
.8	.9996

Picking r and b : The S-curve

- **Picking r and b to get the best S-curve**
 - 50 hash-functions ($r=5$, $b=10$)



Blue area: False Negative rate
Green area: False Positive rate

LSH Summary

- Tune M , b , r to get almost all pairs with similar signatures, but eliminate most pairs that do not have similar signatures
- Check in main memory that **candidate pairs** really do have **similar signatures**
- **Optional:** In another pass through data, check that the remaining candidate pairs really represent similar documents

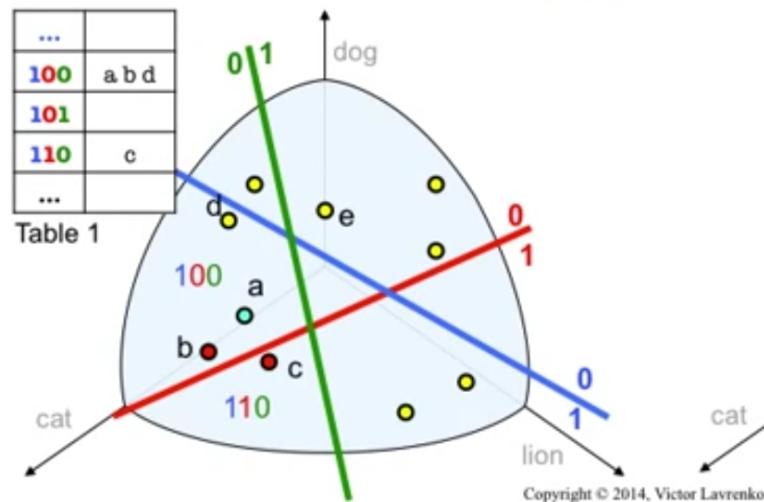
Summary: 3 Steps

- **Shingling:** Convert documents to sets
 - We used hashing to assign each shingle an ID
- **Min-Hashing:** Convert large sets to short signatures, while preserving similarity
 - We used **similarity preserving hashing** to generate signatures with property $\Pr[h_\pi(C_1) = h_\pi(C_2)] = \text{sim}(C_1, C_2)$
 - We used hashing to get around generating random permutations
- **Locality-Sensitive Hashing:** Focus on pairs of signatures likely to be from similar documents
 - We used hashing to find **candidate pairs** of similarity $\geq s$

Thank you.

Locality Sensitive Hashing

1. Want: similar hashcodes for nearby points
2. Generate random hyperplanes: $\mathbf{h}_1 \mathbf{h}_2 \mathbf{h}_3$
3. Hash-code for \mathbf{a} : $H_{\text{step}} [\mathbf{a}^T \mathbf{h}_1 \mathbf{a}^T \mathbf{h}_2 \mathbf{a}^T \mathbf{h}_3] = 100$
4. Compare \mathbf{a} to points with same hash-code
 - \mathbf{b} ... indeed similar to \mathbf{a}
 - \mathbf{d} ... false positive, will be eliminated
 - \mathbf{c} ... different hash-code, will miss it
5. Repeat w. different hyperplanes: $\mathbf{h}_4 \mathbf{h}_5 \mathbf{h}_6$



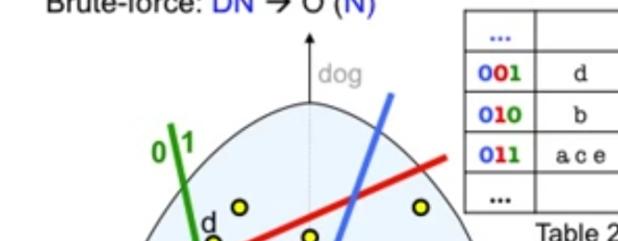
Computational cost:

- N points, D -dimensional, K hyperplanes
- DK ... find bucket where point lands
- $N/2^K$... points in that bucket (on avg)
- $DN/2^K$... cost of comparisons
- repeat everything L times (# tables)

LSH: $LDK + LDN/2^K \rightarrow O(\log N)$ if $K \sim \log N$

Index: $D(ND) / \sqrt{ND} \rightarrow O(\sqrt{N})$

Brute-force: $DN \rightarrow O(N)$



Data Stream Mining

New Topic: Infinite Data

High dim. data

Locality sensitive hashing

Clustering

Dimensionality reduction

Infinite data

Filtering data streams

Queries on streams

Web advertising

Graph data

PageRank,
SimRank

Community
Detection

Spam Detection

Machine learning

SVM

Decision Trees

Perceptron, kNN

Apps

Recommender systems

Association Rules

Duplicate document detection

Data Management Vs Stream Management

- In a DBMS, Input is under the control of the programming staff.
- Imagine a factory with 500 sensors capturing 10 KB of information every second, in one hour is captured nearby 36 GB of information and 432 GB daily. This massive information needs to be analysed in real time (or in the shortest time possible) to detect irregularities or deviations in the system and quickly react.
- In many data mining situations, we do not know the entire data set in advance. Therefore Stream Management is important when the input rate is controlled externally:
 - Example: Google queries
Twitter or Facebook status updates

Stream Model

- Data arrives as sequence of items. Sometimes continuously and at high speed. Therefore We can think of the **data** as **infinite** and **non stationary** (the distribution changes over time)
- Input **elements** enter at a rapid rate, at one or more input ports (i.e., **streams**)
 - **We call elements of the stream tuples**
- Can't store them all in main memory.
- **Q: How do you make critical calculations about the stream using a limited amount of (secondary) memory?**

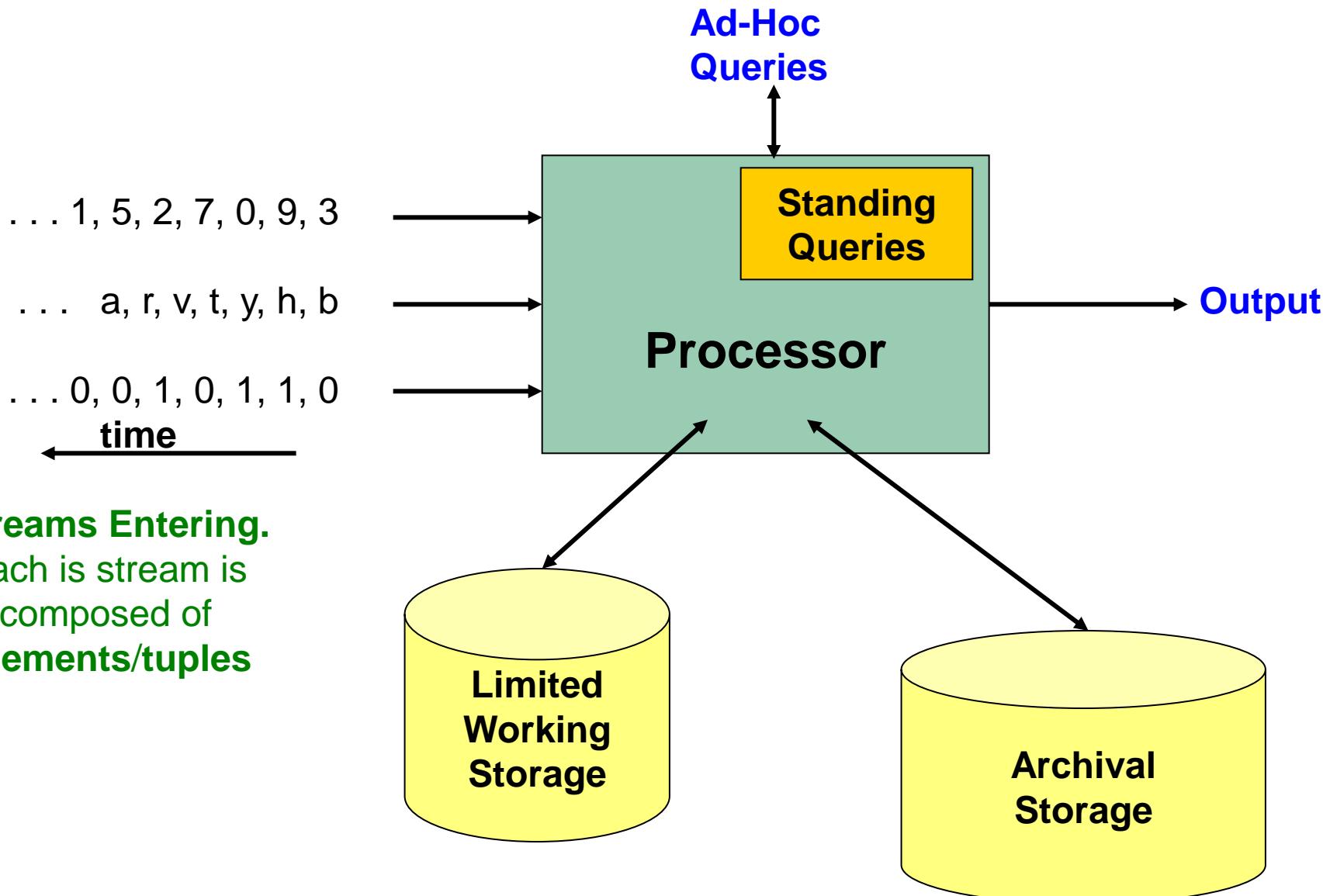
Stream Mining

- Stream Mining enables the analysis of massive quantities of data in real time using bounded resources
- Data Stream Mining is the process of extracting knowledge from continuous rapid data records which comes to the system in a stream. A data stream is an ordered sequence of instances that in many applications of data stream mining can be read only once or a small number of times using limited computing and storage capabilities.
- Data Stream Mining fulfil the following characteristics:
- **Continuous Stream of Data.** High amount of data in an infinite stream. we do not know the entire dataset
- **Concept Drifting.** The data change or evolves over time
- **Volatility of data.** The system does not store the data received (Limited resources). When data is analysed it's discarded or summarised

Two Forms of Query

- **Ad-hoc queries:** Normal queries asked one time about streams.
 - Example: What is maximum value seen so far in stream S?
- **Standing queries:** Queries that are, in principle, asked about the stream at all times.
 - Example: Report each new maximum value ever seen in stream S.

General Stream Processing Model



Applications

- **Mining query streams:** Google wants to know what queries are more frequent today than yesterday
- **Mining click streams:** Yahoo wants to know which of its pages are getting an unusual number of hits in the past hour
- **Mining social network news feeds:** E.g., look for trending topics on Twitter, Facebook
- **Sensor Networks :** Many sensors feeding into a central controller
- **Telephone call records :** Data feeds into customer bills as well as settlements between telephone companies
- **IP packets monitored at a switch:** Gather information for optimal routing, Detect denial-of-service attacks

Applications

- Examples of data streams include
 - computer network traffic,
 - phone conversations,
 - ATM transactions,
 - web searches,
 - sensor data.
- Data stream mining can be considered a subfield of data mining, machine learning, and knowledge discovery.

Sampling from a Data Stream

- Since **we can not store the entire stream**, one obvious approach is to store a **sample**
- **Two different problems:**
 - **(1)** Sample a **fixed proportion** of elements in the stream (say 1 in 10)
 - **(2)** Maintain a **random sample of fixed size** over a potentially infinite stream
 - At any “time” k we would like a random sample of s elements
 - **What is the property of the sample we want to maintain?**
For all time steps k , each of k elements seen so far has equal prob. of being sampled

Sampling a Fixed Proportion

- **Problem 1: Sampling fixed proportion**
- **Scenario:** Search engine query stream
 - **Stream of tuples:** (user, query, time)
 - **Answer questions such as: How often did a user run the same query in a single days**
 - Have space to store $1/10^{\text{th}}$ of query stream
- **Naïve solution:**
 - Generate a random integer in $[0..9]$ for each query
 - Store the query if the integer is **0**, otherwise discard

Maintaining a fixed-size sample

- **Problem 2: Fixed-size sample**
- **Suppose we need to maintain a random sample S of size exactly s tuples**
 - E.g., main memory size constraint
- **Why?** Don't know length of stream in advance
- **Suppose at time n we have seen n items**
 - Each item is in the sample S with equal prob. s/n

How to think about the problem: say $s = 2$

Stream: a x c y z k c d e g...

At $n=5$, each of the first 5 tuples is included in the sample S with equal prob.

At $n=7$, each of the first 7 tuples is included in the sample S with equal prob.

Impractical solution would be to store all the n tuples seen so far and out of them pick s at random

Solution: Fixed Size Sample

- Problem: Maintain a uniform sample x from a stream of unknown length.
- **Algorithm (a.k.a. Reservoir Sampling)**
 - Store all the first s elements of the stream to S
 - Suppose we have seen $n-1$ elements, and now the n^{th} element arrives ($n > s$)
 - With probability s/n , keep the n^{th} element, else discard it
 - If we picked the n^{th} element, then it replaces one of the s elements in the sample S , picked uniformly at random

Solution: Fixed Size Sample

- **Algorithm (a.k.a. Reservoir Sampling)**
 - Store all the first s elements of the stream to S
 - Suppose we have seen $n-1$ elements, and now the n^{th} element arrives ($n > s$)
 - With probability s/n , keep the n^{th} element, else discard it
 - If we picked the n^{th} element, then it replaces one of the s elements in the sample S , picked uniformly at random
- **Claim:** This algorithm maintains a sample S with the desired property:
 - After n elements, the sample contains each element seen so far with probability s/n

Sliding Windows

- A useful model of stream processing is that queries are about a **window** of length N – the N most recent elements received
- **Interesting case:** N is so large that the data cannot be stored in memory, or even on disk
 - Or, there are so many streams that windows for all cannot be stored
- **Amazon example:**
 - For every product X we keep 0/1 stream of whether that product was sold in the n -th transaction
 - We want answer queries, how many times have we sold X in the last k sales

Sliding Window: 1 Stream

- Sliding window on a single stream:

N = 6

q w e r t y u i o p a s d f g h j k l z x c v b n m

q w e r t y u i o p a s d f g h j k l z x c v b n m

q w e r t y u i o p a s d f g h j k l z x c v b n m

q w e r t y u i o p a s d f g h j k l z x c v b n m

← Past Future →

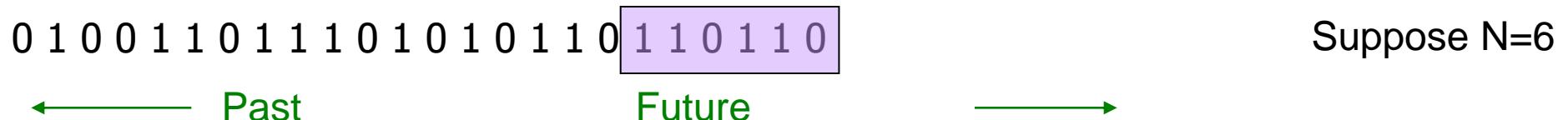
Counting Bits (1)

- **Problem:**
 - Given a stream of **0s** and **1s**
 - Be prepared to answer queries of the form
How many 1s are in the last k bits? where $k \leq N$

- **Obvious solution:**

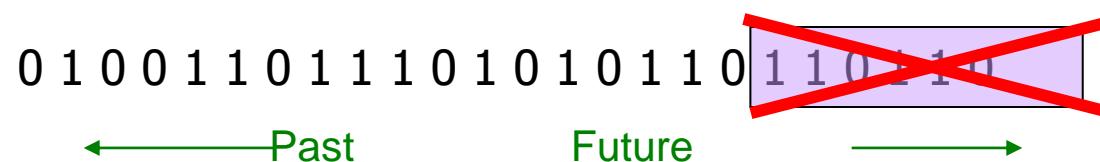
Store the most recent N bits

- When new bit comes in, discard the $N+1^{\text{st}}$ bit



Counting Bits (2)

- You can not get an exact answer without storing the entire window
- Real Problem:
What if we cannot afford to store N bits?
 - E.g., we're processing 1 billion streams and
 $N = 1$ billion



- But we are happy with an approximate answer

An attempt: Simple solution

- **Q: How many 1s are in the last N bits?**
- A simple solution that does not really solve our problem: **Uniformity assumption**



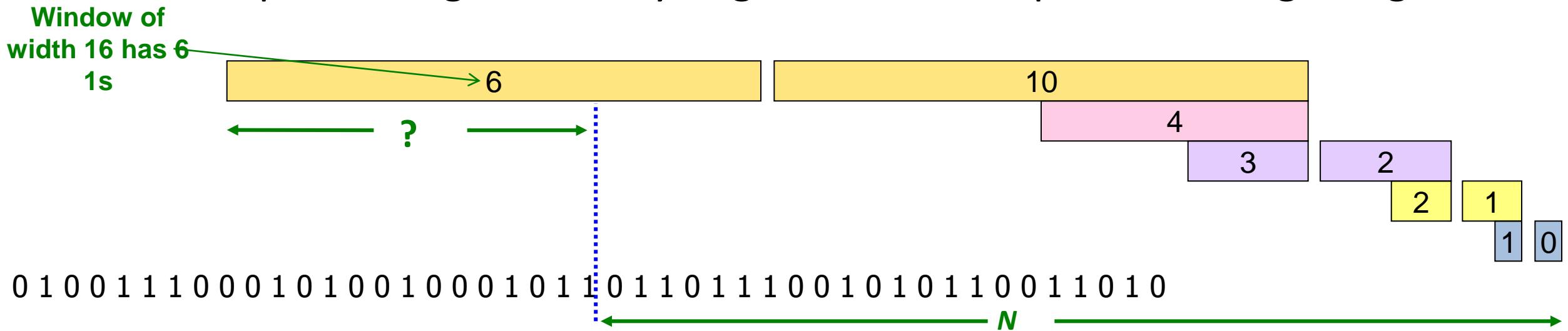
- **Maintain 2 counters:**
 - S : number of 1s from the beginning of the stream
 - Z : number of 0s from the beginning of the stream
- **How many 1s are in the last N bits?** $N \cdot \frac{S}{S+Z}$
- **But, what if stream is non-uniform?**
 - What if distribution changes over time?

DGIM Method

- **DGIM solution that does not assume uniformity**
- We store $O(\log^2 N)$ bits per stream
- **Solution gives approximate answer, never off by more than 50%**
 - Error factor can be reduced to any fraction > 0 , with more complicated algorithm and proportionally more stored bits

Idea: Exponential Windows

- Solution that doesn't (quite) work:
 - Summarize **exponentially increasing** regions of the stream, looking backward
 - Drop small regions if they begin at the same point as a larger region



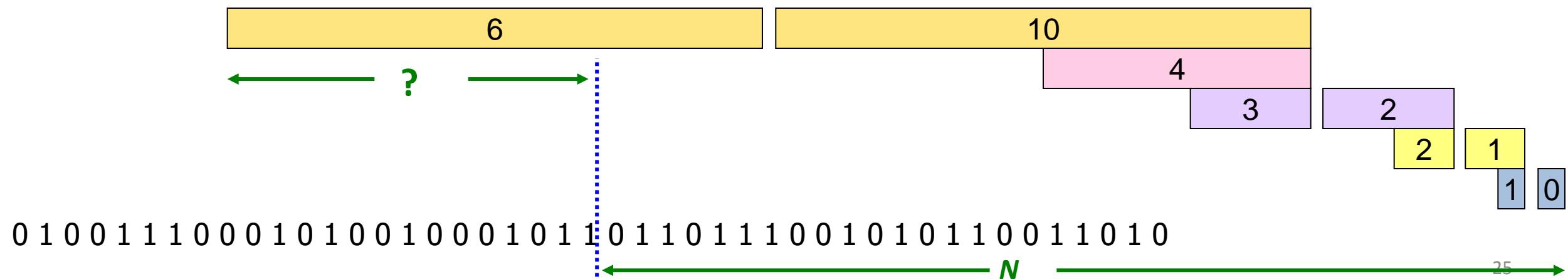
We can reconstruct the count of the last N bits, except we are not sure how many of the last 6 1s are included in the N

What's Good?

- **Stores only $O(\log^2 N)$ bits**
 - $O(\log N)$ counts of $\log_2 N$ bits each
- **Easy update as more bits enter**
- Error in count no greater than the number of **1s** in the “**unknown**” area

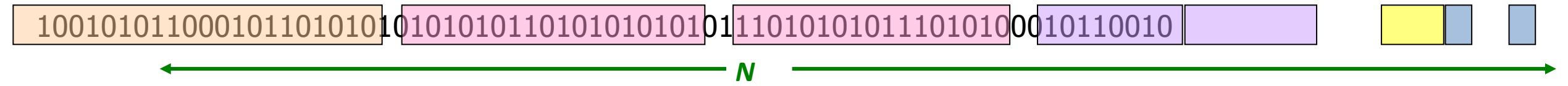
What's Not So Good?

- As long as the **1s** are fairly evenly distributed, the error due to the unknown region is small – **no more than 50%**
- But it could be that all the **1s** are in the unknown area at the end
- In that case, **the error is unbounded!**



Fixup: DGIM method

- **Idea:** Instead of summarizing fixed-length blocks, summarize blocks with specific number of **1s**:
 - Let the block *sizes* (number of 1s) increase exponentially
- **When there are few 1s in the window, block sizes stay small, so errors are small**

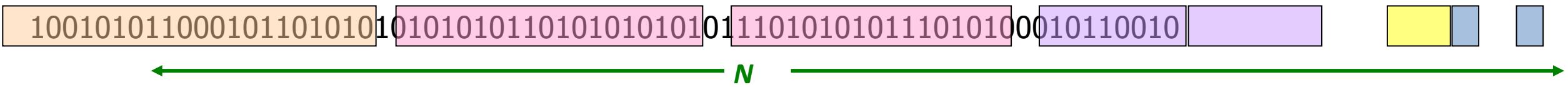


DGIM: Timestamps

- Each bit in the stream has a ***timestamp***, starting **1, 2, ...**
- Record timestamps modulo **N (the window size)**, so we can represent any **relevant** timestamp in **$O(\log_2 N)$** bits

DGIM: Buckets

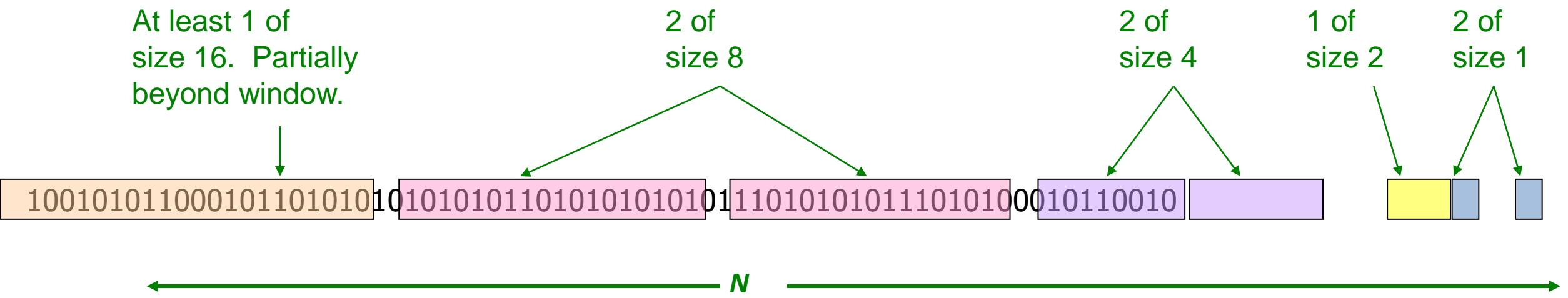
- A ***bucket*** in the DGIM method is a record consisting of:
 - (A) The timestamp of its end [$O(\log N)$ bits]
 - (B) The number of 1s between its beginning and end [$O(\log \log N)$ bits]
- **Constraint on buckets:**
Number of **1s** must be a power of **2**
 - That explains the $O(\log \log N)$ in (B) above



Representing a Stream by Buckets

- Either **one** or **two** buckets with the same **power-of-2 number of 1s**
- **Buckets do not overlap in timestamps**
- **Buckets are sorted by size**
 - Earlier buckets are not smaller than later buckets
- Buckets disappear when their end-time is $> N$ time units in the past

Example: Bucketized Stream



Three properties of buckets that are maintained:

- Either **one** or **two** buckets with the same **power-of-2** number of **1s**
 - Buckets do not overlap in timestamps
 - Buckets are sorted by size

Updating Buckets (1)

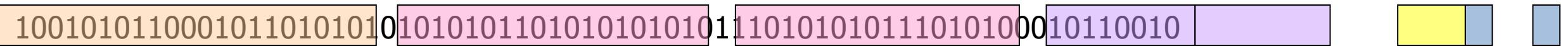
- When a new bit comes in, drop the last (oldest) bucket if its end-time is prior to N time units before the current time
- **2 cases:** Current bit is 0 or 1
- **If the current bit is 0:**
no other changes are needed

Updating Buckets (2)

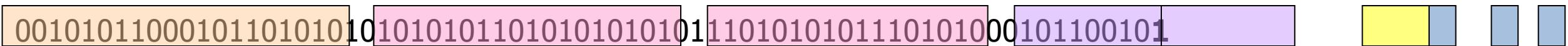
- **If the current bit is 1:**
 - (1) Create a new bucket of size **1**, for just this bit
 - End timestamp = current time
 - (2) If there are now **three buckets of size 1**, **combine the oldest two into a bucket of size 2**
 - (3) If there are now **three buckets of size 2**,
combine the oldest two into a bucket of size 4
 - (4) And so on ...

Example: Updating Buckets

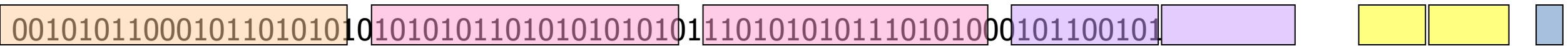
Current state of the stream:



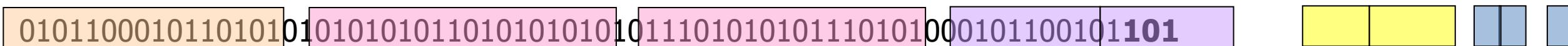
Bit of value 1 arrives



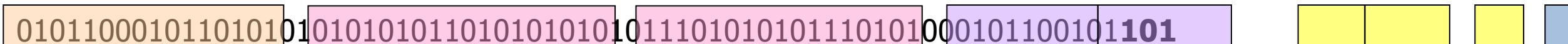
Two orange buckets get merged into a yellow bucket



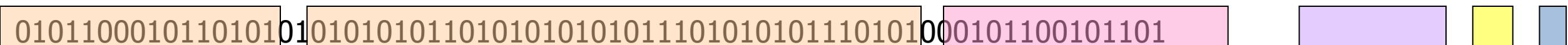
Next bit 1 arrives, new orange bucket is created, then 0 comes, then 1:



Buckets get merged...

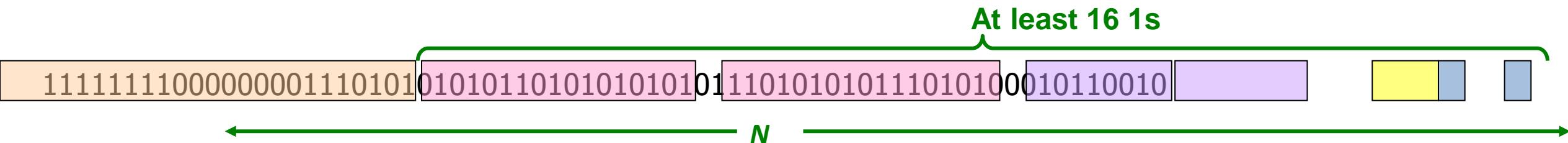


State of the buckets after merging



Error Bound: Proof

- Why is error 50%? Let's prove it!
- Suppose the last bucket has size 2^r
- Then by assuming 2^{r-1} (i.e., half) of its 1s are still within the window, we make an error of at most 2^{r-1}
- Since there is at least one bucket of each of the sizes less than 2^r , the true sum is at least
$$1 + 2 + 4 + \dots + 2^{r-1} = 2^r - 1$$
- Thus, error at most 50%



Summary

- **Sampling a fixed proportion of a stream**
 - Sample size grows as the stream grows
- **Sampling a fixed-size sample**
 - Reservoir sampling
- **Counting the number of 1s in the last N elements**
 - Exponentially increasing windows
 - Extensions:
 - Number of 1s in any last k ($k < N$) elements
 - Sums of integers in the last N elements

Mining Data Streams (Part 2)

More algorithms for streams

- **More algorithms for streams:**
 - **(1) Filtering a data stream: Bloom filters**
 - Select elements with property x from stream
 - **(2) Counting distinct elements: Flajolet-Martin**
 - Number of distinct elements in the last k elements of the stream
 - **(3) Estimating moments: AMS method**
 - Estimate std. dev. of last k elements
 - **(4) Counting frequent items**

Filtering Data Streams

- **Each element of data stream is a tuple**
- Given a list of keys S
- **Determine which tuples of stream are in S**
- **Obvious solution: Hash table**
 - But suppose we **do not have enough memory** to store all of S in a hash table
 - E.g., we might be processing millions of filters on the same stream

Applications

- **Example: Email spam filtering**
 - We know 1 billion “good” email addresses
 - If an email comes from one of these, it is **NOT** spam
- **Publish-subscribe systems**
 - You are collecting lots of messages (news articles)
 - People express interest in certain sets of keywords
 - Determine whether each message matches user’s interest

Counting Distinct Elements

Counting Distinct Elements

- **Problem:**
 - Data stream consists of a universe of elements chosen from a set of size N
 - Maintain a count of the number of distinct elements seen so far
- **Obvious approach:**

Maintain the set of elements seen so far

 - That is, keep a hash table of all the distinct elements seen so far

Applications

- **How many different words are found among the Web pages being crawled at a site?**
 - Unusually low or high numbers could indicate artificial pages (spam?)
- **How many different Web pages does each customer request in a week?**
- **How many distinct products have we sold in the last week?**

Using Small Storage

- Real problem: What if we do not have space to maintain the set of elements seen so far?
- Estimate the count in an unbiased way
- Accept that the count may have a little error, but limit the probability that the error is large

Flajolet-Martin Approach

- Pick a hash function h that maps each of the N elements to at least $\log_2 N$ bits
- For each stream element a , let $r(a)$ be the number of trailing 0s in $h(a)$
 - $r(a)$ = position of first 1 counting from the right
 - E.g., say $h(a) = 12$, then 12 is 1100 in binary, so $r(a) = 2$
- Record $R = \text{the maximum } r(a) \text{ seen}$
 - $R = \max_a r(a)$, over all the items a seen so far
- **Estimated number of distinct elements = 2^R**

Thank you.

4:11 M 🔍 78
ips → 87 ... 92 ... 95 ... 98 ... 100 ... 101 ... 102 ... 103
New bits

new bit = 0 enters:

... 92 ... 95 ... 98 ... 100 ... 101 ... 102 ... 103 ... 104
000 [10111] 0 [11] 00 [101] [11] 0 0 ← 1, 1 1

new bit = 1 enters:

92 95 98 100 101 102 103 104
11 000 [10111] 0 [11] 00 [101] [1] 0 0 [11] ← 1

then next bit = 1 enters:

87 92 95 98 100 101 102 103 104
01011 000 [10111] 0 [11] 00 [10] [1] 0 0 [1] ← 1

87 92 95 98 100 101 102 103 104
01011 000 [10111] 0 [11] 00 [101] [1] 0 0 1 [1] ← 1

87 92 95 98 100 101 102 103 104
01011 000 [10111] 0 [11] 00 [101] [1] 0 0 1 [1] ← 1
 2^2 2^2 2^2 2^2 2^1 2^1 2^0

If it is greater or equal to

Finally answer to query

- How many 1's are there in the last 20 bits?
No. of 1's in the last 20 bits = 11

Made with KINEMASTER

6G VoLTE 78

3:59

DGIM ALGORITHM

- DGIM algorithm (Datar - Gionis - Indyk - Motwani Algorithm)
- Designed to find number of 1's in a data set.
- This algorithm uses $O(\log^2 N)$ bits to represent a window of N bit.
- It allows to estimate the number of 1's in the window with an error of no more than 50%.

Components of DGIM algorithm:

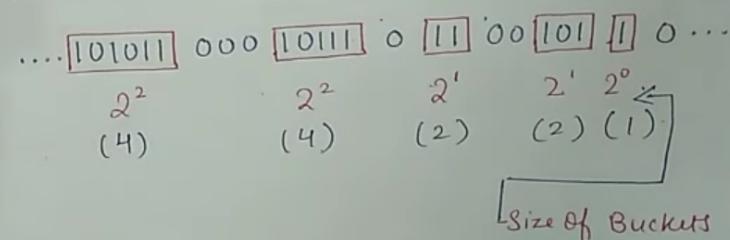
- Timestamp
- Buckets
- Each bit that arrives has a timestamp, for the position at which it arrives.
- If the first bit has a timestamp 1, the second bit has a timestamp 2 & so on... the positions are recognized with the window size N (which are usually taken as multiple of 2).
- The windows are divided into buckets consisting of 1's and 0's.

* Rules for Forming the buckets

- 1) The right side of the bucket should always start with 1. (if it starts with a 0, it is to be neglected) Eg: 1001011 → a bucket of size 4, having four 1's and starting with 1 on its right end.
- 2) Every bucket should have atleast one 1, else no bucket can be formed.
- 3) All buckets should be in powers of 2.
- 4) The buckets cannot decrease in size as we move to the left. (move in increasing order towards left).

Example

...1010110001011010010110...
 $N = 24$ (window size)



- In the given data stream, let us assume the new bit arrives from the right

If New bit = 0 \Rightarrow No change in buckets, as shown below.

timestamps \rightarrow 87 ... 92 ... 95 ... 98 ... 100 ... 101 ... 102 ... 103
 New bits
 $\boxed{101011} \ 000 \ \boxed{10111} \ 0 \ \boxed{11} \ 00 \ \boxed{101} \ \boxed{11} \ 0 \leftarrow 0 \ 1 \ 1$

When new bit = 0 enters:

87 ... 92 ... 95 ... 98 ... 100 ... 101 ... 102 ... 103 ... 104
 $\boxed{101011} \ 000 \ \boxed{10111} \ 0 \ \boxed{11} \ 00 \ \boxed{101} \ \boxed{11} \ 0 \ 0 \leftarrow 1 \ 1 \ 1$

When new bit = 1 enters:

87 92 95 98 100 101 102 103 104
 $\boxed{101011} \ 000 \ \boxed{10111} \ 0 \ \boxed{11} \ 00 \ \boxed{101} \ \boxed{1} \ 0 \ 0 \ \boxed{11} \leftarrow 1$

When next bit = 1 enters:

87 92 95 98 100 101 102 103 104
 $\boxed{101011} \ 000 \ \boxed{10111} \ 0 \ \boxed{11} \ 00 \ \boxed{101} \ \boxed{1} \ 0 \ 0 \ \boxed{11} \ \boxed{1} \leftarrow 1$

87 92 $\boxed{95} \ 98$ 100 101 102 103 104
 $\boxed{101011} \ 000 \ \boxed{10111} \ 0 \ \boxed{11} \ 00 \ \boxed{101} \ \boxed{1} \ 0 \ 0 \ 1 \ \boxed{1} \leftarrow 1$

87 92 $\boxed{98}$ 100 102 103 104
 $\boxed{101011} \ 000 \ \boxed{10111} \ 0 \ \boxed{11} \ 00 \ \boxed{101} \ \boxed{100} \ \boxed{1} \leftarrow 1$

How long can you continue doing this?

- You can continue if Made with KINEMASTER bucket timestamp of window $< N$ ($= 24$ here).
- Eg: $103 - 87 = 16 < 24$ so I continue.
 If it is greater or equal to then I stop.

Finally answer to query

- How many 1's are there in the last 20 bits?

1	2	3	10	11	12	13	14
4	5	6	17	18	19	20	21
7	8	9	23	24	25	26	27
10	11	12	28				
13	14	15					
16	17	18					
19	20	21					
22	23	24					
25	26	27					
28	29	30					

WR 40 • 275-090

02

OCTOBER

SATURDAY

Reservoir Sampling

- How to pick a random elements of an array?
- Ans: choose a random integer and pick the element at that index.
- How to pick 'S' random elements from an infinite stream of elements?
- Maintain a set of 'S' elements that may be updated every time a new element is added to the stream.
- At any given point, each element that has been seen so far should be equally likely to be in the set of S elements.

↓ How?

- Save the first S elements

03

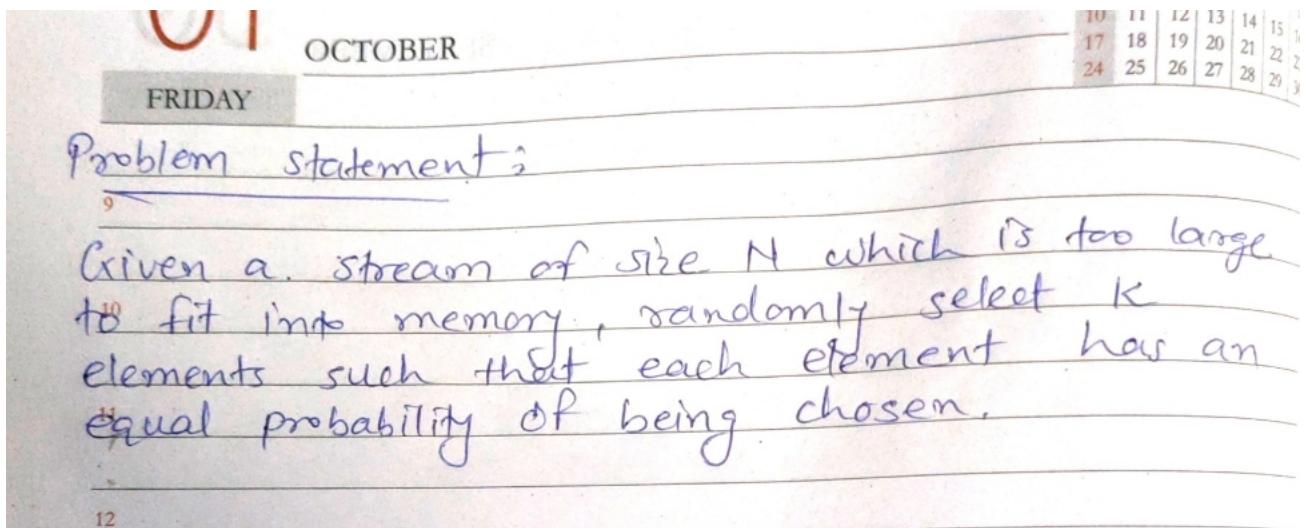
- Suppose we have seen the first $n-1$ elements and now n^{th} element arrives.

- with probability S/n , keep n^{th} element and replace a randomly chosen saved element with the n^{th} element.
- otherwise discard the n^{th} element

2021 Induction :- After n elements, the sample contains each element seen so far with probability S/n

OCTOBER

SUNDAY



08

281-084 • WK 41

OCTOBER

FRIDAY

3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30

Bloom filter

Ex: Create gmail account - Enter user name - everything
User name already exist)
↓
all procedure done quickly
to check ↓
↓ { linear search ?
Binary search ?
Time consuming process
↓ use
Bloom filter

- Bloom filter is a space efficient probabilistic data structure that is used to test whether an element is a member of a set.

for ex checking availability of user name. It is a set membership problem, where the set is a list of all registered user name

- The problem with the bloom filter is that they are probabilistic in nature that means, there might be some false positive results.

False Positive: It might tell that given user name is already taken but actually it's not.

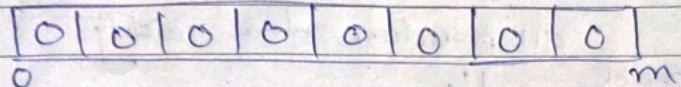
2021

15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30					

THURSDAY

Working of Bloom filter

- An empty bloom filter is a bit array of m bits. all set to 'zero'. like this -



- we need ' k ' number of hash function to calculate the hashes for a given input.

- when we want to add an item in the filter, the bits at k indices $h_1(x), h_2(x), \dots, h_k(x)$ are set, where indices are calculated using hash functions.

Example

- Suppose we want to enter ''throw'' in the filter, we are using 3 hash functions & a bit array of length 10, all set to 0 initially.

- First we will calculate the hours as following suppose,

h_1 ('throws') $\approx 10 = 1$

$$\text{he ('throw')} \cdot 1 \cdot 10 = 4$$

$$h_3('thaw') + 10 = 7$$

WEDNESDAY

17	18	19	20	21	22	23
24	25	26	27	28	29	30

Now, we will set the bits at indices 1, 4, 7 & 9 to 1.

9

0	1	0	0	1	0	1	0	0
0	1	2	3	4	5	6	7	8

10

11

Again we want to enter ''catch'', similarly we will calculate hashes

12

$$h_1('catch') \cdot 10 = 3$$

$$h_2('catch') \cdot 10 = 5$$

$$h_3('catch') \cdot 1 \cdot 10 = 4$$

13

set the bits at indices 3, 5 & 4 to 1.

14

0	1	0	1	1	1	0	1	0	0
0	1	2	3	4	5	6	7	8	9

Now, if we want to check ''throw'' is present in filter or not. we will do the same process but in reverse order. calculating respective hashes using h_1, h_2 & h_3 & check if all indices set to 1 in the bit array

- If all bits are set then we can say that ''throw'' is 'probably present'.

- If any of the bit at these indices are 0 then ''throw'' is definitely not present'.

2021

TUESDAY

false positive in Bloom filters:

The question is why we said 'Probably present'
why this uncertainty.

+ let's take an example:

suppose we want to check whether 'cat' is present or not. we will calculate hashes using $h_1, h_2 \& h_3$.

check array

$$h_1('cat') \cdot 10 = 1 \quad \text{set}$$

$$h_2('cat') \cdot 10 = 3 \quad \text{set}$$

$$h_3('cat') \cdot 10 = 7 \quad \text{set}$$

If we check the bit array, bits at these indices are set to 1 but we know that 'cat' was never added to the filter. Bit at index 1, $\&$ 7 was set when we added 'throw'
 $\&$ bit 3 was set when we added 'catch'.

Control probability of getting False Positive by controlling size of Bloom filter.

↓ probability of false positive \Rightarrow ↑ no. of hash functions.

MONDAY

24 25 26 27 28 29 30

probability of false positive

$$P = \left(1 - \left[1 - \frac{1}{m} \right]^{kn} \right)^k$$

where, m = size of bit array n = no. of expected elements to be inserted k = no. of hash functions.

12

$$\text{Size of Bit array} = m = \frac{-n \log P}{(\log 2)^2}$$

$$\text{Optimal no. of hash functions : } k = \frac{m \log 2}{n}$$

Interesting properties of bloom filters

- Bloom filters never generate false negative result i.e. telling you that a username does not exist when it actually exists.
- Deleting elements from filter is not possible because, if we delete a single element by clearing bits at indices generated by k hash functions, it might cause deletion of few other elements.

1	2	3	10	11	12	13	14
8	9	17	18	19	20	21	
15	16	24	25	26	27	28	
22	23						
29	30						

TUESDAY

Counting Distinct Elements in a Stream

Flajolet - Martin Algorithm (FM Algorithm)

- FM algorithm approximates the no. of unique objects in a stream or a database in one pass.
- If the stream contains n elements with m of them unique ; this algorithm runs in $O(n)$ time & needs $O(\log m)$ memory.
- The FM algorithm is an algorithm for approximating the no. of distinct elements in a stream with a single pass.
- Space consumption logarithmic in the maximal no. of possible distinct elements in the stream.

Example

- Determine the distinct elements in the stream using FM algorithm.

Input stream of integers $x = 1, 3, 2, 1, 2, 3, 4, 3, 1, 2, 3, 1$

Hash function, $h(x) = 6x + 1 \bmod 5$

2021

MONDAY

17	18	19	20	21	22	23
24	25	26	27	28	29	30

Step-1 : calculating Hash function [h(x)]

$$h(x) = 6x + 1 \bmod 5$$

$$\begin{aligned} h(1) &= 6(1) + 1 \bmod 5 \\ &= 7 \bmod 5 \end{aligned}$$

$$h(1) = 2$$

similarly, calculating hash function

for the remaining
input stream.

$$h(1) = 2$$

$$h(4) = 0$$

$$h(3) = 4$$

$$h(3) = 4$$

$$h(2) = 3$$

$$h(1) = 2$$

$$h(1) = 2$$

$$h(2) = 3$$

$$h(2) = 3$$

$$h(3) = 3$$

$$h(3) = 4$$

$$h(1) = 2$$

$$h(3) = 4$$

$$h(1) = 2$$

$$h(3) = 4$$

$$h(1) = 2$$

Step-2 : Write Binary Equivalent for hash function

⁴ calculated

$$h(1) = 2 = 010$$

$$h(4) = 0 = 000$$

$$h(3) = 4 = 100$$

$$h(3) = 4 = 100$$

$$h(2) = 3 = 011$$

$$h(1) = 2 = 010$$

$$h(1) = 2 = 010$$

$$h(2) = 3 = 011$$

$$h(2) = 3 = 011$$

$$h(3) = 3 = 011$$

$$h(3) = 4 = 100$$

$$h(1) = 2 = 010$$

Step-3 : Finding Trailing zeros

Now, write the count of trailing zeros in each hash function bit

2021

1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30					

OCTOBER

Not

trailing zeros - if
bit
change
then
trailing
zero
consider

SATURDAY

$$h(1) = 2 = 010 = 1$$

$$h(4) = 0 = 000 = 0$$

$$h(3) = 4 = \underline{100} = 2$$

$$h(3) = 4 = 100 = 2$$

$$h(2) = 3 = 011 = 0$$

$$h(1) = 2 = 010 = 1$$

$$h(1) = 2 = 010 = 1$$

$$h(2) = 3 = 011 = 0$$

$$h(2) = 3 = 011 = 0$$

$$h(3) = 4 = 100 = 2$$

$$h(3) = 4 = 100 = 2$$

$$h(1) = 2 = 010 = 1$$

Step-4

write the value of maximum no. of trailing zeros.

- The value of $\sigma = 2$

- The distinct value $R = 2^{\sigma}$
 $= 2^2$
 $\boxed{= 4}$

Hence, there are 4 distinct elements 1, 3, 2, 4

10

SUNDAY

PDF Created Using



Camera Scanner

Easily Scan documents & Generate PDF



<https://play.google.com/store/apps/details?id=photo.pdf.maker>

Data Visualization: Some basics

Graphical Options

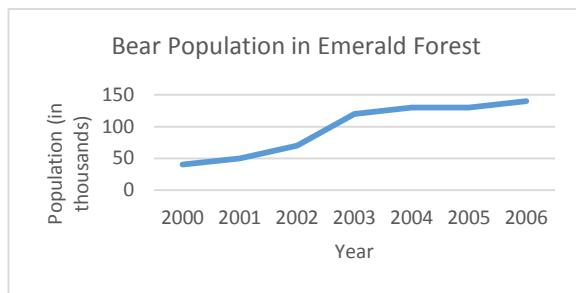
You have many graphical options but certain graphs might be more appropriate depending on the story you want to tell. Here are the most common types of graphs:

Table 1: Types of graphs and their uses

Type of Graph	Uses
Line graph	demonstrating change over time or comparing change over time
Bar chart	comparing differences / similarities between groups
Table	providing exact values
Scatter plot	showing correlation (positive, negative, none) between two variables
Pie chart	illustrating large differences in proportions for simple data

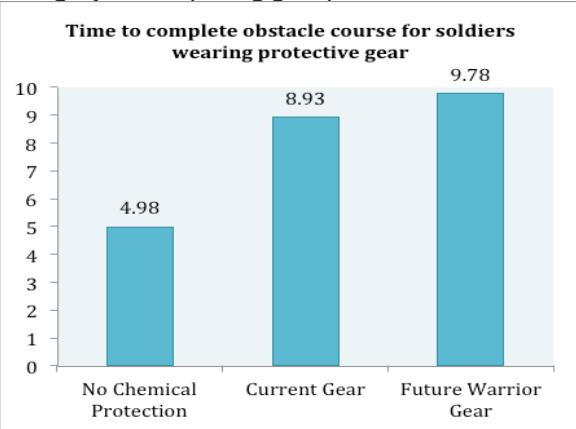
Here are the common types of data visualization with examples and described in more detail:

Line graph: changes over time



Since the unit for the x-axis is Years, we can easily see the growth of the bear population (the y-axis unit) over time by following the line connecting the data points.

Bar graph: comparing groups



A quick glance at this graph shows a large difference between the groups wearing no chemical protection and the groups wearing either the current gear or the future warrior protective gear.

Notice how the gridlines have been removed to eliminate “non data ink” and create a clean look.

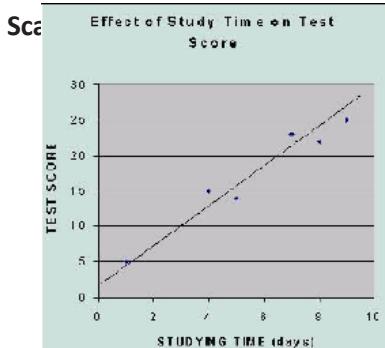
Table: presenting exact values

Table 1: Average time to perform task (in seconds)		
	<i>Firefox 2.0</i>	<i>IE 7.0</i>
Startup speed		
Time to start up from cold	11.6	7.8
Download speeds		
Download benchmark HTML page w/ multiple images	2.0	2.5
Download benchmark Javascript	22.0	36.4

We can easily compare the startup speed between Firefox and IE by glancing at the actual values and knowing that 11.6 is higher than 7.8. It is also useful that we can see not only startup speed in this table, but also download speeds.

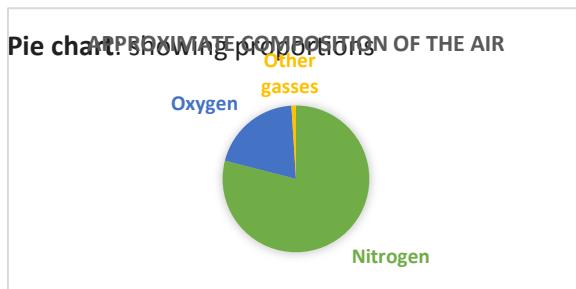
Again, notice how an absence of gridlines simplifies this graph.

Data Visualization: Some basics



In a scatter plot, we can see the spread (i.e., a visualization for how correlated the variables on our axes are); since the points are tight around the line, the correlation is strong. We can also see the general trend that we expect higher test scores for longer studying times.

*It is up to you whether you want to add the trend line to your visualization. Some sources recommend avoiding extra clutter; some prefer the ease of access to the information.



Pie charts can illustrate noticeable differences between a few groups. **USE THIS TYPE OF CHART WITH CAUTION.** It is much easier to distinguish differences in data with a bar graph.

Choosing your story

The graphical representation you choose for your data will depend on the story you want to tell (2).

For example, consider the following data tables. Depending on the way you compile the data, emphasizing total medal count or gold medal count, you could make an argument for whether the US or China “won” the Olympics. Neither interpretation is “wrong” -- they simply create different arguments. First

Table 2: 2008 Summer Olympic medals by country. These two tables differ by how the data is sorted but tell two very different stories about who lead the 2008 Olympics.

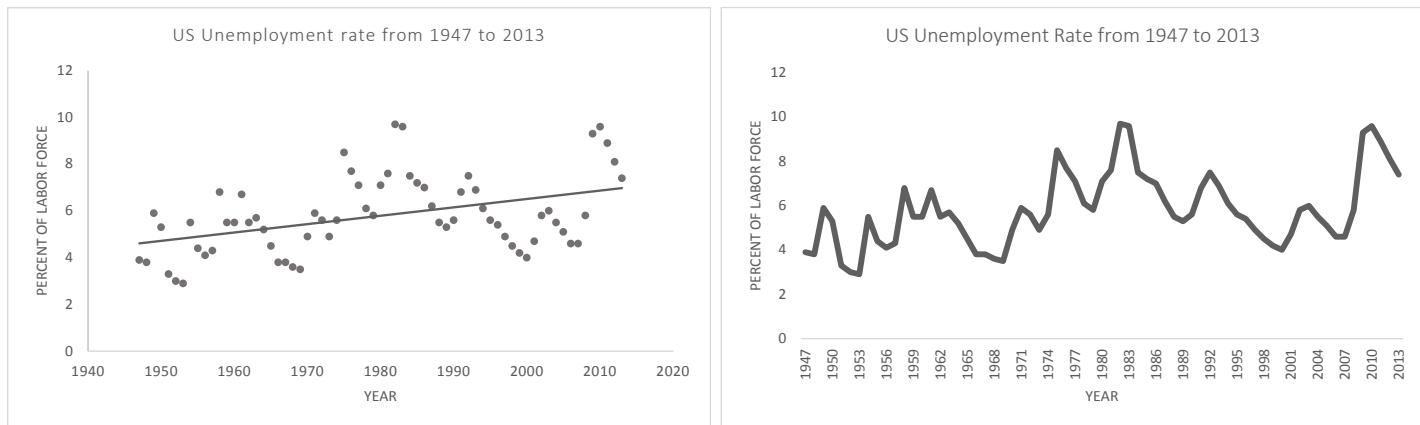
a) Sorted by total medals				
Country	Gold	Silver	Bronze	Total
USA	36	38	36	110
China	51	21	28	100
Russia	23	21	29	73
Britain	19	13	15	47
Australia	14	15	17	46
Germany	16	10	15	41
France	7	16	18	41

b) Sorted by gold medals				
Country	Gold	Silver	Bronze	Total
China	51	21	28	100
USA	36	38	36	110
Russia	23	21	29	73
Britain	19	13	15	47
Germany	16	10	15	41
Australia	14	15	17	46
S. Korea	13	10	8	31

Data Visualization: Some basics

consider your narrative -- then decide how your data visualization can help support it.

Again, consider the below graphs that tell very different stories about the US employment rate over the



Explaining the Data Visualization

Finally, while your data visualize should be clear enough that someone could just look at it and understand the story, you still need to explain the bottom line. You should do this 1)in a caption/title and 2)in the text of your paper.

In a caption OR title, you explain your the basic elements of your visualization:

1. Name the figure or table (Table 1, Figure 1, etc.)
2. Describe the visualization
 - What are the axes
 - What does the data represent

Within the text, you should interpret your visualization:

1. Refer to your table or figure and state the main trend
2. Support the trend with data
3. (if needed) Note any additional trends and support with data
4. (if needed) Note any exceptions to the main trends or unexpected outcomes

For example, consider the interpretation for the table below on internet browser performance.

Average browser performance on speed, memory use, and compliance				
Task	Speed (in seconds)	Memory use (in megabytes)	Compliance (in percent)	JavaScript score
Chrome 31	4.5	261	75%	213.9
Firefox 25	5.3	106	68%	183.1
IE 11	2.3	162	64%	165.3
Opera 17	5.3	200	73%	200.0
Safari 5.1	4.5	130	51%	137.6

Speed was calculated by taking average cold start up, non-cold start up, non-cached page load, and cached page reload speeds. Memory use was calculated by taking average memory for blank tab and 10 open tabs. Compliance was calculated by averaging HTML 5 and CSS 3 compliance rates. See the full report for an explanation of how JavaScript score was calculated

Table 1 suggests that Chrome is the best internet browser as it scored the best on two of the four measures. However, we should note that Chrome had the worst score on memory use. Therefore, we conclude that Chrome is the best internet browser for users whose computers have enough memory to handle it. Users whose computers have lower memory might prefer Firefox 25, while users who want the fastest browser could use IE 11.

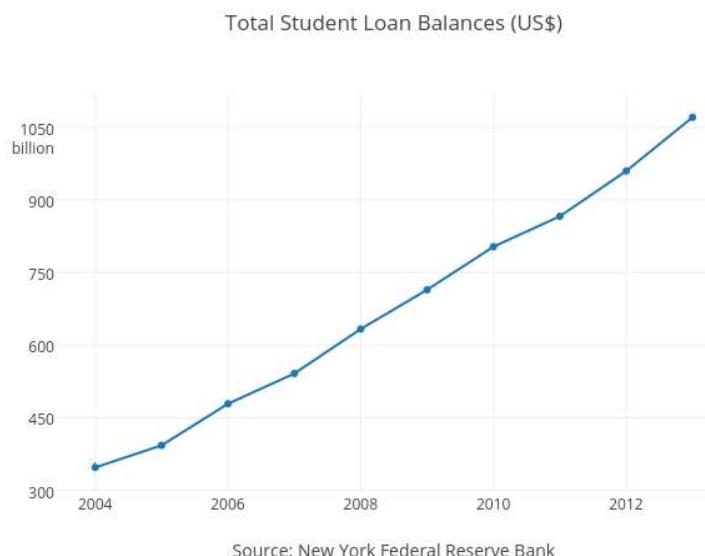
- <https://medium.com/@plotlygraphs/how-to-analyze-data-eight-useful-ways-you-can-make-graphs-328b81c2fa13>
- <https://towardsdatascience.com/10-viz-every-ds-should-know-4e4118f26fc3>
- <https://www150.statcan.gc.ca/n1/edu/power-pouvoir/ch9/pie-secteurs/5214826-eng.htm>
- <https://www.thoughtco.com/what-are-time-series-graphs-3126233>
- <https://towardsdatascience.com/the-complete-guide-to-time-series-analysis-and-forecasting-70d476bfe775> (Time series graph)

In previous section we discussed Two important chart, Bar Chart and Pie Chart. In this session we shall discuss other two charts

- **Line Chart**
- **Area chart**
- **Time series line chart**

I. Line chart

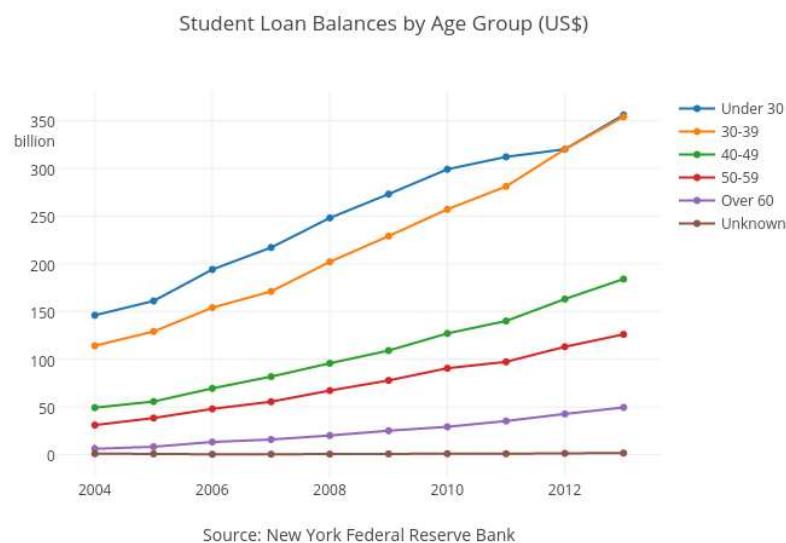
Line charts are similar to bar charts in that they compare values, but the x-axis displays continuous, rather than discrete, data. Each data point has an x and y value and is connected by a line. This basic line chart shows the total American student **loan debt** over time, with the year along the x-axis, and the amount of debt along the y-axis. Line charts emphasize the overall trend or pattern of the data. (You can draw Bar Chart for the same line chart)



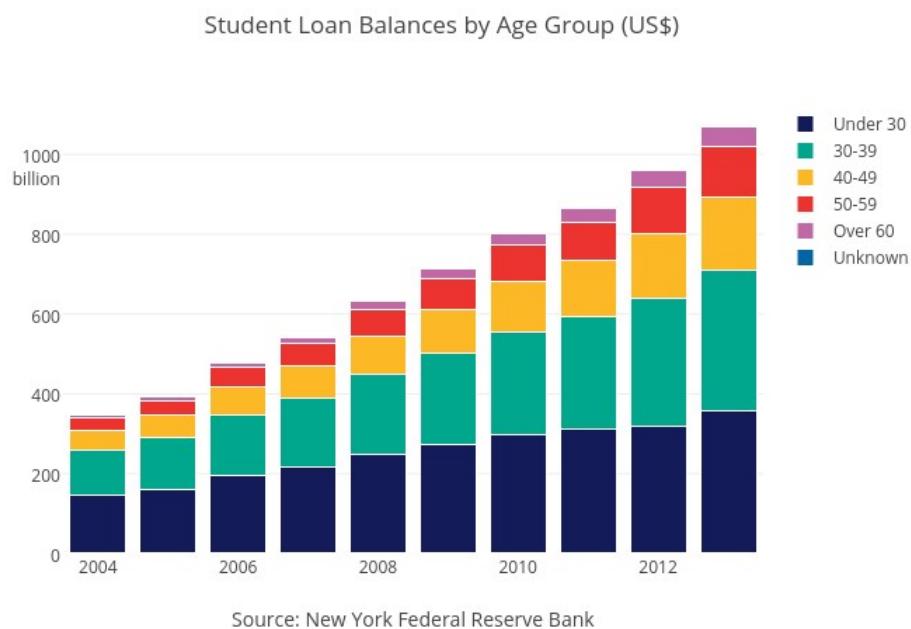
Variation in Line Plotting:

For Better visualization and understanding of Data, the simple line plot can be redrawn

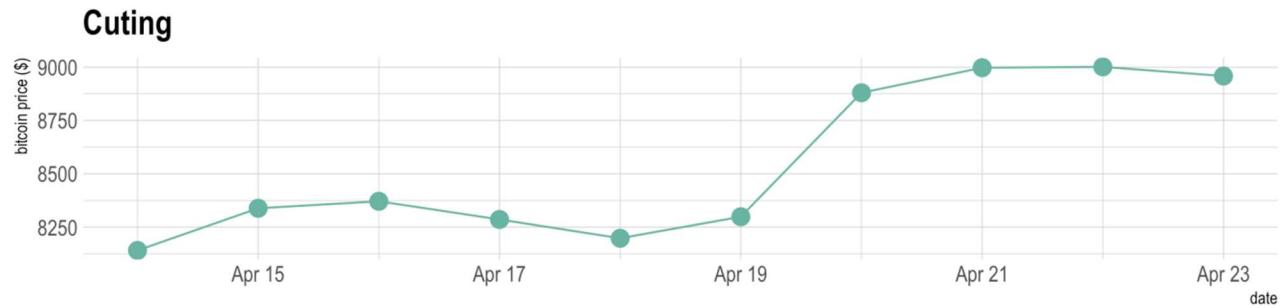
- I. You can compare multiple traces on a line chart. Following graph is made from the same data, but broken down into age group.



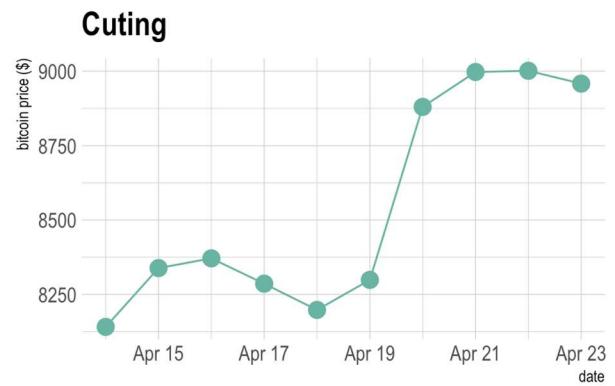
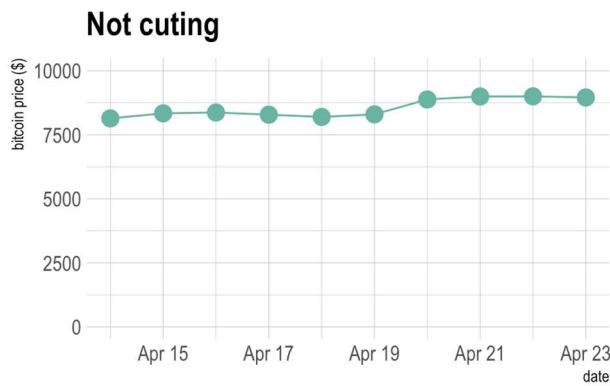
The same data is plotted as a bar chart in next figure. This is a more effective example of a stacked bar chart, as the total debt for all age groups can be compared between years.



- II. If the number of data points is low, it is advised to represent each individual observation with a dot. It allows to understand when exactly the observation have been made:



- III. Cut the Y axis: Whether or not the Y axis must start at 0 is a hot topic leading to intense debates. The graphic below presents the same data, starting at 0 (left) or not (right). Generally, line plot do not need to start at 0 since it allows to observe patterns more efficiently.

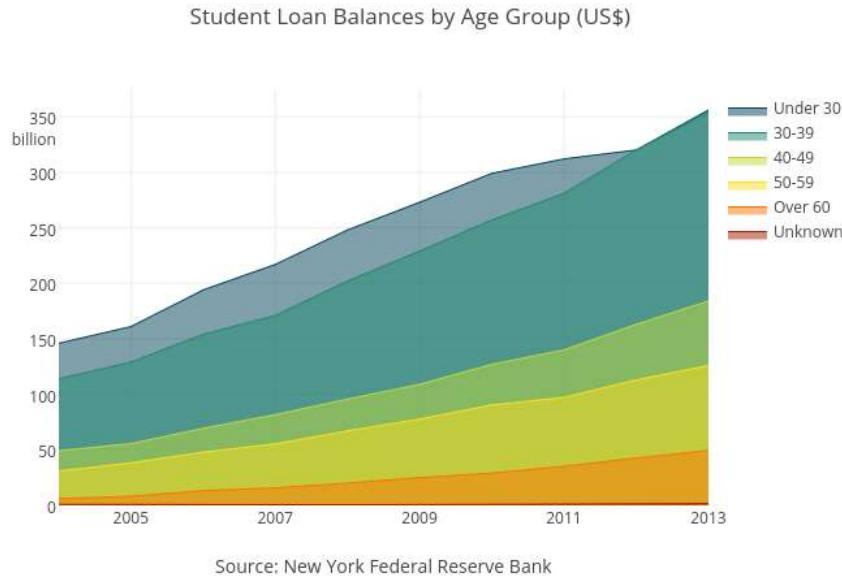


Note :

- It can be tricky to choose between a line and bar chart **when comparing values over time, as time** (Time Chart, Next section) can be represented as **continuous or discrete**.
- The stacked bar chart makes it easier to see the total amount of debt, while the line chart shows the trend for each age group separately.
- Line charts excel when the changes are more subtle or vary a lot over time.
- If you need to compare the evolution of 2 different variables, do not use dual axis. Indeed dual axis can show very different results depending on what range you apply to the axis. Read more about it.
- Mind the **spaghetti chart**: too many lines make the chart unreadable.
- Think about the aspect ratio of the graphic, extreme ratio make the chart unreadable.

II. Area Chart

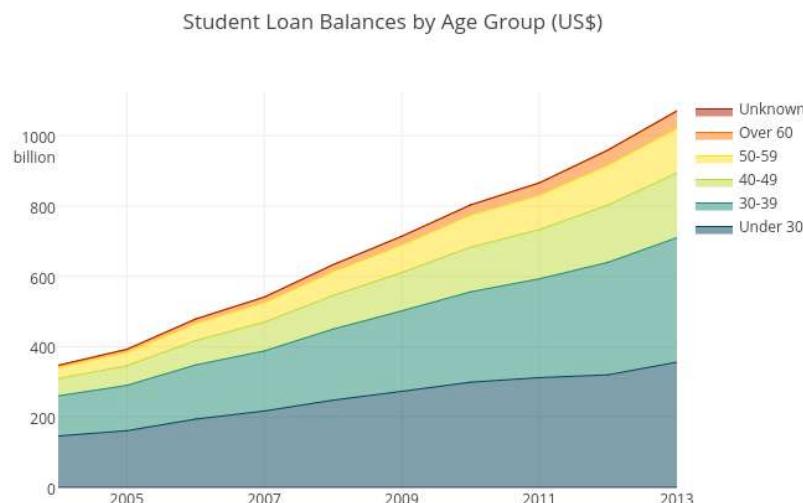
Area charts are very similar to line charts, but the area under the line is filled in. This simple visual difference can be an effective way of drawing attention to the magnitude of difference between traces, or the cumulative value over a period of time. However, it can be confusing if there are many overlapping traces.



Here, all the information from the multiple line chart of previous section is preserved, but use of color draws attention to the volume of debt for each group. In this overlaid area graph, the areas are translucent and placed in front of each other.

Variation:

A stacked area graph adds each trace on top of the last one, such that the areas will never overlap.

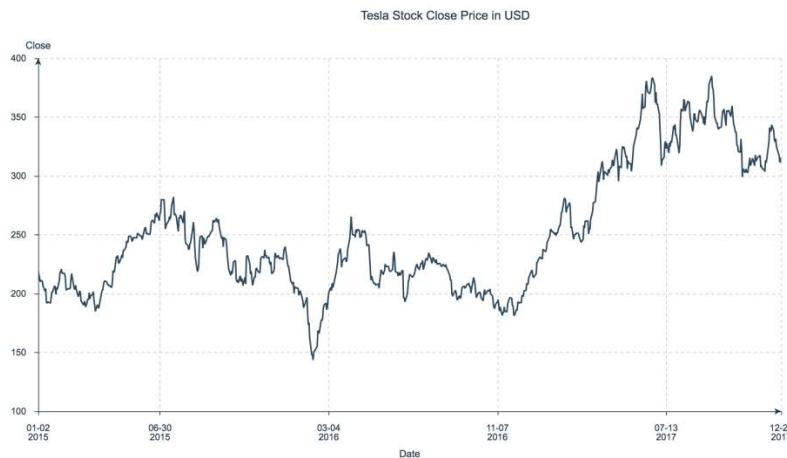


Note :

- Stacked area graphs have the benefit of showing the total of all the groups, but make it difficult to see values and patterns for the individual traces. Switch to stacked area charts by making sure your data is cumulative and then hit ‘fill to next y’ on the mode tab under traces. Fill to Y=0 makes an overlaid area chart.
- stacked area graph are appropriate to study the evolution of the whole and the relative proportions of each group. Indeed, the top of the areas allows to visualize how the whole behaves, like for a classic area chart.
- however they are not appropriate to study the evolution of each individual group: it is very hard to subtract the height of other groups at each time point. For a more accurate but less attractive figure, consider a line chart or area chart using small multiple.

III. Time Series chart (time Plot/Time Series Graph)

One feature of data that you may want to consider is that of time. A graph that recognizes this ordering and displays the change of the values of a variable as time progresses is called a time series graph. It is simply a scatter plot or a line plot with a time range on the x-axis where each dot forms part of a line. Important to note that time is continuous, so this is a continuous graph (Not discrete).



Time Series Plot of Tesla Stock Close Price from 2015–2017

Time series plots are great for visually investigating trends, jumps and drops in data over time, which makes them especially popular for financial and sensor data.

E.g. Suppose that you want to study the climate of a region for an entire month. Every day at noon you note the temperature and write this down in a log. A variety of statistical studies could be done with this data. You could find the mean or the median temperature for the month. You could construct a histogram displaying the number of days that temperatures reach a certain range of values.

But all of these methods ignore a portion of the data that you have collected. Since each date is paired with the temperature reading for the day, you don't have to think of the data as being random. You can instead use the times given to impose a chronological order on the data.

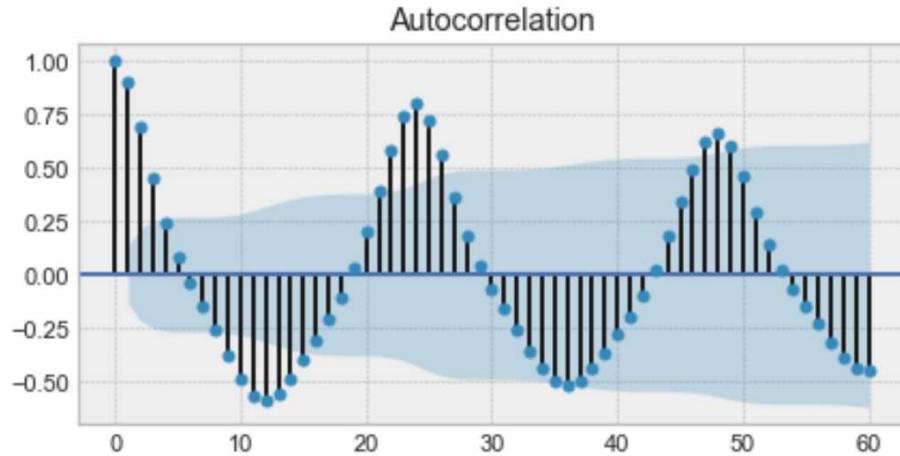
Application of a Time Series chart

When recording values of the same variable over an extended period of time, sometimes it is difficult to discern any trend or pattern.

1. Trend Analysis: Once the same data points are displayed graphically, some frequent patterns jump out. Time series graphs make trends easy to spot. These trends are important as they can be used to project into the future.
2. Pattern Analysis: It is important to model the business and identify cyclical patterns if any exists. The regularly increase/Decrease or up/down in the parameter with respect to time denotes possible cycle

which can be identified from the graph. **This may lead to identify and predict trend. Statistical Characteristics of TS graphs**

3. **Auto correlation:** It is the similarity between observations as a function of the time lag between them.

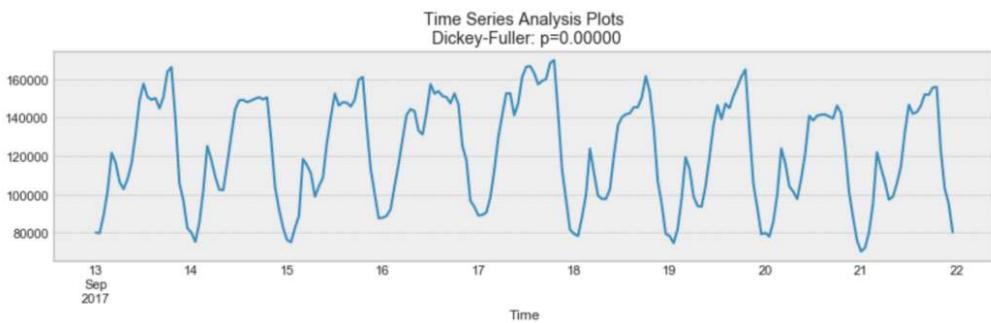


Example of an autocorrelation plot

From above graph we can realize that the first value and the 24th value have a high auto correlation. Similarly, the 12th and 36th observations are highly correlated. This means that we will find a very similar value at every 24 unit of time.

Notice how the plot looks like sinusoidal function. This is a hint for **seasonality**, and you can find its value by finding the period in the plot above, **which would give 24h**.

4. **Seasonality :** It refers to periodic fluctuations. For example, electricity consumption is high during the day and low during night, or online sales increase during Christmas before slowing down again.

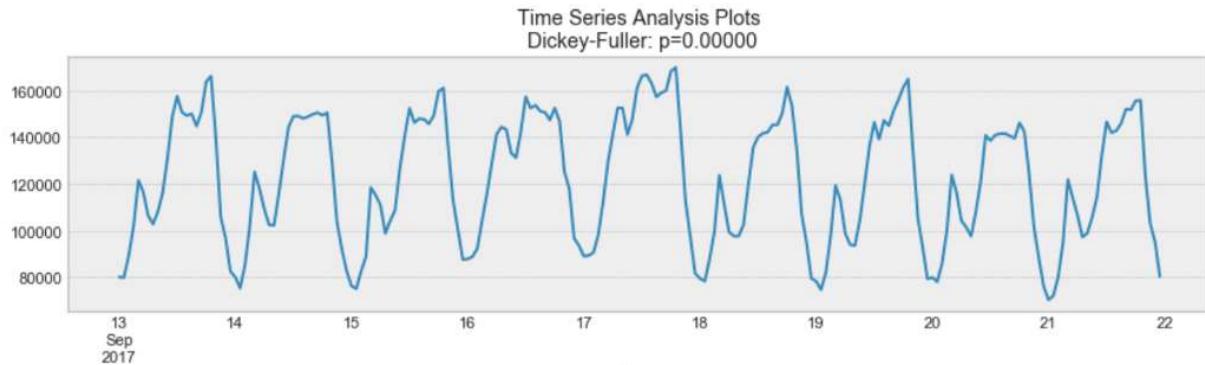


Example of seasonality

As shown in above graph every day, you see a peak towards the evening, and the lowest points are the beginning and the end of each day which shows daily seasonality.

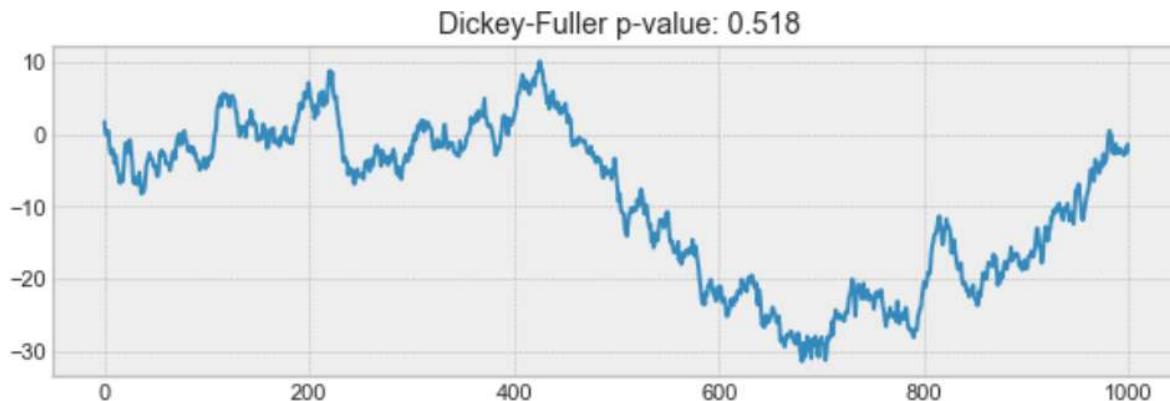
Remember that seasonality can also be derived from an autocorrelation plot if it has a sinusoidal shape. Simply look at the period, and it gives the length of the season.

- 5. Stationarity:** it is an important characteristic of time series. A time series is said to be stationary if its statistical properties do not change over time. **In other words, it has constant mean and variance, and covariance is independent of time.**



Example of a stationary process

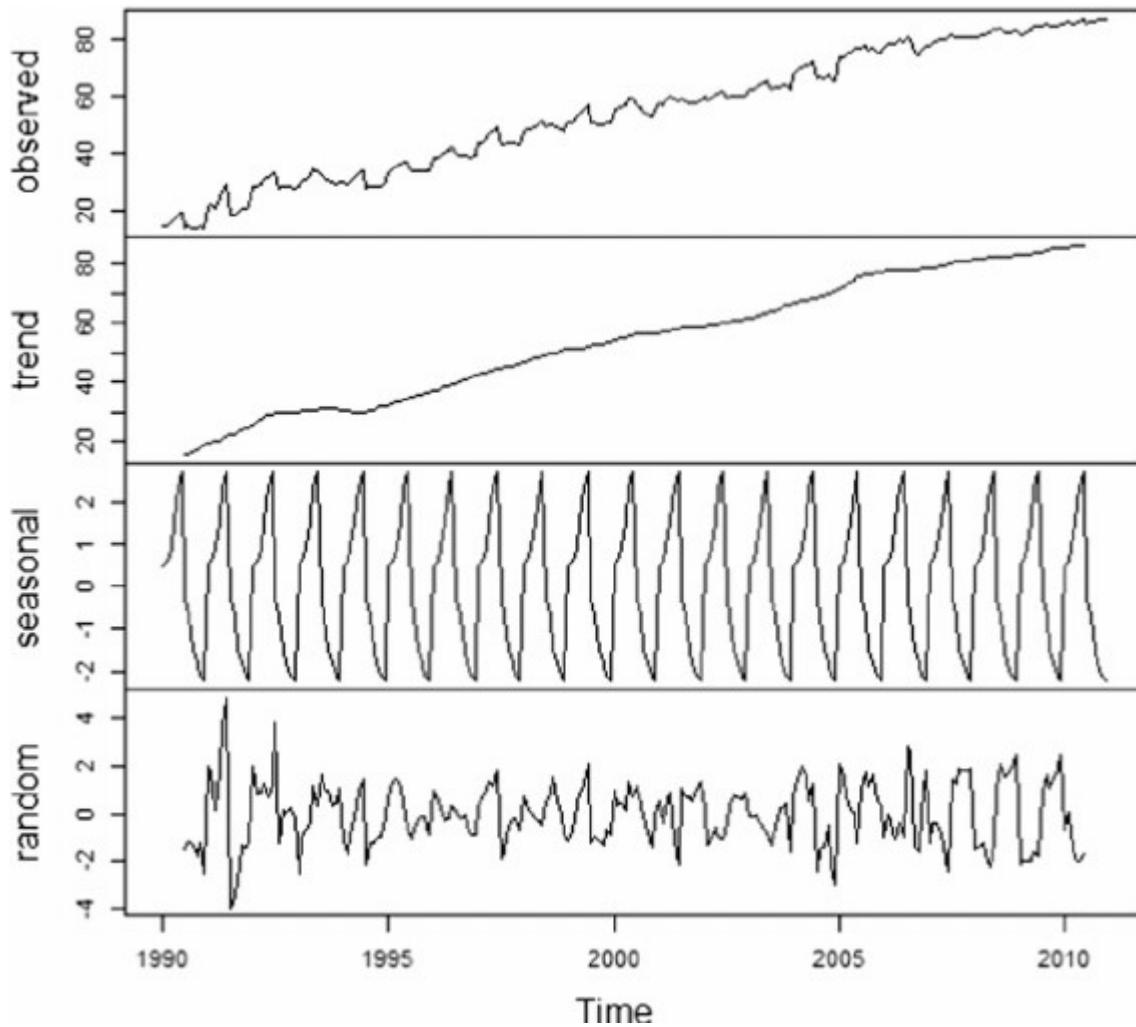
The above graph show that the process for which data is plotted is stationary. The mean and variance do not vary over time. At the same time the following graph is not containing constant mean and variance and so it is not stationary.



Note :

- Often, stock prices are not a stationary process, since we might see a growing trend, or its volatility might increase over time (meaning that variance is changing).
- Ideally, we want to have a stationary time series for modelling. Of course, not all of them are stationary, but we can make different transformations to make them stationary.

Following figure depicts the difference in trend and pattern analysis



For the analysis we need to model the time series data before identifying trend and Patterns.

Modeling time series

There are many ways to model a time series in order to make predictions. Here, we will cover the most famous techniques

I. Moving average

The moving average model is probably the most naive approach to time series modelling. This model simply states that the next observation is the mean of all past observations.

Although simple, this model might be surprisingly good and it represents a good starting point.

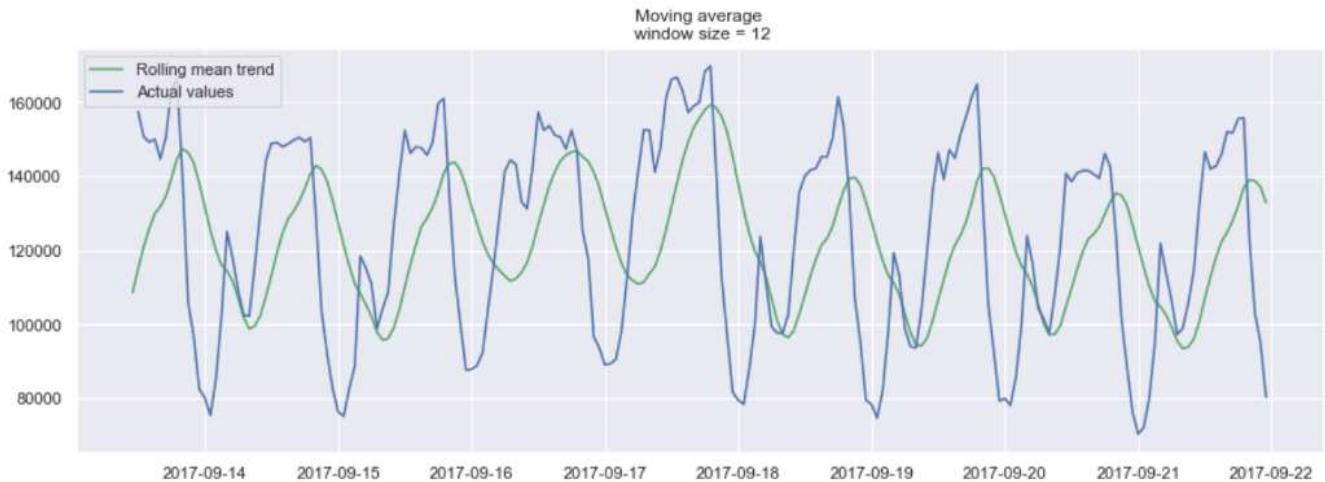
Otherwise, the moving average can be used to identify interesting trends in the data. We can define a window to apply the moving average model to smooth the time series, and highlight different trends.



Example of a moving average on a 24h window

In the plot above, we applied the moving average model to a 24h window. The green line smoothed the time series, and we can see that there are 2 peaks in a 24h period.

Of course, the longer the window, the smoother the trend will be. Below is an example of moving average on a smaller window.



Example of a moving average on a 12h window

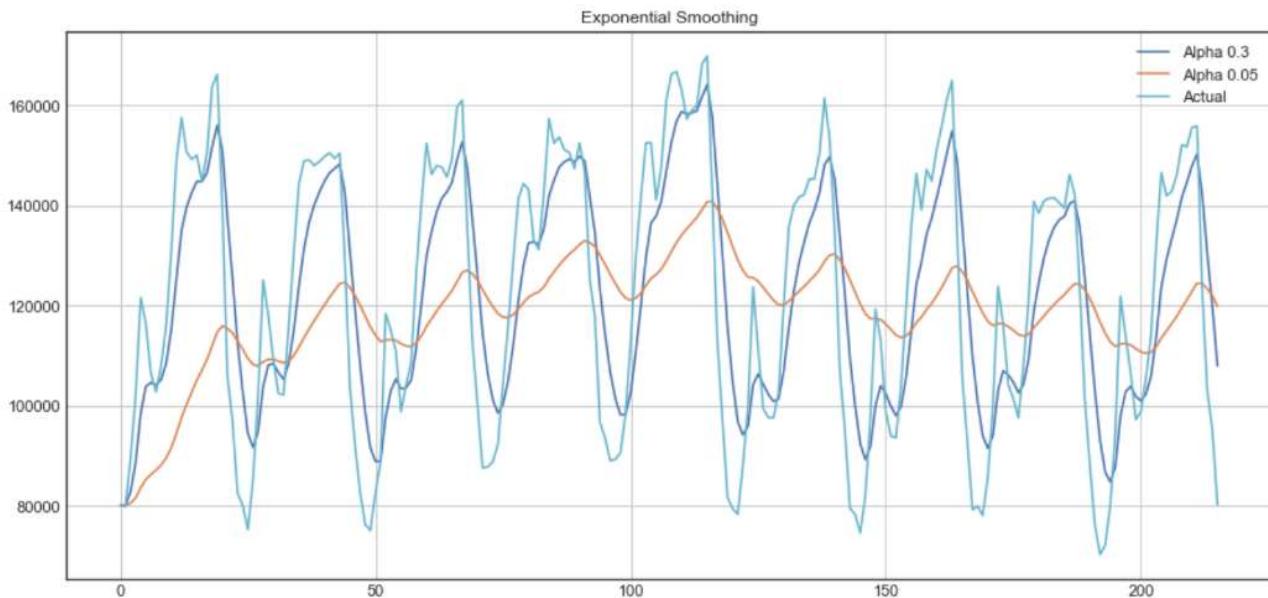
II. Exponential smoothing

Exponential smoothing uses a similar logic to moving average, but this time, a different decreasing weight is assigned to each observations. In other words, less importance is given to observations as we move further from the present.

Mathematically, exponential smoothing is expressed as:

$$y = \alpha x_t + (1 - \alpha)y_{t-1}, t > 0$$

Where alpha is a smoothing factor that takes values between 0 and 1. It determines how fast the weight decreases for previous observations.



The above plot shows that the dark blue line represents the exponential smoothing of the time series using a smoothing factor of 0.3, while the orange line uses a smoothing factor of 0.05.

It also depicts that, smaller smoothing factor will result in smoother time series. As smoothing factor approaches 0, we get the moving average model.

<https://chartio.com/learn/charts/what-is-a-scatter-plot/>

<https://calcworkshop.com/functions-statistics/line-best-fit/>

<https://towardsdatascience.com/understanding-boxplots-5e2df7bcd51>

In previous sections we discussed various charts. This section will cover plots (Some time also referred as charts only)

- A. Scatter plot (Scatter Chart)
- B. Bubble Plot (Chart)
- C. Box Plot
- D. Time line Plot chart

A. Scatter Plots

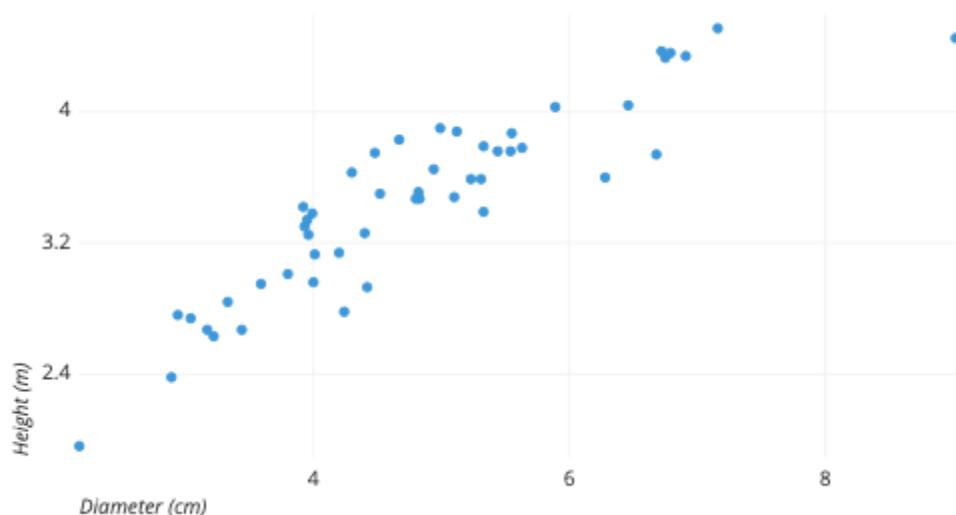
A scatter plot (Scatter Chart, Scatter Graph) uses dots to represent values for two different numeric variables. The position of each dot on the horizontal and vertical axis indicates values for an individual data point. Scatter plots are used to observe relationships between variables.

Example of data structure

DIAMETER HEIGHT

4.20	3.14
5.55	3.87
3.33	2.84
6.91	4.34

... ...

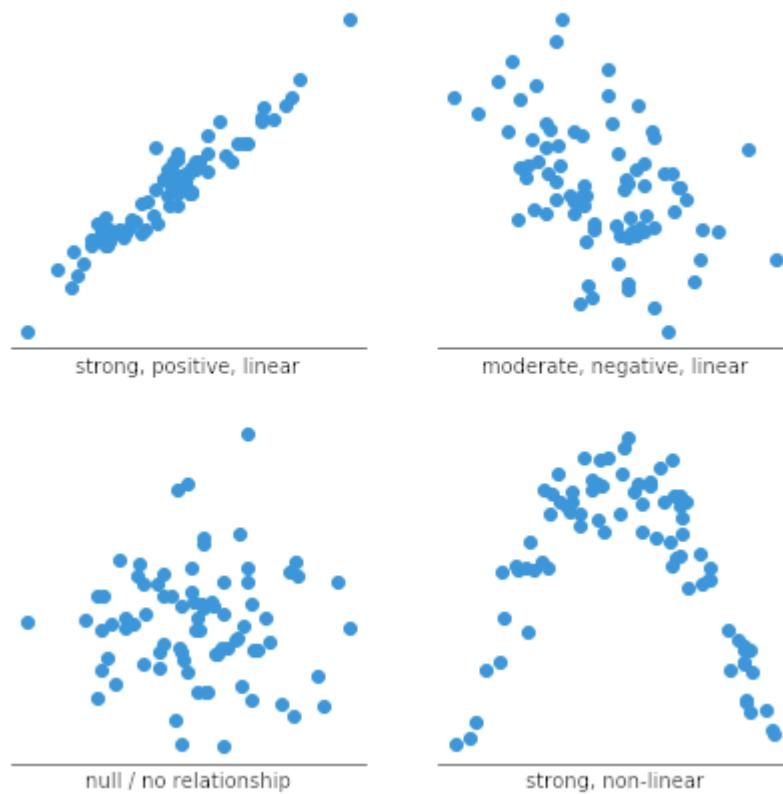
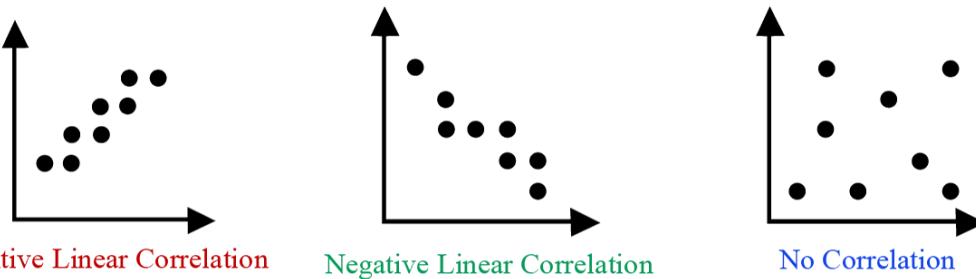


The example scatter plot above shows the diameters and heights for a sample of fictional trees. Each dot represents a single tree; each point's horizontal position indicates that tree's diameter (in centimeters) and the vertical position indicates that tree's height (in meters). From the plot, we can see a generally tight positive correlation between a tree's diameter and its height. We can also observe an outlier point, a tree that has a much larger diameter than the others. This tree appears fairly short for its girth, which might warrant further investigation.

Usage of Scatter plot

1. **Observe and show relationships between two numeric variables:** The dots in a scatter plot not only report the values of individual data points, but also patterns when the data are taken as a whole.

2. **Identification of correlational relationships:** In these cases, we want to know, if we were given a particular horizontal value, what a good prediction would be for the vertical value. You will often see the variable on the horizontal axis denoted an independent variable, and the variable on the vertical axis the dependent variable. Relationships between variables can be described in many ways: positive or negative, strong or weak, linear or nonlinear.



3. **To identify other patterns in data :** We can divide data points into groups based on how closely sets of points cluster together. Scatter plots can also show if there are any unexpected gaps in the data and if there are any outlier points. This can be useful if we want to segment the data into different parts, like in the development of user personas.



In order to create a scatter plot, we need to select two columns from a data table, one for each dimension of the plot. Each row of the table will become a single dot in the plot with position according to the column values.

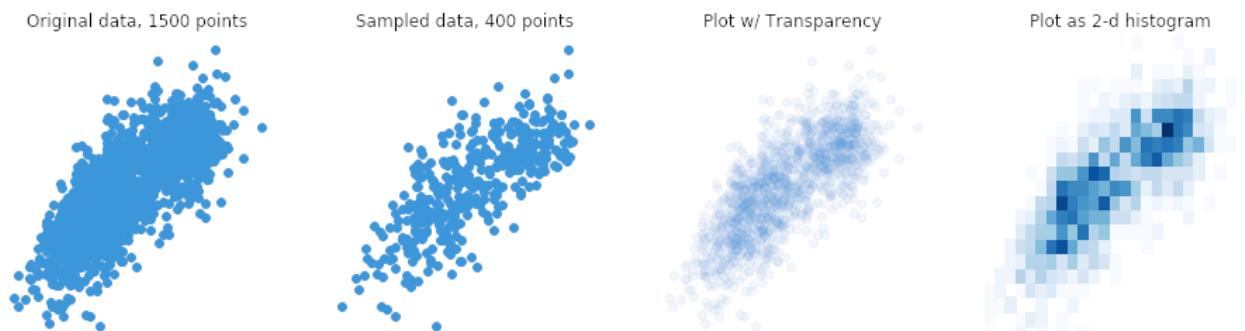
Common issues when using scatter plots

1. Over plotting

When we have lots of data points to plot, this can run into the issue of over plotting. It is the case where data points overlap to a degree where we have difficulty seeing relationships between points and variables. It can be difficult to tell how densely-packed data points are when many of them are in a small area.

Solution for Over Plotting

- i. Sample only a subset of data points: a random selection of points should still give the general idea of the patterns in the full data.
- ii. Change the form of the dots, adding transparency to allow for overlaps to be visible,
- iii. Reduce point size so that fewer overlaps occur.
- iv. Use different chart type like the Heat map (where color indicates the number of points in each bin. Heat maps in this use case are also known as 2-d histograms)



2. Interpreting correlation as causation

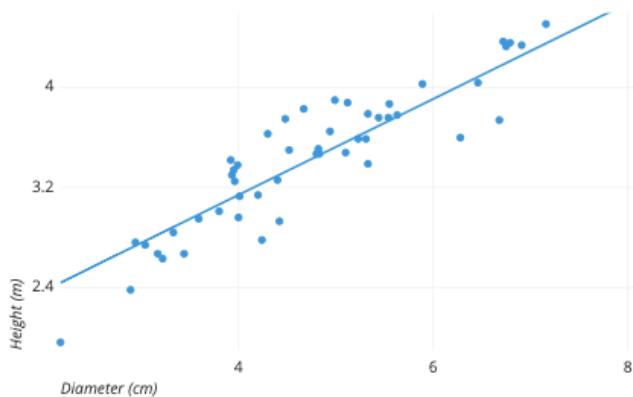
This is not so much an issue with creating a scatter plot as it is an issue with its interpretation. Simply because we observe a relationship between two variables in a scatter plot, **it does not mean that changes in one variable are responsible for changes in the other**. This gives rise to the common phrase in statistics that correlation does not imply causation. It is possible that the observed relationship is driven by some third variable that affects both of the plotted variables, that the causal link is reversed, or that the pattern is simply coincidental.

For example, it would be wrong to look at city statistics for the amount of green space they have and the number of crimes committed and conclude that one causes the other, this can ignore the fact that larger cities with more people will tend to have more of both, and that they are simply correlated through that and other factors. If a causal link needs to be established, then further analysis to control or account for other potential variables effects needs to be performed, in order to rule out other possible explanations. There are options to enhance the interpretation.

- **Common scatter plot options**

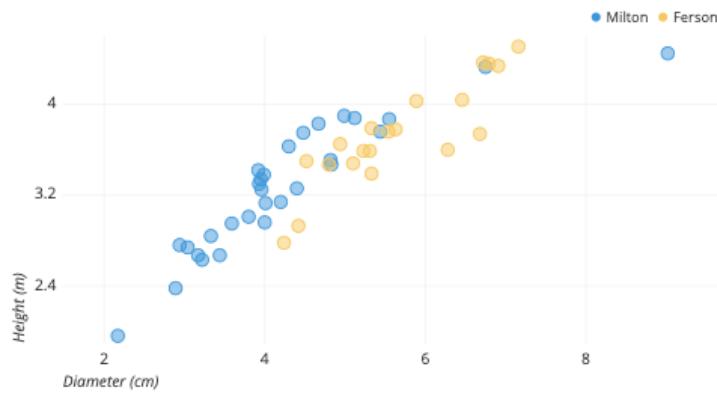
- i. **Add a trend line**

When a scatter plot is used to look at a predictive or correlational relationship between variables, it is common to add a trend line to the plot showing the mathematically best fit to the data. This can provide an additional signal as to how strong the relationship between the two variables is, and if there are any unusual points that are affecting the computation of the trend line.



- ii. **Categorical third variable**

A common modification of the basic scatter plot is the addition of a third variable. Values of the third variable can be encoded by modifying how the points are plotted. For a third variable that indicates categorical values (like geographical region or gender), the most common encoding is through point color. Giving each point a distinct hue makes it easy to show membership of each point to a respective group.



Coloring points by tree type,

From Graph Fersons (yellow) are generally wider than Miltons (blue), but also shorter for the same diameter.

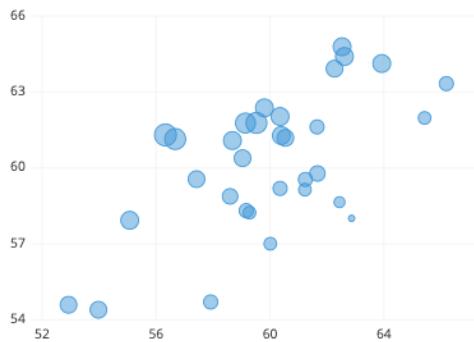
iii. Different shapes as Third variable:

One other option that is sometimes seen for third-variable encoding is that of shape. One potential issue with shape is that different shapes can have different sizes and surface areas, which can have an effect on how groups are perceived. However, in certain cases where color cannot be used (like in print), shape may be the best option for distinguishing between groups.



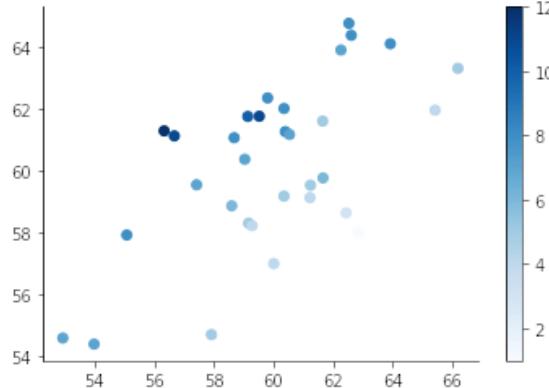
iv. Numeric third variable

- **Size:** For third variables that have numeric values, a common encoding comes from changing the point size. A scatter plot with point size based on a third variable also known as bubble chart. Larger points indicate higher values. (Detail of Bubble chart in following discussion)



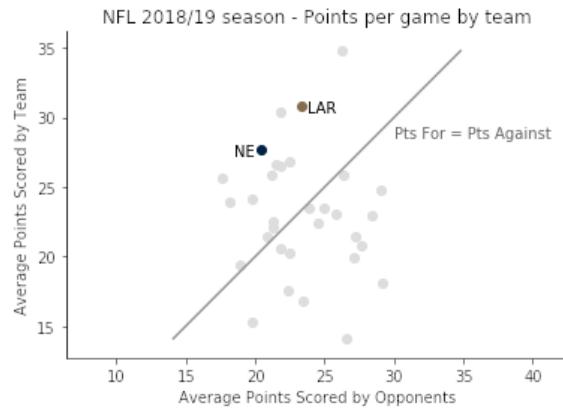
- **Hue:** Rather than using distinct colors for points like in the categorical case, we want to use a continuous sequence of colors, so that, for example, darker colors indicate higher value. Note

that, for both size and color, a legend is important for interpretation of the third variable, since our eyes are much less able to discern size and color as easily as position.



v. *Highlight using annotations and color*

If you want to use a scatter plot to present insights, it can be good to highlight particular points of interest through the use of annotations and color. Desaturating unimportant points makes the remaining points stand out, and provides a reference to compare the remaining points against.



- *Other related representation of scatter plot*

- i. *Scatter Map*

When the two variables in a scatter plot are geographical coordinates – latitude and longitude – we can overlay the points on a map to get a scatter map (aka dot map). This can be convenient when the geographic context is useful for drawing particular insights and can be combined with other third-variable encodings like point size and color.

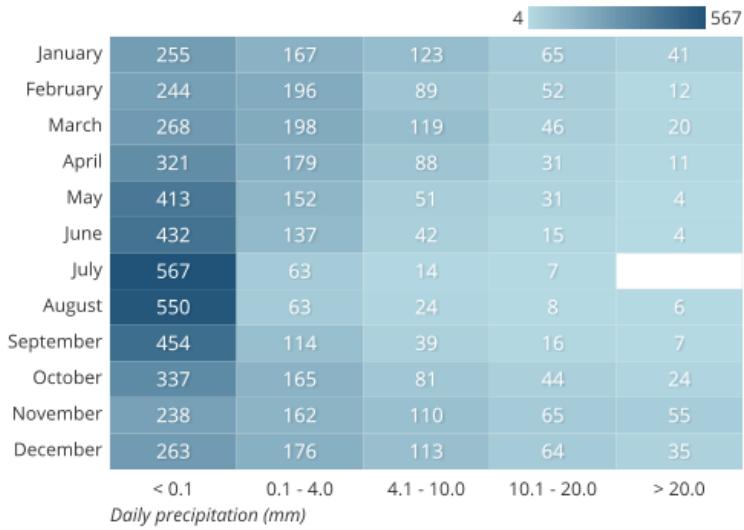


A famous example of scatter map is John Snow's 1854 cholera outbreak map, showing that cholera cases (black bars) were centered around a particular water pump on Broad Street (central dot). Original: Wikimedia Commons

ii. Heat map

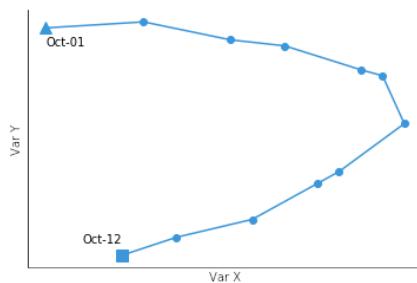
Heat map can be a good alternative to the scatter plot when there are a lot of data points that need to be plotted and their density causes over plotting issues. The heatmap can also be used in a similar fashion to show relationships between variables **when one or both variables are not continuous and numeric**. If we try to depict discrete values with a scatter plot, all of the points of a single level will be in a straight line. Heatmaps can overcome this over plotting through their binning of values into boxes of counts.

Seattle precipitation by month, 1998-2018



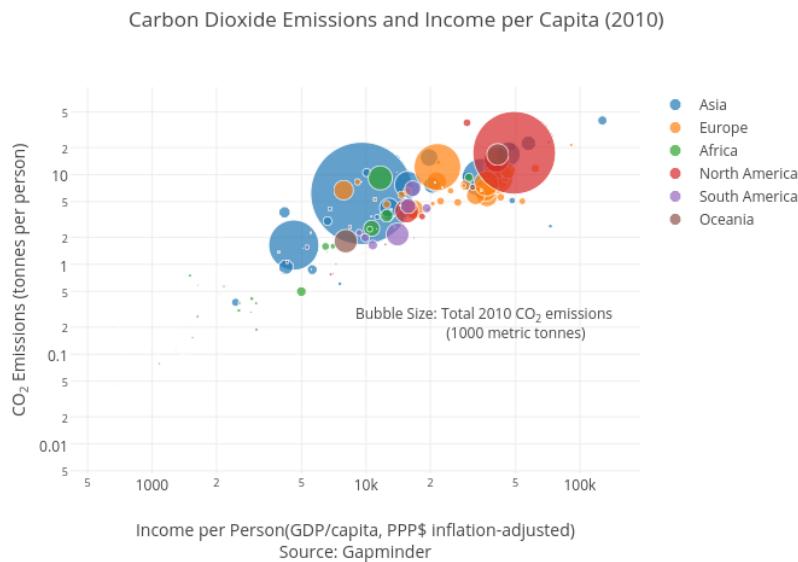
iii. Connected scatter plot

If the third variable we want to add to a scatter plot indicates timestamps, then one chart type we could choose is the connected scatter plot. Rather than modify the form of the points to indicate date, we use line segments to connect observations in order. This can make it easier to see how the two main variables not only relate to one another, but how that relationship changes over time. If the horizontal axis also corresponds with time, then all of the line segments will consistently connect points from left to right, and we have a basic line chart.



B. Bubble Plot (Chart)

A **bubble chart** is a scatter plot that includes a third variable. This third variable is represented as the size of the data point, creating the bubble. Adding another variable can help your data tell a more complete story. This bubble chart uses the same data as the scatter plot, but now the dot size is proportional to the total carbon emissions of the country. The fourth variable shown here is continent, represented with color.



This chart can lend insight that wasn't available with the scatter plot alone. For example, although Qatar has the highest CO₂ emissions per person, the entire country doesn't contribute nearly as much to global CO₂ as China or the United States. If your bubbles range in size a lot, it might be hard to see the smallest bubbles, and the largest bubbles might obscure the surrounding data. You can make some adjustments on the mode tab under traces.

<https://chartio.com/learn/charts/what-is-a-scatter-plot/>

<https://calcworkshop.com/functions-statistics/line-best-fit/>

<https://towardsdatascience.com/understanding-boxplots-5e2df7bcbd51>

<http://web.mnstate.edu/peil/MDEV102/U4/S36>

In previous sections we discussed various charts. This section will cover plots (Some time also referred as charts only)

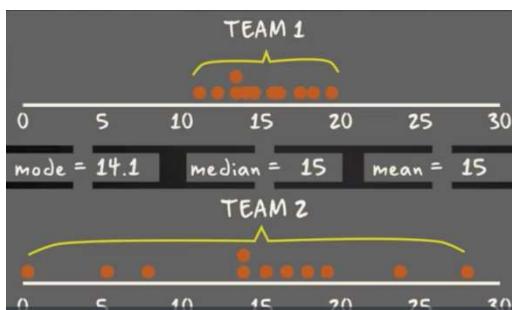
- A. Scatter plot (Scatter Chart)
- B. Bubble Plot (Chart)
- C. Box Plot
- D. Time line Plot chart

C. Box Plot

<https://towardsdatascience.com/understanding-boxplots-5e2df7bcbd51>

What is a Boxplot (Box and whisker Plot)?

For some distributions/datasets, you will find that you need more information than the measures of central tendency (median, mean, and mode).



There are times when mean, median, and mode aren't enough to describe a dataset. In the figure above two Teams with same mode, median and mean are having different dispersion of the data.

A box-plot is a graph that gives you a good indication of how the values in the data are spread out.

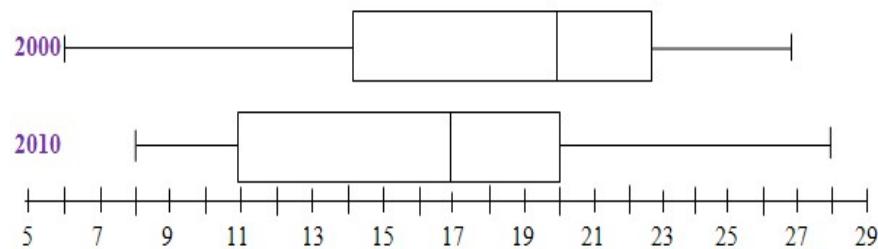
It takes less space compared to histograms and barchart , which is useful when **comparing distributions between many groups or datasets**.

A boxplot is a standardized way of displaying the distribution of data based on a five number summary (“minimum”, first quartile (Q1), median, third quartile (Q3), and “maximum”). It can tell you about your outliers and what their values are. It can also tell you if your data is symmetrical, how tightly your data is grouped, and if and how your data is skewed.

Significance of Box Plot

Interpret the information given in the following box-and-whisker plot.

The results from a pre-test for students for the year 2000 and the year 2010 are illustrated in the box plot. What do these results tell us about how students performed on the 29 question pre-test for the two years



- If we compare only the lowest and highest scores between the two years, we might conclude that
 - the students in 2010 did better than the students in 2000.
 - since the lowest score of 8 in 2010 is greater in value than the lowest score of 6 in 2000.
 - And the highest score of 28 in 2010 is greater in value than the highest score of 27 in 2000.
- But the box portion of the illustration gives us more detailed information.
 - The middle bar in each box shows us that the median score of 20 in 2000 is greater in value than the median score of 17 in 2010.
 - Further, we note that the box and whiskers divide the illustration into four pieces. Each of these four pieces represents the same portion of students.
 - So, the upper half of the students in 2000 scored in the same score range as the upper one-fourth of the students in 2010, see the illustration at a score of 20.
- By considering the upper one-fourth, upper half, and upper three-fourths instead of just the lowest and highest scores, we would conclude that the students as a whole did much better in 2000 than in 2010. We would conclude that as a whole the students in 2010 are less prepared than the students in 2000.
- The Box Plot (box-and-whisker plots) follows **five-number summary**, which consists of the minimum, first quartile, median, third quartile, and maximum.

Median (For reference)

In previous sessions, we worked problems involving the **mean** and **median**. For this session, we primarily use the median. Here is a brief review of terms used with the median:

- A **data set** is any finite set of real numbers.
- A data set is in **increasing order** if the numbers in the data set are arranged from the least value to greatest value with the least value on the left and the greatest value on the right.
- The **median** of a data set is the number that, when the set is put into increasing order, divides the data into two equal parts.
- If a data set has an **odd** number of data points, then the **median is the middle data value** (when the data is in increasing order).
- If a data set has an **even** number of data points, then the **median is the mean of the two middle data values** (when the data is in increasing order).

Example 1: Find the median of the data set {3, 7, 8, 5, 12, 14, 21, 13, 18}.

First, we put the values in the data set into increasing order: 3, 5, 7, 8, 12, 13, 14, 18, 21. Notice that the number of data values is 9, which is odd (there are nine numbers in this data set). Then the middle data value is the 5th value, counting from either the left or the right. Therefore the median is 12.

(The values 3, 5, 7, and 8 are to the left of 12, and 13, 14, 18, and 21 are to the right of 12.)

Example 2: Find the median of the data set {3, 7, 8, 5, 12, 14, 21, 15, 18, 14}.

Note that here we consider the two 14's to be distinct elements and not representing the same item; consider this like you obtained a score of 14 on two different quizzes.

First, we put the values into increasing order: 3, 5, 7, 8, 12, 14, 14, 15, 18, 21. Notice that there are 10 values, which is even. Then the middle data values are the 5th value from the left and the 5th value from the right. That is, 12 and 14 (the leftmost of the two 14s). Therefore the median is the mean of

$$\text{the two middle values: The median is } \frac{12+14}{2} = \frac{26}{2} = 13$$

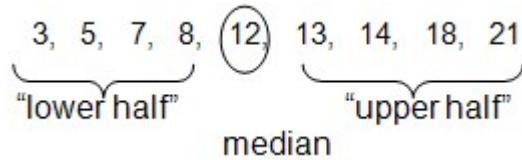
First Quartile and Third Quartile

Definitions:

- The **lower half** of a data set is the set of all values that are to the left of the median value when the data has been put into increasing order.
- The **upper half** of a data set is the set of all values that are to the right of the median value when the data has been put into increasing order.
- The **first quartile**, denoted by Q_1 , is the median of the *lower half* of the data set. This means that about 25% of the numbers in the data set lie below Q_1 and about 75% lie above Q_1 .
- The **third quartile**, denoted by Q_3 , is the median of the *upper half* of the data set. This means that about 75% of the numbers in the data set lie below Q_3 and about 25% lie above Q_3 .

Example 1: Find the first and third quartiles of the data set $\{3, 7, 8, 5, 12, 14, 21, 13, 18\}$.

First, we write data in increasing order: $3, 5, 7, 8, 12, 13, 14, 18, 21$.



As on the previous page, the median is 12.

Therefore, the lower half of the data is: $\{3, 5, 7, 8\}$.

The first quartile, Q_1 , is the median of $\{3, 5, 7, 8\}$.

Since there is an even number of values, we need the mean of the middle two values to find the first quartile:

$$Q_1 = \frac{5+7}{2} = \frac{12}{2} = 6$$

Similarly, the upper half of the data is: $\{13, 14, 18, 21\}$, so

$$Q_3 = \frac{14+18}{2} = \frac{32}{2} = 16$$

Example 2: Find the first and third quartiles of the set $\{3, 7, 8, 5, 12, 14, 21, 15, 18, 14\}$.

Note that here we consider the two 14's to be distinct elements and not representing the same item; consider this like you obtained a score of 14 on two different quizzes.

First, we write the data in increasing order: $3, 5, 7, 8, 12, 14, 14, 15, 18, 21$.

As before, the median is 13 (it is the mean of 12 and 14 — the pair of middle entries).

Therefore, the lower half of the data is: $\{3, 5, 7, 8, 12\}$.

Notice that 12 is included in the lower half since it is below the median value.

Then $Q_1 = 7$ (there are five values in the lower half, so the middle value is the median). Similarly, the upper half of the data is: {14, 14, 15, 18, 21}, so $Q_3 = 15$.

Five-Number Summary

Definitions:

- The **minimum value** of a data set is the least value in the set.
- The **maximum value** of a data set is the greatest value in the set.
- The **range** of a data set is the distance between the maximum and minimum value. To compute the range of a data set, we subtract the minimum from the maximum:
$$\text{range} = \text{maximum} - \text{minimum}$$
.
- The **interquartile range** of a data set is the distance between the two quartiles.
$$\text{Interquartile range} = Q_3 - Q_1$$
.

Example 1: Find the range and interquartile range of the set $\{3, 7, 8, 5, 12, 14, 21, 13, 18\}$.

First, we write the data in increasing order: 3, 5, 7, 8, 12, 13, 14, 18, 21.

$$\text{range} = \text{max} - \text{min} = 21 - 3 = 18.$$

Recall from the previous page that $Q_1 = 6$ and $Q_3 = 16$.

$$\text{Therefore, the interquartile range} = Q_3 - Q_1 = 16 - 6 = 10.$$

The range is 18 and the interquartile range is 10.

Example 2: Find the range and interquartile range of the set $\{3, 7, 8, 5, 12, 14, 21, 15, 18, 14\}$.

First, we write the data in increasing order: 3, 5, 7, 8, 12, 14, 14, 15, 18, 21.

$$\text{range} = \text{max} - \text{min} = 21 - 3 = 18.$$

Recall from the previous page that $Q_1 = 7$ and $Q_3 = 15$.

$$\text{Therefore, the interquartile range} = Q_3 - Q_1 = 15 - 7 = 8.$$

The range is 18 and the interquartile range is 8.

Self Check Problem

The following dollar amounts were the hourly collections from a Salvation Army kettle at a local store one day in December: \$19, \$26, \$25, \$37, \$32, \$28, \$22, \$23, \$29, \$34, \$39, and \$31. Determine the range and interquartile range for the amount collected.

Solution

The five-number summary from the previous page is

Minimum - 19,

Q_1 - 24,

Median - 28.5,

Q_2 - 33,

Maximum - 39.



Definition: The **five-number summary** of a data set consists of the five numbers determined by computing the **minimum**, Q_1 , **median**, Q_3 , and **maximum** of the data set.

Example 1: Find the five-number summary for the data set {3, 7, 8, 5, 12, 14, 21, 13, 18}.

From our Example 1's on the previous pages, we see that the five-number summary is:

Minimum: 3 Q_1 : 6 Median: 12 Q_3 : 16 Maximum: 21

Example 2: Find the five-number summary for the data set {3, 7, 8, 5, 12, 14, 21, 15, 18, 14}.

From our Example 2's on the previous pages, we see that the five-number summary is:

Minimum: 3 Q_1 : 7 Median: 13 Q_3 : 15 Maximum: 21

Self Check Problem

The following dollar amounts were the hourly collections from a Salvation Army kettle at a local store one day in December: \$19, \$26, \$25, \$37, \$32, \$28, \$22, \$23, \$29, \$34, \$39, and \$31. Find the five-number summary for the amount collected.

Solution

The results follow from the Self-Check Problems on the previous pages.

Minimum \$19,

First Quartile \$24,

Median \$28.50,

Third Quartile \$33,

Maximum \$39.

Box-and-Whisker Plot

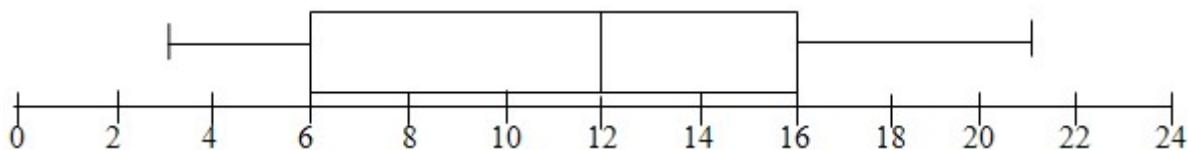
Definition: A **box-and-whisker plot** or **boxplot** is a diagram based on the five-number summary of a data set.

- To construct this diagram, we first draw an **equal interval** scale on which to make our box plot. **Do not** just draw a boxplot shape and label points with the numbers from the 5-number summary. The boxplot is a visual representation of the distribution of the data. Greater distances in the diagram should correspond to greater distances between numeric values.
- Using the equal interval scale, we draw a rectangular box with one end at Q_1 and the other end at Q_3 . And then we draw a vertical segment at the median value. Finally, we draw two horizontal segments on each side of the box, one down to the minimum value and one up to the maximum value, (these segments are called the "**whiskers**").

Example 1: Draw a box-and-whisker plot for the data set $\{3, 7, 8, 5, 12, 14, 21, 13, 18\}$.

From our Example 1 on the previous page, we had the five-number summary:

Minimum: 3, Q_1 : 6, Median: 12, Q_3 : 16, and Maximum: 21.



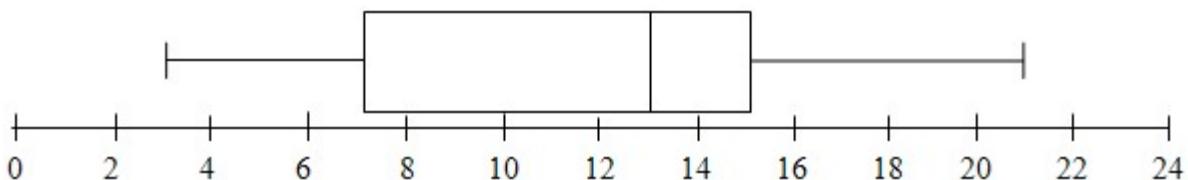
Notice that in any box-and-whisker plot,

- the left-side whisker represents where we find approximately the lowest 25% of the data
- and the right-side whisker represents where we find approximately the highest 25% of the data
- The box part represents the interquartile range and represents approximately the middle 50% of all the data.
- The data is divided into four regions, which each represent approximately 25% of the data. This gives us a nice visual representation of how the data is spread out across the range.

Example 2: Draw a box-and-whisker plot for the data set $\{3, 7, 8, 5, 12, 14, 21, 15, 18, 14\}$.

From our Example 2 on the previous page, we had the five-number summary:

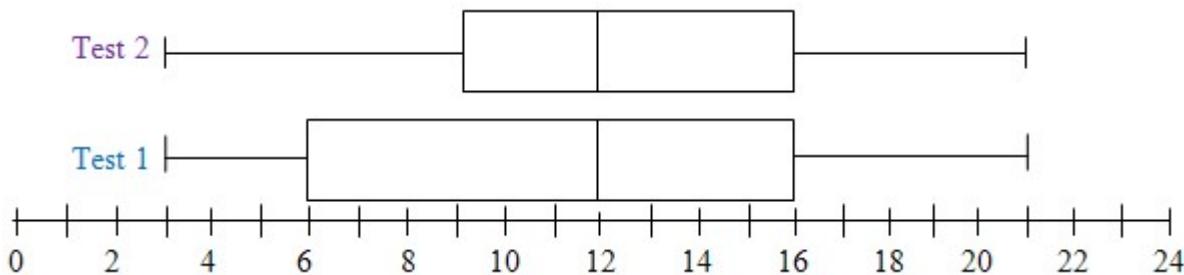
Minimum: 3, Q_1 : 7, Median: 13, Q_3 : 15, and Maximum: 21.



When we relate two data sets based on the same scale, we may examine box-and-whisker plots to get an idea of how the two data sets compare.

Example 3: Suppose that the box-and-whisker plots below represent quiz scores out of 25 points for Quiz 1 and Quiz 2 for the same class.

What do these box-and-whisker plots show about how the class did on **test #2** compared to **test #1**?



These box-and-whisker plots show that the lowest score, highest score, and Q_3 are all the same for both exams, so performance on the two exams were quite similar. However, the movement Q_1 up from a score of 6 to a score of 9 indicates that there was an overall improvement. On the first test, approximately 75% of the students scored at or above a score of 6. On the second test, the same number of students (75%) scored at or above a score of 9.

Self Check Problem

The following dollar amounts were the hourly collections from a Salvation Army kettle at a local store one day in December: \$19, \$26, \$25, \$37, \$32, \$28, \$22, \$23, \$29, \$34, \$39, and \$31. Construct the box-and-whisker plot for the amount collected.

The five-number summary from the previous page is

Minimum - 19, Q_1 - 24, Median - 28.5, Q_2 - 33, Maximum - 39.



Characteristics of box plots

1. Handles Large Data Easily

Due to the five-number data summary, a box plot can handle and present a summary of a large amount of data. A box plot consists of the median, which is the midpoint of the range of data; the upper and lower quartiles, which represent the numbers above and below the highest and lower quarters of the data and the minimum and maximum data values. Organizing data in a box plot by using five key concepts is an efficient way of dealing with large data too unmanageable for other graphs, such as line plots or stem and leaf plots.

2. Exact Values Not Retained

The box plot does not keep the exact values and details of the distribution results, which is an issue with handling such large amounts of data in this graph type. A box plot shows only a simple summary of the distribution of results so that you can quickly view it and compare it with other data. Use a box plot in combination with another statistical graph method, like a histogram, for a more thorough, more detailed analysis of the data.

3. A clear summary

A box plot is a highly visually effective way of viewing a clear summary of one or more sets of data. It is particularly useful for quickly summarizing and comparing different sets of results from different experiments. At a glance, a box plot allows a graphical display of the distribution of results and provides indications of symmetry within the data.

4. Displays outliers

A box plot is one of very few statistical graph methods that show outliers. There might be one outlier or multiple outliers within a set of data, which occurs both below and above the minimum and maximum data values. By extending the lesser and greater data values to a max of 1.5 times the inter-quartile range, the box plot delivers outliers or obscure results. Any results of data that fall outside of the minimum and maximum values known as outliers are easy to determine on a box plot graph.

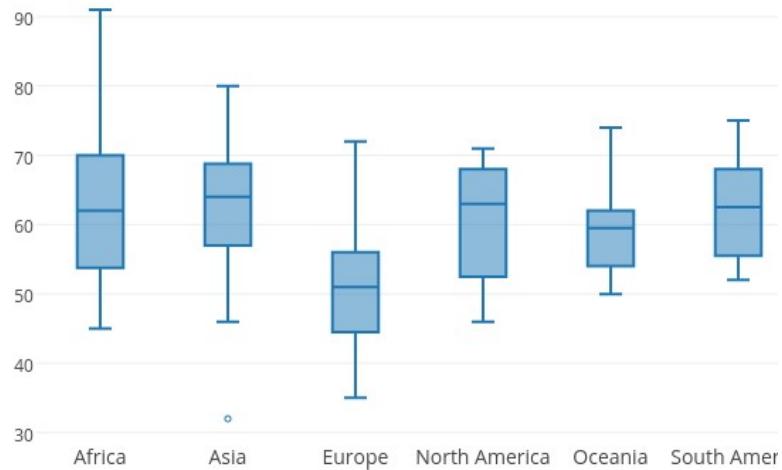
Application of BOX Plot

- If you are given the data for the heads of the governments and you are asked to compare among that of different continents

If you use histogram you need to draw separate histograms for each continent and then compare the graph

Instead the BOX plot representation would be as below from which we can easily compare median age from every continent and percentage of people below or above the median can also be easily compared.

Ages of Heads of Governments Around the World



Source: Wikipedia

Using Africa as an example, the median age of government leaders is 62. The two quartiles (that form the box) are 53.75 and 70, which means that half of the data points are found within this range. The “whiskers” show the minimum and maximum of 45 and 91.

- A box plot is ideal for comparing the distribution of a series of datasets, like this data for each continent.
- It is often used to track different trials of an experiment that is run many times. If the trials are exactly the same, a box plot will show the consistency of results. If they vary by a parameter that is being tested, a box plot could reveal trends or patterns.
- Histogram will demonstrate poor performance when there are few samples or when the boxes are the wrong sizes.
- box-plots provide some information that the histogram does not explicitly, that is, median, 25th and 75th percentile, min/max that is not an outlier and explicitly separates the points that are considered outliers.

<https://dimensionless.in/what-is-a-box-plot/>

Managing Outliers in BOX Plot

Deciding Minimum and Maximum considering Outliers

1. Highest Value (Maximum)

This point in the box plot represents the highest value in the data distribution over which the box plot is built which is not an outlier. This point does not correspond to the Maximum value in your dataset.

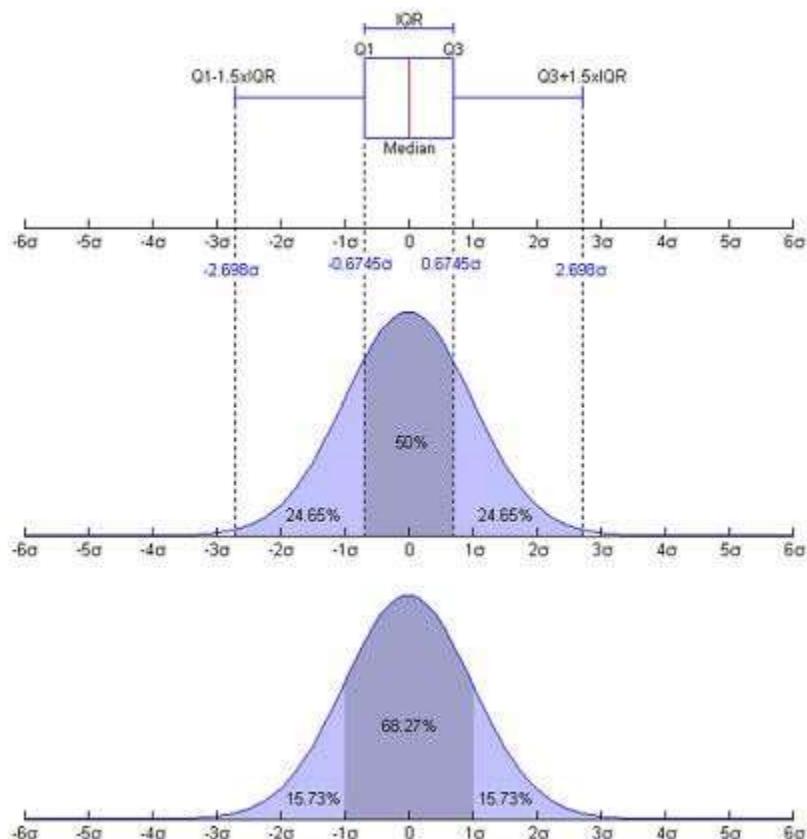
Suppose you have some data like 65,76,87,100,105,100000. Here the Maximum value is 100000 but it is most likely to be an outlier and hence the box plot will not mark this as the maximum value. The most feasible option will be 105 as the maximum value of the box plot.

2. Lowest Value (Minimum)

This point in the box plot represents the lowest value in the data distribution over which the box plot is built and is not an outlier (smallest value in the Interquartile range of the distribution). This point does not correspond to the smallest value in your dataset. Suppose you have some data like

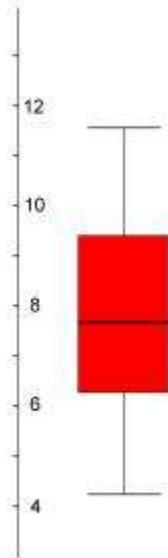
0.005,65,76,87,100,105. Here the smallest value is 0.005 but it is most likely to be an outlier and hence the box plot will not mark this as the minimum value. The most feasible option will be 65 as the minimum value of the box plot.

In this case Boxplot statistic would be based on the updated Maximum and Minimum values



Understanding different box plots

Although boxplots can be drawn in any orientation, most statistical packages seem to produce them vertically by default, as shown below, rather than horizontally. The length of the box becomes its height. The width across the page signifies nothing.



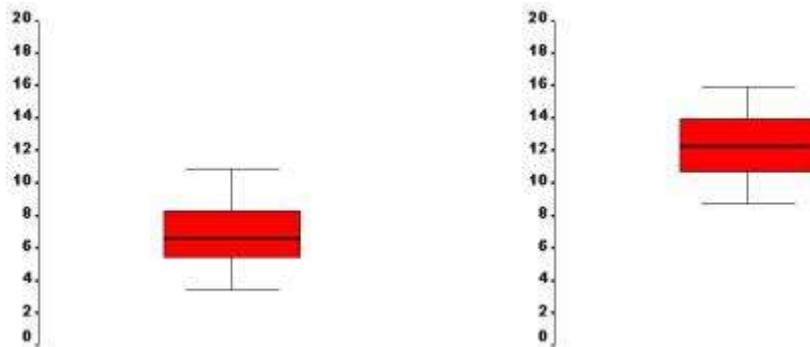
Boxplots of several samples are lined up alongside one another. The box length gives an indication of the sample variability and the line across the box shows where the sample is centred. The position of the box in its whiskers and the position of the line in the box also tells us whether the sample is symmetric or skewed, either to the right or left. For a symmetric distribution, long whiskers, relative to the box length, can betray a heavy tailed population and short whiskers, a short tailed population. So, provided the number of points in the sample is not too small, the boxplot also gives us some idea of the "shape" of the sample, and by implication, the shape of the population from which it was drawn. This is all important when considering appropriate analyses of the data.

Data on different house prices in 5 different areas of Bangalore is plotted. Let us understand the distribution of this data and try to find some insights out of it.



- The Box plot as an Indicator of Centrality**

We will try to gather our first insight by observing the centrality of the box plots. Centerline represents the median value for the house price in different areas. Houses on airport road have the highest median value of the house which makes it a comparatively expensive place to live in whereas houses in Marathali have the least median value which allows us to conclude that houses here are relatively cheapest to live.



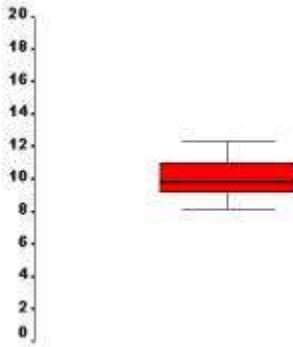
The boxplot of a sample of 20 points from a population centered on 7.

The boxplot of a sample of 20 points from a population centered on 12.

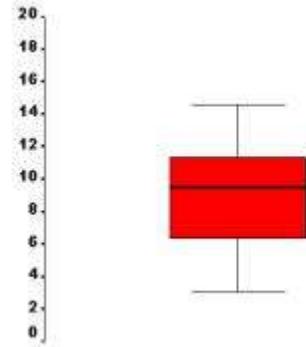
- The Box plot as an indicator of the spread**

The spread of a box plot talks about the variance present in the data. More the spread, more the variance. If you look closely at the first two box plots, both Whitefield and Hoskote areas have the same median house price value so it seems like both places fall into the same budget

category. But if we look more closely, we can observe that width of Hoskote box plot is more than Whitefield box plot. Hoskote area has more variance in house price as compared to Whitefield i.e. Hoskote offers more variety of budget in houses as compared to Whitefield. If we look at the overall graph, we find that Bellathur area has the most spread in its box plot. This clearly states that this area has the widest variety in the budget of the houses.



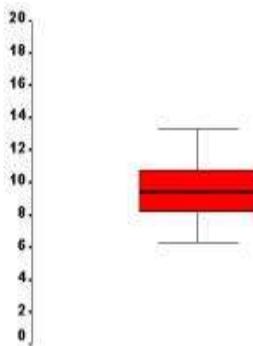
The boxplot of a sample of 20 points from a population centred on 10 with standard deviation 1.



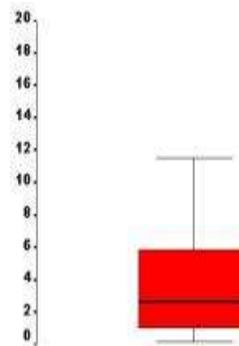
The boxplot of a sample of 20 points from a population centred on 10 with standard deviation 3.

- **The Box plot as an indicator of symmetry**

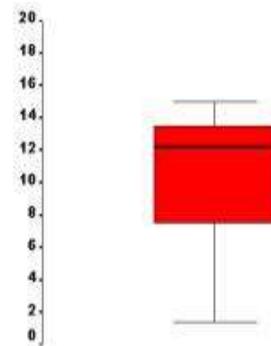
Symmetry around the median talks about skewness present in the data. If the median line is towards the lower half of the box plot, then it is right skewed (positive skew) and if the median line is towards the upper portion of the box plot then it is left-skewed (negative skew). If we look at the box plot representing Marathalli, we can observe that median is towards the lower half of the box plot and hence it is right skewed (positive skew) which means that most of the houses are on the cheaper side in Marathalli and only a few are expensive.



The boxplot of a sample of 20 points from a symmetric population. The line is close to the centre of the box and the whisker lengths are the same.



The boxplot of a sample of 20 points from a population which is skewed to the right. The top whisker is much longer than the bottom whisker and the line is gravitating towards the bottom of the box.

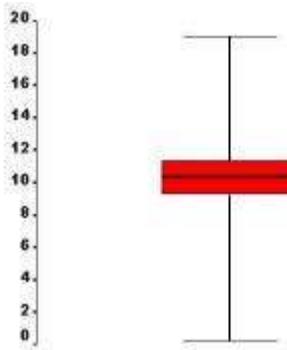


The boxplot of a sample of 20 points from a population which is skewed to the left. The bottom whisker is much longer than the top whisker and the line is rising to the top of the box.

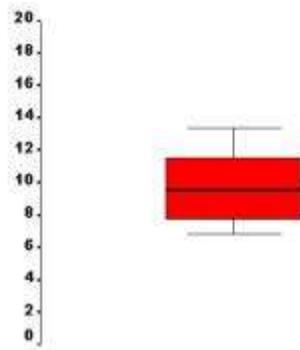
- **The Box plot as an indicator of tail length**

Tail length talks about the kurtosis present in data. There are three cases here. Either your data will be normally distributed or it will have more data in its tail as compared to a normal distribution(platykurtic) or it will have fewer data in tails as compared to a normal distribution(leptokurtic). A long tail shows that the distribution is platykurtic and shorter tail gives the idea of distribution being leptokurtic. In above example, Marathalli has the shortest tail as compared to other box plots which may mean that in Marathalli most of the house prices lie in the interquartile range (q3-q1).

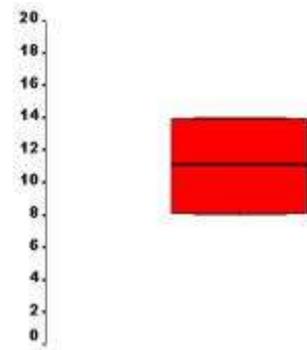
- The tails are the extremities of the sample or population, rather than the centre. Lack of symmetry entails one tail being longer than the other. Populations are usually referred to as being heavy-tailed or light-tailed, or the Greek equivalent, leptokurtic (slender arched) or platykurtic (flat arched). The ideal level of kurtosis, neither too heavy or too light, is represented by the Normal population - the bell shaped curve. The box-plot of a sample from a Normal population should exhibit whiskers about the same length as the box, or perhaps marginally longer. The symmetric example [above](#) is from a Normal population.



The boxplot of a sample of 20 points from a population with long tails. The length of the whiskers far exceeds the length of the box. (A well proportioned tail would give rise to whiskers about the same length as the box, or maybe slightly longer.)



The boxplot of a sample of 20 points from a population with short tails. The length of the whiskers is shorter than the length of the box.



The boxplot of a sample of 20 points from a population with extremely short tails (actually a U-shaped population, with a dip in the middle rather than a hump). The whiskers are absent.

Machine Learning Models

Machine Learning Definition

Simply says Finds pattern in data and uses those patterns to predict the future.

It allows us to discover patterns in existing data and create and make use of a model that identifies those patterns in innovative data.

What does it mean to learn?

Learning Process: Identifying patterns

Recognizing those pattern when you see them again

Why is machine learning so popular currently?

- Plenty of data
- Lots of computer power
- An effective machine learning algorithm

Some examples for ML and AI to make our life easy like

- Google Search
- Intelligent Gaming
- Stock Predictions
- Robotics

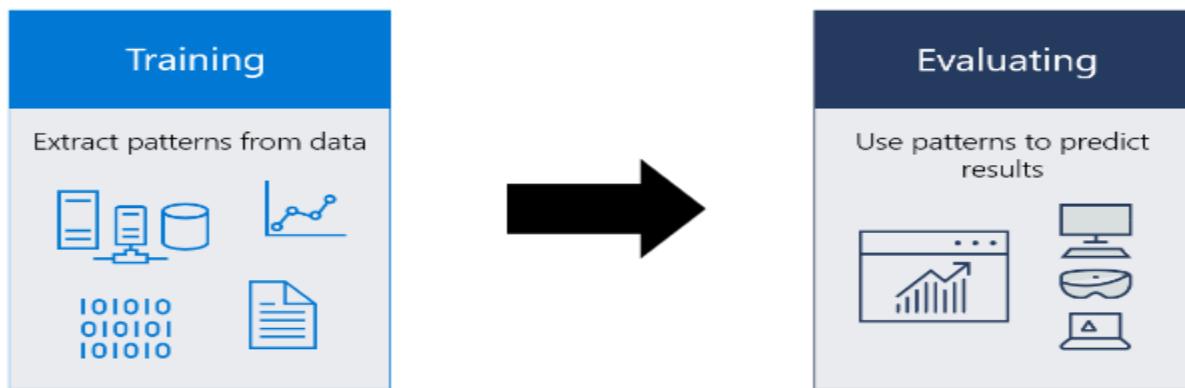
Top Machine Learning Companies

It is becoming an important part of our everyday life. It is really utilized in financial procedures, medical examinations, logistics, posting, and a variety of different fast-rising industries.

- Google – Neural Networks and machines
- Tesla – Autopilot
- Amazon – Echo Speaker Alexa
- Apple – Personalized Hey Siri
- TCS – Machine First Delivery Model with Robotics
- Facebook – Chatbot Army etc.

A machine learning model is a file that has been **trained to recognize certain types of patterns**. You train a model over a set of data, providing it an algorithm that it can use to reason over and learn from those data.

Once you have trained the model, you can use it to reason over data that it hasn't seen before, and make predictions about those data. For example, let's say you want to build an application that can recognize a **user's emotions based on their facial expressions**. You can train a model by providing it with images of faces that are each tagged with a certain emotion, and then you can use that model in an application that can recognize any user's emotion.



A machine learning model is the output of the training process and is defined as the mathematical representation of the real-world process. **The machine learning algorithms find the patterns in the training dataset, which is used to approximate the target function and is responsible for mapping the inputs to the outputs from the available dataset.**

When to use Machine Learning

Good machine learning scenarios often have the following common properties:

- They involve a repeated decision or evaluation which you want to automate and need consistent results.
- It is difficult or impossible to explicitly describe the solution or criteria behind a decision.
- You have labeled data, or existing examples where you can describe the situation and map it to the correct result.

All machine learning models are categorized as either supervised or unsupervised. If the model is a supervised model, it's then sub-categorized as either a regression or classification model.

Supervised Machine Learning

Supervised Machine Learning is defined as the subfield of machine learning techniques in which we used labelled datasets for training the model, making predictions of the output values and comparing its output with the intended, correct output, and then compute the errors to modify the model accordingly. Also, as the system is trained enough using this learning method, it becomes capable enough to provide the target values from any new input.

Think about **Gmail's spam recognition system**. Now there, it will take under consideration a collection of emails (a huge number, just like millions) which have recently been categorized because of **spam or not spam**, from this level, with the ability to identify what features an email that is spam or not spam display. Once gaining knowledge of this, with the ability to classify onset e-mails as spam or otherwise.

When we train the algorithm by providing the labels explicitly, it is known as supervised learning. This type of algorithm uses the available dataset to train the model. The model is of the following form.

$Y=f(X)$ where x is the input variable, y is the output variable, and $f(X)$ is the hypothesis.

The objective of Supervised Machine Learning Algorithms is to find the hypothesis as approx. as possible so that when there is new input data, the output y can be predicted. The application of supervised machine learning is to predict whether a mail is spam or not spam or face unlock in your smartphone.

Types of Supervised Machine Learning Algorithm

Supervised Machine Learning is divided into two parts based upon their output:

- 1. Regression**
- 2. Classification**

Regression

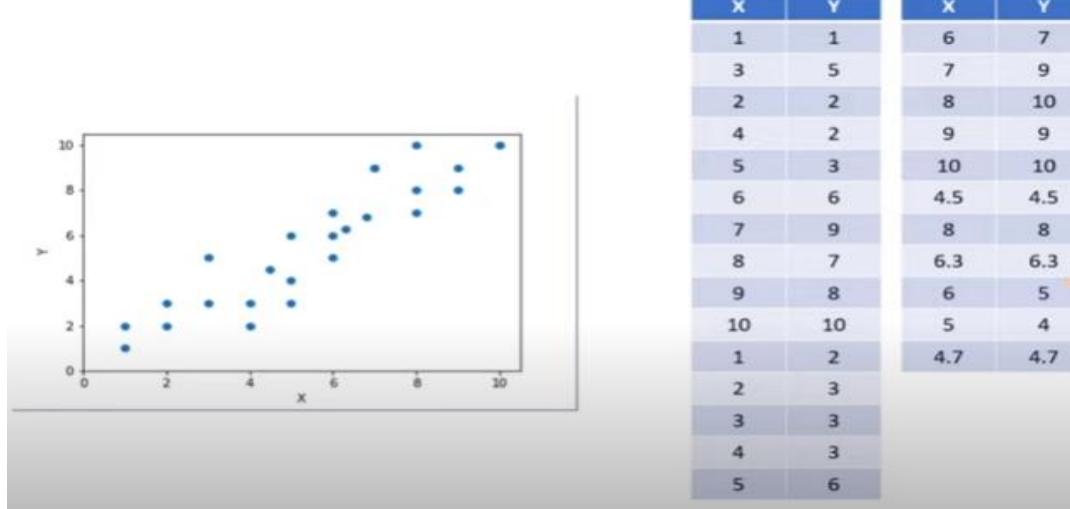
In Regression the output variable is numerical (continuous) i.e. we train the hypothesis($f(x)$) in a way to get continuous output(y) for the input data(x). Since the output is informed of the real number, the regression technique is used in the prediction of quantities, size, values, etc.

For Example, we can use it to predict the price of the house given the dataset containing the features of the house like area, floor, etc.

In regression problems the output values of target class are continuous. Target value must be **integer or float**. In this example target value of house price is an integer value depends on bed rooms in house, floors, condition and location. So, our model will predict an integer value of house price by using these features.

bathrooms	floors	condition	city	price
1.5	3	3	Shoreline	313000
2.5	2	5	Seattle	2384000
2	1	4	Kent	342000
2.25	1	4	Bellevue	420000
2.5	1	4	Redmond	550000
1	1	3	Seattle	490000
2	1	3	Redmond	335000
2.5	2	3	Maple Valley	482000
2.5	1	4	North Bend	452500
2	3	3	Seattle	640000
1.75	1	3	Lake Forest Park	463000
2.5	3	5	Seattle	1400000
1.75	1	3	Sammamish	588500
1	1	4	Seattle	365000

For regression problem suppose you have only one feature x and target value y and you are going to plot in graph for x equal to 1 y also 1, for 3 y equal to 5 and that's going on. Finally, our data will look like this. We can draw a line or curve for predicting new data.

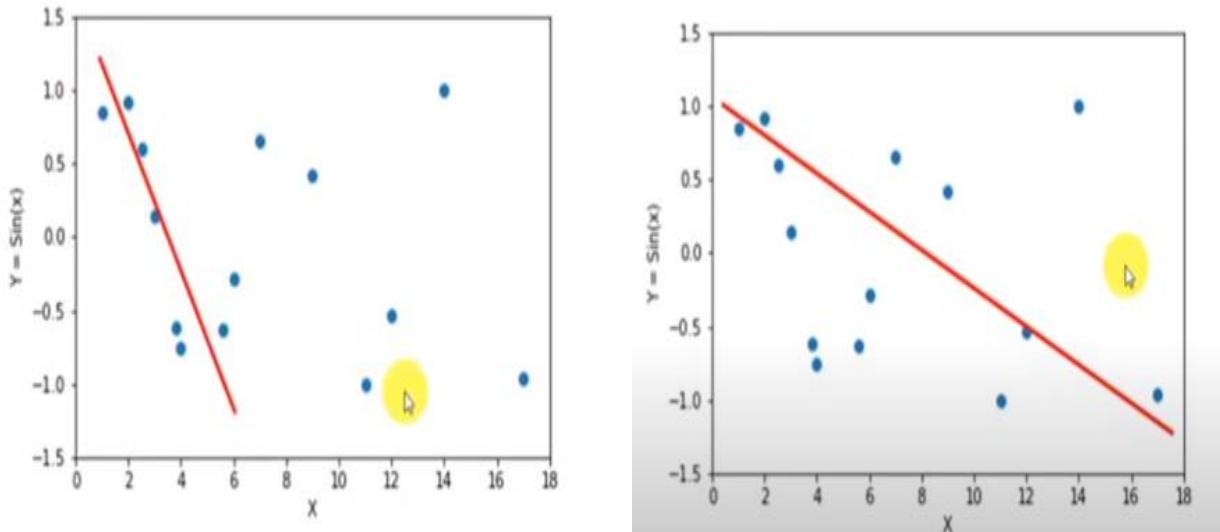


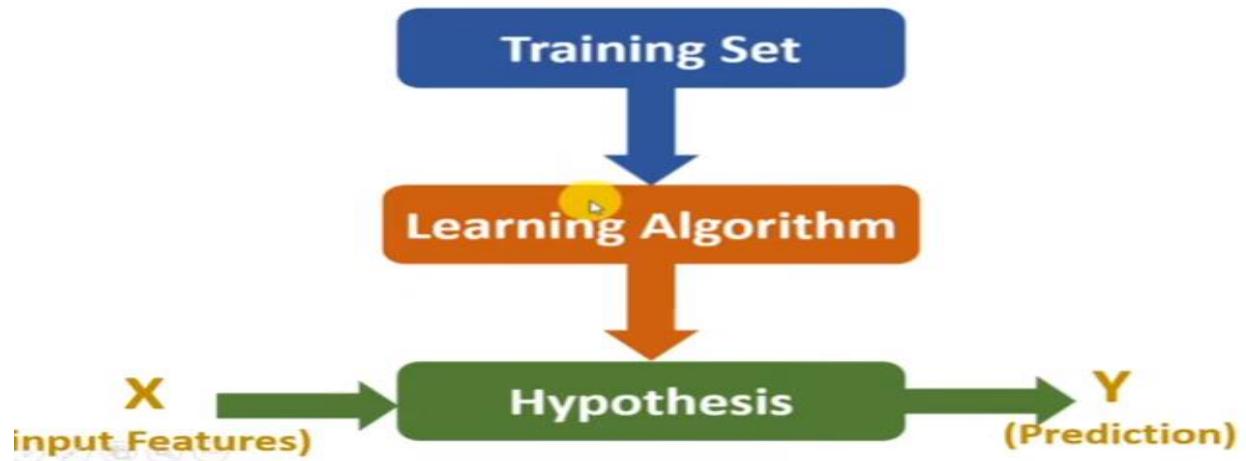
A few popular Regression Algorithm is:

- Linear Regression
- Support Vector Regression
- Poisson Regression

Linear regression is used for finding **linear relationship between target and one or more predictors**. There are two types of linear regression- Simple and Multiple.

The core idea is to obtain a line that **best fits** the data. The **best fit line** is the one for which total prediction error (all data points) are as small as possible. Error is the **distance between the point to the regression line**.





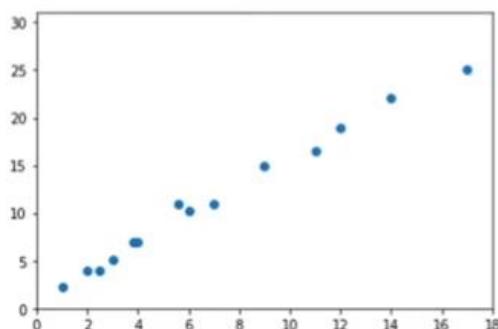
Hypothesis:

$$h_{\theta}(x) = \theta_0 + \theta_1 x \quad \text{Predicted } Y$$

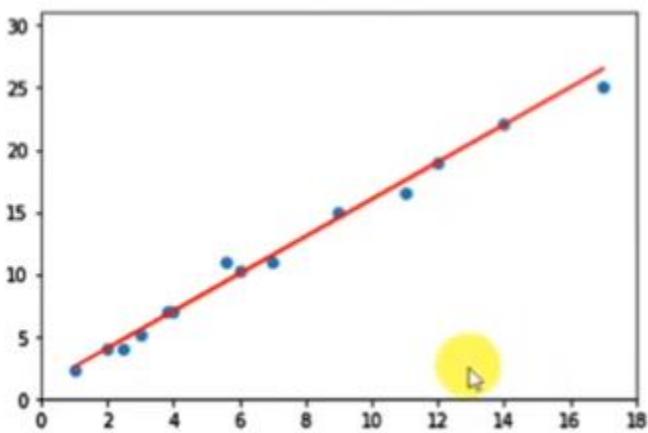
$$\theta_0 = 1, \theta_1 = 1.5$$

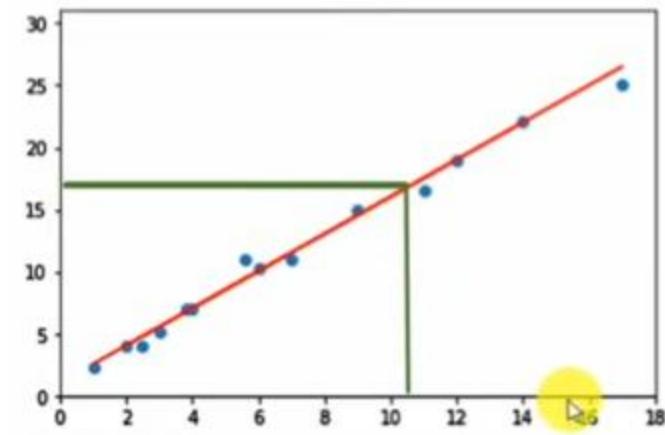
$$h_{\theta}(9) = 1 + 1.5 \times 9$$

$$h_{\theta}(9) = 14.5$$



X	Y	Pred_Y
7	11	11.5
2	4	4
17	25	26.5
9	15	
4	7	
11	16.5	
12	19	
6	10.2	
1	2.3	
3	5.1	
2.5	4	
3.8	7	
5.6	11	
14	22	





Optimizing using gradient descent:

In linear regression, the model targets to get the best-fit regression line to predict the value of y based on the given input value (x). While training the model, the model calculates the cost function which measures the Root Mean Squared error between the predicted value (pred) and true value (y). **The model targets to minimize the cost function.**

To minimize the cost function, the model needs to have the best value of θ_0 and θ_1 . Initially model selects θ_0 and θ_1 values randomly and then iteratively update these values in order to minimize the cost function until it reaches the minimum. By the time model achieves the minimum cost function, it will have the best θ_0 and θ_1 values. Using these finally updated values of θ_0 and θ_1 in the hypothesis equation of linear equation, model predicts the value of x in the best manner it can.

Cost Function:

Minimize $(h_\theta(x) - y)^2$:

minimize the difference between $h(x)$ and y for each/any/every example

$$\frac{1}{2m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)})^2 \quad m = \text{number of iteration}$$

$i = 1, 2, 3, 4, \dots, m$

$$\frac{1}{2 \times 14} \left\{ (11.5 - 11)^2 + (4 - 4)^2 + (26.5 - 25)^2 + (14.5 - 15)^2 + \dots + (22 - 22)^2 \right\}$$

$$\frac{1}{28} (0.25 + 0 + 2.25 + 0.25 + 0 + 1 + 0 + 0.04 + 0.04 + 0.16 + 0.5625 + 0.09 + 2.56 + 0)$$

$$\frac{1}{28} \times 7.2025$$

$$MSE = 0.2572$$

$$\theta_0 = 1, \theta_1 = 1.5$$

$$h_\theta(x) = \theta_0 + \theta_1 x$$

x	y	Pred_Y
7	11	11.5
2	4	4
17	25	26.5
9	15	14.5
4	7	7
11	16.5	17.5
12	19	19
6	10.2	10
1	2.3	2.5
3	5.1	5.5
2.5	4	4.75
3.8	7	6.7
5.6	11	9.4
14	22	22

Gradient Decent

$$\frac{1}{2m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)})^2 = J(\Theta)$$

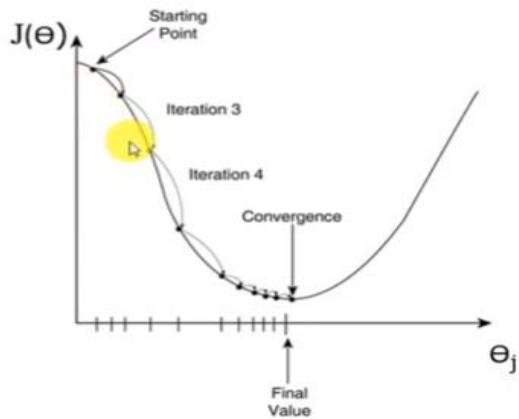
$$\Theta_j = \Theta_j - \alpha \frac{\partial}{\partial \Theta_j} J(\Theta) \quad \alpha = \text{Learning Rate}$$

$$\Theta_j = \Theta_j - \alpha \frac{\partial}{\partial \Theta_j} \times \frac{1}{2m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)})^2$$

$$\Theta_j = \Theta_j - \alpha \frac{1}{m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)})$$

$$\Theta_0 = \Theta_0 - \alpha \frac{1}{m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)})$$

$$\Theta_1 = \Theta_1 - \alpha \frac{1}{m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)}) \times x^{(i)}$$



Example:

1st iteration:

$$\Theta_0 = 1, \Theta_1 = 1$$

$$\therefore h_\theta(x) = \Theta_0 + \Theta_1 x$$

$$h_\theta(7) = \Theta_0 + \Theta_1 x = 1 + 1 \times 7 = 8$$

$$h_\theta(2) = \Theta_0 + \Theta_1 x = 1 + 1 \times 2 = 3$$

$$h_\theta(17) = \Theta_0 + \Theta_1 x = 1 + 1 \times 17 = 18$$

.

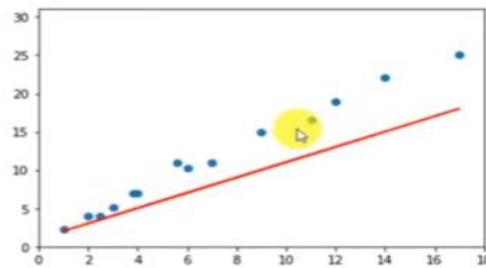
.

.

.

.

$$h_\theta(14) = \Theta_0 + \Theta_1 x = 1 + 1 \times 14 = 15$$



X	Y	Pred_Y
7	11	8
2	4	3
17	25	18
9	15	10
4	7	5
11	16.5	12
12	19	13
6	10.2	7
1	2.3	2
3	5.1	4
2.5	4	3.5
3.8	7	4.8
5.6	11	6.6
14	22	15



1st iteration: $\Theta_0 = 1, \Theta_1 = 1$

$$\Theta_0 = \Theta_0 - \alpha \frac{1}{m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)})$$

$$\frac{1}{14} \left\{ (8 - 11) + (3 - 4) + (18 - 25) + (10 - 15) + (5 - 7) + (12 - 16.5) + (13 - 19) + (7 - 10.2) + (2 - 2.3) + (4 - 5.1) + (3.5 - 4) + (4.8 - 7) + (6.6 - 11) + (15 - 22) \right\}$$

$$\frac{1}{14} (-3 + -1 + -7 + -5 + -2 + -4.5 + -6 + -3.2 + -0.3 + -1.1 + -0.5 + -2.2 + -4.4 + -7)$$

$$\frac{1}{14} \times -47.2 = -3.3714$$

$$\Theta_0 = \Theta_0 - \alpha \times (-3.3714)$$

$$\Theta_0 = 1 - 0.01 \times (-3.3714)$$

$$\Theta_0 = 1.0337$$

1st iteration: $\Theta_0=1, \Theta_1=\frac{1}{14}$

$$\Theta_1 = \Theta_1 - \alpha \frac{1}{m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)}) \times x^{(i)}$$

$$\frac{1}{14} \left\{ (8 - 11) \times 7 + (3 - 4) \times 2 + (18 - 25) \times 17 + (10 - 15) \times 9 + (5 - 7) \times 4 + (12 - 16.5) \times 11 \right.$$

$$+ (13 - 19) \times 12 + (7 - 10.2) \times 6 + (2 - 2.3) \times 1 + (4 - 5.1) \times 3 + (3.5 - 4) \times 2.5 + (4.8 - 7) \times 3.8$$

$$\left. + (6.6 - 11) \times 5.6 + (15 - 22) \times 14 \right\}$$

$$\frac{1}{14} (-21 + -2 + -119 + -45 + -8 + -49.5 + -72 + -19.2 + -0.3 + -3.3 + -1.25 + -8.36 + -24.64 + -98.)$$

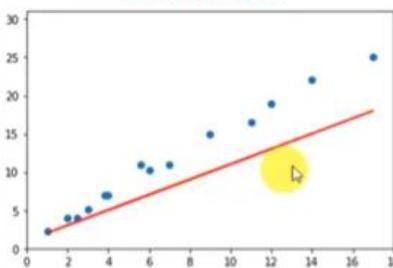
$$\frac{1}{14} \times -471.55 = -33.6821$$

$$\Theta_1 = \Theta_1 - \alpha \times (-33.6821)$$

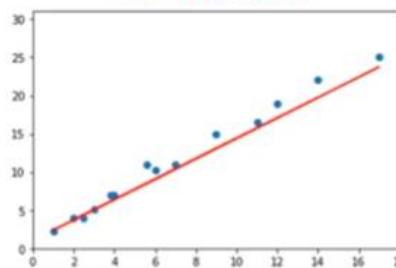
$$\Theta_1 = 1 - 0.01 \times (-33.6821)$$

$$\Theta_1 = 1.336$$

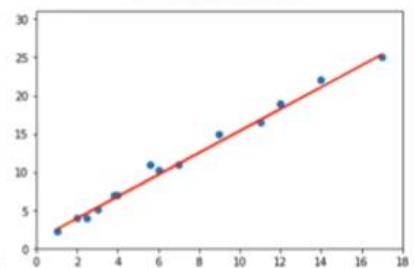
1st iteration:



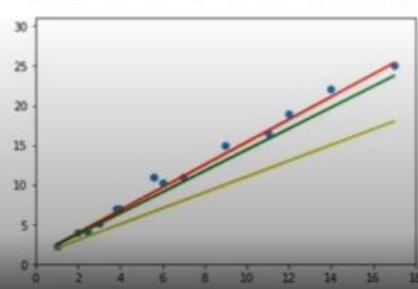
2nd iteration:



3rd iteration:



Compression

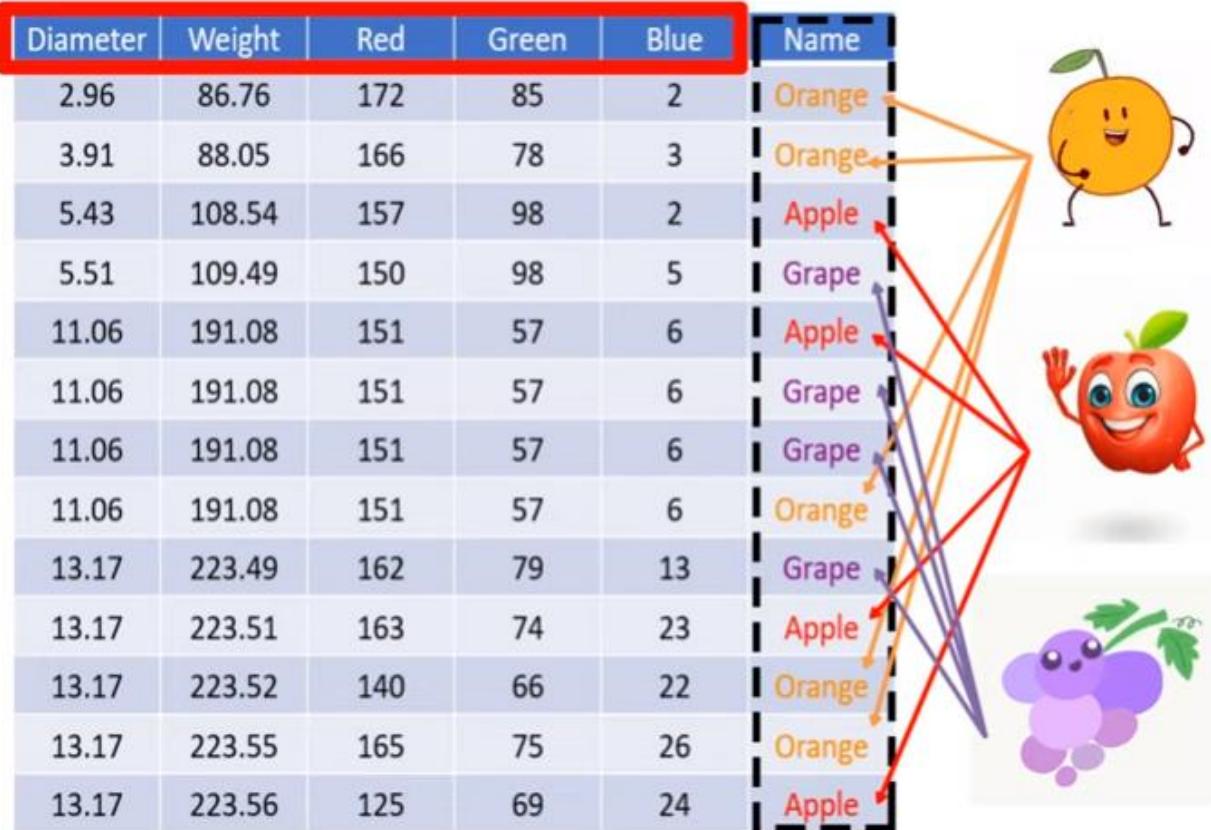


Classification

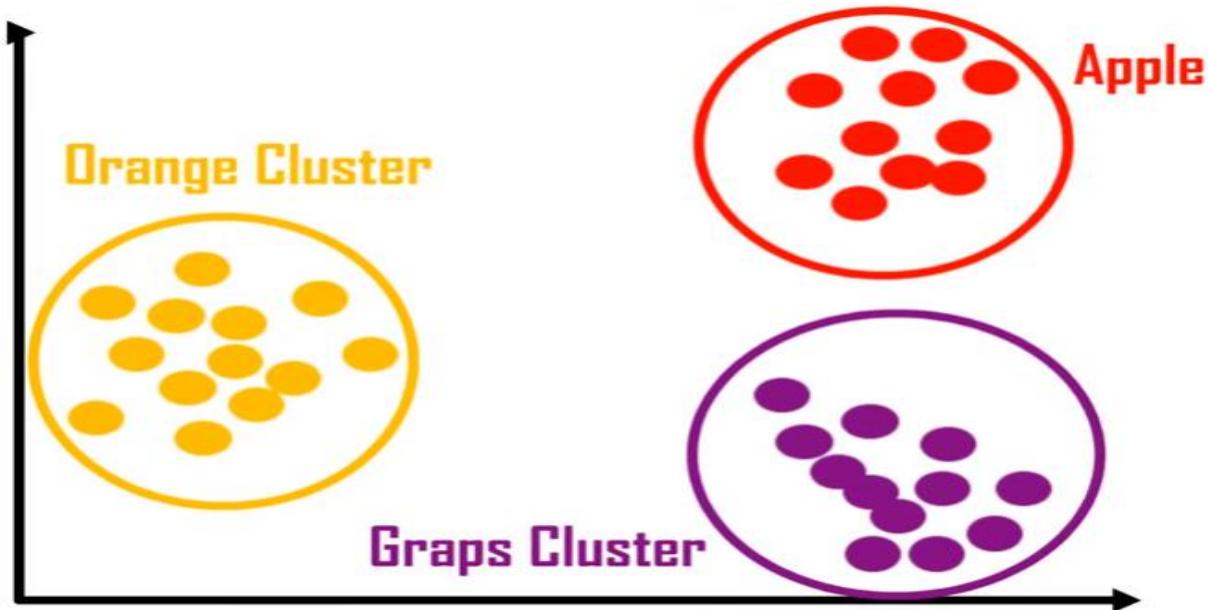
In classification, the output variable is discrete. i.e. we train the hypothesis($f(x)$) in a way to get discrete output(y) for the input data(x). The output can also be termed as a class. For example, by taking the above example of house price, we can use classification to predict whether the house price will be above or below **instead of getting the exact value**. So we have two classes, one if the **price is above** and the other if **it is below**.

In classification problems the output values of target class are discrete. Target value must be belonging to a class. In this example target values are a fruit name

which depends on color, diameter, and weight. In classification we classify our data into classes and predict class for new data.



In classification problem we will separate our data and make classes and our data should looks like this. In this example based on values of x we make clusters for fruits target classes could be 2 or more than 2.



Classification is used in speech recognition, image classification, NLP, etc.

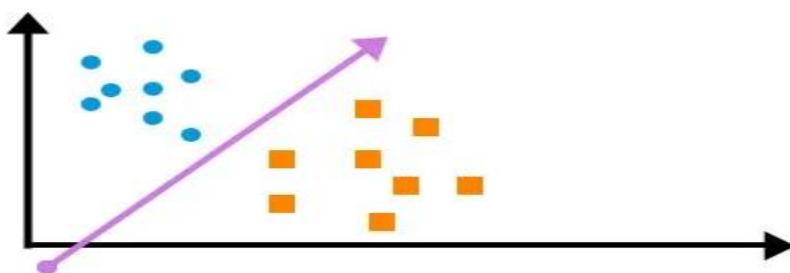
Few Popular Classification Algorithm is:

- Logistic Regression
- Neural Network
- Decision Tree
- Naïve Bayes Classifier

The classification of supervised learning algorithms is used to group similar objects into unique classes.

- **Binary classification** – If the algorithm is trying to group 2 distinct groups of classes, then it is called binary classification.
- **Multiclass classification** – If the algorithm is trying to group objects to more than 2 groups, then it is called multiclass classification.
- **Strength** – Classification algorithms usually perform very well.
- **Drawbacks** – Prone to overfitting and might be unconstrained. For Example – Email Spam classifier

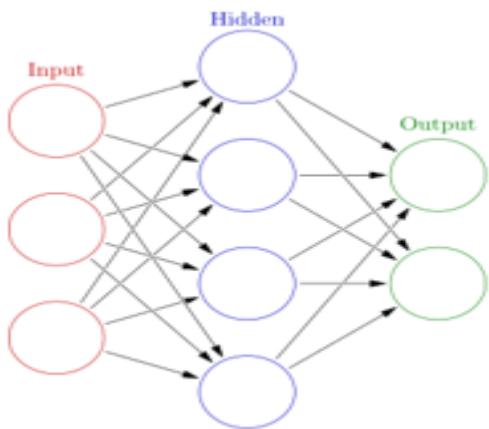
- **Logistic regression/classification** – When the Y variable is a binary categorical (i.e. 0 or 1), we use Logistic regression for the prediction. For Example – Predicting if a given credit card transaction is fraud or not.
- **Naïve Bayes Classifiers** – The Naïve Bayes classifier is based on the Bayesian theorem. This algorithm is usually best suited when the dimensionality of the inputs is high. It consists of acyclic graphs that are having one parent and many children nodes. The child nodes are independent of each other.
- **Decision Trees** – A decision tree is a tree chart like structure that consists of an internal node (test on attribute), a branch that denotes the outcome of the test and the leaf nodes, representing the distribution of classes. The root node is the topmost node. It is a very widely used technique which is used for classification.
- **Support Vector Machine** – In the support vector machine, we use a hyperplane to classify the dependent variable when we have only two dependent variables, i.e. only two classes to predict; then, this hyperplane is nothing but a straight line. The objective of SVM is to get the hyperplane in a way that all the independent variables of one class should be on one side. An optimal SVM function will result in a hyperplane that is at an equal distance from both the class. Fields where SVMs are extensively used, are biometrics, pattern recognition, etc.



Neural Network

The neural network is a classification algorithm that has a minimum of 3 layers. Input – Hidden – Output.

The number of hidden layers may vary based upon the application of the problem. Each hidden layer tries to detect a pattern on the input. As the pattern is detected, it gets forwarded to the other hidden layer in the network till the output layer. Please see the figure below.



The circle in the figure determines the neuron which stores the features (only in the input layer), i.e. the independent variable.

Suppose we have a 32×32 image of a number, then in order to classify the number, we can use a neural network. We will pass the images to the input layers. Since the number of pixels in the image is $32 \times 32 = 1024$, we will have 1024 neurons in the input layers. The number of neurons in hidden layers can be tweaked, but for the output layer, it has to be 10 since, in this example, the number can be anything between 0-9.

The node of the output layers contains the probability. Whichever node has the highest probability that the node is supposed to be the resultant class?

Unsupervised machine learning

Unsupervised Machine Learning is one of the three main techniques of machine learning. It's a self-organized learning algorithm in which we don't need to supervise the data by providing a labelled dataset as it can find a previously unknown pattern in the unlabelled dataset on its own to discover useful

information by performing complex tasks (such as principal component analysis and cluster analysis) as compared to the other machine learning techniques like supervised learning.

Unsupervised learning simply works with the input data. It's essentially ideal for the incoming data going to enable it to be more understandable and organized. Mainly, it studies the input data to discover behavior or commonalities or flaws to your prospects. Possibly considered how Amazon or any type of other online stores can recommend many you can purchase?

This really is because of unsupervised machine learning. Web sites like these consider the prior acquisitions, and they are capable of recommending other activities that you might be thinking about too.

Types of Unsupervised Machine Learning

Unsupervised learning tasks can be broadly divided into 3 categories:

1. Clustering
2. Association rule mining
3. Recommendation system

Clustering

Clustering can be done any data where we do not have the class or label information. We want to group the data such that the observations with similar properties belong to the same cluster/group, and inter-cluster distance should be maximum. At the same time, the intra-cluster distance should be minimum. We can cluster the voter's data to determine the opinion about the government or cluster products based on their features and usage. Segment population based on income features or use clustering in sales and marketing.

We can use K-Means, K-Means++, K-Medoids, Fuzzy C-means (FCM), Expectation-Maximisation (EM), Agglomerative Clustering, DBSCAN, Hierarchical Clustering types as single linkage, complete linkage, median linkage, Ward's method algorithms for clustering.

Association Rule Mining

When we have transactional data for something, it can be for products sold or any transactional data for that matters; I want to know, is there any hidden relationship between buyer and the products or product to product, such that I can somehow leverage this information to increase my sales. Extracting these relationships is the core of Association Rule Mining. We can use the AIS, SETM, Apriori, FP growth algorithms for extracting relationships.

Recommendation System

Recommendation System is basically an extension of Association rule mining in a sense; we are extracting relationships in ARM. In the Recommendation System, we are using these relationships to recommend something which is having higher acceptance chances by the end-user. Recommendation systems have gained popularity after Netflix announced a grand prize of US\$1,000,000 prize in 2009.

Recommendation Systems works on transactional data, be it financial transaction, e-commerce, or grocery shop transactions. Nowadays, giant players in the e-commerce industry are luring customers by making a customized recommendations for each user based on their past purchase history and similar behaviour purchase data from other users.

Methods to develop Recommendation Systems can be broadly divided into Collaborative filtering and Content-Based filtering. In Collaborative filtering, we have user-user Collaborative filtering and Item-Item Collaborative filtering, which are memory-based approaches & Matrix factorization and Singular Value Decomposition (SVD) model-based approaches.

Reinforcement Learning

Reinforcement Learning enables systems to understand depending on previous benefits for its activities. Whenever a system requires a resolution, it can be penalized or honored for its activities. For every action, it should get good feedback, which this discovers if this worked an incorrect or corrective action. This kind of machine learning is usually purely focused on the boosted effectiveness of the function.

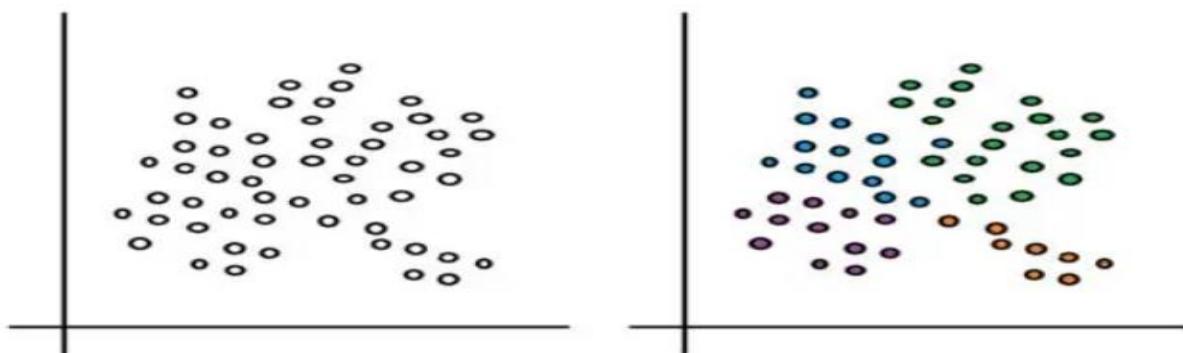
Clustering Methods and Applications

What is Clustering?

Many things around us can be categorized as “this and that” or to be less vague and more specific, we have groupings that could be binary or groups that can be more than two, like a type of pizza base or type of car that you might want to purchase. The choices are always clear – predefined groups and the process predicting that is an important process in the Data Science stack called Classification.

But what if we bring into play a quest where we **don't have pre-defined choices** initially, rather, we derive those choices! **Choices that are based out of hidden patterns, underlying similarities between the constituent variables, salient features from the data** etc. This process is known as Clustering in Machine Learning or Cluster Analysis, where we group the data together into an unknown number of groups and later use that information for further business processes.

So, to put it in simple words, in machine learning clustering is the process by which we create groups in a data, like customers, products, employees, text documents, in such a way that **objects falling into one group exhibit many similar properties with each other** and are different from objects that fall in the other groups that got created during the process.



Clustering algorithms take the data and using some sort of similarity metrics, they form these groups – later these groups can be used in various business processes like information retrieval, pattern recognition, image processing, data compression, bioinformatics etc. In the Machine Learning process for Clustering, as mentioned above, a distance-based similarity metric plays a pivotal role in deciding the clustering.

Types of Clustering Methods

As we made a point earlier that for a successful grouping, we need to attain two major goals: one, a **similarity between one data point with another** and two, a **distinction of those similar data points with others which most certainly, heuristically differ from those points**. The basis of such divisions begins with our ability to scale large datasets and that's a major beginning point for us. Once we are through it, we are presented with a challenge that our data contains different kinds of attributes – categorical, continuous data, etc., and we should be able to deal with them. Now, we know that our data these days is not limited in terms of dimensions, we have data that is multi-dimensional in nature. The clustering algorithm that we intend to use should successfully cross this hurdle as well.

The clusters that we need, should not only be able to distinguish data points but also they should be inclusive. Sure, a distance metric helps a lot but the cluster shape is often limited to being a geometric shape and many important data points get excluded. This problem too needs to be taken care of.

In our progress, we notice that our data is highly “**noisy**” in nature. Many unwanted features have been residing in the data which makes it rather Herculean task to bring about any similarity between the data points – leading to the creation of improper groups. As we move towards the end of the line, we are faced with a challenge of business interpretation. The outputs from the clustering algorithm should be understandable and should fit the business criteria and address the business problem correctly.

To address the problem points above – **scalability, attributes, dimensional, boundary shape, noise, and interpretation** – we have various types of clustering methods that solve one or many of these problems and of course, many statistical and machine learning clustering algorithms that implement the methodology.

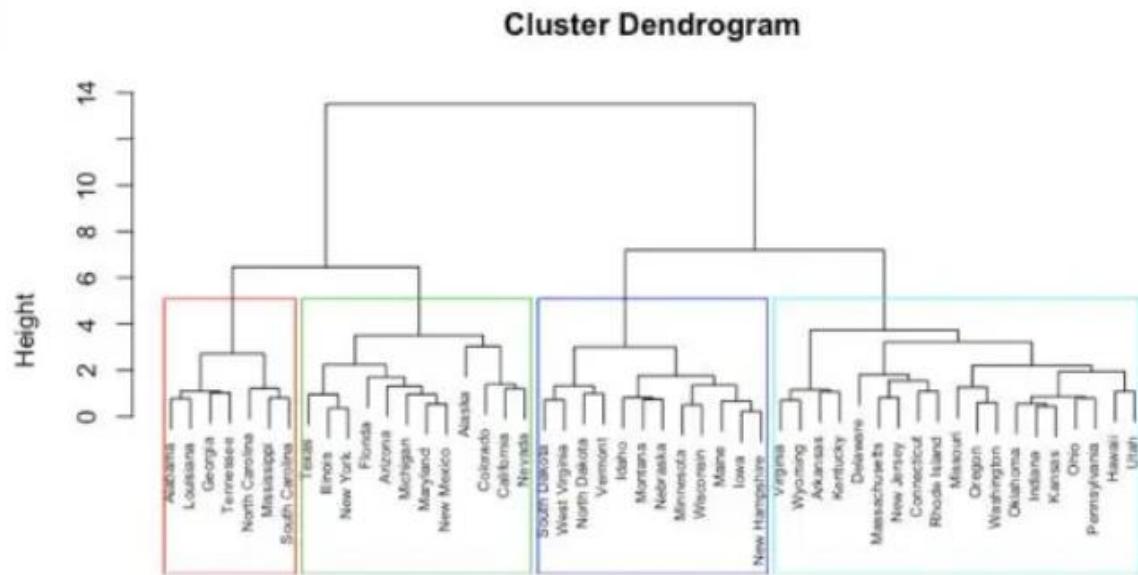
The various types of clustering are:

- Connectivity-based Clustering (Hierarchical clustering)
- Density-based Clustering (Model-based methods)
- Centroids-based Clustering (Partitioning methods)
- Distribution-based Clustering
- Fuzzy Clustering

- Constraint-based (Supervised Clustering)

Connectivity-Based Clustering (Hierarchical Clustering)

Hierarchical Clustering is a method of unsupervised machine learning clustering where it begins with a pre-defined top to bottom hierarchy of clusters. It then proceeds to perform a decomposition of the data objects based on this hierarchy, hence obtaining the clusters. This method follows two approaches based on the direction of progress, i.e., whether it is the top-down or bottom-up flow of creating clusters. These are Divisive Approach and the Agglomerative Approach respectively.

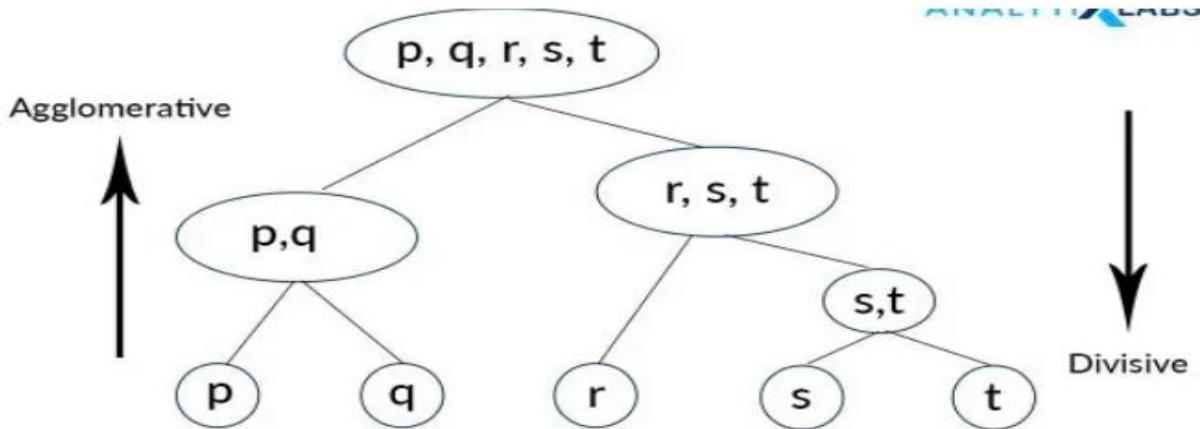


Divisive Approach

This approach of hierarchical clustering follows a top-down approach where we consider that all the data points belong to one large cluster and try to divide the data into smaller groups based on a termination logic or, a point beyond which there will be no further division of data points. This termination logic can be based on the minimum sum of squares of error inside a cluster or for categorical data, the metric can be the GINI coefficient inside a cluster.

Hence, iteratively, we are splitting the data which was once grouped as a single large cluster, to “n” number of smaller clusters in which the data points now belong to.

It must be taken into account that this algorithm is highly “rigid” when splitting the clusters – meaning, once a clustering is done inside a loop, there is no way that the task can be undone.



Agglomerative Approach

Agglomerative is quite the contrary to Divisive, where all the “N” data points are considered to be a single member of “N” clusters that the data is comprised into. We iteratively combine these numerous “N” clusters to fewer number of clusters, let’s say “k” clusters and hence assign the data points to each of these clusters accordingly. This approach is a bottom-up one, and also uses a termination logic in combining the clusters. This logic can be a number based criterion (no more clusters beyond this point) or a distance criterion (clusters should not be too far apart to be merged) or variance criterion (increase in the variance of the cluster being merged should not exceed a threshold, Ward Method)

The Hierarchical clustering Technique can be visualized using a Dendrogram.

A Dendrogram is a tree-like diagram that records the sequences of merges or splits.

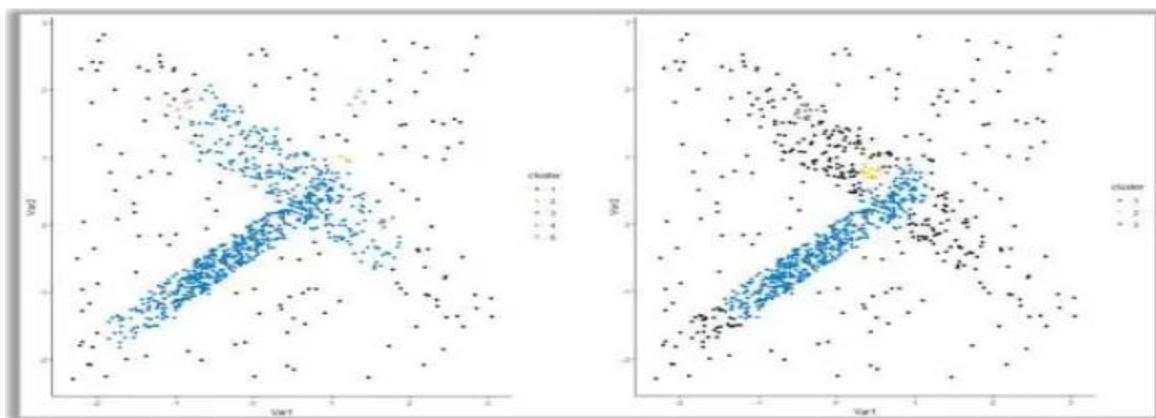
2. Density-based Clustering (Model-based Methods)

If one looks into the previous two methods that we discussed, one would observe that both hierarchical and centroid based algorithms are dependent on a distance (similarity/proximity) metric. The very definition of a cluster is based on this metric. Density-based clustering methods take density into consideration instead of distances. Clusters are considered as the densest region in a data

space, which is separated by regions of lower object density and it is defined as a maximal-set of connected points.

When performing most of the clustering, we take two major assumptions, one, the data is devoid of any noise and two, the shape of the cluster so formed is purely geometrical (circular or elliptical). The fact is, data always has some extent of inconsistency (noise) which cannot be ignored. Added to that, we must not limit ourselves to a fixed attribute shape, it is desirable to have arbitrary shapes so as to not to ignore any data points. These are the areas where density based algorithms have proven their worth!

Density-based algorithms can get us clusters with arbitrary shapes, clusters without any limitation in cluster sizes, clusters that contain the maximum level of homogeneity by ensuring the same levels of density within it, and also these clusters are inclusive of outliers or the noisy data.



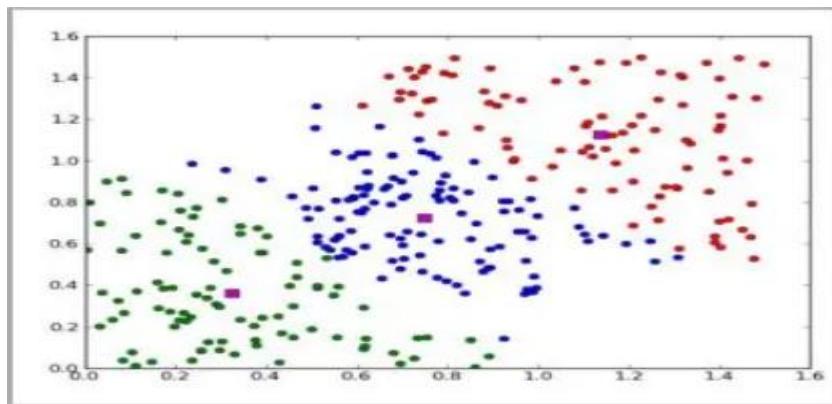
3. Centroid Based Clustering

Centroid based clustering is considered as one of the most simplest clustering algorithms, yet the most effective way of creating clusters and assigning data points to it. The intuition behind centroid based clustering is that a cluster is characterized and represented by a central vector and data points that are in close proximity to these vectors are assigned to the respective clusters.

These groups of clustering methods iteratively measure the distance between the clusters and the characteristic centroids using various distance metrics. These are either of Euclidian distance, Manhattan Distance or Minkowski Distance.

The major setback here is that we should either intuitively or scientifically (Elbow Method) define the number of clusters, “ k ”, to begin the iteration of any clustering machine learning algorithm to start assigning the data points.

Despite the flaws, Centroid based clustering has proven it's worth over Hierarchical clustering when working with large datasets. Also, owing to its simplicity in implementation and also interpretation, these algorithms have wide application areas viz., market segmentation, customer segmentation, text topic retrieval, image segmentation etc.



4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

SEPTEMBER

THURSDAY

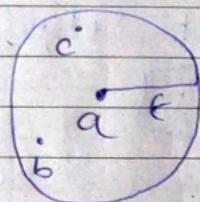
DBSCAN Clustering Algorithm

(Density Based Spatial clustering of Applications with noise)

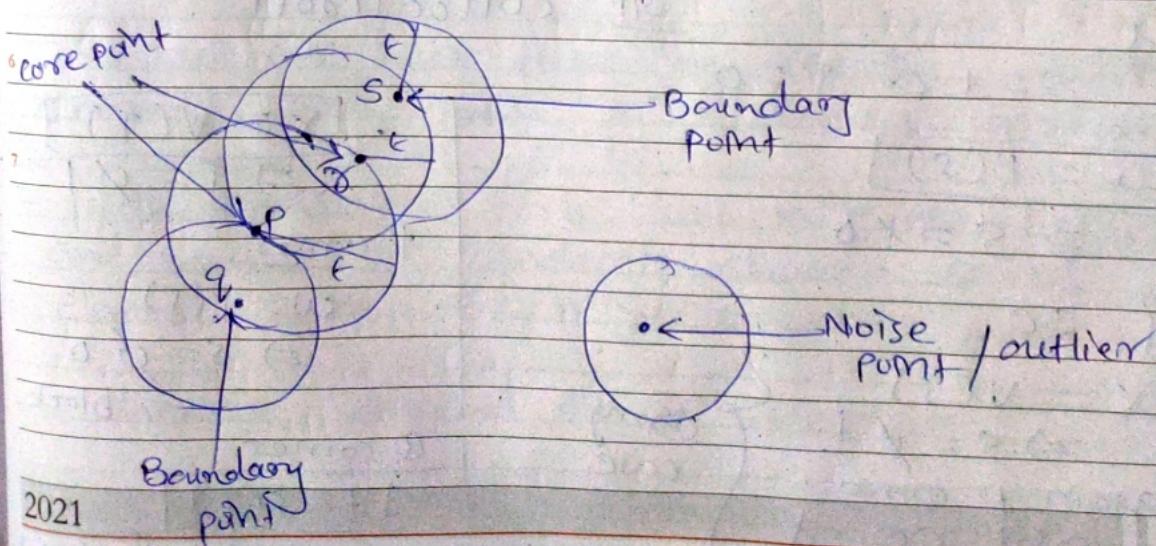
- 1. form cluster based on density
no. of points which are located in a given area
 \uparrow no. of points $\Rightarrow \uparrow$ density

- 2. major input :- ϵ (circum radius)

$\underbrace{\text{minpoints}}_3 = 3$



\downarrow
if $\text{minpoints} = 3$ then
a data point is considered as
 \downarrow
core point



2021

WEDNESDAY

12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30		

Boundary point conditions

1) S point is neighbour of core point

↓

Yes (core point σ)

2) q point is neighbour of core point

↓

Yes (core point p)

⇒ Noise point / outlier (Not a core point / Boundary point)
Advan₂

Robust / identify outliers

↓

if identify then not to include in cluster

⇒ Directly density reachable

q is directly density reachable from point p

1) q must be neighbour of point p \Rightarrow Yes

2) point p must be core point \Rightarrow Yes

1	12	13	14	15	16	17
11	19	20	21	22	23	24
18	26	27	28	29	30	31
2						

SEPTEMBER

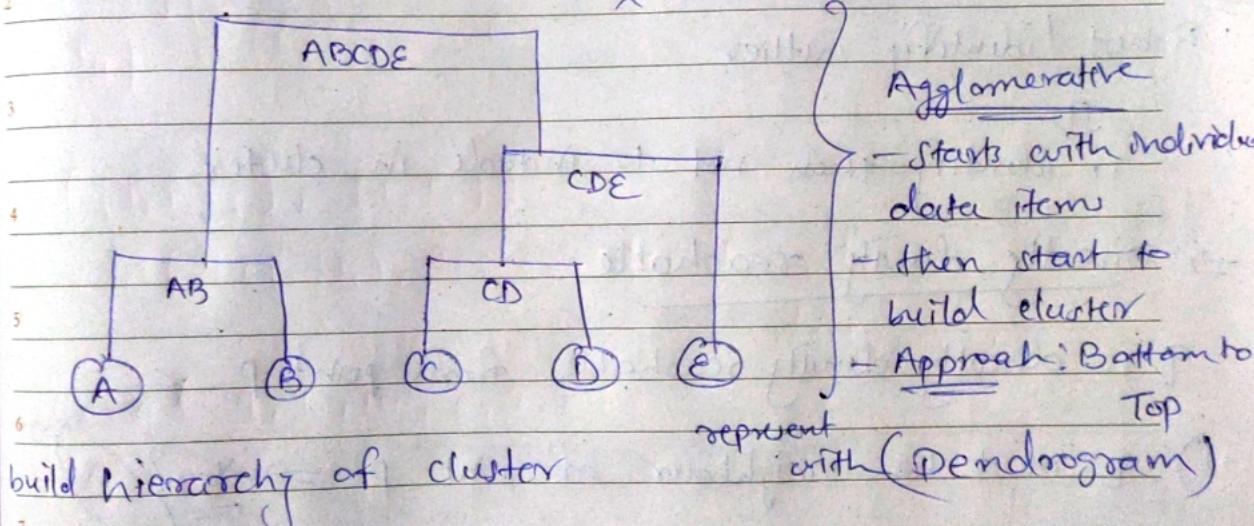
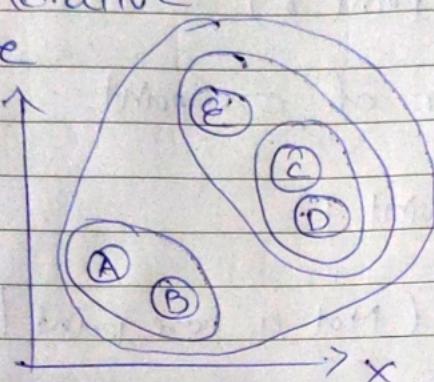
TUESDAY

Hierarchical Clustering

If we want to build hierarchy of cluster

→ Agglomerative

→ Divisive

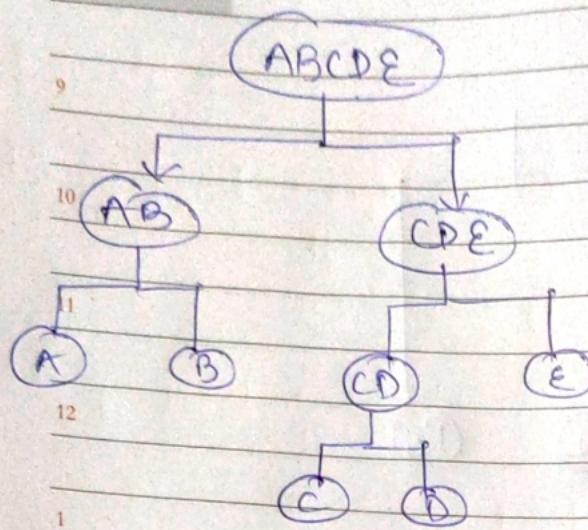


build hierarchy of cluster

⇒ Divisive

- Top to Down Approach follow
- starts with one cluster & that cluster is going to consists of all clusters
- then divide it into no. of clusters

2021



→ Agglomerative clustering (single linkage) $\xrightarrow{\text{min}}$
 distance metric \leftarrow if complete linkage then max

	P ₁	P ₂	P ₃	P ₄	P ₅
P ₁	0				
P ₂	9	0			
P ₃	3	7	0		
P ₄	6	5	9	0	
P ₅	11	10	2	8	0

	P ₁	P ₂	[P ₃ , P ₅]	P ₄
P ₁	0			
P ₂	9	0		
(P ₃ , P ₅)	3	7	0	
P ₄	6	5	8	0

$$d(P_1, [P_3, P_5])$$

$$\Rightarrow \min(d(P_1, P_3), d(P_1, P_5))$$

$$\Rightarrow \min(3, 11) \Rightarrow 3$$

$$d(P_2, [P_3, P_5])$$

$$\Rightarrow \min(d(P_2, P_3), d(P_2, P_5))$$

$$\Rightarrow \min(7, 10) \Rightarrow 7$$

~~$$d(P_4, [P_3, P_5])$$~~

~~$$\Rightarrow \min(d(P_4, P_3), d(P_4, P_5))$$~~

~~$$\Rightarrow \min(9, 8) \Rightarrow 8$$~~



	[P ₁ , P ₃ , P ₅]	P ₂	P ₄
(P ₁ , P ₃ , P ₅)	0		
P ₂	7		
P ₄	6	5	0

$$d(P_2, [P_1, P_3, P_5])$$

$$\Rightarrow \min((d(P_2, P_1), d(P_2, P_3), d(P_2, P_5)))$$

$$\Rightarrow \min(9, 7, 10)$$

2021

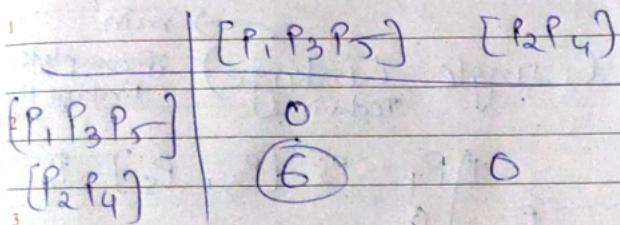
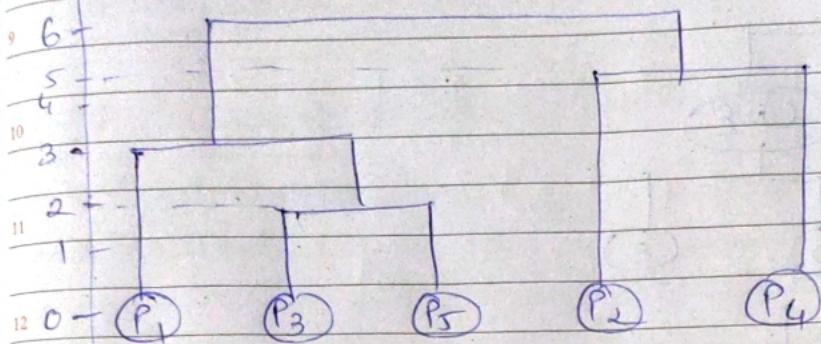
M	5	6	7	1	2	3
4				8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

SEPTEMBER

26

SATURDAY

form dendrogram
cluster through



$$d([P_1, P_3, P_5], [P_2, P_4])$$

$$\Rightarrow \min(d(P_2, P_1), d(P_2, P_3), d(P_2, P_5), d(P_4, P_1), d(P_4, P_3), d(P_4, P_5)) = 6$$

$$\Rightarrow \min(9, 7, 10, 6, 9, 8)$$

SUNDAY

6

PDF Created Using



Camera Scanner

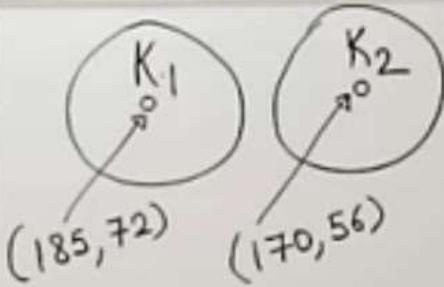
Easily Scan documents & Generate PDF



<https://play.google.com/store/apps/details?id=photo.pdf.maker>

K-means Algorithm

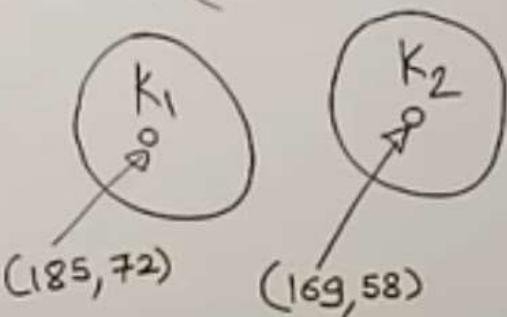
	height	weight
①	185	72
②	170	56
③	168	60
④	179	68
⑤	182	72
⑥	188	77
⑦	180	71
⑧	180	70
⑨	183	84
⑩	180	88
⑪	180	67
⑫	177	76



$$\begin{aligned} \text{ED for } ③ &\rightarrow K_1 \rightarrow \sqrt{(168-185)^2 + (60-72)^2} \\ &\quad \swarrow \\ &\rightarrow K_2 \rightarrow \sqrt{(168-170)^2 + (60-55)^2} \\ &\quad = 4.42 \end{aligned}$$

New Centroid Calculation :-

$$\text{for } K_2 = \left(\frac{170+168}{2}, \frac{60+56}{2} \right) = (169, 58)$$



$$\begin{aligned} \text{ED for } \rightarrow k_1 &= \sqrt{(179-185)^2 + (68-72)^2} \\ (4) &= (6.32) \\ \rightarrow k_2 &= \sqrt{(179-169)^2 + (68-58)^2} \\ &= 14.14 \end{aligned}$$

Euclidean Distance

$$\sqrt{(X_0 - X_c)^2 + (Y_0 - Y_c)^2}$$

$$K_1 \rightarrow \{1, 4, 5, 6, 7, 8, 9, 10, 11, 12\}$$
$$K_2 \rightarrow \{2, 3\}$$