Total No. of Questions : 8] *[Total No. of Printed Pages : 2*

Roll No ....

IT-8003 (2) (CBGS)

B.E. VIII Semester

Examination, May 2019

Choice Based Grading System (CBGS)

Data Science

Time : Three Hours

Maximum Marks : 70

Note: i) Attempt any five questions.

ii) All questions carry equal marks.

1. a) What is Big data? Describe the main features of a data analytical system.

   b) What is Data Science? Also, list the differences between supervised and unsupervised learning.

2. a) Describe in detail about the role of statistical models in Big Data.

   b) What do you understand by the term Normal Distribution?

3. a) Explain the architecture of Hive and how it is useful for data analysis.

   b) What is Hadoop? Explain its infrastructure and HDFS concept.

4. a) Explain Naive Bays classifier with example.

   b) Differentiate Hadoop vs distributed database.

5. a) Show how would you use the sampling distribution and re-sampling

   b) Explain core components of analytical data architecture.

6. a) Explain data manipulation and statistical analysis with R.

   b) Explain data types used in R and loops and conditional statements.

7. a) What is parallel and distributed processing? Explain with example.

   b) Explain the basic building blocks of Hadoop with a neat sketch.

8. Write short notes

   i) Structured and Unstructured Data

   ii) K-Means clustering

   iii) Arrays and Matrices

******

119

1.a	No single definition; here is from Wikipedia: • Big data is the term for a collection of data sets so large and complex that it becomes difficult to

process using on-hand database management tools or traditional data processing applications.
• The challenges include capture, curation, storage, search,sharing, transfer, analysis, and visualization.
• Big data means really a big data, it is a collection of large datasets that cannot be processed using traditional computing techniques. Big data is not only a data, rather it has become a complete subject, which involves various tools, techniques and frameworks.
No single standard definition…
"Big Data" is data whose scale, diversity, and complexity require new architecture, techniques, algorithms, and analytics to manage it and extract value and hidden knowledge from it…
4
Note:- Gartner delivers technology research to global technology business leaders to make informed decisions on key initiatives.
Biggest data research company.

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

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
Types Of Data

Structured data : Relational data.
• Semi Structured data : XML data, JSON.
• Unstructured data : Word, PDF, Text, Media, Logs.

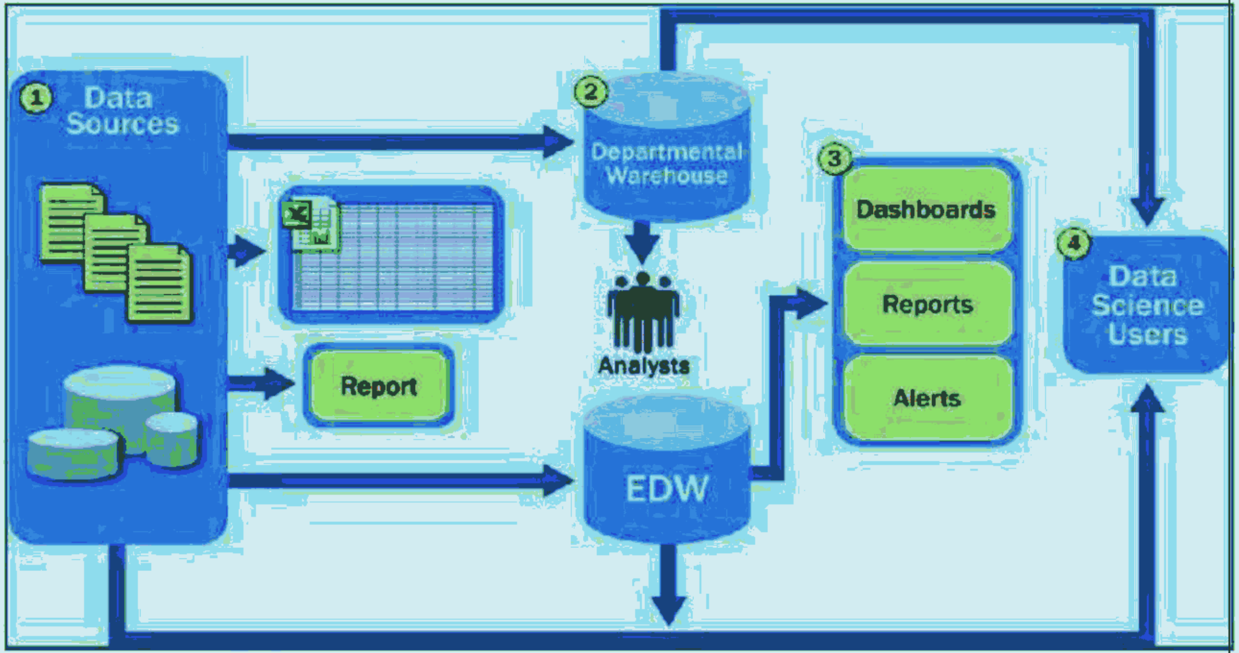
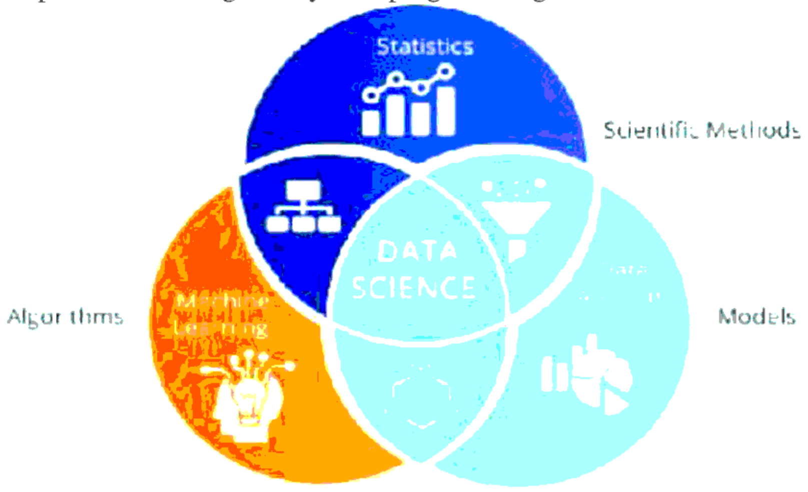


Figure 1.9 Typical analytic architecture

The typical data architectures just described are designed for storing and processing mission-critical data, supporting enterprise applications, and enabling corporate reporting activities. Although reports and dashboards are still important for organizations, most traditional data architectures inhibit data exploration and more sophisticated analysis. Moreover, traditional data architectures have several additional implications for data scientists.

1.b Data Science is a detailed study of the flow of information from the colossal amounts of data present in an organization’s repository. It involves obtaining meaningful insights from raw and unstructured data which is processed through analytical, programming, and business skills



In Supervised learning, you train the machine using data which is well **"labeled."** It means some data is already tagged with the correct answer. It can be compared to learning which takes place in the presence of a supervisor or a teacher.

A supervised learning algorithm learns from labeled training data, helps you to predict outcomes for unforeseen data. Successfully building, scaling, and deploying accurate supervised machine learning Data science model takes time and technical expertise from a team of highly skilled data scientists. Moreover, Data scientist must rebuild models to make sure the insights given remains true until its data changes.

Why supervised learning?
The basic aim is to approximate the mapping function(mentioned above) so well that when there is a new input data (x) then the corresponding output variable can be predicted.
It is called supervised learning because the process of an learning(from the training dataset) can be thought of as a teacher who is supervising the entire learning process. Thus, the “learning algorithm” iteratively makes predictions on the training data and is corrected by the “teacher”, and the learning stops when the algorithm achieves an acceptable level of performance(or the desired accuracy).

Example of Supervised Learning
Suppose there is a basket which is filled with some fresh fruits, the task is to arrange the same type of fruits at one place.
Also, suppose that the fruits are apple, banana, cherry, grape.
Suppose one already knows from their *previous work* (or experience) that, the shape of each and every fruit present in the basket so, it is easy for them to arrange the same type of fruits in one place.
Here, the previous work is called as **training data** in Data Mining terminology. So, it learns the things from the training data. This is because it has a response variable which says y that if some fruit has so and so features then it is grape, and similarly for each and every fruit.
This type of information is deciphered from the data that is used to train the model. This type of learning is called **Supervised Learning**.
Unsupervised Learning: Unsupervised learning is where only the input data (say, X) is present and no corresponding output variable is there.
Why Unsupervised Learning?
The main aim of Unsupervised learning is to model the distribution in the data in order to learn more about the data.It is called so, because there is no correct answer and there is no such teacher(unlike

supervised learning). Algorithms are left to their own devises to discover and present the interesting structure in the data.

Example of Unsupervised Learning

Again, Suppose there is a basket and it is filled with some fresh fruits. The task is to arrange the same type of fruits at one place.

This time there is no information about those fruits beforehand, its the first time that the fruits are being seen or discovered So how to group similar fruits without any prior knowledge about those. First, any physical characteristic of a particular fruit is selected. Suppose *color*.

2.a Statistical model is very help full when we are using big data for specific purpose.

That is called analysis of data. For analyzing big data we apply the many statistical function like meen,median,maxima,minima etc

When analyzing data, it is possible to have a statistical approach. The basic tools that are needed to perform basic analysis are −

- Correlation analysis
- Analysis of Variance
- Hypothesis Testing

When working with large datasets, it doesn't involve a problem as these methods aren't computationally intensive with the exception of Correlation Analysis. In this case, it is always possible to take a sample and the results should be robust.

Previous Page
Next Page

When analyzing data, it is possible to have a statistical approach. The basic tools that are needed to perform basic analysis are −

- Correlation analysis
- Analysis of Variance
- Hypothesis Testing

When working with large datasets, it doesn't involve a problem as these methods aren't computationally intensive with the exception of Correlation Analysis. In this case, it is always possible to take a sample and the results should be robust.

Correlation Analysis

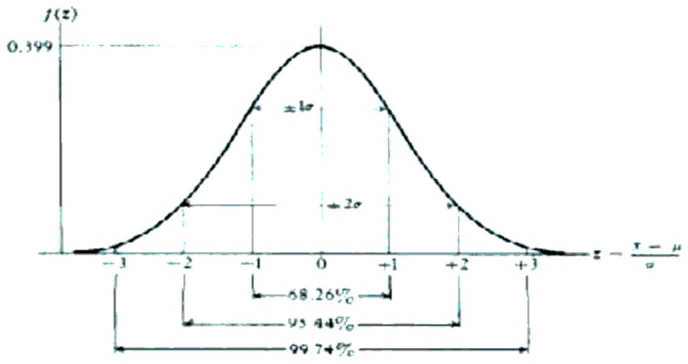
Correlation Analysis seeks to find linear relationships between numeric variables. This can be of use in different circumstances. One common use is exploratory data analysis, in section 16.0.2 of the book there is a basic example of this approach. First of all, the correlation metric used in the mentioned example is based on the **Pearson coefficient**. There is however, another interesting metric of correlation that is not affected by outliers. This metric is called the spearman correlation.

The **spearman correlation** metric is more robust to the presence of outliers than the Pearson method and gives better estimates of linear relations between numeric variable when the data is not normally distributed.

Chi-squared Test

The chi-squared test allows us to test if two random variables are independent. This means that the

	probability distribution of each variable doesn’t influence the other. In order to evaluate the test in R we need first to create a contingency table, and then pass the table to the **chisq.test R** function. For example, let’s check if there is an association between the variables: cut and color from the diamonds dataset. The test is formally defined as − • H0: The variable cut and diamond are independent • H1: The variable cut and diamond are not independent We would assume there is a relationship between these two variables by their name, but the test can give an objective "rule" saying how significant this result is or not. In the following code snippet, we found that the p-value of the test is 2.2e-16, this is almost zero in practical terms. Then after running the test doing a **Monte Carlo simulation**, we found that the p-value is 0.0004998 which is still quite lower than the threshold 0.05. This result means that we reject the null hypothesis (H0), so we believe the variables **cut** and **color** are not independent. T-test The idea of **t-test** is to evaluate if there are differences in a numeric variable # distribution between different groups of a nominal variable. In order to demonstrate this, I will select the levels of the Fair and Ideal levels of the factor variable cut, then we will compare the values a numeric variable among those two groups. The t-tests are implemented in R with the **t.test** function. The formula interface to t.test is the simplest way to use it, the idea is that a numeric variable is explained by a group variable. For example: **t.test(numeric_variable ~ group_variable, data = data)**. In the previous example, the **numeric_variable** is **price** and the **group_variable** is **cut**. From a statistical perspective, we are testing if there are differences in the distributions of the numeric variable among two groups. Formally the hypothesis test is described with a null (H0) hypothesis and an alternative hypothesis (H1). • H0: There are no differences in the distributions of the price variable among the Fair and Ideal groups • H1 There are differences in the distributions of the price variable among the Fair and Ideal groups
2.b	Normal distribution The normal distribution is the most widely known and used of all distributions. Because the normal distribution approximates many natural phenomena so well, it has developed into a standard of reference for many probability problems.



Characteristics of the Normal distribution

• Symmetric, bell shaped • Continuous for all values of X between -∞ and ∞ so that each conceivable interval of real numbers has a probability other than zero. • $-\infty \le X \le \infty$ • Two parameters, μ and σ. Note that the normal distribution is actually a family of distributions, since μ and σ determine the shape of the distribution. • The rule for a normal density function is

$$f(x;\mu,\sigma^2)=\frac{1}{\sqrt{2\pi\sigma^2}}e^{-(x-\mu)^2/2\sigma^2}$$

- The notation $N(\mu, \sigma^2)$ means normally distributed with mean μ and variance σ^2. If we say $X \sim N(\mu, \sigma^2)$ we mean that X is distributed $N(\mu, \sigma^2)$.
- About 2/3 of all cases fall within one standard deviation of the mean, that is

$$P(\mu - \sigma \le X \le \mu + \sigma) = .6826.$$

- About 95% of cases lie within 2 standard deviations of the mean, that is

$$P(\mu - 2\sigma \le X \le \mu + 2\sigma) = .9544$$

II. Why is the normal distribution useful?

• Many things actually are normally distributed, or very close to it. For example, height and intelligence are approximately normally distributed; measurement errors also often have a normal distribution

• The normal distribution is easy to work with mathematically. In many practical cases, the methods developed using normal theory work quite well even when the distribution is not normal. • There is a very strong connection between the size of a sample N and the extent to which a sampling distribution approaches the normal form. Many sampling distributions based on large N can be approximated by the normal distribution even though the population distribution itself is definitely not normal.

The standardized normal distribution. a. General Procedure. As you might suspect from the formula for the normal density function, it would be difficult and tedious to do the calculus every time we had a new set of parameters for μ and σ. So instead, we usually work with the standardized normal distribution, where μ = 0 and σ = 1, i.e. N(0,1). That is, rather than directly solve a problem involving a normally distributed variable X with mean μ and standard deviation σ, an indirect approach is used.

1. We first convert the problem into an equivalent one dealing with a normal variable measured in standardized deviation units, called a standardized normal variable. To do this, if X ~ N(μ, σ²), then

$$Z = \frac{X - \mu}{\sigma} \sim N(0, 1)$$

2. A table of standardized normal values (Appendix E, Table I) can then be used to obtain an answer in terms of the converted problem.

3. If necessary, we can then convert back to the original units of measurement. To do this, simply note that, if we take the formula for Z, multiply both sides by σ, and then add μ to both sides, we get

$$X = Z\sigma + \mu$$

4. The interpetation of Z values is straightforward. Since σ = 1, if Z = 2, the corresponding X value is exactly 2 standard deviations above the mean. If Z = -1, the corresponding X value is one standard deviation below the mean. If Z = 0, X = the mean, i.e. μ.

b. Rules for using the standardized normal distribution. It is very important to understand how the standardized normal distribution works, so we will spend some time here going over it. Recall that, for a random variable X,

$F(x) = P(X \le x)$

3.a

What is Hive?

Hive is an ETL and Data warehousing tool developed on top of Hadoop Distributed File System (HDFS). Hive makes job easy for performing operations like

- Data encapsulation
- Ad-hoc queries
- Analysis of huge datasets

Important characteristics of Hive

- In Hive, tables and databases are created first and then data is loaded into these tables.
- Hive as data warehouse designed for managing and querying only structured data that is stored in tables.
- While dealing with structured data, Map Reduce doesn't have optimization and usability features like UDFs but Hive framework does. Query optimization refers to an effective way of query execution in terms of performance.
- Hive's SQL-inspired language separates the user from the complexity of Map Reduce programming. It reuses familiar concepts from the relational database world, such as tables, rows, columns and schema, etc. for ease of learning.
- Hadoop's programming works on flat files. So, Hive can use directory structures to "partition" data to improve performance on certain queries.
- A new and important component of Hive i.e. Metastore used for storing schema information. This Metastore typically resides in a relational database. We can interact with Hive using methods like
  - **Web GUI**
  - **Java Database Connectivity (JDBC) interface**
- Most interactions tend to take place over a command line interface (CLI). Hive provides a CLI to write Hive queries using Hive Query Language(HQL)

- Generally, HQL syntax is similar to the SQL syntax that most data analysts are familiar with. The Sample query below display all the records present in mentioned table name.
    - **Sample query** : Select * from <TableName>
- Hive supports four file formats those are **TEXTFILE, SEQUENCEFILE, ORC and RCFILE** (Record Columnar File).
- For single user metadata storage, Hive uses derby database and for multiple user Metadata or shared Metadata case Hive uses MYSQL.

For setting up MySQL as database and to store Meta-data information check Tutorial "Installation and Configuration of HIVE and MYSQL"

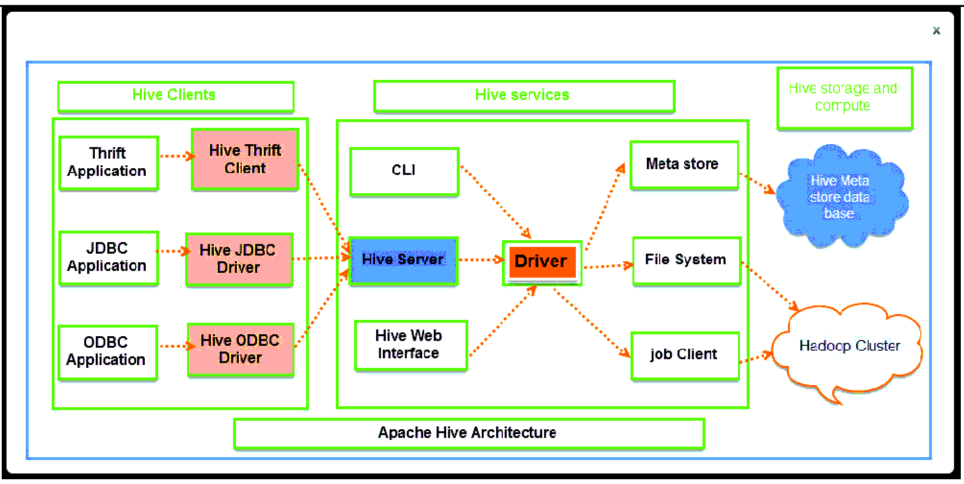
Some of the key points about Hive:

- The major difference between HQL and SQL is that Hive query executes on Hadoop's infrastructure rather than the traditional database.
- The Hive query execution is going to be like series of automatically generated map reduce Jobs.
- Hive supports partition and buckets concepts for easy retrieval of data when the client executes the query.
- Hive supports custom specific UDF (User Defined Functions) for data cleansing, filtering, etc. According to the requirements of the programmers one can define Hive UDFs.

Hive Vs Relational Databases:-

By using Hive, we can perform some peculiar functionality that is not achieved in Relational Databases. For a huge amount of data that is in peta-bytes, querying it and getting results in seconds is important. And Hive does this quite efficiently, it processes the queries fast and produce results in second's time.

Hive Architecture



Hive Consists of Mainly 3 core parts

1. **Hive Clients**
2. **Hive Services**
3. **Hive Storage and Computing**

Hive Clients:

Hive provides different drivers for communication with a different type of applications. For Thrift based applications, it will provide Thrift client for communication.

For Java related applications, it provides JDBC Drivers. Other than any type of applications provided ODBC drivers. These Clients and drivers in turn again communicate with Hive server in the Hive services.

Hive Services:

Client interactions with Hive can be performed through Hive Services. If the client wants to perform any query related operations in Hive, it has to communicate through Hive Services.

CLI is the command line interface acts as Hive service for DDL (Data definition Language) operations. All drivers communicate with Hive server and to the main driver in Hive services as shown in above architecture diagram.

Driver present in the Hive services represents the main driver, and it communicates all type of JDBC, ODBC, and other client specific applications. Driver will process those requests from different applications to meta store and field systems for further processing.

Hive Storage and Computing:

Hive services such as Meta store, File system, and Job Client in turn communicates with Hive storage and performs the following actions

- Metadata information of tables created in Hive is stored in Hive "Meta storage database".
- Query results and data loaded in the tables are going to be stored in Hadoop cluster on HDFS.

The data flow in Hive behaves in the following pattern;

1. Executing Query from the UI(User Interface)
2. The driver is interacting with Compiler for getting the plan. (Here plan refers to query execution) process and its related metadata information gathering
3. The compiler creates the plan for a job to be executed. Compiler communicating with Meta store for getting metadata request
4. Meta store sends metadata information back to compiler
5. Compiler communicating with Driver with the proposed plan to execute the query
6. Driver Sending execution plans to Execution engine
7. Execution Engine (EE) acts as a bridge between Hive and Hadoop to process the query. For DFS operations.

- EE should first contacts Name Node and then to Data nodes to get the values stored in tables.
- EE is going to fetch desired records from Data Nodes. The actual data of tables resides in data node only. While from Name Node it only fetches the metadata information for the query.
- It collects actual data from data nodes related to mentioned query
- Execution Engine (EE) communicates bi-directionally with Meta store present in Hive to perform DDL (Data Definition Language) operations. Here DDL operations like CREATE, DROP and ALTERING tables and databases are done. Meta store will store information about database name, table names and column names only. It will fetch data related to query mentioned.
- Execution Engine (EE) in turn communicates with Hadoop daemons such as Name node, Data nodes, and job tracker to execute the query on top of Hadoop file system

8. Fetching results from driver
9. Sending results to Execution engine. Once the results fetched from data nodes to the EE, it will send results back to driver and to UI (front end)

Hive Continuously in contact with Hadoop file system and its daemons via Execution engine. The dotted arrow in the Job flow diagram shows the Execution engine communication with Hadoop daemons.

Different modes of Hive

Hive can operate in two modes depending on the size of data nodes in Hadoop.

These modes are,

- **Local mode**
- **Map reduce mode**

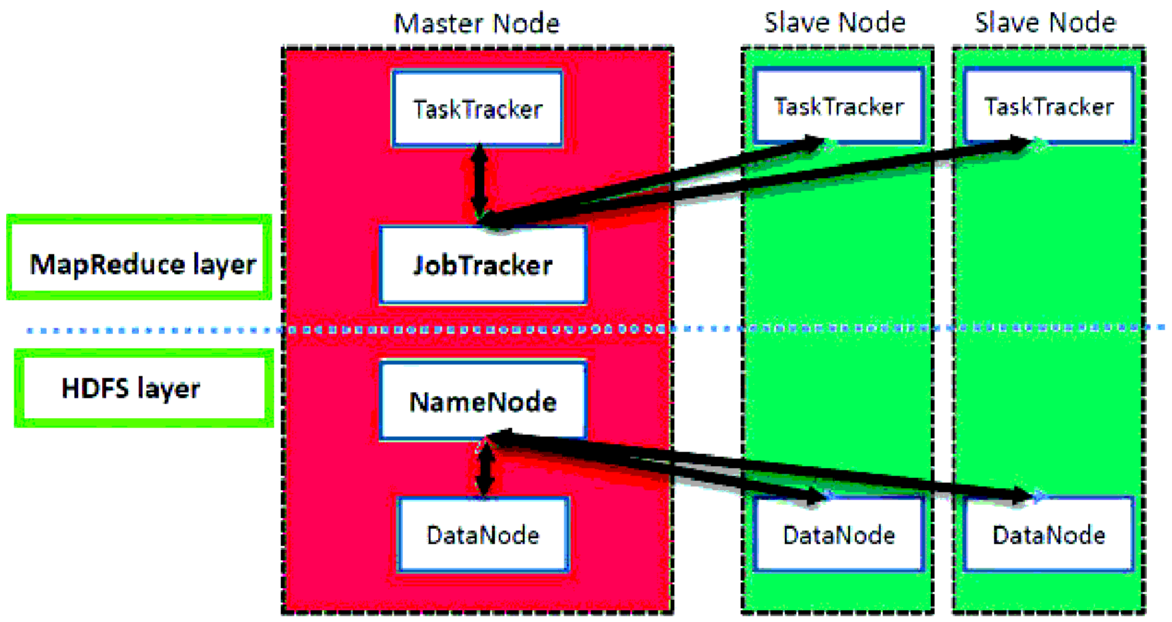
When to use Local mode:

- If the Hadoop installed under pseudo mode with having one data node we use Hive in this mode
- If the data size is smaller in term of limited to single local machine, we can use this mode
- Processing will be very fast on smaller data sets present in the local machine

When to use Map reduce mode:

- If Hadoop is having multiple data nodes and data is distributed across different node we use Hive in this mode
- It will perform on large amount of data sets and query going to execute in parallel way

	• Processing of large data sets with better performance can be achieved through this mode In Hive, we can set this property to mention which mode Hive can work? By default, it works on Map Reduce mode and for local mode you can have the following setting.
3.b	**What is Hadoop?** Apache Hadoop is an open source software framework used to develop data processing applications which are executed in a distributed computing environment. Applications built using HADOOP are run on large data sets distributed across clusters of commodity computers. Commodity computers are cheap and widely available. These are mainly useful for achieving greater computational power at low cost. Similar to data residing in a local file system of a personal computer system, in Hadoop, data resides in a distributed file system which is called as a **Hadoop Distributed File system**. The processing model is based on **'Data Locality'** concept wherein computational logic is sent to cluster nodes(server) containing data. This computational logic is nothing, but a compiled version of a program written in a high-level language such as Java. Such a program, processes data stored in Hadoop HDFS. Hadoop consists of two sub-projects – 1. **Hadoop MapReduce:** MapReduce is a computational model and software framework for writing applications which are run on Hadoop. These MapReduce programs are capable of processing enormous data in parallel on large clusters of computation nodes. 2. **HDFS** (**Hadoop Distributed File System**): HDFS takes care of the storage part of Hadoop applications. MapReduce applications consume data from HDFS. HDFS creates multiple replicas of data blocks and distributes them on compute nodes in a cluster. This distribution enables reliable and extremely rapid computations. **Hadoop Architecture**



Hadoop has a Master-Slave Architecture for data storage and distributed data processing using MapReduce and HDFS methods.

NameNode:

NameNode represented every files and directory which is used in the namespace

DataNode:

DataNode helps you to manage the state of an HDFS node and allows you to interacts with the blocks

MasterNode:

The master node allows you to conduct parallel processing of data using Hadoop MapReduce.

Slave node:

The slave nodes are the additional machines in the Hadoop cluster which allows you to store data to conduct complex calculations. Moreover, all the slave node comes with Task Tracker and a DataNode. This allows you to synchronize the processes with the NameNode and Job Tracker respectively.

In Hadoop, master or slave system can be set up in the cloud or on-premise

Features Of 'Hadoop'

• Suitable for Big Data Analysis

As Big Data tends to be distributed and unstructured in nature, HADOOP clusters are best suited for analysis of Big Data. Since it is processing logic (not the actual data) that flows to the computing nodes, less network bandwidth is consumed. This concept is called as **data locality concept** which helps increase the efficiency of Hadoop based applications.

• Scalability

HADOOP clusters can easily be scaled to any extent by adding additional cluster nodes and thus allows for the growth of Big Data. Also, scaling does not require modifications to application logic.

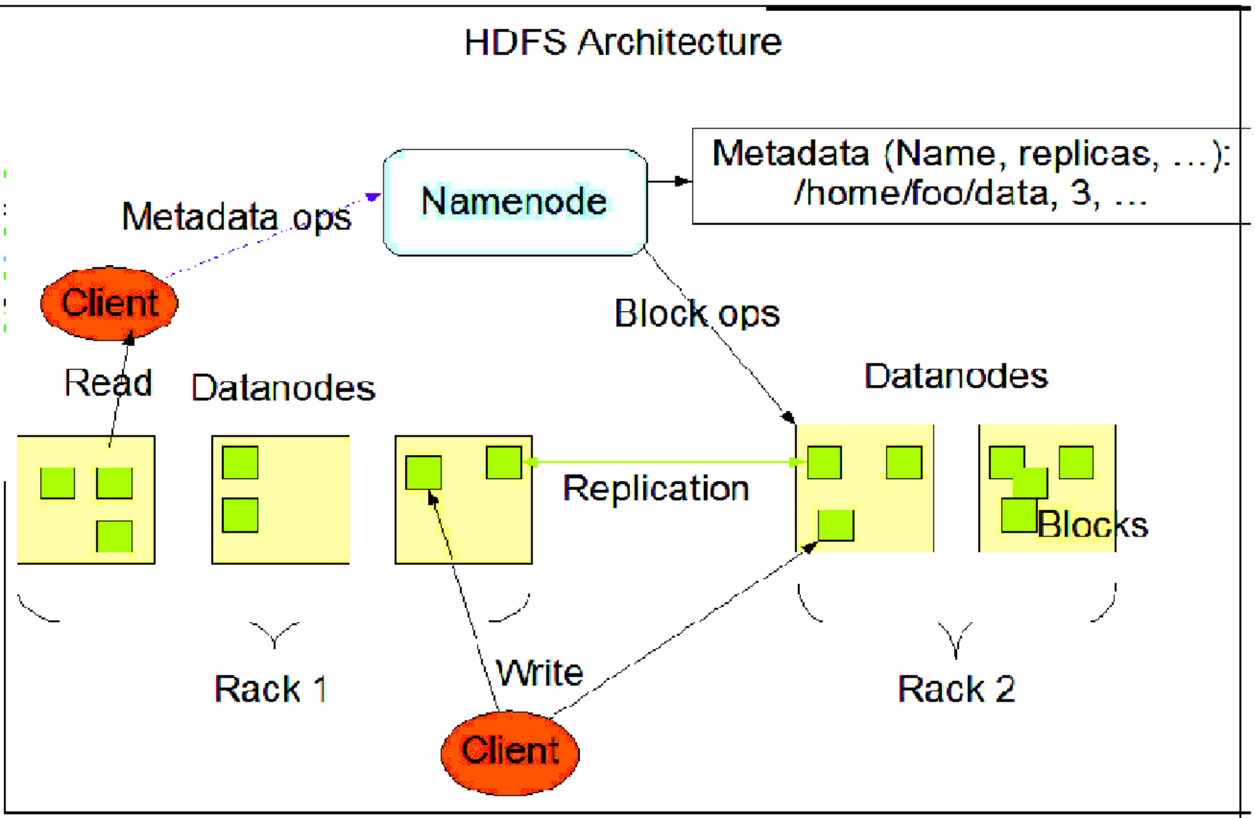
• Fault Tolerance

HADOOP ecosystem has a provision to replicate the input data on to other cluster nodes. That way, in the event of a cluster node failure, data processing can still proceed by using data stored on another cluster node.

Hadoop Distributed File System (HDFS)

The Hadoop Distributed File System (HDFS) is a distributed file system designed to run on commodity hardware. It has many similarities with existing distributed file systems. However, the differences from other distributed file systems are significant. HDFS is highly fault-tolerant and is designed to be deployed on low-cost hardware. HDFS provides high throughput access to application data and is suitable for applications that have large data sets. HDFS relaxes a few POSIX requirements to enable streaming access to file system data. HDFS was originally built as infrastructure for the Apache Nutch web search engine project. HDFS is part of the Apache Hadoop Core project.

HDFS has a master/slave architecture. An HDFS cluster consists of a single NameNode, a master server that manages the file system namespace and regulates access to files by clients. In addition, there are a number of DataNodes, usually one per node in the cluster, which manage storage attached to the nodes that they run on. HDFS exposes a file system namespace and allows user data to be stored in files. Internally, a file is split into one or more blocks and these blocks are stored in a set of DataNodes. The NameNode executes file system namespace operations like opening, closing, and renaming files and directories. It also determines the mapping of blocks to DataNodes. The DataNodes are responsible for serving read and write requests from the file system's clients. The DataNodes also perform block creation, deletion, and replication upon instruction from the NameNode.

	 HDFS Architecture Metadata ops Namenode Metadata (Name, replicas, …): /home/foo/data, 3, … Client Block ops Read Datanodes Datanodes Replication Blocks Rack 1 Write Rack 2 Client The NameNode and DataNode are pieces of software designed to run on commodity machines. These machines typically run a GNU/Linux operating system (OS). HDFS is built using the Java language; any machine that supports Java can run the NameNode or the DataNode software. Usage of the highly portable Java language means that HDFS can be deployed on a wide range of machines. A typical deployment has a dedicated machine that runs only the NameNode software. Each of the other machines in the cluster runs one instance of the DataNode software. The architecture does not preclude running multiple DataNodes on the same machine but in a real deployment that is rarely the case. The existence of a single NameNode in a cluster greatly simplifies the architecture of the system. The NameNode is the arbitrator and repository for all HDFS metadata. The system is designed in such a way that user data never flows through the NameNode.
4.a	What is classifier? A classifier is a machine learning model that is used to discriminate different objects based on certain features. A Naive Bayes classifier is a probabilistic machine learning model that's used for classification task. The crux of the classifier is based on the Bayes theorem. Bayes Theorems $P(A\|B) = \frac{P(B\|A)P(A)}{P(B)}$ Using Bayes theorem, we can find the probability of **A** happening, given that **B** has occurred. Here, **B** is

	the evidence and **A** is the hypothesis. The assumption made here is that the predictors/features are independent. That is presence of one particular feature does not affect the other. Hence it is called naive. **Types of Naive Bayes Classifier:** **Multinomial Naive Bayes:** This is mostly used for document classification problem, i.e whether a document belongs to the category of sports, politics, technology etc. The features/predictors used by the classifier are the frequency of the words present in the document. **Bernoulli Naive Bayes:** This is similar to the multinomial naive bayes but the predictors are boolean variables. The parameters that we use to predict the class variable take up only values yes or no, for example if a word occurs in the text or not. **Gaussian Naive Bayes:** When the predictors take up a continuous value and are not discrete, we assume that these values are sampled from a gaussian distribution. Since the way the values are present in the dataset changes, the formula for conditional probability changes to, $P(x_i\|y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} exp\left(-\frac{(x_i-\mu_y)^2}{2\sigma_y^2}\right)$
4.b	There are a lot of differences between Hadoop and RDBMS(Relational Database Management System). Hadoop is not a database, it is basically a distributed file system which is used to process and store large data sets across the computer cluster. If you are interested to Learn Big Data Hadoop you may join Our Hadoop training program to enhance your skills or you can start a career in this field. It has two main core components HDFS(Hadoop Distributed File System) and MapReduce. HDFS is the storage layer which is used to store a large amount of data across computer clusters. MapReduce is a programming model that processes the large data sets by splitting them into several blocks of data. These blocks are then distributed across the nodes on different machines present inside the computer cluster. On the other hand, RDBMS is a database which is used to store data in the form of tables comprising of several rows and columns. It uses SQL, Structured Query Language, to update and access the data present in these tables.If you want to start your career with hadoop then become part of our advanced Hadoop training program. Now that you have understood the basic difference between Hadoop and RDBMS, let's go through the major working difference between the two. Difference Between Hadoop And Traditional RDBMS

Like Hadoop, traditional RDBMS cannot be used when it comes to process and store a large amount of data or simply big data. Following are some differences between Hadoop and traditional RDBMS.

Data Volume-

Data volume means the quantity of data that is being stored and processed. RDBMS works better when the volume of data is low(in Gigabytes). But when the data size is huge i.e, in Terabytes and Petabytes, RDBMS fails to give the desired results.

On the other hand, Hadoop works better when the data size is big. It can easily process and store large amount of data quite effectively as compared to the traditional RDBMS.

Architecture-

If we talk about the architecture, Hadoop has the following core components:

HDFS(Hadoop Distributed File System), Hadoop MapReduce(a programming model to process large data sets) and Hadoop YARN(used to manage computing resources in computer clusters).

Traditional RDBMS possess ACID properties which are Atomicity, Consistency, Isolation, and Durability.

These properties are responsible to maintain and ensure data integrity and accuracy when a transaction takes place in a database.

These transactions may be related to Banking Systems, Manufacturing Industry, Telecommunication industry, Online Shopping, education sector etc.

Throughput-

Throughput means the total volume of data processed in a particular period of time so that the output is maximum. RDBMS fails to achieve a higher throughput as compared to the Apache Hadoop Framework.

This is one of the reason behind the heavy usage of Hadoop than the traditional Relational Database Management System.

Data Variety-

Data Variety generally means the type of data to be processed. It may be structured, semi-structured and unstructured.

Hadoop has the ability to process and store all variety of data whether it is structured, semi-structured or unstructured. Although, it is mostly used to process large amount of unstructured data.

Traditional RDBMS is used only to manage structured and semi-structured data. It cannot be used to manage unstructured data. So we can say Hadoop is way better than the traditional Relational Database Management System.

Latency/ Response Time –

Hadoop has higher throughput, you can quickly access batches of large data sets than traditional RDBMS, but you cannot access a particular record from the data set very quickly. Thus Hadoop is said to have low latency.

But the RDBMS is comparatively faster in retrieving the information from the data sets. It takes a very little time to perform the same function provided that there is a small amount of data.

Scalability-

RDBMS provides vertical scalability which is also known as 'Scaling Up' a machine. It means you can add more resources or hardwares such as memory, CPU to a machine in the computer cluster.

Whereas, Hadoop provides horizontal scalability which is also known as 'Scaling Out' a machine. It means adding more machines to the existing computer clusters as a result of which Hadoop becomes a fault tolerant. There is no single point of failure. Due to the presence of more machines in the cluster, you can easily recover data irrespective of the failure of one of the machines.

Data Processing-

Apache Hadoop supports OLAP(Online Analytical Processing), which is used in Data Mining techniques.

OLAP involves very complex queries and aggregations. The data processing speed depends on the amount of data which can take several hours. The database design is de-normalized having fewer tables. OLAP uses star schemas.

On the other hand, RDBMS supports OLTP(Online Transaction Processing), which involves comparatively fast query processing. The database design is highly normalized having a large number of tables. OLTP generally uses 3NF(an entity model) schema.

Cost-

Hadoop is a free and open source software framework, you don't have to pay in order to buy the license of the software.

Whereas RDBMS is a licensed software, you have to pay in order to buy the complete software license.

5.a

Data sampling is a statistical analysis technique used to select, manipulate and analyze a representative subset of data points to identify patterns and trends in the larger data set being examined. It enables data scientists, predictive modelers and other data analysts to work with a small, manageable amount of data about a statistical population to build and run analytical models more quickly, while still producing accurate findings.

Advantages and challenges of data sampling

Sampling can be particularly useful with data sets that are too large to efficiently analyze in full -- for example, in big data analytics applications or surveys. Identifying and analyzing a representative sample is more efficient and cost-effective than surveying the entirety of the data or population.

An important consideration, though, is the size of the required data sample and the possibility of introducing a sampling error. In some cases, a small sample can reveal the most important information about a data set. In others, using a larger sample can increase the likelihood of accurately representing the data as a whole, even though the increased size of the sample may impede ease of manipulation and interpretation.

Types of data sampling methods

There are many different methods for drawing samples from data; the ideal one depends on the data set and situation. Sampling can be based on probability, an approach that uses random numbers that correspond to points in the data set to ensure that there is no correlation between points chosen for the sample. Further variations in probability sampling include:

- **Simple random sampling:** Software is used to randomly select subjects from the whole population.
- **Stratified sampling:** Subsets of the data sets or population are created based on a common factor, and samples are randomly collected from each subgroup.
- **Cluster sampling:** The larger data set is divided into subsets (*clusters*) based on a defined factor, then a random sampling of clusters is analyzed.
- **Multistage sampling:** A more complicated form of cluster sampling, this method also involves dividing the larger population into a number of clusters. Second-stage clusters are then broken out based on a secondary factor, and those clusters are then sampled and analyzed. This *staging* could continue as multiple subsets are identified, clustered and analyzed.
- **Systematic sampling:** A sample is created by setting an interval at which to extract data from the larger population -- for example, selecting every 10th row in a spreadsheet of 200 items to create a sample size of 20 rows to analyze.

Sampling can also be based on nonprobability, an approach in which a data sample is determined and extracted based on the judgment of the analyst. As inclusion is determined by the analyst, it can be more difficult to extrapolate whether the sample accurately represents the larger population than when probability sampling is used.

Data sampling is a statistical analysis technique used to select, manipulate and analyze a representative subset of data points to identify patterns and trends in the larger data set being examined. It enables data scientists, predictive modelers and other data analysts to work with a small, manageable amount of data about a statistical population to build and run analytical models more quickly, while still producing accurate findings.

Advantages and challenges of data sampling

Sampling can be particularly useful with data sets that are too large to efficiently analyze in full -- for example, in big data analytics applications or surveys. Identifying and analyzing a representative sample is more efficient and cost-effective than surveying the entirety of the data or population.

An important consideration, though, is the size of the required data sample and the possibility of introducing a sampling error. In some cases, a small sample can reveal the most important information about a data set. In others, using a larger sample can increase the likelihood of accurately representing the data as a whole, even though the increased size of the sample may impede ease of manipulation and interpretation.

Types of data sampling methods

There are many different methods for drawing samples from data; the ideal one depends on the data set and situation. Sampling can be based on probability, an approach that uses random numbers that correspond to points in the data set to ensure that there is no correlation between points chosen for the sample. Further variations in probability sampling include:

- **Simple random sampling:** Software is used to randomly select subjects from the whole population.
- **Stratified sampling:** Subsets of the data sets or population are created based on a common factor, and samples are randomly collected from each subgroup.
- **Cluster sampling:** The larger data set is divided into subsets (*clusters*) based on a defined factor, then a random sampling of clusters is analyzed.
- **Multistage sampling:** A more complicated form of cluster sampling, this method also involves dividing the larger population into a number of clusters. Second-stage clusters are then broken out based on a secondary factor, and those clusters are then sampled and analyzed. This *staging* could

continue as multiple subsets are identified, clustered and analyzed.

- **Systematic sampling:** A sample is created by setting an interval at which to extract data from the larger population -- for example, selecting every 10th row in a spreadsheet of 200 items to create a sample size of 20 rows to analyze.

Sampling can also be based on nonprobability, an approach in which a data sample is determined and extracted based on the judgment of the analyst. As inclusion is determined by the analyst, it can be more difficult to extrapolate whether the sample accurately represents the larger population than when probability sampling is used.

Resampling methods

While reading about Machine Learning and Data Science we often come across a term called **Imbalanced Class Distribution** , generally happens when observations in one of the classes are much higher or lower than any other classes.
As Machine Learning algorithms tend to increase accuracy by reducing the error, they do not consider the class distribution. This problem is prevalent in examples such as Fraud Detection, Anomaly Detection, Facial recognition etc.
Two common methods of Resampling are –

1. Cross Validation
2. Bootstrapping

Cross Validation –

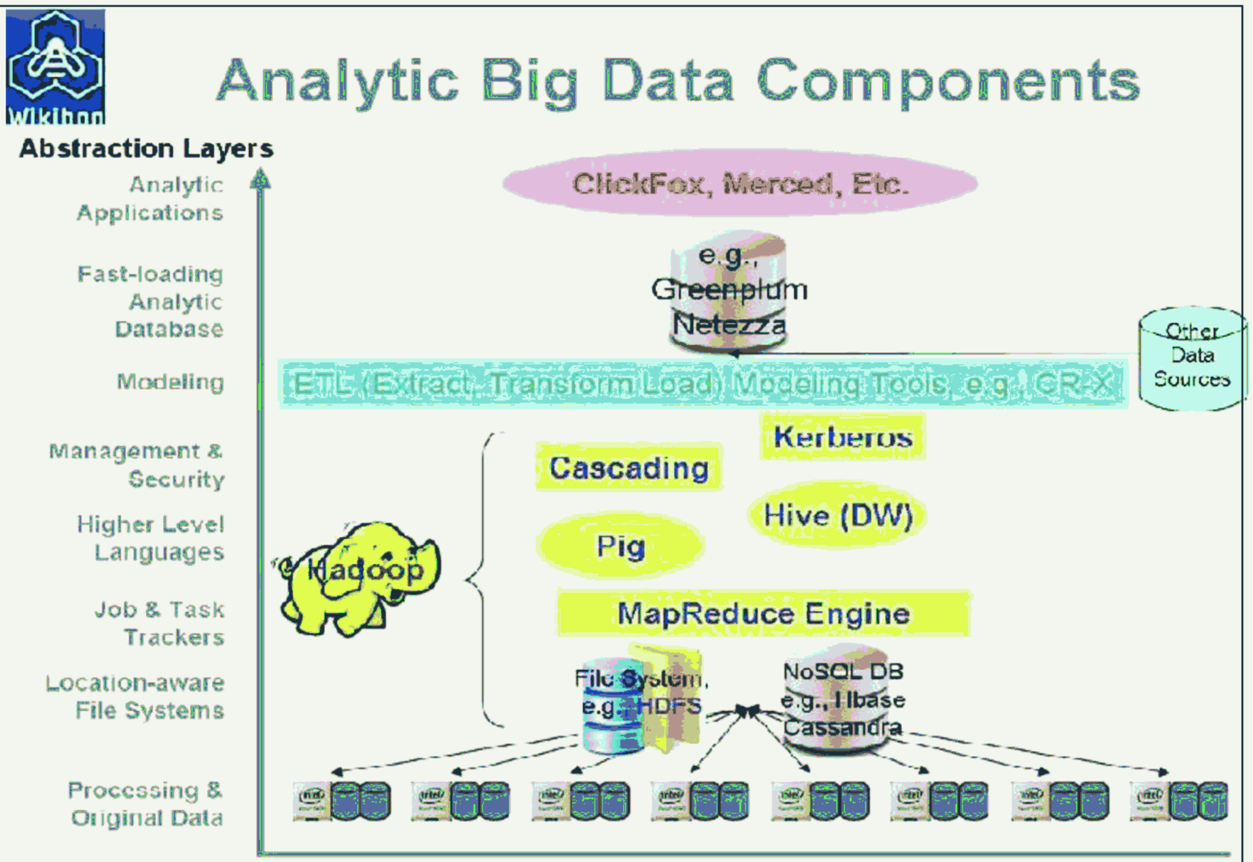
Cross-Validation is used to estimate the test error associated with a model to evaluate its performance.

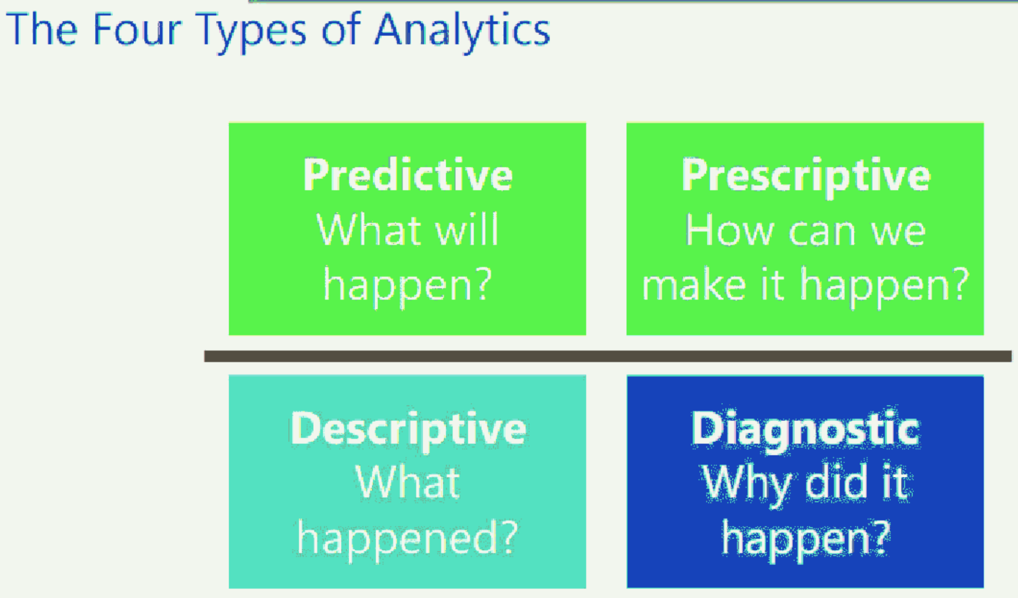
Validation set approach:
This is the most basic approach. It simply involves randomly dividing the dataset into two parts: first a training set and second a validation set or hold-out set. The model is fit on the training set and the fitted model is used to make predictions on the validation set.

Bootstrapping –

Bootstrap is a powerful statistical tool used to quantify the uncertainty of a given model. However, the real power of bootstrap is that it could get applied to a wide range of models where the variability is hard to obtain or not output automatically.

5.b




The Four Types of Analytics
Predictive What will happen?
Prescriptive How can we make it happen?
Descriptive What happened?
Diagnostic Why did it happen?

5 Things Needed for an Effective Data Analytics Program

1. Corporate commitment
2. Data integrity
3. Customer-focused outcomes
4. Data visualization tools

5. Predictive analytics

6.a These R data manipulation topics will provide you with a complete tutorial on ways for manipulating and processing data in R.

What is Data Manipulation in R?

With the help of data structures, we can represent data in the form of data analytics. Data Manipulation in R can be carried out for further analysis and visualisation.

Before we start playing with data in R, you must learn how to import data in R and ways to export data from R to different external sources like SAS, SPSS, text file or CSV file.
One of the most important aspects of computing with Data Manipulation in R is that it enables its subsequent analysis and visualization. Let us see a few basic data structures in R:

1. Vectors in R

These are ordered containers of primitive elements and are used for 1-dimensional data.

Types – *integer, numeric, logical, character, complex.*
2. Matrices in R

These are rectangular collections of elements and are useful when all data is of a single class that is numeric or characters.

Dimensions – *two, three,* etc.
3. Lists in R

These are ordered containers for arbitrary elements and are used for higher dimension data, like customer data information of an organization. When data cannot be represented as an array or a data frame, the list is the best choice. This is because lists can contain all kinds of other objects, including other lists or data frames, and in that sense, they are very flexible.

Learn what all you can do with Lists in R Programming
4. Data Frames

These are two-dimensional containers for records and variables and are used for representing data from spreadsheets etc. It is similar to a single table in the database.

Creating Subsets of Data in R

As we know, data size is increasing exponentially and doing an analysis of complete data is very time-consuming. So, the data is divided into small-sized samples and analysis of samples is done. The process of creating samples is called subsetting.

Different methods of subsetting in R are:

1. $

The dollar sign operator selects a single element of data. The result of this operator is always a vector when we use it with a data-frame.

2. [[

Similar to $ in R, the double square brackets operator in R also returns a single element, but it offers the flexibility of referring to the elements by position rather than by name. It can be used for data frames and lists.

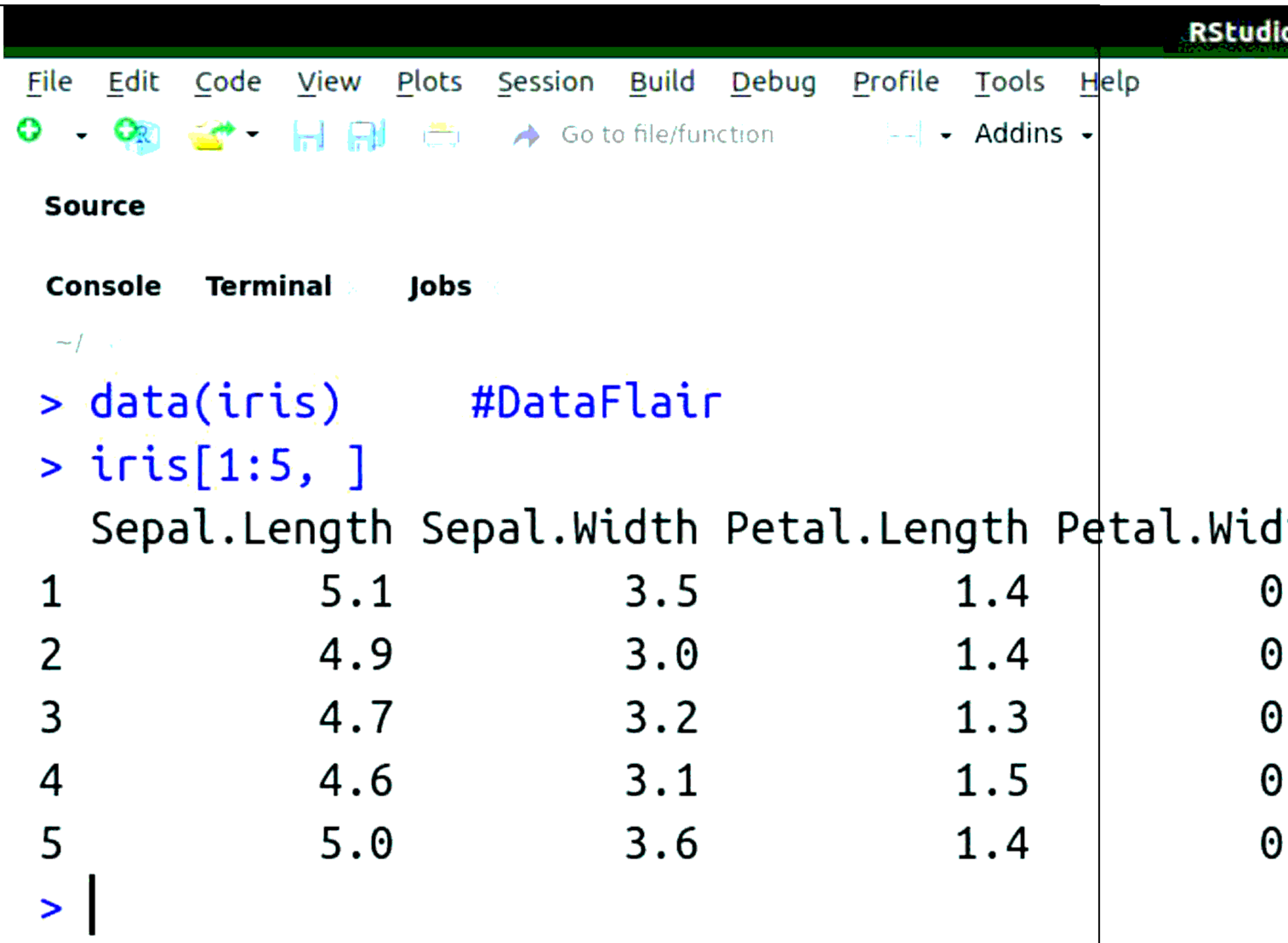
3. [

The single square bracket operator in R returns multiple elements of data. The index within the square brackets can be a numeric vector, a logical vector, or a character vector.

For example: To retrieve 5 rows and all columns of already built-in dataset iris, the below command, is used:

```
> data(iris)
> iris[1:5, ]
```

Output:


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
Source
Console Terminal Jobs
~/

```
> data(iris)     #DataFlair
> iris[1:5, ]
  Sepal.Length Sepal.Width Petal.Length Petal.Wid
1          5.1         3.5          1.4         0
2          4.9         3.0          1.4         0
3          4.7         3.2          1.3         0
4          4.6         3.1          1.5         0
5          5.0         3.6          1.4         0
>
```

Many more function can be used in R programming language for data analysis

6.b There are many basic data types in R, which are of frequent occurrence in coding R calculations and programs. Though seemingly in the clear, they can at a halt deliver surprises. Here you will try to understand all of the different forms of data type well by direct testing with the R code.

Here is the list of all the data types provided by R:

Numeric
Integer
Complex
Logical
Character

Numeric datatype

Decimal values are referred to as numeric data types in R. This is the default working out data type. If you assign a decimal value for any variable x like given below, x will become a numeric type.

```
> g = 62.4      # assign a decimal value to g


> g           # print the variable's value - g
```

Integer data type

If you want to create an integer variable in R, you have to invoke the as.integer() function to define any integer type data. You can be certain that y is definitely an integer by applying the is.integer() function.

```
> s = as.integer(3)


> s         # print the value of s
```

A complex value for coding in R can be defined using the pure imaginary values 'i'.

```
> k = 1 + 2i    # creating a complex number


> k           # printing the value of k
```

The below-mentioned example gives an error since −1 is not a complex value.

```
> sqrt(−1)      # square root of −1
```

And the error message will be something like this:

Warning message:

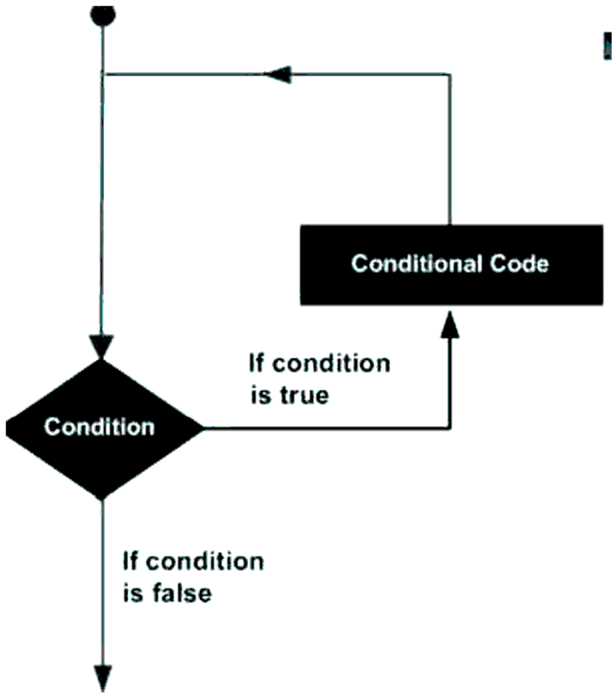
In sqrt(−1) : NaNs produced

LOOPs

There may be a situation when you need to execute a block of code several number of times. In general, statements are executed sequentially. The first statement in a function is executed first, followed by the second, and so on.

Programming languages provide various control structures that allow for more complicated execution paths.

A loop statement allows us to execute a statement or group of statements multiple times and the following is the general form of a loop statement in most of the programming languages −



The **Repeat loop** executes the same code again and again until a stop condition is met.

Syntax

The basic syntax for creating a repeat loop in R is −

```
repeat { 
   commands 
   if(condition) {
      break
   }
}
```

Example

```
v <- c("Hello","loop")
cnt <- 2

repeat {
  print(v)
  cnt <- cnt+1

  if(cnt > 5) {
    break
  }
}
```

When the above code is compiled and executed, it produces the following result −

```
[1] "Hello" "loop"
[1] "Hello" "loop"
[1] "Hello" "loop"
[1] "Hello" "loop"
```

The While loop executes the same code again and again until a stop condition is met.

Syntax

The basic syntax for creating a while loop in R is −

```
while (test_expression) {
  statement
}
```

Flow Diagram

Example

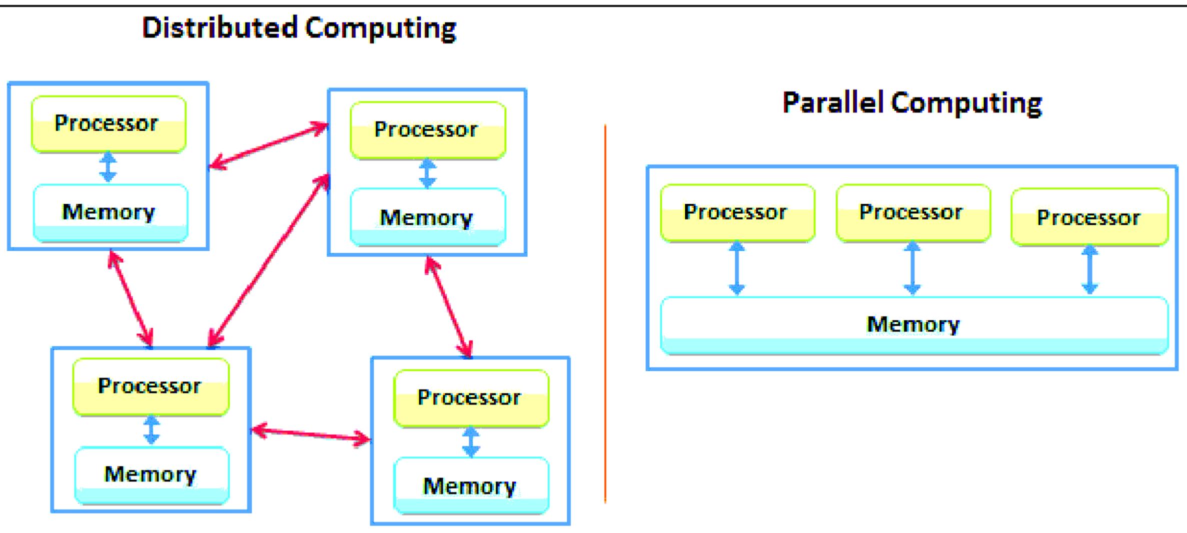
```
v <- c("Hello","while loop")
cnt <- 2

while (cnt < 7) {
  print(v)
  cnt = cnt + 1
}
```

When the above code is compiled and executed, it produces the following result −

```
[1] "Hello"  "while loop"
[1] "Hello"  "while loop"
[1] "Hello"  "while loop"
[1] "Hello"  "while loop"
[1] "Hello"  "while loop"
```

7.a While both distributed computing and parallel systems are widely available these days, the main difference between these two is that a parallel computing system consists of multiple processors that communicate with each other using a shared memory, whereas a distributed computing system contains multiple processors connected by a communication network.



In parallel computing systems, as the number of processors increases, with enough parallelism available in applications, such systems easily beat sequential systems in performance through the shared memory. In such systems, the processors can also contain their own locally allocated memory, which is not available to any other processors.

In distributed computing systems, multiple system processors can communicate with each other using messages that are sent over the network. Such systems are increasingly available these days because of the availability at low price of computer processors and the high-bandwidth links to connect them.

The following reasons explain why a system should be built distributed, not just parallel:

Scalability: As distributed systems do not have the problems associated with shared memory, with the increased number of processors, they are obviously regarded as more scalable than parallel systems.

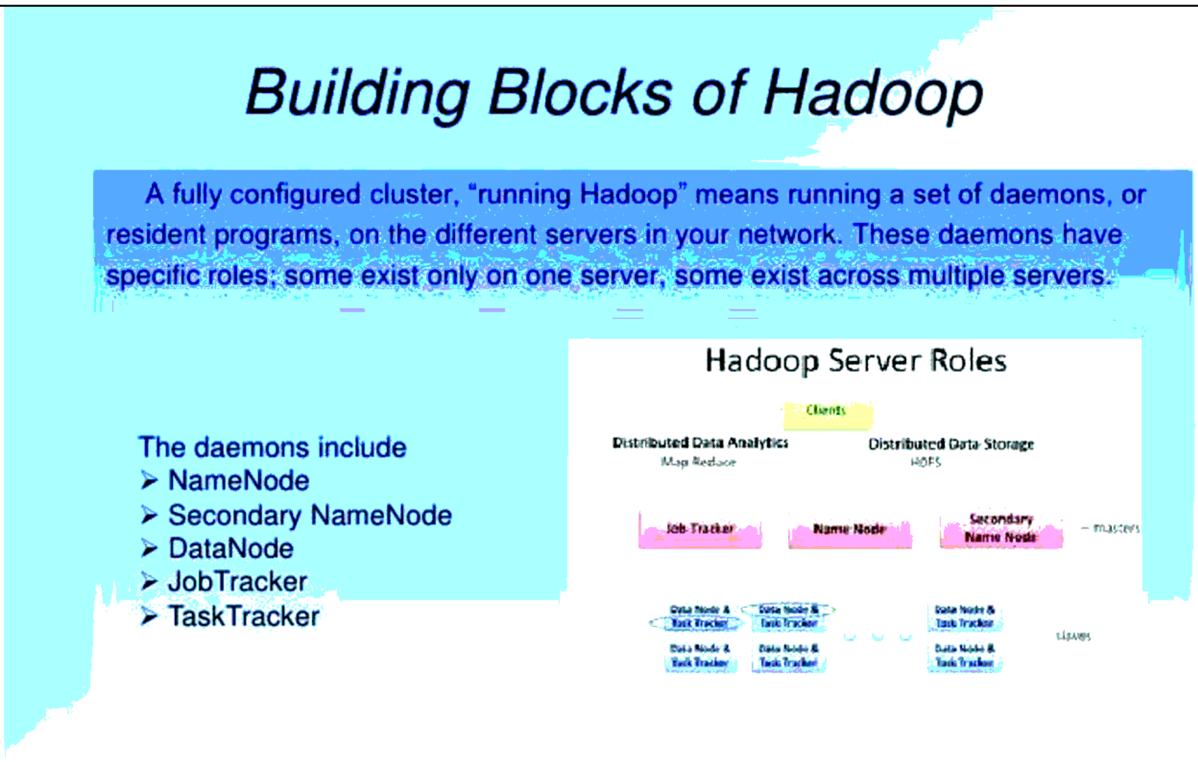
Reliability: The impact of the failure of any single subsystem or a computer on the network of computers defines the reliability of such a connected system. Definitely, distributed systems demonstrate a better aspect in this area compared to the parallel systems.

Data sharing: Data sharing provided by distributed systems is similar to the data sharing provided by distributed databases. Thus, multiple organizations can have distributed systems with the integrated applications for data exchange.

Resources sharing: If there exists an expensive and a special purpose resource or a processor, which cannot be dedicated to each processor in the system, such a resource can be easily shared across distributed systems.

Heterogeneity and modularity: A system should be flexible enough to accept a new heterogeneous processor to be added into it and one of the processors to be replaced or removed from the system without affecting the overall system processing capability. Distributed systems are observed to be more flexible in this respect.

Geographic construction: The geographic placement of different subsystems of an application may be inherently placed as distributed. Local processing may be forced by the low

	communication bandwidth more specifically within a wireless network. **Economic**: With the evolution of modern computers, high-bandwidth networks and workstations are available at low cost, which also favors distributed computing for economic reasons.
7.b	 Building Blocks of Hadoop A fully configured cluster, "running Hadoop" means running a set of daemons, or resident programs, on the different servers in your network. These daemons have specific roles; some exist only on one server, some exist across multiple servers. The daemons include ➢ NameNode ➢ Secondary NameNode ➢ DataNode ➢ JobTracker ➢ TaskTracker Hadoop Server Roles Clients Distributed Data Analytics Map Reduce Distributed Data Storage HDFS Job Tracker Name Node Secondary Name Node masters Data Node & Task Tracker slaves **The building blocks of Hadoop** Hadoop employs a master/slave architecture for both distributed storage and distributed computation. The distributed storage system is called the Hadoop Distributed File System (HDFS). On a fully configured cluster, "running Hadoop" means running a set of daemons, or resident programs, on the different servers in you network. These daemons have specific roles; some exists only on one server, some exist across multiple servers. The daemons include: 1. NameNode 2. DataNode 3. Secondary NameNode 4. JobTracker 5. TaskTracker **NameNode:** The NameNode is the *master* of HDFS that directs the *slave* DataNode daemons to perform the low-level I/O tasks. It is the bookkeeper of HDFS; it keeps track of how your files are broken down into file blocks, which nodes store those blocks and the overall health of the distributed filesystem. The server hosting the NameNode typically doesn't store any user data or perform any computations for a MapReduce program to lower the workload on the machine, hence memory & I/O intensive.

There is unfortunately a negative aspect to the importance of the NameNode - it's a single point of failure of your Hadoop cluster. For any of the other daemons, if their host fail for software or hardware reasons, the Hadoop cluster will likely continue to function smoothly or you can quickly restart it. Not so for the NameNode.

DataNode: Each slave machine in your cluster will host a DataNode daemon to perform the grunt work of the distributed filesystem - reading and writing HDFS blocks to actual files on the local file system

When you want to read or write a HDFS file, the file is broken into blocks and the NameNode will tell your client which DataNode each block resides in. Your client communicates directly with the DataNode daemons to process the local files corresponding to the blocks.

Furthermore, a DataNode may communicate with other DataNodes to *replicate* its data blocks for redundancy.This ensures that if any one DataNode crashes or becomes inaccessible over the network, you'll still able to read the files.

DataNodes are constantly reporting to the NameNode. Upon initialization, each of the DataNodes informs the NameNode of the blocks it's currently storing. After this mapping is complete, the DataNodes continually poll the NameNode to provide information regarding local changes as well as receive instructions to create, move, or delete from the local disk.

8.a **What is Structured Data?**

Most people are familiar with how structured data works. Structured data, as can be assumed from the term, is data that is highly organized and neatly formatted. It's the type of data that can be put into tables and spreadsheets. It might not be the easiest type of data to look through for a human, but compared to unstructured data, it is certainly the easier of the two types for humans to consume. Computers, on the other hand, can search it with ease.

Structured data is also often referred to as quantitative data. These are objective facts which can be looked up in a relational database or a data warehouse. Customer data, for example, would include facts like the customer's name and the transactions he or she engaged in. Searching for these terms would be easy for a computer program when using a structured query language or SQL.

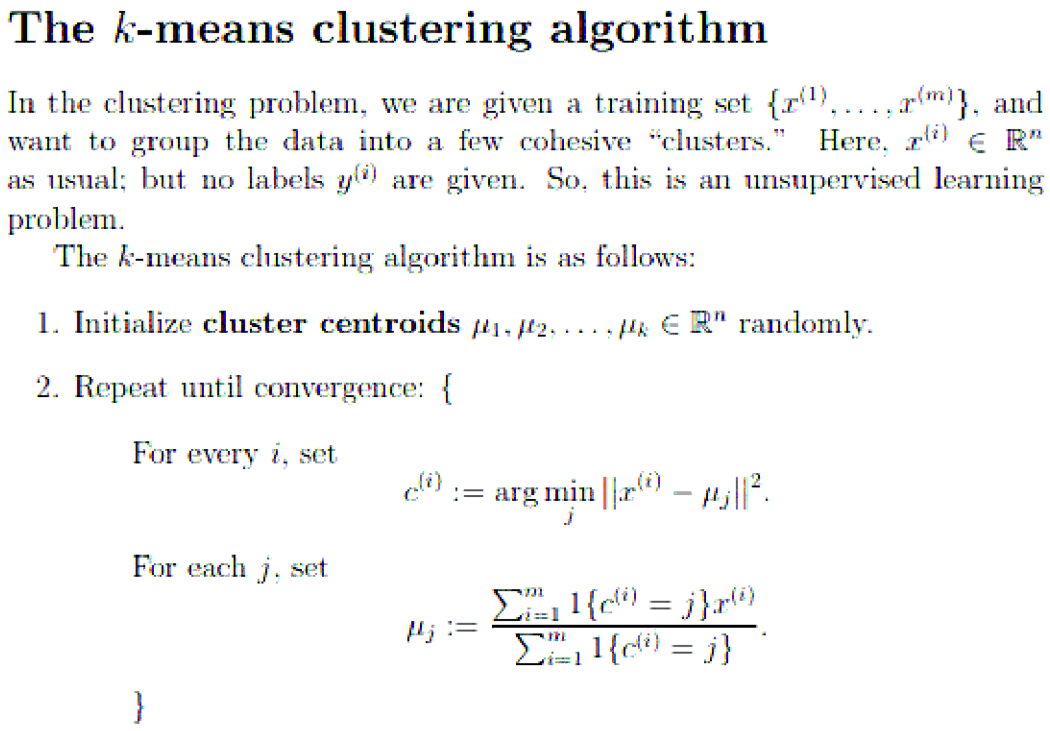
Some other examples of structured data include credit card numbers, dates, financial amounts, phone numbers, addresses, product names, and more. These are all data points that aren't open for interpretation, making it easy for big data applications to collect and analyze.

What is Unstructured Data?

As the term suggests, unstructured data isn't so easily organized or formatted. Collecting, processing, and analyzing unstructured data also represents a significant challenge. That has created some issues since unstructured data makes up the vast majority of available data out there on the web, and it only grows larger every year. With more information becoming available on the web, and most of it unstructured, finding ways to use it has become a vital strategy for many businesses. More traditional data analysis tools and methods aren't enough to get the job done.

Unstructured data can also be called qualitative data, which basically covers everything that structured data does not. It doesn't conform to any predefined models, so it is stored in non-relational databases

	and is queried using NoSQL. Unstructured data is also quite diverse, so examples can make up a long list. Some of the most common unstructured data examples include reports, audio files, images, video files, text files, social media comments and opinions, emails, and more. From these instances, it's clear to see how analysis can be more complex, especially for computer programs to understand.

8.b	
	The k-means clustering algorithm
	
	In the clustering problem, we are given a training set $\{x^{(1)}, \ldots, x^{(m)}\}$, and want to group the data into a few cohesive "clusters." Here, $x^{(i)} \in \mathbb{R}^n$ as usual; but no labels $y^{(i)}$ are given. So, this is an unsupervised learning problem.
	The k-means clustering algorithm is as follows:
	1. Initialize **cluster centroids** $\mu_1, \mu_2, \ldots, \mu_k \in \mathbb{R}^n$ randomly.
	2. Repeat until convergence: {
	For every i, set $c^{(i)} := \arg\min_j \lVert x^{(i)} - \mu_j \rVert^2.$
	For each j, set $\mu_j := \frac{\sum_{i=1}^m 1\{c^{(i)} = j\} x^{(i)}}{\sum_{i=1}^m 1\{c^{(i)} = j\}}.$
	}
	
	In the algorithm above, k (a parameter of the algorithm) is the number of clusters we want to find; and the cluster centroids µj represent our current guesses for the positions of the centers of the clusters. To initialize the cluster centroids (in step 1 of the algorithm above), we could choose k training examples randomly, and set the cluster centroids to be equal to the values of these k examples. (Other initialization methods are also possible.) The inner-loop of the algorithm repeatedly carries out two steps: (i) "Assigning" each training example x (i) to the closest cluster centroid µj , and (ii) Moving each cluster centroid µj to the mean of the points assigned to it. Figure 1 shows an illustration of running k-means.

8.c	Matrices and Arrays The vector variables that we have looked at so far are one-dimensional objects, since they have length but no other dimensions. Arrays hold multidimensional rectangular data. "Rectangular" means that each row is the same length, and likewise for each column and other dimensions. Matrices are a special case of two-dimensional arrays.

Creating Arrays and Matrices

To create an array, you call the array function, passing in a vector of values and a vector of dimensions. Optionally, you can also provide names for each dimension:

```
(three_d_array <- array(
  1:24,
  dim = c(4, 3, 2),
  dimnames = list(
    c("one", "two", "three", "four"),
    c("ein", "zwei", "drei"),
    c("un", "deux")
  )
))
## , , un

##

##       ein zwei drei

## one     1    5    9

## two     2    6   10

## three   3    7   11

## four    4    8   12

##

## , , deux

##

##       ein zwei drei

## one    13   17   21
## two    14   18   22

## three  15   19   23

## four   16   20   24

class(three_d_array)

## [1] "array"
```

The syntax for creating matrices is similar, but rather than passing a dim argument, you specify the number of rows or the number of columns:

```
(a_matrix <- matrix(
  1:12,
  nrow = 4,            #ncol = 3 works the same
  dimnames = list(
    c("one", "two", "three", "four"),
    c("ein", "zwei", "drei")
  )
))
##       ein zwei drei

## one     1    5    9

## two     2    6   10

## three   3    7   11

## four    4    8   12

class(a_matrix)

## [1] "matrix"
```

This matrix could also be created using the array function. The following two-dimensional array is identical to the matrix that we just created (it even has class matrix):

```
(two_d_array <- array(
  1:12,
  dim = c(4, 3),
  dimnames = list(
    c("one", "two", "three", "four"),
    c("ein", "zwei", "drei")
  )
))
##       ein zwei drei

## one     1    5    9

## two     2    6   10

## three   3    7   11

## four    4    8   12

identical(two_d_array, a_matrix)

## [1] TRUE

class(two_d_array)

## [1] "matrix"
```

When you create a matrix, the values that you passed in fill the matrix column-wise. It is also possible to fill the matrix row-wise by specifying the argument byrow = TRUE:

```
matrix(

  1:12,
  nrow = 4,
  byrow = TRUE,
  dimnames = list(
    c("one", "two", "three", "four"),
    c("ein", "zwei", "drei")
  )
)
##       ein zwei drei

## one     1    2    3

## two     4    5    6

## three   7    8    9

## four   10   11   12
```

Rows, Columns, and Dimensions

For both matrices and arrays, the dim function returns a vector of integers of the dimensions of the variable:

```
dim(three_d_array)

## [1] 4 3 2

dim(a_matrix)

## [1] 4 3
```

For matrices, the functions nrow and ncol return the number of rows and columns, respectively:

```
nrow(a_matrix)

## [1] 4

ncol(a_matrix)
```

```
## [1] 3
```

nrow and ncol also work on arrays, returning the first and second dimensions, respectively, but it is usually better to use dim for higher-dimensional objects:

```
nrow(three_d_array)

## [1] 4

ncol(three_d_array)

## [1] 3
```

The length function that we have previously used with vectors also works on matrices and arrays. In this case it returns the product of each of the dimensions:

```
length(three_d_array)

## [1] 24

length(a_matrix)

## [1] 12
```

We can also reshape a matrix or array by assigning a new dimension with dim. This should be used with caution since it strips dimension names:

```
dim(a_matrix) <- c(6, 2)
a_matrix
##      [,1] [,2]

## [1,]    1    7
## [2,]    2    8

## [3,]    3    9

## [4,]    4   10

## [5,]    5   11

## [6,]    6   12
```