fingerTips

Data Intelligence Solutions

Refreshing Material Part-2

POWER BI

Introduction

What is Microsoft Power BI?

Microsoft Power BI is a suite that is a collection of business intelligence tools such as software services, apps, and data connectors. It is a cloud-based platform used to consolidate data from varied sources into a single data set. These data sets are used for data visualization, evaluation, and analysis by making sharable reports, dashboards, and apps. Microsoft offers three types of Power BI platforms i.e. Power BI Desktop (a desktop application), Power BI Service (SaaS i.e., Software as a Service), and Power BI Mobile (for iOS and Android devices).

Power BI can be deployed both on-premise and on-cloud. It can also import data from local databases/data sources, cloud-based data sources, big data sources, simple Excel files, and other hybrid sources. Thus, Power BI, a leader among a lot of other BI tools proves to be an efficient and user-friendly tool for data analysis. The file format for the same is .pbix. It enables the users to consolidate data from multiple sources, make interactive dashboards, evaluate data, create informative reports, and share them with other users.

In Power BI you can create visualizations such as:

- Stacked bar chart
- Stacked column chart
- Clustered bar chart
- Line chart
- Area chart
- Waterfall chart
- Pie chart
- Donut chart
- Map
- Filled map

Features of Power BI

The unique features of Power BI are as follows:

1. **Range of Attractive Visualizations**
   Visualizations i.e. the visual representation of data play a central role in Power BI. It offers a wide range of detailed and attractive visualizations. You can create reports and dashboards using as simple or as complex visualizations as you want to represent your data set. There is also a library available for custom visualizations.

2. Get Data (Data Source)

Get Data feature lets Power BI users select from a range of data sources. The data sources are anywhere in the spectrum from on-premise to cloud-based, unstructured to structured. New data sources are added every month.

Some of the latest available data sources are as follows:

- Excel
- Power BI datasets
- Power BI dataflows
- SQL Server
- MySQL database
- Analysis Services
- Azure
- Text/CSV

3. Datasets Filtration

Dataset is a single set of data created as a result of taking data from multiple data sources. You can use the datasets to create visualizations of different kinds. A dataset can be made of data taken from a single source like an Excel workbook or more than a data source.

You can filter the datasets and have smaller subsets containing only the important data and contextual relevance. Power BI provides the users with a wide range of in-built data connectors such as *Excel, SQL database, Oracle, Azure, Facebook, Salesforce, MailChimp,* etc. Users can easily connect to such data sources and create datasets by importing data from one or more sources.

4. Customizable Dashboards

Dashboards are a collection of visualizations offering meaningful information or insights into data. Typical dashboards in Power BI are composed of multiple visualizations as tiles. They are single pages from the reports. The dashboards are shareable as well as printable.

5. Flexible Tiles

A tile is a single block containing a visualization in a Power BI dashboard. Tiles segregate each informative visualization properly to provide a clearer view. These tiles can be adjusted and the size can also be changed. Also, they can be placed anywhere on the dashboard per the users' convenience.

6. Navigation Pane

The navigation pane has options for datasets, dashboards, and reports. Users can conveniently work in Power BI and navigate between datasets, the dashboard they are working on, and reports they are creating.

Power BI Building Blocks – 4 Major Parts of Power BI

The fundamental Power BI building block are:

- Visualizations
- Datasets
- Reports
- Dashboards

a. Visualizations

Perception is a visual portrayal of information. For example, a diagram, chart, shading-coded outline, and other intriguing things you can make to speak to your information outwardly. Power BI has a wide range of various perception writes, and additionally coming constantly. The accompanying picture demonstrates a gathering of various visualization that was made in the Power BI benefit.

b. Datasets

A dataset is an accumulation of information that Power BI uses to make its representations. These are the collection of datasets that you can import or connect to. You can have a basic dataset in light of a solitary table from the Excel exercise manual, like what's appeared in the accompanying picture.

c. Reports

In Power BI, a report is a gathering of perceptions that seem together on at least one page. Much the same as some other report you may make for a business introduction, or a report you would compose for a school task, in Power BI a report is an accumulation of things that identify with each other. The accompanying picture demonstrates a report in Power BI Desktop – for this situation, it's the fifth page in a six-page report. You can likewise make reports in the Power BI benefit.

Reports let you make numerous perceptions, on various diverse pages if fundamental and give you a chance to organize them in the way best recount your story.

d. Dashboards

When you prepare to share a solitary page from a report or offer an accumulation of perceptions, you make a dashboard. Much like the dashboard in an auto. A Power BI dashboard is a gathering of visuals from a solitary page that you can impart to others. Frequently, it's a chosen gathering of visuals that give a snappy understanding of the information or story you're attempting to exhibit.

A dashboard needs to fit on a solitary page, frequently called a canvas. Consider it like the canvas that a craftsman or painter utilizes. A workspace where you make, consolidate, and adjust fascinating and convincing visuals. You can impart dashboards to different clients or gatherings, who would then be able to communicate with your dashboard when they're in Power BI benefit, or on their cell phone.

How to Create Dashboard in Power BI

In this section, we'll learn how to create a Power BI dashboard from scratch. In our exercise, we'll create a Sales and performance dashboard. To create this dashboard, we will import sales data related to customers, products, regions, orders, and sales details into the Power BI system. Using this data, we'll create different kinds of visualizations to represent various aspects of the imported data.

On the top bar, you have tabs like Home, View, Modeling, and Help to have a range of options in them. On the right are two sections, Visualizations, and Fields. From the Visualizations section, you can select a visual and edit it. The Fields section contains all the fields from the data tables you imported. Just select fields from here to add them to your visual.

Step 1: Importing data

The first task you need to do when you start with creating a dashboard in Power BI is to import data from source files. Click on the Get Data option to select a data source of your choice. Select a data source and click on Connect. You can prepare the data imported in the Power Query Editor.

Step 2: Formatting

From the Data tab, you can access all the imported tables and view them in tabular form. On the right, you'll find a list of tables and fields within those tables. You can select a table or field to perform formatting actions on them. If you have fields such as date, time, city, state, percentage value, currency, etc. you can change the format or data type from the Modeling tab.

Step 3: Modeling

Although the relationships and associations between the tables that you load are already created by the Power BI's engine, still you can view and make changes in the data model from the Model tab given on the left horizontal bar. The relationships between the two tables are indicated by links joining two common fields.

Step 4: Creating a KPI for Total Sales and Gross profit

So, for our dashboard, we imported five tables; customer details, order details, place details, product details, and sales details. The first visualization that we'll make is a KPI. Select KPI from the visualizations section.

Select fields you want to add to the visual from the Fields section. You can also drag and drop the fields into respective columns indicated by the arrow in the image below.

Step 5: Creating a chart showing Sales by State and Category

Next, we'll create a Stacked bar chart that is going to show sales value by state and category of product. Add this chart from the Visualizations.

Add fields in the chart and format the title, data labels, legend, axes, plot area, data colors, etc.

Step 6: Creating a chart showing Sales and Gross profit by Year

Moving on, the next visualization we need to create is a column chart (Line and clustered column chart). We will create this visual to show the total sales and gross profit by year (2017,2018 and 2019)

Select a Line and Clustered Column Chart from the Visualizations section.

Step 7: Creating a chart showing Sales by month

Next, we'll create an area chart that will show the total sales of products over 12 months. We created an area chart by selecting the Area chart from Visualizations and adding respective fields to it.

Step 8: Creating a multi-row card

In addition to visually representing data via graphs and charts, you can display data as textual information using cards or multi-row cards. So, we'll add a Multi-row card from the Visualizations section.

Step 9: Creating a map showing the total units sold by the state

From the wide range of visualizations available, we can also represent information on the map. In our dashboard, we will add a map showing the total units sold per state in the USA. We selected a Filled map from the Visualizations section.

Step 10: Adding a Slicer for Sub-categories of products

Lastly, we'll add a Slicer for Subcategories of products in the record. Using this slicer, users can select specific categories and filter through data. Upon selecting in the slicer, all the other visuals will change and show only the visuals related to the selected field or value.

Step 11: Finish the final dashboard

Now that we are done adding all the different types of visuals and graphics that we needed on our dashboard, we are nearing the final steps. Adjust and resize the visualizations on the dashboard as you like. You can also select a theme for the dashboard, its page size, background, etc.

Step 12: Publish the dashboard

Once your dashboard is ready, you can publish it on the Power BI workspace. First, save your dashboard in your system.

Go to the Publish option to publish the dashboard on the web. Log in to your Power BI account and the publishing process will be successful. Then, you will get a link to a web source where the dashboard is uploaded and available for other users' access.

DAX in Power BI

DAX stands for Data Analysis Expressions i.e. such expressions or formulas that are used for data analysis and calculations. These expressions are a collection and combination of functions, operators, and constants that are evaluated as one formula to yield results (value or values). DAX formulas are very useful in BI tools like Power BI as they help data analysts to use the data sets they have to the fullest potential.

With the help of the DAX language, analysts can discover new ways to calculate data values they have and come up with fresh insights.

DAX is a functional language i.e. its complete code is always a function. An executable DAX expression may contain conditional statements, nested functions, value references, etc.

DAX formulas have two primary data types; Numeric and Non-numeric or Others. The numeric data type includes integers, decimals, currency, etc. Whereas, the non-numeric consists of strings and binary objects.

DAX expressions are evaluated from the innermost function going to the outermost one at the last. This makes the formulation of a DAX formula important.

DAX Functions

A DAX function is a predefined formula that performs calculations on values provided to it in arguments. The arguments in a function need to be in a particular order and can be a column reference, numbers, text, constants, another formula or function, or a logical value such as TRUE or FALSE. Every function performs a particular operation on the values enclosed in an argument. You can use more than one argument in a DAX formula.

Key Points about DAX Functions

Here are some unique facts about DAX functions that you must know to understand them better:

Any DAX function always refers to a complete column/field or a table. It will never refer to individual values. If you want to use the functions on separate values within a column, you need to apply filters in a DAX formula.

DAX functions provide the flexibility to create a formula that is applied on a row-by-row basis. The calculations or formulas get applied as per the context of the values in each row.

In some cases, DAX functions return a full table which can be used in other DAX formulas that need a complete set of values. However, you cannot display this table's contents.

DAX functions have a category known as time intelligence functions. Such functions are used to calculate time/date ranges and periods.

Types of Filters in Power BI – Edit View in Power BI Filter Pane

Filters in Power BI

In Power BI benefit, reports can open in Editing perspective or Reading view. In Editing view, and Desktop Report sees, report proprietors can add filters to a report and those Power BI filters are spared with the report. Individuals seeing the report from a Reading perspective can associate with the filters in Power BI and spare their progressions, yet can't add new filters to the report.

How to Clear & Add Filters in Power BI

a. Clear the Power BI Filter

In either progressed or fundamental filtering mode, select the eraser symbol to clear the filters in Power BI.

b. Add the Power BI Filter

Follow this step to add Filters in Power BI:

In Desktop and Power BI benefit Editing view, add a filter to a visual, page, drill through, or report by choosing a field from the Fields sheet and hauling it into the suitable filter well, where you see the words Drag fields here. Once a field has been included as a filter, tweak it utilizing the Basic filtering and Advanced filtering controls (portrayed underneath).

Hauling another field into the Visual level filter zone does not add that field to the visual, but rather it allows you to filter the visual with this new field. In the case beneath, Chain is added as another filter to the visual. Notice that essentially including Chain as a filter does not change the visual until the point when you utilize the Basic or Advanced filtering controls.

Python

Difference between Python and R

Python	R
General purpose, object-oriented language.	Open-source programming language for statistical computing.
Can easily handle large data sets.	Not suitable for large data sets.
Essential libraries are Numpy, Pandas, etc.	Essential libraries are caret, tidyverse, etc.
Python outshines R in terms of speed.	R is inefficient when it comes to speed.
Easier to learn for beginners.	It is not easy to learn for beginners.

Difference between Python and Java

Python	Java
Dynamically-typed	Statically-typed
No need for semicolon.	It Shows error without a semicolon.
Codes are shorter and easier to read.	Not easy when compared to Python
More productive as codes are short.	Less productive compared to Python.

Python Variable

Python is not "statically typed", here you don't need to declare variables before using them, or even declaration on their type is not needed. A variable is created at the same moment you assign a value to it. A variable can be a name given to a memory location or a basic unit of storage in a program.

- Values stored in a variable can be changed during program execution.
- A variable is a name given to a memory location, and all the operations done on it effects that memory location.

There are some rules for creating variables in Python:

- A variable name must start with a letter or the underscore character.
- The name of the variable should not start with a number.
- It can only contain alpha-numeric characters and underscores like A-Z, 0-9, and _).

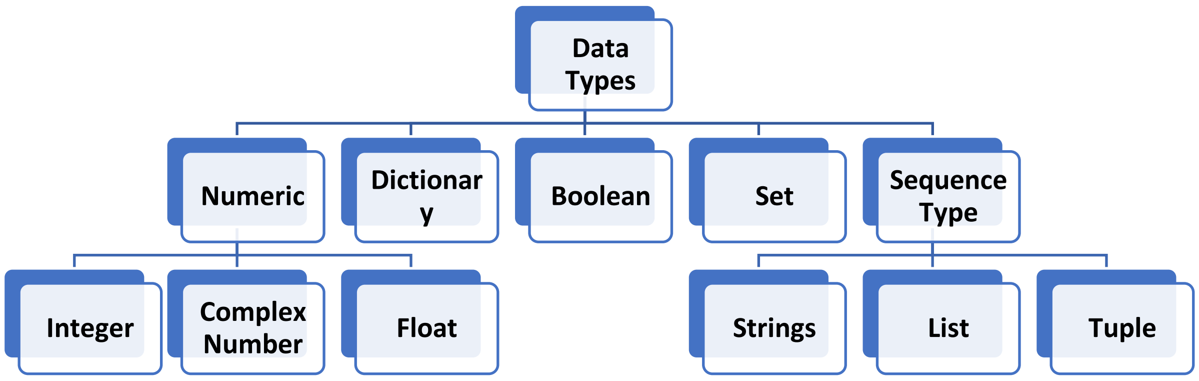
- Variable names are case-sensitive (boy, Boy and BOY these all are three different variables).

  The reserved words (keywords) cannot be used to name the variable.

Data Type:

In Python data Types concepts are important; Variables can store data of different types and these different types of data can do different things.

Data Type	Represents	Examples
integer	whole numbers	-5, 0, 123
floating point (real)	fractional numbers	-87.5, 0.0, 3.14159
string	A sequence of characters	"Hello world!"
Boolean	logical true or false	true, false
nothing	no data	null



1. Numeric Data Type – Integers, Float and Complex Numbers

Integers

This section contains positive or negative whole numbers but without fractions or decimals.

Float

This section contains float values. The values are specified by a decimal point.

Complex Numbers

It is represented by a complex class. It contains real and imaginary numbers. For example, 2+3j

2. Sequence Data Types – Strings, List and Tuple.

String

A string is a collection of characters inserted in a single quote, double quote, or triple quote. Example is “hello”.

List

The list is similar to arrays for the collection of data. The items in the list do not need to be of the same type. Example = [1,2,3]

Tuple

A tuple is similar to List except that Tuples are immutable. They can't be modified after they're created. Example = (1,2,3)

3. **Boolean**

   In Python, Boolean Data Types are either True or False. For example, 1<2 is True, while 1=0 is False.

4. **Set**

   In Python, a Set is a collection of data that is changeable and has no duplicate elements. The order of elements in a set is undefined though it may consist of various elements. Example: - {"apple", "banana", "cherry"}

5. **Dictionary**

   Dictionary in Python is a collection of data that are unstructured, changeable, and indexed. Dictionaries are written with curly brackets.
   Example: -thisdict = {"brand": "Ford", "model": "Mustang","year": 1964}

Python Operators

Python Operators are used to performing operations on values and variables.

Arithmetic Operations

These operations are used to perform mathematical operations like addition(+), subtraction(-), multiplication(*), division (/ - float & // - floor), Modulus(% - remainder values) and Power(**).

Comparison Operators

Comparison Operators compare the value among the operators. It displays either True or False depending upon the condition. Examples are Greater (>), Less than (<), Equal to (==), Not Equal to (!=), Greater Than equal to (>=) and Less than equal to (<=).

Logical Operators

Logical operators are used on combined conditional statements. These operators perform Logically AND, Logical OR, Logical NOT.

Bitwise Operators

Bitwise Operators work on bit-by-bit operations. These operators are used to operate on binary numbers.

Operator	Description	Syntax
&	Bitwise AND	a & b
|	Bitwise OR	a | b
~	Bitwise NOT	~a
^	Bitwise XOR	a ^ b
>>	Bitwise right shift	a>>
<<	Bitwise left shift	a<<

Assignment Operators

Assignment operators are used to assigning values in operators.

Operator	Description	Example
=	Assigns values from right side operands to left side operand	c = a + b assigns value of a + b into c

+= Add AND	It adds right operand to the left operand and assign the result to left operand	c += a is equivalent to c = c + a
-= Subtract AND	It subtracts right operand from the left operand and assign the result to left operand	c -= a is equivalent to c = c - a
*= Multiply AND	It multiplies right operand with the left operand and assign the result to left operand	c *= a is equivalent to c = c * a
/= Divide AND	It divides left operand with the right operand and assign the result to left operand	c /= a is equivalent to c = c / a
%= Modulus AND	It takes modulus using two operands and assign the result to left operand	c %= a is equivalent to c = c % a
**= Exponent AND	Performs exponential (power) calculation on operators and assign value to the left operand	c **= a is equivalent to c = c ** a
//= Floor Division	It performs floor division on operators and assign value to the left operand	c //= a is equivalent to c = c // a

Identify Operators

Is and is not are identity operators. They are used in the following manner.

Is	TRUE if operands are identical
Is not	TRUE if operands are not identical

Python Pre-defined Functions or Built-in functions

These are the functions that are already defined in the Python programing language. The users can easily use these functions without defining them first. A few of the built-in functions are:

- open() – Used to open a file
- print() – It is used to print statement
- list() – It returns a list
- sum() – It is used to sum the values inside a sequence
- type() – It is used to return the type of object.

User-defined Functions

These functions are framed by the programmers to ease their complexity while programming and to use them according to their need.

Conditional and Control Statements

Conditional Statements: -

The if statement

It checks whether <Expressions> evaluates to True and if it happens it immediately blocks <Statements>, otherwise it is not executed. You need to note the indentation before the statement block <Statements>.

Syntax of the if statement:

```
if <Expression>:
   <Statements>
```

The if...else statements

It evaluates the statement of the block 1 if <Expression> is True and statement block 2 otherwise.

Syntax of the if..else statement

```
if <Expression 1>
    <statements 1>
else:
     <statements 2>
```

The if...else...else statements

It evaluates the statements block 1 if the expression block 1 is True or evaluates the statements block 2 if the expression of the block 2 is True. Otherwise it evaluates the statement block 3

Syntax of if... elif...else statement will be

```
if <Expression 1>:
    <Statements 1>
elif  <Expression 2>:
       <Statements 2>
else :
        <Statements 3>
```

Control Statements

The for loop

It enables to iterate on an ordered collection of objects and then execute the same sequence of the statements for each of the element.

Syntax of the for loop :

```
for Element in Set :
       <Statements>
```

The While Loop

It evaluates Initial_Condition, if it is true then evalutes <Statements>, until <Condition> is false.

Syntax of the While Loop :

```
Initial_Condition
While <Condition>:
          <Statements>
```

The break statement

It provides a way to exit the while loop even when Condition evaluates to True

Syntax of the break Statement

```
Initial Condition
while <Condition>:
          <Statements>
          if <Condition 2 >:
              break
```

The Continue Statement

Python generally provides a way to skip the evaluation some instances of a for loop using the continue statement.

Syntax of the Continue Statement

```
 for i in list:
<Statements>
if <Condition 2>:
      Continue
```

Here it Evaluates Statements for each i in the list, while the process it skips evaluating any instance of Statements whenever Condition 2 is True.

Python Lambda expression

A lambda function is an anonymous function and can take any number of arguments, but can only have one expression.
Syntax : lambda arguments : expression

This expression is executed and the result is returned. We can take a example here to understand more clearly
Let's add 25 to argument a , and return the result:
x = lambda a : a + 25
print(x(5))

Output will be: 30

Why do we need to use Lambda functions?

Lambda functions are better when you use them with anonymous function inside another function.

Introduction to Data Visualization

Data Visualization is a graphical representation of any data or information. It includes visual elements such as charts, graphs, and maps are the few data visualization tools that provide the viewers an easy and accessible way of understanding the represented information as data visualization will enable you to make decisions of any sector or industry to look into analytical reports and understand about the concepts that might be difficult to understand.

Introduction to Matplotlib

Matplotlib is one of the most used and powerful libraries in Python that is used for Data Visualization. . It is a 2D plotting library. The library is structured in a way that within a few lines of code, it can generate a visual data plot.
Matplotlib was originally written as an open-source alternative for MATLAB. The key to understanding the working of plots is to understand Matplotlib pyplot API:

- Figure: The figure is the top-level container. It visualizes everything in a plot including one or more Axes.

- Axes: It is the area where data is plotted. It includes X-Axis, Y-Axis, and sometimes a Z-Axis also.

Box Plot

This box plot is used to show a summary of the whole dataset. It contains minimum, first quartile, median, third quartile, and maximum. The median is present between the first and third quartile. A Box Plot is the visual representation of the statistical five-number summary of a given data set.

Scatter Plots

Scatter plots use the dots to showcase the plotting. It easily shows the relationships between the variables. The function scatters() is used for Scatter Plots. The closer the dots to the line, the weaker the relation will be in the plot.

Pie Chart

The circular chat is used to show the percentage of the data. It is used to compare the individual categories to the whole data. The function Pie() is used to create this visualization.

Line Plots

The plots are used to study the relationship between the x and y-axis. Here, x and y are the coordinates of the horizontal and vertical axis. In Matplotlib, the plot() function is used to create the visual graph.

Distribution Plots

Distribution plots of Seaborn can be compared with histograms in Matplotlibs. In histograms, frequency plots will be used and in the distribution plot, approximate probability density across the y-axis is plotted. The function **sns.distplot()** is used in the code.

Introduction to Seaborn

Seaborn is Library in Python that is especially used for making statistical graphics. The library is built on top of Matplotlib and is integrated with Pandas. To learn Seaborn, the user needs to get familiar with Numpy, Matplotlib, and Pandas.

Heatmaps

The Heatmaps showcase the data in 2D form. It shows the data in a colored graph which makes the understanding of data easy. The function **sns.heatmap()** is used to create the Heatmap.

Pair Plots

Pair plots are used to study the relationship pattern among more than 3 variables. The function sns.pairplot() is used in the code to create pair plots using Seaborn.

Distribution Plots

Distribution plots of Seaborn can be compared with histograms in Matplotlibs. In histograms, frequency plots will be used and in the distribution plot, approximate probability density across the y-axis is plotted. The function **sns.distplot()** is used in the code.

Introduction to Pandas

Pandas are a library of Python that helps you to work with tabular data, time-series data, matrix data, etc. The library allows you to work on big data, analyze it, and derive conclusions from it. A few of the things for which Pandas are used are:

- Data wrangling and Manipulation
- Merging multiple Datasets
- Filling out missing values in Datasets
- Easily import data from different sheets
- And, exporting data to different sheets

Features of Panda Library

Panda, the Python Library, is used to work with datasets, mainly used for analyzing, cleaning, and manipulating data. The name of the Library "Panda" has a reference to "Panel Data", and "Python Data Analysis". The top features of Panda Library are mentioned below:

- Data Handling
  Panda library has made it easy to manage and explore the data. The library comes with Series and DataFrames that makes representation and manipulation of data easy.
- Organization of Data
  Having a huge chunk of data without it being properly aligned is useless as unorganized data is hard to interpret. The organization and labeling of data are perfectly done by Panda's intelligent methods of alignment and indexing.
- Handling Missing Data
  Data is very important for every organization but one of the big problems associated with it is missing data. It is of utmost importance to handle the missing data properly so that the insights derived from the data are of the utmost accuracy. Feature of handling missing data is integrated with Panda Library.
- Unique Data
  Data is not always filtered, it comes with numerous repetitions and therefore, you must analyze the data that have unique values. Panda library in Python allows the users to see unique values in the dataset.
- Visualize
  The important step in data science is the visualization of data. Visualization makes the data understandable easily. Panda libraries have in-built features to help the users to plot the data and see various kinds of a graph that can be formed.
- Grouping
  The option of separating the data based on certain criteria and then grouping it differently is important. There's a feature in Panda named GroupBy that allows the user to split the data into different categories.
- Multiple file formats supported
  Data is not found from one source, not in one particular format. Instead, it comes from different sources in different formats. Therefore, libraries must support various file formats. Panda supports a huge amount of file formats, right from JSON to CSV, including Excel and many more.

Series objects in Pandas

A Series is a one-dimensional array that is capable of holding any type of data. To understand it in simple terms, Panda Series is nothing but a column in an excel sheet. For example, columns "name", and "age" represent a series.

A Panda Series can be created via the following constructors:

- data

  Data takes various forms like ndarray, list, constants

- index

  an index is used to label the rows. The values of the index should be unique and of the same length as the data.

- dtype

  dtype is used to specify the datatype of the series. If not present, the datatype will be inferred.

- name

  This parameter is used to give a name to the series.

- copy

  It is used to copy the input data.

Stacking and Unstacking

Stacking in Panda means converting the innermost column Index into the innermost row Index. On the other hand, unstacking is its opposite. It means converting the innermost row index into the innermost column index.

Introduction To Numpy

It is an array processing package that provides you with a high-performance multidimensional array object and tools so that you can work with these arrays.

It is the fundamental package for scientific computing with python, it contains some of important features like:

- It has a powerful N-dimensional array object
- Sophisticated (broadcasting) functions
- Some tools for integrating C/C++ and Fortran Code
- Capabilities such as linear algebra, Fourier transform, and a random number.

NumPy can also be used as an efficient multi-dimensional container that contains generic data. Using Numpy arbitrary data types can be easily defined which will allow Numpy to speedily get integrated with a wide variety of databases.

Arrays in NumPy

The main objective of the NumPy is the homogeneous multidimensional array.

- It is a table of elements (which are usually numbers), all are of the same type indexed by a tuple of positive integers
- Dimensions in NumPy are called axes. The number of axes is rank.
- NumPy's array class is called ndarray, which is also known as the alias array.

Elements	Description
reshape	It gives a new shape to an array without changing its data.
flat	It is A 1-D iterator over the array.
flatten	It returns a copy of the array collapsed into one dimension.
ravel	It returns a contiguous flattened array.
transpose	It permutes the dimensions of an array
ndarray.T	It is the same as self.transpose()
rollaxis	It Rolls the specified axis backward.
swapaxes	Here interchanges get done between the two axes of an array.
broadcast	It produces an object that mimics broadcasting
broadcast_to	Broadcasts an array to a new shape
expand_dims	Expands the shape of an array
squeeze	It Removes single-dimensional entries from the shape of an array.
concatenate	It joins a sequence of arrays along an existing axis
stack	Here the sequence of arrays gets joined along a nex axis
hstack	Stacks arrays in sequence horizontally (Column wise)
vstack	stacks arrays in sequence vertically which is row-wise
split	It splits an array into multiple sub-arrays
hsplit	Array gets splits into multiple sub-arrays horizontally (ccolumn-wise
vsplit	Splits an array into multiple sub-arrays vertically (row-wise)
resize	It returns a new array with the specified shape
append	It appends the value to the end of an array
insert	insertion of the values along the given axis before the given indices.
delete	Here new array with sub-arrays along an axis is deleted.
unique	It Finds the unique elements of the array.

Functions	Description
add()	Here arguments can be added element-wise.
positive()	They are Numerical positive, element-wise.
negative()	They are Numerical negative, element-wise.
multiply()	Multiply arguments element-wise.
power()	Here first array elements get raised to powers from the second array, element-wise.
subtract()	It subtracts arguments, element-wise
true_divide()	It returns a true division of the input, element-wise.
floor_divide()	It returns the largest integer smaller or equal to the division of the inputs.
float_power()	Here first array elements get raised to powers from the second array, element-wise
mod()	Return the element-wise remainder of the division.
remainder()	Return the element-wise remainder of the division.
divmode()	Return element-is quotient and remainder simultaneously

Function	Description
Convolve()	It returns the discrete, linear convolution of two one dimensional
sqrt()	It returns all of the nonnegative square roots of an array, element-wise
square()	It returns the element-wise square of the input.
absolute()	Calculate the absolute value element-wise.
fabs()	It also calculates the absolute values element-wise.
sign()	It returns an element-wise indication of the sign of a number.
intern()	One-dimensional linear interpolation
maximum()	Element-wise maximum of array elements.
minimum()	Element-wise minimum of array elements.
real_if_close()	If complex input returns a real array if complex parts are close to zero.
nan_to_num()	It Replaces NaN with zero and infinity with large finite numbers.
Heaviside()	All the Heaviside step functions are computed by it.

MACHINE LEARNING

What is Machine Learning?

Machine Learning (ML) is the part of artificial intelligence that prepares the machines to become more accurate at predicting the outcomes without being specially trained for it. The algorithms in machine learning use historical data to provide the output.

Need for Machine Learning

Top companies like Netflix, and Amazon use algorithms of Machine Learning models with a huge amount of data to derive useful insights and obtain accurate results.

Data mining is considered to be one of the popular terms for machine learning as it extracts meaningful information from a large pile of datasets and is used for decision-making tasks. Some of the top reasons why Machine Learning is important are:

- **Decision Making –** The algorithms help to make better business decision for the business. For example, it is used to forecast sales, predict stock market conditions, identify risks, etc.

- **The increased amount of Data –** Every single industry creates a huge amount of data which is ultimately required by them for analysis purposes. And machine learning helps exactly in that. It uses these data to solve problems. The algorithm of Machine Learning helps to complete the most complicated task of an organization with utmost ease.

- **Identify patterns and trends –** The most important part of Machine Learning is finding hidden patterns. With help of statistical techniques, Machine Learning goes into detail and studies the data minutely. On the other hand, understanding data manually will make this a long process. Machine Learning can perform such operations in just a few seconds.

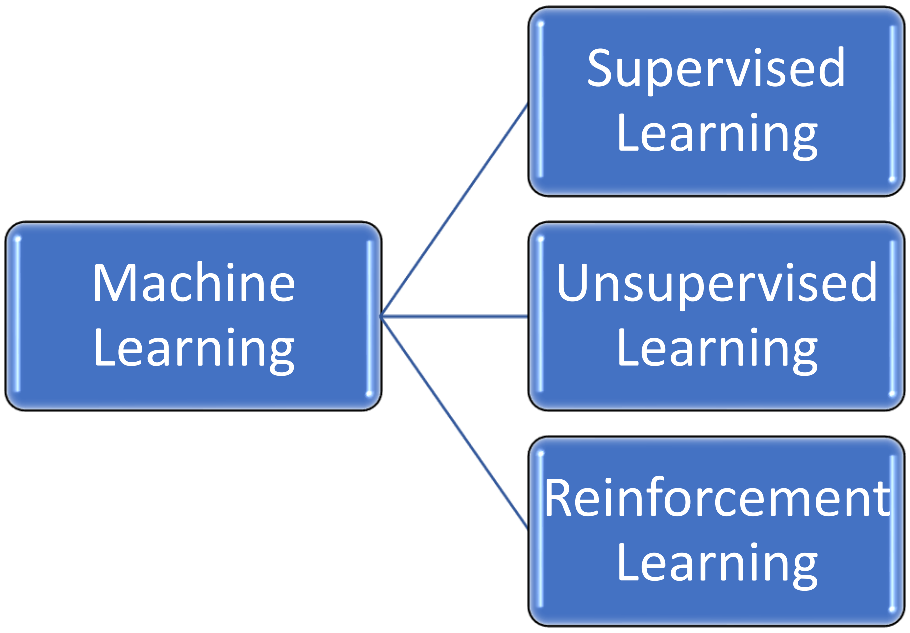
Let's see a few examples of Machine Learning:

- The Gmail of Google uses Algorithms to filter spam messages and label them as spam, promotional, etc.

- Netflix uses Machine Learning to recommend to its users their next shows and series. More than 75% of recommendations are from these algorithms.

Types of Machine Learning

There are various ways of applying the process of Machine Learning, but mainly there are three main categories namely:



Supervised Learning

It is the way where we teach machines using well-labeled data. This learning is easiest to understand and very simple to implement. The user can feed the data learning to the algorithm and then allow the algorithm to predict the results, also allowing the feedback option to make machines know whether they're right or wrong.

Over a while, the algorithms tend to find a relation between different parameters, finding cause and effect relationships between different variables. And by the end of the training, the algorithm learns how data works in the dataset.

Here, the level of accuracy majorly depends mainly on two factors, the availability of labeled data and the algorithm used by machines. In addition to these, other factors are:

- Data Scientist needs to be careful while feeding the data to machines. The data must be balanced and utmost cleaned. Duplicate and missing data would affect the accuracy of machines.

- The input of diversified data would help the machines learn new cases. If there are not enough diversified data the machine will fail to provide reliable answers.

- Avoid overfitting. Overfitting is a situation where the model learns the detail of data that it negatively impacts the result of a new data set. Therefore, it is important to keep test data different from training data.

Application of Supervised Learning

Speech Recognition

In this application, the user will feed his/her voice to the algorithm and it will be able to recognize you. Digital assistance like Google Assistance and Siri will answer only to your voice is one of the biggest examples of a Real-Life Scenario.

Spam Detection

This feature is used to block unreal/ artificial/machine-based messages and emails. You might have seen G-Mail transferring some messages to the spam folder. G-mal has an algorithm that over a while learns various keywords that may be spam and uses that to identify mail as spam.

Bioinformatics

This is one of the most used implementations of supervised learning. Bioinformatics involves the data of human biological such as fingerprints, iris texture, etc. Smartphones in today's generation are uses biological information to increase the system's security.

- Supervised learning is used for fraud transactions and customer detection. The algorithms use historic data to identify the patterns of possible fraud.

- This learning is also used for spam detection. The emails considered spam are directly transferred to the spam folder.

- These algorithms are also used in speech recognition. The algorithm is trained with voice data for voice-activated passwords and voice commands.

Algorithms that come under supervised learning are:

- Linear regression
- Logistic Regression
- Decision trees
- Support vector machine (SVM)
- k-Nearest Neighbours
- Naive Bayes
- SVM

Types of Supervised Learning

Supervised Learning is further divided into two parts, namely:

- Regression
- Classification

Regression

Regressions are used when there’s a relationship between the input variable and the output variable and one of them is a dependent variable and the other is an independent one. Regression analysis helps to understand how a change in the value of an independent variable affects the dependent variable. This analysis is used in prediction, such as weather forecasting, market trends, etc.

Types of Regression

Various types of Regression are used in Machine Learning. Below mentioned are a few of the important types of Regression:

- Linear Regression
- Logistic Regression
- Polynomial Regression
- Support Vector Regression (SVR)
- Decision Tree Regression
- Ridge Regression
- Lasso regression

Classification

- It is a type of supervised learning used to identify categories of new data. In the classification method, an algorithm learns from already present data and then classifies new data into two classes. For example, classifying as yes & no, spam or not spam, cat or dog, etc.
- Unlike regression, the output in classification is classes and not values.

Types of Classification

Various types of Classification are used in Machine Learning. Below mentioned are a few of the important types of Classification :

- K-Nearest Neighbors (K-NN)
- Naive Bayes
- Support Vector Machine (SVM)

Unsupervised Learning

In unsupervised learning, the machines work with unlabelled data. This means that the assistance of humans is not required. The machines are trained with unlabelled datasets and the machine provides insights without human supervision.

This learning aims to find similarities, patterns, and even differences from the unfiltered data.

Unsupervised Learning Algorithms:

Clustering algorithms that come under unsupervised learning algorithms are:

- K-Means clustering
- Hierarchical clustering

- Anomaly detection
- Neural networks

Reinforcement Learning

This learning works on the feedback process, where an AI agent (which is a software component) explores the data with the hit and trial method. It learns from experience and improves its performance.

When the algorithm finds the correct solution, it provides a reward to the algorithm. On the other hand, in the opposite case, the algorithm keeps on working unless and until the correct solution is found.

Application of Reinforcement Learning

- Reinforcement Learning is used in gaming applications. The learning is used to gain super-human performance.

- RL is also used in Robotics. The robots are generally used in the industrial and manufacturing area, and these are made of powerful reinforcement learning.

Text mining uses RL to transfer the free text in documents into organized data to make it suitable for analysis.

Regression in Machine Learning

- Regression analysis is a technique or a statistical method to model and investigates the relationship between an independent variable, features, and a dependent variable or outcome with one or more independent variables.
- An Independent variable is also known as a predictor variable whose values are gathered by experiments. The other variable is called Target variables (dependent variable) as their values can be easily derived from the predictor variable.
- Regression Analysis is widely used for forecasting and prediction.
- It helps us to understand how the value of the dependent variable is changing corresponding to the independent variable when other independent variables are help fixed.
- It also predicts the real and continuous values such as temperature, age, salary, price, etc.

Some Common uses for machine learning regression models:

- Forecasting about the outcomes like stock prices, sales, etc. is done based on Regression analysis.
- Predicting about the success of future retail sales or marketing campaigns to ensure about the resources are being used effectively or not.

- Predicting about the user trends or the customers on streaming services or e-commerce websites.
- Analyzing the datasets so that a relationship between variables and an output can be established.
- Stock prices and rates of interest can be predicted easily by analyzing a variety of factors.
- Creating time series visualizations.

Linear Regression

- Linear regression attempts to find a linear relationship between a target and one or more predictors.
- It is a statistical regression method used for predictive analysis.
- It is one of the most simple and easy algorithms which works on the regression and shows the relationship between the continuous variables
- In machine learning, regression problems can be easily solved by linear regression.
- It shows the linear relationship between the X-axis which is the independent variable and the Y-axis which is a dependent variable hence called linear regression.

Applications and Uses of Regression

- It is a prominent machine learning technique that can be used in various fields from stock markets to scientific research.
- Engine performance can be easily analyzed from test data in automobiles.
- It is used to model causal relationships between parameters in biological systems.
- It can also be used in weather data analysis.
- It is often used in customer survey result analysis and market research studies.

It is also used in observational astronomy for astronomical data analysis.

Types of Linear Regression

Linear Regression is divided into two types:

- Simple Linear regression
- Multiple Linear Regression

A simple linear regression equation is graphed as a straight line, where:

- β0 is the y-intercept of the regression line.
- β1 is the slope.
- E(y) is the mean or we can say that the expected value of y for a given value of x.

Limits of Simple Linear Regression

Regression analysis is commonly used in research, so we get to know that a correlation exists between variables. But correlation is not the same as causation; a relationship between two variables does not mean one causes the other to happen. Even we can say that a line in a simple linear regression that fits the data points may not fully guarantee a cause and effect relationship. Using a linear regression model that will allow you to discover whether a relationship between variables exists or not. To understand exactly what this relationship is about, and whether one variable causes another, we need to work and research

Loss function

- The loss function is a method of evaluating algorithm models of the dataset. And if your predictions are totally off, the loss function will output a higher number.
- It is a way to measure "how well your algorithm models your dataset."
- If your algorithm models are good, it'll output a lower number. When you try to change pieces of your algorithm and improve your model, loss function will tell you if you're doing it right and getting anywhere.
- When your predictions are off, a high number will get produced by the loss function, if they're good it will lead to a lower amount
- It will inform you whether you need to change your algorithm when you try to refine your model or not, It lets one understand how distinct the expected values are from the real value.

Loss Function V/S Cost Function

Loss Function	Cost Function
• Loss Functions are used for the single training examples.	• Cost function is the average loss over the entire training dataset.
• Loss function is also known as error functions	• Optimization strategies in the Cost function aim at minimizing the cost function.
• It measures how well your model performs on a single training example	• It considers the entire training set and then tries to measure how well the model is performing on it.
• Loss function measures the error for a single training example.	• Cost function measures the average error for the entire training set.

Measuring Performance Metrics

Regression analysis aims to model the relationship between a certain number of features and a continuous target variable.

These are some of the performance metrics used for evaluating a regression model:

- Mean Absolute Error (MAE)
- Mean Squared Error (MSE)
- Root Mean Squared Error (RMSE)
- R- Squared
- Adjusted R - squared

Mean Squared Error (MAE)

The average of the squared difference can be easily found by using the mean squared error between the predicted and actual values. Since it has a convex shape, it will be easier to optimize it.

Mean Absolute Error (MSE)

It is the average of the absolute difference between the target value and the predicted by the model. It is not preferred in cases where outliers are prominent.

Root Mean Squared error (RMSE)

Root mean Squared Error is the root of the average of squared residuals. As residuals are the measure of how
distant the points are from the regression line. Thus, RMSE measures the scatter of these residuals.

R- Squared

It is calculated by dividing the sum of squares of residuals
(SSres) from the regression model by the total sum of squares (SStot) of errors from the average model and then just subtract it from 1.

It is also known as the Coefficient of Determination. As it explains the degree to which the input variables explain the variation of the output/ predicted variable.

Adjusted R-squared

Here, N = The Total sample size (number of rows) and P = the number of predictors (number of columns).
There is a limitation of R-squared which is that it will either stay the same or increases just with the addition of some more variables even if there is no relationship with the output variable.

Learning about Ridge Regression

Ridge regression is a tuning method used to analyze any data that suffer from multicollinearity. As it performs L2 regularization. When a multicollinearity issue occurs, least-squares are unbiased and variances are large, this results in predicted values being far away from the actual values.

For a regression machine learning model, the usual regression equation forms the base which can be written as:

$$Y = XB + e$$

- Y is the dependent variable.
- X represents about the independent variables.
- B is the regression coefficients
- And e represents about the errors that are residuals.

Learning about Lasso Regression

Lasso regression is a type of linear regression that uses shrinkage. As shrinkage is where the data values are shrunk towards a central point same as the mean. The lasso procedure encourages simple, sparse models.

"LASSO" stands for Least Absolute Shrinkage and Selection Operator.
It adds the "absolute value of magnitude" of the coefficient as a penalty term to the loss function.

What is Logistics Regression?

Logistic regression is a statistical method used to predict the outcome of a dependent variable based on the previous observation. It is a type of regression analysis that is a commonly used algorithm for solving binary classification problems.
It is a classification algorithm that predicts a binary outcome based on a series of independent variables. Logistic regression can also be used to solve regression problems.
It is also referred to as binomial logistic regression or binary logistic regression, when there are two classes of the response variable it's called multinomial logistic regression.

Some important terms used in regression analysis:

- **Variable:** It refers to any number, characteristics, or even quantity that can be measured or can be counted. Speed, gender, and income are some of the examples.
- **Coefficient:** It is a number that is usually an integer multiplied by the variable that it accompanies.
- EXP: It is a short form of exponential.
- **Outliers:** They are the data points that significantly differ from the rest.
- **Estimator:** They are the algorithms or formula that generates estimates of parameters.
- **Chi-squared test:** It’s a hypothesis testing method to check whether the data is as expected.
- **Standard error:** It is the approximate standard deviation of a statistical sample population**.**
- **Regularization:** It is a method used for reducing the error.
- Multicollinearity: It is an occurrence of intercorrelations between two or more independent variables**.**
- The goodness of fit**:** It’s a description of how well a statistical model fits a set of observations.
- Odds ratio: They are the measure of the strength of association between two events.

What is a logistic function?

In statistics, it is used to describe about the properties of population growth. Sigmoid function and logit function are some variations of the logistic function.

When to use logistic regression:

Logistic regression can be applied to predict about the categorical dependent variable like yes or no, true or false, 0 to 1.

In the case of predictor variables, they can be the part of following categories:

- **Continuous data:** It can be measured on an infinite scale and can take any amount of value between two numbers.
- **Discrete, nominal data:** It fits into the named categories.
- **Discrete, ordinal data:** Here the Data fits into some form of order on a scale.

By logistic regression we can predict about following data:

- An email is spam or not.
- It is going to rain today or not?
- Detecting fraud online transactions
- A contestant will win an election or not?

Some assumptions about logistic regression:

- There is little to no multicollinearity between the independent variables.
- The independent variable is linearly related to the log odds.
- The dependent variable is binary.
- There are no non-meaningful variables as they can lead to error.
- The data sample sizes are larger.
- There are no outliers.

Types of logistic regression

There are three main types of logistic regression:

- Binary logistic regression:
  It is a statistical method to predict about the relationship between a dependent variable and an independent variable. Here the dependent variable is only able to take two values such as yes or no, true or false, and 0 or 1).
  An example of binary logistics is determining whether an email is a spam or not.
- Multinomial logistic regression:
  It is an extension of binary logistic regression. It can have more than two categories of the outcome or dependent variable.
  Multinomial logistic regression is similar to binary logistic regression but can have more than two possible outcomes.
  For example, the dependent variable can be represented by "Type A," "Type B," or" Type C".

- Ordinal logistic regression:
  It is also known as ordinal regression, used to predict the dependent variable with more than three possible ordered types.
  For example, the dependent variable may represent "Strongly disagree" or "Strongly Agree".

Advantages of logistic regression algorithm:

- It is simple to understand, easy to implement, and efficient to train.
- It performs well when the dataset is linearly separable.
- It provides good accuracy for the smaller dataset.
- It doesn’t make any assumptions about the distribution of classes.
- It offers the direction of association (it can be positive or negative).

Some Disadvantages of logistic regression algorithm:

- It constructs linear boundaries.
- Sometimes it is challenging to obtain complex relationships. Algorithms like neural networks are more suitable and powerful.
- They can't solve non-linear problems.
- They can be only used to predict discrete functions.
- They are sensitive to outliers.

Linear Regression Vs Logistic Regression

Linear Regression	Logistic Regression
It is used to predict about the continuous dependent variables by using a given set of independent variables	It is used to predict about the categorical dependent variable using a given set of independent variables
Linger regression is used when you need to solve problems related to regression	Logistic regression is used when you need to solve Classification problems.
Here we predict the values of continuous variables.	Here we predict the values of categorical variables.
We find the best fit line in linear regression by which we can easily be able to predict about the output.	We find the S- curve by which we can easily classify the samples.
Here in linear regression least square estimation method is used for the estimation of accuracy.	Here Maximum likelihood estimation method is used for the estimation of accuracy.
The output of linear regression must be a continuous value such as price, age, etc.	The output of logistic Regression must be a categorical value such as 0 or 1, Yes or No, etc.
Here it is required that the relationship between a dependent variable and an independent variable must be linear	In Logistic regression, it is not required to have a linear relationship between dependent and independent variables.
There may be collinearity between the independent variables.	There should not be collinearity between the independent variables.

What is a Decision Tree?

- Decision Tree comes under the Supervised learning technique that can be used for both problems like classifications and Regression, but mostly it is used and preferred for solving Classification problems. As it is a tree-structured classifier, where internal nodes represent about the features of a dataset, branches represent about decision rules and each leaf node represents the outcome.
- The decision tree consists of two nodes named as Decision Node and Leaf Node. Decision Nodes are used to make any decision and also have multiple branches, whereas leaf nodes are just the output of those decisions and also it doesn't contain any further branches.
- All of the tests and decisions are made or performed on the basis of features of the given dataset.
- A decision tree is the graphical representation for getting all of the possible solutions to a problem/ decision based on given conditions.

- A decision tree has its structure completely same to a tree, it starts with the root node and expands on further branches, and constructs a tree-like structure.
- If we need to build a tree, CART algorithm can be used which stands for Classification and Regression tree algorithm.
- It can contain Yes/No as well as numeric data too.

Some Terminologies of Decision Tree

- **Root Node:** It is where the decision tree starts from. This reflects the whole dataset, which can be easily split into two or more homogeneous sets.

- **Leaf Node:** It is the last output node in the decision tree after this the tree will not be more separated.

- **Splitting:** It separates the decision node/root node according to the specified conditions into the sub-nodes.

- **Branch/Sub Tree:** It is a subtree formed just by splitting the tree.

- **Pruning:** It is a practice of removing unnecessary branches from the tree is pruning.

Parent/Child node: Root node of the tree is considered as the parent node and the other nodes are known as child nodes.

Why do we need to use Decision Tree?

Machine learning has various algorithms, and choosing the best suitable algorithm for the dataset you are working upon is the main point to remember while creating models in machine learning.

Decision Trees has the same capabilities to think as human while making a decision so that it can be easy to understand.

And the logic behind the decision tree is very clear and can be understood easily as it shows a tree-like structure.

Advantages of the Decision Tree

- It is a simple process to understand as it follows the same process we humans follow while making any decision in real life.
- It is a very useful technique for solving decision-related problems.
- It helps to think about all the possible outcomes for a problem
- There is less requirement for data cleaning compared to other algorithms.

Disadvantages of the Decision Tree

- When a decision tree is formed it contains lots of layers, which makes it complex.
- Sometimes it may have an overfitting issue, which can be resolved using the random forest algorithm.

Overfitting of Decision Trees

When the model completely fits the training data but it fails to generalize testing of the unseen data then it refers to the Overfitting of the Decision Tree. This condition takes place when the model memorizes the noise of the training data and fails to capture important patterns.
A decision tree that is perfectly fit performs well for the training data but poorly for the unseen test data.
When the decision tree is allowed to train to its full strength then the model will surely be going to overfit the training data.

There are some techniques to handle the problem of overfitting such as:

- **Pruning:** Here in this technique, the decision tree can grow easily up to its full depth. It's a technique to remove the parts of the decision tree to prevent it from growing to its full depth and this can be done by tuning the hyperparameters of the decision tree model to prune the trees and prevent them from overfitting.

There are two types of pruning:

- **Pre-Pruning**
- **Post-Pruning**
- **Ensemble- Random Forest:**

  It is an ensemble technique for classification and regression by bootstrapping multiple decision trees. Overfitting is prevented by random forest as it follows bootstrap sampling and aggregation techniques.
  It can be easily implemented using the Scikit-Learn library.

Information Gain

- It is the calculation of entropy changes when a dataset is segmented based on an attribute.
- How much information a feature gives us about a class can be measured by Information Gain.
- According to the value of information, a gain node can be split and a decision tree can be built.

Entropy

It is metric so that impurity in a given attribute can be measured, in the data it defines randomness and can be calculated as:

- Entropy(s)= -P(yes)log2 P(yes)- P(no) log2 P(no)
- Here, S = Total number of samples.
- P(yes) = probability of yes
- P(no) = probability of no

Gini Index

Gini index is a measure of the impurity or purity used in the CART (Classification and Regression Tree) algorithm when creating a decision tree.
In Comparison to a high Gini index, an attribute with a low Gini Index should be preferred.

Ensemble Learning

Ensemble models in machine learning combine the decision and insights from different and multiple models to perform better and increase their overall decision-making capabilities.

In learning models, there are some major sources like noise, variance, and bias. The ensemble methods here in machine learning help to minimize these errors causing factors that eventually ensures about the accuracy and stability of the machine learning algorithms.

You can say that Ensembles are a divide and conquer approach and we use them to improve the overall performance.

Some Simple Ensemble Methods are:

- **Mode:** Moving on to statistics, "mode" is the number or value that appears in a dataset. In the ensemble technique, machine learning professionals use number of models to make predictions about each of the data points present in the model. All these predictions are made by different models which are taken as separate votes. And in final these predictions made by the models are treated as the ultimate prediction.
- **Mean/Average:** In this technique, data analysts go with the average predictions made by all the models while making the ultimate prediction.
- **The Weighted Average :** In this most of the time data scientist assign different weights to all of the models in order to just make a prediction, where the assigned weight defines about the relevance of each model.

Standard ensemble learning strategies such as:

- **Bagging**
- **Stacking**
- **Boosting**

Bagging Ensemble learning

- It is an ensemble learning method that seeks a diverse group of ensemble members by varying the training data. Where it involves using a single machine learning algorithm which is an unpruned decision tree, and training each model on a different sample of the same training dataset , after this the predictions made by the ensemble members are then get combined using simple statistics, such as voting or averaging.
- It is the manner in which each sample of the dataset is prepared to train ensemble members and each model gets its own unique sample of the dataset. Rows which are drawn from the dataset at random, although with the replacement can be the example.

Some popular ensemble algorithms based on this approach are:

- Bagged Decision Trees.
- Random Forest
- Extra Trees

Stacking Ensemble Learning

It is a method that seeks a diverse group of members just by varying the model types fit on the training data and later on using that model to combine predictions.

Some popular ensemble algorithms are:

- Stacked Models (canonical stacking)
- Blending
- Super Ensemble

Boosting ensemble Learning

- It is an ensemble method where training data can be changed to focus attention on the examples that previous fit models on the training dataset have gotten wrong.
- The Models are fit and added to the ensemble sequentially such that the second model attempt to correct the predictions of the first model and the third corrects to the second model, and so on.

Random Forest

Random forest is a Supervised Machine Learning Algorithm that is used widely in Classification and Regression problems. It builds decision trees on different samples and takes their majority vote for classification and average in case of regression.

One of the most important features of the Random Forest Algorithm is that it can handle the data set containing continuous variables as in the case of regression and categorical variables as in the case of classification. It performs better results for classification problems.

Working of Random Forest

Bagging– It creates a different training subset from sample training data with replacement & the final output is based on majority voting. For example, Random Forest.

Boosting– It combines weak learners into strong learners by creating sequential models such that the final model has the highest accuracy. For example, ADA BOOST, XG BOOST

Difference Between Random Forest & Decision Trees

Decision Tree	Random Tree
Decision trees normally suffer from the problem of overfitting if it's allowed to grow without any control.	Random forests are created from subsets of data and the final output is based on average or majority ranking and hence the problem of overfitting is taken care of.
A single decision tree is faster in computation.	It is comparatively slower.
When a data set with features is taken as input by a decision tree it will formulate some set of rules to do prediction.	Random forest randomly selects observations, builds a decision tree and the average result is taken. It doesn't use any set of formulas.

What is the AdaBoost Algorithm?

AdaBoost also called Adaptive Boosting is a technique in Machine Learning used as an Ensemble Method. The most common algorithm used with AdaBoost is decision trees with one level which means with Decision trees with only 1 split. These trees are also called Decision Stumps.
What this algorithm does is that it builds a model and gives equal weights to all the data points. It then assigns higher weights to points that are wrongly classified. Now all the points which have higher weights are given more importance in the next model. It will keep training models until and unless a low error is received.

XG Boost

XGBoost is an algorithm that has recently been dominating applied machine learning and Kaggle competitions for structured or tabular data.
It is an implementation of gradient boosted decision trees designed for speed and performance.

XG Boost Feature

The library is laser-focused on computational speed and model performance, as such there are few frills. Nevertheless, it does offer a number of advanced features like gradient boosting, Regularized gradient boosting and Stochastic gradient boosting.

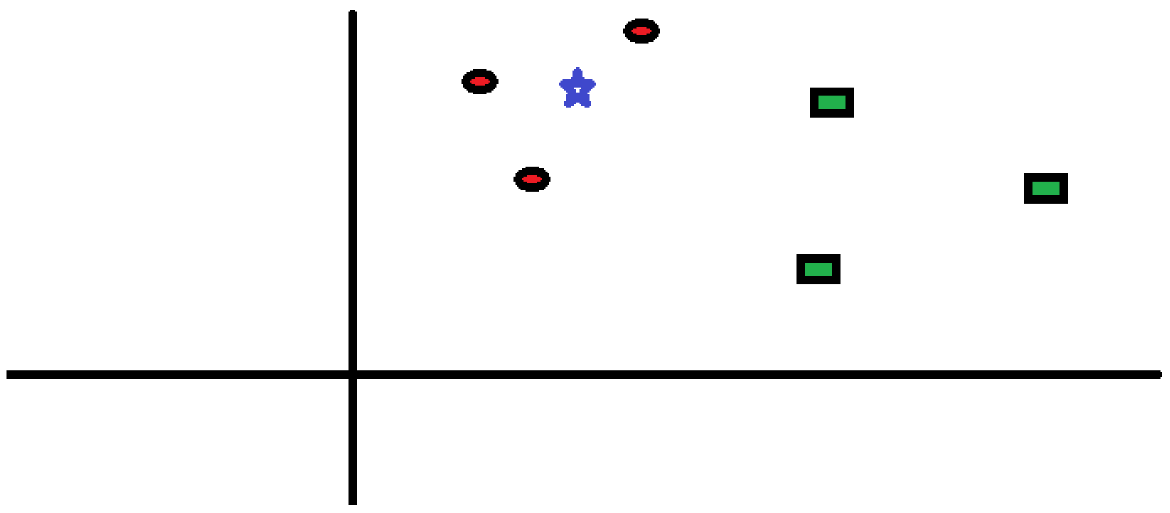
K Nearest Neighbour

Introduction

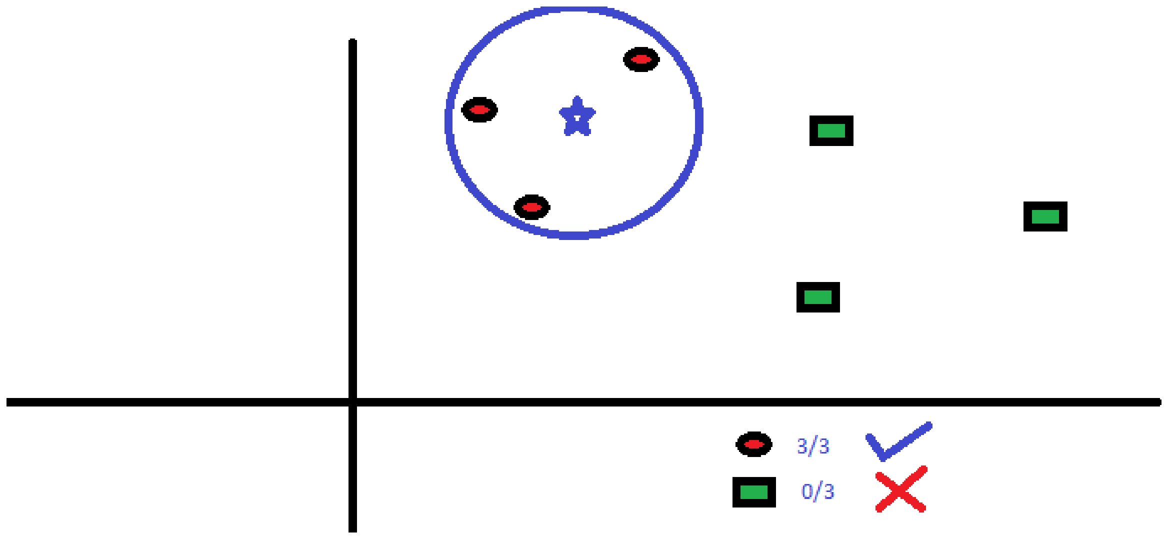
Unsupervised Learning is the algorithm that does not depends on labelled data at the time of input to make a machine learn a function and produce an appropriate output. In the KNN process, the “K” is selected and the closest neighbour to it is determined. The crucial element in this algorithm is a selection of the "K" element.

How does the KNN algorithm work?

To understand the working of the algorithm, we will take an example: there are two categories below i.e. circles and squares.



We're required to find the category of a blue star. The star can be in the category circle or square. We will now make a circle with BS as the center just as big as enclosing only three data points on the plane.



The three closest points to the star are all the circles and therefore, the star would belong to the category of circle. In the whole process, the selection of the K is most important as the accuracy of the result depends upon it.

Selecting the value of K

- There is not any predefined method to determine the value of K apart from elbow method. You can start computing with a random value of K and later on increase it or decrease it according to the accuracy.
- The value of K depends on the amount of data. In the case of different scenarios, the value of K may vary. It is similar to the hit and trial method.

- Selecting small K will lead to an unstable output. Those outputs won’t be 100% accurate. On the hand, if we increase the value of K, our predictions will become more stable and they’ll be more likely to make more accurate predictions.

The goal of KNN is to find the nearest neighbours of a particular data point. To perform this task, KNN has a few requirements:

Determine Distance metrics

To determine which data point is closest to the query it is necessary to calculate the distance between the measures. Euclidean Distance, Manhattan Distance, Minkowski Distance and Hamming Distance.

Advantages of KNN

- **Easy Implementation**
  This is one of the first algorithms that data scientists will learn. The Algorithm's simplicity and accuracy make it easy to implement.
- **Adapts easily**
  Whenever new training samples are added, the algorithm adjusts itself for the new training data.
- **Fewer hyperparameters**
  KNN requires the “K” values and distance metric. These hyperparameters are low as compared to other machine learning algorithms.

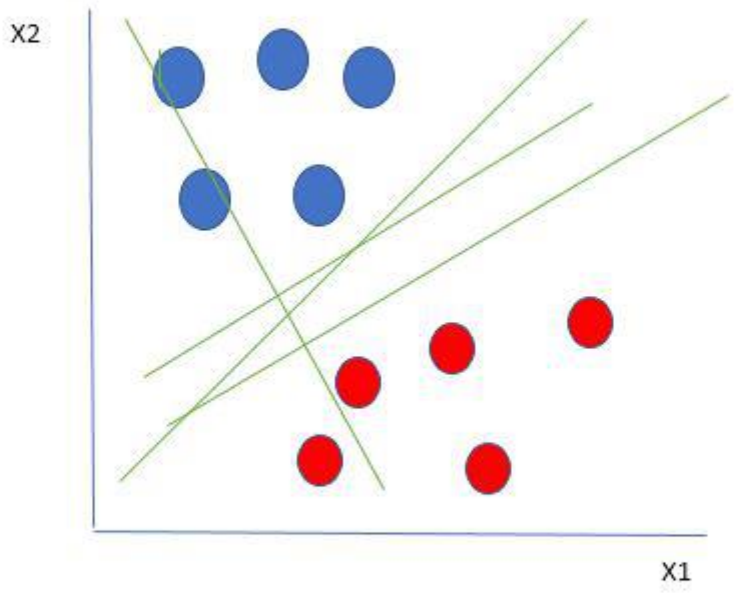
Disadvantages of KNN

- **Doesn’t scale well**
  KNN takes more memory and data storage compared to other machine learning algorithms. More time and storage ultimately increase business expenses.
- **Problem of dimensionality**
  KNN doesn’t perform well with high-dimensional data. The additional features many times increase the number of errors.
- **Possibility of overfitting**
  Due to the problem of dimensionality, KNN is prone to the problem of overfitting. The situation depends on the value of "K". Lower value of K can overfit the data while a high value of K can underfit it.

Support Vector Machine (SVM)

Introduction to Support Vector Machine (SVM)

SVM is the type of Supervised Learning algorithm that is used for both classification and regression. The main goal of SVM is to search and find a Hyperplane in N-dimensional space to differentiate and classify the data points.



Now, how will we select the best Hyperplane that separates our data point? One valid method is to select the Hyperplane that represents the largest separation between the two classes.

If such a Hyperplane is present, it is called a maximum-margin Hyperplane.

Types of SVM

Linear SVM	Non-Linear SVM
Can be separated with a single line.	Cannot be easily separated with a single line.
Data Classification is done with the help of Hyperplane.	Use of Kernels to classify the data.
Easy classification with a single straight line.	Mapping of data into HD space is required to be done for classification.

Margins in SVM

Soft Margin:

In the linearly separable case, the Support Vector Machine is trying to find the line that maximizes the margin, which is the distance between those closest dots to the line. This is called the Soft Margin.
The motive of soft margin is simple, to keep the margin wide as possible and allow SVM to make a certain number of mistakes.

Hard Margin:

If we strictly impose that all instances must be off the street and to the right side, this is called hard margin classification. There are two main issues with hard margin classification. First, it only works if the data is linearly separable, and second, it is quite sensitive to outliers.
In the case of hard margin, classification in SVM is very rigid. It tries to work well even in a training set and thus, causes overfitting.

Regularization Method in SVM:

Machine Learning often faces a problem named overfitting, which means failure of the training set to perform well on unseen and new data. Regularization is the technique that can be used to reduce the number of errors and avoid the situation of overfitting.

What is Kernel?

Kernels are used to solve non-linear problems and this is known as the Kernel trick method. It helps to frame the hyperplane in an extremely high dimension without raising any complexity.

Important components of Kernel SVC

The two important components of Kernel SVC are Gamma and the 'C' parameter:

- **Gamma:** Gamma decides the amount of effect single training has on the final output. This affects the decision boundaries in the model. When the values of Gamma are small, the data points farther from it are considered similar. On the other hand, larger points cause the data point to be much closer and ultimately result in overfitting.
- **The 'C' parameter:** This parameter controls the regularization amount of data. Greater values of C lower the amount of regularization. And, lower values of C result in C regularization.

Types of Kernel Functions

Linear Kernel
This is the simplest Kernel function. The Kernel is used for text classification problems as they can easily be separated through Linear Kernel. The formula for the same is:

F(x, y) = sum (x.y)

In the formula, x and y represent the data that you're supposed to classify.

- **Polynomial Kernel**
  This type of Kernel is used when the training data are all standardized and normalized. Polynomial Kernel is less preferred due to its less efficiency and accuracy. The formula for this function is:

  F(x, y) = (x.y+1) ^d

  The dot in the formula shows the product of both the values and d shows the degree.

- **Gaussian Radial Basis Function (RBF)**
  RBF is used for non-linear data classification and is the most preferred form of Kernel function for the same.

  F(x, y) = exp (-gamma * ||x - y||^2)

The value of gamma varies in the range of 0 to 1. The user is supposed to manually input the value of Gamma and the most used value is 0.1.

- **Sigmoid Kernel**
  It is mostly preferred for neural networks. The formula for Sigmoid Kernel is:

  F(x, y) = tanh (αxay + c)

- **Anova radial basis Kernel**
  Similar to Gaussian Kernel, Anova Kernel is used for multidimensional regression problems. And, as the name suggests it comes under the category of radial basis Kernel. The formula for Anova Kernel is:

$$k(x,y) = \sum_{k=1}^{n} \exp(-\sigma(x^k - y^k)^2)^d$$

- **Rational Quadratic Kernel**
  This Kernel function is less intensive than Gaussian Kernel but it can be alternatively used in its place when Gaussian becomes expensive. The formula for the same is:

$$k(x,y) = 1 - \frac{||x-y||^2}{||x-y||^2 + c}$$

- **Multiquadric Kernel**
  This Kernel function can be used in the same situation as Rational Quadratic Kernel. It is an example of a non–positive definite Kernel. The formula for Multiquadric Kernel is:

$$k(x,y) = \sqrt{||x-y||^2 + c^2}$$

Advantages of SVM

- SVM provides a technique called Kernel. With the usage of this function, any complex problem can be solved easily. The kernel is applied to non-linear classes and is also known as a non-parametric function.
- SVM doesn't get the problem of overfitting and it performs well in terms of memory.

- SVM can efficiently solve both classification and regression problems. SVM is used for classification problems and SVR is used for regression problems.
- Compared with Naive Bayes (another technique for classification) SVM is faster and more accurate at prediction.
- SVM can be applied to semi-supervised learning models. It also applies not only to labeled data but even to unlabeled data.

Disadvantages of SVM

- SVM can prove to be quite costly for the user. The cost of training them especially non-linear models is high.
- Data in SVM need to have feature vectors in advance. This needs pre-processing and this is not always an easy task.
- Selecting an appropriate Kernel function is tricky and complex. Selecting higher or lower Kernel can prove to be wrong for your output.
- SVM tends to take a long training time on mainly large data sets.
- SVM requires a good amount of computation capability. This is required to tune the hyper-parameters including the value of the 'C' parameter and gamma.

Naive Bayes

It is a supervised learning algorithm which helps us to solve classification problems; Naive bayes is mostly used in text classification where high dimensional training datasets are included.

Why is it known as Naive Bayes?

Naive Bayes comprises of two words Naive and Bayes and they can described as:

- **Naive:** It is known as Naive because it assumes about the occurrence of a certain features is independent of the occurrence of other features. Such as the fruits that can be easily identified on the bases of color , shape, and taste. Hence each of the feature contributes individually to identify that it is an apple without depending on each other.
- **Bayes:** It is known as bayes since it depends on the principle of the bayes Theorem.

Understanding about maths behind the Bayes' Theorem

The Bayes's Theorem is also known as bayes' Rule or the law of the bayes as it finds the probability of an event occurring given the probability of another event that has already occurred or going to be. Bayes' theorem is stated by a mathematically equation:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Here A and B are the events and P(B) ≠ 0

- Basically, here we are trying to find probability of the event A, given that the event B is true. Event B is termed as evidence.
- P (A) is the priori of A (the prior probability, i.e. probability of event before evidence is seen). The evidence is an attribute value of an unknown instance (here, it is event B).
- P (A|B) is a posterior probability of B, i.e. the probability of event after evidence is seen.

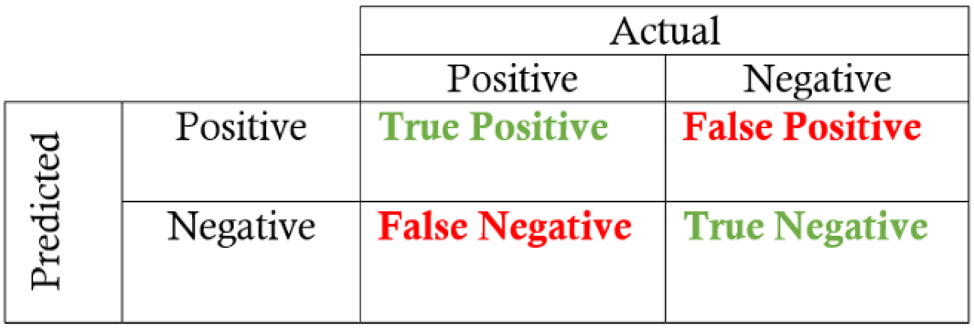
Some advantages of naive Bayes classifier:

- It is one of the easy ML algorithms to predict about a class of the datasets.
- It can handle binary as well as Multi class Classifications.

- It performs in the Multi-class predictions as compared to other algorithms.
- Text classification problems can be easily solved by using it

When we need to use evaluation metrics?

As Accuracy is not a valid choice of evaluation for problems related to classification problems which are well balanced and not skewed or no class imbalance.



		Actual	
		Positive	Negative
Predicted	Positive	True Positive	False Positive
	Negative	False Negative	True Negative

Confusion Matrix

Confusion Matrix is just the performance check and measurement for the machine learning classification problem where there can be two or more classes of the output. It is a table with 4 different combinations of the predicted and actual values.

By the help of Confusion matrix you will be able to measure about Recall, Precision, Specificity, Accuracy and F1-Score.

Accuracy = (TP + TN) / (TP+FP+FN+TN)

Precision = TP/TP+FP

Recall = TP/TP+FN

True Positive Rate(tpr) = TP/TP+FN

False Positive Rate(fpr) = FP/FP+TN

$$F_1 = 2 \times \frac{precision \times recall}{precision + recall}$$

True Positive:
Interpretation: You predicted covid positive and it’s true.
You predicted that a patient is covid positive and he/she actually is.

True Negative:
Interpretation: You predicted negative and it's true.
You predicted that another patient is not and he/she actually is not.

False Positive: (Type 1 Error)
Interpretation: You predicted positive and it's false.
You predicted about a patient who is positive but in actually he/she is not.

False Negative: (Type 2 Error)
Interpretation: You predicted negative and it's false.
You predicted about a patient that he/she is not covid positive but in real she is.

Unsupervised Learning Flow

The unsupervised algorithm is handling data without prior training, working with unlabeled data. The main purpose of unsupervised learning is working under the conditions of result being unknown.

Unsupervised machine learning algorithm is mostly used to:

- Exploring about the structure of the information and detect distinct patterns
- Extracting about valuable insights
- Implementing this into the operation in order to increase the efficiency of the decision-making process

Some Common types of unsupervised learning approaches

Unsupervised learning models are utilized for three main tasks named as clustering, association and dimensionality reduction.

Clustering

Clustering algorithms are used to process raw, unclassified data object into the groups represented by structures or patterns in the information. It can be categorized into a few types such as exclusive, overlapping, hierarchical and probabilistic.

Exclusive and Overlapping Clustering

It is a form of grouping that stipulates a data point that can exist only in one cluster and this can also be referred as "hard" clustering. The K- means clustering algorithm is the example of exclusive clustering.

Understanding about K- means clustering

It is a common example of an exclusive clustering method where data points are assigned into the K groups; here k represents the number of the clusters based on the distance from each group's centroid.

Hierarchical clustering

Hierarchical clustering is also known as hierarchical cluster analysis (HCA), It is an unsupervised clustering algorithm that can be categorized in two ways:

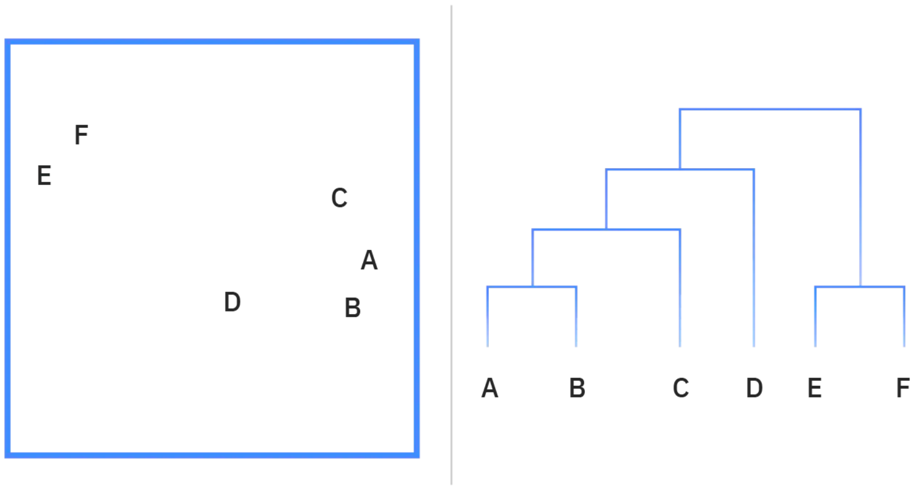
- ✓ **Agglomerative**

It is considered as “bottoms-up-approach.” Its data points are isolated as separate groupings initially, and then they get merged together on the basis of similarity until one of the cluster achieved.

In Agglomerative method there are some commonly used measure similarities known as:

- ✓ **Ward’s linkage:** It states that the distance between two clusters is defined by the increase in the sum of the squared after getting clustered or merged.
- ✓ **Average linkage**: It is defined by the mean distance between two points on each cluster.
- ✓ **Complete (or maximum) linkage:** This method is defined by the maximum distance between the two points in each clusters
- ✓ **Single (or minimum) linkage:** It is defined by the minimum distance between the two points in each of cluster.

Divisive clustering: It can be defined as the opposite of agglomerative clustering, as it takes a top-down approach. And In this case a single data cluster is divided on the differences between the data points.

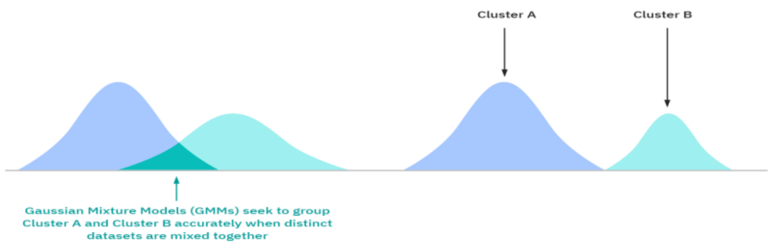


Probabilistic clustering

It is an unsupervised technique that helps to solve problems related to density estimation or “soft” clustering problems. In probabilistic clustering, data points are clustered based that they belong to a particular distribution.

Gaussian Mixture Models

They are classified as the mixture of the models, made up of an unspecified number of probability distribution function .They are leveraged to determine about the Gaussian, or normal, probability distribution a given data point belongs to. If the mean or variance is known then determination of a given data point can be done



Some challenges of unsupervised learning

- It has to deal with Computational complexity due to a high volume of training data.
- Unsupervised learning has to face longer training times.
- It has higher chances of showing inaccurate results.
- Human intervention to validate output variables.
- Lack of transparency into the basis on which data was clustered.

Principle Component Analysis (PCA)

PCA is used for reducing the dimensions of the variables. It transforms a large set of data variables into a smaller one that holds almost all the information of the large data set. This is used because smaller data sets make the analyzing process easier.

Basic Terminologies in Factor Analysis

What is Dimensionality?

Dimensionality is several variables in the dataset. In simple words, it is the total number of columns present in the dataset.

What is Correlation?

It depicts how strongly two variables are interconnected to each other. That is if one variable gets changed, another variable will also get affected due to interdependency.

What is orthogonal?

The term is used to describe that the variables are not related to each other. When the correlation between the pair is zero, it is said to be Orthogonal.

What are Eigenvectors?

If there is a square matrix M, a non-zero vector v is given. Then v will be eigenvector if Av is the scalar multiple of v.

Steps involved in PCA: -

There are four different steps to perform PCA, all of which are mentioned below:

Step.1 Standardizing the Dataset

$$Z = \frac{value - mean}{standard\ deviation}$$

Step.2 Calculating Covariance Matrix

For Population

$$\text{Cov}(x,y) = \frac{\Sigma\,(x_i - \overline{x}) * (y_i - \overline{y})}{N}$$

For Sample

$$\text{Cov}(x,y) = \frac{\Sigma\,(x_i - \overline{x}) * (y_i - \overline{y})}{(N-1)}$$

Step.3 Calculating eigenvalues and eigen vectors
Eigenvalues and eigen vectors are the linear algebra components that are computed from the covariance matrix to identify the principal components of the data.
Principal components are the new variables that are a combination of the initial variables. Representing information via principal components will allow the user to reduce the dimensionality without adjusting with information quantity.

Step.4 Feature Vector
In this step, the user selects from all the principal components framed in the previous step. The user decides whether to keep all the components or to remove the components of lesser significance. This is an important step in the process of PCA. As suppose the user decides to keep x number of components out of n then the matrix will contain a total of x components.

Feature Scaling

In machine learning, it is required to bring all the variables in the same proportion, this is done to make sure that one number doesn't impact the results. This whole process is known as feature scaling. This process can make difference between a weak machine learning model and the stronger one. The two most used techniques for the process of feature scaling are Normalization and Standardization.

- Normalizing Scaling
  Also known as Min-Max Scaling, Normalizing scaling is used when we want to range our values between two numbers mostly between [0,1] or [-1,1]. The formula to calculate it is:

$$\boldsymbol{X_{new}} = \frac{(\boldsymbol{X} - \boldsymbol{X\_min})}{(\boldsymbol{X\_max} - \boldsymbol{X\_min})}$$

- Standardization
  Also known as Z-score scaling, standardization scaling is performed by subtracting the values from the mean ad then dividing it by the standard deviation. The formula for it is:

$$\mathbf{X\_new} = \frac{(\mathbf{X} - \mathbf{mean})}{\mathbf{Std}}$$

Feature Encoding

Machine learning works only with numerical data. And, therefore, it is necessary to convert any other form of data into numerical form. This process is known as Feature Encoding. The inserted data is majorly processed with the following techniques of encoding:

- **One-hot encoding**
  One hot encoding is the process of converting categorical data variables into numerical values. This process is performed so that the data can be provided to machine learning algorithms. The one-hot encoding provides numerical values to label values.
  For example, there are two variables named red and blue. We will provide binary values to them as 0 to red and 1 to blue. The data can be represented in binary terms and can successfully be transferred to machine learning algorithms.
- **Target-mean encoding**
  This process is another way of transforming categorical data variables into numerical ones. The process includes removing and then replacing the categorical data with the average value of the data.
  However, the main challenge in the process of the target-mean encoding process is the problem of overfitting. As we're changing the variables based on the mean value that may result in data leakage.
- **Frequency encoding**

Feature Selection

Feature selection is the process of separating the consistent and relevant features used in the model. The main goal of feature selection is to improve the performance of the model by reducing the size of datasets. The process decreases the number of variables by removing the irrelevant variables. Different types of Feature Selection Methods are:

Difference Between Filter, Wrapper, and Embedded Method

Filter Method	Wrapper Method	Embedded Method
Faster as compared to wrapper method considering the time factor.	High computation compared to additional features	Speed between Filter and wrapper method in consideration with the time factor.
Low possibility of over-fitting	High chances of over-fitting	The method used to reduce the problem of over-fitting
Do not use Machine Learning Algorithms.	Works with specific Algorithms of Machine Learning	Feature selection by the model building process.
Examples can be Correlation, ANOVA, ETC.	Examples can be Forward and backward selection.	An example can be Lasso, Ridge Regression.

Outlier Treatment

An outlier is a particular data point in a data set that is extremely different from the rest of the observation.

Outlier can be caused due to the following mentioned reasons,

- Error in recording of data
- Error in observation
- Measurement Error
- Data variability

Different methods to detect outliers

There are four different outlier detection techniques:

- Numeric Outlier

  Numeric Outlier is capable to detect outlier in a One-dimension space. The data is measured in terms of Interquartile Range (IQR). First, the 1st and 3rd quartile (Q1 and Q3) are calculated. An Outlier will the data point that resides outside the interquartile range. This Technique helps in easy detection of an outlier.

- Z-score

  Z-score identifies Outlier by assuming Gaussian distribution of the data. The mean is found out and an outlier is the distribution that will be far from the mean. The formula for Z-score is:

$$z = \frac{x - \mu}{\sigma}$$

- DBSCAN

  The method is based on DBSCAN clustering method suitable for multi-dimensional feature space. Here, the data is divided between three different points i.e. Core points, Border Points, and Noise points.

  Core points are the points that have at least Minimum Points neighbouring data points. Border Points are the values nearby Core Points and are part of dataset. The data points far from data set are Nosie Points and are identified as outliers.

- Isolation Forest

  Isolation Forest uses the concept of isolation number. The isolation number is the total number of splits required to differentiate a particular data point. Here, the outlier have lower isolation number as compared to non-outlier point.

DEEP LEARNING

Biological Neuron

It is a typical biological individual cell mostly found in the animal brain, composed of a cell nucleus which is soma and many extended tendrils. Tendrils can be classified into one *dendrite*, whreceiveseive short electrical impulses from other neurons and bring it to the cell body. Wneuronsuron receive sufficient number of signals from the other neuron *axons* send information from the cell body to other neurons.

In short, Artificial Neural networks are an ensemble of a large number of simple artificial neurons. This network learns to conducta few tasks such as recognizing an apple by firing a neuron in a certain way when a given particular input is an apple. Next, we will see a perceptron which was proposeatin the very beginning of the research area of machine learning and is also a building block of a Neural network.

Artificial Neuron (Perceptron)

An artificial neuron only in a few aspects resembles a biological neuron. A neural network is an interconnected system of perceptron's, so it is safe to say perceptron's are the foundation of any neural network. Perceptron's can be viewed as building blocks in a single layer in a neural network, made up of four different parts:

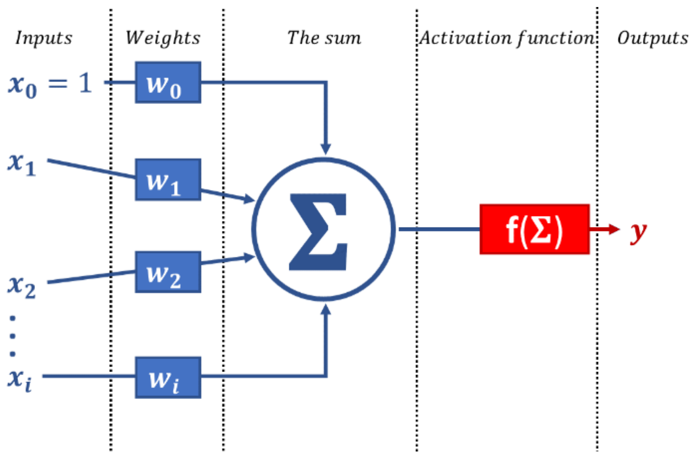
1. Input Values or One Input Layer

2. Weights and Bias

3. Net sum

4. Activation function

5. Outputs



An artificial single neuron is represented by a mathematical function. It takes i inputs x, and each of them usually has its own weight w. The neuron calculates the sum and it is passed through the activation function to the network further.

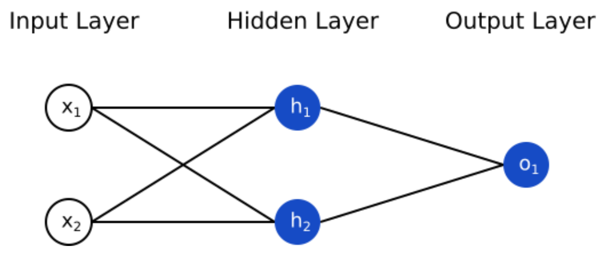
Relationship between Biological neural network and artificial neural network:

Biological Neural Network	Artificial Neural Network

Dendrites	Inputs
Cell nucleus	Nodes
Synapse	Weights
Axon	Output

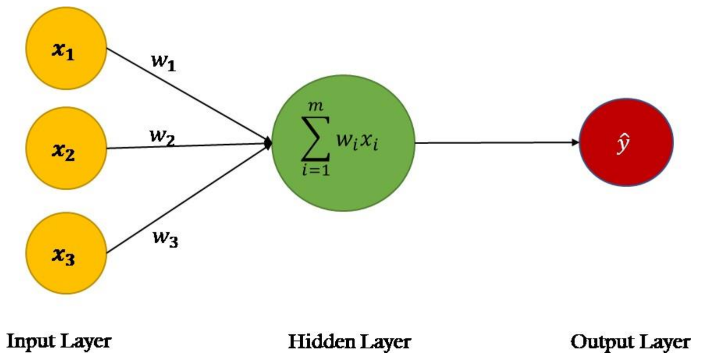
What is Neural Network

A neural network is nothing more than a bunch of neurons connected together. Here's what a simple neural network might look like:
This network has 2 inputs, a hidden layer with 2 neurons (h1 and h2), and an output layer with 1 neuron (o1). Notice that the inputs for o1 are the outputs from h1 and h2 — that's what makes this a network.



Components of a Neural Network

The building block of a neural network is the single neuron. The diagram below shows the structure of a neutron



The **input to the neuron is x**, which has a **weight w** associated with it. Weights shows the strength of the particular node. The weight is the intrinsic parameter, the parameter the model has control over in order to get a better fit for the output. When we pass an input into a neuron, we multiply it by its weight, giving us x * w.

The second element of the input is called **bias**. A bias value allows you to shift the activation function curve up or down. The bias adds an element of unpredictability to our model, which helps it generalize and gives our model the flexibility to adapt to different unseen inputs when using testing data.

The combination of the bias and input produces our output y, giving us a formula of **w*x + b =y**.

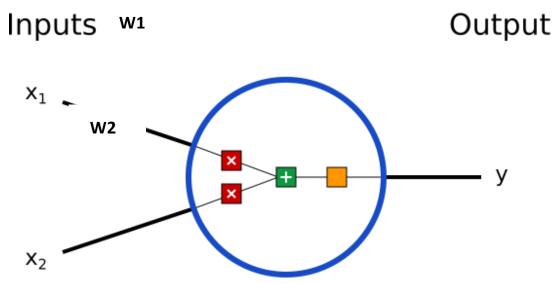
This should look familiar as a modification of the equation of a straight line, y = mx + c. Neural Networks are made up of tens, hundreds, or many even thousands of interconnected neurons, which run its own regression

Layers

Neural networks organize neurons into layers. A layer in which every neuron is connected to every other neuron in its next layer is called a dense layer.

Working of Perceptron

A perceptron consists of four parts: input values, weights and a bias, a weighted sum, and activation function. Assume we have a single neuron and two inputs x1, x2 multiplied by the weights w1, w2 respectively as shown below



$x_1 \to x_1 * w_1$

$x_2 \to x_2 * w_2$

Next, all the weighted inputs are added together with a bias b:

$$(x_1 * w_1) + (x_2 * w_2) + b$$

This looks like a good function, but what if we wanted the outputs to fall into a certain range say 0 to 1. Finally, the sum is passed through an activation function

$$y = f(x_1 * w_1 + x_2 * w_2 + b)$$

The activation function is used to turn an unbounded input into an output that has a nice, predictable form. An activation function is a function that converts the input given (the input, in this case, would be the weighted sum) into a certain output based on a set of rules.

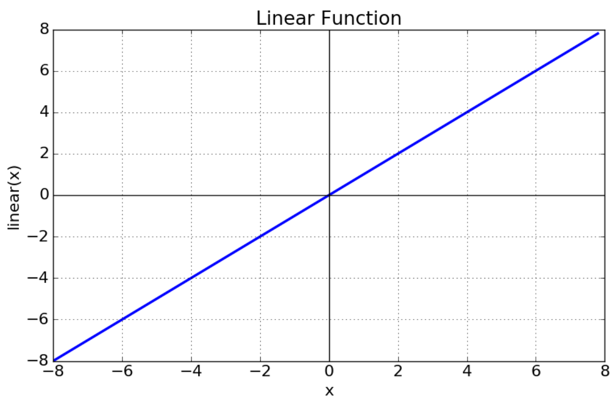
Activation Function

Activation is used to determine the output of neural network like yes or no. It maps the resulting values in between 0 to 1 or -1 to 1 etc. The Activation Functions can be basically divided into 2 types-

1. Linear Activation Function
2. Non-linear Activation Functions

Linear or Identity Activation Function

As you can see the function is a line or linear. Therefore, the output of the functions will not be confined between any range.

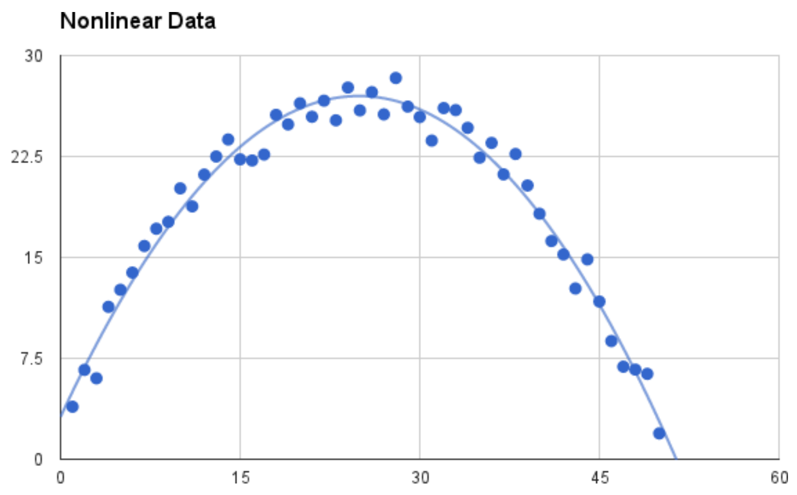


Equation: f(x) = x

Range: (-infinity to infinity)

Non-linear Activation Function

The Nonlinear Activation Functions are the most used activation functions. Nonlinearity helps to makes the graph look something like this

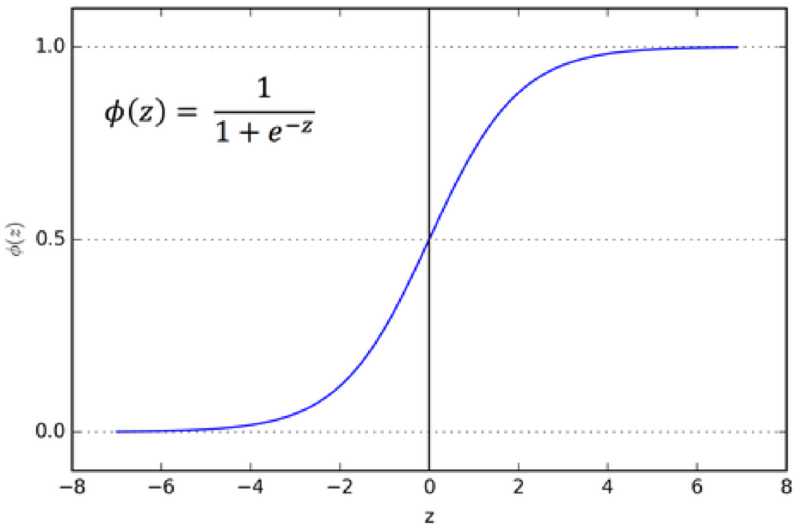


It makes it easy for the model to generalize or adapt with variety of data and to differentiate between the output.

Different Activation Functions are:

Sigmoid or Logistic Activation Function

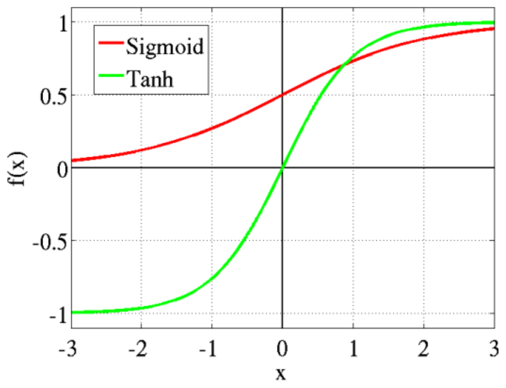
The Sigmoid Function curve looks like a S-shape.



The main reason why we use sigmoid function is because it exists between **(0 to 1).** Therefore, it is especially used for models where we have to **predict the probability** as an output. Since probability of anything exists only between the range of **0 and 1,** sigmoid is the right choice.

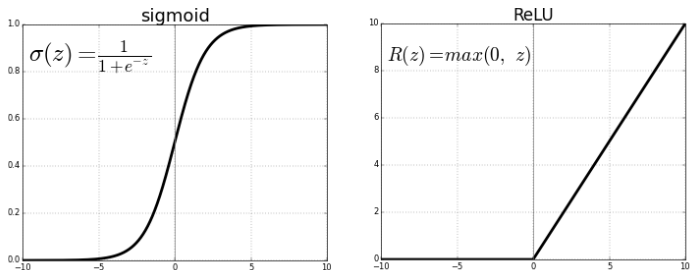
Tanh or hyperbolic tangent Activation Function

tanh is also like logistic sigmoid but better. The range of the tanh function is from (-1 to 1). tanh is also sigmoidal (s - shaped).



The advantage is that the negative inputs will be mapped strongly negative and the zero inputs will be mapped near zero in the tanh graph. **ReLU (Rectified Linear Unit) Activation Function**

The ReLU is the most used activation function in the world right now.Since, it is used in almost all the convolutional neural networks or deep learning.

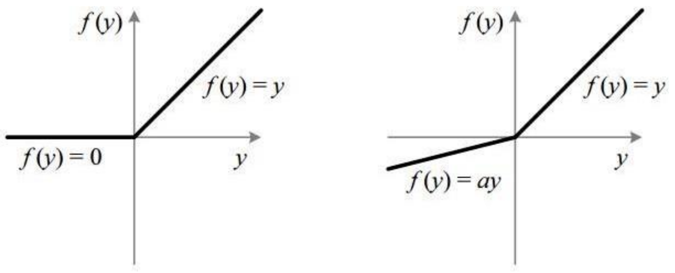


As you can see, the ReLU is half rectified (from bottom). f(z) is zero when z is less than zero and f(z) is equal to z when z is above or equal to zero.

Range: [0 to infinity)

Leaky ReLU

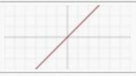
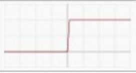
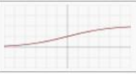
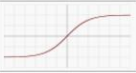
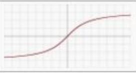
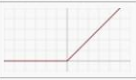
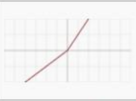
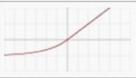
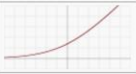
It is an attempt to solve the dying ReLU problem



The leak helps to increase the range of the ReLU function. Usually, the value of a is 0.01 or so.

Therefore, the range of the Leaky ReLU is (-infinity to infinity).

Activation Function Cheat sheet

Name	Plot	Equation	Derivative
Identity		$f(x) = x$	$f'(x) = 1$
Binary step		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x \neq 0 \\ ? & \text{for } x = 0 \end{cases}$
Logistic (a.k.a Soft step)		$f(x) = \frac{1}{1+e^{-x}}$	$f'(x) = f(x)(1-f(x))$
TanH		$f(x) = \tanh(x) = \frac{2}{1+e^{-2x}} - 1$	$f'(x) = 1 - f(x)^2$
ArcTan		$f(x) = \tan^{-1}(x)$	$f'(x) = \frac{1}{x^2+1}$
Rectified Linear Unit (ReLU)		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Parameteric Rectified Linear Unit (PReLU)[2]		$f(x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Exponential Linear Unit (ELU)[3]		$f(x) = \begin{cases} \alpha(e^x - 1) & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} f(x) + \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
SoftPlus		$f(x) = \log_e(1 + e^x)$	$f'(x) = \frac{1}{1+e^{-x}}$

Multi-layer neural Networks (ANN)

A multi-layer neural network contains more than one layer of artificial neurons or nodes. They differ widely in design. It is important to note that while single-layer neural networks were useful early in the evolution of AI, the vast majority of networks used today have a multi-layer model.

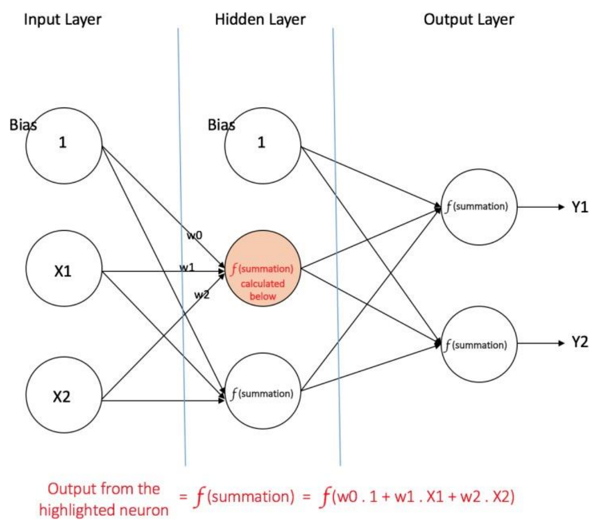
Some terminologies in Multi-layer neural network (ANN)

1. **Backpropagation**, a procedure to repeatedly adjust the weights so as to minimize the difference between actual output and desired output

2. **Hidden Layers**, which are neuron nodes stacked in between inputs and outputs, allowing neural networks to learn more complicated features (such as XOR logic)

Architecture of an Artificial Neural Network

To understand the concept of the architecture of an artificial neural network, we have to understand what a neural network consists of. In order to define a neural network that consists of a large number of artificial neurons, which are termed units arranged in a sequence of layers. Let's us look at various types of layers available in an artificial neural network.


Input Layer
Hidden Layer
Output Layer
Bias
1
Bias
1
X1
X2
w0
w1
w2
f(summation) calculated below
f(summation)
f(summation)
f(summation)
Y1
Y2
Output from the highlighted neuron = f(summation) = f(w0 . 1 + w1 . X1 + w2 . X2)

Input Layer: As the name suggests, it accepts inputs in several different formats provided by the programmer. In the diagram, there are three nodes in the Input Layer. The Bias node has a value of 1. X1 and X2 are taken by the other two nodes as external inputs (which are numerical values depending upon the input dataset).

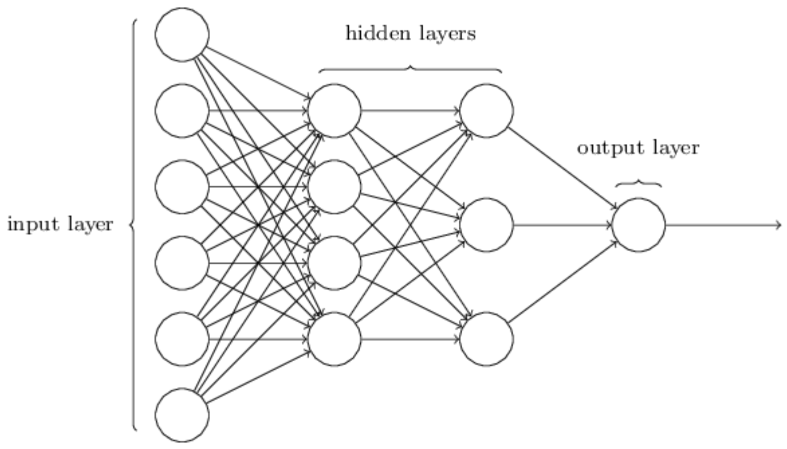
Hidden Layer: The hidden layer presents in-between input and output layers. It performs all the calculations to find hidden features and patterns. As shown in the figure, the hidden layer also has three nodes. The Hidden layer also has three nodes where the Bias node has an output of 1. The output of the other two nodes of Hidden layer depends on the outputs from the Input layer (1, X1, X2) and the weights associated with the edges.

The figure indicates output calculation for one of the hidden nodes

Similarly, it is possible to measure the output from another hidden node. Note that f corresponds to the activation function. These outputs are then fed into the Output layer nodes.

Output Layer: The input goes through a series of transformations using the hidden layer, which finally results in output that is conveyed using this layer. - There are two nodes in the output layer that take inputs from the hidden layer and perform identical computations as seen for the hidden node that is highlighted in the diagram. The values determined (Y1 and Y2) as a result of these computations are considered as the result of the multi-layer perceptron.

For example, the following four-layer network has two hidden layers:

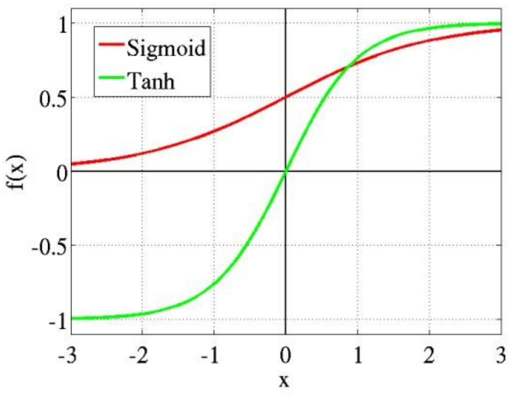


Function of Hidden Neurons

The hidden neurons act as feature detectors; as such, they play a critical role in the operation of a multilayer neural networks. As the learning process progresses across the multilayer neural networks, the hidden neurons begin to gradually "discover" the salient features that characterize the training data. They do so by performing operations on the input data into a new space called the feature space. In this new space, the classes of interest in a pattern-classification task, for example, may be more easily separated from each other than could be the case in the original input data space.

Tanh or hyperbolic tangent Activation Function

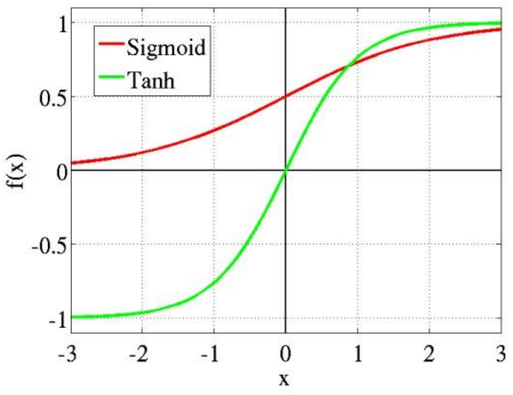
tanh is also like logistic sigmoid but better. The range of the tanh function is from (-1 to 1). tanh is also sigmoidal (s-shaped).



The advantage is that the negative inputs will be mapped strongly negative and the zero inputs will be mapped near zero in the tanh graph.

Tanh or hyperbolic tangent Activation Function

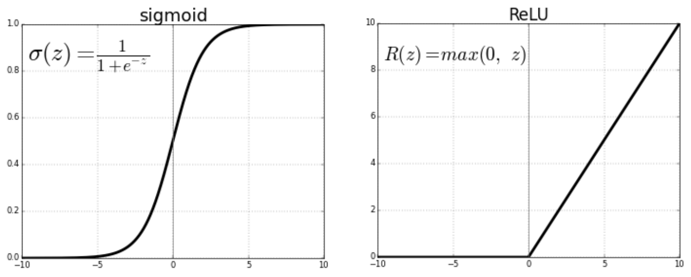
tanh is also like logistic sigmoid but better. The range of the tanh function is from (-1 to 1). tanh is also sigmoidal (s-shaped).



The advantage is that the negative inputs will be mapped strongly negative and the zero inputs will be mapped near zero in the tanh graph.

ReLU (Rectified Linear Unit) Activation Function

The ReLU is the most used activation function in the world right now. Since, it is used in almost all convolutional neural networks or deep learning.

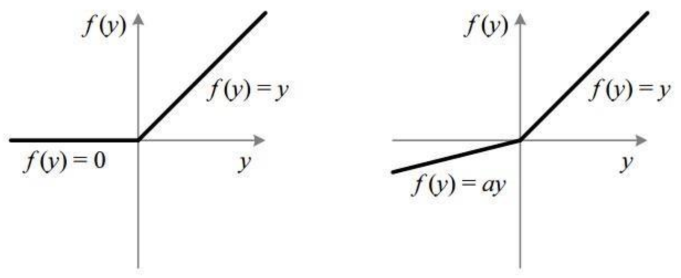


As you can see, the ReLU is half rectified (from the bottom). f(z) is zero when z is less than zero and f(z) is equal to z when z is above or equal to zero.

Range: [0 to infinity)

Leaky ReLU

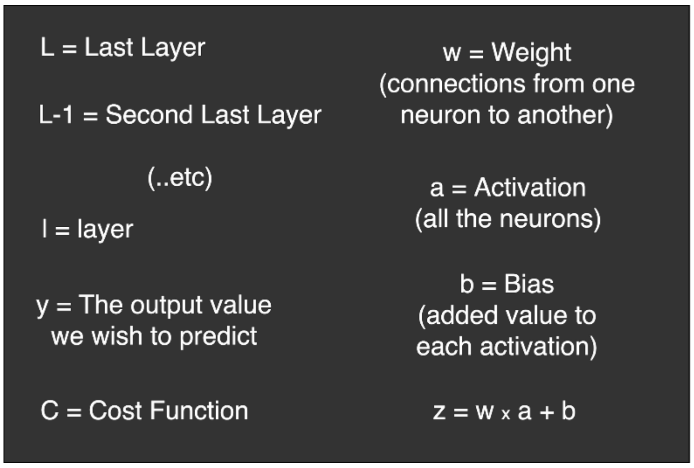
It is an attempt to solve the dying ReLU problem



The leak helps to increase the range of the ReLU function. Usually, the value of a is 0.01 or so. Therefore, the range of the Leaky ReLU is (-infinity to infinity).

Computing Gradients

Now, before the equations, let's define what each variable means. We have already defined some of them, but it's good to summarize. Some of this should be familiar to you



L = Last Layer	w = Weight (connections from one neuron to another)
L-1 = Second Last Layer	
(..etc)	a = Activation (all the neurons)
l = layer	
y = The output value we wish to predict	b = Bias (added value to each activation)
C = Cost Function	z = w x a + b

Firstly, let's start by defining the relevant equations.

$$z^{(L)} = w^{(L)} \times a + b$$

$$a^{(L)} = \sigma\left(z^{(L)}\right)$$

$$C = (a^{(L)} - y)^2$$

More on the cost function later in the cost function section.

Mathematically, this is why we need to understand partial derivatives since they allow us to compute the relationship between components of the neural network and the cost function.

And as should be obvious, we want to minimize the cost function. When we know what affects it, we can effectively change the relevant weights and biases to minimize the cost function.

$$\frac{\partial C}{\partial w^{(L)}} = \frac{\partial C}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L)}} \frac{\partial z^{(L)}}{\partial w^{(L)}} = 2\left(a^{(L)} - y\right) \sigma'\left(z^{(L)}\right) a^{(L-1)}$$

Although w^e are not directly found in the cost function, we start by considering the change of w in the z equation, since that z equation holds a w. Next, we consider the change of z^L in a^L, and then the change a^L in function C. Effectively, this measures the change of a particular weight with a cost function.

We need to move backward in the network and update the weights and biases. One equation for weights, one for biases, and one for activations:

$$\frac{\partial C}{\partial w^{(L)}} = \frac{\partial C}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L)}} \frac{\partial z^{(L)}}{\partial w^{(L)}}$$

$$\frac{\partial C}{\partial b^{(L)}} = \frac{\partial C}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L)}} \frac{\partial z^{(L)}}{\partial b^{(L)}}$$

$$\frac{\partial C}{\partial a^{(L-1)}} = \frac{\partial C}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L)}} \frac{\partial z^{(L)}}{\partial a^{(L-1)}}$$

Each partial derivative from the weights and biases is saved in a gradient vector, that has as many dimensions as you have weights and biases. The gradient is the triangle symbol ∇, and n is several weights and biases:

$$-\nabla C(w_1, b_1, \ldots, w_n, b_n) = \begin{bmatrix} \frac{\partial C}{\partial w_1} \\ \frac{\partial C}{\partial b_1} \\ \vdots \\ \frac{\partial C}{\partial w_n} \\ \frac{\partial C}{\partial b_n} \end{bmatrix}$$

You compute the gradient according to a mini-batch (often 16 or 32 is best) of your data, i.e., you subsample your observations into batches. For each observation in your mini-batch, you average the output for each weight and bias. Then the average of those weights and biases becomes the output of the gradient, which creates a step in the average best direction over the mini-batch size.

Then you would update the weights and biases after each mini-batch. Each weight and bias are 'nudged' a certain amount for each layer l:

$$w^{(l)} = w^{(l)} - \text{learning rate} \times \frac{\partial C}{\partial w^{(l)}}$$

$$b^{(l)} = b^{(l)} - \text{learning rate} \times \frac{\partial C}{\partial b^{(l)}}$$

The learning rate is usually written as an alpha α or eta η.

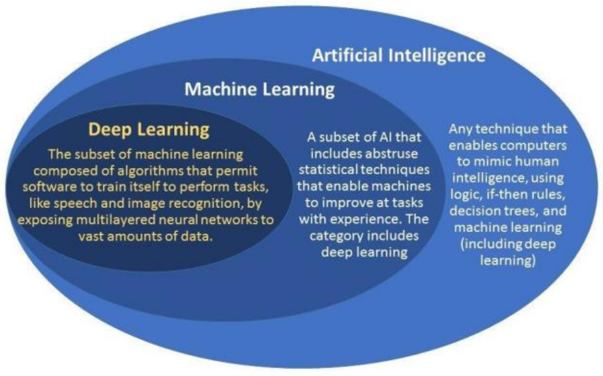
Gradient Descent

Gradient descent is an optimization algorithm that's used when training a machine learning model. Gradient descent is essentially used to find the values of the parameters of a function (coefficients) that minimize, as much as possible, a cost function. The goal of Gradient Descent is to minimize the objective convex function f(x) using iteration. If they're pretty good, they're going to yield a lower amount. Your loss function will inform you whether or not you're changing when you tune your algorithm to try to refine your model. 'Loss' lets one understand how distinct the expected value is from the real value.

What is AI and Deep learning?

Artificial Intelligence

Artificial intelligence (AI) is a broad area of computer science that focuses on creating intelligent machines which can perform tasks that would usually require human intelligence. While AI is a multidisciplinary science with many methods, developments in machine learning and deep learning are causing a paradigm shift in nearly every field of the tech industry.


Artificial Intelligence
Machine Learning
Deep Learning
The subset of machine learning composed of algorithms that permit software to train itself to perform tasks, like speech and image recognition, by exposing multilayered neural networks to vast amounts of data.
A subset of AI that includes abstruse statistical techniques that enable machines to improve at tasks with experience. The category includes deep learning
Any technique that enables computers to mimic human intelligence, using logic, if-then rules, decision trees, and machine learning (including deep learning)

Deep Learning

Deep learning is a subset of machine learning, not anything distinct from it. As we are about to start deep learning, many of us try to find out what the difference between machine learning and deep learning is. Machine learning and deep learning both are all about learning from past experience and making predictions based on future evidence.

In deep learning, artificial neural networks are used to learn from previous data (a mathematical model mimicking the human brain). At a high level, above diagram illustrates the differences between machine learning and deep learning.

Some Real-Life Applications of Deep Learning

Computer Vision

Text Analysis & Understanding

Speech Recognition

Computer Games

AI Cybersecurity

Health Care

General Flow of Deep Learning Project

Any deep learning project, including predictive modeling, can be broken down into five common tasks:
1. Define and prepare the problem
2. Summarize and understand data
3. Process and prepare data
4. Evaluate algorithms
5. Improve results

Workflow of AI Project

We can define the AI workflow in 5 stages.

1. Gathering data
2. Data pre-processing
3. Researching the model that will be best for the type of data
4. Training and testing the model
5. Evaluation

The Importance of Data Splitting

Supervised learning is about creating models that precisely map the given inputs (independent variables, or predictors) to the given outputs (dependent variables, or responses). What's most important to understand is that you usually need unbiased evaluation to properly use these measures, assess the predictive performance of your model, and validate the model.
This means that you can't evaluate the predictive performance of a model with the same data you used for training. You need evaluate the model with fresh data that hasn't been seen by the model before. You can accomplish that by splitting your dataset before you use it.

Stratification

Let's assume you are doing a multiclass classification and have an imbalanced dataset that has 5 different classes. You do a simple train-test split that does a random split totally disregarding the

distribution or proportions of the classes. What happens in this scenario is that you end up with a train and a test set with totally different data distributions.
The solution to this problem is something called stratification which will lock the distribution of classes in train and test sets. In order to get a similar distribution, we need to stratify the dataset based on the class values. So, we pass our class labels part of the data to this parameter and check what happens. Hence your model won't face the problem of validation against an imbalanced test set and give you a wrong sense of its actual performance.

K-Folds Cross Validation

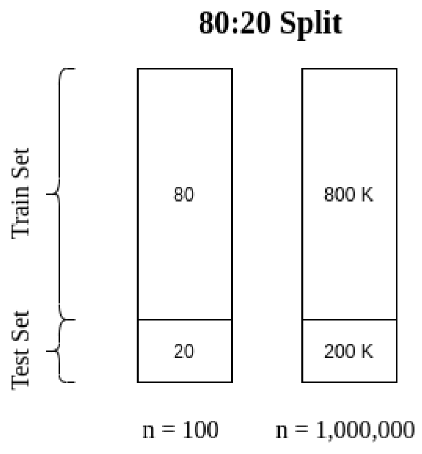
In K-Folds Cross Validation we split our data into k different subsets (or folds). We use k-1 subsets to train our data and leave the last subset (or the last fold) as test data. We then average the model against each of the folds and then finalize our model.

How to Split the Dataset?

If we search the Internet for the best train-test ratio, the first answer to pop will be 80:20. This means we use 80% of the observations for training and the rest for testing. Older sources and some textbooks would tell us to use a 70:30 or even a 50:50 split. On the contrary, sources on deep learning or big data would suggest a 99:1 split.
The dataset size implies a split ratio. Obviously, using the same ratio on datasets with different sizes results in varying train and test set sizes

Train-Validation-Test Split Evaluation

Splitting your dataset is essential for an unbiased evaluation of prediction performance. In most cases, it's enough to split your dataset randomly into three subsets:
1. The **training set** is applied to train, or fit, your model. For example, you use the training set to find the optimal weights, or coefficients, for linear regression, logistic regression, or neural networks.



2. The **validation set** is used for unbiased model evaluation during hyperparameter tuning. For example, when you want to find the optimal number of neurons in a neural network you experiment with different values. For each considered setting of hyperparameters, you fit the model with the training set and assess its performance with the validation set.

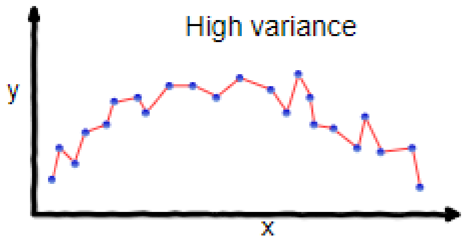
3. The **test set** is needed for an unbiased evaluation of the final model. You shouldn't use it for fitting or validation.

What is Bias?

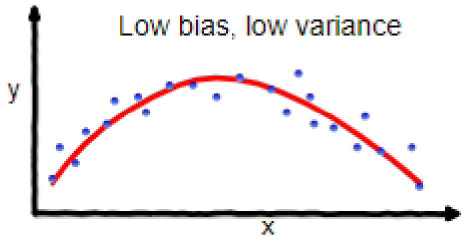
Bias is the difference between the average prediction of our model and the correct value which we are trying to predict. Model with high bias pays very little attention to the training data and oversimplifies the model. It always leads to high error on training and test data. High Bias can also be termed as underfitting. Underfitting is the case where the model has “not learned enough” from the training data, resulting in low generalization and unreliable predictions.

What is Variance?

Variance is the variability of model prediction for a given data point or a value which tells us spread of our data. Model with high variance pays a lot of attention to training data and does not generalize on the data which it hasn’t seen before. As a result, such models perform very well on training data but has high error rates on test data. High Variance can also be termed as Overfitting. Overfitting is the case where the overall cost is really small, but the generalization of the model is unreliable. This is due to the model learning “too much” from the training data set.



A good balanced for the above model will be:



Bias-Variance Trade off

Ultimate goal of any machine learning algorithm is to build a prediction model with Low Bias and Low Variance that is measure of reduced error and correct prediction power of any model.
 High Variance-High Bias — The model is inconsistent and also inaccurate on prediction

 Low Variance-High Bias — Models is consistent but low on prediction

 High Variance-Low Bias — Somewhat accurate but inconsistent on prediction

 Low Variance-Low Bias — Ideal scenario, the model is consistent and highly accurate on prediction

High Bias is mainly caused due to:

1. If the hypothesis function is too simple.
2. If the hypothesis function uses very few features.

Basic Recipe to Solve High bias problem

Always first check for Bias. To check for high bias, observe and infer from the training data performance. Solution for highly biased model:

High Variance is mainly caused due to:

1. If the hypothesis function is too complex.
2. If the hypothesis function uses too many features.

Regularization

This is a form of regression, that constrains/ regularizes or shrinks the coefficient estimates towards zero. In other words, this technique discourages learning a more complex or flexible model, so as to avoid the risk of overfitting.

Different Regularization Techniques in Deep Learning

L2 & L1 regularization

L1 and L2 are the most common types of regularization. These update the general cost function by adding another term known as the regularization term.
Cost function = Loss (say, binary cross entropy) + Regularization term
Due to the addition of this regularization term, the values of weight matrices decrease because it assumes that a neural network with smaller weight matrices leads to simpler models. Therefore, it will also reduce overfitting to quite an extent.
In L2, we have:

$$Cost\,function = Loss + \frac{\lambda}{2m} * \sum \|w\|^2$$

Here, lambda is the regularization parameter. It is the hyperparameter whose value is optimized for better results. L2 regularization is also known as weight decay as it forces the weights to decay towards zero (but not exactly zero).
In L1, we have:

$$Cost\,function = Loss + \frac{\lambda}{2m} * \sum \|w\|$$

In this, we penalize the absolute value of the weights. Unlike L2, the weights may be reduced to zero here. Hence, it is very useful when we are trying to compress our model. Otherwise, we usually prefer L2 over it.

Dropout

This is the one of the most interesting types of regularization techniques. It also produces very good results and is consequently the most frequently used regularization technique in the field of deep learning.

Data Augmentation

The simplest way to reduce overfitting is to increase the size of the training data. In machine learning, we were not able to increase the size of training data as the labeled data was too costly.

But, now let’s consider we are dealing with images. In this case, there are a few ways of increasing the size of the training data – rotating the image, flipping, scaling, shifting, etc. In the below image, some transformation has been done on the handwritten digits’ dataset.

This technique is known as data augmentation. This usually provides a big leap in improving the accuracy of the model.

Early stopping

Early stopping is a kind of cross-validation strategy where we keep one part of the training set as the validation set. When we see that the performance on the validation set is getting worse, we immediately stop the training on the model. This is known as early stopping.

Batch Normalization

Normalization is a data pre-processing tool used to bring the numerical data to a common scale without distorting its shape.
Generally, when we input the data to a machine or deep learning algorithm, we tend to change the values to a balanced scale. The reason we normalize is partly to ensure that our model can generalize appropriately.
But what is the reason behind the term "Batch" in batch normalization? A typical neural network is trained using a collected set of input data called batch. Similarly, the normalizing process in batch normalization takes place in batches, not as a single input.

By using Batch normalization, we will make neural networks faster and more stable through adding extra layers in a deep neural network. The new layer performs the standardizing and normalizing operations on the input of a layer coming from a previous layer.
Advantages of Batch Normalization

- Speed Up the Training
- Handles internal covariate shift
- Smoothens the Loss Function

What are Hyperparameter?

Hyperparameters are all the training variables set manually with a pre-determined value before starting the training. These include embeddings, number of layers, activation function, learning rate, dropout rate and so on.

We have four main strategies available for searching for the best configuration of our hyperparameters:

- Babysitting (aka Trial & Error)
- Grid Search
- Random Search
- Bayesian Optimization

Grid Search

Grid Search – a naive approach of simply trying every possible configuration.
Here's the workflow:

- Define a grid on n dimensions, where each of these maps for an hyperparameter. e.g. n = (learning_rate, dropout_rate, batch_size)
- For each dimension, define the range of possible values: e.g. batch_size = [4, 8, 16, 32, 64, 128, 256]
- Search for all the possible configurations and wait for the results to establish the best one: e.g. C1 = (0.1, 0.3, 4) -> acc = 92%, C2 = (0.1, 0.35, 4) -> acc = 92.3%, etc...

Random Search

The only real difference between Grid Search and Random Search is on the step 1 of the strategy cycle – Random Search picks the point randomly from the configuration space.

Optimization
Optimizers are algorithms or methods used to change the attributes of your neural network such as weights and learning rate in order to reduce the losses.

Gradient Descent
Gradient Descent is the most basic but most used optimization algorithm. It's used heavily in linear regression, classification and neural network algorithms. Backpropagation in neural networks also uses a gradient descent algorithm.
Gradient descent is a first-order optimization algorithm which is dependent on the first-order derivative of a loss function. It calculates that which way the weights should be altered so that the function can reach a minima.
Algorithm: $\theta=\theta-\alpha\cdot\nabla J(\theta)$
Advantages:

- Easy computation
- Easy to implement
- Easy to understand

Disadvantages:

- May trap at local minima
- Weights are changed after calculating gradient on the whole dataset. So, if the dataset is too large than this may take years to converge to the minima
- Requires large memory to calculate gradient on the whole dataset

Stochastic Gradient Descent
It's a variant of Gradient Descent. It tries to update the model's parameters more frequently. In this, the model parameters are altered after computation of loss on each training example. So, if the dataset contains 1000 rows SGD will update the model parameters 1000 times in one cycle of dataset instead of one time as in Gradient Descent.
Algorithm: $\theta= \theta-\alpha\cdot\nabla J(\theta,x(i),y(i))$, where $\{x(i) ,y(i)\}$ are the training examples.

Mini-Batch Gradient Descent
It's best among all the variations of gradient descent algorithms. It is an improvement on both SGD and standard gradient descent. It updates the model parameters after every batch. So, the dataset is divided into various batches and after every batch, the parameters are updated.
Algorithm: $\theta=\theta-\alpha\cdot\nabla J(\theta; B(i))$, where $\{B(i)\}$ are the batches of training examples

Adam Adam (Adaptive Moment Estimation) works with momentums of first and just because we can jump over the minimum, we want to decrease the velocity a little bit for a careful search.

Advantages:

- The method is too fast and converges rapidly.
- Rectifies vanishing learning rate, high variance.

Disadvantages:

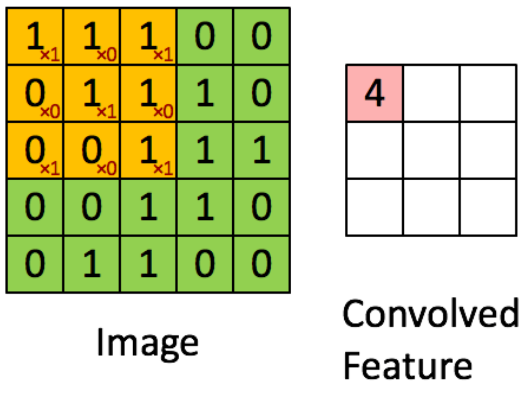
- Computationally costly.

Introduction to CNN

Convolutional Neural Network (ConvNet/CNN) is a Deep Learning method that can take in an input image and assign importance (learnable weights and biases) to distinct aspects and objects in the image while being able to distinguish between them. Comparatively speaking, a ConvNet requires substantially less pre-processing than other classification techniques. ConvNets have the capacity to learn these filters and properties, whereas in primitive techniques filters are hand-engineered.

A ConvNet's architecture was influenced by how the Visual Cortex is organised and is similar to the connectivity network of neurons in the human brain. Only in this constrained area of the visual field, known as the Receptive Field, do individual neurons react to stimuli. The entire visual field is covered by a series of such fields that overlap.

Convolution Layer — The Kernel



The green area in the demonstration above mimics our 5x5x1 input image, I. The Kernel/Filter, K, which is symbolised by the colour yellow, is the component that performs the convolution process in the initial portion of a convolutional layer. K has been chosen as a 3x3x1 matrix.

The Kernel shifts nine times as a result of Stride Length = 1 (Non-Strided), conducting a matrix multiplication operation between K and the area P of the picture that the kernel is now hovering over each time.

Until it has parsed the entire width, the filter travels to the right with a specific Stride Value. Once the entire image has been traversed, it hops back up to the image's beginning (on the left) with the same Stride Value.

Pictures having several channels, like RGB images, have a kernel with the same depth as the input image. A squashed one-depth channel Convoluted Feature Output is produced by performing matrix multiplication across the Kn and In stacks ([K1, I1]; [K2, I2]; and [K3, I3]). All of the results are then added together with the bias.

The Convolution Operation's goal is to take the input image's high-level characteristics, such edges, and extract them. There is no requirement that ConvNets have just one convolutional layer. Typically, low-level features like edges, colour, gradient direction, etc. are captured by the first

ConvLayer. With more layers, the architecture adjusts to High-Level characteristics as well, giving us a network that comprehends the dataset's images holistically, much like we do.

The procedure yields two different types of results: one where the dimensionality of the convolved feature is decreased as compared to the input, and the other where it is either increased or stays the same.

Applying Valid Padding in the first scenario or Same Padding in the second accomplishes this.

When the 5x5x1 image is enhanced into a 6x6x1 image and the 3x3x1 kernel is applied to it, we see that the convolved matrix has the dimensions 5x5x1. Therefore, Same Padding was born.

On the other hand, if we carry out the identical operation without padding, we are given a matrix called Valid Padding that has the same dimensions as the kernel itself (3x3x1).

Pooling Layer

The Pooling layer, like the Convolutional Layer, is in charge of shrinking the Convolved Feature's spatial size. Through dimensionality reduction, the amount of computing power needed to process the data will be reduced. Furthermore, it aids in properly training the model by allowing the extraction of dominating characteristics that are rotational and positional invariant.

Max Pooling and Average Pooling are the two different types of pooling. The maximum value from the area of the image that the Kernel has covered is returned by Max Pooling. The average of all the values from the area of the image covered by the Kernel is what is returned by average pooling, on the other hand.

Additionally, Max Pooling functions as a noise suppressant. It also does de-noising and dimensionality reduction in addition to completely discarding the noisy activations. Average Pooling, on the other hand, merely carries out dimensionality reduction as a noise-suppression strategy. Therefore, we can conclude that Max Pooling outperforms Average Pooling significantly.

The i-th layer of a convolutional neural network is made up of the convolutional layer and the pooling layer. The number of these layers may be expanded to capture even more minute details, but doing so will require more computer power depending on how complex the images are.

Fully Connected Layer

A (typically) inexpensive method of learning non-linear combinations of the high-level characteristics represented by the output of the convolutional layer is to add a Fully-Connected layer. In that area, the Fully-Connected layer is now learning a function that may not be linear.

We will now flatten the input image into a column vector after converting it to a format that is appropriate for our multi-level perceptron. A feed-forward neural network receives the flattened output, and backpropagation is used for each training iteration. The model can categorise images using the Softmax Classification method across a number of epochs by identifying dominant and specific low-level features.

After going through the approach outlined above, we were able to successfully help the model comprehend the features. Next, we will flatten the output for classification purposes and feed it into a standard neural network.

There are numerous CNN architectures that may be used, and these architectures have been essential in creating the algorithms that power and will continue to power AI as a whole in the near future. Below is a list of a few of them:

- LeNet
- AlexNet
- VGGNet
- GoogLeNet
- ResNet
- ZFNet

Understanding Recurrent Neural Network

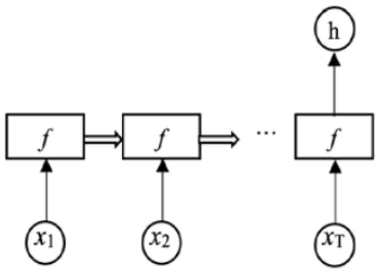
A recurrent neural network (RNN) is a type of artificial neural network which uses sequential data or time-series data. These deep learning algorithms are commonly used for ordinal or temporal problems, such as language translation, natural language processing (NLP), speech recognition, and image captioning; they are incorporated into popular applications such as Siri, voice search, and Google Translate. Like feed-forward and convolutional neural networks (CNNs), recurrent neural networks utilize training data to learn. They are distinguished by their "memory" as they take information from prior inputs to influence the current input and output. While traditional deep neural networks assume that inputs and outputs are independent of each other, the output of recurrent neural networks depends on the prior elements within the sequence. While future events would also help determine the output of a given sequence, unidirectional recurrent neural networks cannot account for these events in their predictions.

Types of recurrent neural networks

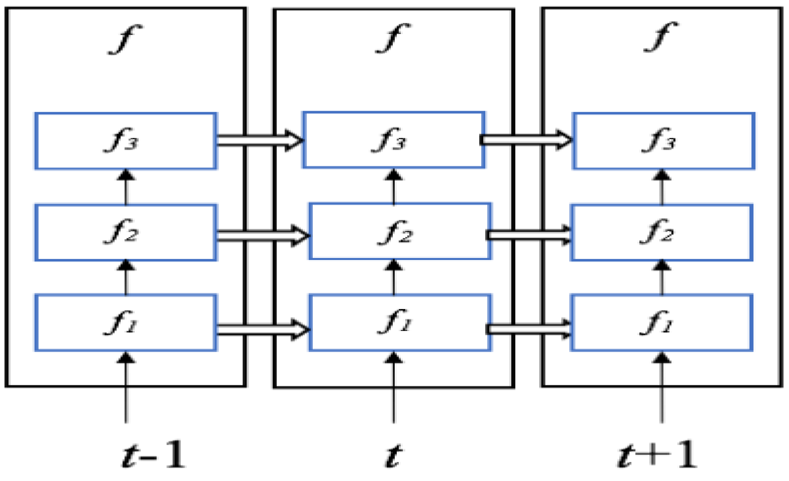
Feedforward networks map one input to one output, and while we've visualized recurrent neural networks in this way in the above diagrams, they do not have this constraint. Instead, their inputs and outputs can vary in length, and different types of RNNs are used for different use cases, such as music generation, sentiment classification, and machine translation. Different types of RNNs are usually expressed using the following diagrams:

Many-to-one RNNs

The most intuitive type of RNN is probably many-to-one. A many-to-one RNN can have input sequences with as many time steps as you want, but it only produces one output after going through the entire sequence. The following diagram depicts the general structure of a many-to-one RNN:



Many-to-one RNN

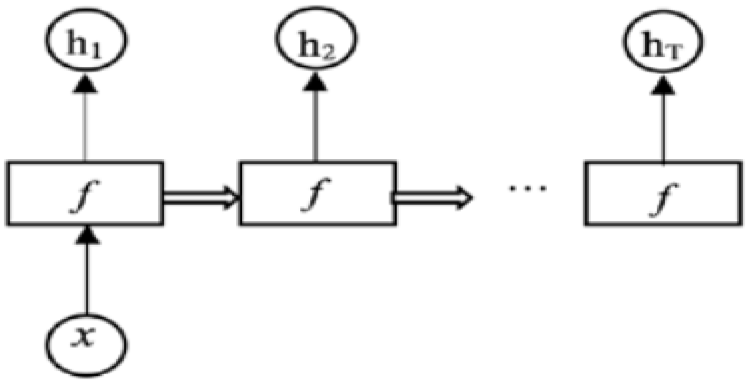


In the first diagram, f represents one or more recurrent hidden layers, where an individual layer takes in its own output from the previous time step. The second diagram shows an example of three hidden layers stacking up.

Many-to-one RNNs are widely used for classifying sequential data. Sentiment analysis is a good example of this and is where the RNN reads the entire customer review, for instance, and assigns a sentiment score (positive, neutral, or negative sentiment). Similarly, we can also use RNNs of this kind in the topic classification of news articles. Identifying the genre of a song is another application as the model can read the entire audio stream. We can also use many-to-one RNNs to determine whether a patient is having a seizure based on an ECG trace.

One-to-many RNNs

One-to-many RNNs are the exact opposite of many-to-one RNNs. They take in only one input (not a sequence) and generate a sequence of outputs. A typical one-to-many RNN is presented in the following diagram:



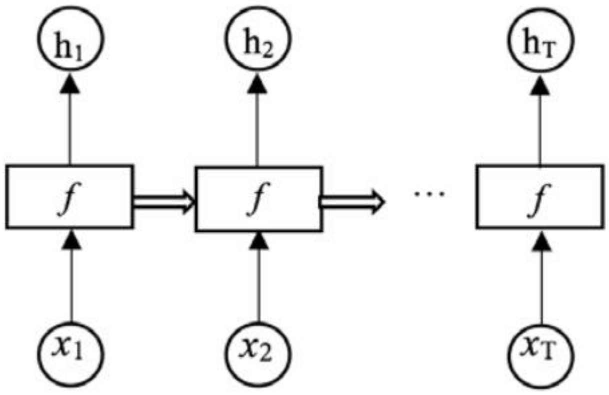
One-to-many RNN

Again, f represents one or more recurrent hidden layers.

Note that “one” here doesn’t mean that there is only one input feature. It means the input is from a one-time step, or it is time-independent. One-to-many RNNs are commonly used as sequence generators. For example, we can generate a piece of music given a starting note or/and a genre. Similarly, we can write a movie script like a professional screenwriter using one-to-many RNNs with a starting word we specify. Image captioning is another interesting application: the RNN takes in an image and outputs the description (a sentence of words) of the image.

Many-to-many (synced) RNNs

The third type of RNN, many-to-many (synced), allows each element in the input sequence to have an output. Let us look at how data flows in the following many-to-many (synced) RNN:



Many-to-many (synced) RNN

As you can see, each output is calculated based on its corresponding input and all the previous outputs. One common use case for this type of RNN is time series forecasting, where we want to perform rolling prediction at every time step based on the current and previously observed data. Here are some examples of time series forecasting where we can leverage synced many-to-many RNNs:

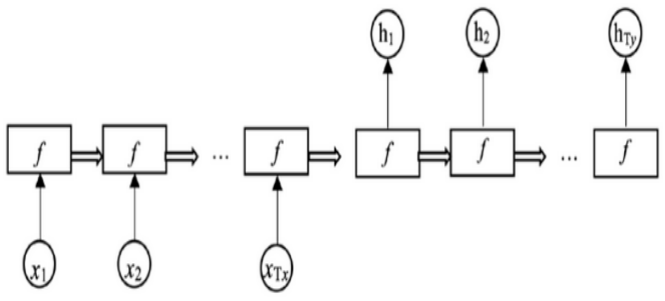
- Product sales each day for a store
- The daily closing price of a stock
- Power consumption of a factory each hour

They are also widely used in solving NLP problems, named entity recognition, and real-time speech recognition.

Many-to-many (unsynced) RNNs

Sometimes, we only want to generate the output sequence after we've processed the entire input sequence. This is the unsynced version of many-to-many RNN.

Refer to the following diagram for the general structure of a many-to-many (unsynced) RNN:



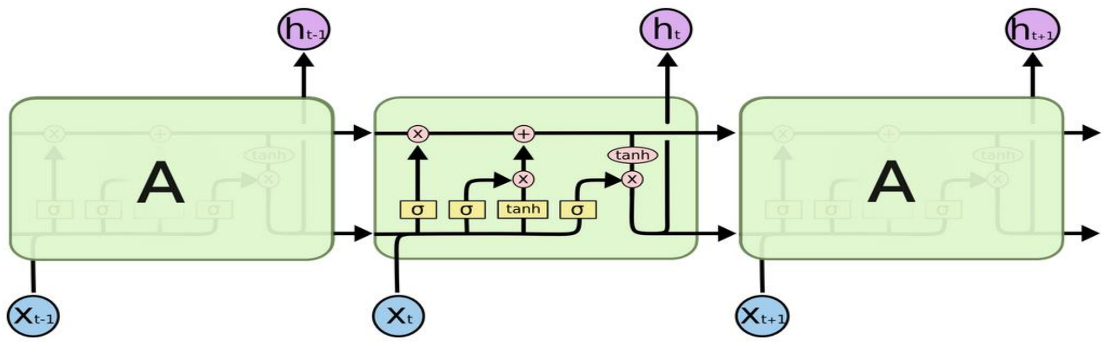
Many-to-many (unsynced) RNN

Note that the length of the output sequence (Ty in the preceding diagram) can be different from that of the input sequence (Tx in the preceding diagram). This provides us with some flexibility. This type of RNN is a go-to model for machine translation. In French-English translation, for example, the model first reads a complete sentence in French and then produces a translated sentence in English. Multi-step ahead forecasting is another popular example: sometimes, we are asked to predict sales for multiple days in the future when given data from the past month.

LSTM Networks

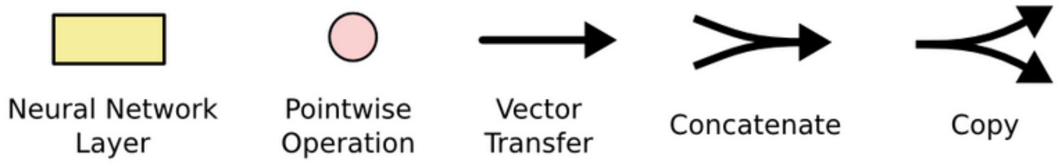
Long Short Term Memory networks – usually just called "LSTMs" – are a special kind of RNN, capable of learning long-term dependencies. They work tremendously well on a large variety of problems and are now widely used.

LSTMs are explicitly designed to avoid the long-term dependency problem. These long-term dependency problems are vanishing and exploding gradients problems. Remembering information for long periods of time is practically LSTM default behaviour, not something they struggle to learn! All recurrent neural networks have the form of a chain of repeating modules of a neural network. In standard RNNs, this repeating module will have a very simple structure, such as a single tanh layer. LSTMs also have this chain-like structure, but the repeating module has a different structure. Instead of having a single neural network layer, there are four, interacting in a very special way.



The repeating module in an LSTM contains four interacting layers.

Don’t worry about the details of what’s going on. We’ll walk through the LSTM diagram step by step later. For now, let’s just try to get comfortable with the notation we’ll be using.



In the above diagram, each line carries an entire vector, from the output of one node to the inputs of others. The pink circles represent point-wise operations, like vector addition, while the yellow boxes are learned neural network layers. Lines merging denote concatenation, while a line forking denotes its content being copied and the copies going to different locations.

Walkthrough the architecture: -

Now we are ready to look into the LSTM architecture step by step:

- We’ve got a new value xt and value from the previous node ht-1 coming in.
- These values are combined and go through the sigmoid activation function, where it is decided if the forget valve should be open, closed or open to some extent.
- The same values, or vectors of values, go in parallel through another layer operation “tanh”, where it is decided what value we’re going to pass to the memory pipeline, and also sigmoid layer operation, where it is decided if that value is going to be passed to the memory pipeline and to what extent.
- Then, we have a memory flowing through the top pipeline. If we have forget valve open and the memory valve closed then the memory will not change. Otherwise, if we have forget valve closed and the memory valve open, the memory will be updated completely.
- Finally, we’ve got xt and ht-1 combined to decide what part of the memory pipeline is going to become the output of this module.

That’s basically, what’s happening within the LSTM network.

BIG DATA

What is Big Data?
Data can be analyzed computationally to reveal patterns, trends, and associations to human or machine behavior and interaction. Big data is just a collection of very large datasets (PetaBytes) produced in huge volumes whose computation cannot be done using traditional computing techniques. Big data isn't a single technique or a tool, rather it has become a complete subject, which involves various tools, techniques, and frameworks.

Types in Big Data
Big Data involves data of heterogeneous nature, produced in huge volumes and at a very high rate. The data is generally classified into three categories.

Structured data: Structured data is data that has been organized into a formatted repository, typically a database. It concerns all data which can be stored in relational tables having a fixed schema. This type of data is the most processed in the development and simplest way to manage information. Eg. Relational data is stored in tables with rows and columns.

Semi-Structured data: Semi-structured data is a form of structured data that does not obey the tabular structure of data models associated with relational databases or other forms of data tables, nonetheless contains tags or other markers to separate semantic elements and enforce some sort of hierarchies. Eg. XML data

Unstructured data: Unstructured data is data that is not organized in a predefined manner and does not fit in a predefined data model, thus it is not a good fit for a mainstream relational database. Although there are alternative platforms for storing and managing and are used by organizations in a variety of business intelligence and analytics applications. Eg. Text, Media, logs, etc.

The five important V's of Big Data are:
1. Value – It refers to changing data into value, which allows businesses to generate revenue.
2. Velocity – Any data growing at an increasing rate is known as its variety. Social media is an important
factor contributing to the growth of data.
3. Variety – Data can be of different types such as texts, audios, videos, etc. which are known as variety.
4. Volume – It refers to the amount of any data that is growing at an exponential rate.
5. Veracity – It refers to the uncertainty found in the availability of data. It mainly arises due to the high
demand for data which results in inconsistency and incompleteness.

The three essential steps involved in Big Data are:
Data Ingestion
Data Storage
Data Processing
Data Ingestion is the first step of Big Data Solutions. This step refers to the extraction of data from different sources. Different sources data could include CRM, for instance, Salesforce;

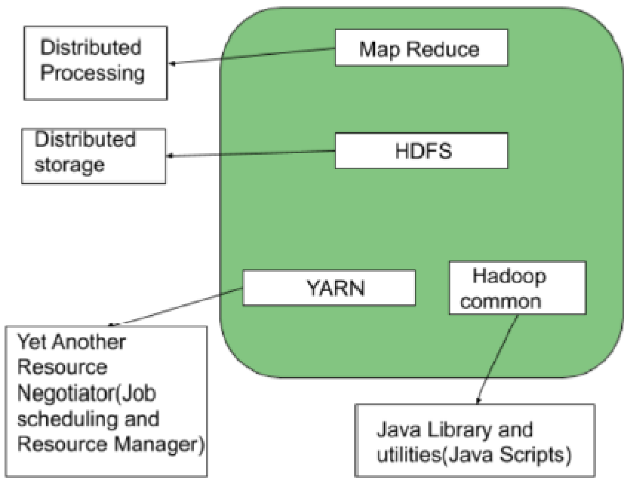
RDBMS such as MySQL, various Enterprise Resource Planning Systems such as SAP other with other log files, social media feeds, documents, papers, etc. All the data that is extracted is then stored in HDFS.

Data Storage is the next step in Big Data Solutions. In this step, the data is extracted from the first step is stored in HDFS or NoSQL database, also known as HBase. The HDFS storage is widely used for sequential access. On the contrary, HBase is used for random read or write access.

Data Processing is the final step of Big Data Solutions. In this step, with the help of different processing frameworks, the data is processed. Various processing frameworks used are Pig, MapReduce, Spark, etc.

Hadoop is an open source software framework for distributed storage and distributed processing of large data sets. Open source means it is freely available and even we can change its source code as per our requirements. Apache Hadoop makes it possible to run applications on the system with thousands of commodity hardware nodes. It's distributed file system has the provision of rapid data transfer rates among nodes. It also allows the system to continue operating in case of node failure.

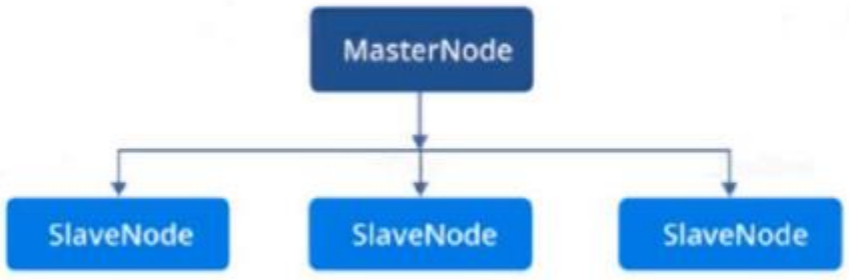
Main Components of Hadoop: -
Storage layer – HDFS
Batch processing engine – MapReduce
Resource Management Layer – YARN
HDFS - HDFS (Hadoop Distributed File System) is the storage unit of Hadoop. It is responsible for storing different kinds of data as blocks in a distributed environment. It follows master and slave topology.
Components of HDFS are NameNode and DataNode
MapReduce - For processing large data sets in parallel across a hadoop cluster, Hadoop MapReduce framework is used. Data analysis uses a two-step map and reduce process.
YARN - YARN (Yet Another Resource Negotiator) is the processing framework in Hadoop, which manages resources and provides an execution environment to the processes.
Main Components of YARN are Node Manager and Resource Manager.


Distributed Processing
Map Reduce
Distributed storage
HDFS
YARN
Hadoop common
Yet Another Resource Negotiator(Job scheduling and Resource Manager)
Java Library and utilities(Java Scripts)

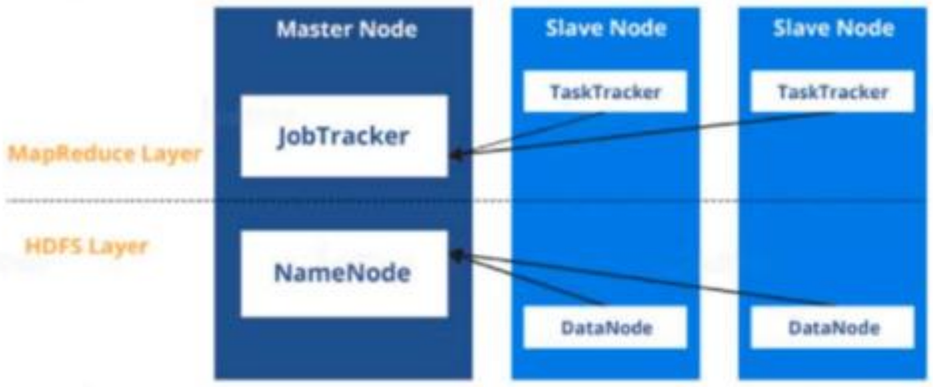
Why do we need Hadoop?
Storage – Since data is very large, so storing such huge amount of data is very difficult.

Security – Since the data is huge in size, keeping it secure is another challenge.
Analytics – In Big Data, most of the time we are unaware of the kind of data we are dealing with. So analyzing that data is even more difficult.
Data Quality – In the case of Big Data, data is very messy, inconsistent and incomplete.
Discovery – Using a powerful algorithm to find patterns and insights are very difficult.

Hadoop Architecture: -
Hadoop follows a master-slave architecture for storing data and data processing.



A cluster in the Hadoop ecosystem consists of one master node and numerous slave nodes. The components included in the master-node of the architecture are Job-tracker and Name-node, whereas the components of the slave-nodes are Task-tracker and data-node.



In the HDFS Layer:
Name-node: Name-node essentially performs a supervisory role that monitors and instructs the Data-nodes to perform the computations. Name-node contains the meta-data for all the files and by extension all the file segments in data blocks. It contains a table of information about the file namespace, Ids of the blocks, and the data nodes possessing them for each file inside the cluster.
Secondary Name-node: Secondary Name-node is a dedicated node to ensure availability in case of a primary name-node failover.
Data-node: Data-node is a component of the slave nodes which is used to store the processes and the data blocks in the HDFS Layer.

In The MapReduce Layer:
Job-tracker: The role of the Job-tracker in the master node is to accept jobs from the client, locate the files residing in data nodes, and hand out tasks to various data nodes for

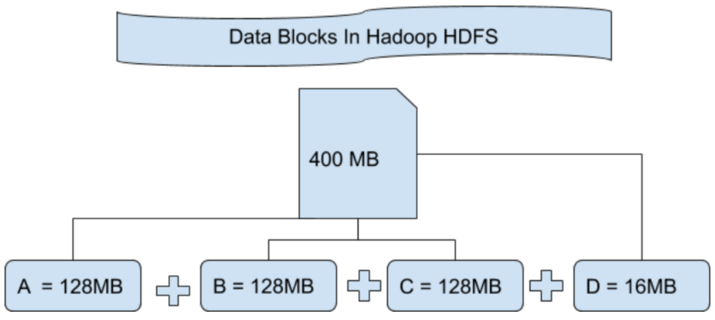
computation. The job is broken down into tasks, and the tasks are carried out in multiple data nodes in order to obtain the final results.
Task-tracker: Every Data-node designated to perform some task would be equipped with a Task-tracker. The role of Task-Tracker is to oversee the actual processes(tasks) being computed on the data nodes, by accepting the tasks from the Job-tracker and relaying the status of the tasks to the Job-tracker periodically.
Cluster: Cluster is the entire collection of machines such as Data-Nodes, Name-Nodes, Secondary Name-Nodes, etc

The HDFS Layer
HDFS stands for Hadoop Distributed File System. All the data in the Hadoop cluster is managed by this file system. As Hadoop is designed to enforce distributed storage, HDFS supports the storage of huge datasets across multiple machines. HDFS is more than just a regular file system, it has special features to make parallel processing reliable and effective.

Some key features in HDFS
It is designed with distributed storage and parallel processing in mind, which makes it both computationally and hardware cost-wise effective.
It is scalable as the data is distributed in data nodes across the cluster, and these data nodes can be scaled depending on the requirement.
It has monitoring enabled to track the status of the tasks being performed on the nodes inside the cluster.
It is designed keeping the Fault-tolerance and High-Availability in mind.
It provides data integrity, authentication, and authorization.
It provides effective use of bandwidth utilization as the programmed logic is moved to the data, and not the other way around. Transferring logic over a network is much more effective than transferring the data in huge volumes during computation.

Distributed Storage in HDFS:
All the data which includes all the large user datasets and files are stored in the HDFS. The files are broken down into small chunks in a way that they can be stored on numerous data nodes across the cluster. These chunks or segments of files are called Blocks.

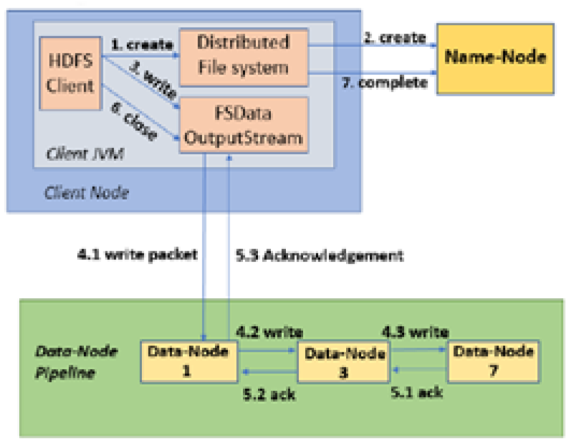
Replication in HDFS
HDFS has the reputation of being one of the most reliable file systems in the world. One of the reasons for this is replication. Every block is replicated by a factor of 3 by default, meaning that each block is copied to 3 different data nodes. The reason for this redundancy is to ensure fault tolerance and high availability. If a data node goes down due to some failure, we can ensure that we have at least 2 more copies of the same block which can be used when required. The figure below shows an instance where the blocks of files are replicated and stored in the data nodes.

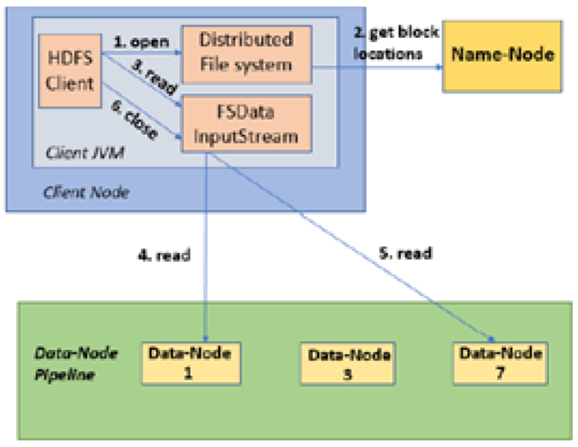


a. HDFS Data Write Pipeline Workflow

Now let’s understand complete end to end HDFS data write pipeline. As shown in the above figure the data write operation in HDFS is distributed, client copies the data distributedly on datanodes, the steps by step explanation of data write operation is:

i) The HDFS client sends a create request on DistributedFileSystem APIs.
ii) DistributedFileSystem makes an RPC call to the namenode to create a new file in the file system’s namespace. The namenode performs various checks to make sure that the file doesn’t already exist and that the client has the permissions to create the file. When these checks pass, then only the namenode makes a record of the new file; otherwise, file creation fails and the client is thrown an IOException.
iii) The DistributedFileSystem returns a FSDataOutputStream for the client to start writing data to. As the client writes data, DFSOutputStream splits it into packets, which it writes to an internal queue, called the data queue. The data queue is consumed by the DataStreamer, whichI is responsible for asking the namenode to allocate new blocks by picking a list of suitable datanodes to store the replicas.
iv) The list of datanodes form a pipeline, and here we’ll assume the replication level is three, so there are three nodes in the pipeline. The DataStreamer streams the packets to the first datanode in the pipeline, which stores the packet and forwards it to the second datanode in the pipeline. Similarly, the second datanode stores the packet and forwards it to the third (and last) datanode in the pipeline.
v) DFSOutputStream also maintains an internal queue of packets that are waiting to be acknowledged by datanodes, called the ack queue. A packet is removed from the ack queue only when it has been acknowledged by the datanodes in the pipeline. Datanode sends the acknowledgment once required replicas are created (3 by default). Similarly, all the blocks are stored and replicated on the different datanodes, the data blocks are copied in parallel.
vi) When the client has finished writing data, it calls close() on the stream.
vii) This action flushes all the remaining packets to the datanode pipeline and waits for acknowledgments before contacting the namenode to signal that the file is complete. The namenode already knows which blocks the file is made up of, so it only has to wait for blocks to be minimally replicated before returning successfully.





a. HDFS File Read Workflow

Now let’s understand complete end to end HDFS data read operation. As shown in the above figure the data read operation in HDFS is distributed, the client reads the data parallelly from datanodes, the steps by step explanation of data read cycle is:

i) Client opens the file it wishes to read by calling **open()** on the *FileSystem* object, which for HDFS is an instance of *DistributedFileSystem*.
ii) *DistributedFileSystem* calls the namenode using RPC to determine the locations of the blocks for the first few blocks in the file. For each block, the namenode returns the addresses of the datanodes that have a copy of that block and datanode are sorted according to their proximity to the client.
iii) *DistributedFileSystem* returns a *FSDataInputStream* to the client for it to read data from. *FSDataInputStream*, thus, wraps the *DFSInputStream* which manages the datanode and namenode I/O. Client calls **read()** on the stream. DFSInputStream which has stored the datanode addresses then connects to the closest datanode for the first block in the file.
iv) Data is streamed from the datanode back to the client, as a result client can call **read()** repeatedly on the stream. When the block ends, DFSInputStream will close the connection to the datanode and then finds the best datanode for the next block.
v) If the *DFSInputStream* encounters an error while communicating with a datanode, it will try the next closest one for that block. It will also remember datanodes that have failed so that it doesn’t needlessly retry them for later blocks. The *DFSInputStream* also verifies checksums for the data transferred to it from the datanode. If it finds a corrupt block, it reports this to the namenode before the *DFSInputStream* attempts to read a replica of the block from another datanode.
vi) When the client has finished reading the data, it calls **close()** on the stream.

Rack awareness in HDFS
In a Hadoop cluster, there can be multiple racks. And data nodes are placed within these racks. The replication should be done keeping the network latency in mind.

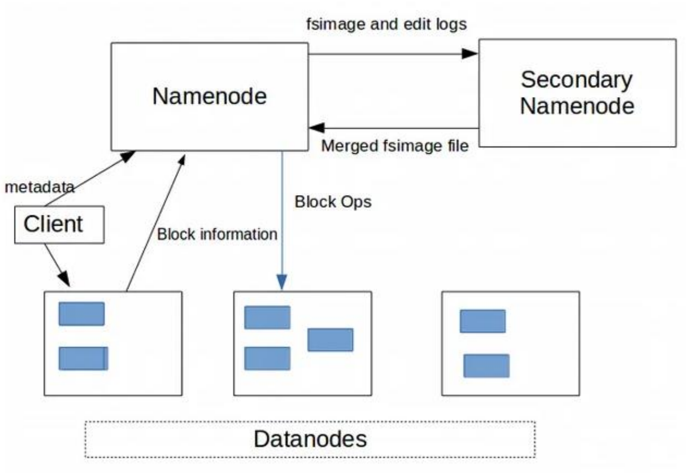
Explain what is heartbeat in HDFS?
Heartbeat is referred to a signal used between a data node and Name node, and between task tracker and job tracker, if the Name node or job tracker does not respond to the signal, then it is considered there is some issues with data node or task tracker.

High Availability in HDFS
High Availability when a datanode fails:-
When a datanode fails Jobtracker and namenode detect the failure. On the failed node all tasks are re-scheduled Namenode replicates the users data to another node. The user can request data-read by accessing the data from other data nodes containing a copy of data, with no downtime. Thus the cluster is highly available to the client even if any of the data nodes fail.

HA when a Name-node fails: -
So, if an active name node fails, then a passive name node becomes the active Name node, takes the responsibility, and serves the client request without interruption.
This allows for the fast failover to the new machine even if the machine crashes. Thus, data is available and accessible to the client even if the Name node itself goes down.

Secondary Name-node/ Checkpoint Name-node: -

Checkpoint Node keeps track of the latest checkpoint in a directory that has same structure as that of NameNode's directory. Checkpoint node creates checkpoints for the namespace at regular intervals by downloading the edits and fsimage file from the NameNode and merging it locally. The new image is then again updated back to the active NameNode.



Backup Node

The Backup node provides the same checkpointing functionality as the Checkpoint node, as well as maintaining an in-memory, up-to-date copy of the file system namespace that is always synchronized with the active NameNode state. Along with accepting a journal stream of file system edits from the NameNode and persisting this to disk, the Backup node also applies those edits into its own copy of the namespace in memory, thus creating a backup of the namespace.

The Backup node does not need to download fsimage and edits files from the active NameNode in order to create a checkpoint, as would be required with a Checkpoint node or Secondary NameNode, since it already has an up-to-date state of the namespace state in memory. The Backup node checkpoint process is more efficient as it only needs to save the namespace into the local fsimage file and reset edits. As the Backup node maintains a copy of the namespace in memory, its RAM requirements are the same as the NameNode. The NameNode supports one Backup node at a time. No Checkpoint nodes may be registered if a Backup node is in use. Using multiple Backup nodes concurrently will be supported in the future.

Map Reduce: -

Introduction to MapReduce
MapReduce is the programming model that is active in the Hadoop processing layer, it provides easy scalability and data computation. MapReduce works on top of the Hadoop HDFS layer. This programming model is designed to process large datasets by achieving parallelism. Parallelism in the MapReduce layer is achieved by breaking down the job into a set of tasks that are independent by nature.

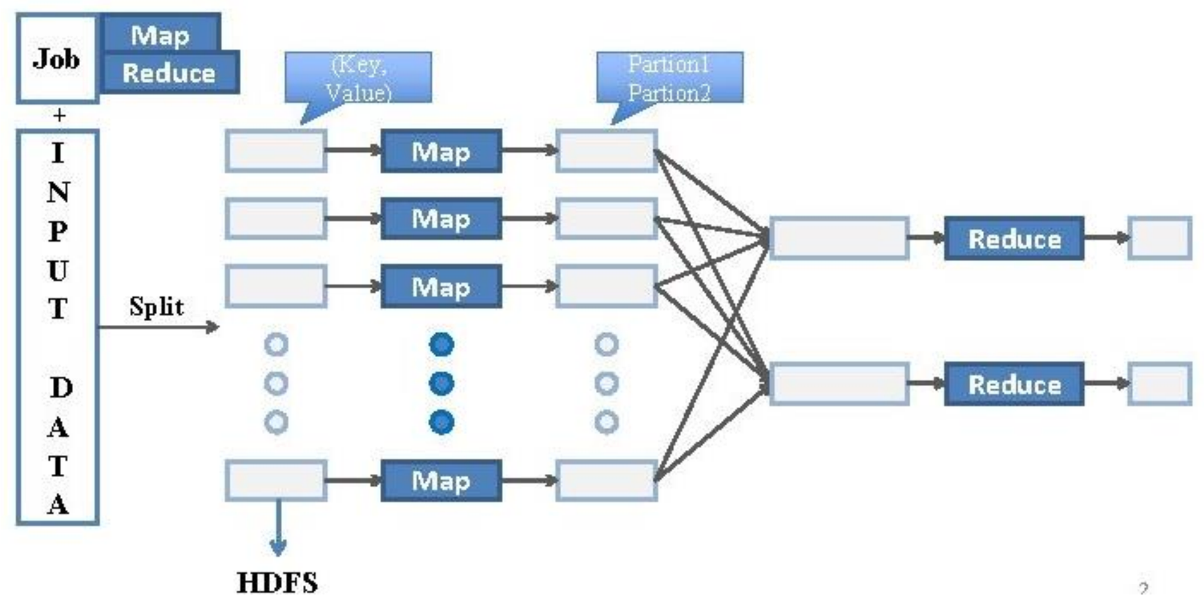
Map-Reduce Job

A Map-Reduce job submitted by a user goes through two layers of processing, namely Map and Reduce consisting and many phases. A client application has to submit the input data and the Map-reduce program along with its configuration. The data is then sliced into lists of inputs and sent to the map and reduce processes.

Task in MapReduce
A task in MapReduce is an execution of a Mapper or a Reducer on a slice of data on a node.
Task Attempt is a particular instance of an attempt to execute a task on a node.
It is possible that any machine can go down at any time. Eg., while processing an application task if any node goes down, the framework automatically reschedules the task to some other active node. This rescheduling of the task cannot happen infinitely, there is an upper limit for that. The default value for task attempts is 4. After a task (Mapper or reducer) has failed 4 times, the job is considered to be a failed job. For high-priority jobs, the value of task attempts can also be increased.

The map layer
The mapping process takes a key-value pair as input. The data might be in a structured or unstructured format, but the framework converts the incoming data into key and value.
Key is a reference to the input value.
Value is the dataset on which to operate.

Processing in the map layer
A function based on some business logic written by the client is applied to every value in the input.
The map layer then produces a new list of key-value pairs as an intermediate output. This output is stored on the local disk of the data nodes and is sent through various stages before reaching the reducer.
The reduce layer
Reduce takes intermediate Key / Value pairs as input and processes the output of the mapper.
Generally, in the reducer, we do aggregation or summation types of computations.
Input given to reducer is generated by Map (intermediate output)
Key-Value pairs provided to reduce are sorted by key
Processing in the Reduce layer
Here too, a function based on some business logic is applied to the key-value pairs to get the final output.
An Iterator supplies the values for a given key to the Reducer function.
Reducer generates a final list of key-value pairs.
An output of Reduce is called Final output.
It can be of a different type from the input pair.
The output of Reducer can be stored in HDFS or a different file system



Key-Value pair in MapReduce
In MapReduce, the map function processes key-value pairs and emit a certain number of key-value pairs, and then the Reduce function processes values grouped by the same key and emit another set of key-value pairs as the output. The output types of the Map should match the input types of the Reduce as shown below:
Map: (K1, V1) -> list (K2, V2)
Reduce: {(K2, list (V2 }) -> list (K3, V3)

Input format, Input split, and Record Reader
Input format
It defines how the files given as input are split up to be given to the map phase. TextInputFormat is the default input format in MapReduce. The role of Input Format is to generate Input Splits. TextInputFormat treats each line in the input file as a separate record and doesn't perform any parsing. This is especially useful for unformatted or line-based records such as log files.

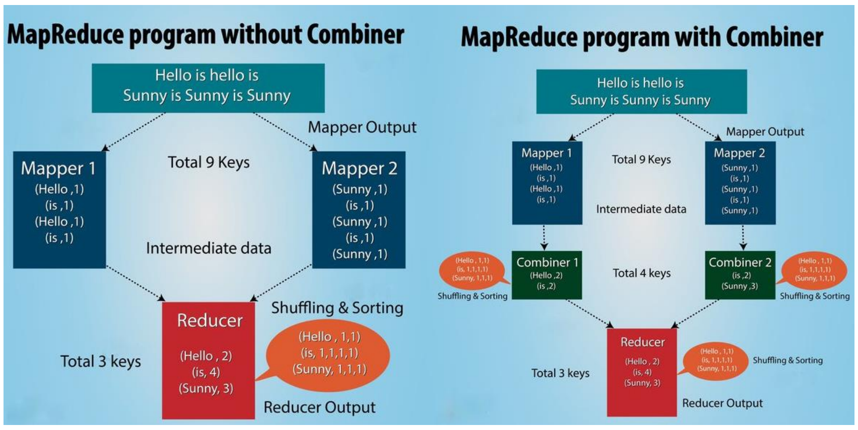
Input Split
InputSplit in Hadoop MapReduce is the logical representation of data. This describes a unit of work that contains a single mapper task inside a MapReduce program.
In Hadoop, InputSplit represents the data that is processed by an individual Mapper. Each split is divided into records. Hence, the mapper processes each record (that is a key-value pair).
The important thing to notice is that Inputsplit does not contain the input data; it is just a reference to the data. By default, the split size is approximately equal to the block size in HDFS. InputSplit is a logical chunk of data i.e. it just has the information about blocks addresses or locations.

Record Reader
Input Format computes splits for each file and then sends them to the Job-Tracker (in Name-node), which uses their storage locations to schedule map tasks to process them on the Task-Trackers (in Data-nodes). Map task then passes the split to the task tracker to obtain a RecordReader for that split. The RecordReader loads data from its source and converts it into key-value pairs suitable for reading by the mapper (map function). It communicates with the input split until the file reading is not completed.

The Map Phase
Mapper
Mapper takes the set of records as key-value pairs provided by the RecordReader and generates intermediate key-value pairs using the business logic provided by the client in the form of a map function.
The intermediate key-value pairs are sent through the combining, partitioning, and shuffling-sorting phases before being written into the local disk. These intermediate outputs are then sent to reducers to generate the final output.

The total number of blocks of the input files handles the number of map tasks in a program.
Hence, No. of Mapper= {(total data size)/ (input split size)}
For example, if data size is 1 TB and InputSplit size is 100 MB then,
No. of Mapper= (1000*1000)/100= 10,000

Combiner
Combiner is also known as “Mini-Reducer” that summarizes the Mapper output record with the same Key before passing to the Reducer.
On huge datasets when we run MapReduce job, large chunks of intermediate data are generated by the Mapper, and this intermediate data is passed on the Reducer for further processing, which would lead to enormous network congestion. MapReduce framework employs a function known as Hadoop Combiner that plays a vital role in decreasing network congestion.


MapReduce program without Combiner
Hello is hello is Sunny is Sunny is Sunny
Mapper Output
Total 9 Keys
Mapper 1 (Hello ,1) (is ,1) (Hello ,1) (is ,1)
Mapper 2 (Sunny ,1) (is ,1) (Sunny ,1) (is ,1) (Sunny ,1)
Intermediate data
Reducer (Hello , 2) (is, 4) (Sunny, 3)
Shuffling & Sorting
(Hello , 1,1) (is, 1,1,1,1) (Sunny, 1,1,1)
Total 3 keys
Reducer Output
MapReduce program with Combiner
Hello is hello is Sunny is Sunny is Sunny
Mapper Output
Mapper 1 (Hello ,1) (is ,1) (Hello ,1) (is ,1)
Total 9 Keys
Mapper 2 (Sunny ,1) (is ,1) (Sunny ,1) (is ,1) (Sunny ,1)
Intermediate data
(Hello , 1,1) (is, 1,1,1,1) (Sunny, 1,1,1)
Shuffling & Sorting
Combiner 1 (Hello ,2) (is ,2)
Total 4 keys
Combiner 2 (is ,2) (Sunny ,3)
(Hello , 1,1) (is, 1,1,1,1) (Sunny, 1,1,1)
Shuffling & Sorting
Total 3 keys
Reducer (Hello , 2) (is, 4) (Sunny, 3)
(Hello , 1,1) (is, 1,1,1,1) (Sunny, 1,1,1)
Shuffling & Sorting
Reducer Output

Hadoop Combiner reduces the time taken for data transfer between mapper and reducer.
It decreases the amount of data that needed to be processed by the reducer.
The use of Combiner enhances the overall performance of the reducer.

Partitioner
The Partitioner in MapReduce regulates the partitioning of the key of the intermediate output produced by mappers. By the means of a hash function, a key (or a subset of the key) is used to derive a partition. The total number of partitions is dependent on the number of reducer tasks.

Partitioner in Hadoop MapReduce redirects the mapper output to the reducer by determining the reducer responsible for that particular key.
The total number of running Partitioners is equal to the number of reducers. The data from a single partitioner is processed by a single reducer, and the partitioner exists only when there are multiple reducers.

The Reduce Phase
Shuffling and Sorting
The shuffle phase in Hadoop transfers the intermediate output from Mapper to a Reducer in MapReduce. Sort phase in MapReduce covers the merging and sorting of map outputs.
Shuffling
The process of transferring data from the mappers to reducers is known as shuffling i.e. the process by which the framework transfers the map output to the reducer as input

Sorting
The keys generated by the mapper functions are automatically sorted by MapReduce Framework, i.e. Before starting a reducer, all intermediate key-value pairs in MapReduce that are generated by the mappers get sorted by key and not by value.

Reducer
The reducer takes the output of the Mapper (intermediate key-value pairs) as input, processes each of them, and generates the output also as key-value pairs. The output from the reducers is the final output, which is stored in HDFS or a different file system.

Output Format
Output Format checks the Output-Specification of the job. It determines how RecordWriter implementation is used to write output to output files.
Record Writer
As mentioned, Reducer takes as input a set of an intermediate key-value pair produced by the mapper and runs a reducer function on them to generate output that is again zero or more key-value pairs.
RecordWriter writes these output key-value pairs generated after the Reducer phase to the output files.
Output Specification
The Output specification is described by the Output Format for a Map-Reduce job. On the basis of output specification MapReduce job checks that the output directory does not already exist

Why use MapReduce?
MapReduce is extremely fast compared to the traditional processing methods when it comes to huge datasets as it uses decentralized computation with parallel processing. It splits the whole job into sub-tasks and assigns them to multiple machines (data nodes) across the Hadoop cluster.

MapReduce supports Data-Locality: -
Since Hadoop works on huge volumes of data, it is not viable to move such volume over the network. Hence it had to come up with an innovative principle of moving algorithm to data rather than the other way around. This is called data locality. Data locality increases the overall throughput of the system.

Yarn Resource Manager: -

Issues in the old architecture of Hadoop

Scalability: Due to a single job tracker the cluster had a fixed number of nodes and therefore only a fixed number of tasks could be run on the cluster. In Hadoop 1.0 cluster only up to 10,000 nodes running 100,000 tasks were supported.

Availability: As the master daemon was a single point of failure, a failure meant killing all the jobs in the queue.

Resource Utilization: A fixed number of tasks meant a fixed number of Map and Reduce jobs which would give rise to resource utilization issues.
Hadoop 1.0 could only run MapReduce: This architecture had only one option in programming models to be run in the processing layer, which was MapReduce. It wasn't able to run non-MapReduce applications.

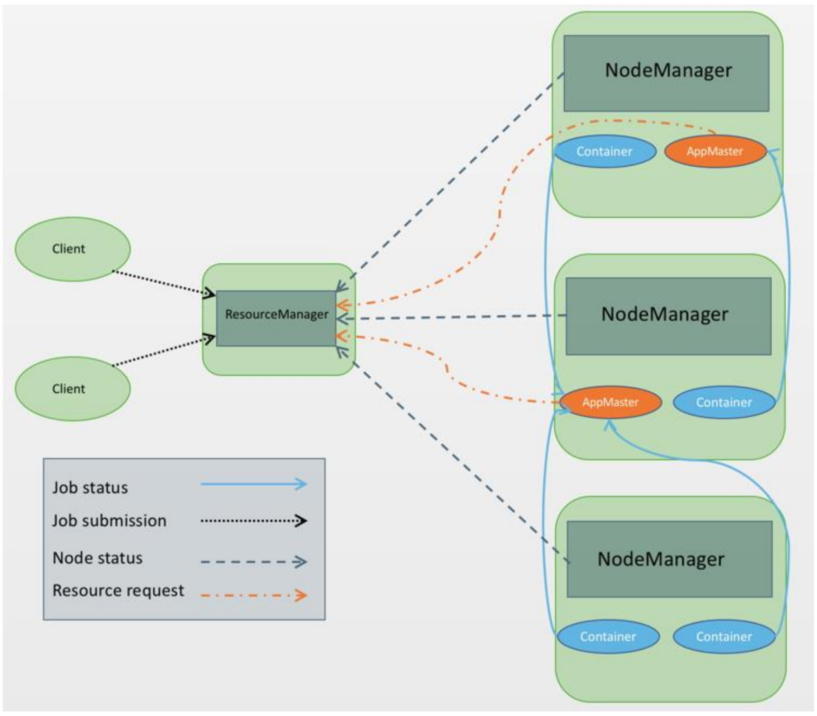
Hadoop 2.0 YARN
The introduction of YARN in Hadoop 2.0 solved the a for mentioned problems. YARN stands for Yet Another Resource Negotiator.

YARN is the Resource Management layer that operates on top of the Hadoop Distributed File System, and supports not only MapReduce but a variety of the processing frameworks, to name a few; Tez, HBASE, StoResource Manager, Giraph, Spark, etc.

What does YARN have to offer?
Scalability: In Hadoop 2.0 a cluster size of more than 10,000 nodes running over 100,000 tasks is possible.
Compatibility and Multi-tenancy: As mentioned, YARN supports a variety of frameworks in its processing layer which allows us to have applications like batch processing, streaming, graph processing, etc. This enables multi-tenancy as well.

High Availability: Having just one Resource Manager would mean a single point of failure in case the resource manager dies, therefore YARN adds a redundancy feature as an Active-Standby pair of Resource Managers.
Resource Utilization: YARN allows dynamic allocation of resources, to avoid utilization issues.

YARN architecture: -



The client would submit the job request to the Resource Manager in Hadoop 2.0 (YARN) as opposed to the Job Tracker in the older version. Thus, the Resource Manager is the Master Daemon in the YARN architecture.
Node Manager is the Slave daemon in the YARN architecture, which is similar to the Task-tracker in Hadoop MapReduce.

Components in YARN
Resource Manager
This is the master daemon in the YARN Architecture, its duty is to manage the assignment of resources which include CPU cores, Memory, Network I/O for the applications submitted by the clients.
The applications from the clients compete for resources and the Resource Manager resolves the resource allocation, which is why it is called a resource negotiator.
The two main components in the Resource Manager are Scheduler and Application Manager.

Scheduler
The scheduler is the component of the Resource Manager that does the actual allocation of the cluster resources to the applications. Its only job is to assign the resources to the competing jobs as a resource negotiator. It does not concern itself with scheduling, tracking, or monitoring of jobs.

Application Manager
The responsibility of the Application Manager is to manage the application masters that are running in the cluster. Its duty is to spawn the first application master, monitoring them, and also restarting them if there is a failure.

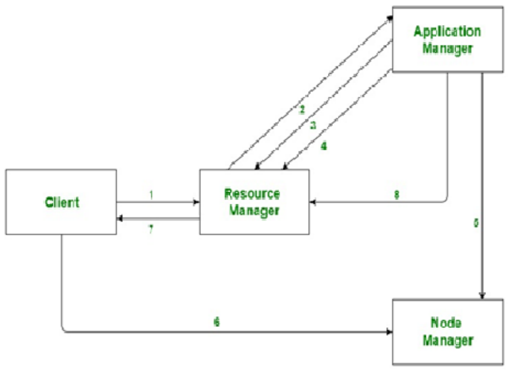
Node Manager
It is the slave daemon of Yarn. Every data node has its Node Manager, which manages the user process on that machine. Node Manager helps the Resource Manger keep its data updated, and Node Manager can also kill containers based on the instructions from the Resource Manager.

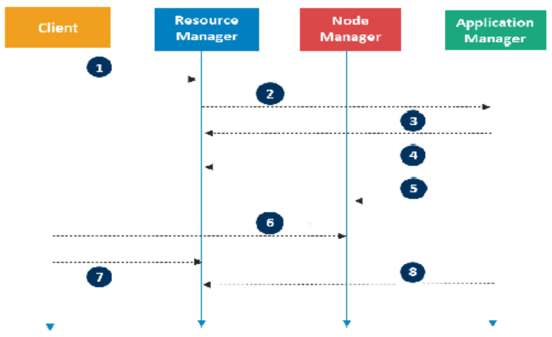
Application master
One application master runs per application. It negotiates resources from the resource manager and works with the node manager. It Manages the application life cycle.
The Application Manager acquires containers from the Scheduler before contacting the corresponding Node Manager to start the application's individual tasks.

Container
It is a collection of physical resources such as RAM, CPU cores, and disks on a single node.

Application workflow in Hadoop YARN:

1. Client submits an application
2. The Resource Manager allocates a container to start the Application Manager
3. The Application Manager registers itself with the Resource Manager
4. The Application Manager negotiates containers from the Resource Manager
5. The Application Manager notifies the Node Manager to launch containers
6. Application code is executed in the container
7. Client contacts Resource Manager/Application Manager to monitor application's status
8. Once the processing is complete, the Application Manager un-registers with the Resource Manager.





Application Workflow
1. The client submits an application
2. Resource Manager allocates a container to start Application Manager
3. Application Manager registers with Resource Manager
4. Application Manager asks containers from Resource Manager
5. Application Manager notifies Node Manager to launch containers
6. Application code is executed in the container
7. Client contacts Resource Manager/Application Manager to monitor application's status
8. Application Manager unregisters with Resource Manager

High Availability in YARN
Yarn Resource Manager High availability

The Resource Manager (master) is responsible for handling the resources in a cluster, and scheduling multiple applications (e.g., spark apps or MapReduce).
The High Availability feature adds redundancy in the Resource Manager of an Active/Standby Resource Manager pair to remove this otherwise single point of failure.

Resource Manager Restart
The Resource Manager has the authority to manage resources and schedule applications that are running on the YARN cluster. Hence, it is potentially a SPOF(single point of failure) in an Apache YARN cluster.
There are two types of restart for Resource Manager:
Non-work-preserving Resource Manager restart: In this type of restart, the Resource Manager persists application/attempt state in a pluggable state-store. Resource Manager will reload the same information from the state-store on the restart and will re-establish the previously running applications

Work-preserving Resource Manager restart: This type of Resource Manager restart focuses on reconstructing the running state of Resource Manager by combining the container status from Node Managers and container requests from Application Masters on restart. The key difference from Non-work-preserving Resource Manager restart is that already running apps will not be stopped after Resource Manager restarts, so applications will not lose their processed data because of Resource Manager/master outage.

Scheduler Plug-ins

Plug-in Schedulers such as Fair Scheduler and the Capacity Scheduler are widely used in YARN applications.

FIFO Scheduler

FIFO Scheduler runs the applications in submission order by placing them in a queue. Application submitted first, gets resources first and on completion, the scheduler serves the next application in the queue.

Capacity Scheduler

The Capacity Scheduler allows sharing of a Hadoop cluster between organizations. Each organization is set up with a dedicated queue that is configured to use a given fraction of the cluster capacity. Queues may be further divided in a hierarchical fashion, and it follows FIFO scheduling for a single queue. Capacity Scheduler may allocate the spare resources to jobs in the queue, even if the queue’s capacity is exceeded.

Fair Share Scheduler

Fair Scheduler attempts to allocate resources so that all running applications get the same share of resources. It enforces dynamic allocation of resources among the competing queues, with no need for prior capacity. It also allows for a priority mechanism within the queues. When a high-priority job arises in the same queue, the task is processed in parallel by replacing some portion from the already dedicated slots.