

Now Let's Start Our Journey

First Let's Understand What Is.....



Data

Data Is Plural Word and Its Singular Form is Datum.

Datum Is a Latin Word Which Means Something Given .

Data Is a Collection Of - **Text , Number , dates , Symbol , Images**



Types Of Data

1. Structured Data

Arranged In Row And Columns
Easy To Locate
Example : Excel

	A	B	C	D	E
1	Name	Capital	Continent	Area	Population
2	Argentina	Buenos Aires	South Amer	2777815	32300003
3	Bolivia	La Paz	South Amer	1098575	7300000
4	Brazil	Brasilia	South Amer	8511196	150400000
5	Canada	Ottawa	North Amer	9976147	26500000
6	Chile	Santiago	South Amer	756943	13200000
7	Colombia	Bagota	South Amer	1138907	33000000
8	Cuba	Havana	North Amer	114524	10600000
9	Ecuador	Quito	South Amer	455502	10600000
10	El Salvador	San Salvador	North Amer	20865	5300000
11	Guyana	Georgetown	South Amer	214969	800000
12	Jamaica	Kingston	North Amer	11424	2500000
13	Mexico	Mexico City	North Amer	1967180	88600000
14	Nicaragua	Managua	North Amer	139000	3900000
15	Paraguay	Asuncion	South Amer	406576	4660000
16	Peru	Lima	South Amer	1285215	21600000
17	United States of America	Washington	North Amer	9363130	249200000
18	Uruguay	Montevideo	South Amer	176140	3002000
19	Venezuela	Caracas	South Amer	912047	19700000

2. Unstructured Data

Not in Row and Columns
Difficult To Locate
Example : Images , Videos



Python

Python

Modules:

- 1. Fundamentals of Python**
- 2. Variable and Data types**
- 3. Operators**
- 4. Collections**
- 5. Conditional Statements**
- 6. Looping Statements**
- 7. Control Statements**
- 8. Functions**
- 9. Scope Of Variables**
- 10. Input-Output**
- 11. Files and Exceptions Handling**
- 12. OOPS Concepts**

Introduction of Python

- Python is an interpreted, object-oriented, high-level programming language with dynamic semantics.
- Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development
- Python supports modules and packages, which encourages program modularity and code reuse.
- The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

- Python supports modules and packages, which encourages program modularity and code reuse.
- The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

Why Python ?

Designed to be easy to learn and master

- Clean, clear syntax
- Very few keywords

Highly portable

- Runs almost anywhere - high end servers and workstations, down to windows CE
- Uses machine independent byte-code

Extensible

- Designed to be extensible using C/C++,
- allowing access to many external libraries

Features of Python

Clean syntax plus high-level data types

- Leads to fast coding (First language in many universities abroad!)

Uses white-space to delimit blocks

- Humans generally do, so why not the language?
- Try it, you will end up liking it

Uses white-space to delimit blocks

- Variables do not need declaration
- Although not a type-less language

Features of Python

- Reduced development time Code is 2-10x shorter than C, C++, Java
- Improved program maintenance Code is extremely readable
- Less training Language is very easy to learn

Programming Style

- Python programs/modules are written as text files with traditionally a .py extension.
- Each Python module has its own discrete namespace.
- Name space within a Python module is a global one.
- Python modules and programs are differentiated only by the way they are called.

Programming Style

- py files executed directly are programs (often referred to as scripts)
- .py files referenced via the import statement are modules.
- Thus, the same .py file can be a program/script, or a module.

Installation



Anaconda Installation

Individual Edition

[Click Here](#)



Individual Edition

Your data science toolkit

With over 20 million users worldwide, the open-source Individual Edition (Distribution) is the easiest way to perform Python/R data science and machine learning on a single machine. Developed for solo practitioners, it is the toolkit that equips you to work with thousands of open-source packages and libraries.

Download

print() function

- The print function in Python is a function that outputs to your console window whatever you say you want to print out.
- At first blush, it might appear that the print function is rather useless for programming, but it is actually one of the most widely used functions in all of python. The reason for this is that it makes for a great debugging tool.

Refer this example:

1.1.1 Print sample

- [Click here](#)

1.1.2 single quotation and double quotation

- [Click here](#)

Escape Sequences

Escape Sequence	use
\'	Single quote
\"	Double quote
\\\	backslash
\n	New line
\t	tab
\b	backspace

1.1.3 Escape sequence:

[Click here](#)

end= “ ”

- The end=' ' is just to say that you want a space after the end of the statement instead of a new line character.

Refer This Example :

- 1.1.4 end practical: [click here](#)

sep= “ “

- The separator between the arguments to print() function in Python is space by default (softspace feature) , which can be modified and can be made to any character, integer or string as per our choice.
- The ‘sep’ parameter is used to achieve the same, it is found only in python 3.x or later. It is also used for formatting the output strings.

Refer This Example :

- 1.1.5 sep practical:[click here](#)

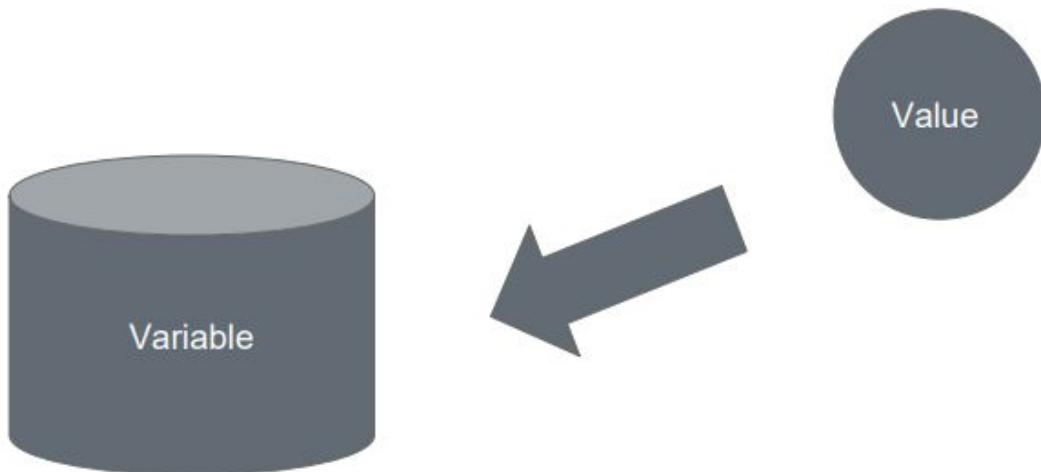
Comments

- A comment is a programmer-readable explanation or annotation in the source code of a computer program. They are added with the purpose of making the source code easier for humans to understand, and are generally ignored by compilers and interpreters.
- Types of comments :
 - Single line comment Indicate by #
 - Document comment “”” statements “””

Variable and Data Types

Variables

Variable : A name which can store a value



Variables

- Unlike other programming languages, Python has no command for declaring a variable.
- A variable is created the moment you first assign a value to it.

E.g. number=20

age = 21

Note : Variables do not need to be declared with any particular type

Variable Declaration Rules :

- A variable can have a short name (like a and b) or a more descriptive name (age,username,product_price).
- A variable name must start with a letter or the underscore character
- A variable name cannot start with a number

10name = “python”

- A variable name can only contain alphanumeric characters and underscores (A-z, 0-9, and _)

@name=“Python”

Variable Declaration Rules :

- Variable names are case-sensitive (age, Age and AGE are three different variables)

```
NAME="python" print(name)
```

Error : NameError: name 'name' is not defined

Refer This Examples

: 1.1.6 Variable sample

- [Click here](#)

<https://github.com/TopsCode/Python/blob/master/Module1/1.1%20Programming%20Style/1.1.6%20Variable.py>

1.1.7 Sum of two numbers

- [Click here](#)

[https://github.com/TopsCode/Python/blob/master/Module1/1.1%20Programming%20Style/1.1.7%20sum%20of%20two%20numbers\(variable\).py](https://github.com/TopsCode/Python/blob/master/Module1/1.1%20Programming%20Style/1.1.7%20sum%20of%20two%20numbers(variable).py)

1.1.8 Swaping of two numbers

- [Click here](#)

Data Types

1. Immutable:

Whose Value can't be modified.

Ex: int, float, string, tuple

2. Mutable:

Whose Value can be modified.

Ex: List, Set, Dictionary

Integer:

Integers are positive or negative whole numbers with no decimal point.

Float

Float represent real numbers and are written with a decimal point.

Strings

Python does not have a character data type, a single character is simply a **string** with a length of 1. Square brackets can be used to access elements of the **string**.

Operators in Python

- To perform specific operations we need to use some symbols and that symbols are operator

Example :

A + B

Here, + is a operator

A and B is operand And

A+B is expression

Arithmetic Operators

Operator	Name
+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Modulus
**	Exponentiation
//	Floor division

Assignment Operators

Operator	Example
=	a=10
+=	a+=10
=	a=10
/=	a/=10
%=	a%=10
=	a=10
//=	a//=10

Logical Operators

Operator	name
and	And operator
or	Or operator
not	Not operator

Comparison Operators

Operator	Name
<code>==</code>	Equal
<code>!=</code>	Not equal
<code>></code>	Greater than
<code><</code>	Less than
<code>>=</code>	Greater than or equal to
<code><=</code>	Less than or equal to

Identity Operators

Operator	Example
is	A is B
is not	A is not B

Membership Operators

Operator	Example
in	A in student_list
Not in	A not in student_list

Collections

- List

Operations , Functions and methods

- Tuple

Operations , Functions and methods

- Dictionaries

Operations , Functions and methods

- Set

Operations , Functions and methods

List []

1. Introduction
2. Accessing list
3. Operations
4. Working with lists
5. Function and Methods

1. Introduction

- Python knows a number of compound data types, used to group together other values.
- The most versatile is the list, which can be written as a list of comma-separated values (items) between square brackets.
- Lists might contain items of different types, but usually the items all have the same type.

2. Accessing List

- Accessing List

Like strings (and all other built-in sequence type), lists can be indexed and Sliced

```
fruits = ['apple', 'orange', 'banana', 'grapes']
```

Example: fruits[0]

Example :fruits[-3:-1]

Unlike strings, which are immutable, lists are a mutable type, i.e. it is possible to change their content.

3. Operations

- “in” operator :- This operator is used to check if an element is present in the list or not.

Returns true if element is present in list else returns false.

- “not in” operator :- This operator is used to check if an element is not present in the list or not.

Returns true if element is not present in list else returns false.

4. Working:

2.1.1 List as Queue

- [Click here](#)

2.1.2 List Demo

- [Click Here](#)

2.1.3 List operations

- [Click here](#)

2.1.4 List pattern

- [Click here](#)

5. Functions and Methods

Name	Description
len(list)	Gives the total length of the list.
max(list)	Returns item from the list with max value.
min(list)	Returns item from the list with min value.
list(seq)	Converts a tuple into list.

5. Functions and Methods

Methods	Description
list.append(x)	Add an item to the end of the list. Equivalent to a [len(a):]=[x].
list.extend(L)	Appends the contents of L to list
list.insert(I,x)	Insert an item at a given position. The first argument is the index of the element before which to insert, so a.insert(0,x) inserts at the front of the list, and a.insert(len(a),x) is equivalent to a.append(x).
list.count(obj)	Returns count of how many times obj occurs in list
list.index(obj)	Returns the lowest index in list that obj appears
List.pop(obj=1 ist[-1])	Removes and returns last object or obj from list.

5. Functions and Methods

Name	Description
list.reverse()	Reverses objects of list in place
list.sort([fun])	Sorts objects of list, use compare fun if given
list.remove(obj)	Removes object obj from list

Tuple

1. Introduction
2. Accessing tuples
3. Operations
4. Working
5. Functions and Methods

1. Introduction

- A tuple is a sequence of immutable Python objects.
- Tuples are sequences, just like lists.
- The differences between tuples and lists are, the tuples cannot be changed unlike lists and tuples use parentheses, whereas lists use square brackets.
- Eg fruits=("Mango","Banana","Oranges",23,44)
- Eg numbers=(11,22,33,44)
- Eg fruits="Mango","Banana","Oranges"

Introduction

- Unlike lists, tuples are immutable.

This means that elements of a tuple cannot be changed once it has been assigned.

- But if the element is itself a mutable data type like list, its nested items can be changed.
- We can also assign a tuple to different values (reassignment).
- Also We cannot delete or remove items from a tuple.
- But deleting the tuple entirely is possible using the keyword del.

2. Accessing tuples

- There are various ways in which we can access the elements of a tuple.
- We can use the index operator [] to access an item in a tuple.
- Index starts from 0. The index must be an integer.
- Python allows negative indexing for its sequences.
- The index of -1 refers to the last item, -2 to the second last item and so on.
- We can access a range of items in a tuple by using the slicing operator (colon).
- Eg. fruits [2:]

3. Operations

- With Tuples we can do concatenation ,repetition,iterations etc

4. Working

2.2.1 add item tuple

[Click here](#)

2.2.2 convert list tuple

[Click here](#)

2.2.3 convert tuple string

[Click here](#)

2.2.4 create tuple with numbers

[Click here](#)

5. Functions

Name	Description
len(tuple)	Gives the total length of the tuple.
max(tuple)	Returns item from the tuple with max value.
min(tuple)	Returns item from the tuple with min value.
tuple(seq)	Converts a seq into tuple.

5. Functions

Method	Description
count(obj)	Returns count of how many times obj occurs in tuple
index(obj)	Returns the lowest index in tuple that obj appears

Refer this example :

2.2.5 create tuple

[Click here](#)

2.2.6 find repeat item tuple

[Click here](#)

2.2.7 slice tuple

[Click here](#)

Set

A set is a Sequence of unique elements.

To create a set:

```
set1=set((1,2,3,4,3,4,5,6,5,5,4))
```

```
print(set1)
```

Output: {1, 2, 3, 4, 5, 6}

Dictionaries

1. Introduction
2. Accessing values in dictionaries
3. Working with dictionaries
4. Properties
5. Functions

Introduction

- Dictionaries are sometimes found in other languages as “associative memories” or “associative arrays”.
- Unlike sequences, which are indexed by a range of numbers, dictionaries are indexed by keys, which can be any immutable type; strings and numbers can always be keys.
- Tuples can be used as keys if they contain only strings, numbers, or tuples; if a tuple contains any mutable object either directly or indirectly, it cannot be used as a key.

1. Introduction

- You can't use lists as keys, since lists can be modified in place using index assignments, slice assignments, or methods like `append()` and `extend()`.
- The main operations on a dictionary are storing a value with some key and extracting the value given the key.
- Like lists they can be easily changed, can be shrunk and grown ad libitum at run time. They shrink and grow without the necessity of making copies. Dictionaries can be contained in lists and vice versa.

Introduction

- A list is an ordered sequence of objects, whereas dictionaries are unordered sets.
- But the main difference is that items in dictionaries are accessed via keys and not via their position.

2. Accessing Values

- To access dictionary elements, we can use the familiar square brackets along with the key to obtain its value.
- It is an error to extract a value using a non-existent key.
- We can also create a dictionary using the built-in class `dict()` (constructor).
- We can test if a key is in a dictionary or not using the keyword `in`.
- The membership test is for keys only, not for values.

3. Working

2.3.1 Dictionary example

[Click here](#)

2.3.2 Dictionary method demo

[Click here](#)

4. Properties

- Properties of Dictionaries
 - Dictionary values have no restrictions.
 - They can be any arbitrary Python object, either standard objects or user-defined objects.
 - However, same is not true for the keys.
- More than one entry per key not allowed.
 - Which means no duplicate key is allowed. ○ When duplicate keys encountered during assignment, the last assignment wins.

Properties

- Keys must be immutable.
 - Which means you can use strings, numbers or tuples as dictionary keys but something like ['key'] is not allowed.

5. Methods and Functions

Method	Description
dist.copy()	A dictionary can be copied with the method copy():.
dist.update()	It merges the keys and values of one dictionary into another, overwriting values of the same key.
dist.values()	It returns the list of dictionary dict's values.
dist.keys()	It returns the list of dictionary dict's keys.
dist.items()	It returns the list of dictionary dict's keys,values in tuple pairs.
dist.clear()	Removes all elements of dictionary dict.

Conditional Statements

Types of conditional Statements

- If .. Statement
- If.. else Statement
- If..Elif..else Statement
- Nested if Statement

If Statements

- It is similar to that of other languages.
- The if statement contains a logical expression using which data is compared and a decision is made based on the result of the comparison.
- Syntax :

```
if condition:  
    statements  
    :  
    ...
```

If .. else statement

- It is similar to that of other languages.
- It is frequently the case that you want one thing to happen when a condition is true, and something else to happen when it is false.
- For that we have the if else statement.
- Syntax :

if condition:

statements

else:

statement(s)

If..elif..else statement

- It is similar to that of other languages.
- The elif is short for else if. It allows us to check for multiple expressions.
- If the condition for if is False, it checks the condition of the next elif block and so on.
- If all the conditions are False, body of else is executed.
- Only one block among the several if...elif...else blocks is executed according to the condition.

If..elif..else statement

- Syntax :

If condition:

 statement

Elif condition:

 statement

Nested if....else statement

- There may be a situation when you want to check for another condition after a condition resolves to true.
- In such a situation, you can use the nested if construct.
- Syntax :

```
if condition: statements
```

```
    if condition: statements
```

```
        else:
```

```
            statement(s)
```

Refer this Example :

1.2.1 if statement

[Click here](#)

1.2.2 if else statement

[Click here](#)

1.2.3 elif statement

Click here

1.2.4 Nested if statement

[Click here](#)

Looping Statements

Loop Statement

- A loop statement allows us to execute a statement or group of statements multiple times.
- Python programming language provides following types of loops to handle looping requirements.
 - For Loop
 - While Loop

For Loops

- For loop has the ability to iterate over the items of any sequence, such as a list or a string.
- Syntax :

```
for iterating_var in sequence:
```

```
    statements(s)
```

- If a sequence contains an expression list, it is evaluated first.
- Then, the first item in the sequence is assigned to the iterating variable iterating_var.

- Next, the statements block is executed.
- Each item in the list is assigned to iterating_var, and the statement(s) block is executed until the entire sequence is exhausted

Refer this Example :

1.3.1 for loop

[Click here](#)

1.3.2 for loop with list

[Click here](#)

Nested Loops

Nested loop means a loop statement inside another loop statement.

Syntax :

```
for iterating_var in sequence:
```

```
    for iterating_var in sequence:
```

```
        statements(s)
```

```
    statements(s)
```

Refer this Example :

1.3.5 nested for loop: [Click here](#)

1.3.6 pattern 1: [Click here](#)

1.3.7 pattern 2: [Click here](#)

1.3.8 pattern 3: [Click here](#)

1.3.9 pattern 4: [Click here](#)

range() function

- To loop through a set of code a specified number of times, we can use the range() function,
- The range() function returns a sequence of numbers, starting from 0 by default, and increments by 1 (by default), and ends at a specified number.
- The range() function defaults to 0 as a starting value, however it is possible to specify the starting value by adding a parameter: range(2, 6), which means values from 2 to 6 (but not including 6):

Refer this Example :

1.3.3 for loop with range

[Click here](#)

1.3.4 for loop with decrement range

[Click here](#)

While Loop

- A while loop statement in Python programming language repeatedly executes a target statement as long as a given condition is true.
- Here, statement(s) may be a single statement or a block of statements. The condition may be any expression, and true is any non-zero value. The loop iterates while the condition is true.
- Syntax :

while expression:

 statement(s)

1.3.10 while loop: [Click here](#)

Control Statements

Control Statements

- Loop control statements change execution from its normal sequence.
- When execution leaves a scope, all automatic objects that were created in that scope are destroyed.
- Python supports the following control statements.
 1. Break
 2. Continue
 3. Pass

Break Statement

- It brings control out of the loop and transfers execution to the statement immediately following the loop.
- Syntax : break

Refer this Example :

1.4.1 break statement

[Click Here](#)

Continue Statements

- It continues with the next iteration of the loop.
- Syntax : continue

Refer this Example :

1.4.2 continue statement

[Click here](#)

Pass Statements

- The pass statement does nothing.
- It can be used when a statement is required syntactically but the program requires no action.
 - Syntax : pass
 - Refer this Example :

1.4.3 pass statement

[Click here](#)

FUNCTIONS

Function Definition

- A function is a block of organized, reusable code that is used to perform a single, related action.
- Functions provide better modularity for your application and a high degree of code reusing.

Types of Function

Functions can be of

Built-in functions	Functions that come built into the Python language itself are called built-in functions and are readily available to us. Eg: <code>input()</code> , <code>eval()</code> , <code>print()</code> etc...
User defined functions	Functions that we define ourselves to do certain specific task are referred as user-defined functions. Eg: <code>checkNoEvenOdd(20)</code>

Defining a Functions

- Defining a function
 - Python gives us many built-in functions like `print()`, etc. but we can also create our own functions.
 - A function in Python is defined by a `def` statement. The general syntax looks like this:

`def function-name(Parameter list):`

statements, i.e. the function body

`return [expression]`

Defining a Functions

- The keyword "def" introduces a function definition.
- It must be followed by the function name and the parenthesized list of formal parameters. The statements that form the body of the function start at the next line, and must be indented.
- The "return" statement returns with a value from a function."return" without an expression argument returns None.
- Falling off the end of a function also returns None

Calling a Functions

- Once function define, we can call it directly or in any other function also.
- Syntax :

functionname() or functionname(argument)

3.1.1 Create function

[Click here](#)

Function Arguments

- It is possible to define functions with a variable number of arguments.
- The function arguments can be
 - Default arguments values
 - Keyword arguments

Function Arguments

- Default arguments values
 - The most useful form is to specify a default value for one or more arguments.
 - This creates a function that can be called with fewer arguments than it is defined to allow.
 - Eg. def employeeDetails(name,gender='male',age=35)
 - This function can be called in several ways:
 - giving only the mandatory argument : employeeDetails("Ramesh")

Function Arguments

- giving one of the optional arguments: employeeDetails("Ramesh",'Female')
- or even giving all arguments : employeeDetails("Ramesh",'Female',31)
- Note : The default value is evaluated only once. This makes a difference when the default is a mutable object such as a list, dictionary, or instances of most classes.

Keyword Arguments

- Functions can also be called using keyword arguments of the form `kwarg=value`.
- For instance, the following function:
- `def parrot(voltage, state='a stiff', action='voom', type='Norwegian Blue'):`
- accepts one required argument (`voltage`) and three optional arguments (`state`, `action`, and `type`).

Function Arguments

```
def parrot(voltage, state='a stiff', action='voom', type='Norwegian Blue'):
```

parrot(1000)	<i>positional argument</i>
--------------	----------------------------

parrot(voltage=1000)	<i>keyword argument</i>
----------------------	-------------------------

parrot(voltage=1000000, action='VOOOOOM')	<i>keyword arguments</i>
--	--------------------------

parrot(action='VOOOOOM', voltage=1000000)	<i>keyword arguments</i>
--	--------------------------

parrot('a million', 'bereft of life', 'jump')	<i>positional arguments</i>
--	-----------------------------

parrot('a thousand', state='pushing up the daisies')	<i>positional, 1 keyword</i>
---	------------------------------

Function Arguments

- Arbitrary Argument Lists
 - These arguments will be wrapped up in a tuple . Before the variable number of arguments, zero or more normal arguments may occur.
 - Eg :

```
def write_multiple_items(file, separator, *args):  
    file.write(separator.join(args))
```
 - Normally, these variadic arguments will be last in the list of formal parameters, because they scoop up all remaining input arguments that are passed to the function.

Function Arguments

- Any formal parameters which occur after the *args parameter are ‘keyword-only’ arguments, meaning that they can only be used as keywords rather than positional arguments.

```
def concat(*args, sep="/"):
```

```
    return sep.join(args)
```

```
concat("earth", "mars", "venus") O/P->'earth/mars/venus '
```

```
concat("earth", "mars", "venus", sep=".") O/P- >'earth.mars.venus'
```

Refer this example :

3.1.2 Function with parameter: [click here](#)

3.1.3 Function with default value: [click here](#)

3.1.4 Function with return values: [click here](#)

3.1.5 tuple as parameter: [click here](#)

3.1.6 dict as parameter: [click here](#)

Lambda functions

- Lambda functions can have any number of arguments but only one expression.
- The expression is evaluated and returned.
- Lambda functions can be used wherever function objects are required.
- Python supports a style of programming called functional programming where you can pass functions to other functions to do stuff.

3.2.1 lambda function

[Click here](#)

Scope of Variables

Global variables

- Defining a variable on the module level makes it a global variable, you don't need to global keyword.
- The global keyword is needed only if you want to reassign the global variables in the function/method.
- Defining a variable on the module level makes it a global variable, you don't need to global keyword.
- The global keyword is needed only if you want to reassign the global variables in the function/method.

Local variables

- If a variable is assigned a value anywhere within the function's body, it's assumed to be a local unless explicitly declared as global.
- Local variables of functions can't be accessed from outside.

3.3.1 global variable: [click here](#)

3.3.2 local variable: [click here](#)

3.3.3 global keyword: [click here](#)

Modules

Introductions

- A module is a file containing Python definitions and statements.
- The file name is the module name with the suffix .py appended.
- Within a module, the module's name (as a string) is available as the value of the global variable `__name__`.

Importing Module

- Modules can import other modules.
- It is customary but not required to place all import statements at the beginning of a module (or script, for that matter).
- The imported module names are placed in the importing module's global symbol table.

Eg : import fibo

- Using the module name we can access functions.

fibo.fib(10)

fibo.fib2(10)

Importing Module

- There is a variant of the import statement that imports names from a module directly into the importing module's symbol table.

For example:

```
from fibo import fib, fib2
```

```
fib(500)
```

another way to import is

```
from fibo import *
```

4.1.1 module example: [click here](#)

Math Module

- This module is always available.
- It provides access to the mathematical functions defined by the C standard.
- These functions are divided into some categories like Number theoretic and representation functions, Power and logarithmic functions, Trigonometric functions, Angular conversion, Hyperbolic functions, Special functions.
- Constants
 - `math.pi` : The mathematical constant $\pi = 3.141592\dots$, to available precision.
 - `math.e` : The mathematical constant $e = 2.718281\dots$, to available precision.,

Math Module

- `math.inf` : A floating-point positive infinity. (For negative infinity, use `-math.inf`.)
Equivalent to the output of `float('inf')`.
- `math.nan` : A floating-point “not a number” (NaN) value. Equivalent to the output of `float('nan')`

Function	Description
<code>math.ceil(x)</code>	Return the ceiling of x , the smallest integer greater than or equal to x . If x is not a float, delegates to <code>x.__ceil__()</code> , which should return an Integral value.
<code>math.factorial(x)</code>	Return x factorial

Function	Description
math.floor(x)	Return the floor of x , the largest integer less than or equal to x . If x is not a float, delegates to $x.__floor__()$, which should return an Integral value.
math.trunc(x)	Return the Real value x truncated to an Integral (usually an integer).
math.pow(x, y)	Return x raised to the power y .
And so more...	

4.3.1 Math module

[Click here](#)

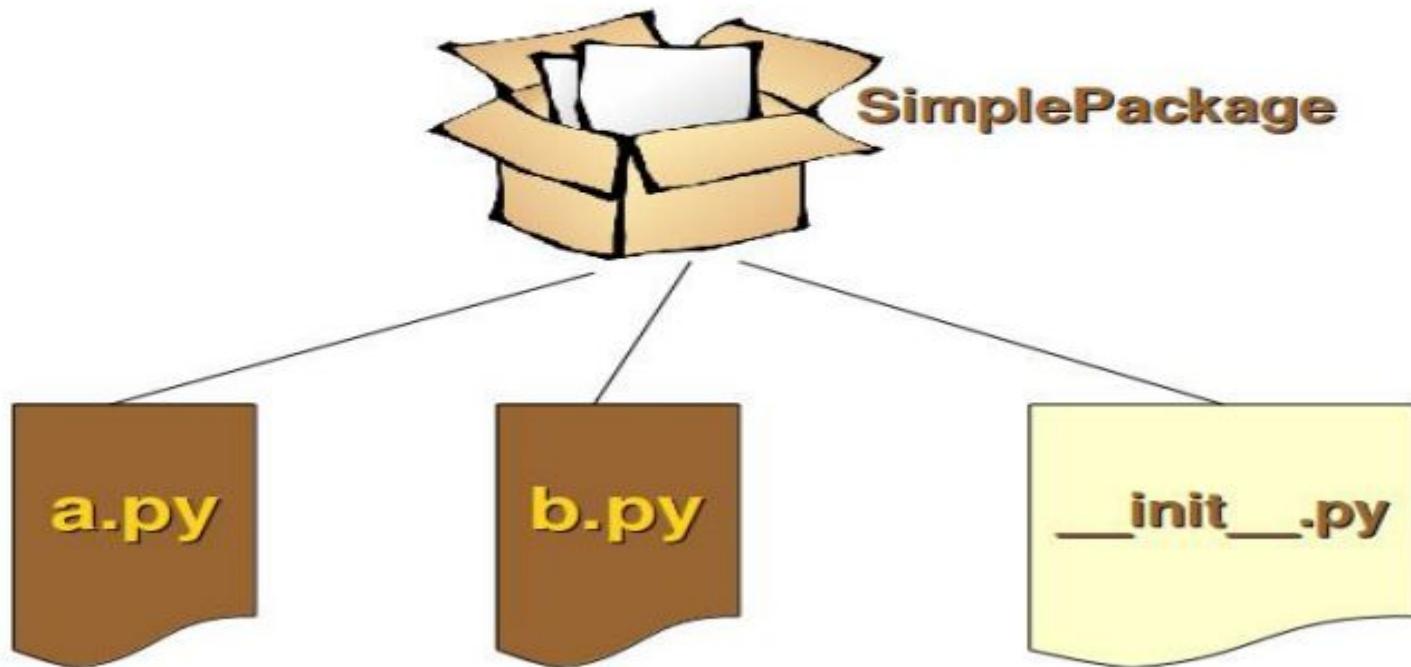
Packages

- Packages are a way of structuring Python's module namespace by using "dotted module names".
- For example, the module name A.B designates a submodule named B in a package named A.
- A package is basically a directory with Python files.

4.4.1 Package example

[Click here](#)

Packages



Input-Output

Reading from keyboard

- To read data from keyboard “`input()`” is use.
- The input of the user will be returned as a string without any changes.
- If this raw input has to be transformed into another data type needed by the algorithm, we can use either a casting function or the `eval` function

Printing on screen

- “print()” is use to print on screen.
- `print(value, ..., sep=' ', end='\n', file=sys.stdout)`
- prints the values to a stream, or to `sys.stdout` by default.
- Optional keyword arguments:

`file`: a file-like object (stream); defaults to the current `sys.stdout`. `sep`: string

inserted between values, default a space. `end`: string appended after the last value, default a newline.

Files and Exceptions

Handling

Opening and Closing file

- To read data from keyboard “input()” is use.
- “open()” is use to open the file.
- It returns file object.
- syntax • `open(fileName,mode)`.
- `fileName`: Name of the file that we wants to open.
- `mode`: ‘r’ (only for reading), ‘w’ (only for writing), ‘a’ (for append) , r+ (for read and write). • Normally, files are opened in text mode, that means, you read and write strings from and to the file, which are encoded in a specific encoding.
- If encoding is not specified, the default is platform dependent

Close the file

call `f.close()` to close it and free up any system resources taken up by the open file.

Reading and writing files

- `file.read(size)` :This function is use to read a file's contents.
- If size is omitted or negative then the entire content is returned.
- If the end of the file has been reached, `f.read()` will return an empty string ('').
- `f.readline()` reads a single line from the file; a newline character (`\n`) is left at the end of the string, and is only omitted on the last line of the file if the file doesn't end in a newline.
- For reading lines from a file, you can loop over the file object.
- This is memory efficient, fast, and leads to simple code:

Reading and writing files

- `f.write(string)` : writes the contents of string to the file, returning the number of characters written.
- Other types of objects need to be converted – either to a string (in text mode) or a bytes object (in binary mode) –before writing them.
- `f.tell()` : It returns an integer giving the file object's current position in the file represented as number of bytes from the beginning of the file when in binary mode and an opaque number when in text mode.
- `f.seek()`: To change the file object's position. `f.seek(offset, from_what)`

Refer this examples

5.1.1 File example

[Click here](#)

5.2.1 Create folder

[Click here](#)

5.2.2 delete folder

[Click here](#)

Exception Handling

Exception

Exception handling

Try , except and finally clause

User Defined Exceptions

Exception

- In python , there are two distinguishable kinds of errors: syntax errors and exceptions.
- Syntax errors, also known as parsing errors
- Eg: `for i in range(1,10)`
`print(i)` Here missing `#"."` after `for` is syntax error.
- Exceptions : Even if a statement or expression is syntactically correct, it may cause an error when an attempt is made to execute it.
- Errors detected during execution are called exceptions

Exception

- Exceptions handling in Python is very similar to Java.
- But whereas in Java exceptions are caught by catch clauses, we have statements introduced by an "except" keyword in Python.
- Eg : n = int(input("Please enter a number: "))

Please enter a number: 23.50 Exception occurs like

ValueError: invalid literal for int() with base 10: '23.5'

Except clause

- A try statement may have more than one except clause for different exceptions.
- But at most one except clause will be executed.

6.1.1 Exception example 1

[Click here](#)

6.1.2 Exception example 2

[Click here](#)

Tryfinally clause

- try statement had always been paired with except clauses. But there is another way to use it as well.
- The try statement can be followed by a finally clause.
- Finally clauses are called clean-up or termination clauses, because they must be executed under all circumstances, i.e. a "finally" clause is always executed regardless if an exception occurred in a try block or not.

6.2.1 Try finally use

[Click Here](#)

User defined Exception

- Python also allows you to create your own exceptions by deriving classes from the standard built-in exceptions.

```
class MyNewError(Exception):  
    pass  
  
raise MyNewError("Something happened in my program")
```

OOPs Concept

- **OOPs Concept**

- **Class and object**
-
- **Attributes**
-
- **Inheritance**
-
- **Overloading**
-
- **Overriding**

Class And Object

- Python classes provide all the standard features of Object Oriented Programming: the class inheritance mechanism allows multiple base classes, a derived class can override any methods of its base class or classes, and a method can call the method of a base class with the same name.
- The class definition looks like :

```
class ClassName:
```

```
    Statement 1
```

```
    Statement 2 .....
```

```
    Statement N
```

Class And Object

- The statements inside a class definition will usually be function definitions, but other statements are also allowed.
- When a class definition is entered, a new namespace is created, and used as the local scope—thus, all assignments to local variables go into this new namespace.
- In particular, function definitions bind the name of the new function here.

Member methods in class

- The `class_suite` consists of all the component statements defining class members, data attributes and functions.
- The class attributes are data members (class variables and instance variables) and methods, accessed via dot notation.
- Eg. `displayDetails()`

Object

- Class objects support two kinds of operations: attribute references and instantiation.
- Attribute references use the standard syntax used for all attribute references in Python: `obj.name`
- Valid attribute names are all the names that were in the class's namespace when the class object was created..

Object

- Class objects support two kinds of operations: attribute references and instantiation.
- Class instantiation uses function notation.
- Just pretend that the class object is a parameterless function that returns a new instance of the class.

7.1.1 class and object example1: [Click here](#)

7.1.2 class and object example2: [Click here](#)

Inheritance

- Python supports inheritance, it even supports multiple inheritance. • Classes can inherit from other classes.
- A class can inherit attributes and behaviour methods from another class, called the superclass.
- A class which inherits from a superclass is called a subclass, also called heir class or child class.
- Superclasses are sometimes called ancestors as well.

7.2.1 Inheritance: [Click Here](#)

Overloading

- Python supports operator and function overloading.
- Method overloading
 - Overloading is the ability to define the same method, with the same name but with a different number of arguments and types.
 - It's the ability of one function to perform different tasks, depending on the number of parameters or the types of the parameters.
 - Python operators work for built-in classes.
 - But same operator behaves differently with different types.

Overloading

- For example, the + operator will, perform arithmetic addition on two numbers, merge two lists and concatenate two strings.
- This feature in Python, that allows same operator to have different meaning according to the context is called operator overloading.

7.3.1 Operator overloading example1: [Click here](#)

7.3.2 Operator overloading example2: [Click here](#)

7.3.3 Operator overloading example3: [Click here](#)

SQL

SQL

Modules:

- DBMS and RDBMS
- E-R Relational Schema
- Types of database
- Normalization
- Algebra
- Database Programming language SQL
- Keys: Primary Key, Unique Key, Foreign Key
- SQL Statement types: DML, DDL, TQL, TCL
- Joins
- Function
- Transaction Concept (Rollback, Commit, Save Point)

DBMS

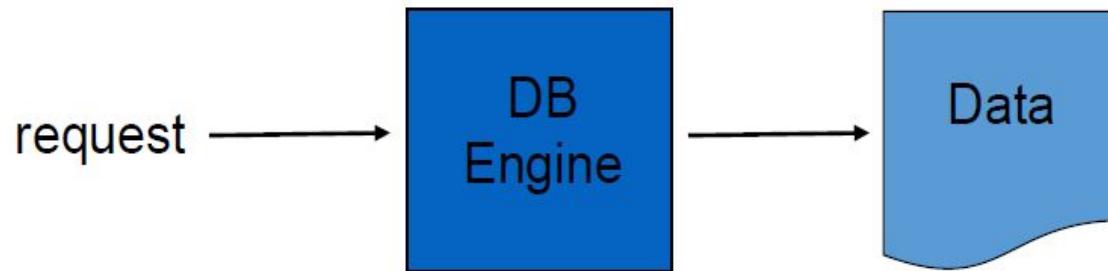
- DBMS stands for Data Base Management System.

Data + Management System

- **Database** is a collection of inter-related data and Management System is a set of programs to store and retrieve those data.
- **DBMS** is a collection of inter-related data and set of programs to store & access those data in an easy and effective manner.
- For Example, university database organizes the data about students, faculty, and admin staff etc. which **helps in efficient retrieval, insertion and deletion of data from it**.

Database Management Systems

- A DBMS consists of 2 main pieces:
 - the data
 - the DB engine
 - the data is typically stored in one or more files



- Two most common types of DBMS are:
 - Local
 - Server

Popular DBMS Software

Here, is the list of some popular DBMS system:

- MySQL
- Microsoft Access
- Oracle
- PostgreSQL
- SQLite
- Mongo DB
- IBM DB2, etc.

What is the need of DBMS?

Database systems are basically developed for large amount of data. When dealing with huge amount of data, there are two things that require optimization: **Storage of data and retrieval of data.**

Storage:

- According to the principles of database systems, the data is stored in such a way that it acquires lot less space as the redundant data (duplicate data) has been removed before storage.

Fast Retrieval of data:

- Along with storing the data in an optimized and systematic manner, it is also important that we retrieve the data quickly when needed. Database systems ensure that the data is retrieved as quickly as possible.

PURPOSE OF DBMS

The main purpose of database systems is to manage the data.

Consider a university that keeps the data of students, teachers, courses, books etc. To manage this data we need to store this data somewhere where we can add new data, delete unused data, update outdated data, retrieve data, to perform these operations on data we need a Database management system that allows us to store the data in such a way so that all these operations can be performed on the data efficiently.

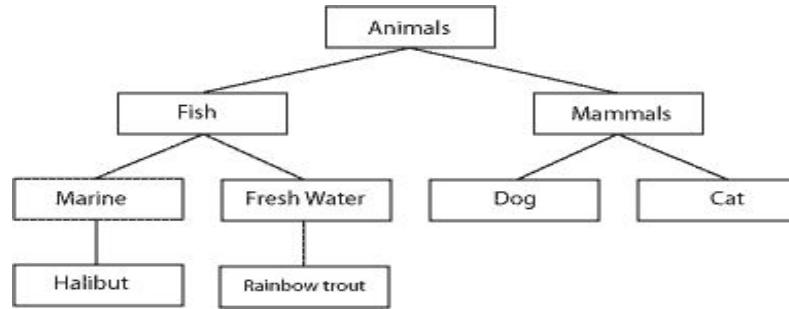
RDBMS

- Stands for "**Relational Database Management System.**"
- An RDBMS is a type of DBMS designed specifically for relational databases.
- A relational database refers to a database that stores data in a structured format, using rows and columns.
- This makes it easy to **locate and access specific values** within the database.
- It is "relational" because the values within each table are related to each other.
- **Tables may also be related to other tables.** The relational structure makes it possible to run queries across multiple tables at once.
- The RDBMS refers to the software that executes queries on the data, including adding, updating, and searching for values.
- An RDBMS may also provide a visual representation of the data. For example, **it may display data in a tables like a spreadsheet, allowing you to view and even edit individual values in the table.**

Types of DBMS ...

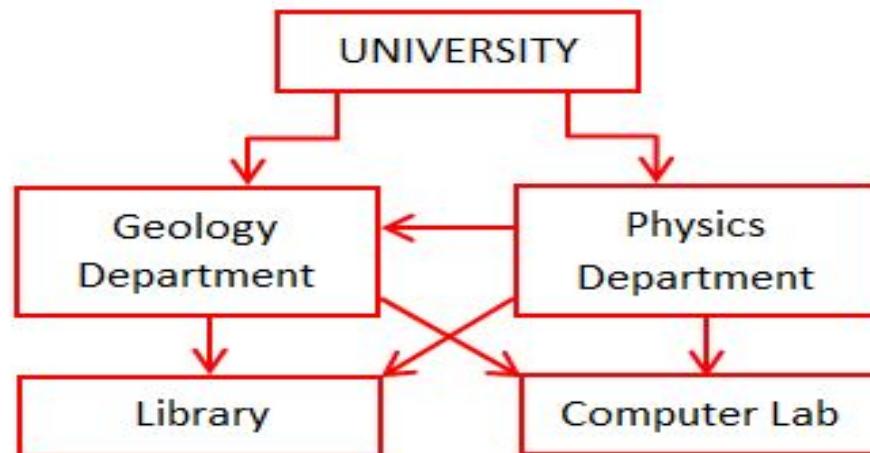
1.Hierarchical databases-

- It is very fast and simple.
- This kind of database model uses a tree-like structure which links a number of dissimilar elements to one primary record –the "owner" or "parent".
- Each record in a hierarchical database contains information about a group of parent child relationships.



2. Network databases –

- The network database model can be viewed as a net-like form where a single element can point to multiple data elements and can itself be pointed to by multiple data elements.
- The network database **model allows each record to have multiple parents** as well as multiple child records, which can be visualized as a web-like structure of networked records.



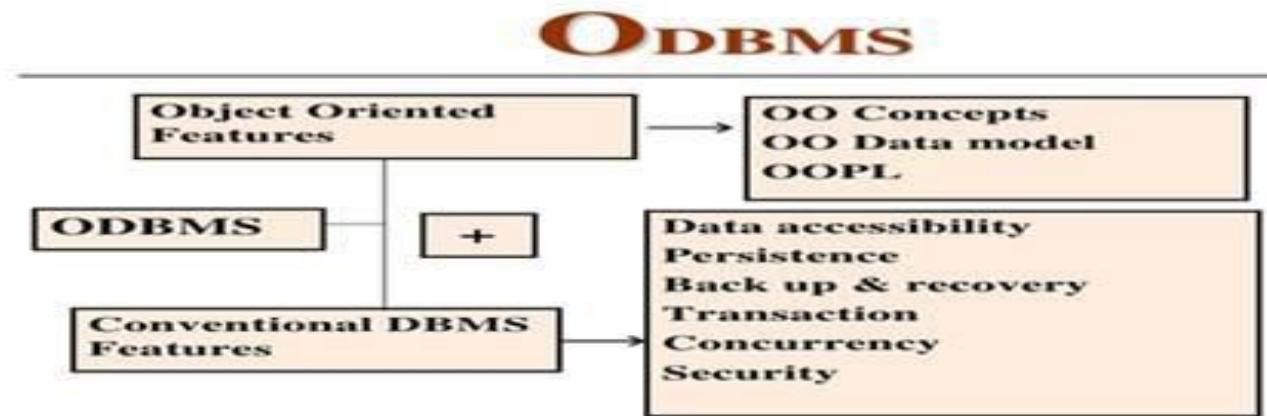
3. Relational databases –

- A relational database is one in which data is stored in the form of tables, using **rows and columns**.
- This arrangement makes it easy to locate and access specific data within the database. It is “**relational**” because the data within each table are related to each other.



4. Object-oriented databases -

- The recent development in database technology is the incorporation of the object concept that has become significant in programming languages.
- In object-oriented databases, all data are objects. Objects may be linked to each other by an “is-part-of” relationship to represent larger, composite objects
- Object DBMS's increase the semantics of the C++ and Java



Relational Databases

- RDBMS is the basis for SQL, and for all modern database systems like MS SQL Server, IBM DB2, Oracle, MySQL, and Microsoft Access.
- Most of today's databases are relational:
 - database contains 1 or more tables
 - table contains 1 or more records
 - record contains 1 or more fields
 - fields contain the data
- So why is it called "relational"?
 - tables are related (joined) based on common fields

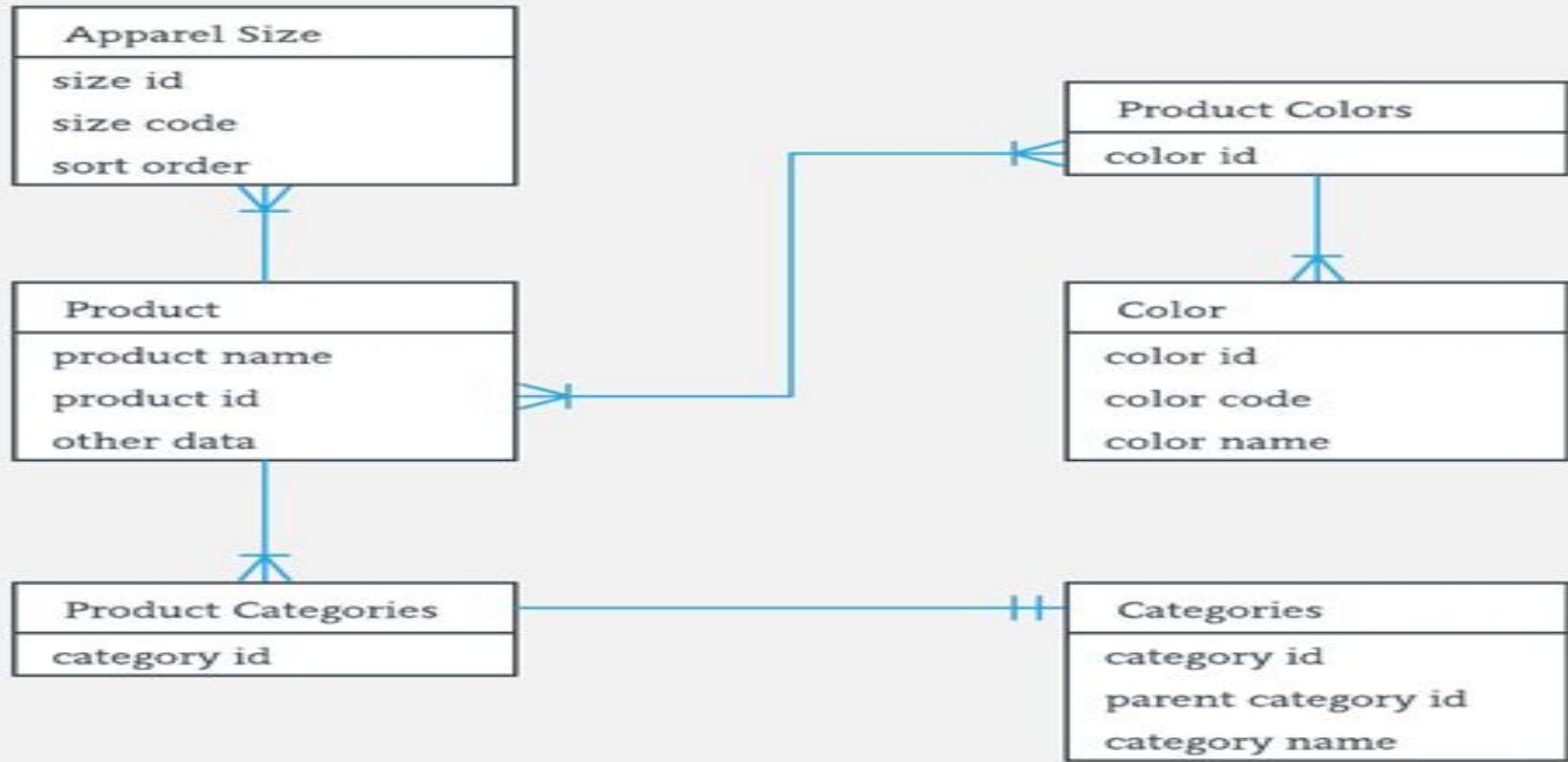
E-R Model ...

- The ER or (Entity Relational Model) is a high-level conceptual data model diagram.
- Entity-Relation model is based on the notion of real-world entities and the relationship between them.
- ER modeling helps you to analyze data requirements systematically to produce a well-designed database.
- So, it is considered a best practice to complete ER modeling before implementing your database.

ER Diagram

- Entity relationship diagram displays the relationships of entity set stored in a database.
- In other words, we can say that ER diagrams help you to explain the logical structure of databases.
- At first look, an ER diagram looks very similar to the flowchart. However, ER Diagram includes many specialized symbols, and its meanings make this model unique.

ER Diagram



Components of ER Diagram

- Entities
- Attributes
- Relationships
- For example, in a University database, we might have entities for Students, Courses, and Lecturers. Students entity can have attributes like Rollno, Name, and DeptID. They might have relationships with Courses and Lecturers.

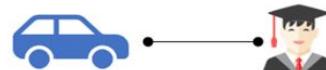
Components of ER Diagram



Entity

Person, place, object, event or concept about which data is to be maintained

Example: Car, Student



Jack



Attribute
Property or characteristic of an entity

Example: Color of car Entity
Name of Student Entity

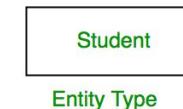


Association between the instances of one or more entity types

Example: Blue Car Belongs to Student Jack

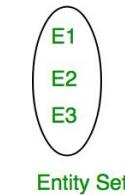
Entity

- A real-world thing either living or non-living that is easily recognizable and non recognizable. It is anything in the enterprise that is to be represented in our database. It may be a physical thing or simply a fact about the enterprise or an event that happens in the real world.
- An entity can be place, person, object, event or a concept, which stores data in the database. The characteristics of entities are must have an attribute, and a unique key. Every entity is made up of some 'attributes' which represent that entity.
- Set of all entity is called **entity set**



Examples of entities:

- **Person:**Employee, Student, Patient
- **Place:**Store, Building



Attribute

Attributes are the properties which define the entity type. For example, Roll_No, Name, DOB, Age, Address, Mobile_No are the attributes which defines entity type Student. In ER diagram, attribute is represented by an oval.

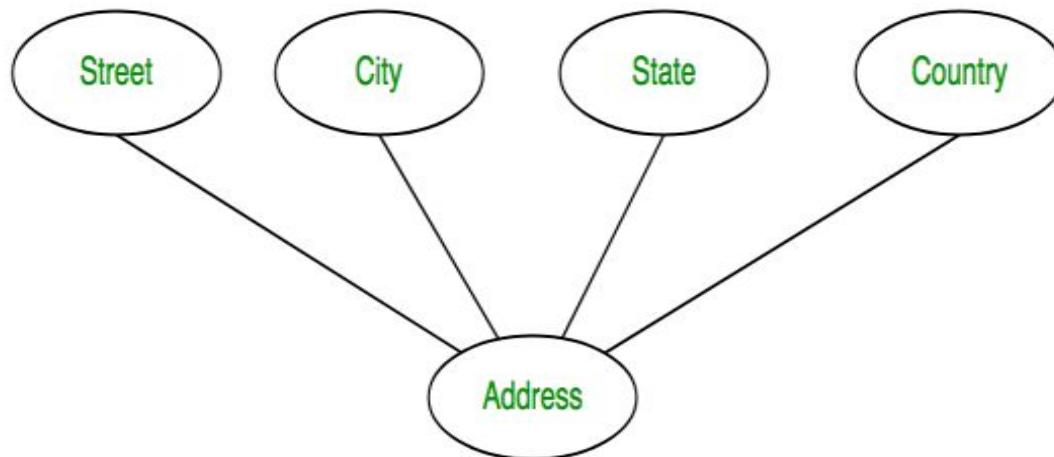
1. Key Attribute:

- The attribute which uniquely identifies each entity in the entity set is called key attribute.
- For example, Roll_No will be unique for each student. In ER diagram, key attribute is represented by an oval with underlying lines.



2. Composite Attribute –

An attribute composed of many other attribute is called as composite attribute. For example, Address attribute of student Entity type consists of Street, City, State, and Country. In ER diagram, composite attribute is represented by an oval comprising of ovals



3. Multivalued Attribute –

An attribute consisting more than one value for a given entity. For example, Phone_No(can be more than one for a given student). In ER diagram, multivalued attribute is represented by double oval.

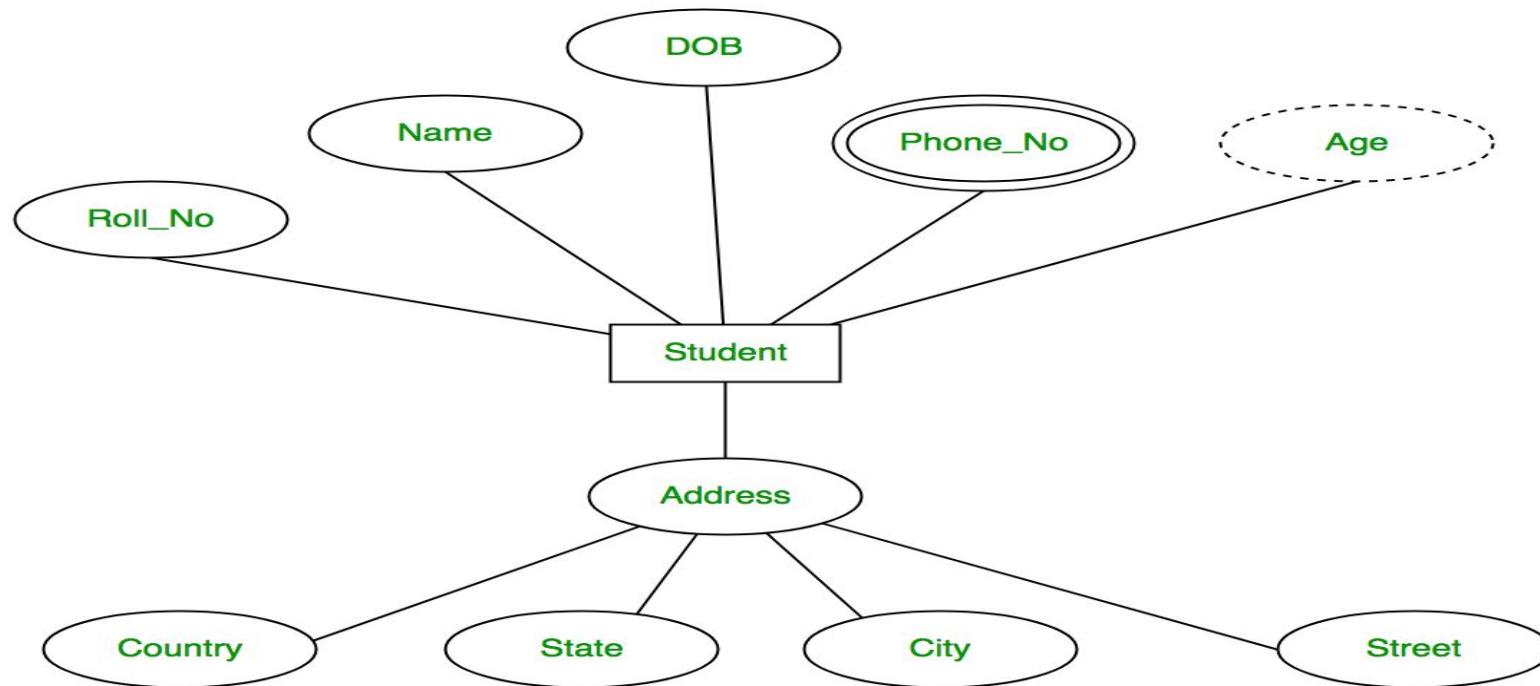


4. Derived Attribute –

An attribute which can be derived from other attributes of the entity type is known as derived attribute. e.g.; Age (can be derived from DOB). In ER diagram, derived attribute is represented by dashed oval.



The complete entity type Student with its attributes can be represented as:

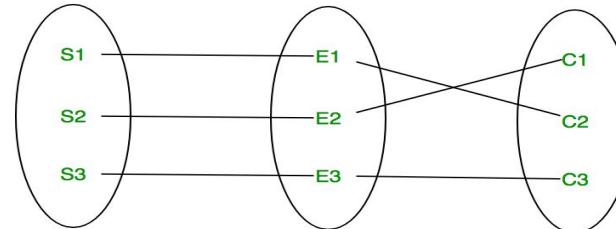


Relationship Type and Relationship Set:

A relationship type represents the association between entity types. For example, ‘Enrolled in’ is a relationship type that exists between entity type Student and Course. In ER diagram, relationship type is represented by a diamond and connecting the entities with lines.



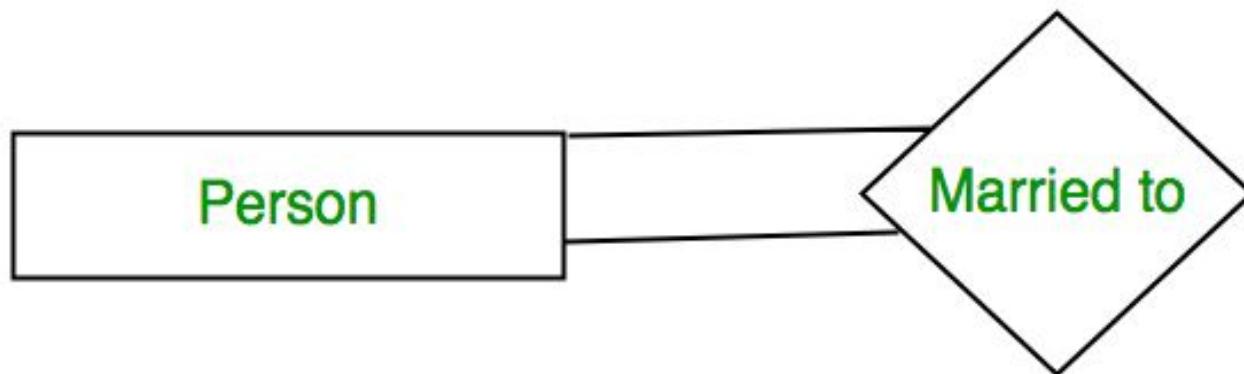
A set of relationships of same type is known as relationship set. The following relationship set depicts S1 is enrolled in C2, S2 is enrolled in C1 and S3 is enrolled in C3.



Degree of a relationship set:

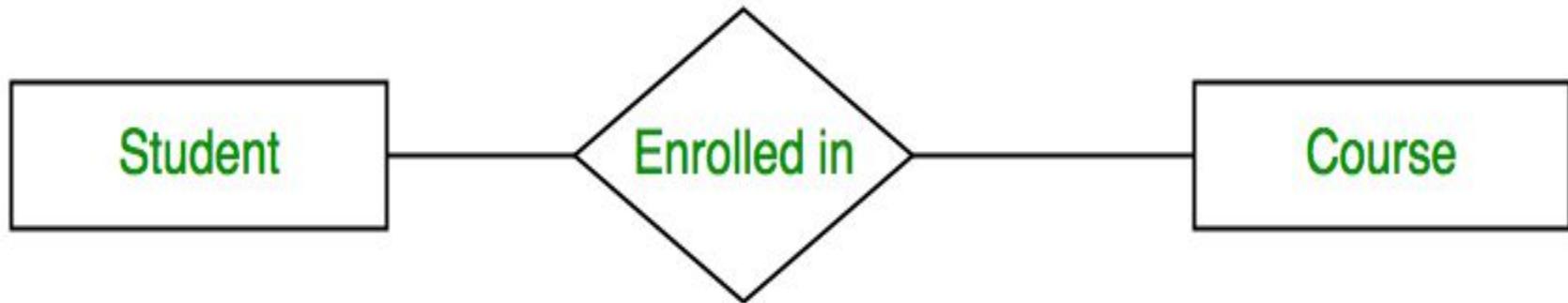
1. Unary Relationship –

When there is only ONE entity set participating in a relation, the relationship is called as unary relationship. For example, one person is married to only one person.



2. Binary Relationship–

When there are TWO entities set participating in a relation, the relationship is called as binary relationship. For example, Student is enrolled in Course.

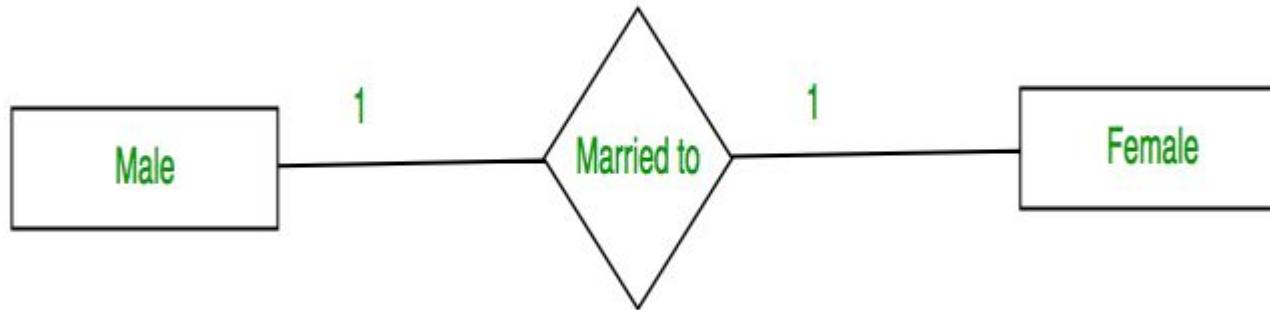


3. N-ary Relationship –

When there are n entities set participating in a relation, the relationship is called as n-ary relationship.

The number of times an entity of an entity set participates in a relationship set is known as cardinality. Cardinality can be of different types:

One to one – When each entity in each entity set can take part only once in the relationship, the cardinality is one to one. Let us assume that a male can marry to one female and a female can marry to one male. So the relationship will be one to one.



Many to one –When entity set can take part only once in the relationship set and entities in other entity set can take part more than once in the relationship set, cardinality is many to one.



Many to many –When entities in all entity sets can take part more than once in the relationship cardinality is many to many. Let us assume that a student can take more than one course and one course can be taken by many students. So the relationship will be many to many.



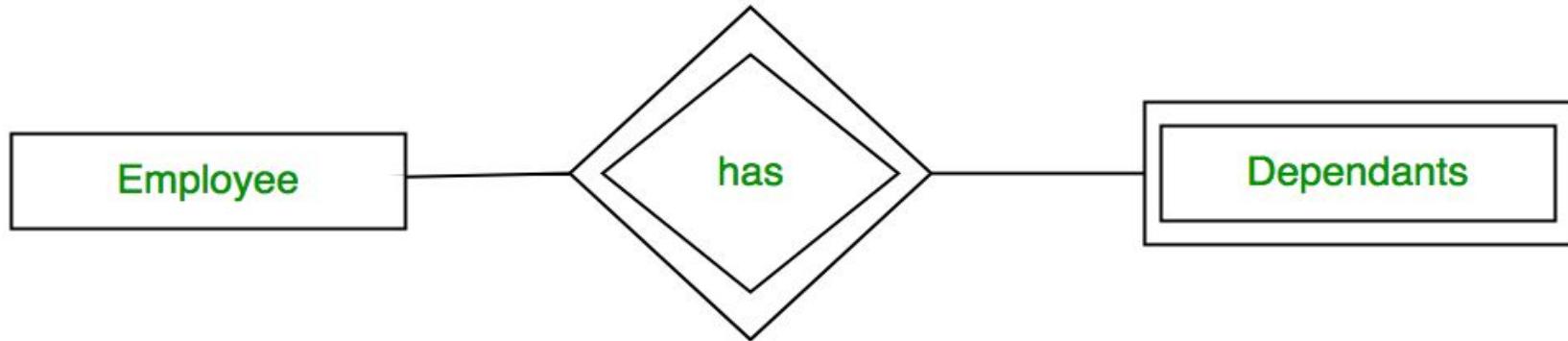
Participation Constraint: Participation Constraint is applied on the entity participating in the relationship set.

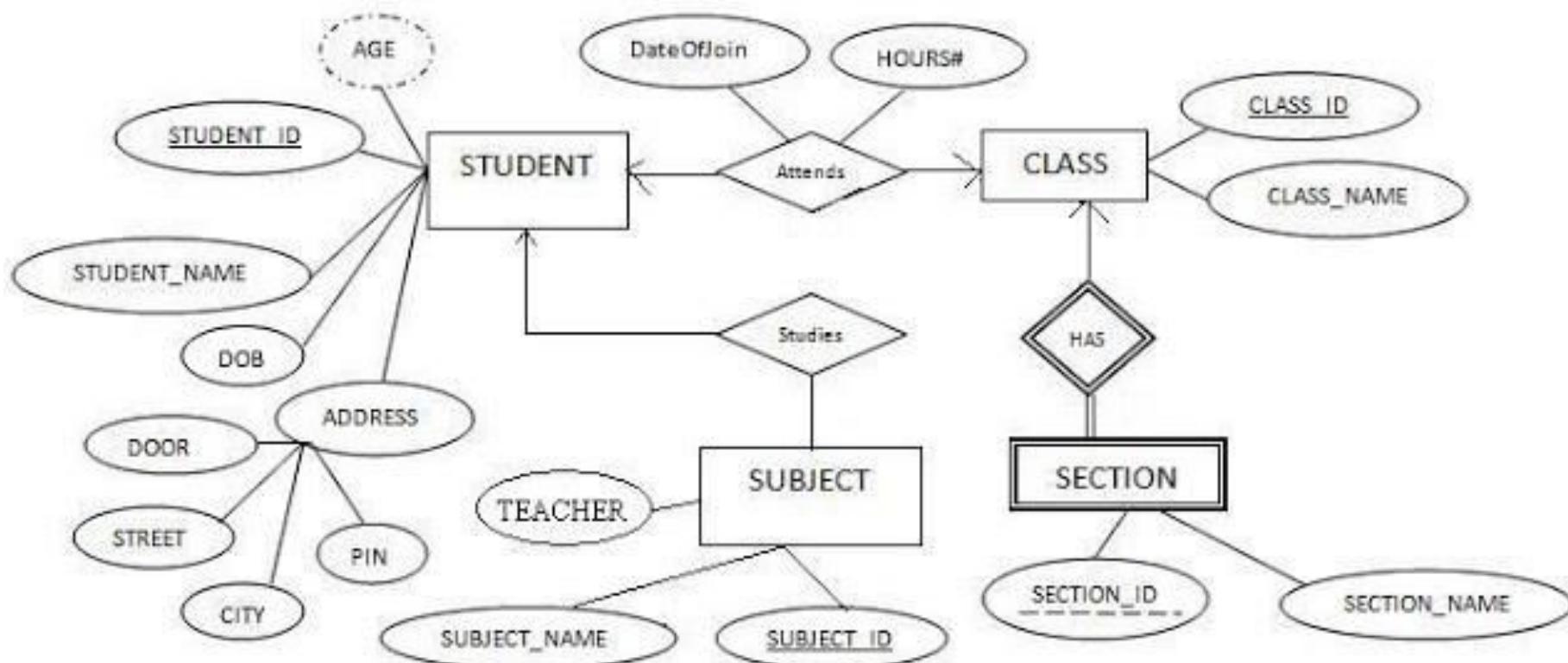
Total Participation –Each entity in the entity set must participate in the relationship. If each student must enroll in a course, the participation of student will be total. Total participation is shown by double line in ER diagram.

Partial Participation –The entity in the entity set may or may NOT participate in the relationship. If some courses are not enrolled by any of the student, the participation of course will be partial. The diagram depicts the ‘Enrolled in’ relationship set with Student Entity set having total participation and Course Entity set having partial participation

Weak Entity Type and Identifying Relationship:

- An entity type has a key attribute which uniquely identifies each entity in the entity set.
- But there exists some entity type for which key attribute can't be defined. These are called Weak Entity type.
- For example, A company may store the information of dependants (Parents, Children, Spouse) of an Employee. But the dependants don't have existence without the employee. So Dependant will be weak entity type and Employee will be Identifying Entitytype for Dependant.





Algebra

Relational Algebra is procedural query language, which takes Relation as input and generate relation as output. Relational algebra mainly provides theoretical foundation for relational databases and SQL.

Unary Relational Operations

- SELECT (symbol: σ)
- PROJECT (symbol: π)
- RENAME (symbol:)

Binary Relational Operations

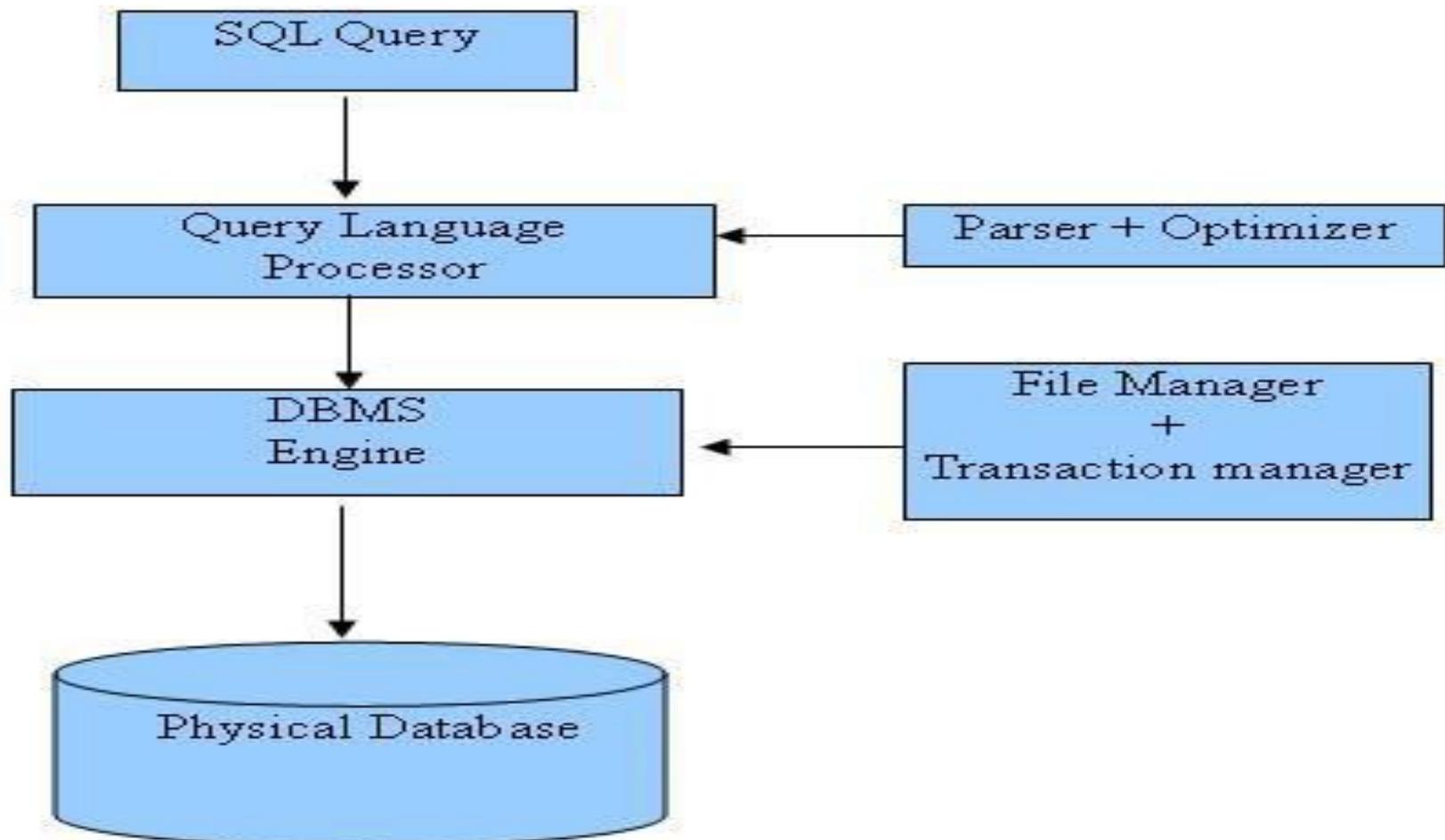
- JOIN
- DIVISION

Relational Algebra Operations From Set Theory

- UNION (u)
- INTERSECTION (),
- DIFFERENCE (-)
- CARTESIAN PRODUCT (x)

What is SQL?

- SQL is **Structured Query Language**, which is a computer language for **storing, manipulating and retrieving data** stored in relational database.
- SQL is a language of database, it includes database creation, deletion, fetching rows and modifying rows etc.
- **SQL is the standard language for Relation Database System.** All relational database management systems like MySQL, MS Access, Oracle, Sybase, Informix, postgre and SQL Server use SQL as standard database language.
- Also, they are using different dialects, such as:
- MS SQL Server using T-SQL, ANSI SQL
- Oracle using PL/SQL
- MS Access version of SQL is called JET SQL (native format) etc



Why SQL?

- Allows users to access data in relational database management systems.
- Allows users to describe the data.
- Allows users to define the data in database and manipulate that data.
- Allows to embed within other languages using SQL modules, libraries & pre-compilers.
- Allows users to create and drop databases and tables.
- Allows users to create view, stored procedure, functions in a database.
- Allows users to set permissions on tables, procedures, and views

What is SQL? (Cont...)

- SQL stands for Structured Query Language
- SQL allows you to access a database
- SQL is an ANSI standard computer language
- SQL can execute queries against a database
- SQL can retrieve data from a database
- SQL can insert new records in a database
- SQL can delete records from a database
- SQL can update records in a database
- SQL is easy to learn
- SQL is written in the form of queries
- action queries insert, update & delete data
- select queries retrieve data from DB

Keys

Primary Key:

- A primary key is a column of table which uniquely identifies each tuple (row) in that table.
- Primary key enforces integrity constraints to the table.
- **Only one primary key is allowed to use in a table.**
- The primary key does not accept the **any duplicate and NULL values**.
- The primary key value in a table changes very rarely so it is chosen with care where the changes can occur in a seldom manner.
- A primary key of one table can be referenced by foreign key of another table.

Table : Student

Roll_number	Name	Batch	Phone_number	Citizen_ID
Primary key				

Keys

Unique Key:

- Unique key constraints also identifies an individual table uniquely in a relation or table.
- A table **can have more than one unique key** unlike primary key.
- Unique key constraints can accept **only one NULL value** for column.
- Unique constraints are also referenced by the foreign key of another table.
- It can be used when someone wants to enforce unique constraints on a column and a group of columns which is not a primary key.

Table : Student

Roll_number	Name	Batch	Phone_number	Citizen_ID
12345	John Doe	CS-101	9876543210	1234567890

Unique key

Keys

Foreign Key:

- When, "one" table's primary key field is added to a related "many" table in order to create the common field which relates the two tables, it is called a foreign key in the "many" table.
- In the example given below, salary of an employee is stored in salary table. Relation is established via is stored in "Employee" table. To identify the salary of "Jforeign key column "Employee_ID_Ref" which refers "Employee_ID" field in Employee table.of "Jhon" is stored in "Salary" table. But his employee info For example, salary hon", his "employee id" is stored with each salary record.

Table : Employee	
Employee_ID	Employee Name
1	Jhon
2	Alex
3	James
4	Roy
5	Kay

Table : Salary				
Employee_ID_Ref	Year	Month	Salary	
1	2012	April	30000	
1	2012	May	31000	
1	2012	June	32000	
2	2012	April	40000	
2	2012	May	41000	
2	2012	June	42000	



Database Normalization ...

- Normalization is the process of minimizing redundancy (duplicity) from a relation or set of relations.
- Redundancy in relation may cause insertion, deletion and updation anomalies. So, it helps to minimize the redundancy in relations.

Most Commonly used normal forms:

- First normal form(1NF)
- Second normal form(2NF)
- Third normal form(3NF)
- Boyce & Codd normal form (BCNF)

First Normal Form

- If a relation contain composite or multi-valued attribute, it violates first normal form or a relation is in first normal form if it does not contain any composite or multi-valued attribute.
- A relation is in first normal form if every attribute in that relation is singled valued attribute.

Example 1 –Relation STUDENT in table 1 is not in 1NF because of multi-valued attribute STUD_PHONE. Its decomposition into 1NF has been shown in table 2.

STUD_NO	STUD_NAME	STUD_PHONE	STUD_STATE	STUD_COUNTRY
1	RAM	9716271721, 9871717178	HARYANA	INDIA
2	RAM	9898297281	PUNJAB	INDIA
3	SURESH		PUNJAB	INDIA

Table 1

Conversion to first normal form

STUD_NO	STUD_NAME	STUD_PHONE	STUD_STATE	STUD_COUNTRY
1	RAM	9716271721	HARYANA	
1	RAM	9871717178	HARYANA	INDIA
2	RAM	9898297281	PUNJAB	INDIA
3	SURESH		PUNJAB	INDIA

Table 2

Second Normal Form

- To be in second normal form, a relation must be in first normal form and relation must not contain any partial dependency.
- relation is in 2NF if it has No Partial Dependency,i.e.,no non-prime attribute (attributes which are not part of any candidate key) is dependent on any proper subset of any candidate key of the table.
- Partial Dependency –If the proper subset of candidate key determines non-prime attribute, it is called partial dependency.
- The table is not in 2nf form

STUD_NO	COURSE_NO	COURSE_FEE
1	C1	1000
2	C2	1500
1	C4	2000
4	C3	1000
4	C1	1000
2	C5	2000

- Note that, there are many courses having the same course fee.
- COURSE_FEE cannot alone decide the value of COURSE_NO or STUD_NO;
- COURSE_FEE together with STUD_NO cannot decide the value of COURSE_NO;
- COURSE_FEE together with COURSE_NO cannot decide the value of STUD_NO;
- Hence, COURSE_FEE would be a non-prime attribute, as it does not belong to the one only candidate key {STUD_NO, COURSE_NO} ;
- But, COURSE_NO \rightarrow COURSE_FEE , i.e., COURSE_FEE is dependent on COURSE_NO, which is a proper subset of the candidate key.
- Non-prime attribute COURSE_FEE is dependent on a proper subset of the candidate key, which is a partial dependency and so this relation is not in 2NF.

To convert the above relation to 2NF, we need to split the table into two tables such as :
Table 1: STUD_NO, COURSE_NOM
Table 2: COURSE_NO, COURSE_FEE

STUD_NO	COURSE_NO	COURSE_FEE
1	C1	1000
2	C2	1500
1	C4	2000
4	C3	1000
4	C1	1000
2	C5	2000



Table 1		Table 2	
STUD_NO	COURSE_NO	COURSE_NO	COURSE_FEE
1	C1	C1	1000
2	C2	C2	1500
1	C4	C3	1000
4	C3	C4	2000
4	C1	C5	2000

Third Normal Form

- A relation is in third normal form, if there is no transitive dependency for non-prime attributes as well as it is in second normal form. A relation is in 3NF if at least one of the following condition hold sin every non-trivial function dependency $X \rightarrow Y$
 - X is a super key.
 - Y is a prime attribute (each element of Y is part of some candidate key).
- **Transitive dependency** –If $A \rightarrow B$ and $B \rightarrow C$ are two FDs then $A \rightarrow C$ is called transitive dependency.

STUD_NO	STUD_NAME	STUD_STATE	STUD_COUNTRY	STUD AGE
1	RAM	HARYANA	INDIA	20
2	RAM	PUNJAB	INDIA	19
3	SURESH	PUNJAB	INDIA	21

Table 4

In relation STUDENT given in Table 4, FD set: {STUD_NO \rightarrow STUD_NAME, STUD_NO \rightarrow STUD_STATE, STUD_STATE \rightarrow STUD_COUNTRY, STUD_NO \rightarrow STUD_AGE} Candidate Key: {STUD_NO}

For this relation in table 4, STUD_NO \rightarrow STUD_STATE and STUD_STATE \rightarrow STUD_COUNTRY are true. STUD_COUNTRY is transitively dependent on STUD_NO. It violates the third normal form. To convert it in third normal form, we will decompose the relation STUDENT (STUD_NO, STUD_NAME, STUD_PHONE, STUD_STATE, STUD_COUNTRY, STUD_AGE) as: STUDENT (STUD_NO, STUD_NAME, STUD_PHONE, STUD_STATE, STUD_AGE) STATE_COUNTRY (STATE, COUNTRY)

Boyce Codd normal form (BCNF)

- It is an advance version of 3NF that's why it is also referred as 3.5NF. BCNF is stricter than 3NF.
- A table complies with BCNF if it is in 3NF and for every functional dependency $X \rightarrow Y$, X should be the super key of the table.

Example: Suppose there is a company wherein employees work in more than one department. They store the data like this:

emp_id	emp_nationality	emp_dept	dept_type	dept_no_of_emp
1001	Austrian	Production and planning	D001	200
1001	Austrian	stores	D001	250
1002	American	design and technical support	D134	100
1002	American	Purchasing department	D134	600

- **Functional dependencies :**
 - emp_id-> emp_nationality
 - emp_dept->{dept_type, dept_no_of_emp}
- **Candidate key:** {emp_id, emp_dept}
 - The table is not in BCNF as neither emp_id nor emp_dept alone are keys.
- To make the table comply with BCNF we can break the table in three tables like this:

emp_nationality table:

emp_id	emp_nationality
1001	Austrian
1002	American

emp_dept table:

emp_dept	dept_type	dept_no_of_emp
Production and planning	D001	200
stores	D001	250
design and technical support	D134	100
Purchasing department	D134	600

emp_dept_mapping table:

emp_id	emp_dept
1001	Production and planning
1001	stores
1002	design and technical support
1002	Purchasing department

SQL Process

- When you are executing an SQL command for any RDBMS, the system determines the best way to carry out your request and SQL engine figures out how to interpret the task.
- There are various components included in the process.
 - These components are Query Dispatcher, Optimization Engines, Classic Query Engine and SQL Query Engine, etc.
 - Classic query engine handles all non-SQL queries but SQL query engine won't handle logical files.

SQL Statement Types

- **DDL**–Data Definition Language
- **DML**–Data Manipulation Language
- **DCL**–Data Control Language
- **DQL**–Data Query Language

DDL -Data Definition Language

Command	Description
CREATE	Creates a new table, a view of a table, or other object in database
ALTER	Modifies an existing database object, such as a table.
DROP	Deletes an entire table, a view of a table or other object in the database.

DQL –Data Query Language

Command	Description
SELECT	Retrieves certain records from one or more tables

DML –Data Manipulation Language

Command	Description
INSERT	Creates a record
UPDATE	Modifies records
DELETE	Deletes records

DCL –Data Control Language

Command	Description
GRANT	Gives a privilege to user
REVOKE	Takes back privileges granted from user

Create, Drop, Use Database Syntax

- SQL CREATE DATABASE STATEMENT
 - `CREATE DATABASE database_name;`
- SQL DROP DATABASE Statement:
 - `DROP DATABASE database_name;`
- SQL USE STATEMENT
 - `USE DATABASE database_name;`

Create, Drop, Alter Table Syntax

- SQL CREATE TABLE STATEMENT
 - `CREATE TABLE table_name(column1 datatype, column2 datatype, column3 datatype, , columnN datatype, PRIMARY KEY(one or more columns));`
- SQL DROP TABLE STATEMENT
 - `DROP TABLE table_name;`
- SQL TRUNCATE TABLE STATEMENT
 - `TRUNCATE TABLE table_name;`
- SQL ALTER TABLE STATEMENT
 - `ALTER TABLE table_name ADD NEW_COL INT`
 - `ALTER TABLE table_name{ADD|DROP|MODIFY}column_name{data_ype};`
- SQL ALTER TABLE STATEMENT (RENAME)
 - `ALTER TABLE table_nameRENAME TO new_table_name;`

Insert, Update, Delete Syntax

- SQL INSERT INTO STATEMENT
 - `INSERT INTO table_name(column1, column2....columnN) VALUES (value1, value2....valueN);`
- SQL UPDATE STATEMENT
 - `UPDATE table_nameSET column1 = value1, column2 = value2....columnN=valueN[WHERE CONDITION];`
- SQL DELETE STATEMENT
 - `DELETE FROM table_nameWHERE {CONDITION};`

Select Statement Syntax

- SQL SELECT STATEMENT
 - `SELECT column1, column2....columnN FROM table_name;`
- SQL DISTINCT CLAUSE
 - `SELECT DISTINCT column1, column2....columnN FROM table_name;`
- SQL WHERE CLAUSE
 - `SELECT column1, column2....columnN FROM table_name WHERE CONDITION;`
- SQL AND/OR CLAUSE
 - `SELECT column1, column2....columnN FROM table_name WHERE CONDITION-1 {AND|OR} CONDITION-2;`

Select Statement Syntax

- SQL IN CLAUSE
 - `SELECT column1, column2....columnN FROM table_name WHERE column_name IN (val-1, val-2,...val-N);`
- SQL BETWEEN CLAUSE
 - `SELECT column1, column2....columnN FROM table_name WHERE column_name BETWEEN val-1 AND val-2;`
- SQL LIKE CLAUSE
 - `SELECT column1, column2....columnN FROM table_name WHERE column_name LIKE { PATTERN };`
- SQL ORDER BY CLAUSE
 - `SELECT column1, column2....columnN FROM table_name WHERE CONDITION ORDER BY column_name{ASC|DESC};`

Select Statement Syntax

- SQL GROUP BY CLAUSE
 - `SELECT SUM(column_name) FROM table_name WHERE CONDITION
GROUP BY column_name;`
- SQL COUNT CLAUSE
 - `SELECT COUNT(column_name)FROM table_name WHERE CONDITION;`
- SQL HAVING CLAUSE
 - `SELECT SUM(column_name) FROM table_name WHERE CONDITION
GROUP BY column_nameHAVING (arithmeticfunction condition);`

Create and Drop Index Syntax

- SQL CREATE INDEX Statement :
 - CREATE UNIQUE INDEX index_nameON table_name(column1, column2,...columnN);
- SQL DROP INDEX STATEMENT
- ALTER TABLE table_nameDROP INDEX index_name;
- SQL DESC Statement :
 - DESC table_name;

Commit and Rollback Syntax

- SQL COMMIT STATEMENT
 - COMMIT;
- SQL ROLLBACK STATEMENT
 - ROLLBACK;

SQL Join Types

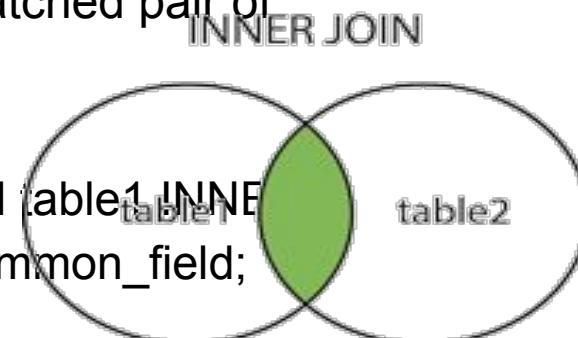
- **INNER JOIN:** returns rows when there is a match in both tables.
- **LEFT JOIN:** returns all rows from the left table, even if there are no matches in the right table.
- **RIGHT JOIN:** returns all rows from the right table, even if there are no matches in the left table.
- **FULL JOIN:** returns rows when there is a match in one of the tables.

JOIN

- A SQL Join statement is used to combine data or rows from two or more tables **based on a common field between them.**
- Different types of Joins are:
 - 1.INNER JOIN
 - 2.LEFT JOIN
 - 3.RIGHT JOIN
 - 4.FULL JOIN

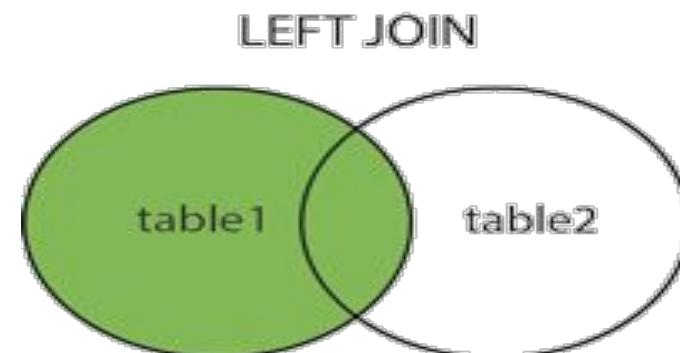
Inner Join Syntax

- The most frequently used and important of the joins is the INNER JOIN. They are also referred to as an EQUIJOIN.
-
- The INNER JOIN creates a new result table by combining column values of two tables (table1 and table2) based upon the join-predicate.
- The query compares each row of table1 with each row of table2 to find all pairs of rows which satisfy the join-predicate. When the join-predicate is satisfied, column values for each matched pair of rows of A and B are combined into a result row.
- SYNTAX:
 - `SELECT table1.column1, table2.column2...FROM table1 INNER JOIN table2 ON table1.common_field = table2.common_field;`



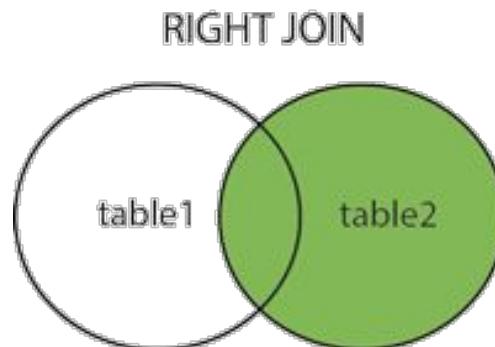
Left Join Syntax

- The SQL LEFT JOIN returns all rows from the left table, even if there are no matches in the right table. This means that if the ON clause matches 0 (zero) records in right table, the join will still return a row in the result, but with NULL in each column from right table.
- This means that a left join returns all the values from the left table, plus matched values from the right table or NULL in case of no matching join predicate.
- SYNTAX:
 - `SELECT table1.column1, table2.column2...FROM table1 LEFT JOIN table2 ON table1.common_field = table2.common_field;`



Right Join Syntax

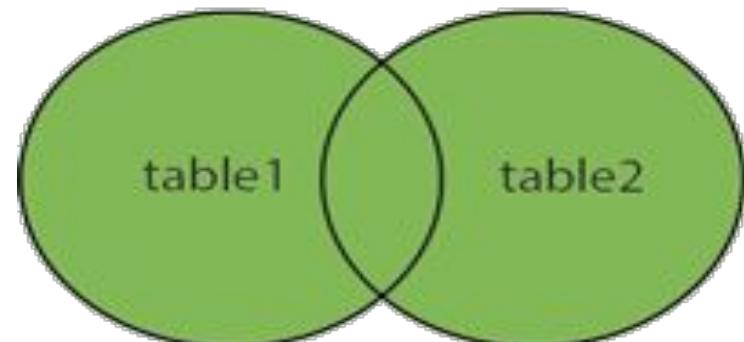
- The SQL RIGHT JOIN returns all rows from the right table, even if there are no matches in the left table. This means that if the ON clause matches 0 (zero) records in left table, the join will still return a row in the result, but with NULL in each column from left table.
- This means that a right join returns all the values from the right table, plus matched values from the left table or NULL in case of no matching join predicate.
- SYNTAX:
 - `SELECT table1.column1, table2.column2...FROM table1 RIGHT JOIN table2 ON table1.common_field = table2.common_field;`



Full Join Syntax

- The SQL FULL JOIN combines the results of both left and right outer joins.
- The joined table will contain all records from both tables, and fill in NULLs for missing matches on either side.
- SYNTAX:
 - `SELECT table1.column1, table2.column2...FROM table1 FULL JOIN table2 ON table1.common_field = table2.common_field;`

FULL OUTER JOIN



Function

SQL has many built-in functions for performing calculations on data. They are divided into 2 categories:

1. Aggregate Function
2. Scalar Function

Aggregate Function

These functions are used to do operations from the values of the column and a single value is returned.

- AVG()-Returns the average value
- COUNT()-Returns the number of rows
- FIRST()-Returns the first value
- LAST()-Returns the last value
- MAX()-Returns the largest value
- MIN()-Returns the smallest value
- SUM() -Returns the sum

1. AVG()

- Syntax:
 - `SELECT AVG(column_name) FROM table_name;`
- Example:
 - `SELECT AVG(AGE) AS AvgAge FROM Students;`
- Output:

AvgAge

19.4

Student Table

ID	NAME	MARKS	AGE
1	Harsh	90	19
2	Suresh	50	20
3	Pratik	80	19
4	Dhanraj	95	21
5	Ram	85	18

2. COUNT()

- Syntax:
 - `SELECT COUNT(column_name) FROM table_name;`
- Example:
 - `SELECT COUNT(*) AS NumStudents FROM Students;`

- Output:

NumStudents
5

Student Table

ID	NAME	MARKS	AGE
1	Harsh	90	19
2	Suresh	50	20
3	Pratik	80	19
4	Dhanraj	95	21
5	Ram	85	18

3. FIRST()

- Syntax:
 - `SELECT FIRST(column_name) FROM table_name;`
- Example:
 - `SELECT FIRST(MARKS) AS MarksFirst FROM Students;`

- Output:

MarksFirst
90

Student Table

ID	NAME	MARKS	AGE
1	Harsh	90	19
2	Suresh	50	20
3	Pratik	80	19
4	Dhanraj	95	21
5	Ram	85	18

4. LAST()

- Syntax:
 - `SELECT LAST(column_name) FROM table_name;`
- Example:
 - `SELECT LAST(MARKS) AS MarksLast FROM Students;`

MarksLast

- Output:

82

Student Table

ID	NAME	MARKS	AGE
1	Harsh	90	19
2	Suresh	50	20
3	Pratik	80	19
4	Dhanraj	95	21
5	Ram	85	18

5. MAX()

- Syntax:
 - `SELECT MAX(column_name) FROM table_name;`
- Example:
 - `SELECT MAX(MARKS) AS MaxMarks FROM Students;`

- Output:

MaxMarks
95

Student Table

ID	NAME	MARKS	AGE
1	Harsh	90	19
2	Suresh	50	20
3	Pratik	80	19
4	Dhanraj	95	21
5	Ram	85	18

6. MIN()

- Syntax:
 - `SELECT MIN(column_name) FROM table_name;`
- Example:
 - `SELECT MIN(MARKS) AS MinMarks FROM Students;`

MinMarks
• Output: 50

Student Table

ID	NAME	MARKS	AGE
1	Harsh	90	19
2	Suresh	50	20
3	Pratik	80	19
4	Dhanraj	95	21
5	Ram	85	18

7. SUM()

- Syntax:
 - `SELECT SUM(column_name) FROM table_name;`
- Example:
 - `SELECT SUM(MARKS) AS TotalMarks FROM Students;`

• Output:

TotalMarks
400

Student Table

ID	NAME	MARKS	AGE
1	Harsh	90	19
2	Suresh	50	20
3	Pratik	80	19
4	Dhanraj	95	21
5	Ram	85	18

Scalar functions

These functions are based on user input, these too returns single value.

- **UCASE()** -Converts a field to upper case
- **LCASE()** -Converts a field to lower case
- **MID()** -Extract characters from a text field
- **LENGTH()** -Returns the length of a text field
- **ROUND()** -Rounds a numeric field to the number of decimals specified
- **NOW()** -Returns the current system date and time
- **FORMAT()** -Formats how a field is to be displayed

1. UCASE()

- Syntax:
 - `SELECT UCASE(column_name) FROM table_name;`
- Example:
 - `SELECT UCASE(NAME) FROM Students;`

NAME

HARSH

SURESH

PRATIK

DHANRAJ

RAM

Student Table

ID	NAME	MARKS	AGE
1	Harsh	90	19
2	Suresh	50	20
3	Pratik	80	19
4	Dhanraj	95	21
5	Ram	85	18

2. LCASE()

- Syntax:
 - `SELECT LCASE(column_name) FROM table_name;`
- Example:
 - `SELECT LCASE(NAME) FROM Students;`
- Output:

NAME

harsh

suresh

pratik

dhanraj

ram

Student Table

ID	NAME	MARKS	AGE
1	Harsh	90	19
2	Suresh	50	20
3	Pratik	80	19
4	Dhanraj	95	21
5	Ram	85	18

3. MID()

- Syntax:
 - SELECT MID(column_name,start,length) AS some_nameFROM table_name;
 - specifying length is optional here, and start signifies start position (starting from 1)
- Example:
 - SELECT MID(NAME,1,4) FROM Students;
- Output:

NAME
HARS
SURE
PRAT
DHAN
RAM

Student Table

ID	NAME	MARKS	AGE
1	Harsh	90	19
2	Suresh	50	20
3	Pratik	80	19
4	Dhanraj	95	21
5	Ram	85	18

4. LENGTH()

- Syntax:
 - `SELECT LENGTH(column_name) FROM table_name;`
- Example:
 - `SELECT LENGTH(NAME) FROM Students;`

NAME
5
6
6
7
3

Student Table

ID	NAME	MARKS	AGE
1	Harsh	90	19
2	Suresh	50	20
3	Pratik	80	19
4	Dhanraj	95	21
5	Ram	85	18

5. ROUND()

- Syntax:
 - `SELECT ROUND(column_name,decimals) FROM table_name;`
 - `decimals`-number of decimals to be fetched.
- Example:
 - `SELECT ROUND(MARKS,0) FROM table_name;`

MARKS
90
50
80
95
85

- Output:

Student Table

ID	NAME	MARKS	AGE
1	Harsh	90	19
2	Suresh	50	20
3	Pratik	80	19
4	Dhanraj	95	21
5	Ram	85	18

6. NOW()

- Syntax:
 - `SELECT NOW() FROM table_name;`
- Example:
 - `SELECT NAME, NOW() AS DateTime FROM Students`

	NAME	DateTime
• Output:	HARSH	1/13/2017 1:30:11 PM
	SURESH	1/13/2017 1:30:11 PM
	PRATIK	1/13/2017 1:30:11 PM
	DHANRAJ	1/13/2017 1:30:11 PM
	RAM	1/13/2017 1:30:11 PM

Student Table

ID	NAME	MARKS	AGE
1	Harsh	90	19
2	Suresh	50	20
3	Pratik	80	19
4	Dhanraj	95	21
5	Ram	85	18

7. FORMAT()

- Syntax:
 - `SELECT FORMAT(column_name) FROM table_name;`
- Example:
 - `SELECT NAME, FORMAT(Now(),'YYYY-MM-DD') AS Date FROM Students;`

	NAME	Date
	HARSH	2017-01-13
	SURESH	2017-01-13
	PRATIK	2017-01-13
	DHANRAJ	2017-01-13
	RAM	2017-01-13

Student Table

ID	NAME	MARKS	AGE
1	Harsh	90	19
2	Suresh	50	20
3	Pratik	80	19
4	Dhanraj	95	21
5	Ram	85	18

PROCEDURE

- A stored procedure is a prepared SQL code that you can save, so the code can be reused over and over again.
- So if you have an SQL query that you write over and over again, save it as a stored procedure, and then just call it to execute it.
- You can also pass parameters to a stored procedure, so that the stored procedure can act based on the parameter value(s) that is passed.

- **To create procedure, use syntax:**

```
CREATEPROCEDUREprocedure_name  
AS  
Sql_statement  
GO;
```

- **To execute created procedure, use syntax:**

```
EXECprocedure_name;
```

Refer This Example

- Cursor Example:
 - https://github.com/TopsCode/Data_AnalyticsWithPython_ML_AI/blob/master/SQL/Curso_r/example
- Stored Procedure with one parameter:
 - https://github.com/TopsCode/Data_AnalyticsWithPython_ML_AI/blob/master/SQL/Curso_r/cursorExampleOneparam
- Stored Procedure with multiple parameter:
 - https://github.com/TopsCode/Data_AnalyticsWithPython_ML_AI/blob/master/SQL/Curso_r/multipleParam

Trigger

- A trigger is a stored procedure in database which automatically invokes whenever a special event in the database occurs
- For example, a trigger can be invoked when a row is inserted into a specified table.
- **Syntax:**

```
create trigger [trigger_name]  
[before | after]  
{insert | update | delete}  
on [table_name]  
[for each row]  
[trigger_body]
```

Explanation of syntax:

create trigger [trigger_name]:Creates or replaces an existing trigger with the trigger_name.

[before | after]:This specifies when the trigger will be executed.

{insert | update | delete}:This specifies the DML operation.

on [table_name]:This specifies the name of the table associated with the trigger.

[for each row]: This specifies a row-level trigger, i.e., the trigger will be executed for each row being affected.

[trigger_body]: This provides the operation to be performed as trigger is fired

BEFORE and AFTER of Trigger:

BEFORE triggers run the trigger action before the triggering statement is run.

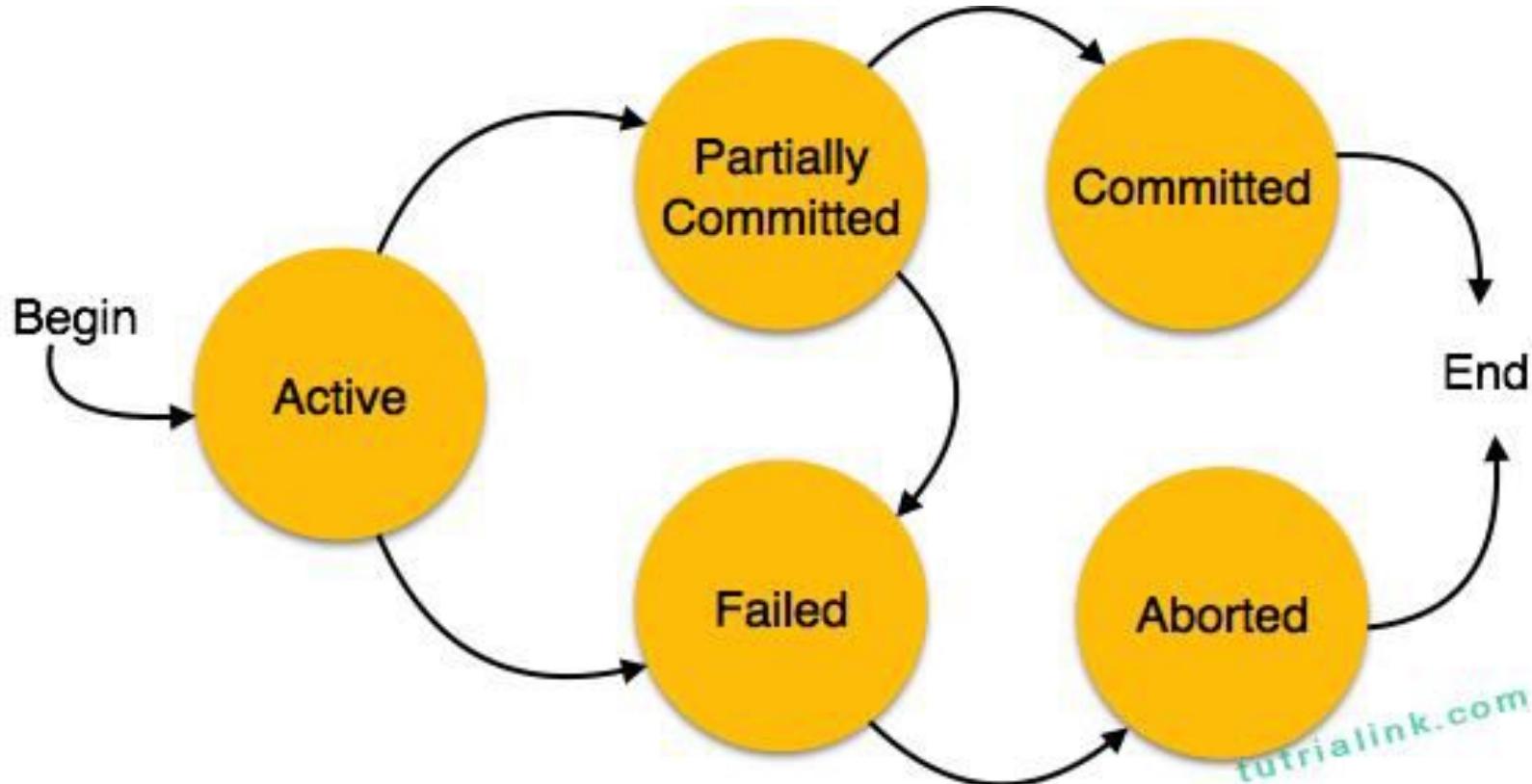
AFTER triggers run the trigger action after the triggering statement is run.

Refer This Example

- Before Trigger:
 - https://github.com/TopsCode/Data_AnalyticsWithPython_ML_AI/blob/master/SQL/Trigger/beforeTrigger
- After Trigger:
 - https://github.com/TopsCode/Data_AnalyticsWithPython_ML_AI/blob/master/SQL/Trigger/afterTrigger

Transaction

- A transaction is a logical unit of work of database processing that includes one or more database access operations.
- A transaction can be defined as an action or series of actions that is carried out by a single user or application program to perform operations for accessing the contents of the database.
- The operations can include retrieval, (Read), insertion (Write), deletion and modification.
- Each transaction begins with a specific task and ends when all the tasks in the group successfully complete. If any of the tasks fail, the transaction fails. Therefore, a transaction has only two results:success or failure.
- In order to maintain consistency in a database, before and after transaction, certain properties are followed. These are called **ACID** properties(**A**tomicity, **C**onsistency, **I**solation, **D**urability) .



ACID PROPERTY

3. Isolation:

- In a database system where more than one transaction are being executed simultaneously and in parallel, the property of isolation states that all the transactions will be carried out and executed as if it is the only transaction in the system.
- No transaction will affect the existence of any other transaction.
- For example, in an application that transfers funds from one account to another, the isolation property ensures that another transaction sees the transferred funds in one account or the other, but not in both, nor in neither.

Transaction Control

The following commands are used to control transactions.

- **COMMIT**- to save the changes.
- **ROLLBACK**- to roll back the changes.
- **SAVEPOINT**- creates points within the groups of transactions in which to ROLLBACK.

1. COMMIT

- The COMMIT command is the transactional command used to save changes invoked by a transaction to the database.
- The COMMIT command saves all the transactions to the database since the last COMMIT or ROLLBACK command.
- The syntax for the COMMIT command is as follows:
 - COMMIT;

2. ROLLBACK

- The ROLLBACK command is the transactional command used to undo transactions that have not already been saved to the database.
- This command can only be used to undo transactions since the last COMMIT or ROLLBACK command was issued.
- The syntax for the COMMIT command is as follows:
 - ROLLBACK;

3. SAVEPOINT

- A SAVEPOINT is a point in a transaction when you can roll the transaction back to a certain point without rolling back the entire transaction.
- The syntax for a SAVEPOINT command is as shown below.
 - `SAVEPOINT SAVEPOINT_NAME;`
- This command serves only in the creation of a SAVEPOINT among all the transactional statements. The ROLLBACK command is used to undo a group of transactions.
- The syntax for rolling back to a SAVEPOINT is as shown below.
 - `ROLLBACK TO SAVEPOINT_NAME;`

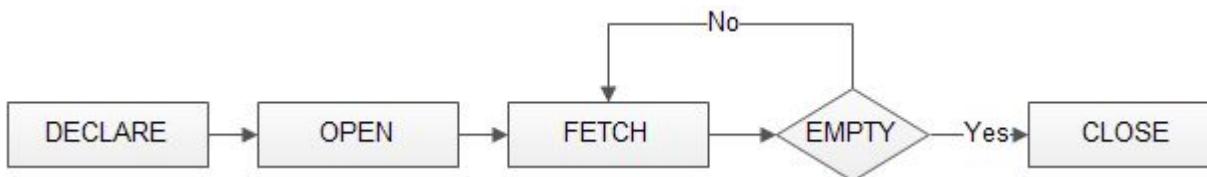
Cursor

- It is a temporary area for work in memory system while the execution of a statement is done.
- A Cursor in SQL is an arrangement of rows together with a pointer that recognizes a present row.
- It is a database object to recover information from a result set one row at once.
- It is helpful when we need to control the record of a table in a singleton technique, at the end of the day one row at any given moment. The arrangement of columns the cursor holds is known as the dynamic set.

Main components of Cursors

Each cursor contains the followings 5 parts,

- **Declare Cursor:** In this part we declare variables and return a set of values.
 - `DECLARE cursor_nameCURSOR FOR SELECT_statement;`
- **Open:** This is the entering part of the cursor.
 - `OPEN cursor_name;`
- **Fetch:** Used to retrieve the data row by row from a cursor.
 - `FETCH cursor_nameINTO variables list;`
- **Close:** This is an exit part of the cursor and used to close a cursor.
 - `CLOSE cursor_name;`



Syntax:

```
DECLARE variables;  
records;  
create a cursor;  
BEGIN  
OPEN cursor;  
FETCH cursor;  
process the records;  
CLOSE cursor;  
END;
```

Database Backup and Recovery

- Database Backup is storage of data that means the copy of the data.
- It is a safeguard against unexpected data loss and application errors.
- It protects the database against data loss.
- If the original data is lost, then using the backup it can reconstructed.
- The backups are divided into two types,
 1. Physical Backup
 2. Logical Backup

1. Physical Backups

- Physical Backups are the backups of the physical files used in storing and recovering your database, such as datafiles, control files and archived redo logs, log files.
- It is a copy of files storing database information to some other location, such as disk, some offline storage like magnetic tape.
- Physical backups are the foundation of the recovery mechanism in the database.
- Physical backup provides the minute details about the transaction and modification to the database.

2. Logical Backup

- Logical Backup contains logical data which is extracted from a database.
- It includes backup of logical data like views, procedures, functions, tables, etc.
- It is a useful supplement to physical backups in many circumstances but not a sufficient protection against data loss without physical backups, because logical backup provides only structural information.

Importance of Backups

- Planning and testing backup helps against failure of media, operating system, software and any other kind of failures that cause a serious data crash.
- It determines the speed and success of the recovery.
- Physical backup extracts data from physical storage (usually from disk to tape). Operating system is an example of physical backup.
- Logical backup extracts data using SQL from the database and store it in a binary file.
- Logical backup is used to restore the database objects into the database. So the logical backup utilities allow DBA (Database Administrator) to back up and recover selected objects within the database.

Causes of Failure

1. System Crash

- System crash occurs when there is a hardware or software failure or external factors like a power failure.
- The data in the secondary memory is not affected when system crashes because the database has lots of integrity. Checkpoint prevents the loss of data from secondary memory.

2. Transaction Failure

- The transaction failure is affected on only few tables or processes because of logical errors in the code.
- This failure occurs when there are system errors like deadlock or unavailability of system resources to execute the transaction.

3. Network Failure

- A network failure occurs when a client –server configuration or distributed database system are connected by communication networks.

4. Disk Failure

- Disk Failure occurs when there are issues with hard disks like formation of bad sectors, disk head crash, unavailability of disk etc.

5. Media Failure

- Media failure is the most dangerous failure because, it takes more time to recover than any other kind of failures.
- A disk controller or disk head crash is a typical example of media failure.

Recovery

- Recovery is the process of restoring a database to the correct state in the event of a failure.
- It ensures that the database is reliable and remains in consistent state in case of a failure.

Database recovery can be classified into two parts;

1. **Rolling Forward** applies redo records to the corresponding data blocks.
2. **Rolling Back** applies rollback segments to the datafiles. It is stored in transaction tables.

Excel

Modules:

- 1. Introductions to Excel**
- 2. Excel Functions**
- 3. Excel Charts**



Course Objectives

- ❖ Set up the chart.
- ❖ Differentiate between formulas and **functions** in Excel.
- ❖ Use Functions of Excel.
- ❖ Access and manipulate data.

Importance of Excel

- ❖ Microsoft Excel is one of the most important software programs ever created.
- ❖ Businesses large and small, in countries all around the world, run off of systems and processes built on and around Excel.
- ❖ So many people using Excel as part of their day-to-day jobs.
- ❖ Excel is not going anywhere, and businesses will continue to use Excel as a primary tool.
- ❖ Excel is a powerful tool that has become entrenched in business processes worldwide whether for analyzing or to Organize Data

Excel Vs Tableau

- ❖ Tableau is a Powerful **Data Visualization Tool**.
- ❖ But When it comes to Data Creation and Data Organization **Tableau is not Optimal**.
- ❖ Excel helps in generating computational solutions using built-in functions in easy way.
- ❖ User can learn to perform tasks like data manipulation, Simple Calculations and Charting just as easily, if not more easily, in Tableau.
- ❖ Learning Excel further helps you in fields like Data Analytics, Machine Learning, etc. where data preparation is a most important tasks.

Quick Access Toolbar

The screenshot shows the Microsoft Excel ribbon interface. The ribbon is divided into several tabs: File, Home, Insert, Page Layout, Formulas, Data, Review, View, Help, and Acrobat. A search bar at the top right says "Tell me what you want to do". The "Home" tab is selected, showing various tools like Cut, Copy, Paste, Font, Number, Styles, Cells, and Editing. The "Formula Bar" displays the cell reference "A1". The main workspace shows a grid with columns labeled A through V and rows labeled 1 through 22. The cell at row 9, column A (A9) is highlighted and labeled "Active Cell". The row number 9 is labeled "Row Number". The column label "J" is labeled "Column Number". The bottom left corner shows the "Work Sheet Tab" with "Sheet1" selected. The status bar at the bottom right shows "Ready" and "100%".

Ribbon

Name Box

Formula Bar

Active Cell

Row Number

Column Number

Work Sheet Tab

Book1 - Excel

File Home Insert Page Layout Formulas Data Review View Help Acrobat Tell me what you want to do

Cut Copy Paste Calibri 11 A A Merge & Center General \$ % , +0.00 Conditional Format as Table Cell Styles Insert Delete Format Cells AutoSum Fill Clear Sort & Find & Filter Select

A1

A B C D E F G H I J K L M N O P Q R S T U V

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

Sheet1

Ready 100%

Introduction to the Excel Interface

- ❖ **Title Bar** – contains the name of the workbook.
- ❖ **Worksheet Tabs** – a list of all the worksheets in the workbook.
- ❖ **Ribbon Tabs** – the top-level menu items consists of **Home**, **Insert**, **Page Layout**, **Formulas**, etc.
- ❖ **Ribbon** – a collection of Excel capabilities organized into **groups** corresponding to some ribbon tab.
- ❖ **Active Cell** – displays the currently referenced cell.
- ❖ **Name Box** – contains the address of the active cell.
- ❖ **Formula Bar** – contains the contents of the active cell.
- ❖ **File** – menu containing a number of options, such as open, save, and print.

CELL Reference

A **cell reference** or **cell address** is a combination of a column letter and a row number that identifies a **cell** on a worksheet.

CELL RANGE

A group of **cells** is known as a **cell range**.

1	A	B	C	D	E	F
2	Team	Country	Champions	Games play	Points earned	
3	Borussia	Germany	Yes	34	71	=A1:E5
4	Milan	Italy	Yes	38	57	
5	Manchester	England	Yes	38	64	
6	Hamburger	Germany	No	34	27	
7	Lazio	Italy	No	38	56	
8	Fiorentina	Italy	No	38	65	
9	Bayern	Germany	Yes	34	90	

A1:E5

Formula and Functions

C8	A	B	C	D
1				
2				
3				
4			Marks	
5			89	
6			93	
7				
8	Formula		182	
9				

C7	A	B	C
1			
2			
3			
4			Marks
5			89
6			93
7	Function		182

- ❖ Formula:
 - A **formula** is an expression which calculates the value of a cell.

- ❖ Functions
 - **Functions** are predefined **formulas and** are already available in **Excel**.

Example : [click here for formula and function practical](#)

COUNT:

- ❖ Use the COUNT function to get the number of entries in a **number field**.

- ❖ Syntax:
`=Count(range)`
- ❖ Arguments:
 - **range** - The range of cells to count.

Example : [click here for count practical](#)

Let's use count for numbers entries.

COUNTIF:

- ❖ Use the COUNT function Only If a Condition is Satisfied
- ❖ Syntax
 - =COUNTIF(range, criteria)
- ❖ Arguments
 - **range** - The range of cells to count.
 - **criteria** - The criteria that controls which cells should be counted.

Example : [click here for countif practical](#)

COUNTIFS

- ❖ Use the COUNT function For more than one IF Conditions.
- ❖ Syntax
 - =COUNTIFS (range1, criteria1, [range2], [criteria2], ...)
- ❖ Arguments
 - **range1** - The first range to evaluate.
 - **criteria1** - The criteria to use on range1.
 - **range2** - [optional] The second range to evaluate.
 - **criteria2** - [optional] The criteria to use on range2.

Example : [click here for countifs practical](#)

SUM:

- ❖ The Microsoft **Excel SUM function** adds all numbers in a range of cells and returns the result
- ❖ Syntax
 - =SUM (number1, [number2], [number3], ...)
- ❖ Arguments
 - **number1** - The first value to sum.
 - **number2** - [optional] The second value to sum.
 - **number3** - [optional] The third value to sum.

Example : [click here for sum practical](#)

SUMIF

- ❖ Adds all numbers in a range of cells If Condition is True and returns the result.
- ❖ Syntax
 - =SUMIF (range, criteria, [sum_range])
- ❖ Arguments
 - **range** - The range of cells that you want to apply the criteria against.
 - **criteria** - The criteria used to determine which cells to add.
 - **sum_range** - [optional] The cells to add together. If sum_range is omitted, the cells in range are added together instead.

Example : [click here for sumif practical](#)

AVERAGE

Returns the Average (Arithmetic Mean) of the Arguments.

❖ Syntax

- =AVERAGE (number1, [number2], ...)

❖ Arguments

- **number1** - A number or cell reference that refers to numeric values.
- **number2** - [optional] A number or cell reference that refers to numeric values.

Example : [click here for average practical](#)

AVERAGEIF

RETURNS THE AVERAGE (**ARITHMETIC MEAN**) OF THE ARGUMENTS IF A CONDITION IS TRUE

❖ Syntax

- `=AVERAGEIF (range, criteria, [average_range])`

❖ Arguments

- **range** - One or more cells, including numbers or names, arrays, or references.
- **criteria** - A number, expression, cell reference, or text.
- **average_range** - [optional] The cells to average. When omitted, range is used.

Example : [click here for averageif practical](#)

AVERAGEIFS

❖ Syntax

- =AVERAGEIFS (**average_range**, range1, criteria1, [range2], [criteria2], ...)

❖ Arguments

- average_range**- The range to average.
- range1** - The first range to evaluate.
- criteria1** - The criteria to use on range1.
- range2** - [optional] The second range to evaluate.
- criteria2** - [optional] The criteria to use on range2.

Example : [click here for averageifs practical](#)

IFERROR

The Microsoft Excel IFERROR function returns an alternate **value** if a formula results in an error.

❖ Syntax

- =IFERROR (value, value_if_error)

❖ Arguments

- **value** - The value, reference, or formula to check for an error.
- **value_if_error** - The value to return if an error is found.

Example : [click here for iferror practical](#)

LOOKUP

The Excel LOOKUP function performs an approximate match lookup in a one-column or one-row **range**, and returns the corresponding **value** from another one-column or one-row **range**.

- ❖ VLOOKUP
- ❖ HLOOKUP

VLOOKUP

❖ Syntax

- =VLOOKUP (value, table, col_index, [range_lookup])

❖ Arguments

- **value** - The value to look for in the first column of a table.
- **table** - The table from which to retrieve a value.
- **col_index** - The column in the table from which to retrieve a value.
- **range_lookup** - [optional] TRUE = approximate match (default).
FALSE = exact match.

Example : [click here for vlookup practical](#)

HLOOKUP

This performs a horizontal lookup by searching for a value in the top row of the *table* and returning the value in the same column based on the *index-number*

❖ Syntax

=HLOOKUP (value, table, row_index, [range_lookup])

❖ Arguments

- value** - The value to look up.
- table** - The table from which to retrieve data.
- row_index** - The row number from which to retrieve data.
- range_lookup** - [optional] A boolean to indicate exact match or approximate match. Default = TRUE = approximate match.

Example : [click here for hlookup practical](#)

Pivot Table

- ❖ **Pivot tables** are one of Excel's most powerful features.
- ❖ A **pivot table** is a **table** of statistics that summarizes the data of a more extensive **table**
- ❖ **Pivot tables** are **used** to summarize, sort, reorganize, group, count, total or average data stored in a database. It allows its users to transform columns into rows and rows into columns.

TO CREATE A PIVOT TABLE:

- SELECT THE **TABLE** OR CELLS (INCLUDING COLUMN HEADERS) CONTAINING THE DATA YOU WANT TO USE.
- FROM THE INSERT TAB, CLICK THE **PIVOT TABLE** COMMAND.
- THE **CREATE PIVOT TABLE** DIALOG BOX WILL APPEAR.

Practice

Pivot Data Link: [Click Here](#)

Pivot Solution Link: [Click Here](#)

Charts

A **chart** is a graphical representation of data.

A **chart** is often called a **graph**, helps to **visualize** data.

EXCEL CHARTS - TYPES

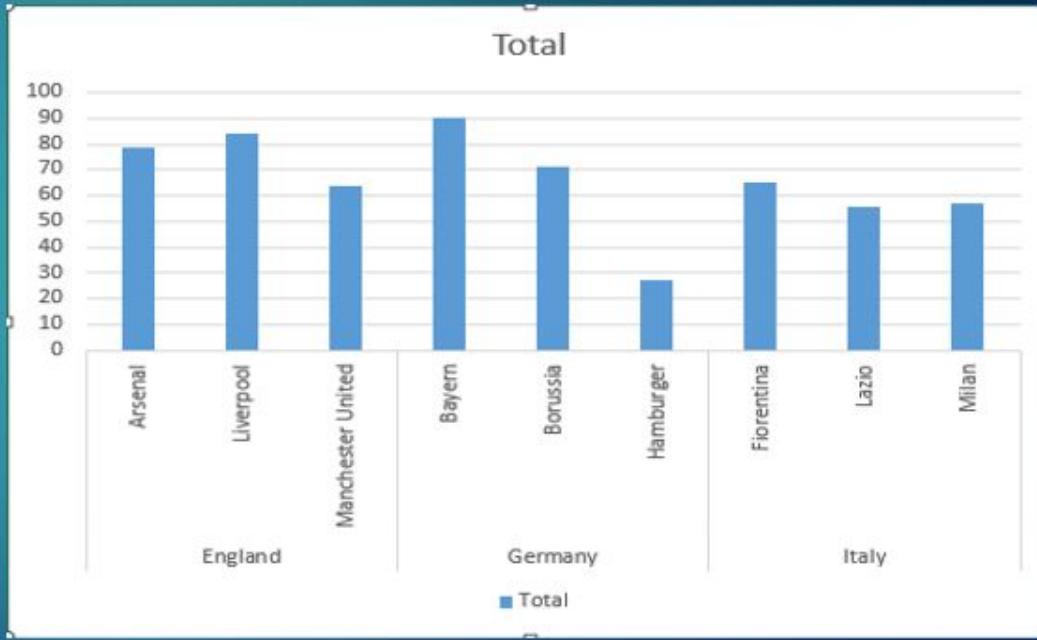
- **COLUMN** CHART
- **LINE** CHART
- **PIE** CHART
- **BAR** CHART
- **AREA** CHART
- XY (**SCATTER**) CHART

Column Chart

Column charts are used to compare values across categories by using vertical bars.

To create a column chart, follow these steps:

1. Enter data in a spreadsheet.
2. Select the data.
3. Click Insert > Insert **Column** or **Bar Chart** icon, and select a **column chart** option of your choice.



Example : [click here for column chart practical](#)

Bar Chart

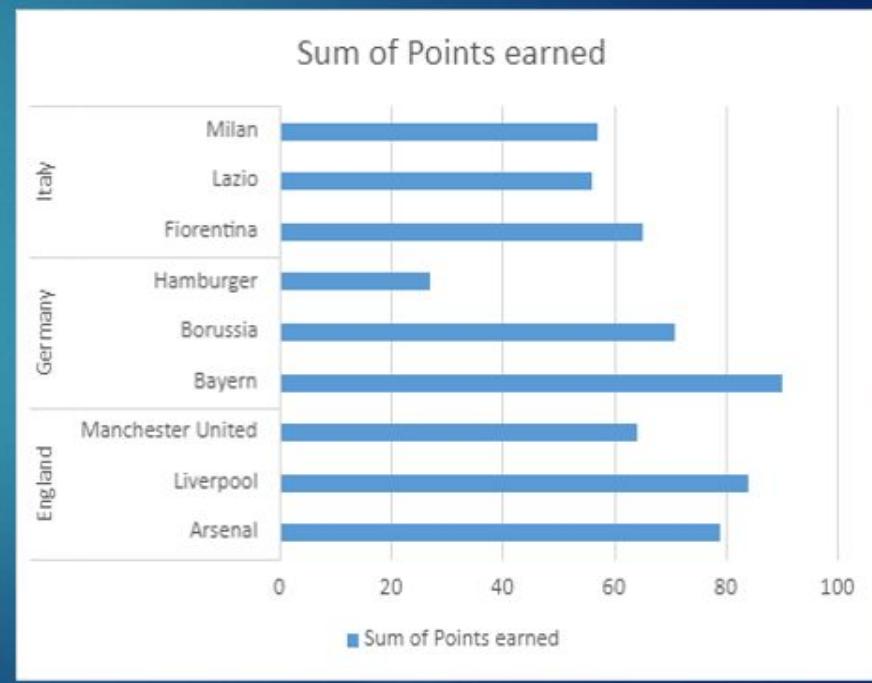
A bar chart is the horizontal version of a column chart. Use a bar chart if you have large text labels.

To create a Bar chart, follow these steps:

1. Select all the data that you want included in the **bar chart**.

2. Be sure to include the column and row headers, which will become the labels in the **bar chart**.

3. Click on the Insert tab and then on Insert Column or Bar Chart button in the **Charts** group



[Example : click here for bar chart practical](#)

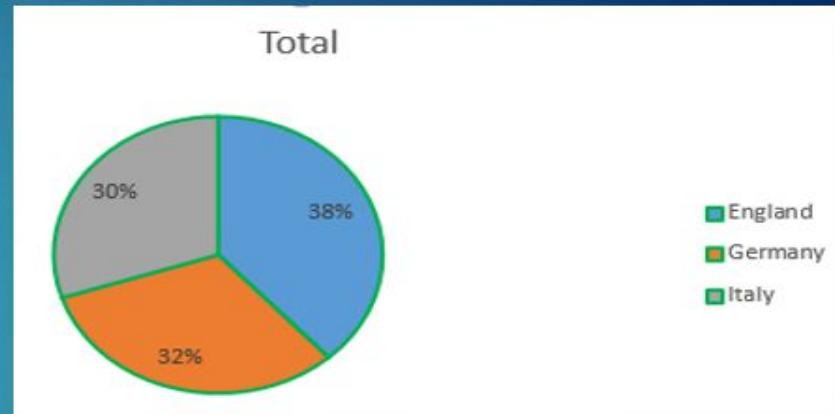
Pie Chart

Pie charts are used to display the contribution of each value (slice) to a total (pie). Pie charts always use one data series.

To create a **Pie chart**, follow these steps:

1. In your spreadsheet, select the data to use for your **pie chart**.

2. Click Insert > Insert **Pie** or Doughnut **Chart**, and then pick the **chart** you want.



Example : [click here for pie chart practical](#)

Scatter Chart

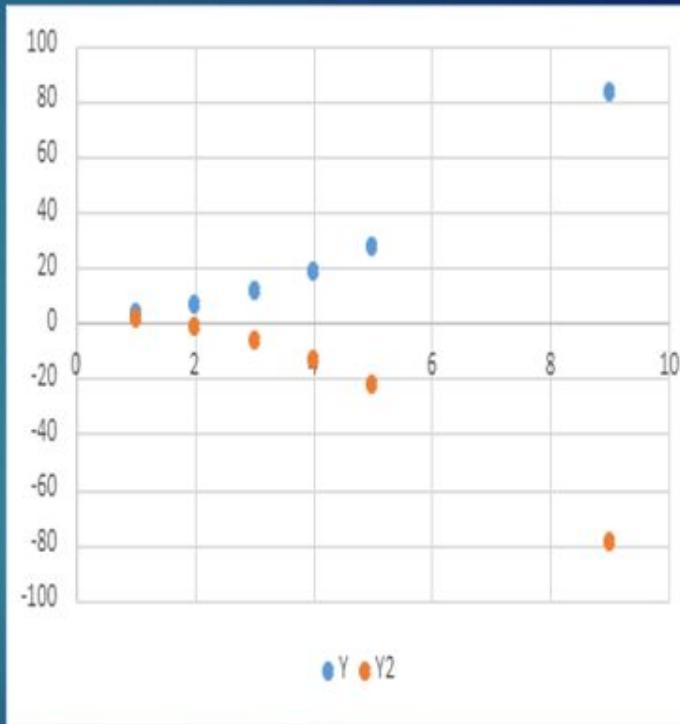
Scatter plots are often used to find out if there's a relationship between two variables suppose X and Y.

A scatterplot is used to represent a correlation between two **variables**.

There are two types of correlations: positive and negative.

Variables that are positively correlated move in the same direction, while **variables** that are negatively correlated move in opposite directions.

Example : [click here for scatter chart practical](#)

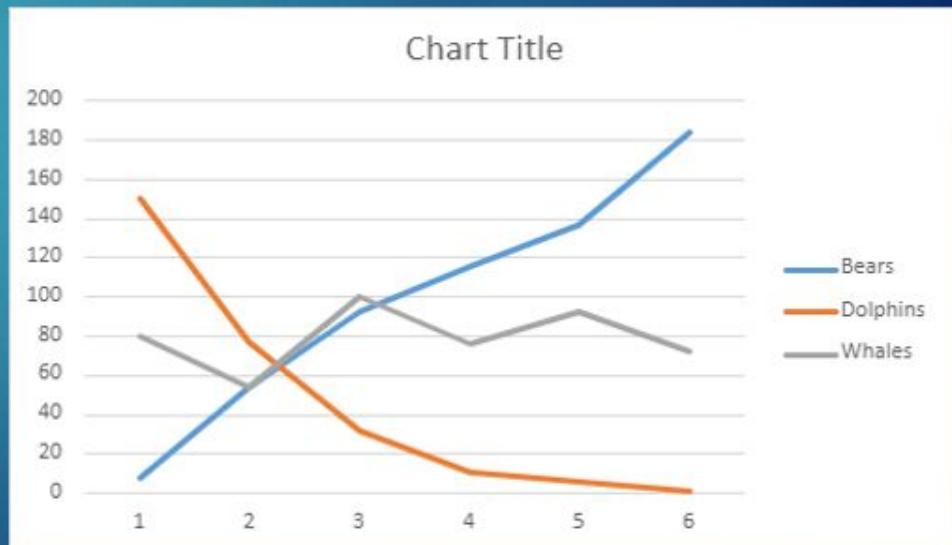


Line Charts

A **line chart** is often **used to** visualize a trend in data over intervals of time – a time series.

Hence **Line graphs** can be used to show how information or data change over time for ex: Trends.

Example : [click here for line chart practical](#)

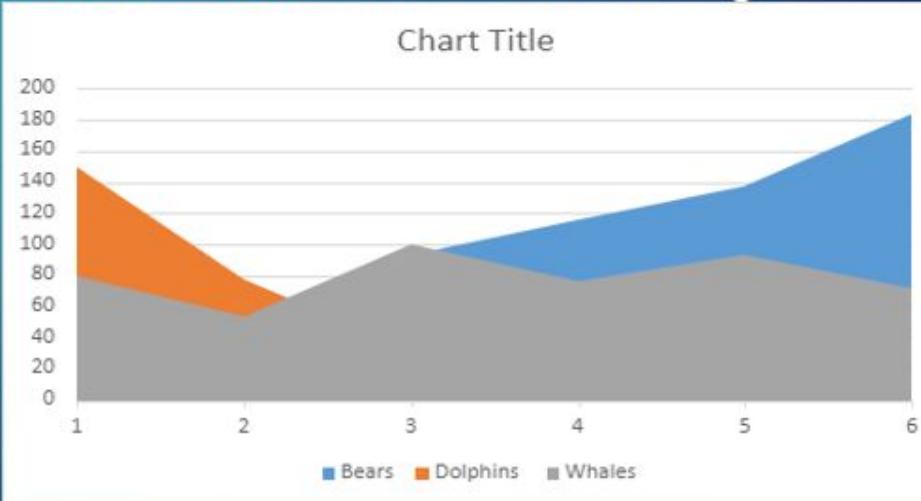


Area Chart

- ❖ An **area chart** represents the change in a one or more quantities over time.
- ❖ It is made by plotting a series of data points over time, connecting those data points with line segments
- ❖ Then filling in the **area** between the line and the x-axis with color or shading.

It can be seen as the Area chart is a Combination of the **Bar Chart** and **Line Chart**

Example : [click here for area chart practical](#)

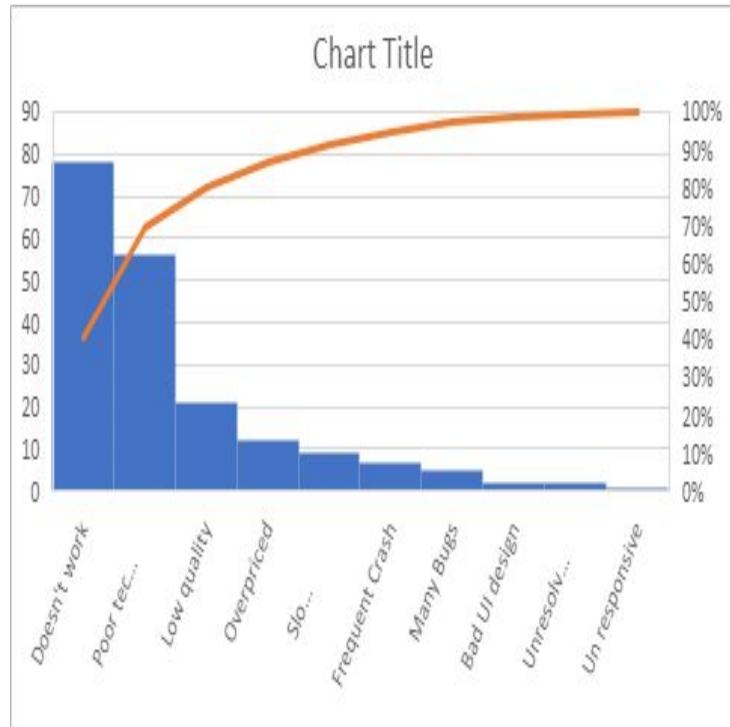


Pareto charts highlight the biggest factors in a set of data.

Following the idea of 80/20 analysis, they try to show which (approximately) 20% of the categories contribute 80% of the data being measured.

Pros [Example : click here for average practical](#)

- Quickly highlights most important data
- Excel automatically builds histogram and adds Pareto line



Statistics

Modules:

- Introduction
- Population
- Sample
- Statistic and Parameter
- Introduction of Descriptive Statistics
- Measures of Central Tendency
- Measures of Spread
- Introduction of Probability
- Terminology
- Types of Probability
- Probability Distribution
- Introduction to Inferential Statistics
- Estimation and errors
- Point estimation
- Confidence Interval
- Hypothesis and its types

What is Statistics?

Statistics is a mathematical science including methods of collecting, organizing and analyzing data in such a way that meaningful conclusions can be drawn from them. In general, its investigations and analyses fall into two broad categories called descriptive and inferential statistics.

Descriptive statistics deals with the processing of data without attempting to draw any inferences from it. The data are presented in the form of tables and graphs. The characteristics of the data are described in simple terms. Events that are dealt with include everyday happenings such as accidents, prices of goods, business, incomes, epidemics, sports data, population data.

Inferential statistics is a scientific discipline that uses mathematical tools to make forecasts and projections by analyzing the given data. This is of use to people employed in such fields as engineering, economics, biology, the social sciences, business, agriculture and communications.

Introduction to Population and Sample

A **population** often consists of a large group of specifically defined elements. For example, the population of a specific country means all the people living within the boundaries of that country.

Usually, it is not possible or practical to measure data for every element of the population under study. We randomly select a small group of elements from the population and call it a **sample**. Inferences about the population are then made on the basis of several samples.

Example:

- A company is thinking about buying 50,000 electric batteries from a manufacturer. It will buy the batteries if no more than 1% of the batteries are defective. It is not possible to test each battery in the population of 50,000 batteries since it takes time and costs money. Instead, it will select few samples of 500 batteries each and test them for defects. The results of these tests will then be used to estimate the percentage of defective batteries in the population.

Quantitative Data and Qualitative Data

Data is quantitative if the observations or measurements made on a given variable of a sample or population have numerical values.

Example: height, weight, number of children, blood pressure, current, voltage.

Data is qualitative if words, groups and categories represents the observations or measurements.

Example: colors, yes-no answers, blood group.

Quantitative data is discrete if the corresponding data values take discrete values and it is continuous if the data values take continuous values.

Example of discrete data: number of children, number of cars.

Example of continuous data: speed, distance, time, pressure.

Statistic and Parameter

Many have trouble understanding the difference between parameter and statistic, but it's important to know what exactly these measures mean and how to distinguish them.

Parameter vs statistic – both are similar, yet different measures. The first one describes the whole population, while the second describes a part of the population.

What is Parameter?

It is a measure of a characteristic of an entire population (a mass of all units under consideration that share common characteristics) based on all the elements within that population. For example, all people living in one city, all-male teenagers in the world, all elements in a shopping trolley, or all students in a classroom.

If you ask all employees in a factory what kind of lunch they prefer and half of them say pasta, you get a parameter here – 50% of the employees like pasta for lunch. On the other hand, it's impossible to count how many men in the whole world like pasta for lunch, since you can't ask all of them about their choice. In that case, you'd probably survey just a representative sample (a portion) of them and extrapolate the answer to the entire population of men. This brings us to the other measure called statistic.

It's a measure of characteristic saying something about a fraction (a sample) of the population under study. A sample in statistics is a part or portion of a population. The goal is to estimate a certain population parameter. You can draw multiple samples from a given population, and the statistic (the result) acquired from different samples will vary, depending on the samples. So, using data about a sample or portion allows you to estimate the characteristics of an entire population.

Parameter vs Statistics

Can you tell the difference between statistics and parameters now?

- A parameter is a fixed measure describing the whole population (population being a group of people, things, animals, phenomena that share common characteristics.) A statistic is a characteristic of a sample, a portion of the target population.
- A parameter is fixed, unknown numerical value, while the statistic is a known number and a variable which depends on the portion of the population.
- Sample statistic and population parameters have different statistical notations:

In population parameter, population proportion is represented by P , mean is represented by μ (Greek letter mu), σ^2 represents variance, N represents population size, σ (Greek letter sigma) represents standard deviation, $\sigma_{\bar{x}}$ represents Standard error of mean, σ/μ represents Coefficient of variation, $(X-\mu)/\sigma$ represents standardized variate (z), and σ_p represents standard error of proportion.

In sample statistics, mean is represented by \bar{x} (x-bar), sample proportion is represented by \hat{p} (p-hat), s represents standard deviation, s^2 represents variance, sample size is represented by n , $s_{\bar{x}}$ represents Standard error of mean, s_p represents standard error of proportion, $s/(\bar{x})$ represents Coefficient of variation, and $(x-\bar{x})/s$ represents standardized variate (z).

Example of Parameters

- 20% of U.S. senators voted for a specific measure. Since there are only 100 senators, you can count what each of them voted.

Example of Statistic

- 50% of people living in the U.S. agree with the latest health care proposal. Researchers can't ask hundreds of millions of people if they agree, so they take samples, or part of the population and calculate the rest.

What Are The Differences Between Population Parameters and Sample Statistics?

The average weight of adult men in the U.S. is a parameter with an exact value – but, we don't know it.

Standard deviation and population mean are two common parameters.

A statistic is a characteristic of a group of population, or sample. You get sample statistics when you collect a sample and calculate the standard deviation and the mean. You can use sample statistics to make certain conclusions about an entire population thanks to inferential statistics. But, you need particular sampling techniques to draw valid conclusions. Using these techniques ensures that samples deliver unbiased estimates – correct on average. When it comes to based estimates, they are systematically too low or too high, so you don't need them.

To estimate population parameters in inferential statistics, you use sample statistics. For instance, if you collect a random sample of female teenagers in the U.S. and measure their weights, you can calculate the sample mean. You can use the sample mean as an unbiased estimate of the population mean.

Introduction of Descriptive Statistics

Descriptive statistics is the term given to the analysis of data that helps describe, show or summarize data in a meaningful way such that, for example, patterns might emerge from the data. Descriptive statistics do not, however, allow us to make conclusions beyond the data we have analysed or reach conclusions regarding any hypotheses we might have made. They are simply a way to describe our data.

Descriptive statistics are very important because if we simply presented our raw data it would be hard to visualize what the data was showing, especially if there was a lot of it. Descriptive statistics therefore enables us to present the data in a more meaningful way, which allows simpler interpretation of the data. For example, if we had the results of 100 pieces of students' coursework, we may be interested in the overall performance of those students. We would also be interested in the distribution or spread of the marks. Descriptive statistics allow us to do this. How to properly describe data through statistics and graphs is an important topic and discussed in other Laerd Statistics guides. Typically, there are two general types of statistic that are used to describe data

Introduction of Descriptive Statistics

- **Measures of central tendency:** these are ways of describing the central position of a frequency distribution for a group of data. In this case, the frequency distribution is simply the distribution and pattern of marks scored by the 100 students from the lowest to the highest. We can describe this central position using a number of statistics, including the mode, median, and mean.
- **Measures of spread:** these are ways of summarizing a group of data by describing how spread out the scores are. For example, the mean score of our 100 students may be 65 out of 100. However, not all students will have scored 65 marks. Rather, their scores will be spread out. Some will be lower and others higher. Measures of spread help us to summarize how spread out these scores are. To describe this spread, a number of statistics are available to us, including the range, quartiles, absolute deviation, variance and standard deviation.
- When we use descriptive statistics it is useful to summarize our group of data using a combination of tabulated description (i.e., tables), graphical description (i.e., graphs and charts) and statistical commentary (i.e., a discussion of the results).

Measures of Central Tendency

A **measure of central tendency** (also referred to as **measures of centre** or **central location**) is a summary measure that attempts to describe a whole set of data with a single value that represents the middle or centre of its distribution.

There are three main measures of central tendency:

- Mode
- Median
- Mean

Each of these measures describes a different indication of the typical or central value in the distribution.

Mode

The mode is the *most frequent occurring value* in a distribution.

Consider this dataset showing the retirement age of 11 people, in whole years:

54, 54, 54, 55, 56, 57, 57, 58, 58, 60, 60

This table shows a simple frequency distribution of the retirement age data.

Age	Frequency
54	3
55	1
56	1
57	2
58	2
60	2

Total Frequency: 11

The most commonly occurring value is 54, therefore the **mode** of this distribution is **54 years**.

Advantage of the mode:

The mode has an advantage over the median and the mean as it can be found for both numerical and categorical (non-numerical) data.

Limitations of the mode:

There are some limitations to using the mode. In some distributions, the mode may not reflect the centre of the distribution very well. When the distribution of retirement age is ordered from lowest to highest value, it is easy to see that the centre of the distribution is 57 years, but the mode is lower, at 54 years.

54, 54, 54, 55, 56, **57**, 57, 58, 58, 60, 60

It is also possible for there to be more than one mode for the same distribution of data, (bi-modal, or multi-modal). The presence of more than one mode can limit the ability of the mode in describing the centre or typical value of the distribution because a single value to describe the centre cannot be identified.

In some cases, particularly where the data are continuous, the distribution may have no mode at all (i.e. if all values are different).

In cases such as these, it may be better to consider using the median or mean, or group the data in to appropriate intervals, and find the modal class.

Median

The median is the *middle value* in distribution when the values are arranged in ascending or descending order.

The median divides the distribution in half (there are 50% of observations on either side of the median value). In a distribution with an odd number of observations, the median value is the middle value.

Median

Looking at the retirement age distribution (which has 11 observations), the median is the middle value, which is 57 years:

54, 54, 54, 55, 56, **57**, 57, 58, 58, 60, 60

When the distribution has an even number of observations, the median value is the mean of the two middle values. In the following distribution, the two middle values are 56 and 57, therefore the median equals **56.5 years**:

52, 54, 54, 54, 55, **56, 57**, 57, 58, 58, 60, 60

Advantage of the median:

The median is less affected by **outliers** and **skewed** data than the mean, and is usually the preferred measure of central tendency when the distribution is not symmetrical.

Limitations of the median:

The median cannot be identified for categorical nominal data, as it cannot be logically ordered.

Mean

The mean is the sum of the value of each observation in a dataset divided by the number of observations. This is also known as the arithmetic average.

Looking at the retirement age distribution again:

54, 54, 54, 55, 56, 57, 57, 58, 58, 60, 60

The mean is calculated by adding together all the values
 $(54+54+54+55+56+57+57+58+58+60+60 = 623)$ and dividing by the number of observations
(11) which equals 56.6 years.

Advantage of the mean:

The mean can be used for both continuous and discrete numeric data.

Limitations of the mean:

The mean cannot be calculated for categorical data, as the values cannot be summed.

As the mean includes every value in the distribution the mean is influenced by outliers and skewed distributions.

What else do I need to know about the mean?

The population mean is indicated by the Greek symbol μ (pronounced ‘mu’). When the mean is calculated on a distribution from a sample it is indicated by the symbol \bar{x} (pronounced X-bar).

Measures of Spread

Measures of spread describe how similar or varied the set of observed values are for a particular variable (data item). Measures of spread include the range, quartiles and the interquartile range, variance and standard deviation.

There are three main measures of spread:

- Range
- Quartiles
- Standard Deviation
- Variance

When can we measure spread?

The spread of the values can be measured for quantitative data, as the variables are numeric and can be arranged into a logical order with a low end value and a high end value.

Why do we measure spread?

Summarising the dataset can help us understand the data, especially when the dataset is large. As discussed in the Measures of Central Tendency , the mode, median, and mean summarise the data into a single value that is typical or representative of all the values in the dataset, but this is only part of the 'picture' that summarises a dataset. Measures of spread summarise the data in a way that shows how scattered the values are and how much they differ from the mean value.

Example

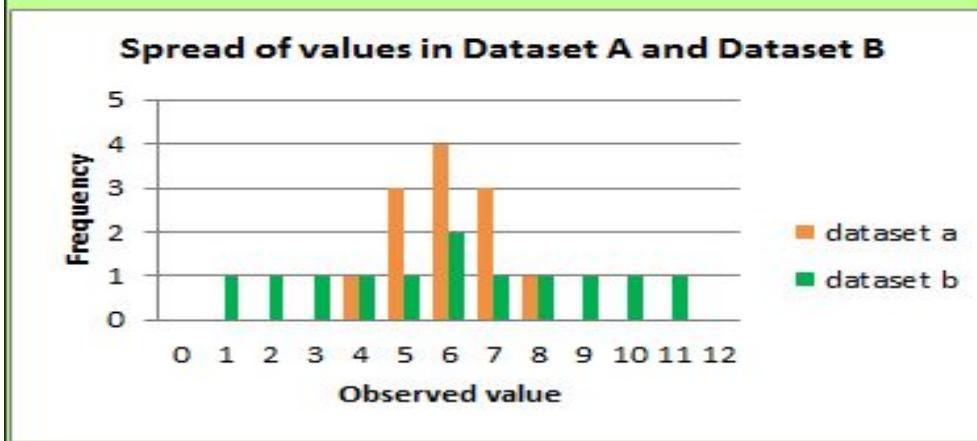
For example:

Dataset A	Dataset B
4, 5, 5, 5, 6, 6, 6, 6, 7, 7, 7, 8	1, 2, 3, 4, 5, 6, 6, 7, 8, 9, 10, 11

The mode (most frequent value), median (middle value*) and mean (arithmetic average) of both datasets is 6.
(*note, the median of an even numbered data set is calculated by taking the mean of the middle two observations).

If we just looked at the measures of central tendency, we may assume that the datasets are the same.

However, if we look at the spread of the values in the following graph, we can see that Dataset B is more dispersed than Dataset A. Used together, the measures of central tendency and measures of spread help us to better understand the data



Range

The range is the difference between the smallest value and the largest value in a dataset.

Example

Calculating the Range

Dataset A

4, 5, 5, 5, 6, 6, 6, 6, 7, 7, 7, 8

The range is 4, the difference between the highest value (8) and the lowest value (4).

Dataset B

1, 2, 3, 4, 5, 6, 6, 7, 8, 9, 10, 11

The range is 10, the difference between the highest value (11) and the lowest value (1).

Dataset A

0	1	2	3	4	5	6	7	8	9	10	11	12	13
---	---	---	---	---	---	---	---	---	---	----	----	----	----

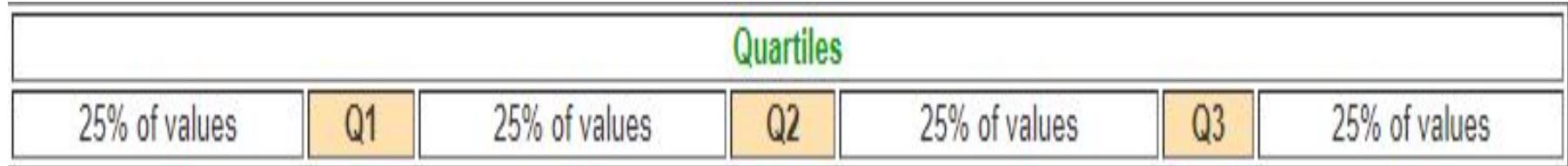
Dataset B

0	1	2	3	4	5	6	7	8	9	10	11	12	13
---	---	---	---	---	---	---	---	---	---	----	----	----	----

On a number line, you can see that the range of values for Dataset B is larger than Dataset A.

Quartiles

Quartiles divide an ordered dataset into four equal parts, and refer to the values of the point *between* the quarters. A dataset may also be divided into quintiles (five equal parts) or deciles (ten equal parts).



The **lower quartile (Q1)** is the point between the lowest 25% of values and the highest 75% of values. It is also called the **25th percentile**.

The **second quartile (Q2)** is the middle of the data set. It is also called the **50th percentile**, or the **median**.

The **upper quartile (Q3)** is the point between the lowest 75% and highest 25% of values. It is also called the **75th percentile**.

Example

Calculating Quartiles

Dataset A

4	5	5	Q1	5	6	6	Q2	6	6	7	Q3	7	7	8
---	---	---	----	---	---	---	----	---	---	---	----	---	---	---

As the quartile point falls between two values, the mean (average) of those values is the quartile value:

$$Q1 = (5+5) / 2 = 5$$

$$Q2 = (6+6) / 2 = 6$$

$$Q3 = (7+7) / 2 = 7$$

Dataset B

1	2	3	Q1	4	5	6	Q2	6	7	8	Q3	9	10	11
---	---	---	----	---	---	---	----	---	---	---	----	---	----	----

As the quartile point falls between two values, the mean (average) of those values is the quartile value:

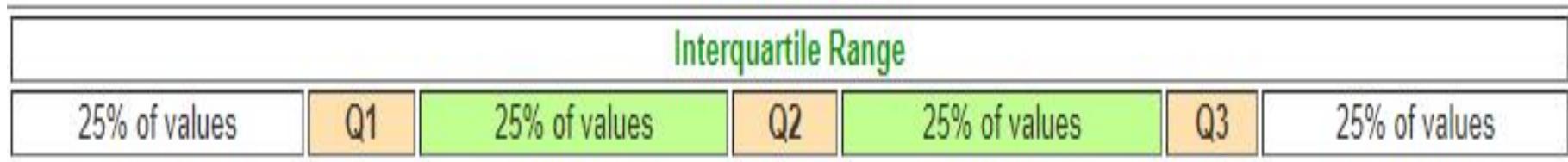
$$Q1 = (3+4) / 2 = 3.5$$

$$Q2 = (6+6) / 2 = 6$$

$$Q3 = (8+9) / 2 = 8.5$$

Interquartile Range(IQR)

The interquartile range (IQR) is the difference between the upper (Q3) and lower (Q1) quartiles, and describes the middle 50% of values when ordered from lowest to highest. The IQR is often seen as a better measure of spread than the range as it is not affected by outliers.



Introduction of Probability

How **likely** something is to happen.

Many events can't be predicted with total certainty. The best we can say is how **likely** they are to happen, using the idea of probability.

Tossing a Coin

When a coin is tossed, there are two possible outcomes:

- **Heads(H) or**
- **Tails(T)**

We say that the probability of the coin landing **H** is $\frac{1}{2}$

And the probability of the coin landing **T** is $\frac{1}{2}$



Introduction of Probability

Throwing Dice

When a single die is thrown, there are six possible outcomes: 1, 2, 3, 4, 5, 6.

The probability of any one of them is $\frac{1}{6}$



Probability

In general:

$$\text{Probability of an event happening} = \frac{\text{Number of ways it can happen}}{\text{Total number of outcomes}}$$

Example

- The chances of rolling a "4" with a die

Number of ways it can happen: 1 (there is only 1 face with a "4" on it)

Total number of outcomes: 6 (there are 6 faces altogether)

$$\text{So the probability} = \frac{1}{6}$$

- There are 5 marbles in a bag: 4 are blue, and 1 is red. What is the probability that a blue marble gets picked?

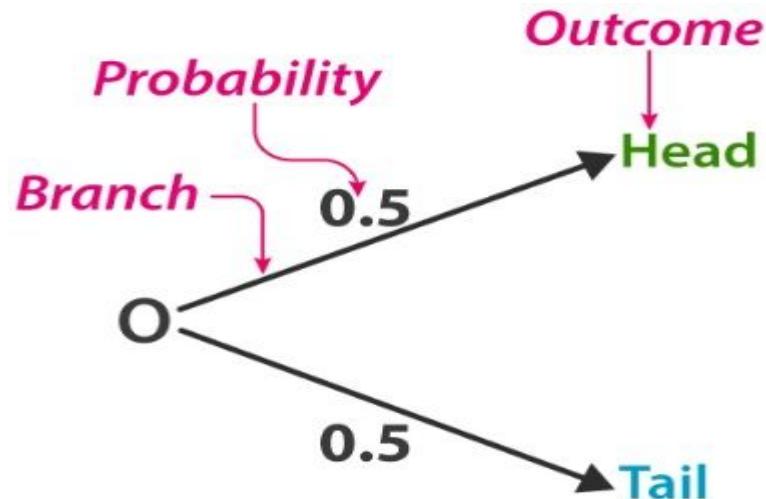
Number of ways it can happen: 4 (there are 4 blues)

Total number of outcomes: 5 (there are 5 marbles in total)

$$\text{So the probability} = \frac{4}{5} = 0.8$$

Probability Tree

The **tree diagram** helps to organize and visualize the different possible outcomes. Branches and ends of the tree are two main positions. Probability of each branch is written on the branch, whereas the ends are containing the final outcome. Tree diagram is used to figure out when to multiply and when to add. You can see below a tree diagram for the coin:



Types of Probability

There are two major types of probabilities:

- Theoretical Probability
- Experimental Probability

Theoretical Probability

Theoretical Probability is what is expected to happen based on mathematics.

$$P(\text{event}) = \frac{\text{number of favorable outcomes}}{\text{total number of possible outcomes}}$$

Example:

A coin is tossed.

$$P(\text{head}) = \frac{1}{2}$$

$$P(\text{tail}) = \frac{1}{2}$$

Experimental Probability

Experimental Probability is found by repeating an experiment and observing the outcomes.

$$P(\text{event}) = \frac{\text{number of times event occurs}}{\text{total number of trials}}$$

Example:

A Coin is tossed 10 times: A head is recorded 7 times and a tail 3 times.

$$P(\text{head}) = \frac{7}{10}$$

$$P(\text{tail}) = \frac{3}{10}$$

Probability Distribution

A probability distribution is a statistical function that describes all the possible values and likelihoods that a random variable can take within a given range. This range will be bounded between the minimum and maximum possible values, but precisely where the possible value is likely to be plotted on the probability distribution depends on a number of factors. These factors include the distribution's mean (average), standard deviation, skewness.

Types of Distributions:

1. Bernoulli Distribution
2. Uniform Distribution
3. Binomial Distribution
4. Normal Distribution
5. Poisson Distribution
6. Exponential Distribution

Bernoulli Distribution

All you cricket junkies out there! At the beginning of any cricket match, how do you decide who is going to bat or ball? A toss! It all depends on whether you win or lose the toss, right? Let's say if the toss results in a head, you win. Else, you lose. There's no midway.

A **Bernoulli distribution** has only two possible outcomes, namely 1 (success) and 0 (failure), and a single trial. So the random variable X which has a Bernoulli distribution can take value 1 with the probability of success, say p , and the value 0 with the probability of failure, say q or $1-p$.

the occurrence of a head denotes success, and the occurrence of a tail denotes failure.

Probability of getting a head = 0.5 = Probability of getting a tail since there are only two possible outcomes.

Bernoulli Distribution

The probability mass function is given by: $p^x(1-p)^{1-x}$ where $x \in (0, 1)$.

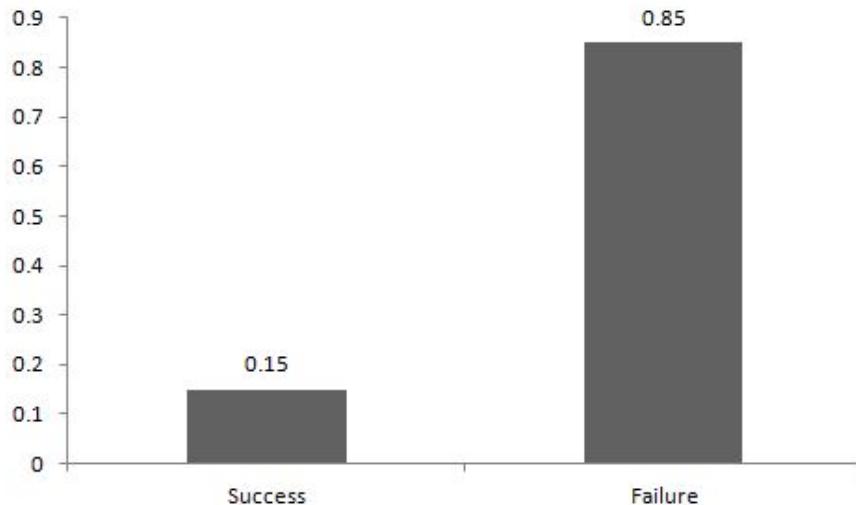
It can also be written as

$$P(x) = \begin{cases} 1 - p, & x = 0 \\ p, & x = 1 \end{cases}$$

The probabilities of success and failure need not be equally likely, like the result of a fight between me and Undertaker. He is pretty much certain to win. So in this case probability of my success is 0.15 while my failure is 0.85

Bernoulli Distribution

Here, the probability of success(p) is not same as the probability of failure. So, the chart below shows the Bernoulli Distribution of our fight.



Bernoulli Distribution

Here, the probability of success = 0.15 and probability of failure = 0.85. The expected value is exactly what it sounds. If I punch you, I may expect you to punch me back. Basically expected value of any distribution is the mean of the distribution. The expected value of a random variable X from a Bernoulli distribution is found as follows:

$$E(X) = 1*p + 0*(1-p) = p$$

The variance of a random variable from a bernoulli distribution is:

$$V(X) = E(X^2) - [E(X)]^2 = p - p^2 = p(1-p)$$

There are many examples of Bernoulli distribution such as whether it's going to rain tomorrow or not where rain denotes success and no rain denotes failure and Winning (success) or losing (failure) the game.

Binomial Distribution

Suppose that you won the toss today and this indicates a successful event. You toss again but you lost this time. If you win a toss today, this does not necessitate that you will win the toss tomorrow. Let's assign a random variable, say X , to the number of times you won the toss. What can be the possible value of X ? It can be any number depending on the number of times you tossed a coin.

There are only two possible outcomes. Head denoting success and tail denoting failure. Therefore, probability of getting a head = 0.5 and the probability of failure can be easily computed as: $q = 1 - p = 0.5$.

A distribution where only two outcomes are possible, such as success or failure, gain or loss, win or lose and where the probability of success and failure is same for all the trials is called a Binomial Distribution.

The outcomes need not be equally likely. Remember the example of a fight between me and Undertaker? So, if the probability of success in an experiment is 0.2 then the probability of failure can be easily computed as $q = 1 - 0.2 = 0.8$.

Binomial Distribution

Each trial is independent since the outcome of the previous toss doesn't determine or affect the outcome of the current toss. An experiment with only two possible outcomes repeated n number of times is called binomial. The parameters of a binomial distribution are n and p where n is the total number of trials and p is the probability of success in each trial.

On the basis of the above explanation, the properties of a Binomial Distribution are

1. Each trial is independent.
2. There are only two possible outcomes in a trial- either a success or a failure.
3. A total number of n identical trials are conducted.
4. The probability of success and failure is same for all trials. (Trials are identical.)

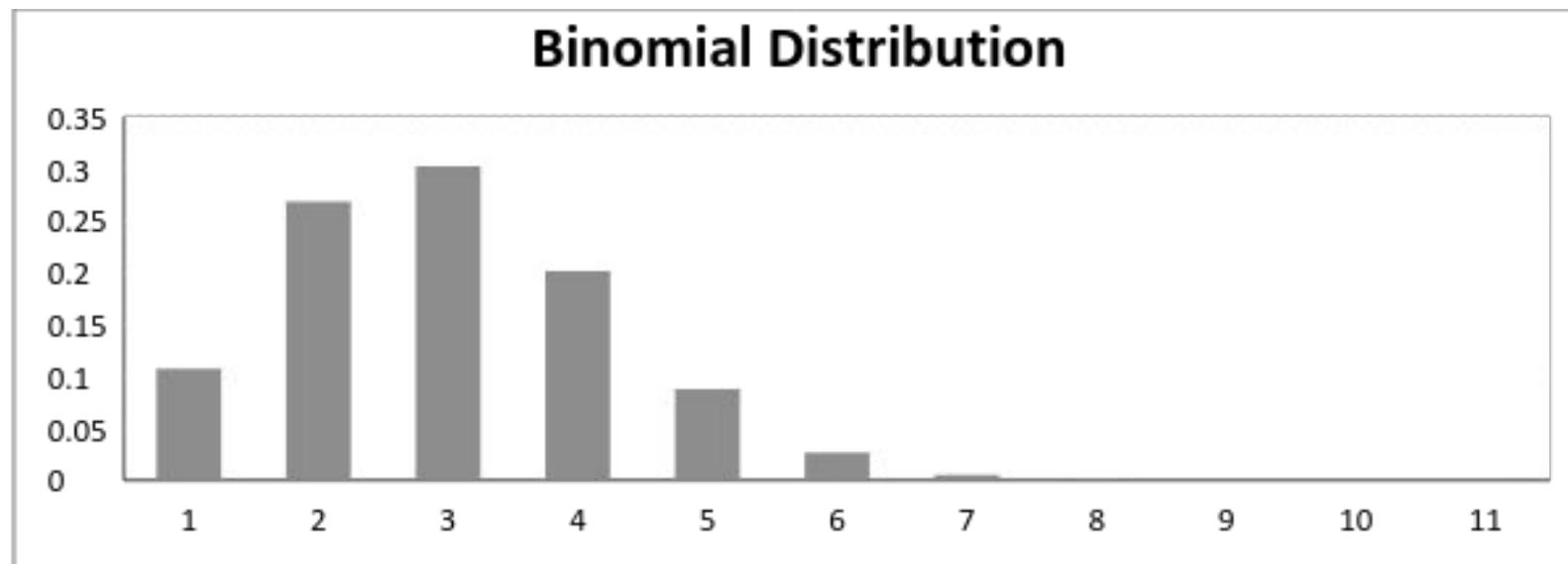
Binomial Distribution

The mathematical representation of binomial distribution is given by:

$$P(x) = \frac{n!}{(n-x)!x!} p^x q^{n-x}$$

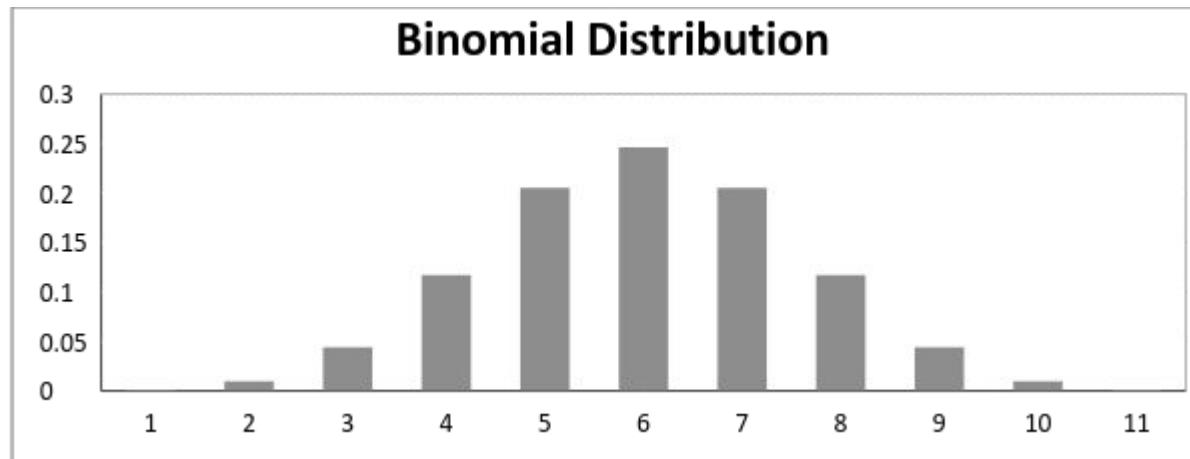
Binomial Distribution

A binomial distribution graph where the probability of success does not equal the probability of failure looks like



Binomial Distribution

Now, when probability of success = probability of failure, in such a situation the graph of binomial distribution looks like



Binomial Distribution

The mean and variance of a binomial distribution are given by:

$$\text{Mean} \rightarrow \mu = n \cdot p$$

$$\text{Variance} \rightarrow \text{Var}(X) = n \cdot p \cdot q$$

Geometric Distribution

The geometric distribution represents the number of failures before you get a success in a series of Bernoulli trials. This discrete probability distribution is represented by the probability density function.

$$f(x) = (1 - p)^{x-1} p = p * (1-p)^{x-1}$$

For example, you ask people outside a polling station who they voted for until you find someone that voted for the independent candidate in a local election. The geometric distribution would represent the number of people who you had to poll before you found someone who voted independent. You would need to get a certain number of failures before you got your first success.

Geometric Distribution

If you had to ask **3** people, then **X=3**; if you had to ask **4** people, then **X=4** and so on. In other words, there would be $X-1$ failures before you get your success.

If **X=n**, it means you succeeded on the n th try and failed for $n-1$ tries. The probability of failing on your first try is **1-p**. For example, if **p = 0.2** then your probability of success is **.2** and your probability of failure is **1 – 0.2 = 0.8**.

Independence (i.e. that the outcome of one trial does not affect the next) means that you can multiply the probabilities together. So the probability of failing on your second try is **(1-p)(1-p)** and your probability of failing on the $n-1$ tries is **(1-p)ⁿ⁻¹**. If you succeeded on your 4th try, **n = 4**, **n – 1 = 3**, so the probability of failing up to that point is **(1-p)(1-p)(1-p) = (1-p)³**.

Geometric Distribution

Example:-

If your probability of success is 0.2, what is the probability you meet an independent voter on your third try?

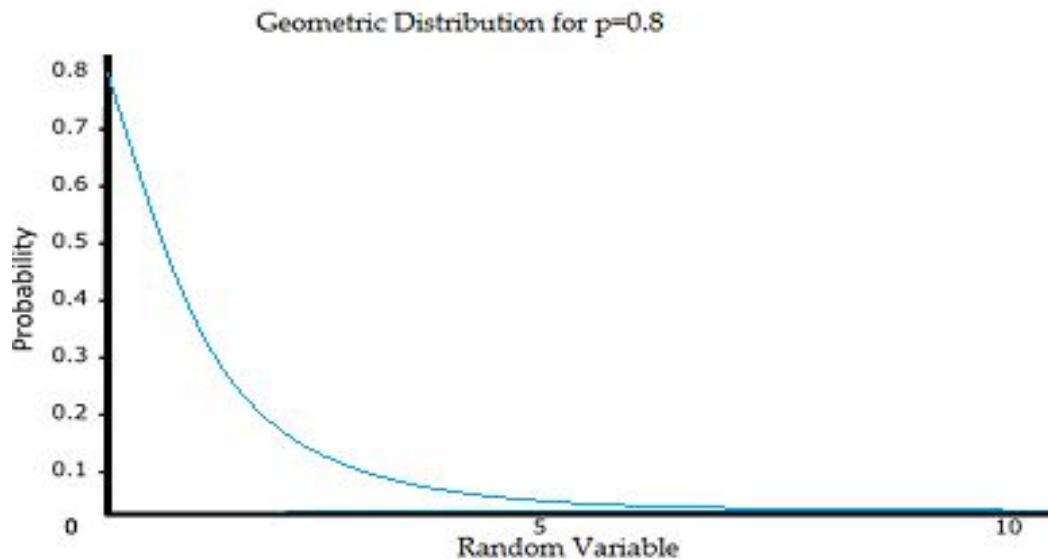
Inserting 0.2 as p and with X = 3, the probability density function becomes:

$$f(x) = (1 - p)^{x-1} * p$$

$$P(X=3) = (1 - 0.2)^3 - 1(0.2)$$

$$P(X=3) = (0.8)^2 * 0.2 = 0.128.$$

Geometric Distribution



Theoretically, there are an infinite number of geometric distributions. The value of any specific distribution depends on the value of the probability p .

Assumptions for the Geometric Distribution

The three assumptions are:

- There are two possible outcomes for each trial (success or failure).
- The trials are independent.
- The probability of success is the same for each trial.

Uniform Distribution

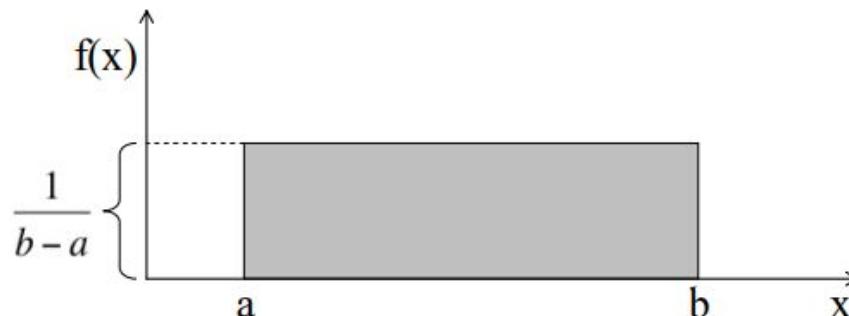
When you roll a fair die, the outcomes are 1 to 6. The probabilities of getting these outcomes are equally likely and that is the basis of a uniform distribution. Unlike Bernoulli Distribution, all the n number of possible outcomes of a uniform distribution are equally likely.

A variable X is said to be uniformly distributed if the density function is:

$$f(x) = \frac{1}{b-a}$$

for $-\infty < a \leq x \leq b < \infty$

The graph of a uniform distribution curve looks like



Uniform Distribution

You can see that the shape of the Uniform distribution curve is rectangular, the reason why Uniform distribution is called rectangular distribution.

For a Uniform Distribution, a and b are the parameters.

Uniform Distribution

The number of bouquets sold daily at a flower shop is uniformly distributed with a maximum of 40 and a minimum of 10.

Let's try calculating the probability that the daily sales will fall between 15 and 30.

The probability that daily sales will fall between 15 and 30 is $(30-15)*(1/(40-10)) = 0.5$

Similarly, the probability that daily sales are greater than 20 is $= 0.667$

The mean and variance of X following a uniform distribution is:

Mean -> $E(X) = (a+b)/2$

Variance -> $V(X) = (b-a)^2/12$

The standard uniform density has parameters $a = 0$ and $b = 1$, so the PDF for standard uniform density is given by:

$$f(x) = \begin{cases} 1, & 0 \leq x \leq 1 \\ 0, & \text{otherwise} \end{cases}$$

Exponential Distribution

Let's consider the call center example one more time. What about the interval of time between the calls ? Here, exponential distribution comes to our rescue. Exponential distribution models the interval of time between the calls.

Other examples are:

1. Length of time between metro arrivals,
2. Length of time between arrivals at a gas station
3. The life of an Air Conditioner

Exponential distribution is widely used for survival analysis. From the expected life of a machine to the expected life of a human, exponential distribution successfully delivers the result.

Exponential Distribution

A random variable X is said to have an **exponential distribution** with PDF:

$$f(x) = \{ \lambda e^{-\lambda x}, \quad x \geq 0$$

and parameter $\lambda > 0$ which is also called the rate.

For survival analysis, λ is called the failure rate of a device at any time t , given that it has survived up to t .

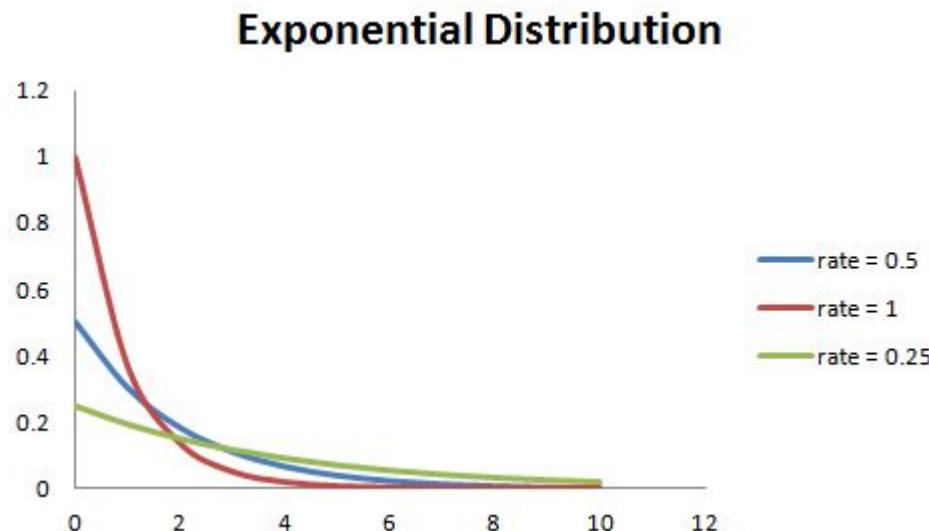
Mean and Variance of a random variable X following an exponential distribution:

$$\text{Mean} \rightarrow E(X) = 1/\lambda$$

$$\text{Variance} \rightarrow \text{Var}(X) = (1/\lambda)^2$$

Exponential Distribution

Also, the greater the rate, the faster the curve drops and the lower the rate, flatter the curve. This is explained better with the graph shown below.



Exponential Distribution

To ease the computation, there are some formulas given below.

$P\{X \leq x\} = 1 - e^{-\lambda x}$, corresponds to the area under the density curve to the left of x .

$P\{X > x\} = e^{-\lambda x}$, corresponds to the area under the density curve to the right of x .

$P\{x_1 < X \leq x_2\} = e^{-\lambda x_1} - e^{-\lambda x_2}$, corresponds to the area under the density curve between x_1 and x_2 .

Normal Distribution

Normal distribution represents the behavior of most of the situations in the universe (That is why it's called a “normal” distribution. I guess!). The large sum of (small) random variables often turns out to be normally distributed, contributing to its widespread application. Any distribution is known as Normal distribution if it has the following characteristics:

1. The mean, median and mode of the distribution coincide.
2. The curve of the distribution is bell-shaped and symmetrical about the line $x=\mu$.
3. The total area under the curve is 1.
4. Exactly half of the values are to the left of the center and the other half to the right.

A normal distribution is highly different from Binomial Distribution. However, if the number of trials approaches infinity then the shapes will be quite similar.

Normal Distribution

The PDF of a random variable X following a normal distribution is given by:

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2} \quad \text{for } -\infty < x < \infty.$$

The mean and variance of a random variable X which is said to be normally distributed is given by:

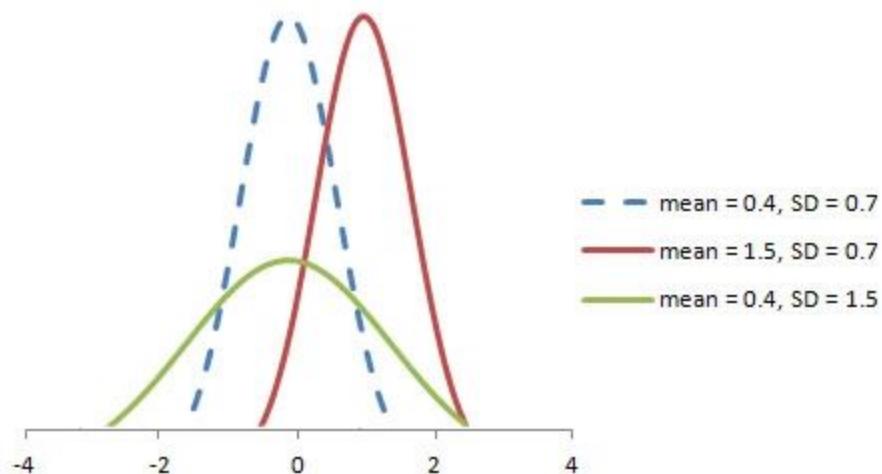
Mean -> $E(X) = \mu$

Variance -> $\text{Var}(X) = \sigma^2$

Normal Distribution

Here, μ (mean) and σ (standard deviation) are the parameters.

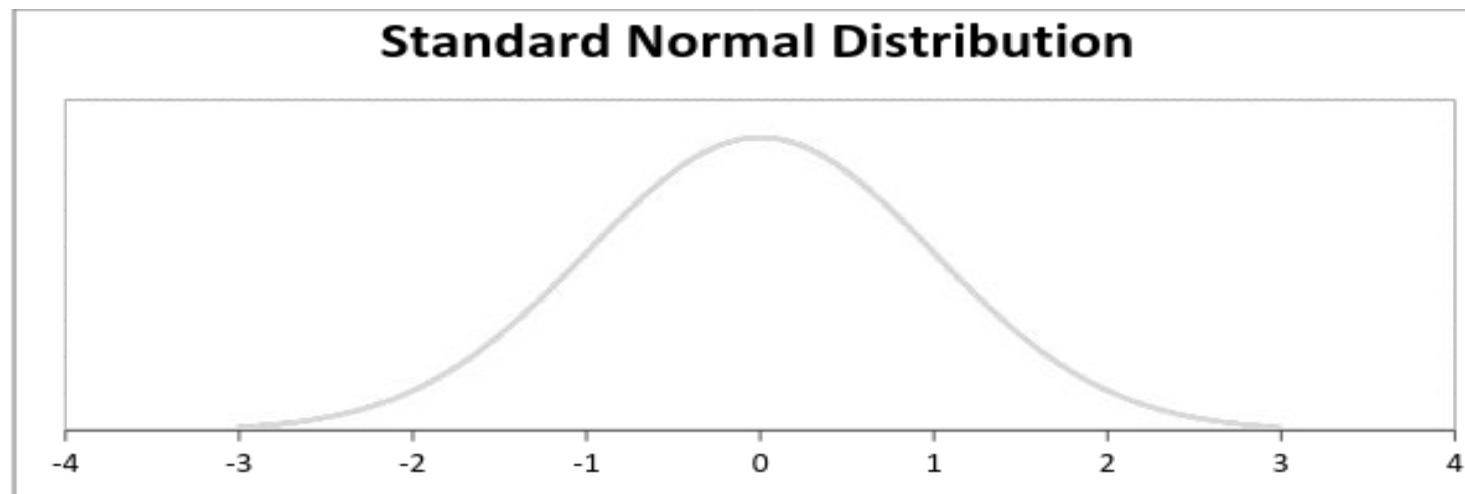
The graph of a random variable $X \sim N(\mu, \sigma)$ is shown below.



Normal Distribution

A standard normal distribution is defined as the distribution with mean 0 and standard deviation 1. For such a case, the PDF becomes:

$$f(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2} \quad \text{for } -\infty < x < \infty$$



Introduction of Inferential Statistics

It is about using data from sample and then making inferences about the larger population from which the sample is drawn. The goal of the inferential statistics is to draw conclusions from a sample and generalize them to the population. It determines the probability of the characteristics of the sample using probability theory. The most common methodologies used are hypothesis tests, Analysis of variance etc.

For example: Suppose we are interested in the exam marks of all the students in India. But it is not feasible to measure the exam marks of all the students in India. So now we will measure the marks of a smaller sample of students, for example 1000 students. This sample will now represent the large population of Indian students. We would consider this sample for our statistical study for studying the population from which it's deduced.



We evaluate 2 mutual exclusive statement on population data using sample data

Steps:

1. Make initial Assumption
2. Collecting data
3. Gather evidence to reject or not to reject the null Hypothesis .

Data Analytics and Machine Learning

Modules

- Introduction to Data Analytics
- Intro to types of Algorithms
- Regression
- Classification
- Dimensionality Reduction Techniques
- Clustering
- Model Selection and
- Hyperparameter Tuning
- Ensemble Learning
- Associate Rules Mining and
- Recommendations (with case study)
- Time Series Forecasting
- Reinforcement Learning

Introduction to Data Analytics

- 1. What is Data Analytics?**
- 2. Importance of Data Analytics**
- 3. Types of Data Analytics**

What is Data Analytics?

A domain focussed to gain meaningful and valuable insights from the data available to do predictions.



Importance of Data Analytics

- Helps businesses optimize their performances.
- Analysis is done to study purchase patterns.
- This can also help in improving managerial operations and leverage organisations to next level.
- Data and information are increasing rapidly.

Types of Data Analytics

Descriptive analytics describes what has happened over a given period of time. Have the number of views gone up?

Diagnostic analytics focuses more on why something happened. Did the weather affect beer sales? Did that latest marketing campaign impact sales?

Predictive analytics moves to what is likely going to happen in the near term. What happened to sales the last time we had a hot summer?

Prescriptive analytics suggests a course of action.

Descriptive analytics

- Descriptive analytics is the interpretation of historical data to draw a better Comparison.
- Takes raw data and parses that data to draw conclusions.
- Return on Investment Capital (ROIC) is a descriptive analytic created by taking three data points:

net income, dividends, and total capital, and turning those data points into an easy-to-understand percentage that can be used to compare one company's performance to others.

Diagnostic analytics

Diagnostic analytics is a form of advanced analytics that examines data or content to answer the question, “Why did it happen?”

It is characterized by techniques such as **data discovery, data mining and correlations.**

Predictive analytics

Predictive analytics is the use of **data**, statistical algorithms and machine learning techniques to identify the likelihood of future outcomes based on historical **data**.

Prediction

Forecasting, etc



Prescriptive analytics

- Prescriptive analytics makes use of machine learning to help businesses decide a course of action based on a computer program's predictions.
- Prescriptive analytics works with predictive analytics, which uses data to determine near-term outcomes

Intro to types of Algorithms

- **Supervised**
- **UnSupervised**
- **Reinforcement Learning**
- **Applications**

Supervised Machine Learning:

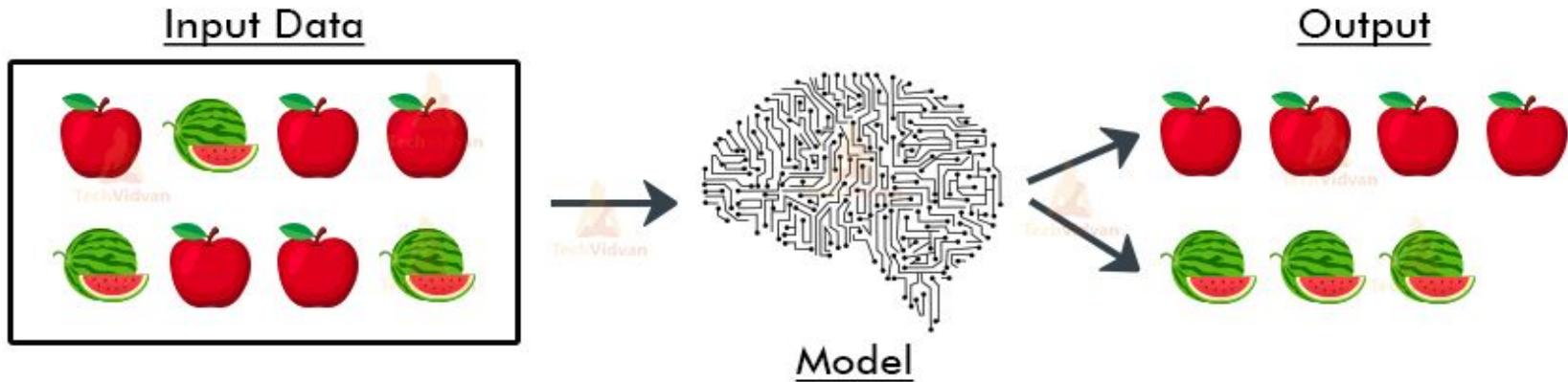
Supervised learning is when the model is getting trained on a labelled dataset.

User ID	Gender	Age	Salary	Purchased
15624510	Male	19	19000	0
15810944	Male	35	20000	1
15668575	Female	26	43000	0
15603246	Female	27	57000	0
15804002	Male	19	76000	1
15728773	Male	27	58000	1
15598044	Female	27	84000	0
15694829	Female	32	150000	1
15600575	Male	25	33000	1
15727311	Female	35	65000	0

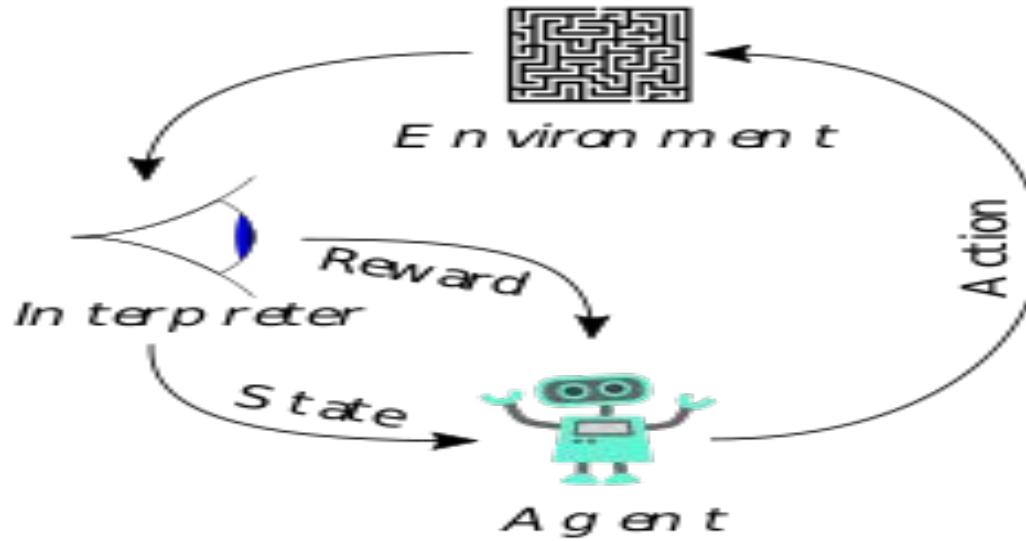
Un-Supervised Machine Learning:

Un-Supervised learning is when the model is getting trained on a Un-labelled dataset.

Unsupervised Learning in ML



Reinforcement Machine Learning:



Positive Reward for Good Action and Negative for Bad Action given a State in the Environment, then agent tries to Maximize the Reward points

Machine Learning

Regression

Regression Algorithms

- Linear Regression
- Ridge Regression
- Lasso Regression
- Polynomial Regression

Linear Regression

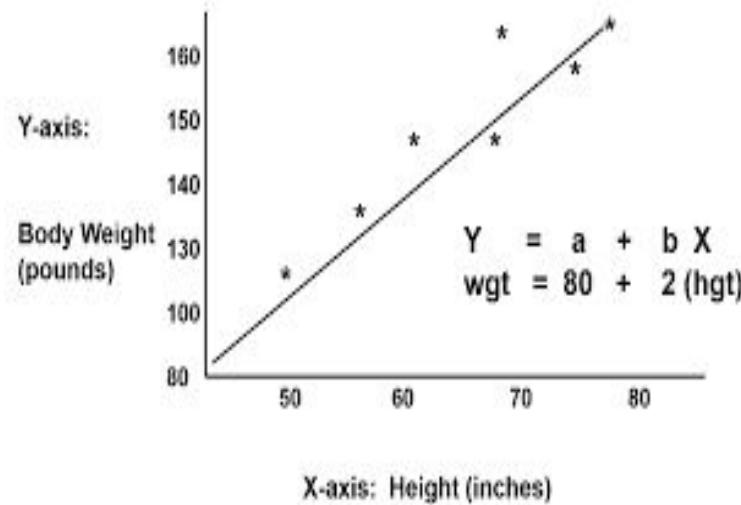
Simple Linear Regression:

y = dependent variable x = independent variable

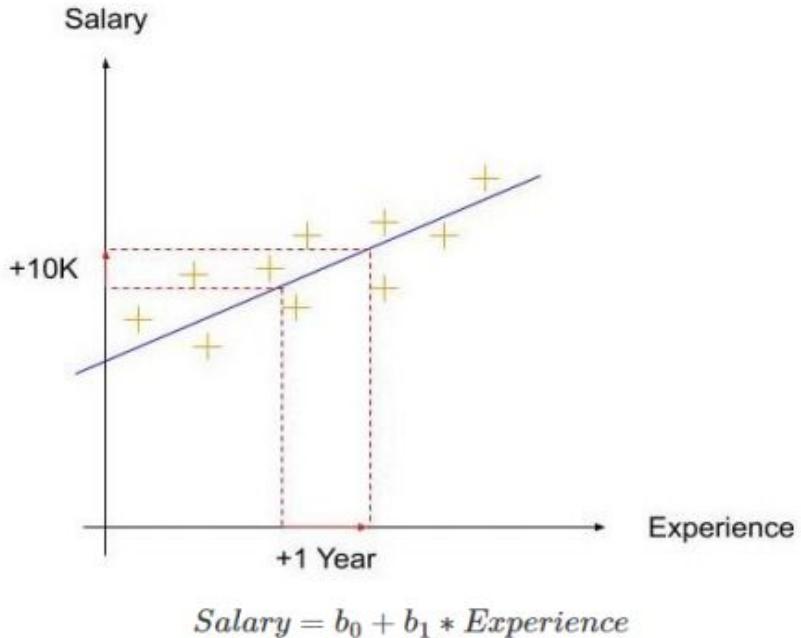
b0 = constant

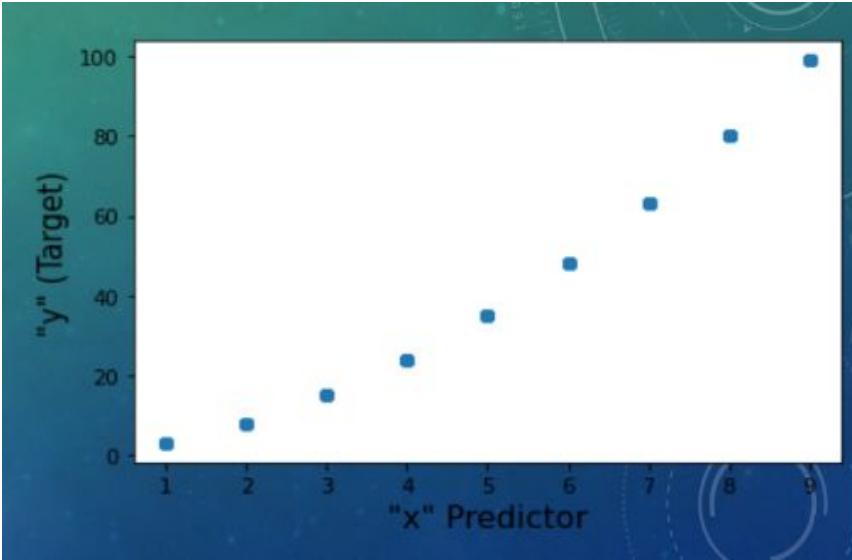
b1 = determines how a unit change in x will make a unit change in y.

It is also known as the slope of the line which determines to which extent the change will inflate or deflate.

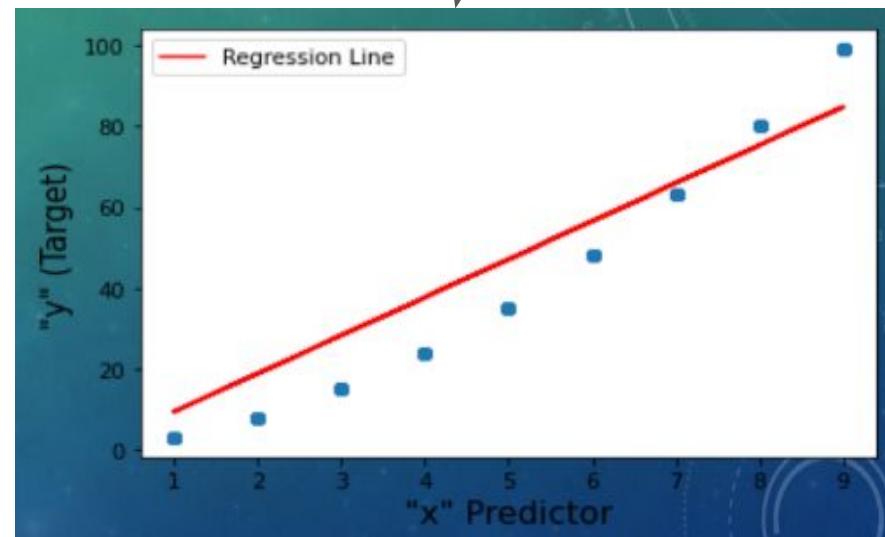


- Simple Linear Regression is used with data having only one feature and one label.
- SLR is more suited for data visualization as there are only two axes to plot the variables.





DATA



train - test Split

```
>>>from sklearn.linear_model import train_test_split.  
x_train, x_test, y_train, y_test = train_test_split(x,y, test_size=0.2)
```

- Split arrays or matrices into random train and test subsets.
- “`test_size`” should be between 0.0 and 1.0 and represent the proportion of the dataset to include in the test split.

Training Linear Regression

```
>>> from sklearn.linear_model import LinearRegression  
  
>>>model=LinearRegression()  
  
>>>model.fit(x_train, y_train)
```

Mean Square Error:

Measures the Average of the Squares of the error

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^n (Y_i - \hat{Y}_i)^2.$$

- Where One is Actual Value and the other one is the predicted value.
- Predicted Value = $mx + b$, where “m” is slope and “b” is the intercept

r2_Score

```
>>> from sklearn.metrics import r2_score
```

$$R^2 = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \mu)^2}.$$

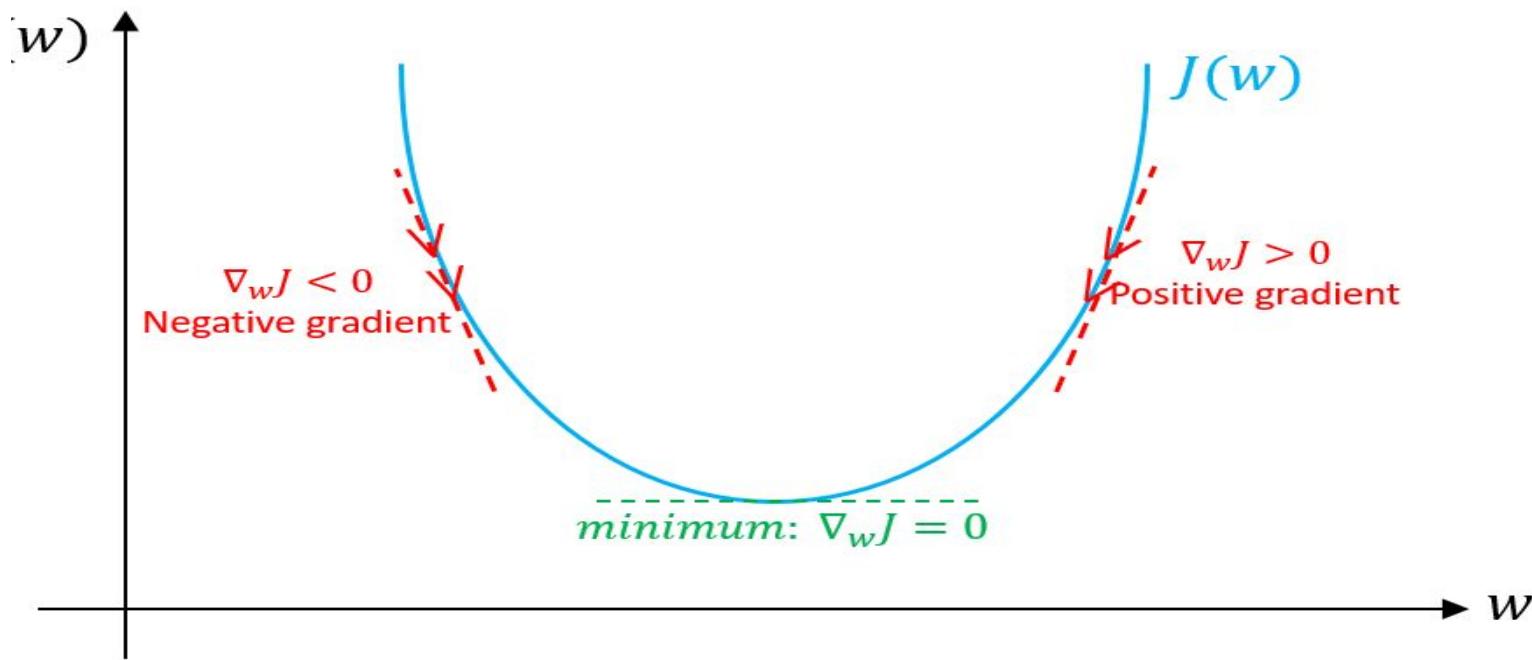
r2_score=1 means:

- prediction = actual value

Simple Linear Regression Practical

Gradient Descent

Gradient Descent is an Optimization Algorithm that find the local minimum of a Loss/ Error function.



$$MSE = \frac{1}{2m} \sum_{i=1}^m (\hat{y}^{(i)} - y^{(i)})^2$$

- i - ith Sample
- \hat{y} - Predicted Value
- y - Actual Value

Gradients

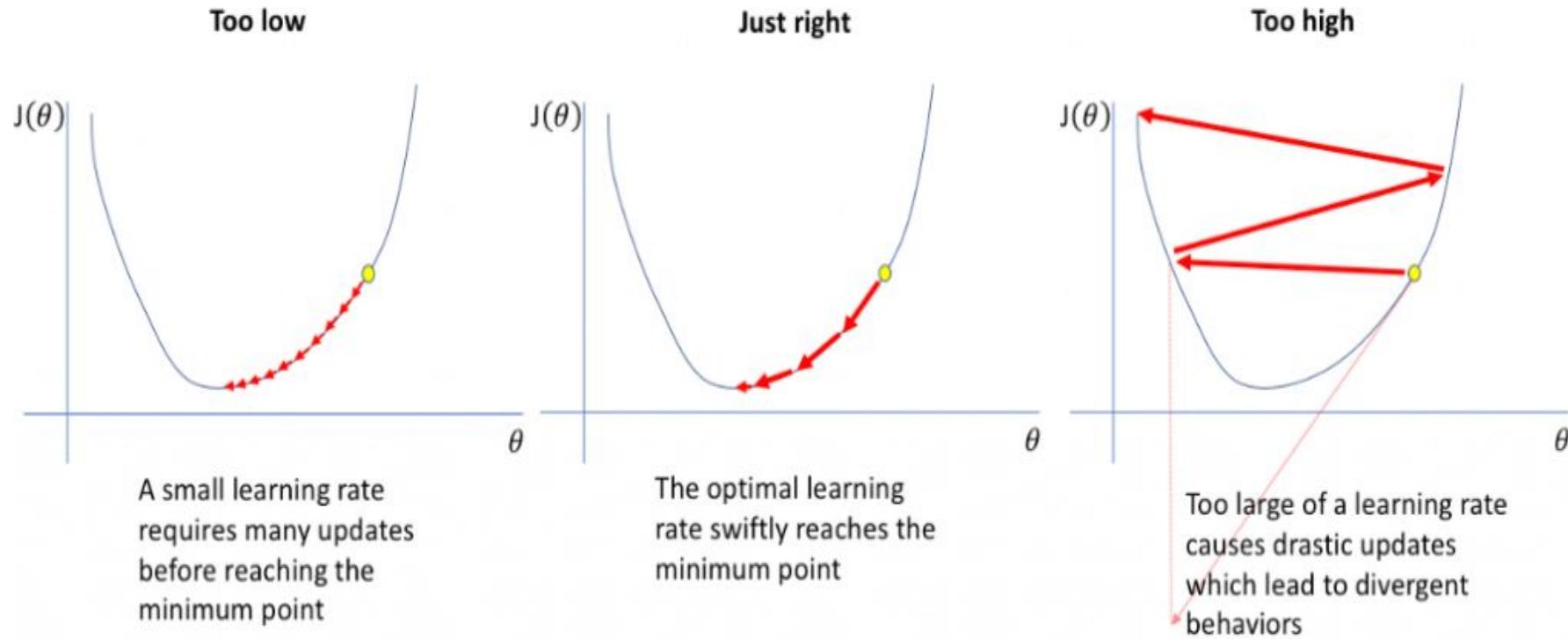
$$\frac{\partial}{\partial m} = \frac{2}{N} \sum_{i=1}^N -x_i(y_i - (mx_i + b))$$

predicted value = mx + b

$$\frac{\partial}{\partial b} = \frac{2}{N} \sum_{i=1}^N -(y_i - (mx_i + b))$$

Learning Rate (α)

Learning rate controls how quickly or slowly a model learns a problem.



Updating Parameters

```
> m = m - alpha*gradient_m
```

```
> b = b - alpha*gradient_b
```

Practical For Gradient Descent

Multiple Linear Regression:

y = dependent variable

b0 = constant

b1, b2 ... bn = coefficients

x1, x2, ... xn = independent variables

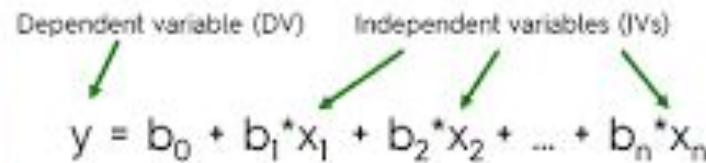
Simple
Linear
Regression

$$y = b_0 + b_1 * x_1$$

Multiple
Linear
Regression

$$y = b_0 + b_1 * x_1 + b_2 * x_2 + \dots + b_n * x_n$$

Dependent variable (DV) Independent variables (IVs)



Multiple Linear Regression

- Uses several explanatory(predictor) variables to predict the outcome of a response variable.

Necessary Assumption

- Linearity
- Homoscedasticity
- Multivariate Normality
- Independence of Error
- Lack of Multicollinearity

Practical for Multiple Linear Regression

Ridge Regression

Ridge Regression is a technique for analyzing multiple **regression** data that suffer from multicollinearity as well used when we have a high stepped line and we want to reduce it using some regularization technique

It Try to increase bias and to lower the variance.

Linear least squares with l2 regularization

Minimizes the objective function:

$$\|y - Xw\|^2 + \alpha * \|w\|^2$$

- `from sklearn.linear_model import Ridge`
- `Model = Ridge(alpha=1.0)`

Ridge Regression Practical

Lasso Regression

“LASSO” stands for **Least Absolute Shrinkage and Selection Operator**.

Lasso regression is a type of linear **regression** that uses shrinkage.

Does same job as ridge Regression but hear it also used as feature selection as its value shrink to zero wheres ridge move toward zero but never reach there.

This particular type of regression is well-suited for models showing high levels of multi-collinearity.

- `from sklearn.linear_model import Lasso`
- `Model = Lasso(alpha=1.0)`

Practical Lasso Regression

$$\|y - Xw\|^2 + \text{alpha} * \|w\|$$

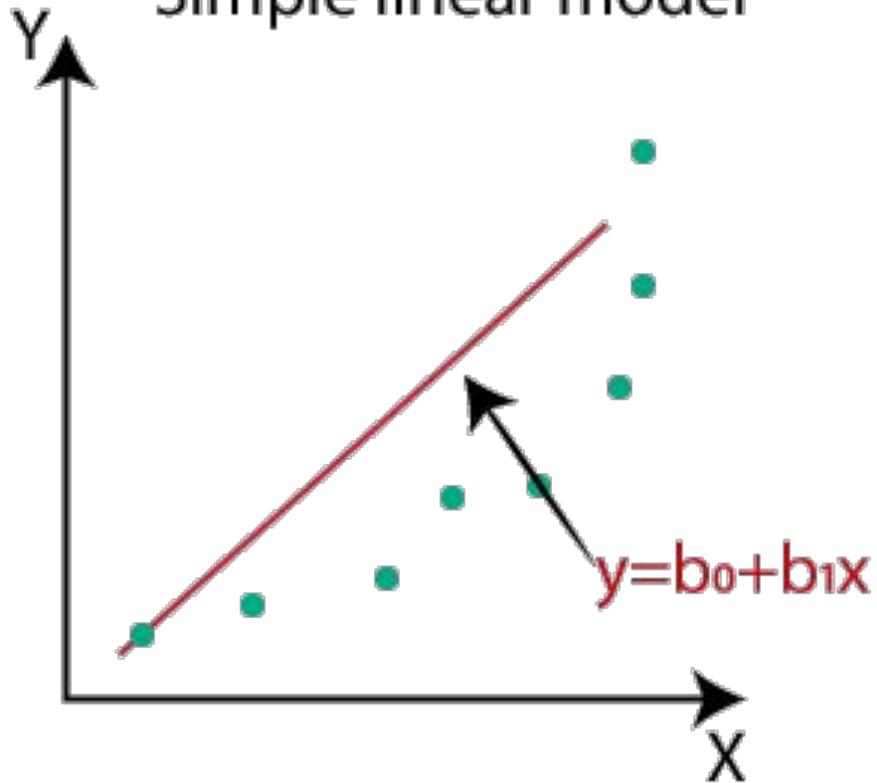
Polynomial Linear Regression

In Linear regression

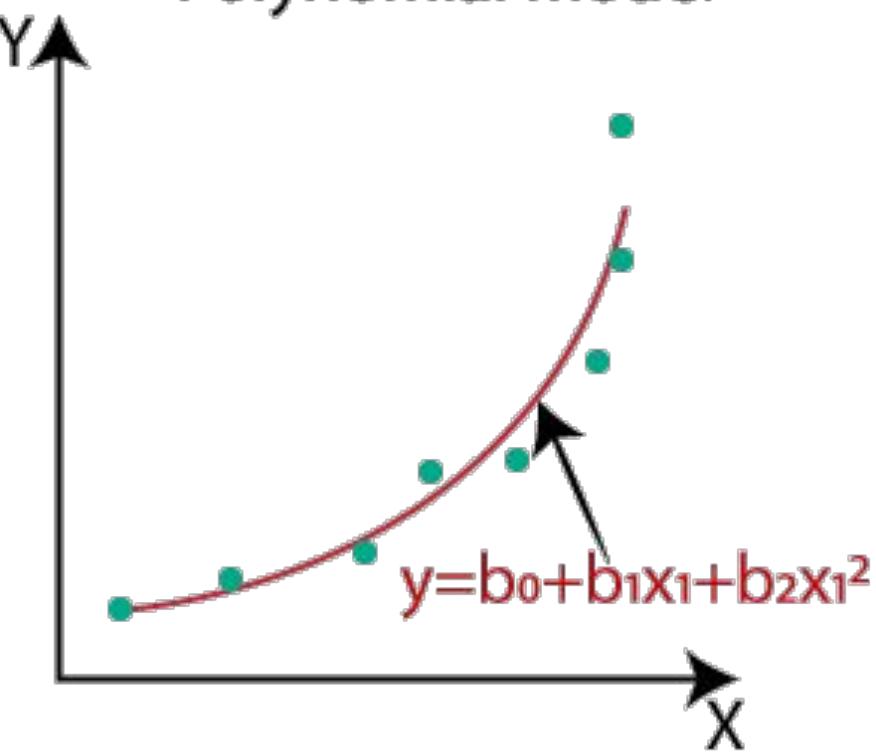
$$y = b_0 + b_1x_1 + b_2x_1^2 + b_3x_1^3 + \dots + b_nx_1^n$$

- It is not necessary to have data which can be plotted linearly, sometimes data can be in a nonlinear form as well, like a parabola as shown in the figure.
- In polynomial regression, we have a variable with different powers

Simple linear model



Polynomial model



Polynomial Regression

Practical

Classification Algorithms

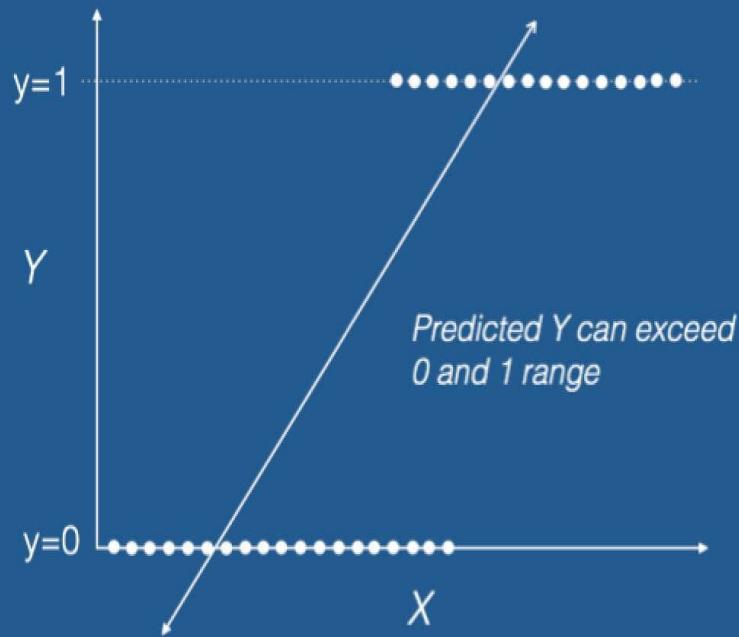
- Logistic Regression
- k-Nearest Neighbors
- Decision Tree Classifier
- Naive bayes Classifier
- SVM Classifier

Logistic Regression

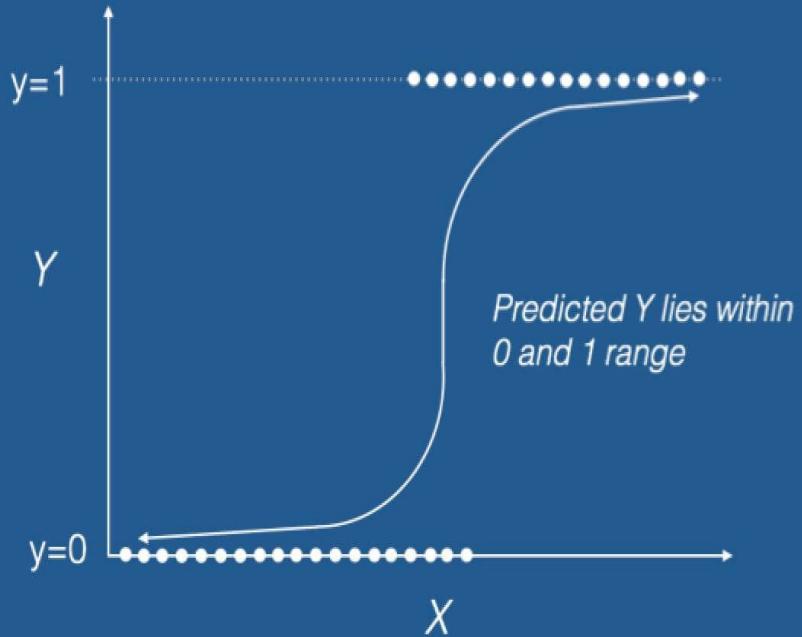
The **logistic regression** is a predictive analysis.

Logistic Regression, also known as **Logit Regression** or **Logit Model**, is a mathematical model used in statistics to estimate (guess) the **probability** of an event occurring having been given some previous data.

Linear Regression



Logistic Regression



Sigmoid Function

Simple Logistic Regression:

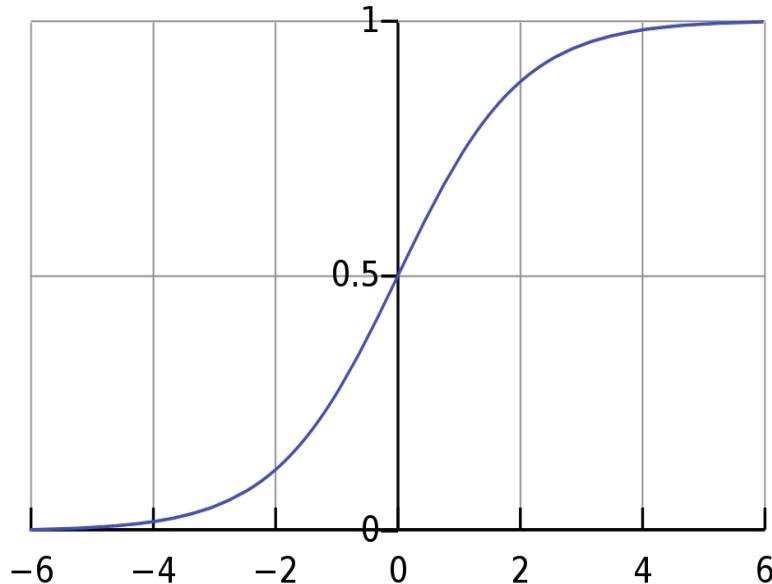
Output: 0 or 1

Hypothesis: $K = W * X + B$

$h\Theta(x) = \text{sigmoid}(K)$

Sigmoid Function:

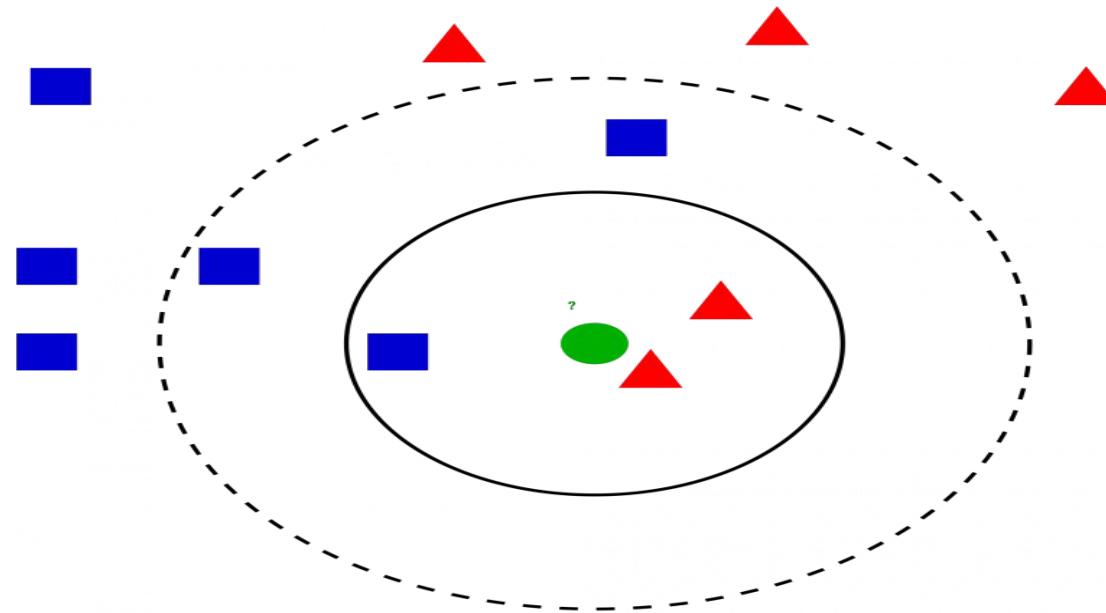
$$S(x) = \frac{1}{1 + e^{-x}} = \frac{e^x}{e^x + 1}.$$



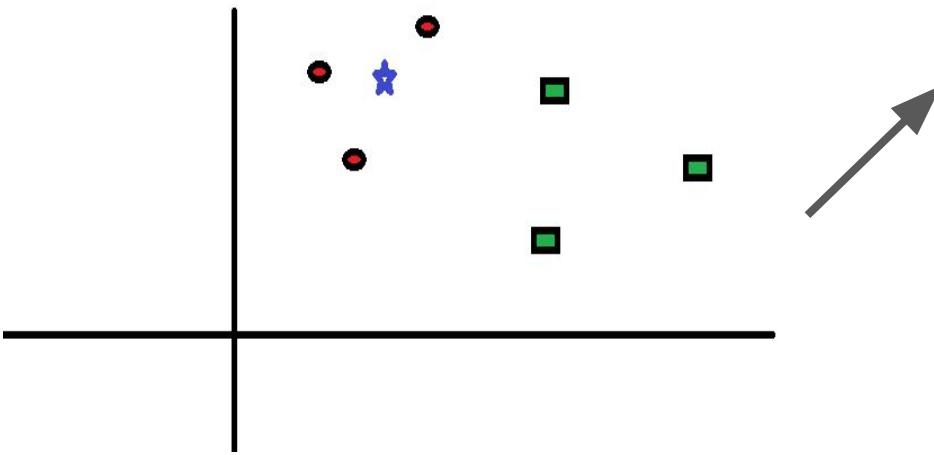
Practical For Logistic Regression

K-Nearest Neighbour

- It's a Classification technique.

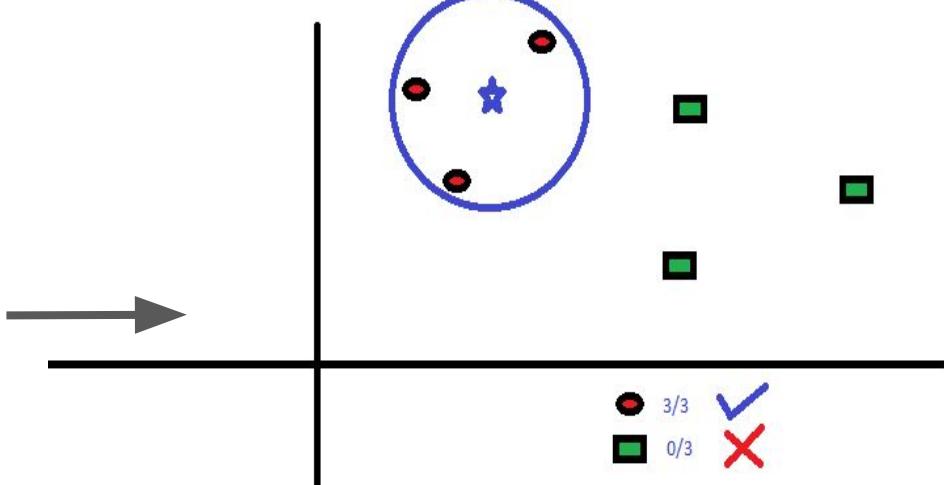


How does it work?



- You intend to find out the class of the blue star.
- The “K” in KNN algorithm is the nearest neighbor we wish to take the vote from. Let’s say $K = 3$.

- Hence, we will now make a circle with BS as the center just as big as to enclose only three data points on the plane.
- The three closest points to BS is all RC. Hence, with a good confidence level, we can say that the BS should belong to the class RC.



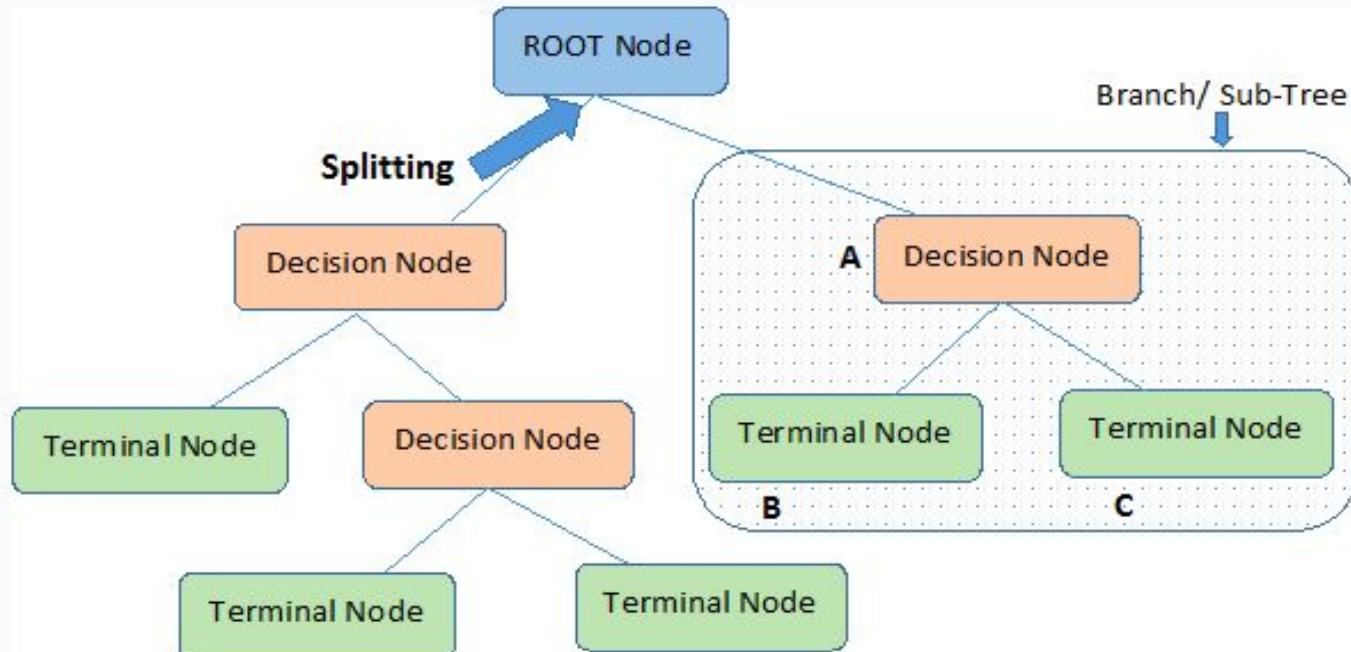
How to apply?

```
from sklearn.neighbors import  
KNeighborsClassifier  
  
clf=KNeighborsClassifier(n_neighbors=3)
```

Now you can simply fit this Classifier Algorithm, just like a normal classifier, using `clf.fit()` method.

Practical For KNN Classifier

Decision Tree



Note:- A is parent node of B and C.

Important Terminology related to Decision Trees

1. **Root Node:** It represents the entire population or sample and this further gets divided into two or more homogeneous sets.
2. **Splitting:** It is a process of dividing a node into two or more sub-nodes.

3. Decision Node: When a sub-node splits into further sub-nodes, then it is called the decision node.

4. Leaf / Terminal Node: Nodes do not split is called Leaf or Terminal node.

5. Pruning: When we remove sub-nodes of a decision node, this process is called pruning. You can say the opposite process of splitting.

6. Branch / Sub-Tree: A subsection of the entire tree is called branch or sub-tree.

7. Parent and Child Node: A node, which is divided into sub-nodes is called a parent node of sub-nodes whereas sub-nodes are the child of a parent node.

Solving this attribute selection problem

- 1. Ginni Index**
- 2. Entropy**
- 3. Information Gain**

Gini Index

$$Gini = 1 - \sum_{i=1}^C (P_i)^2$$

Steps to Calculate Gini index for a split

1. Calculate Gini for sub-nodes, using the above formula for success(p) and failure(q)
2. Calculate the Gini index for split using the weighted Gini score of each node of that split.

CART (Classification and Regression Tree) uses the Gini index method to create split points.

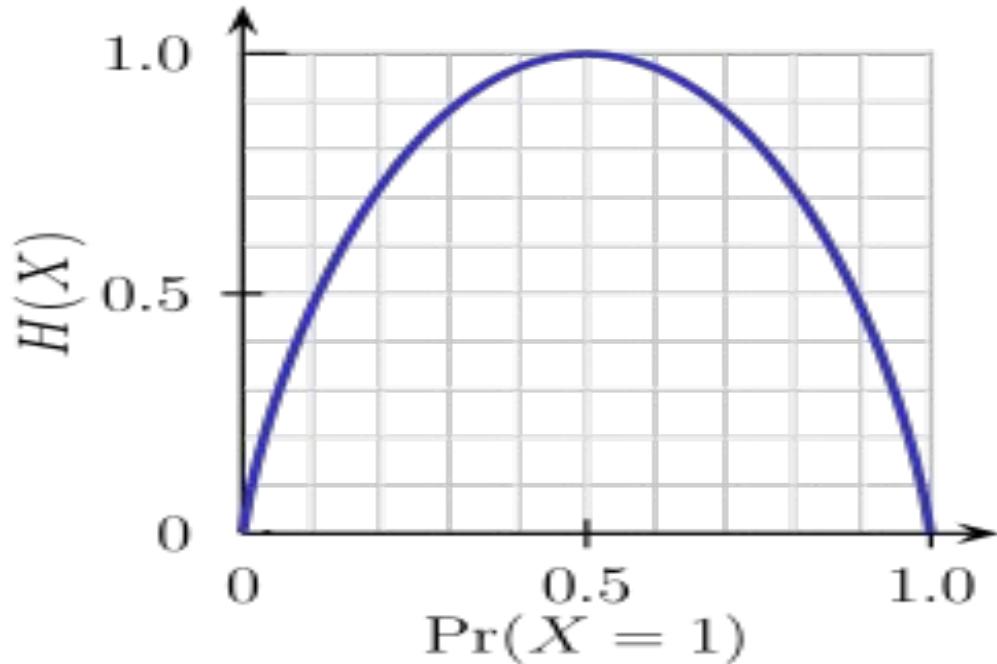
What is Entropy and Information Gain?

Entropy:

Entropy is a measure of the randomness in the information being processed. The higher the entropy, the harder it is to draw any conclusions from that information.

$$E = - \sum_1^m p(x) \log p(x)$$

Where $p(x)$ = fraction of examples in a class.



Flipping a coin is an example of an action that provides information that is random.

Information Gain:

$$IG = Entropy(Parent) - \sum_1^m ((WA)_i * (Entropy(Child)_i))$$

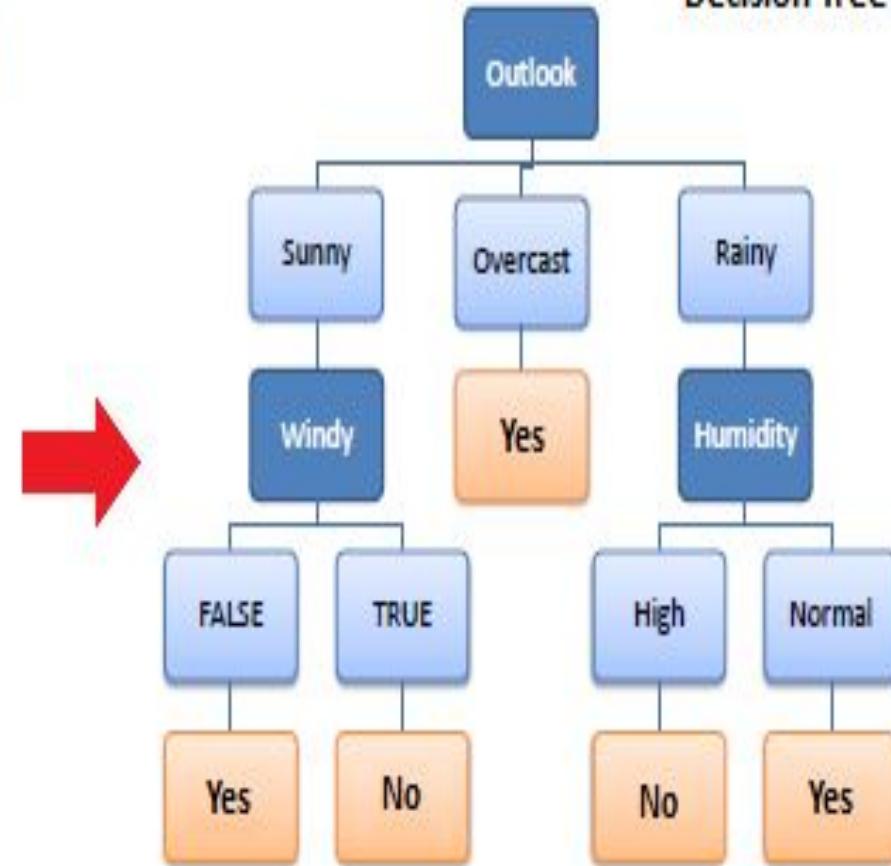
Where IG = Information gain, and WA = Weighted Average.

Decision Tree - Classification

Outlook	Temperature	Humidity	Windy	PlayTennis
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Rainy	Mild	High	True	No

Predictors				Target
Outlook	Temp.	Humidity	Windy	Play Golf
Rainy	Hot	High	False	No
Rainy	Hot	High	True	No
Overcast	Hot	High	False	Yes
Sunny	Mild	High	False	Yes
Sunny	Cool	Normal	False	Yes
Sunny	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Rainy	Mild	High	False	No
Rainy	Cool	Normal	False	Yes
Sunny	Mild	Normal	False	Yes
Rainy	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Sunny	Mild	High	True	No

Decision Tree



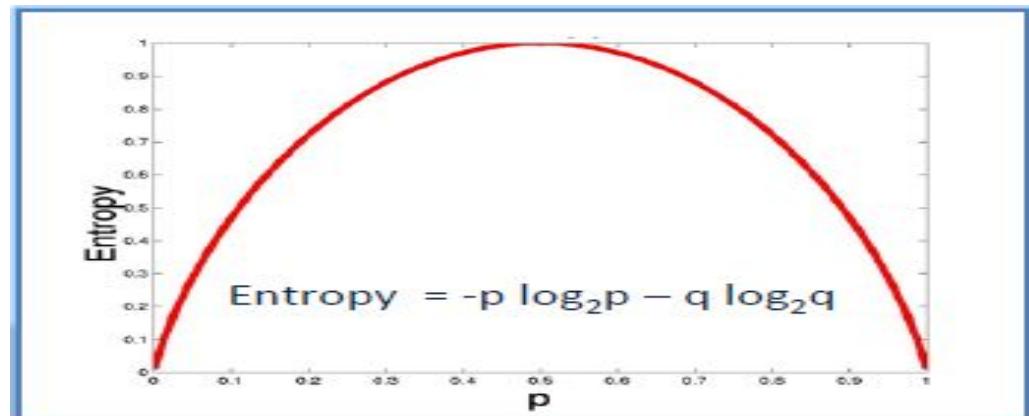
Algorithm

The core algorithm for building decision trees called **ID3** by J. R. Quinlan which employs a top-down, greedy search through the space of possible branches with no backtracking.

ID3 uses *Entropy* and *Information Gain* to construct a decision tree.

Entropy

A decision tree is built top-down from a root node and involves partitioning the data into subsets that contain instances with similar values (homogenous). ID3 algorithm uses entropy to calculate the homogeneity of a sample. If the sample is completely homogeneous the entropy is zero and if the sample is an equally divided it has entropy of one.



$$\text{Entropy} = -0.5 \log_2 0.5 - 0.5 \log_2 0.5 = 1$$

Entropy

Entropy is a measure of randomness. In other words, its a measure of unpredictability. We will take a moment here to give entropy in case of binary event(like the coin toss, where output can be either of the two events, head or tail) a mathematical face:

$$\text{Entropy} = -(\text{probability}(a) * \log_2(\text{probability}(a))) - (\text{probability}(b) * \log_2(\text{probability}(b)))$$

To build a decision tree, we need to calculate two types of entropy using frequency tables as follows:

a) Entropy using the frequency table of one attribute:

$$E(S) = \sum_{i=1}^c -p_i \log_2 p_i$$

Play Golf	
Yes	No
9	5



$$\begin{aligned}\text{Entropy(PlayGolf)} &= \text{Entropy}(5,9) \\ &= \text{Entropy}(0.36, 0.64) \\ &= -(0.36 \log_2 0.36) - (0.64 \log_2 0.64) \\ &= 0.94\end{aligned}$$

b) Entropy using the frequency table of two attributes:

$$E(T, X) = \sum_{c \in X} P(c)E(c)$$

		Play Golf		
		Yes	No	
Outlook	Sunny	3	2	5
	Overcast	4	0	4
	Rainy	2	3	5
				14



$$\begin{aligned} E(\text{PlayGolf}, \text{Outlook}) &= P(\text{Sunny}) * E(3,2) + P(\text{Overcast}) * E(4,0) + P(\text{Rainy}) * E(2,3) \\ &= (5/14) * 0.971 + (4/14) * 0.0 + (5/14) * 0.971 \\ &= 0.693 \end{aligned}$$

Information Gain

The information gain is based on the decrease in entropy after a dataset is split on an attribute. Constructing a decision tree is all about finding attribute that returns the highest information gain (i.e., the most homogeneous branches).

Step 1: Calculate entropy of the target.

$$\text{Entropy}(\text{PlayGolf}) = \text{Entropy}(5,9)$$

$$= \text{Entropy}(0.36, 0.64)$$

$$= - (0.36 \log_2 0.36) - (0.64 \log_2 0.64)$$

$$= 0.94$$

Step 2: The dataset is then split on the different attributes. The entropy for each branch is calculated. Then it is added proportionally, to get total entropy for the split. The resulting entropy is subtracted from the entropy before the split. The result is the Information Gain, or decrease in entropy.

		Play Golf	
		Yes	No
Outlook	Sunny	3	2
	Overcast	4	0
	Rainy	2	3
Gain = 0.247			

		Play Golf	
		Yes	No
Temp.	Hot	2	2
	Mild	4	2
	Cool	3	1
Gain = 0.029			

		Play Golf	
		Yes	No
Humidity	High	3	4
	Normal	6	1
Gain = 0.152			

		Play Golf	
		Yes	No
Windy	False	6	2
	True	3	3
Gain = 0.048			

$$Gain(T, X) = Entropy(T) - Entropy(T, X)$$

$$\begin{aligned} G(\text{PlayGolf}, \text{Outlook}) &= E(\text{PlayGolf}) - E(\text{PlayGolf}, \text{Outlook}) \\ &= 0.940 - 0.693 = 0.247 \end{aligned}$$

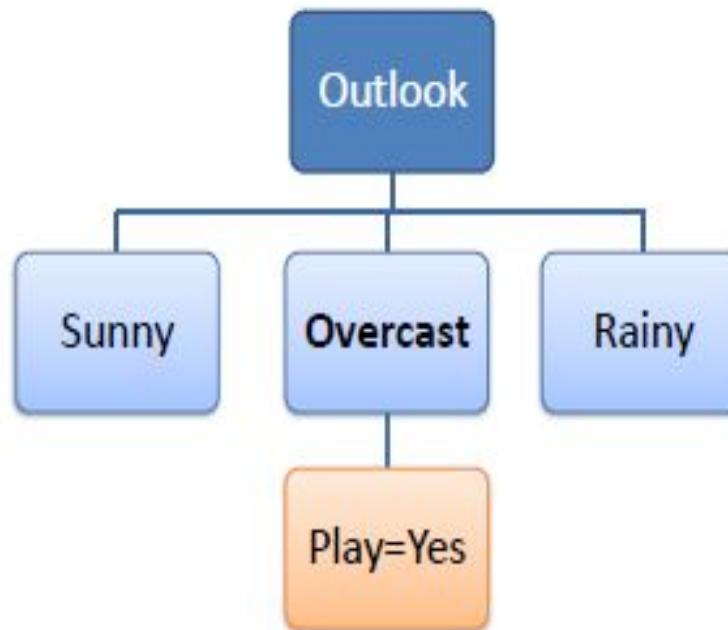
Step 3: Choose attribute with the largest information gain as the decision node, divide the dataset by its branches and repeat the same process on every branch.

		Play Golf	
		Yes	No
Outlook	Sunny	3	2
	Overcast	4	0
	Rainy	2	3
Gain = 0.247			

	Outlook	Temp	Humidity	Windy	Play Golf
Outlook	Sunny	Mild	High	FALSE	Yes
Outlook	Sunny	Cool	Normal	FALSE	Yes
Outlook	Sunny	Cool	Normal	TRUE	No
Outlook	Sunny	Mild	Normal	FALSE	Yes
Outlook	Sunny	Mild	High	TRUE	No
Outlook	Overcast	Hot	High	FALSE	Yes
Outlook	Overcast	Cool	Normal	TRUE	Yes
Outlook	Overcast	Mild	High	TRUE	Yes
Outlook	Overcast	Hot	Normal	FALSE	Yes
Outlook	Rainy	Hot	High	FALSE	No
Outlook	Rainy	Hot	High	TRUE	No
Outlook	Rainy	Mild	High	FALSE	No
Outlook	Rainy	Cool	Normal	FALSE	Yes
Outlook	Rainy	Mild	Normal	TRUE	Yes

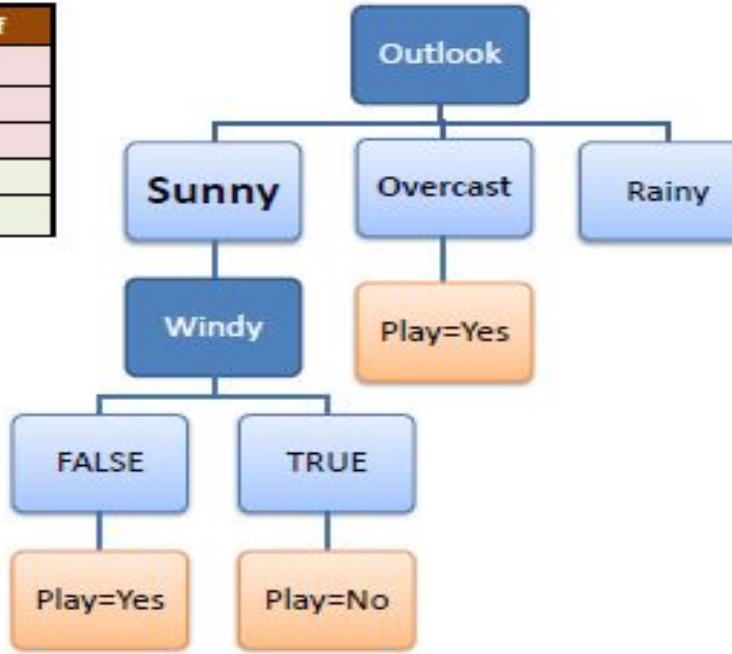
Step 4a: A branch with entropy of 0 is a leaf node.

Temp	Humidity	Windy	Play Golf
Hot	High	FALSE	Yes
Cool	Normal	TRUE	Yes
Mild	High	TRUE	Yes
Hot	Normal	FALSE	Yes



Step 4b: A branch with entropy more than 0 needs further splitting.

Temp	Humidity	Windy	Play Golf
Mild	High	FALSE	Yes
Cool	Normal	FALSE	Yes
Mild	Normal	FALSE	Yes
Cool	Normal	TRUE	No
Mild	High	TRUE	No



Step 5: The ID3 algorithm is run recursively on the non-leaf branches, until all data is classified.

$R_1: \text{IF } (\text{Outlook}=\text{Sunny}) \text{ AND } (\text{Windy}=\text{FALSE}) \text{ THEN Play}=\text{Yes}$

$R_2: \text{IF } (\text{Outlook}=\text{Sunny}) \text{ AND } (\text{Windy}=\text{TRUE}) \text{ THEN Play}=\text{No}$

$R_3: \text{IF } (\text{Outlook}=\text{Overcast}) \text{ THEN Play}=\text{Yes}$

$R_4: \text{IF } (\text{Outlook}=\text{Rainy}) \text{ AND } (\text{Humidity}=\text{High}) \text{ THEN Play}=\text{No}$

$R_5: \text{IF } (\text{Outlook}=\text{Rain}) \text{ AND } (\text{Humidity}=\text{Normal}) \text{ THEN Play}=\text{Yes}$



Naive Bayes Classifier

Naive Bayes Classifier

The Naive Bayesian classifier is based on Bayes' theorem with the independence assumptions between predictors.

Naive Bayesian model is easy to build, with no complication.

$$P(c|x) = \frac{P(x|c)P(c)}{P(x)}$$

Likelihood Class Prior Probability
↓ ↓
Posterior Probability Predictor Prior Probability

$$P(c|X) = P(x_1|c) \times P(x_2|c) \times \cdots \times P(x_n|c) \times P(c)$$

- $P(c|x)$ is the posterior probability of class (*target*) given *predictor* (*attribute*).
- $P(c)$ is the prior probability of class.
- $P(x|c)$ is the likelihood which is the probability of *predictor* given *class*.
- $P(x)$ is the prior probability of *predictor*.

- The posterior probability can be calculated by first, constructing a frequency table for each attribute against the target.
- Then, transforming the frequency tables to likelihood tables and finally use the Naive Bayesian equation to calculate the posterior probability for each class.
- The class with the highest posterior probability is the outcome of prediction.

Outlook	Temperature	Humidity	Windy	PlayTennis
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Rainy	Mild	High	True	No

$$P(x | c) = P(\text{Sunny} | \text{Yes}) = 3 / 9 = 0.33$$

Frequency Table		Play Golf	
		Yes	No
Outlook	Sunny	3	2
	Overcast	4	0
	Rainy	2	3

Likelihood Table		Play Golf		
		Yes	No	
Outlook	Sunny	3/9	2/5	5/14
	Overcast	4/9	0/5	4/14
	Rainy	2/9	3/5	5/14
		9/14	5/14	

$P(x) = P(\text{Sunny})$
 $= 5 / 14 = 0.36$

$P(c) = P(\text{Yes}) = 9 / 14 = 0.64$

Posterior Probability:

$$P(c | x) = P(\text{Yes} | \text{Sunny}) = 0.33 \times 0.64 \div 0.36 = 0.60$$



$$P(x | c) = P(\text{Sunny} | \text{No}) = 2 / 5 = 0.4$$

Frequency Table		Play Golf	
		Yes	No
Outlook	Sunny	3	2
	Overcast	4	0
	Rainy	2	3



		Play Golf		
		Yes	No	
Outlook	Sunny	3	2	5
	Overcast	4	0	4
	Rainy	2	3	5
		9	5	14

$$\begin{aligned}P(x) &= P(\text{Sunny}) \\&= 5 / 14 = 0.36\end{aligned}$$

$$P(c) = P(\text{No}) = 5 / 14 = 0.36$$

Posterior Probability:

$$P(c | x) = P(\text{No} | \text{Sunny}) = 0.40 \times 0.36 / 0.36 = 0.40$$



Example 2:

In this example we have 4 inputs (predictors). The final posterior probabilities can be standardized between 0 and 1.

Outlook	Temp	Humidity	Windy	Play
Rainy	Cool	High	True	?

$$P(Yes | X) = P(Rainy | Yes) \times P(Cool | Yes) \times P(High | Yes) \times P(True | Yes) \times P(Yes)$$

$$P(Yes | X) = 2/9 \times 3/9 \times 3/9 \times 3/9 \times 9/14 = 0.00529 \quad 0.2 = \frac{0.00529}{0.02057 + 0.00529}$$

$$P(No | X) = P(Rainy | No) \times P(Cool | No) \times P(High | No) \times P(True | No) \times P(No)$$

$$P(No | X) = 3/5 \times 1/5 \times 4/5 \times 3/5 \times 5/14 = 0.02057 \quad 0.8 = \frac{0.02057}{0.02057 + 0.00529}$$

Naive Bayes Practical

Support Vector Machine - Classification (SVM)

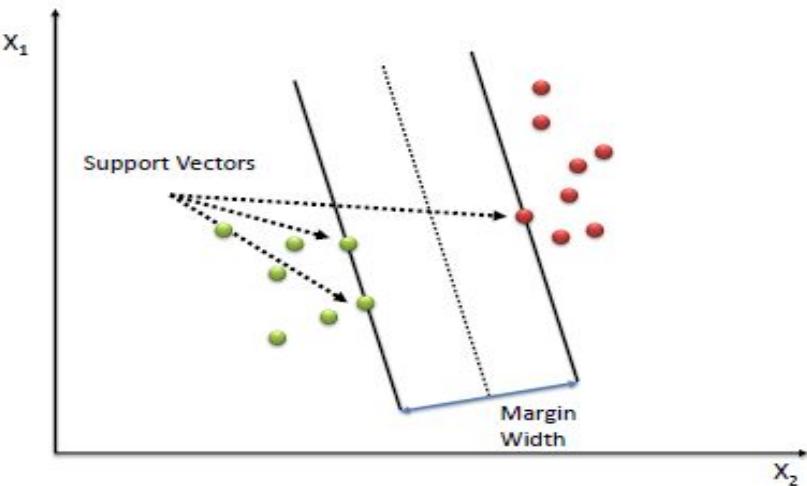
Subset of training points in the decision function (called support vectors)

A Support Vector Machine (**SVM**) performs classification by finding the **hyperplane** that maximizes the **margin** between the two classes. The vectors (cases) that define the hyperplane are the support vectors.

It is used for both classification or regression challenges.

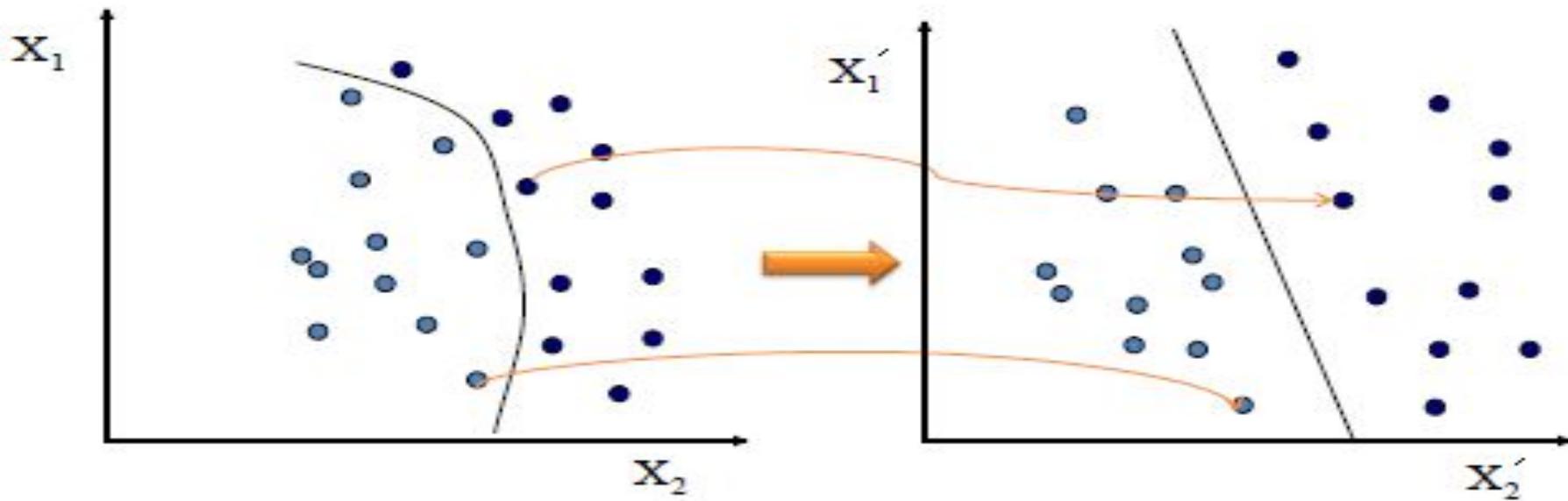
we plot each data item as a point in n-dimensional space (where n is number of features you have) with the value of each feature being the value of a particular coordinate.

we perform classification by finding the hyper-plane that differentiates the two classes very well



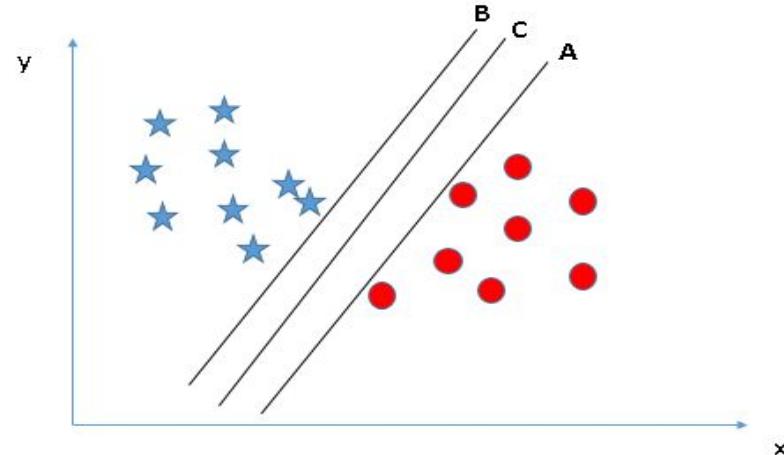
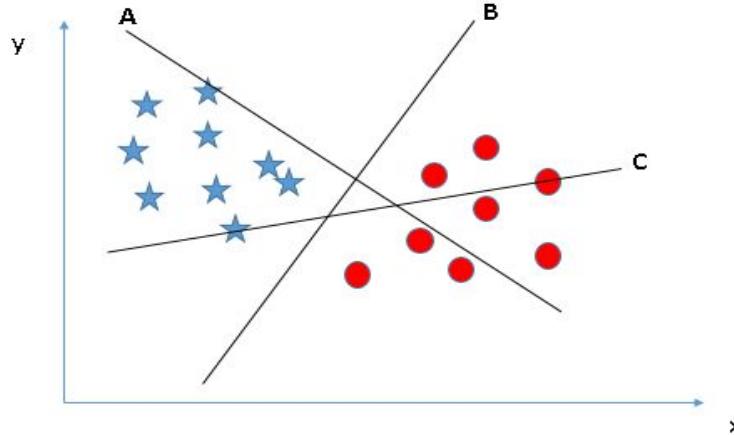
The simplest way to separate two groups of data is with a **straight line** (1 dimension), **flat plane** (2 dimensions) or an **N-dimensional hyperplane**.

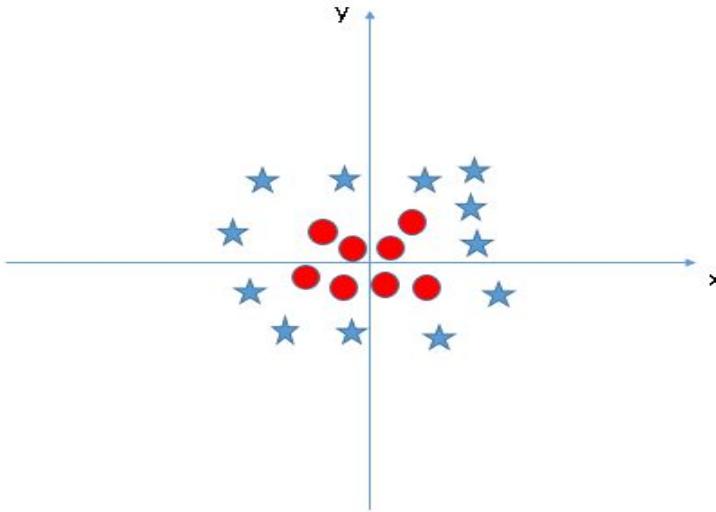
However, there are situations where a nonlinear region can separate the groups more efficiently. SVM handles this by using a **kernel function (nonlinear)** to map the data into a different space where a hyperplane (linear) cannot be used to do the separation.



SVM algorithms use a set of mathematical functions that are defined as the kernel. The function of kernel is to take data as input and transform it into the required form. Different **SVM** algorithms use different types of kernel functions. These functions can be different types. For example ***linear, nonlinear, polynomial, radial basis function (RBF), and sigmoid.***

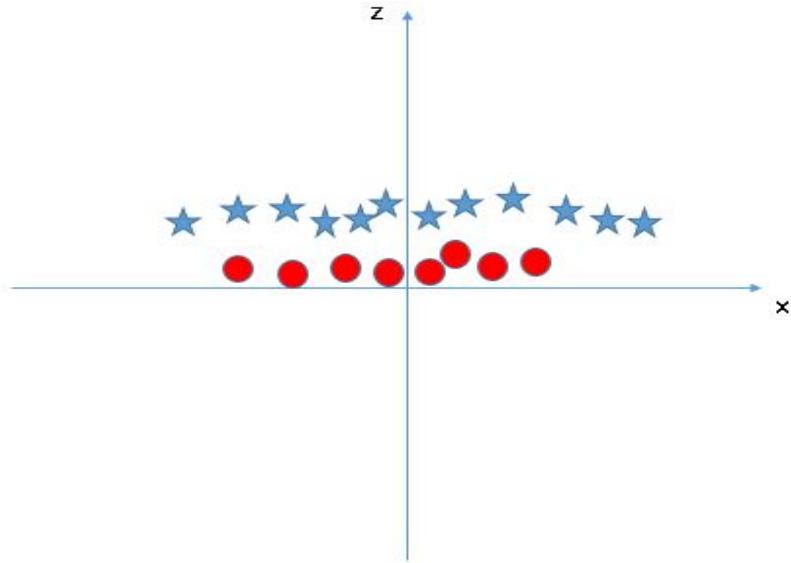
Identify the right hyper-plane





$$z = x^2 + y^2$$

- All values for z would be positive always because z is the squared sum of both x and y
- In the original plot, red circles appear close to the origin of x and y axes, leading to lower value of z and star relatively away from the origin result to higher value of z .



kernel: We have already discussed about it. Here, we have various options available with kernel like, “linear”, “rbf”, “poly” and others (default value is “rbf”). Here “rbf” and “poly” are useful for non-linear hyper-plane. Let’s look at the example, where we’ve used linear kernel on two feature of iris data set to classify their class.

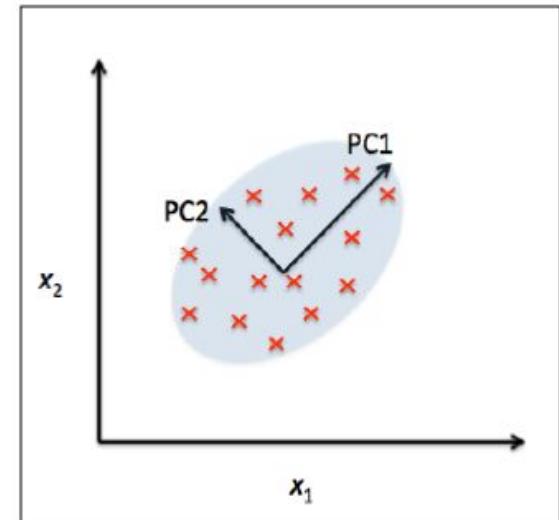
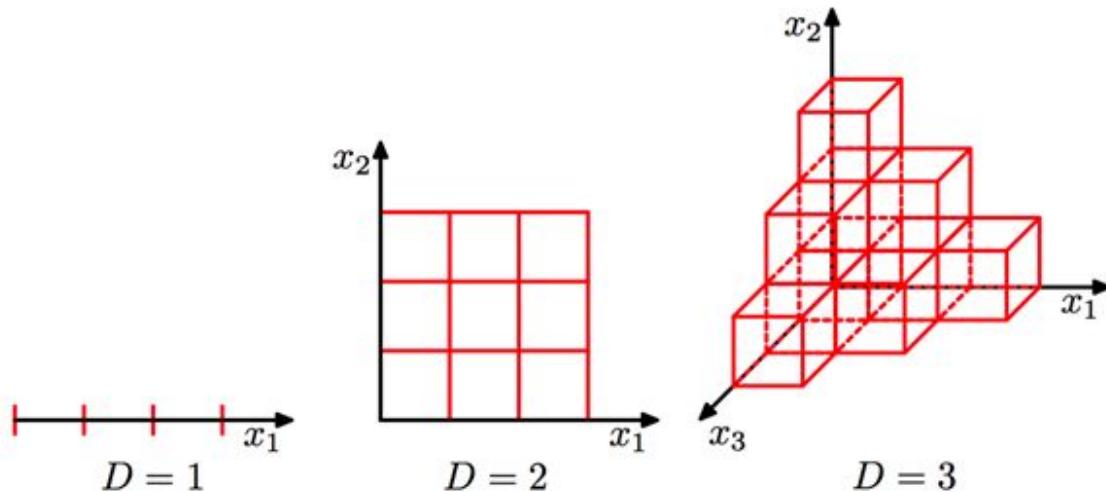
Support Vector Machine -Practical

Dimensionality Reduction Techniques

PCA

Principal Component Analysis (PCA) is an unsupervised,

High dimensionality means that the dataset has a large number of features.



Principal Component

The first principal component expresses the most amount of variance.

Each additional component expresses less variance and more noise, so representing the data with a smaller subset of principal components preserves the signal and discards the noise.

$$PC1 = w_{1,1}(\text{Feature A}) + w_{2,1}(\text{Feature B}) + w_{3,1}(\text{Feature C}) \dots + w_{n,1}(\text{Feature N})$$

$$PC2 = w_{1,2}(\text{Feature A}) + w_{2,2}(\text{Feature B}) + w_{3,2}(\text{Feature C}) \dots + w_{n,2}(\text{Feature N})$$

$$PC3 = w_{1,3}(\text{Feature A}) + w_{2,3}(\text{Feature B}) + w_{3,3}(\text{Feature C}) \dots + w_{n,3}(\text{Feature N})$$

PCA Limitations

Model performance: PCA can lead to a reduction in model performance on datasets with no or low feature correlation or does not meet the assumptions of linearity.

Classification accuracy: Variance based PCA framework does not consider the differentiating characteristics of the classes. Also, the information that distinguishes one class from another might be in the low variance components and may be discarded.

Outliers: PCA is also affected by outliers, and normalization of the data needs to be an essential component of any workflow.

Interpretability: Each principal component is a combination of original features and does not allow for the individual feature importance to be recognized.

Supervised:

LDA

Clustering

Clustering vs Classification

- Observations (or data points) in a classification task have labels. Each observation is classified according to some measurements. Classification algorithms try to model the relationship between measurements (features) on observations and their assigned class. Then the model predicts the class of new observations.
- Observations (or data points) in clustering do not have labels. We expect the model to find structures in the dataset so that similar observations can be grouped into clusters. We basically ask the model to label observations.

Clustering Algorithm

- K-mean Clustering**
- Hierarchical**

K-mean Clustering

K-means clustering aims to partition data into k clusters in a way that data points in the same cluster are similar and data points in the different clusters are farther apart.

There are many methods to measure the distance. [Euclidean distance](#) (minkowski distance with $p=2$) is one of most commonly used distance measurements.

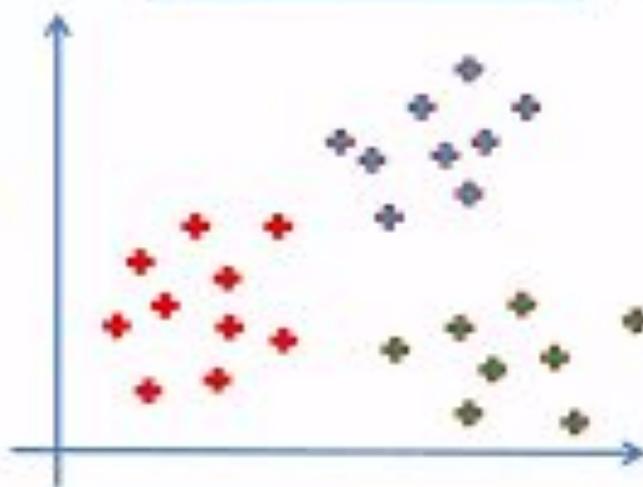
K-means clustering tries to minimize distances within a cluster and maximize the distance between different clusters. K-means algorithm is not capable of determining the number of clusters.

Before K-Means



K-Means

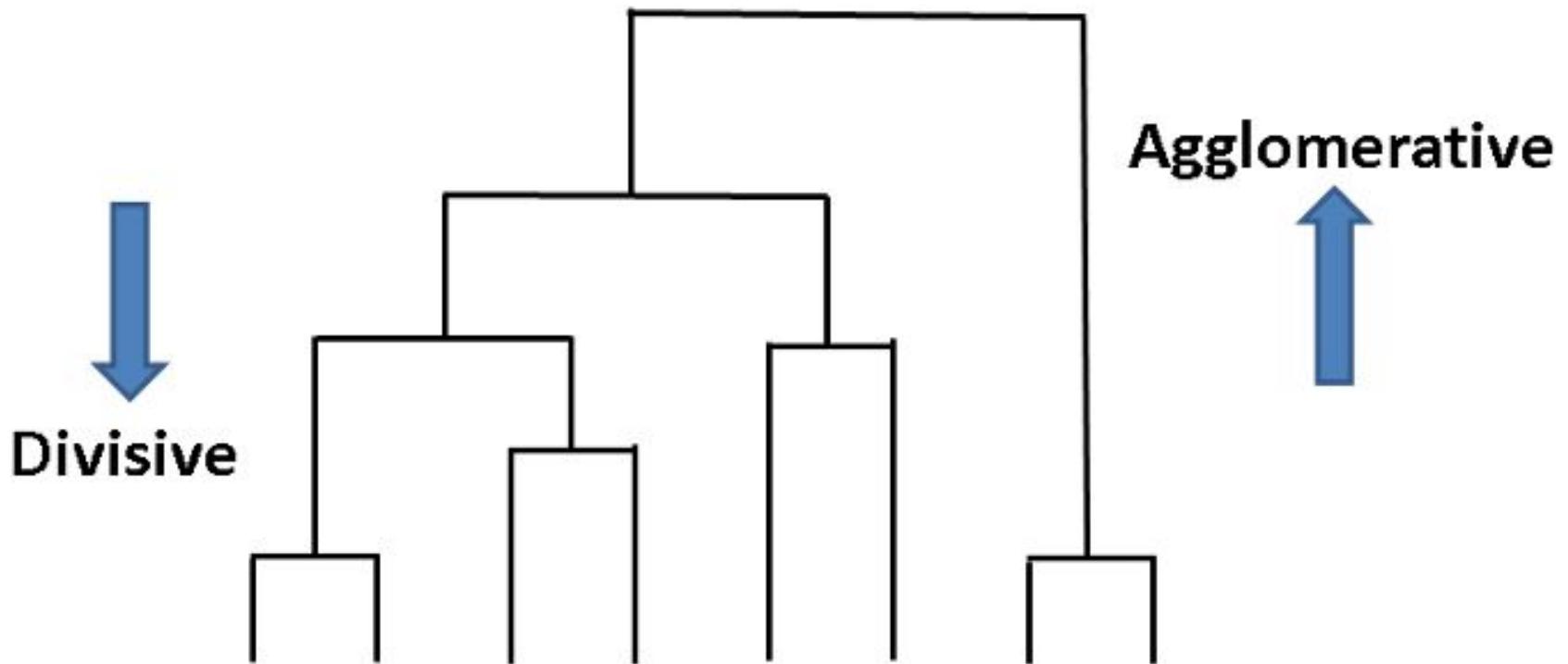
After K-Means



Hierarchical Clustering

Hierarchical clustering starts by treating each observation as a separate cluster. Then, it repeatedly executes the following two steps: (1) Identify the two clusters that are closest together, and (2) Merge the two most similar clusters. There are two types of **hierarchical clustering**:

Agglomerative clustering and Divisive clustering

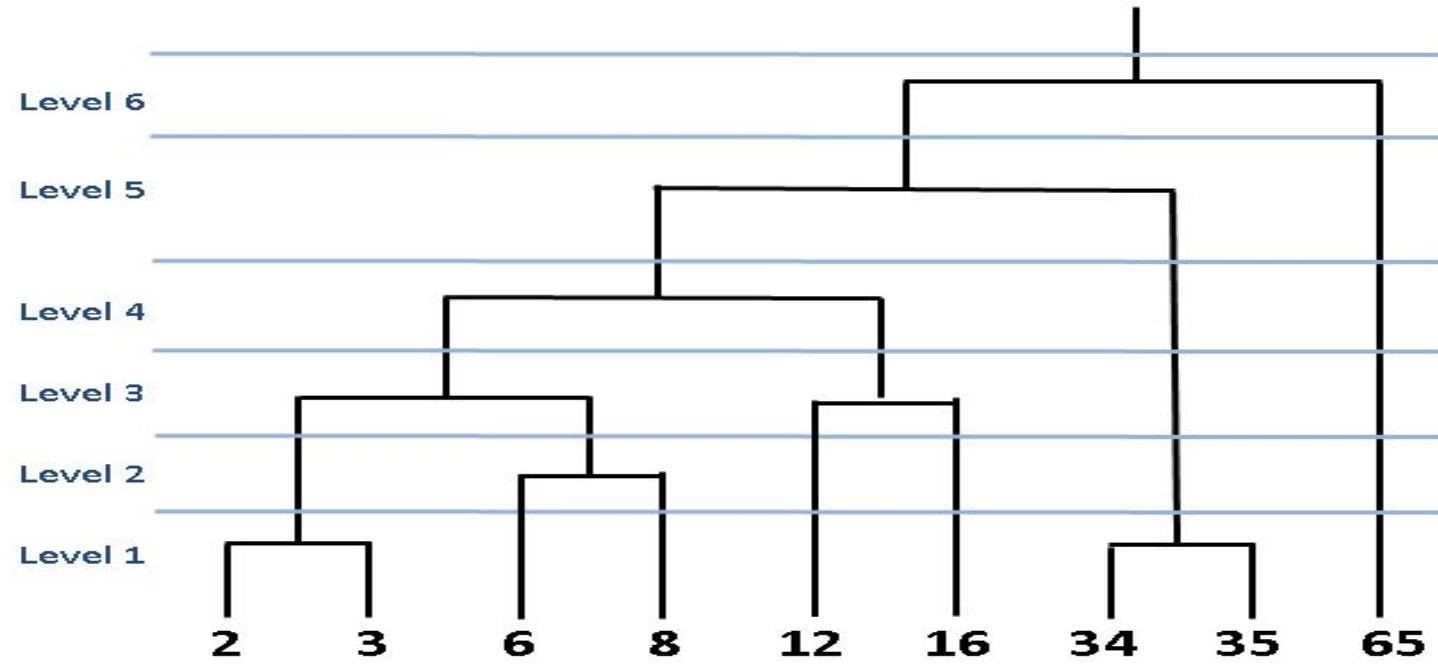


Hierarchical clustering typically works by sequentially merging similar clusters, as shown above. This is known as **agglomerative hierarchical clustering**. In theory, it can also be done by initially grouping all the observations into one cluster, and then successively splitting these clusters. This is known as **Divisive hierarchical clustering**. Divisive clustering is rarely done in practice.

Agglomerative clustering

Agglomerative clustering is kind of a bottom-up approach. Each data point is assumed to be a separate cluster at first. Then the similar clusters are iteratively combined.

- Stop after a number of clusters is reached (**n_clusters**)
- Set a threshold value for linkage (**distance_threshold**). If the distance between two clusters are above the threshold, these clusters will not be merged.



The figure above is called **dendrogram** which is a diagram representing tree-based approach. In hierarchical clustering, dendograms are used to visualize the relationship among clusters.

Math Intuition (What Is Linkage)

Here's one way to calculate similarity – Take the distance between the centroids of these clusters. The points having the least distance are referred to as similar points and we can merge them. We can refer to this as a **distance-based algorithm** as well (since we are calculating the distances between the clusters).

In hierarchical clustering, we have a concept called a **proximity matrix**. This stores the distances between each point.

During both the types of hierarchical clustering, the distance between two sub-clusters needs to be computed. The different types of linkages describe the different approaches to measure the distance between two sub-clusters of data points.

- **Single Linkage**

$$D(c_1, c_2) = \min D(x_i, x_j)$$

Minimum distance or distance between closest elements in clusters

- **Complete Linkage**

$$D(c_1, c_2) = \max D(x_i, x_j)$$

Maximum distance between elements in clusters

- **Average Linkage**

$$D(c_1, c_2) = \frac{1}{|c_1|} \frac{1}{|c_2|} \sum \sum D(x_i, x_j)$$

Average of the distances of all pairs

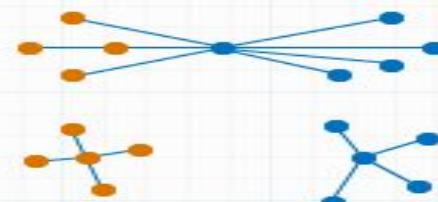
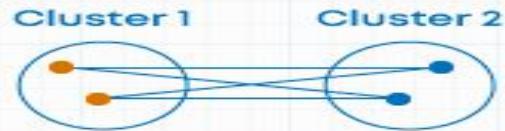
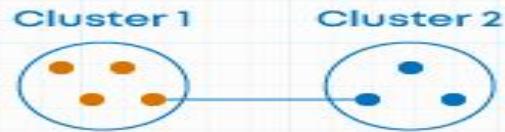
- **Centroid Method**

Combining clusters with minimum distance between the centroids of the two clusters

- **Ward's Method**

- Combining clusters where increase in within cluster variance is to the smallest degree.

- Objective is to minimize the total within cluster variance



Perform Hierarchical Clustering

Problem: Teacher Wants To Separate Students According Their Marks.

Student_ID	Marks
1	10
2	7
3	28
4	20
5	35

Initialization

Creating a Proximity Matrix

All the Distance Are Calculated By Euclidean Distance Formula

Distance between point 1 and 2:

$$\sqrt{(10-7)^2} = \sqrt{9} = 3$$

ID	1	2	3	4	5
1	0	3	18	10	25
2	3	0	21	13	28
3	18	21	0	8	7
4	10	13	8	0	15
5	25	28	7	15	0

Next, we will look at the smallest distance in the proximity matrix and merge the points with the smallest distance. We then update the proximity matrix:

Let's look at the updated clusters and accordingly update the proximity matrix:

ID	1	2	3	4	5
1	0	3	18	10	25
2	3	0	21	13	28
3	18	21	0	8	7
4	10	13	8	0	15
5	25	28	7	15	0

Student_ID	Marks
(1,2)	10
3	28
4	20
5	35

Updated Metrics

We will repeat step 2 until only a single cluster is left

We started with 5 clusters and finally have a single cluster. **This is how agglomerative hierarchical clustering works.**

ID	(1,2)	3	4	5
(1,2)	0	18	10	25
3	18	0	8	7
4	10	8	0	15
5	25	7	15	0

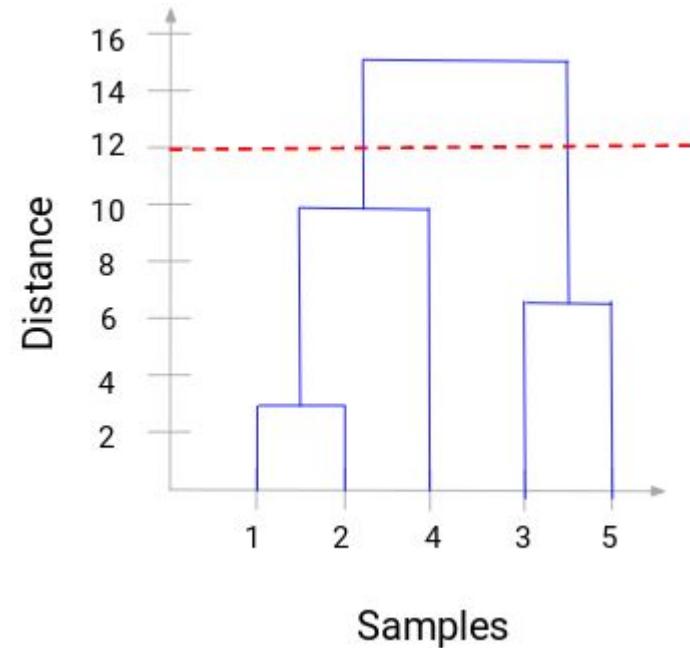
How should we Choose the Number of Clusters in Hierarchical Clustering?

More the distance of the vertical lines in the dendrogram, more the distance between those clusters.

Now, we can set a threshold distance and draw a horizontal line

(*Generally, we try to set the threshold in such a way that it cuts the tallest vertical line*).

The number of clusters will be the number of vertical lines which are being intersected by the line drawn using the threshold.

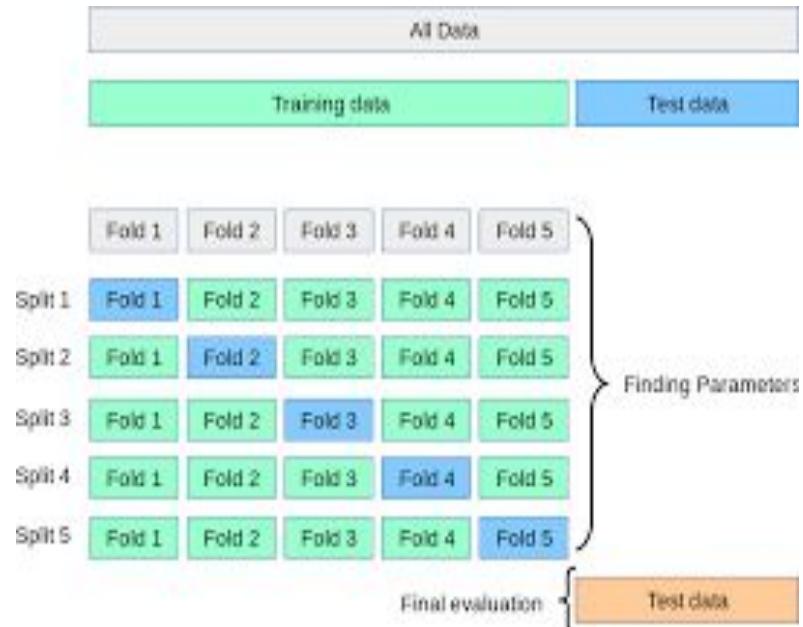


PRACTICAL

Cross Validation

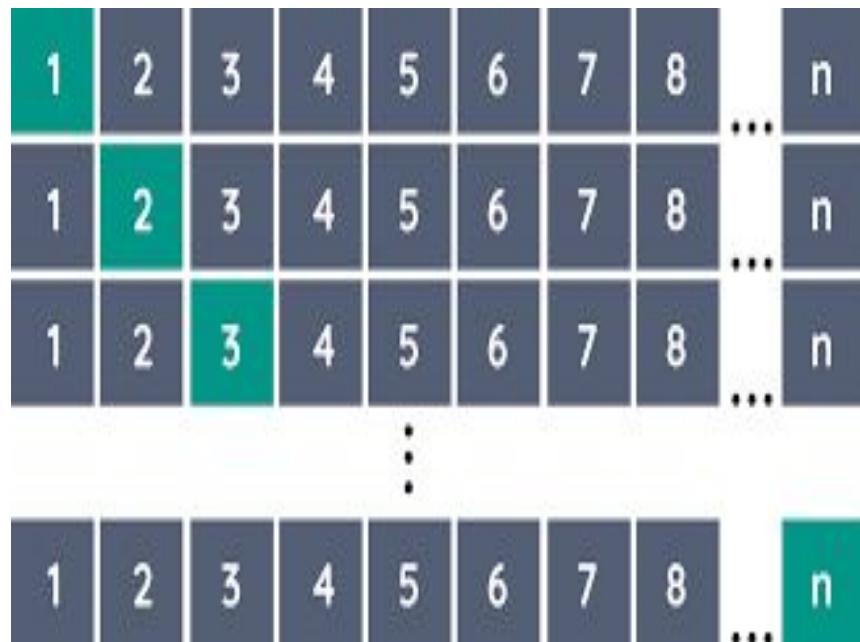
1. K-Fold
 - a. Split dataset into k consecutive folds (without shuffling by default).
2. LOOCV
 - a. Provides train/test indices to split data in train/test sets. Each sample is used once as a test set (singleton) while the remaining samples form the training set.

K-FOLD



```
class  
sklearn.model_selection.KFold(n_sp  
lits=5)
```

LOOCV



```
sklearn.model_selection.LeaveOne  
Out
```

PRACTICAL

BOOTSTRAP

th cross validation and bootstrapping are *resampling* methods.

- **bootstrap resamples with replacement** (and usually produces new "surrogate" data sets with the same number of cases as the original data set). Due to the drawing with replacement, a bootstrapped data set may contain multiple instances of the same original cases, and may completely omit other original cases.
- This is done by training the model on the sample and **evaluating the skill of the model on those samples not included in the sample**. These samples not included in a given sample are called the **out-of-bag samples**, or OOB for short.

```
# scikit-learn bootstrap
from sklearn.utils import resample
# data sample
data = [0.1, 0.2, 0.3, 0.4, 0.5, 0.6]
# prepare bootstrap sample
boot = resample(data, replace=True, n_samples=4,
random_state=1)
print('Bootstrap Sample: %s' % boot)
# out of bag observations
oob = [x for x in data if x not in boot]
```

Grid Search CV

GridSearchCV implements a “fit” and a “score” method.

```
class
```

```
sklearn.model_selection.GridSearchCV(estimator,  
param_grid, *, scoring=None, cv=None)
```

```
param_grid = [
```

```
    {'C': [1, 10, 100, 1000], 'kernel': ['linear']} ,
```

```
    {'C': [1, 10, 100, 1000], 'gamma': [0.001, 0.0001],  
     'kernel': ['rbf']} , ]
```

Randomized Search CV

RandomizedSearchCV implements a “fit” and a “score” method.

In contrast to GridSearchCV, not all parameter values are tried out, but rather a fixed number of parameter settings is sampled from the specified distributions.

class

```
sklearn.model_selection.RandomizedSearchCV(  
estimator, param_distributions, scoring=None, cv = None)
```

PRACTICAL

Ensemble Learning

Ensemble learning helps improve **machine learning** results by combining several models. **Ensemble** methods **are** meta-algorithms that combine several **machine learning** techniques into one predictive model in order to decrease variance (bagging), bias (boosting).

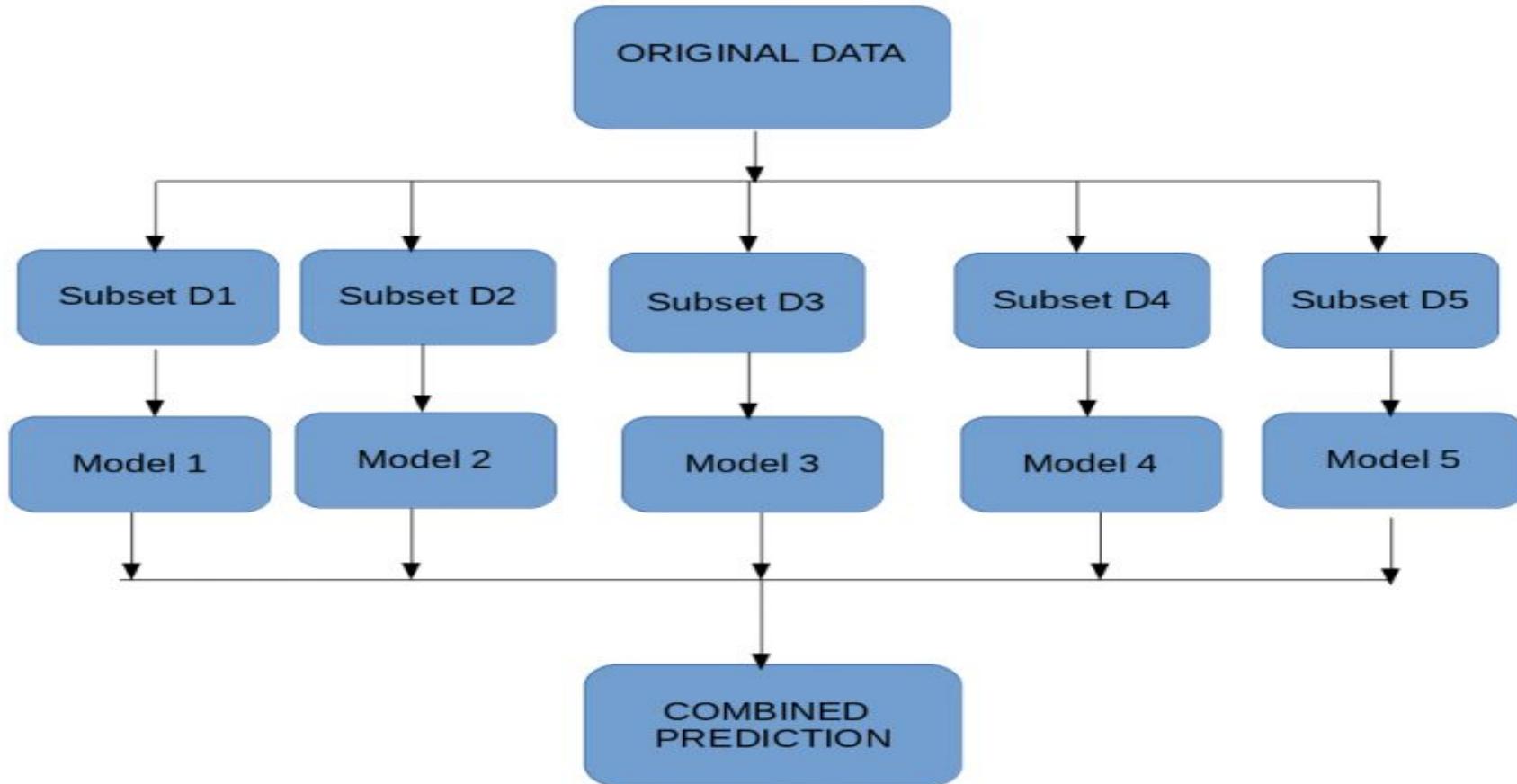
Bagging

- Random Forest

Boosting

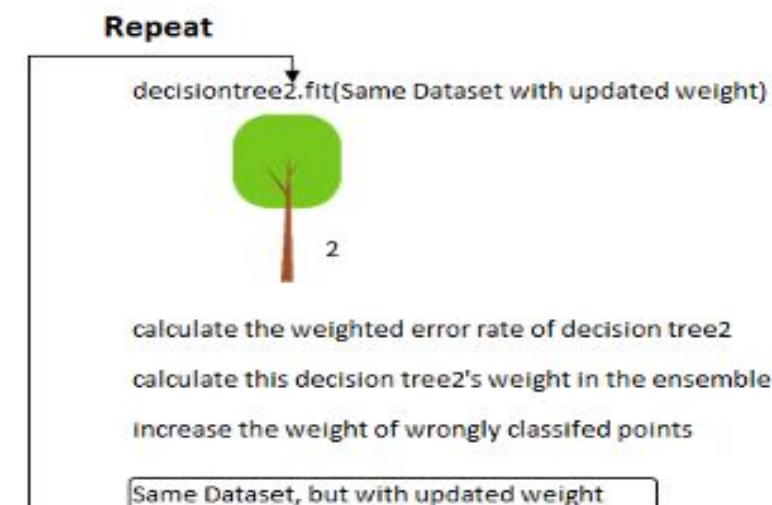
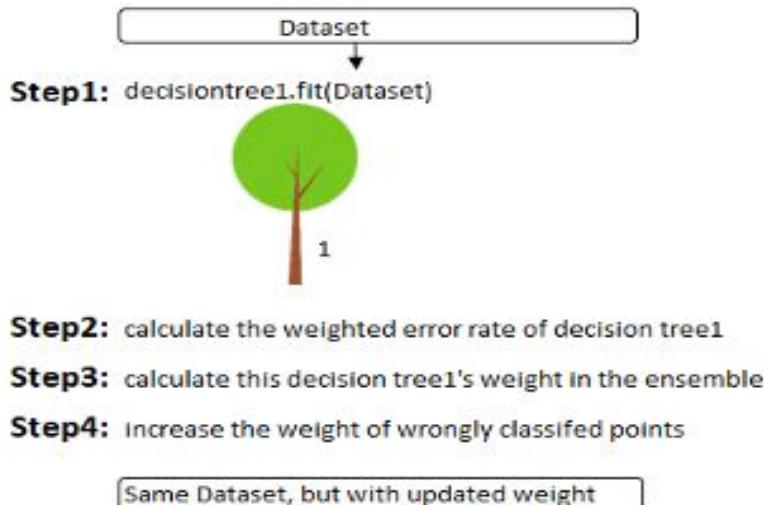
- AdaBoost
- XGBoost

Bagging



Boosting

Adaptive boosting or AdaBoost is one of the simplest boosting algorithms. Usually, decision trees are used for modelling. Multiple sequential models are created, each correcting the errors from the last model. AdaBoost assigns weights to the observations which are incorrectly predicted and the subsequent model works to predict these values correctly.

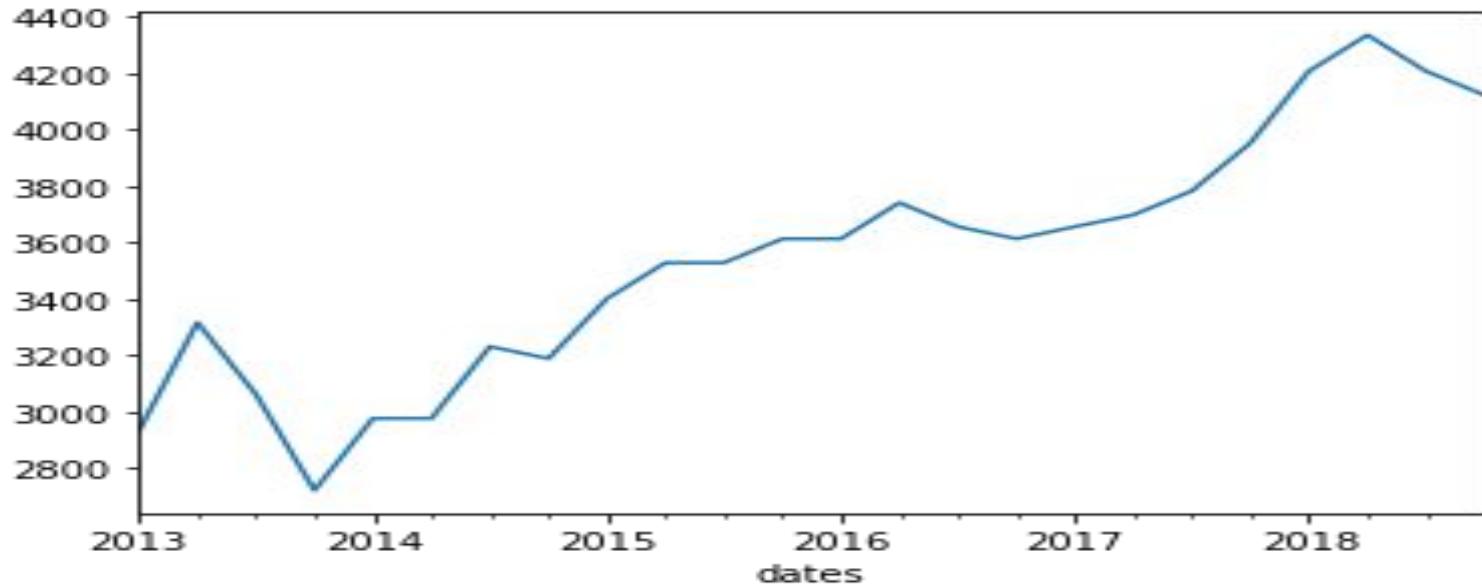


Case Study

Time - Series Forecasting

DATA:

TIME SERIES FOR APARTMENT PRICE IN VESU

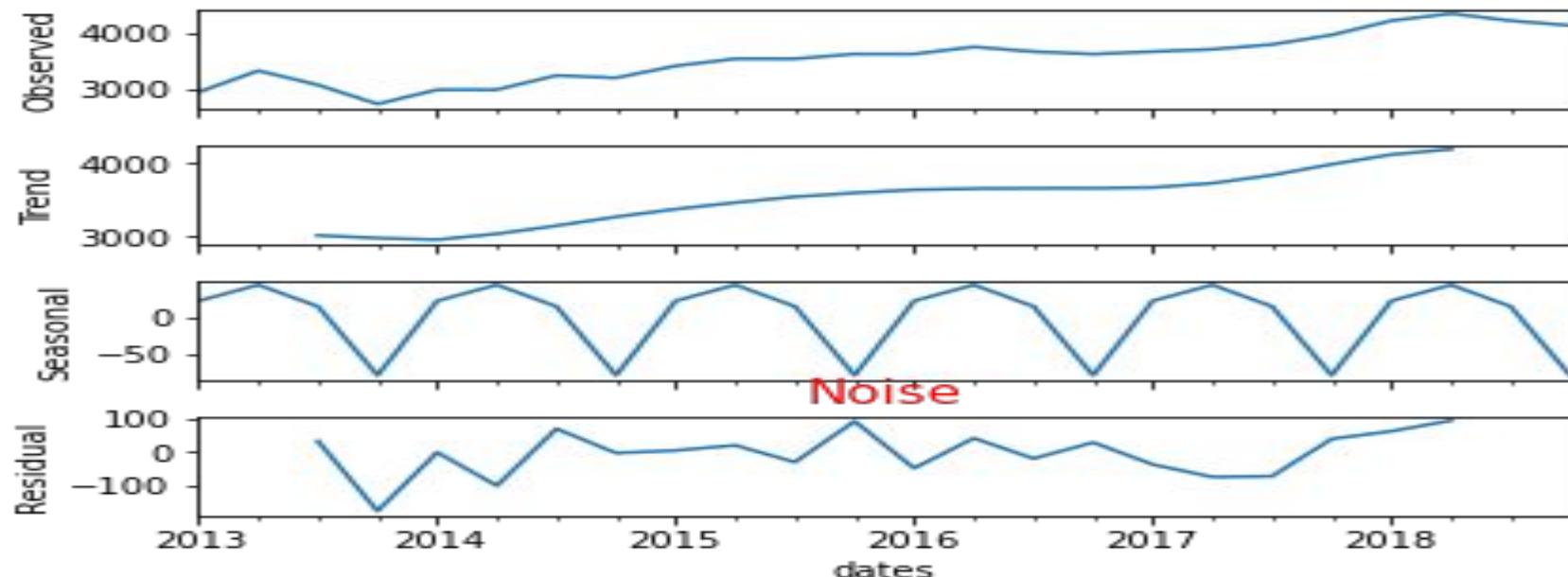


Time –Series Data:

- Sequence of Data Observed at Regular Time-Interval.
- For ex: Hourly, Daily, Weekly, Monthly, Quaterly, Yearly.

COMPONENTS OF TIME SERIES

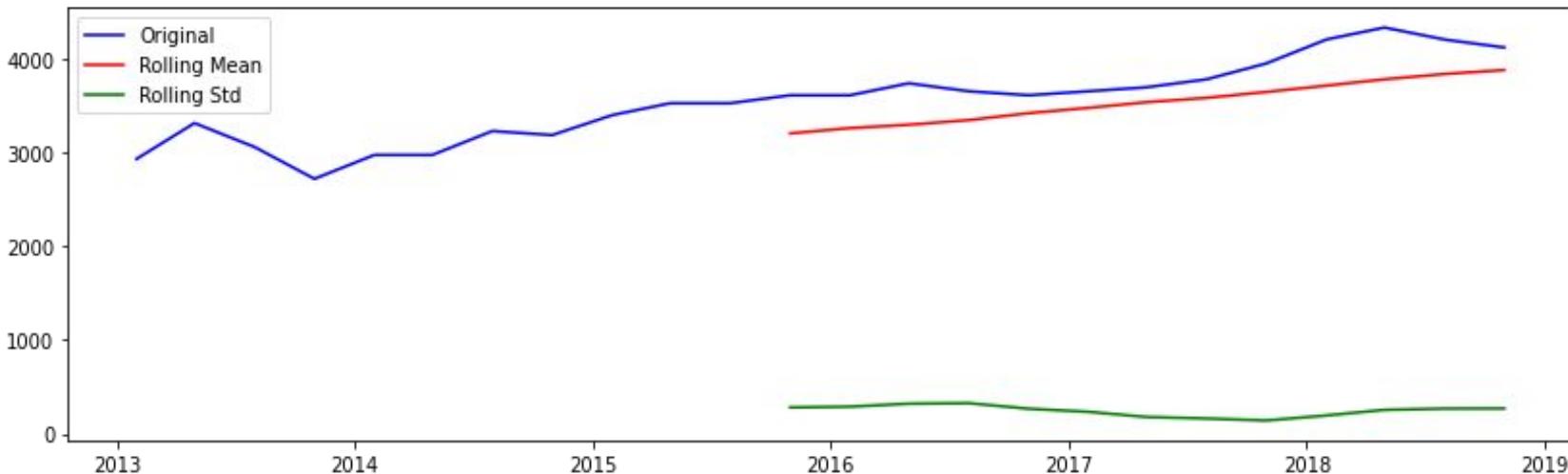
1. SEASONAL: REPEAT OVER A SPECIFIC PERIOD.
2. TREND: VARIATIONS THAT MOVE UP OR DOWN
3. CYCLICAL
4. RANDOM VARIATION: UNPREDICTABLE, ALSO CALLED NOISE



STATIONARITY

STATIONARY PROPERTIES OF TIME SERIES (DON'T CHANGE OVER TIME)
EX: CONSTANT "MEAN" AND "VARIANCE"

```
DickeyFuller Test
ADF Statistic: -0.48122925318079446
p-Value: 0.8956277242226353
Critical Values:
1% : -3.889
5% : -3.054
10% : -2.667
```



Time series forecasting with ARIMA

We are going to apply one of the most commonly used method for time-series forecasting, known as **ARIMA**, which stands for Autoregressive Integrated Moving Average. ARIMA

models are denoted with the notation **ARIMA (p, d, q)**. These three parameters

account for seasonality, trend, and noise in data

GENERAL TRANSFORMATIONS TO MAKE DATA STATIONARY:

1. Log
2. Difference
3. Square root, etc

Augmented DickeyFuller Test.

- This checks the data is stationary an important test before applying time-series models.

Null Hypothesis:

- Given Data is not Stationary, Hence if we consider 95% Confidence Interval the if obtained $p \leq 0.05$ we can reject Null Hypothesis which signifies that our data is "Stationary"

Note: After Transforming you will loose the original data, hence be sure that you have option to transform prediction Similar to Original original value Somehow

- Autocorrelation helps us study how each time series observation is related to its recent (or not so recent) past.

Forcasting time series using ARIMA

The ARIMA forecasting for stationary time series is nothing but linear equation(like linear regression).

- The predictor depend on (p, d, q) of Arima model.
- ACF and PACF grpahically summarizes the strength of relationship with an observation in timeseries with observations at prior times steps

AUTOCORRELATION:

Autocorrelation helps us study how each time series observation is related to its past.

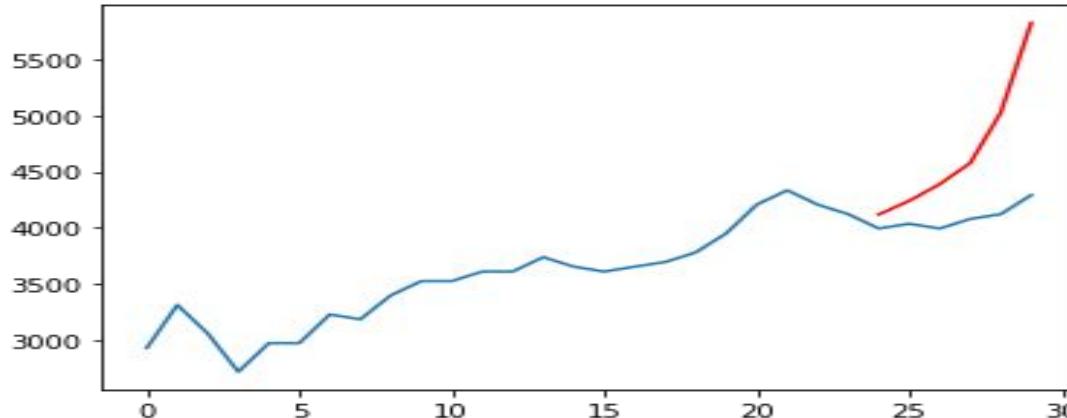
- **p** means the number of preceding (“lagged”) Y values that have to be added/subtracted to Y in the model, so as to make better predictions based on local periods of growth/decline in our data. This captures the “autoregressive” nature of ARIMA.
- **q** represents the number of preceding/lagged values for the error term that are added/subtracted to Y. This captures the “moving average” part of ARIMA

d represents the number of times that the data have to be “differenced” to produce a stationary signal (i.e., a signal that has a constant mean over time). This captures the “integrated” nature of ARIMA. If $d=0$, this means that our data does not tend to go up/down in the long term (i.e., the model is already “stationary”).

FITTING AN ARIMA MODEL

```
from statsmodels.tsa.arima_model import ARIMA  
model=ARIMA(series_diff[2:], order=(5,0,1)) #(p, d, q) if d=0, this model will be same as ARMA.  
result=model.fit()
```

```
result.forecast(6,)[0]
```



Apriori Algorithm

For finding frequent itemsets in a dataset

Apriori because it uses prior knowledge of frequent itemset properties.

What is Frequent Itemset?

Frequent itemsets are those items whose support is greater than the threshold value or user-specified minimum support. It means if A & B are the frequent itemsets together, then individually A and B should also be the frequent itemset.

Suppose there are the two transactions: A= {1,2,3,4,5}, and B= {2,3,7}, in these two transactions, 2 and 3 are the frequent itemsets.

Steps for Apriori Algorithm

Step-1: Determine the support of itemsets

Step-2: Take all supports in the transaction with higher support value than the minimum

Step-3: Find all the rules of these subsets that have higher confidence value than the threshold

Step-4: Sort the rules as the decreasing order of lift.

TID	ITEMSETS
T1	A, B
T2	B, D
T3	B, C
T4	A, B, D
T5	A, C
T6	B, C
T7	A, C
T8	A, B, C, E
T9	A, B, C

Given: Minimum Support= 2, Minimum Confidence= 50%

Itemset	Support_Count
A	6
B	7
C	5
D	2
E	1

Itemset	Support_Count
A	6
B	7
C	5
D	2

**As per Minimum support
is 02**

Itemset	Support_Count
{A, B}	4
{A, C}	4
{A, D}	1
{B, C}	4
{B, D}	2
{C, D}	0

Itemset	Support_Count
{A, B}	4
{A, C}	4
{B, C}	4
{B, D}	2

A, B, C, D

Itemset	Support_Count
{A, B, C}	2
{B, C, D}	1
{A, C, D}	0
{A, B, D}	0

Finding the association rules for the subsets?

To generate the association rules, first, we will create a new table with the possible rules from the occurred combination {A, B,C}. For all the rules, we will calculate the Confidence using formula **sup(A ^B)/A**. After calculating the confidence value for all rules, we will exclude the rules that have less confidence than the minimum threshold(50%).

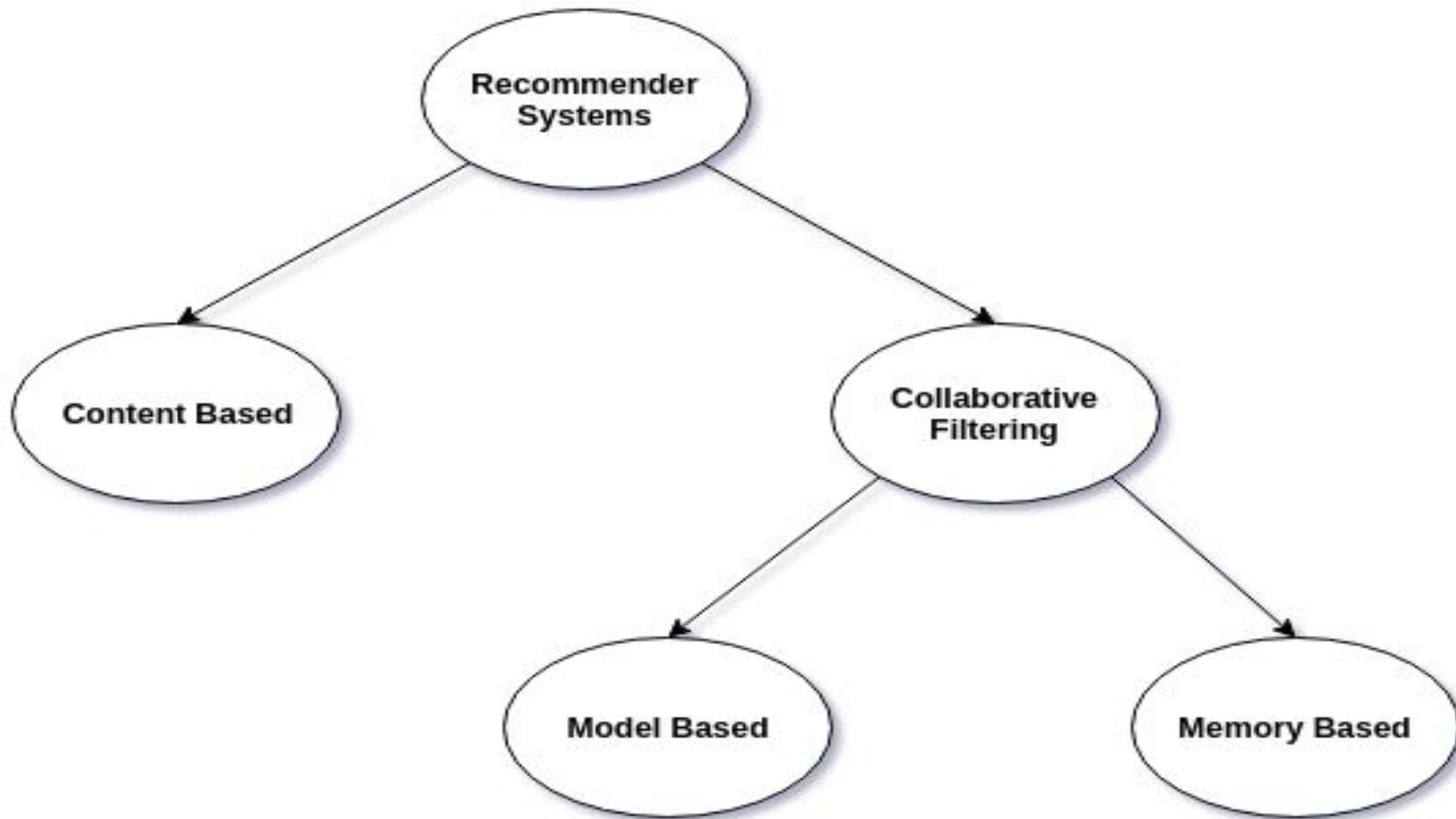
$$Support = \frac{frq(X, Y)}{N}$$

Rule: $X \Rightarrow Y$  Confidence = $\frac{frq(X, Y)}{frq(X)}$


$$Lift = \frac{Support}{Supp(X) \times Supp(Y)}$$

Rules	Support	Confidence
$A \wedge B \rightarrow C$	2	$\text{Sup}\{(A \wedge B) \wedge C\}/\text{sup}(A \wedge B) = 2/4=0.5=50\%$
$B \wedge C \rightarrow A$	2	$\text{Sup}\{(B \wedge C) \wedge A\}/\text{sup}(B \wedge C) = 2/4=0.5=50\%$
$A \wedge C \rightarrow B$	2	$\text{Sup}\{(A \wedge C) \wedge B\}/\text{sup}(A \wedge C) = 2/4=0.5=50\%$
$C \rightarrow A \wedge B$	2	$\text{Sup}\{C \wedge (A \wedge B)\}/\text{sup}(C) = 2/5=0.4=40\%$
$A \rightarrow B \wedge C$	2	$\text{Sup}\{A \wedge (B \wedge C)\}/\text{sup}(A) = 2/6=0.33=33.33\%$
$B \rightarrow B \wedge C$	2	$\text{Sup}\{B \wedge (B \wedge C)\}/\text{sup}(B) = 2/7=0.28=28\%$

Recommendation System



Collaborative filtering

methods for

recommender systems are methods that are solely based on the past interactions between users and the target items. Thus, the input to a collaborative filtering system will be all historical data of user interactions with target items. This data is typically stored in a matrix where the rows are the users, and the columns are the items.

The core idea behind such systems is that the historical data of the users should be enough to make a prediction. I.e we don't need anything more than that historical data, no extra push from the user, no presently trending information, etc.

Content-based

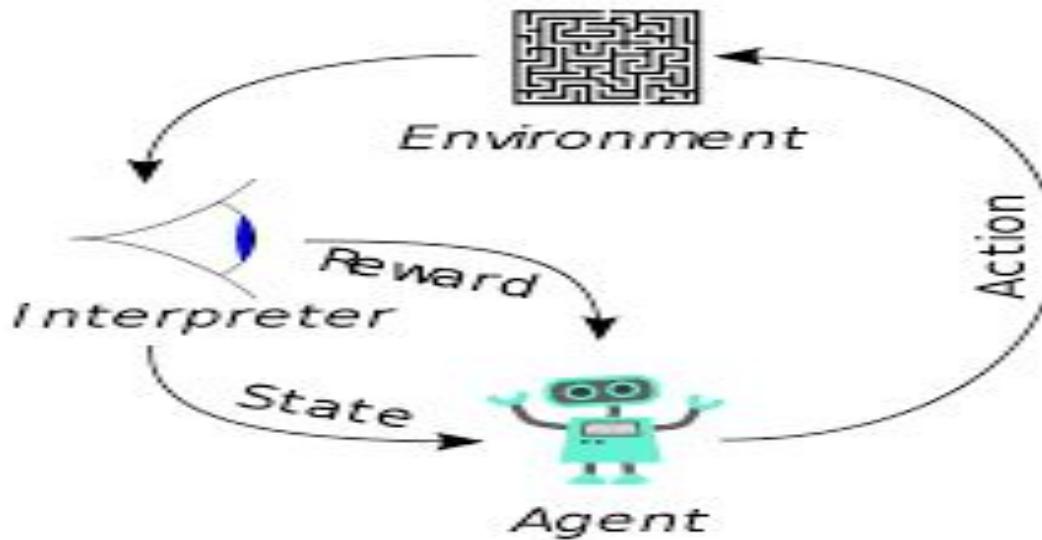
In contrast to collaborative filtering, content-based approaches will use additional information about the user and / or items to make predictions.

Content-based recommenders treat **recommendation** as a user-specific classification problem and learn a classifier for the user's likes and dislikes **based** on an item's features. In this system, keywords are used to describe the items and a user profile is built to indicate the **type** of item this user likes.

Practical

Reinforcement Learning

REINFORCEMENT LEARNING



- In this type of Machine Learning, there is an **agent** which tries to learn what **action** it should take for a given state in the **environment** in order to maximize the cumulative **Reward**.
- In short Learning through **Experience**.

Actions: Actions are the *Agent's* methods which allow it to interact and change its *environment*, and thus transfer between *states*. Every action performed by the Agent yields a *reward* from the environment. The decision of which action to choose is made by the *policy*.

Agent: The learning and *acting* part of a *Reinforcement Learning* problem, which tries to maximize the *rewards* it is given by the *Environment*.

Environment: Everything which isn't the *Agent*; everything the Agent can interact with, either directly or indirectly. The environment changes as the Agent performs *actions*; every such change is considered a *state*-transition. Every action the Agent performs yields a *reward* received by the Agent.

Thomson Sampling

Thompson Sampling makes use of Probability Distribution and Bayes Theorem to generate success rate distributions.

Thompson Sampling is also sometimes referred to as Posterior Sampling or Probability Matching.
Follows exploration and exploitation.

New choices are explored to maximize rewards while exploiting the already explored choices.

Basic Intuition Behind Thompson Sampling

1. To begin with, all machines are assumed to have a uniform distribution of the probability of success, in this case getting a reward
2. For each observation obtained from a Slot machine, based on the reward a new distribution is generated with probabilities of success for each slot machine
3. Further observations are made based on these prior probabilities obtained on each round or observation which then updates the success distributions.
4. After sufficient observations, each slot machine will have a success distribution associated with it which can help the player in choosing the machines wisely to get the maximum rewards.

Multi-Arm Bandits

Bandits: Formally named “k-Armed Bandits” after the nickname “one-armed bandit” given to slot-machines, these are considered to be the simplest type of **Reinforcement Learning** tasks. Bandits have no different **states**, but only one — and the **reward** taken under consideration is only the immediate one.

Hence, bandits can be thought of as having single-state *episodes*. Each of the k-arms is considered an **action**, and the objective is to learn the *policy* which will maximize the expected reward after each action (or arm-pulling).



Practical

UCB

Step 1: Two values are considered for each round of exploration of a machine

1. The number of times each machine has been selected till round n
2. The sum of rewards collected by each machine till round n

Step 2: At each round, we compute the average reward and the confidence interval of the machine i up to n rounds as follows:

$$\bar{r}_i(n) = \frac{R_i(n)}{N_i(n)}$$

$$[\bar{r}_i(n) - \Delta_i(n), \bar{r}_i(n) + \Delta_i(n)]$$

$$\Delta_i(n) = \sqrt{\frac{3 \log(n)}{2 N_i(n)}}$$

Average

Confidence Interval

$$\bar{r}_i(n) + \Delta_i(n)$$

UCB

UCB Practical

Q-learning

Q-learning is an off policy reinforcement learning algorithm that seeks to find the best action to take given the current state. It's considered off-policy because the **q-learning** function learns from actions that are outside the current policy, like taking random actions, and therefore a policy isn't needed.

What is Q?

Q - stands for the quality, how useful a given action is in gaining rewards (Finding Optimal Path)

Create a q-table

When q-learning is performed we create what's called a *q-table* or matrix that follows the shape of [state, action] and we initialize our values to zero. We then update and store our *q-values* after an episode. This q-table becomes a reference table for our agent to select the best action based on the q-value.

Q-learning and making updates

The next step is simply for the agent to interact with the environment and make updates to the state

action pairs in our q-table $Q[\text{state},$

$\text{action}]$.

Here are the 3 basic steps:

1. Agent starts in a state (s_1) takes an action (a_1) and receives a reward (r_1)
2. Agent selects action by referencing Q-table with highest value (max) **OR** by random (epsilon, ε)
3. Update q-values

```
# Update q values
```

```
Q[state, action] = Q[state, action] + lr *  
(reward + gamma * np.max(Q[new_state, :]) -  
Q[state, action])
```

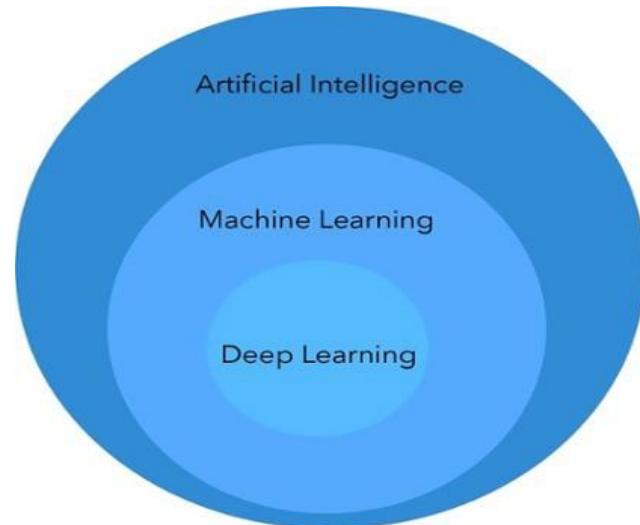
Module 06

AI and Deep Learning

Intro To Deep Learning

Deep learning is part of a broader family of machine learning methods based on artificial neural networks with representation learning.

Deep Learning is a subfield of machine learning concerned with algorithms inspired by the structure and function of the brain called artificial neural networks.



DL VS ML

1. The main difference between deep learning and machine learning is due to the way data is presented in the system. Machine learning algorithms almost always require structured data, while deep learning networks rely on layers of ANN (artificial neural networks).
2. Machine learning algorithms are designed to “learn” to act by understanding labeled data and then use it to produce new results with more datasets. However, when the result is incorrect, there is a need to “teach them”.

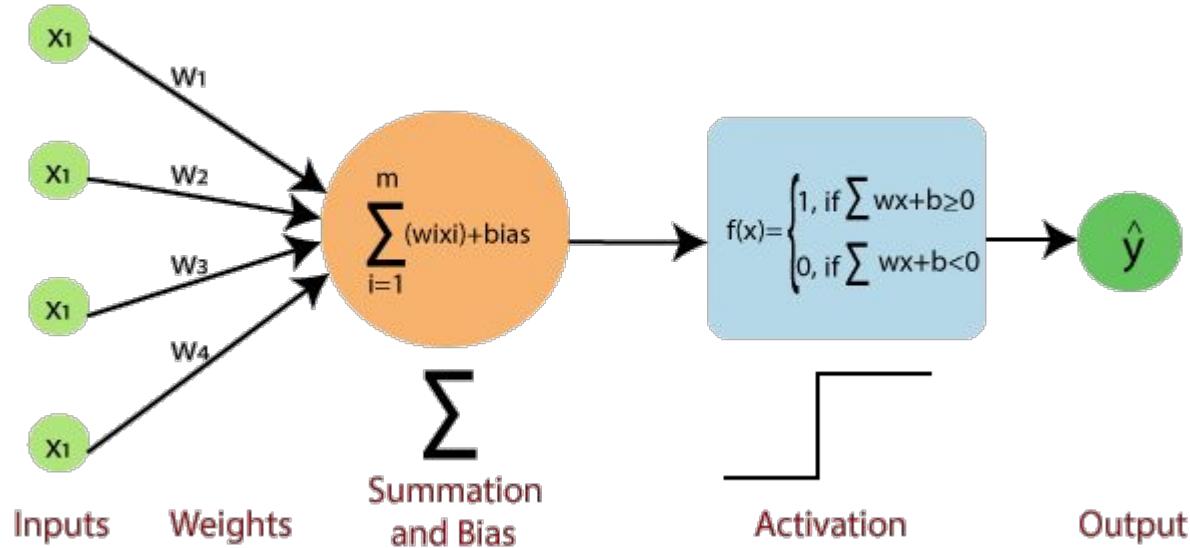
1. Deep learning networks do not require human intervention, as multilevel layers in neural networks place data in a hierarchy of different concepts, which ultimately learn from their own mistakes. However, even they can be wrong if the data quality is not good enough.
2. Data decides everything. It is the quality of the data that ultimately determines the quality of the result.

DL VS Human Brain

Deep learning consists of artificial neural networks that are modeled on similar networks present in the human brain. As data travels through this artificial mesh, each layer processes an aspect of the data, filters outliers, spots familiar entities, and produces the final output.

The human brain functions in a similar fashion — but only at a highly advanced level. The human brain is a far more complex web of diverse neurons where each node performs a separate task. Our understanding of things is far more superior. If we are taught that lions are dangerous, we can deduce that bears are too.

Single Layer perceptron



Working

Single layer perceptron is the first proposed neural model created. The content of the local memory of the neuron consists of a vector of weights. The computation of a single layer perceptron is performed over the calculation of sum of the input vector each with the value multiplied by corresponding element of vector of the weights. The value which is displayed in the output will be the input of an activation function.

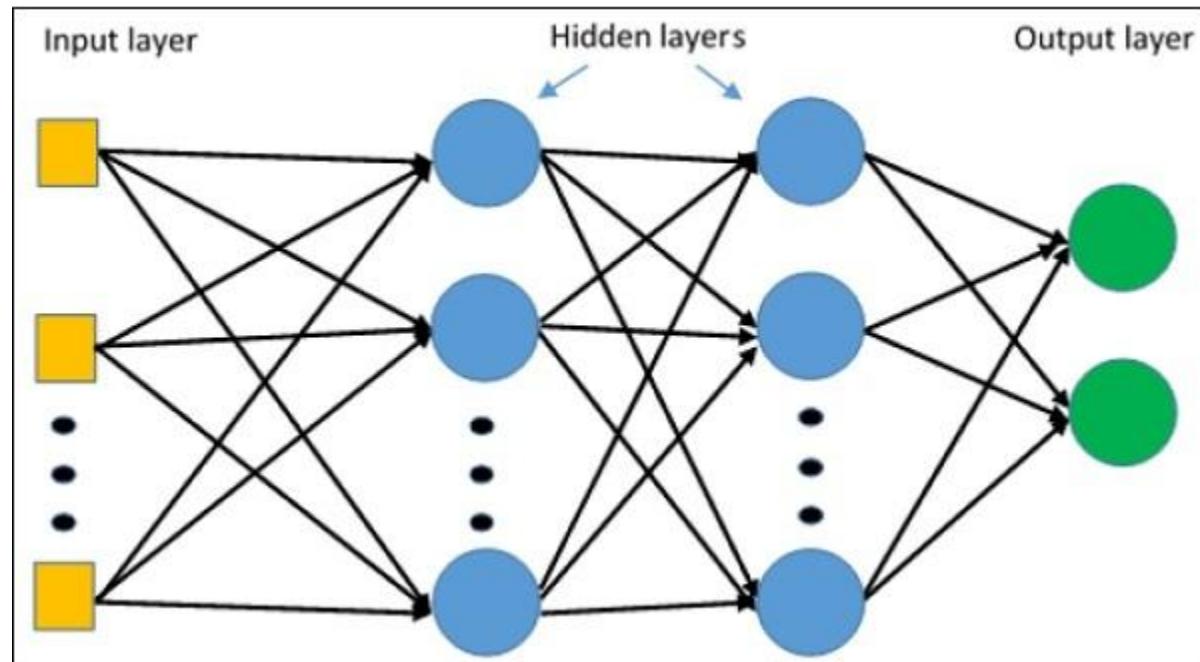
Tacking Activation function as Sigmoid function

- The weights are initialized with random values at the beginning of the training.
- For each element of the training set, the error is calculated with the difference between desired output and the actual output. The error calculated is used to adjust the weights.
- The process is repeated until the error made on the entire training set is not less than the specified threshold, until the maximum number of iterations is reached.

Practical

Multilayer Perceptron

Multi-Layer perceptron defines the most complicated architecture of artificial neural networks. It is substantially formed from multiple layers of perceptron.



Intro

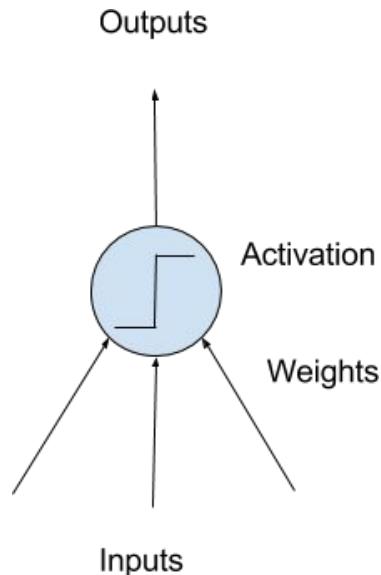
The field of artificial neural networks is often just called neural networks or multi-layer perceptrons after perhaps the most useful type of neural network. A perceptron is a single neuron model that was a precursor to larger neural networks.

The predictive capability of neural networks comes from the hierarchical or multi-layered structure of the networks. The data structure can pick out (learn to represent) features at different scales or resolutions and combine them into higher-order features. For example from lines, to collections of lines to shapes.

What is Neurons.....

The building block for neural networks are artificial neurons.

These are simple computational units that have weighted input signals and produce an output signal using an activation function.

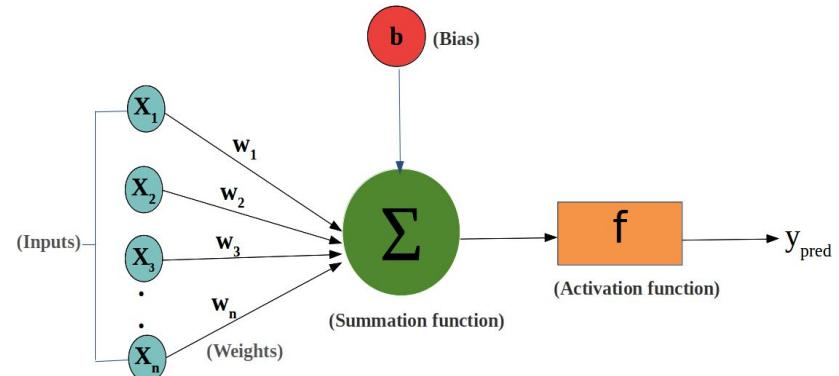


You may be familiar with linear regression, in which case the weights on the inputs are very much like the coefficients used in a regression equation.

Like linear regression, each neuron also has a bias which can be thought of as an input that always has the value 1.0 and it too must be weighted.

For example, a neuron may have two inputs in which case it requires three weights. One for each input and one for the bias.

Weights are often initialized to small random values, such as values in the range 0 to 0.3, although more complex initialization schemes can be used.

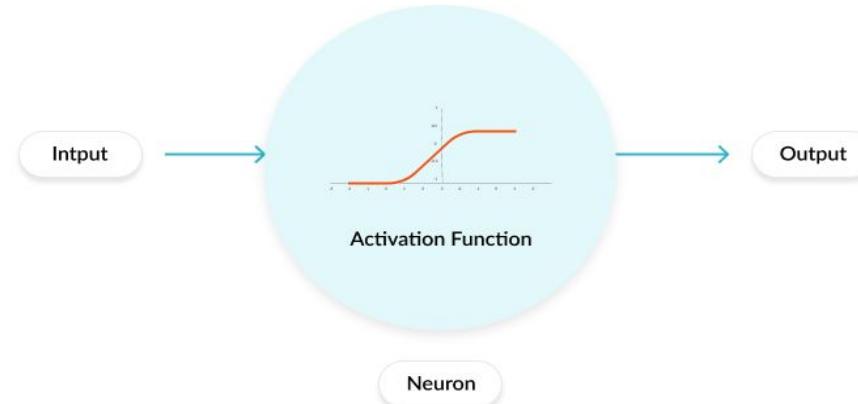


Activation Function

The weighted inputs are summed and passed through an activation function, sometimes called a transfer function.

An activation function is a simple mapping of summed weighted input to the output of the neuron. It is called an activation function because it governs the threshold at which the neuron is activated and strength of the output signal.

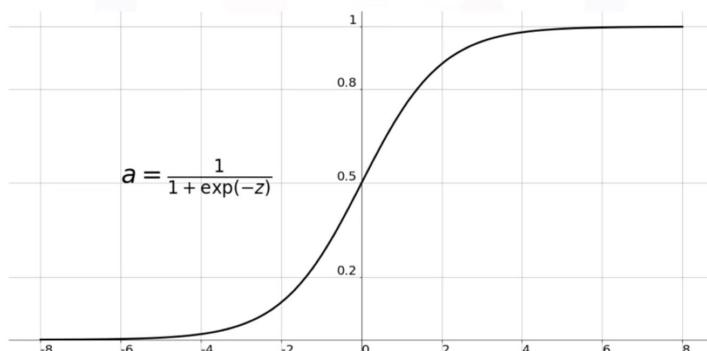
Historically simple step activation functions were used where if the summed input was above a threshold, for example 0.5, then the neuron would output a value of 1.0, otherwise it would output a 0.0.



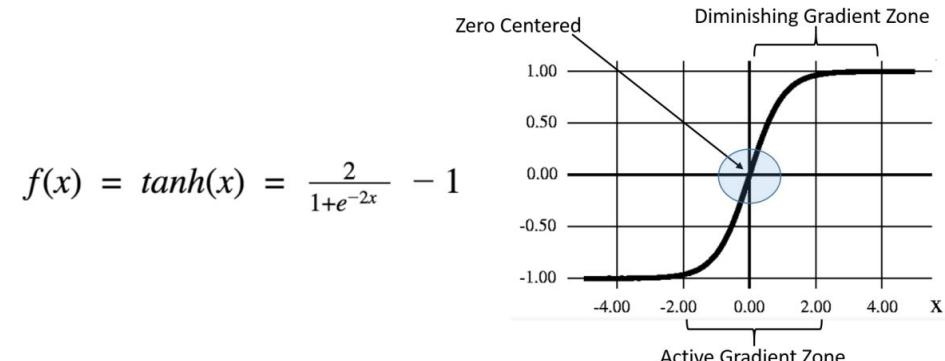
Traditionally non-linear activation functions are used. This allows the network to combine the inputs in more complex ways and in turn provide a richer capability in the functions they can model. Non-linear functions like the logistic also called the sigmoid function were used that output a value between 0 and 1 with an s-shaped distribution, and the hyperbolic tangent function also called tanh that outputs the same distribution over the range -1 to +1.

More recently the rectifier activation function has been shown to provide better results.

Sigmoid Function



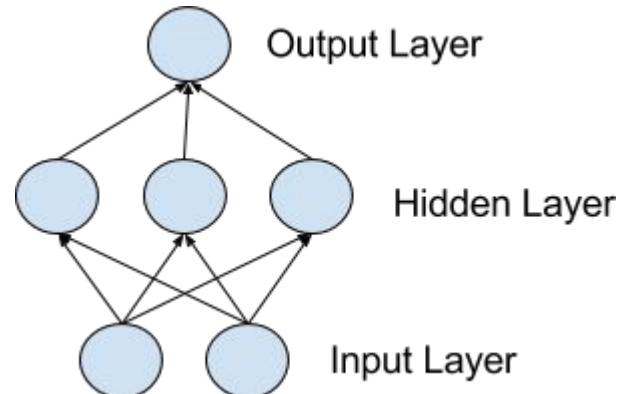
Tanh Function



Layers In DL

Neurons are arranged into networks of neurons.

A row of neurons is called a layer and one network can have multiple layers. The architecture of the neurons in the network is often called the network topology.



Input Layer

The bottom layer that takes input from your dataset is called the visible layer, because it is the exposed part of the network. Often a neural network is drawn with a visible layer with one neuron per input value or column in your dataset. These are not neurons as described above, but simply pass the input value through to the next layer.

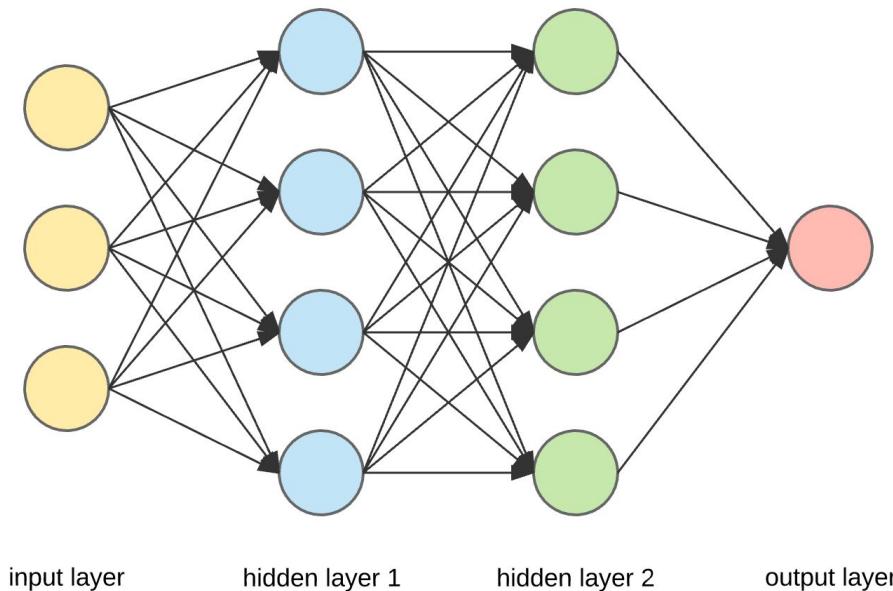
Hidden Layer

Layers after the input layer are called hidden layers because they are not directly exposed to the input. The simplest network structure is to have a single neuron in the hidden layer that directly outputs the value.

Given increases in computing power and efficient libraries, very deep neural networks can be constructed. Deep learning can refer to having many hidden layers in your neural network. They are deep because they would have been unimaginably slow to train historically, but may take seconds or minutes to train using modern techniques and hardware.

Output Layer

The final hidden layer is called the output layer and it is responsible for outputting a value or vector of values that correspond to the format required for the problem.



Choice Of Activation function

The choice of activation function in The output layer is strongly constrained by the type of problem that you are modeling. For example:

- A regression problem may have a single output neuron and the neuron may have no activation function.
- A binary classification problem may have a single output neuron and use a sigmoid activation function to output a value between 0 and 1 to represent the probability of predicting a value for the class 1. This can be turned into a crisp class value by using a threshold of 0.5 and snap values less than the threshold to 0 otherwise to 1.
- A multi-class classification problem may have multiple neurons in the output layer, one for each class
- In this case a softmax activation function may be used to output a probability of the network predicting each of the class values. Selecting the output with the highest probability can be used to produce a crisp class classification value.

Types of Activation function

Sigmoid Function

In an ANN, the sigmoid function is a non-linear AF used primarily in feedforward neural networks. It is a differentiable real function, defined for real input values, and containing positive derivatives everywhere with a specific degree of smoothness. The sigmoid function appears in the output layer of the deep learning models and is used for predicting probability-based outputs. The sigmoid function is represented as:

$$f(x) = \left(\frac{1}{(1 + exp^{-x})} \right) - (1.4)$$

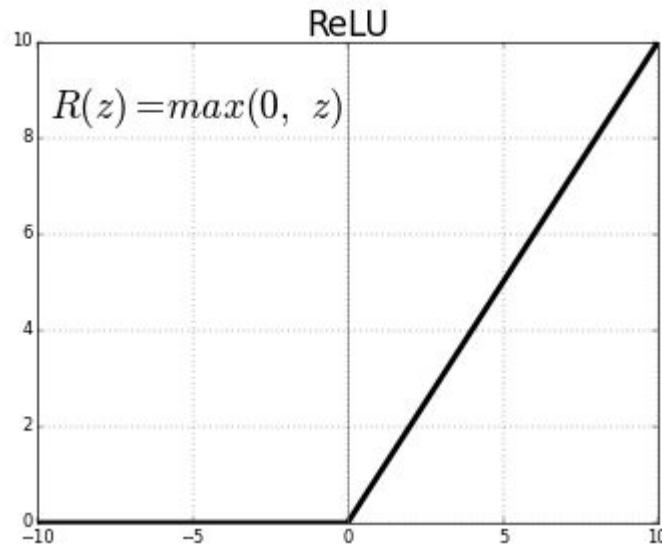
Relu Activation

Max(0,Z) ,

If Negative == 0

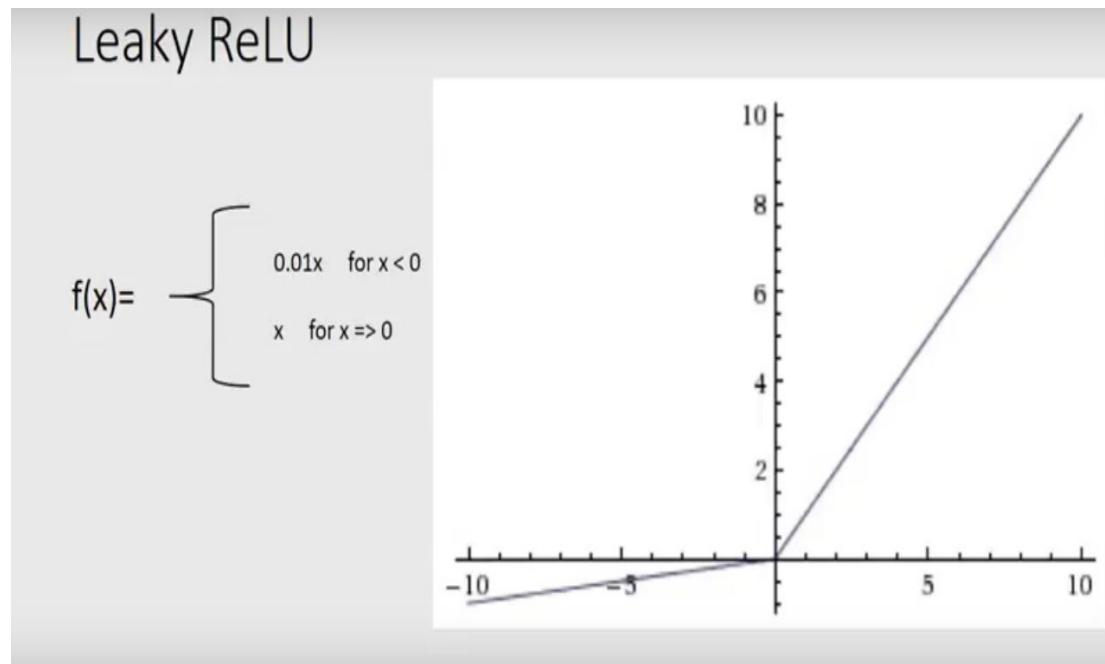
Else

All The values of Z Or input value

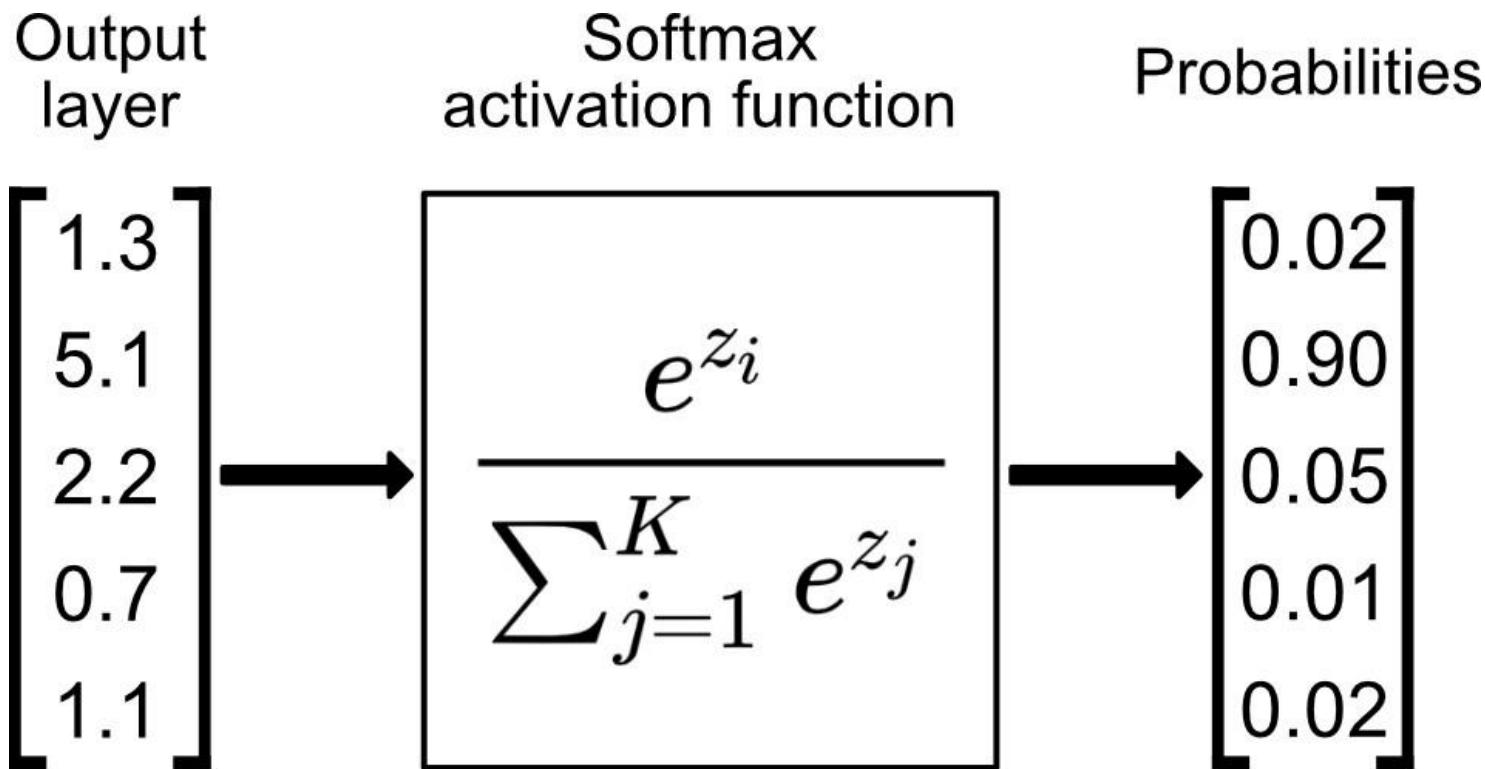


Leaky relu

$$\max(0.01 * Z, Z)$$



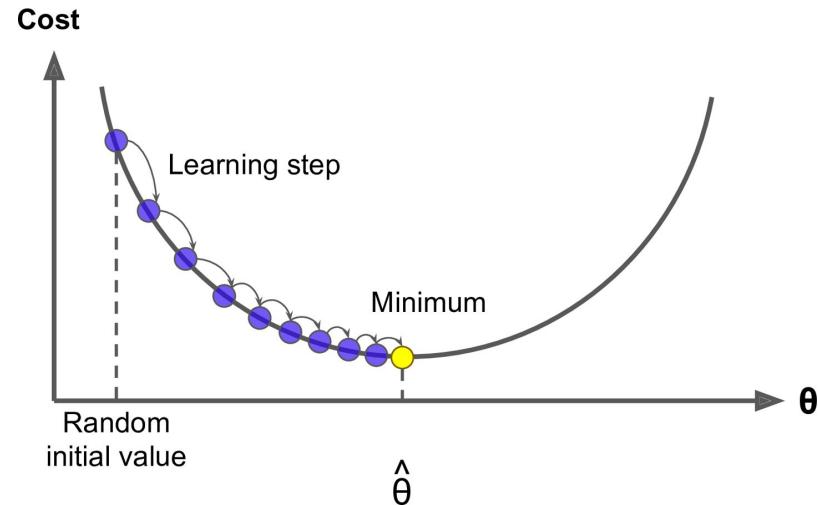
SoftMax



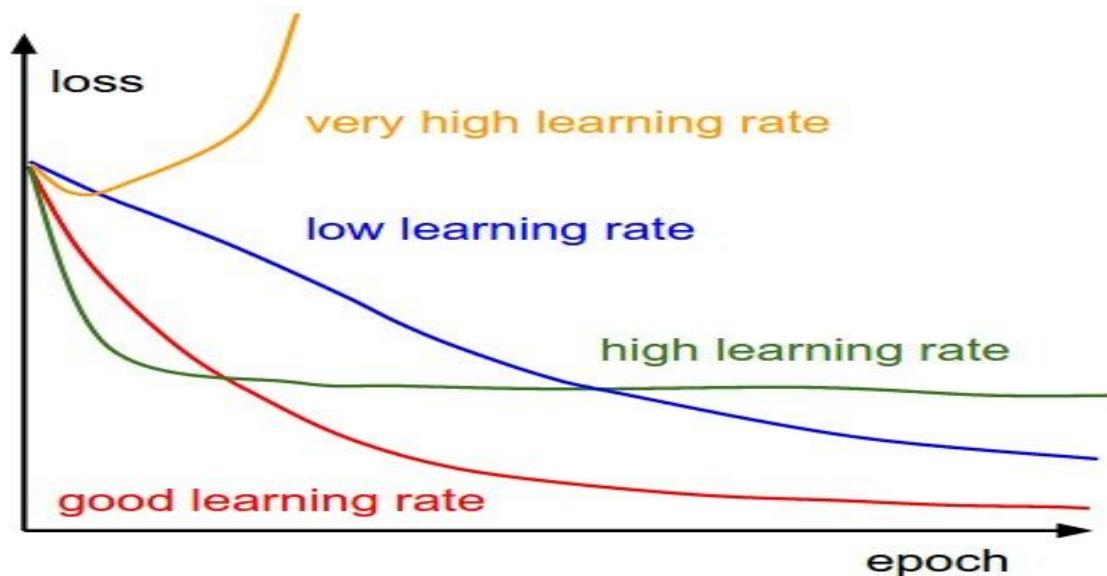
What Is Learning Rate

the learning rate is a configurable hyperparameter used in the training of neural networks that has a small positive value, often in the range between 0.0 and 1.0.

The learning rate controls how quickly the model is adapted to the problem. Smaller learning rates require more training epochs given the smaller changes made to the weights each update, whereas larger learning rates result in rapid changes and require fewer training epochs.



A learning rate that is too large can cause the model to converge too quickly to a suboptimal solution, whereas a learning rate that is too small can cause the process to get stuck.

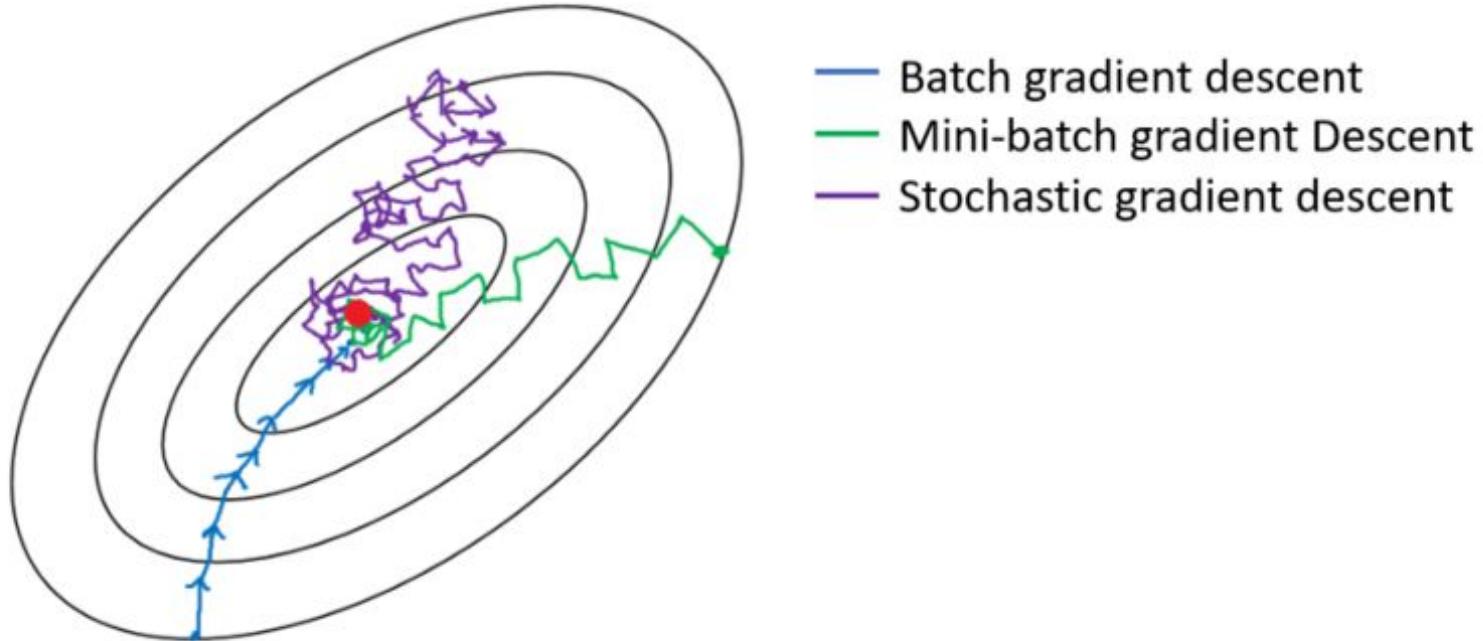


What Is Batch

The batch size is a hyperparameter that defines the number of samples to work through before updating the internal model parameters.

Think of a batch as a for-loop iterating over one or more samples and making predictions. At the end of the batch, the predictions are compared to the expected output variables and an error is calculated. From this error, the update algorithm is used to improve the model, e.g. move down along the error gradient

When all training samples are used to create one batch, the learning algorithm is called batch gradient descent. When the batch is the size of one sample, the learning algorithm is called stochastic gradient descent. When the batch size is more than one sample and less than the size of the training dataset, the learning algorithm is called mini-batch gradient descent.

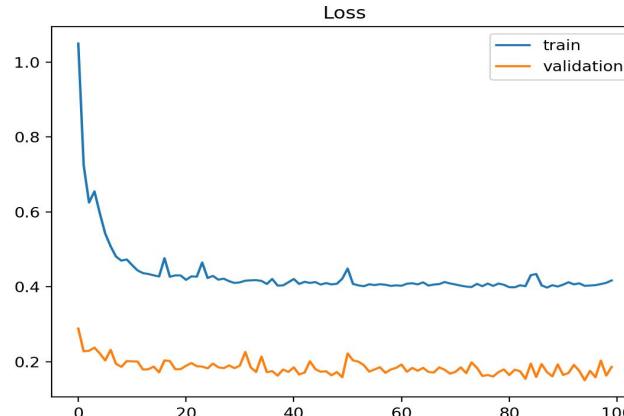


Epoch

The number of epochs is a hyperparameter that defines the number times that the learning algorithm will work through the entire training dataset.

One epoch means that each sample in the training dataset has had an opportunity to update the internal model parameters. An epoch is comprised of one or more batches. an epoch that has one batch is called the batch gradient descent learning algorithm.

It is common to create line plots that show epochs along the x-axis as time and the error or skill of the model on the y-axis. These plots are sometimes called learning



Loss Function

with neural networks, we seek to minimize the error. As such, the objective function is often referred to as a cost function or a loss function and the value calculated by the loss function is referred to as simply “*loss*.”

The cost or loss function has an important job in that it must faithfully distill all aspects of the model down into a single number in such a way that improvements in that number are a sign of a better model.

It is important, therefore, that the function faithfully represent our design goals. If we choose a poor error function and obtain unsatisfactory results, the fault is ours for badly specifying the goal of the search.

Choice OF Loss Function

The choice of cost function is tightly coupled with the choice of output unit. Most of the time, we simply use the cross-entropy between the data distribution and the model distribution. The choice of how to represent the output then determines the form of the cross-entropy function

Regression Problem

- **Loss Function:** Mean Squared Error (MSE).

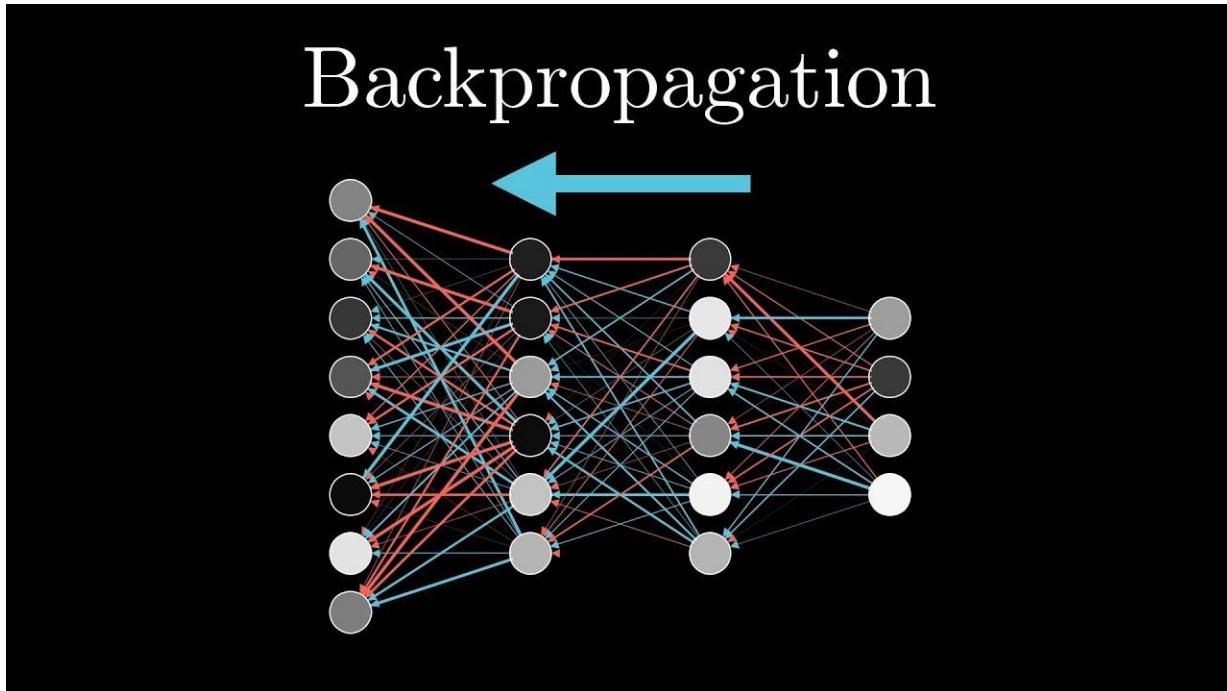
Binary Classification Problem

- **Loss Function:** Cross-Entropy

Multi-Class Classification Problem

- **Loss Function:** MultiClass Cross-Entropy,

BackPropagation



What is BackPropagation

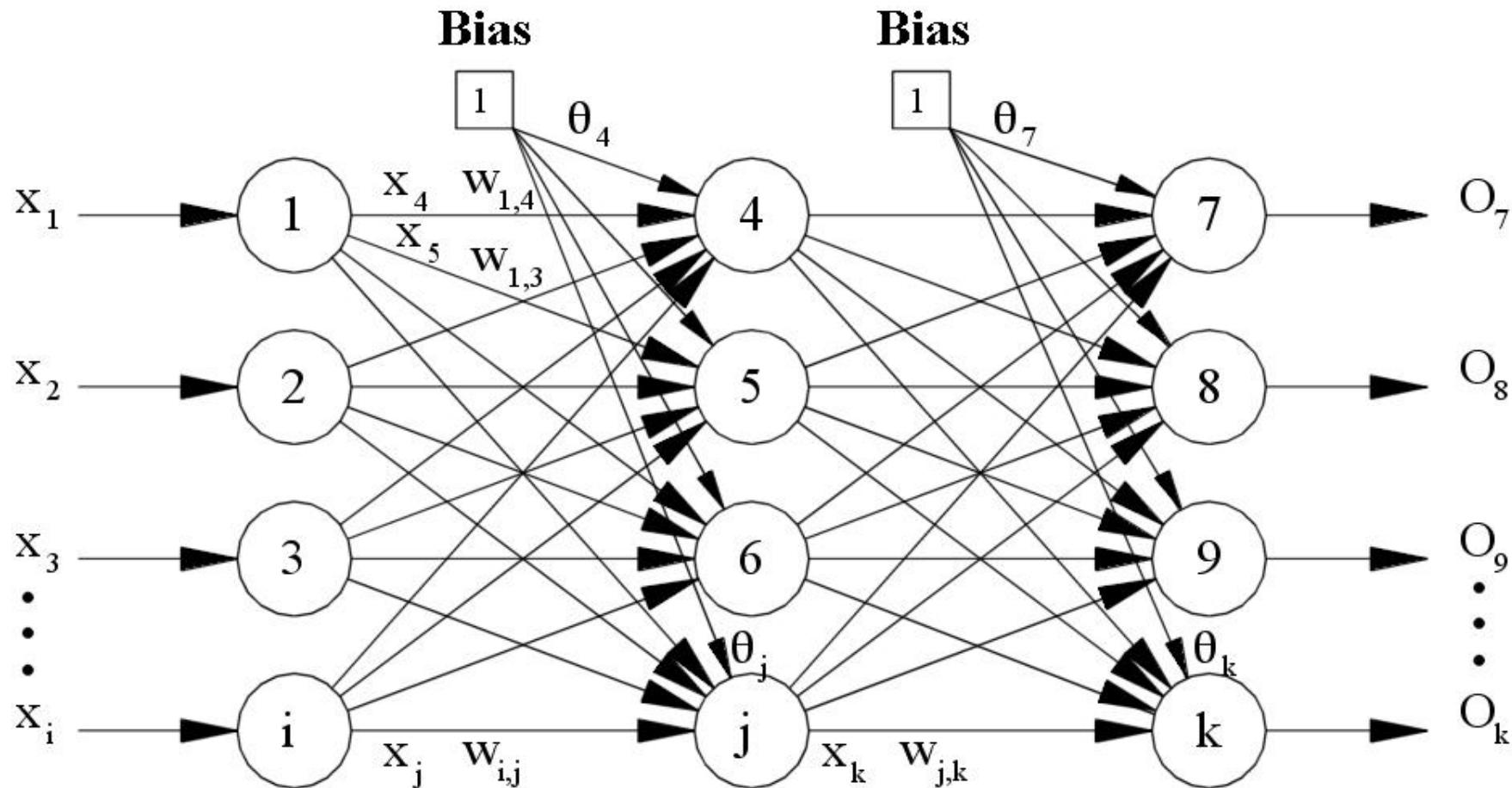
Back-propagation is the essence of neural net training. It is the method of fine-tuning the weights of a neural net based on the error rate obtained in the previous epoch (i.e., iteration). Proper tuning of the weights allows you to reduce error rates and to make the model reliable by increasing its generalization.

It is a standard method of training artificial neural networks. This method helps to calculate the gradient of a loss function with respects to all the weights in the network.

How Does It works

1. Inputs X, arrive through the preconnected path
2. Input is modeled using real weights W. The weights are usually randomly selected.
3. Calculate the output for every neuron from the input layer, to the hidden layers, to the output layer.
4. Calculate the error in the outputs
5. Travel back from the output layer to the hidden layer to adjust the weights such that the error is decreased.

Keep repeating the process until the desired output is achieved



Update the Weight Of W

Equations : $W1 = W1 - \text{Alpha} * DL/DW1$

DL / DW = Chain Rule Equation

Like $DL / DW1 = DL / DZ * DZ / DW1$: Till You Requires W Point

Same For Bias

$B = B - \text{Alpha} * DL/DB$

Optimizers

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.

How you should change your weights or learning rates of your neural network to reduce the losses is defined by the optimizers you use. Optimization algorithms or strategies are responsible for reducing the losses and to provide the most accurate results possible.

Types

Gradient Descent

Stochastic Gradient Descent

Mini-batch Gradient Descent

Adagrad

Adam and Rms Prop

Etc.

Gradient descent

All Data point At a Time : $J(\Theta) = \frac{1}{2m} \sum (h(X) - Y)^2$

Loop {

Theta = Theta - alpha * D J / D Theta

(j = 0,...,n)

}

Stochastic Gradient Descent

Single Data Point at a Time

1. Randomly shuffle dataset
2. Loop {
 - For $i = 1, \dots, m$ {
 - $\Theta_j = \Theta_j - \alpha * D(h(X) - Y)X_j / D\Theta_j$
 $(j = 0, \dots, n)$
 - }

SGD With Momentum

Concept Of Exponential Weighted Average

T1 T2 T3 T4

b1 b2 b3 b4

Then

$$V_{t1} = b1$$

$$V_{t2} = Y * V_{t1} + b2$$

$$V_{t3} = Y * V_{t2} + b3$$

$$Y = 0 \text{ to } 1$$

Equation For Weight Updation

$$W = W - \text{Alpha} * DL / DW = \text{Old}$$

$$W = W - [Y * V_{t-1} + n * DL / DW_{old}] = \text{New}$$

$$V_{t-1} = 1 * [DI / Dw_old] + Y[DL / Dw_old]_{t-1} + Y^2 [DL / DW_old]_{t-2}$$

MiNi Batch Gradient Descent

Single Data Point at a Time

```
1. Batch = 20 , dataPoint = 100  
2. Loop {  
    For i = 20,..,100 {  
        Theta_j = Theta_j - alpha * 1/20 sum ( h(X) - Y ) X  
        ( j = 0,..., n ) ( sum range = i to i + 19 )  
    }  
}
```

AdaGrade

Equation = $W = W - \text{Alpha} * DL / DW$

Here Very the Alpha (learning Rate)

$\text{Alpha} = \text{Alpha} / \text{Sqrt} (\eta_t + \sigma)$

Where σ = any small positive value

$\eta_t = \sum (DL / DW_t)^2 ; \text{sum range} = t = 1 \text{ to } t$

AdaDelta And RmsProp

$W = W - \text{Alpha} * DL / DW$

$\text{Alpha} = \text{Alpha} / \text{Sqrt} (W_{avg} + \text{Sigma})$

$W_{avg} = \text{Bita} * W_{avg_t-1} + (1 - \text{Bita}) * (DL / DW)^2$

Generally $B = 0.95$

CNN

In deep learning, a convolutional neural network is a class of deep neural networks, most commonly applied to analyzing visual imagery.

WHY To Use CNN Compare To MLP?

Yaa , There Is a Normal Artificial Neural network Like DNN, MLP Are Available to Do The Work So What Is the Need of The CNN ?

Simply Say , CNN Is Being Used Most When We Have a Data in Not Structured Form , specifically In Images

What Happen if we apply images to MLP?

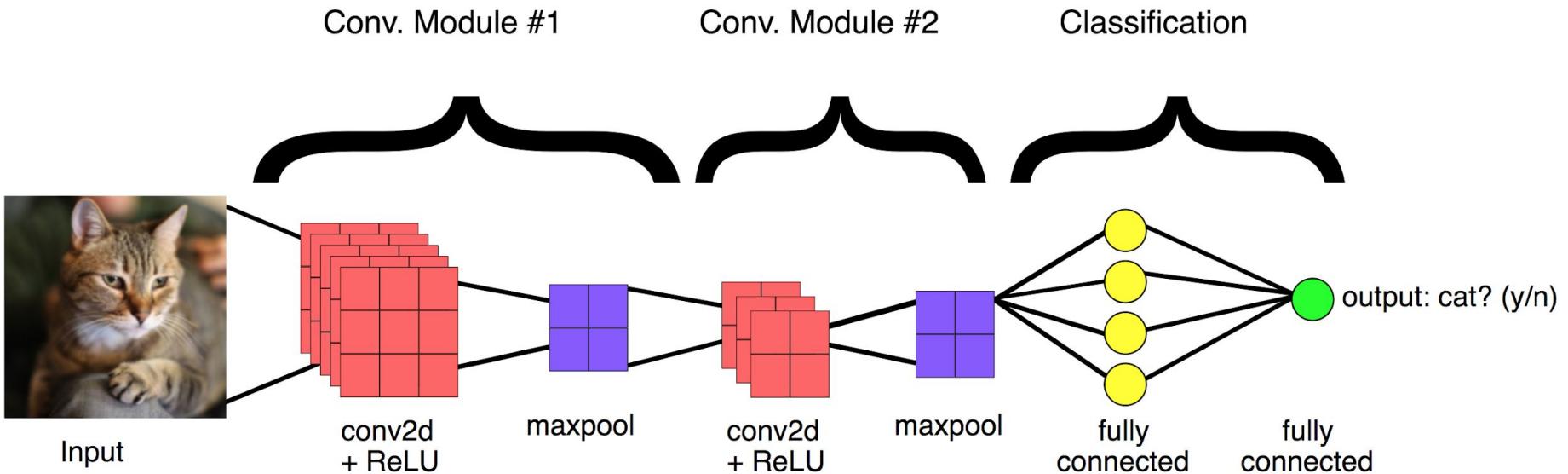
MLPs use one perceptron for each input (e.g. pixel in an image) and the amount of weights rapidly becomes unmanageable for large images. It includes too many parameters because it is fully connected. Each node is connected to every other node in next and the previous layer, forming a very dense web — resulting in redundancy and inefficiency. As a result, difficulties arise whilst training and *overfitting* can occur which makes it lose the ability to generalize.

Another common problem is that MLPs react differently to an input (images) and its shifted version — they are not **translation invariant**. For example, if a picture of a cat appears in the top left of the image in one picture and the bottom right of another picture, the MLP will try to correct itself and assume that a cat will always appear in this section of the image.

Hence, MLPs are not the best idea to use for image processing. One of the main problems is that **spatial information** is lost when the image is flattened(matrix to vector) into an MLP.

We thus need a way to leverage the spatial correlation of the image features (pixels) in such a way that we can see the cat in our picture no matter where it may appear

Architecture of CNN..



Convolution layers

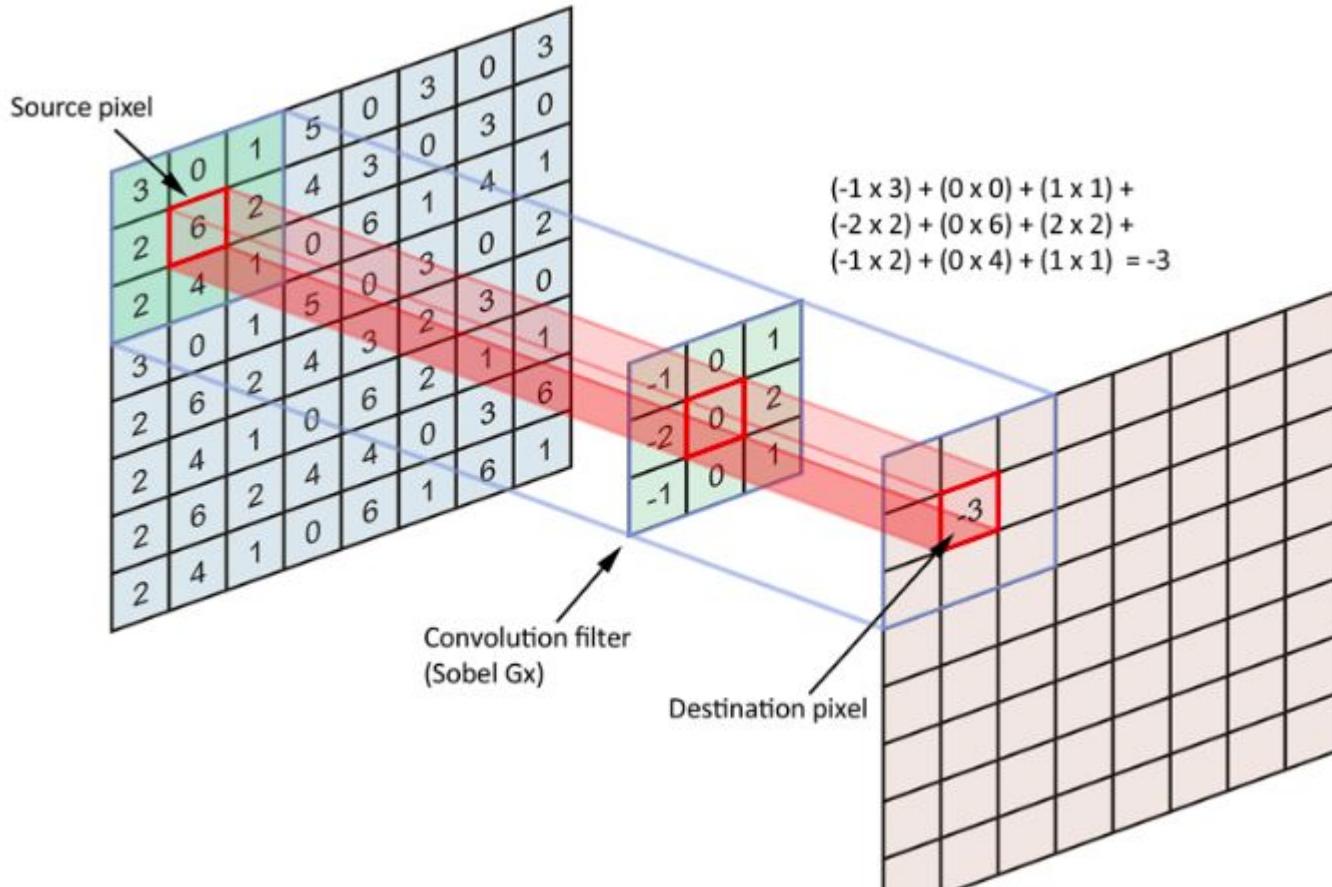
There are Main Three Types of layers

- 1. Convolution Layer**
- 2. Pooling Layer**
- 3. Fully connected Layer**

Convolution Layer

This layer is the first layer that is used to extract the various features from the input images. In this layer, the mathematical operation of convolution is performed between the input image and a filter of a particular size $M \times M$. By sliding the filter over the input image, the dot product is taken between the filter and the parts of the input image with respect to the size of the filter ($M \times M$).

The output is termed as the Feature map which gives us information about the image such as the corners and edges. Later, this feature map is fed to other layers to learn several other features of the input image.



Filters Examples

Edge detection

$$\text{Image} * \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix} = \text{Kernel}$$


Sharpen

$$\text{Image} * \begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix} = \text{Image}$$

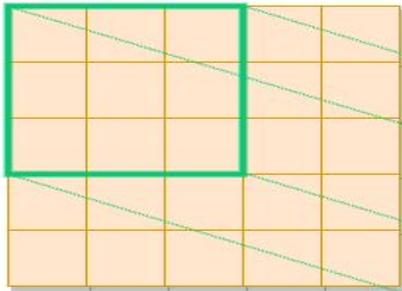

What Is padding?

Padding essentially makes the feature maps produced by the filter kernels the same size as the original image. This is very useful for deep CNN's as we don't want the output to be reduced

Basic Padding Type :

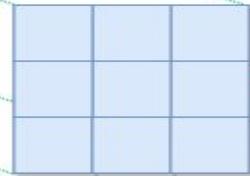
**Zero Padding
Near padding**

Input D x D: 5 x 5

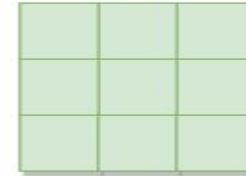


Padding VALID
Output dimension = $D - N + 1$
 $5 - 3 + 1 = 3$

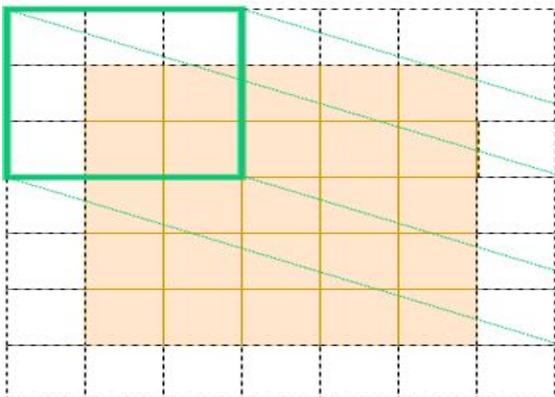
Filter N x N: 3 x 3



Output: 3 x 3



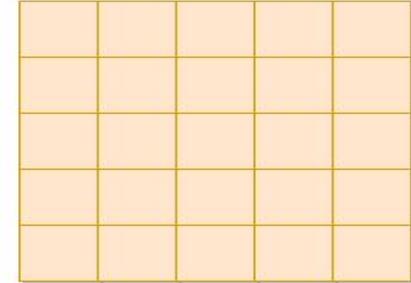
Input D x D: 5 x 5
Plus added padding of size 1



Padding SAME
Output dimension = Input dimension

Filter N x N: 3 x 3

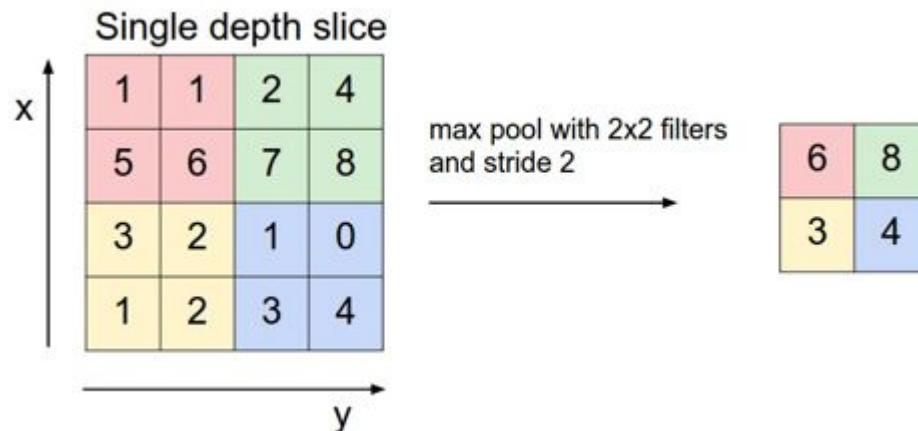
Output: 5 x 5



Pooling Layer

In most cases, a Convolutional Layer is followed by a Pooling Layer. The primary aim of this layer is to decrease the size of the convolved feature map to reduce the computational costs. This is performed by decreasing the connections between layers and independently operates on each feature map. Depending upon method used, there are several types of Pooling operations.

In Max Pooling, the largest element is taken from feature map.



Fully Connected Layer

The Fully Connected (FC) layer consists of the weights and biases along with the neurons and is used to connect the neurons between two different layers. These layers are usually placed before the output layer and form the last few layers of a CNN Architecture.

In this, the input image from the previous layers are flattened and fed to the FC layer. The flattened vector then undergoes few more FC layers where the mathematical functions operations usually take place. In this stage, the classification process begins to take place.

5	4	6
1	2	3
7	9	8

Feature map

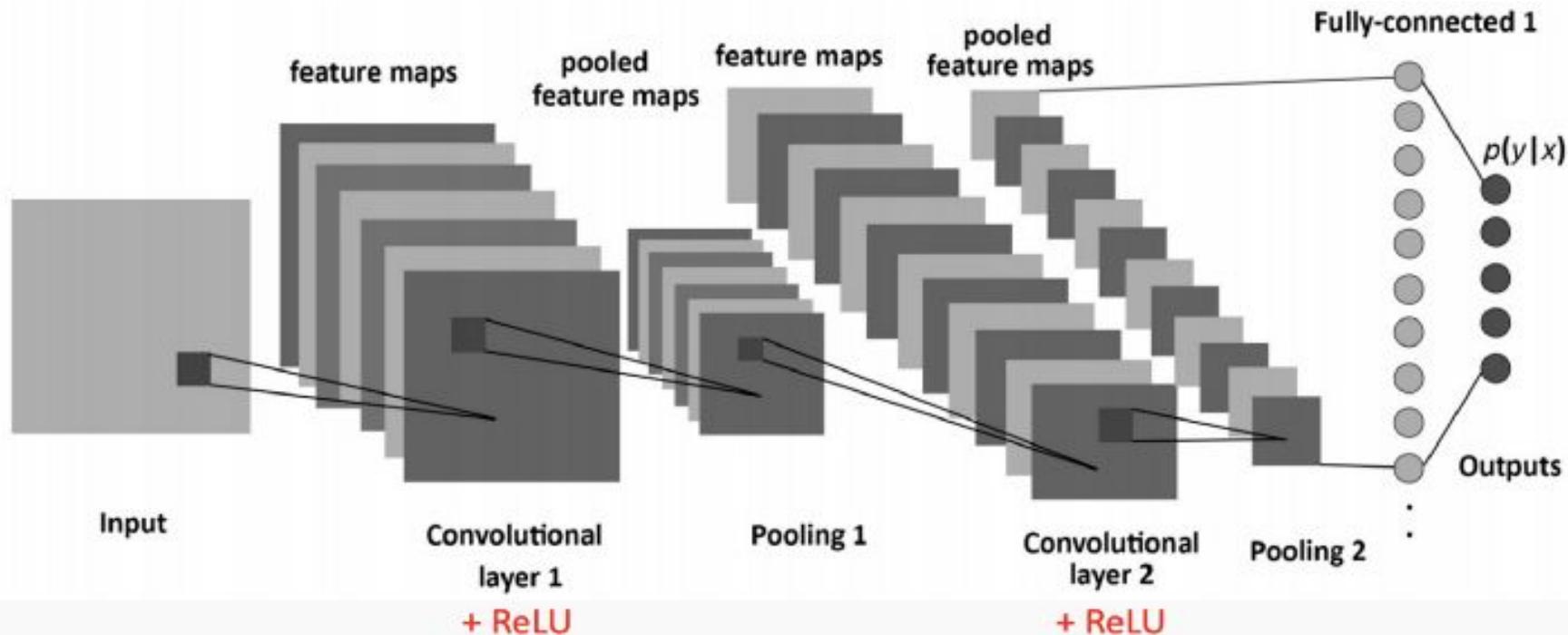
Flatten layer



5
4
6
1
2
3
7
9
8

Flatten output

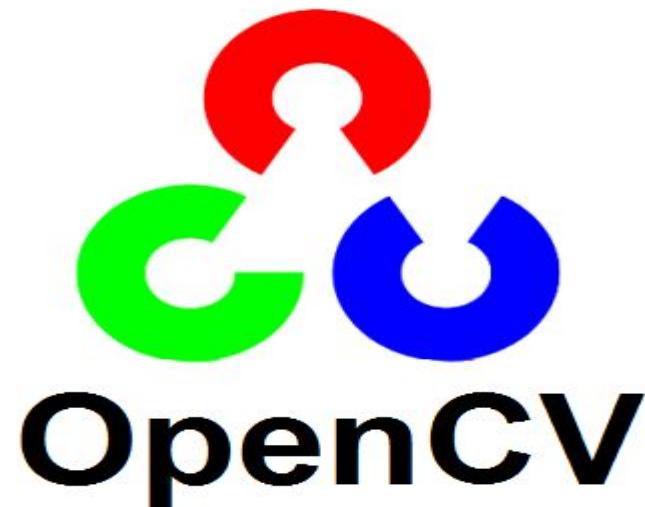
How CNN Works?



Practical

OPENCV

OpenCV is a cross-platform library using which we can develop real-time computer vision applications. It mainly focuses on image processing, video capture and analysis including features like face detection and object detection.



Application of Opencv With python

With the help of Open CV in python, its possible to process images, videos easily and can extract useful information out of that, as there are lots of functions available. Some of the common applications are,

1. Image Processing:
2. Face Detection:
3. Face Recognition:
4. Object Detection:

Installation

Windows :

In the command prompt,

```
pip install OpenCV-python
```

In the anaconda prompt,

```
conda install -c conda-forge OpenCV
```

Linux :

```
sudo apt-get install libopencv-dev python-Open
```

Basic Functions

Read, write and display the image

Images in python represented as NumPy array, basically array of pixels.
code to read and display image using OpenCV

Import CV2

Import Numpy as np

```
Im = cv2.imread("images name.jpg")
```

```
cv2.imshow(im)
```

- **cv2.IMREAD_COLOR**: Default format to read the image.
- **cv2.IMREAD_GRAYSCALE**: Load the image in grayscale format.
- **cv2.IMREAD_UNCHANGED**: Load the image in the same format in which it stored.

cv2.cvtColor (im ,cv2.COLOR_BGR2GRAY)

Save image to local machine

CV2.imwrite(“image_name.jpg”,im)

Flip The Images

```
image =cv2.flip(src, 0)
```

Resize the Image

```
cv2.resize(src, dsize[, dst[, fx[, fy[, interpolation]]]])
```

Src = source/input image

Dsize = desired size for the output image

Fx = scale factor along the horizontal axis

Fy = scale factor along the vertical axis

Finterpolation = flag that takes one of the following methods.

INTER_NEAREST ,INTER_LINEAR ,INTER_AREA

Draw Shapes

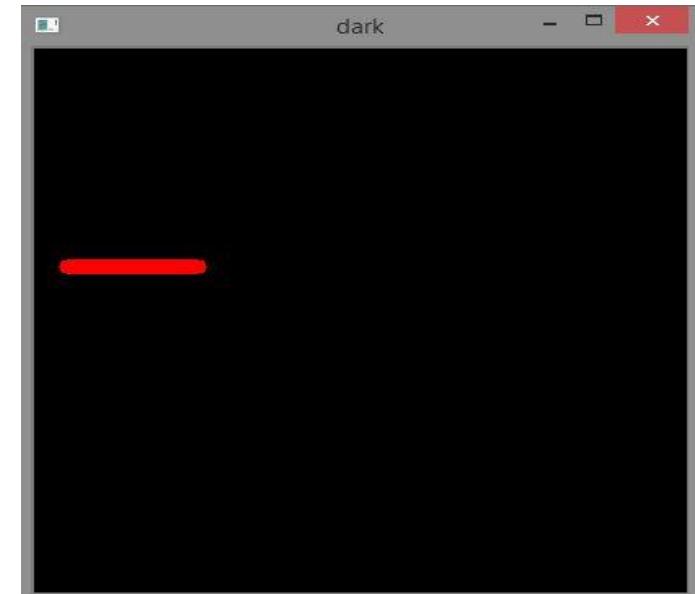
Draw Line :

Syntax:

```
cv2.line(imageObjectName, ('start_coordinates'), ('end_coordinates'), ('color_in_bgr'),  
'line_thickness')
```

Example

```
import numpy as np  
import cv2  
  
# Creating a black image with 3 channels  
# RGB and unsigned int datatype  
img = np.zeros((400, 400, 3), dtype = "uint8")  
  
# Creating line  
cv2.line(img, (20, 160), (100, 160), (0, 0, 255), 10)  
  
cv2.imshow('dark', img)  
  
# Allows us to see image  
# untill closed forcefully  
cv2.waitKey(0)
```



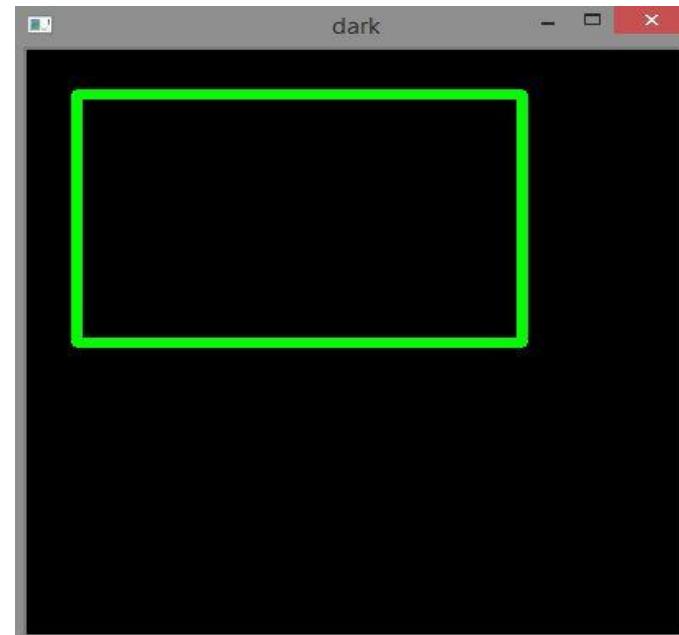
Rectangle Shape

Syntax :

```
cv2.rectangle(imageObjectName, ('top_left_vertex_coordinates'), ('lower_right_vertex_coordinates'),  
('stroke_color_in_bgr'), 'stroke_thickness')
```

Example

```
import numpy as np  
import cv2  
  
# Creating a black image with 3  
# channels RGB and unsigned int datatype  
img = np.zeros((400, 400, 3), dtype = "uint8")  
  
# Creating rectangle  
cv2.rectangle(img, (30, 30), (300, 200), (0, 255, 0), 5)  
  
cv2.imshow('dark', img)  
  
# Allows us to see image  
# until closed forcefully  
cv2.waitKey(0)  
cv2.destroyAllWindows()
```

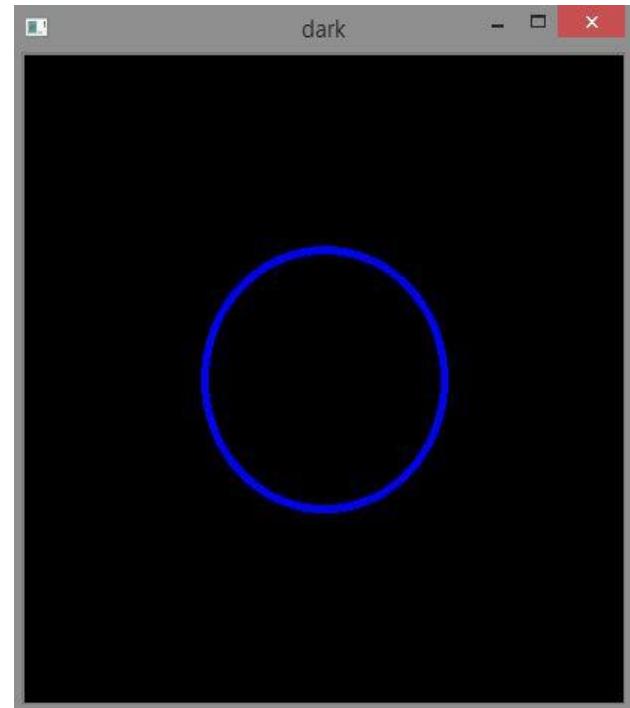


Draw a Circle :

```
cv2.circle(imageObjectName, ('center_coordinates'), ('circle_radius'), ('color_in_bgr'),  
'stroke_thickness')
```

Example

```
cv2.circle(img, (200, 200), 80, (255, 0, 0), 3)
```



Writing Text

```
cv2.putText(imageObjectName, 'TextContent', ('text_starting_point_coordinates'),  
'fontToBeUsed', 'font_size', ('text_color', 'text_thickness', 'line_type')
```

```
# writing text  
font = cv2.FONT_HERSHEY_SIMPLEX  
cv2.putText(img, 'Tops', (50, 50),  
           font, 0.8, (0, 255, 0), 2, cv2.LINE_AA)
```

Face Detection Practical

Video Capturing Practical

Object Detection Practical

NLP

Natural Language Processing or NLP is a field of Artificial Intelligence that gives the machines the ability to read, understand and derive meaning from human languages.

NLP

How much text you see: Email, SMS, Web Pages, endless

Linguistics is the scientific study of language, including its grammar, semantics, and phonetics.

Understanding natural language require large amounts of knowledge about morphology, syntax, semantics and pragmatics as well as general knowledge about the world.

Machine learning methods off the promise of automatic the acquisition of this knowledge from language corpora.

Statistical Natural Language

Processing

With NLP we are concerned with building computational tools that do useful things with language, e.g., machine translation, summarization, question-answering, etc.

The statistical dominance of the field also often leads to NLP being described as Statistical Natural Language Processing, perhaps to distance it from the classical computational linguistics methods.

Bag of Words

	words	rain	a	paper	they	slip	the	universe	...
<i>Words are flowing out like endless rain into a paper cup,</i>	1	1	1	1	0	0	0	0	...
<i>They slither while they pass, they slip away across the universe</i>	0	0	0	0	3	1	1	1	...

Tokenization:

Is the process of segmenting running text into sentences and words. In essence, it's the task of cutting a text into pieces called *t*okens, and at the same time throwing away certain characters, such as punctuation.

Words

are

flowing

out

like

endless

rain

into

a

paper

cup

They

slither

while

they

pass

they

slip

away

across

the

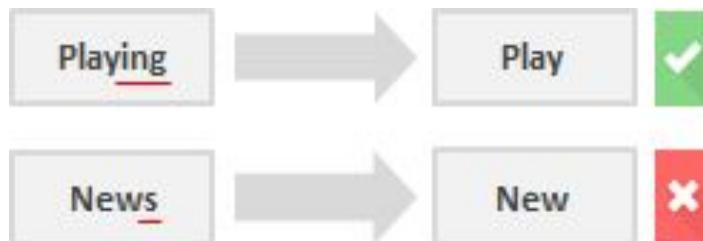
universe

Stop Words

Very common words that appear to provide little or no value to the NLP objective are filtered and excluded from the text to be processed, hence removing widespread and frequent terms that are not informative about the corresponding text.

Stemming

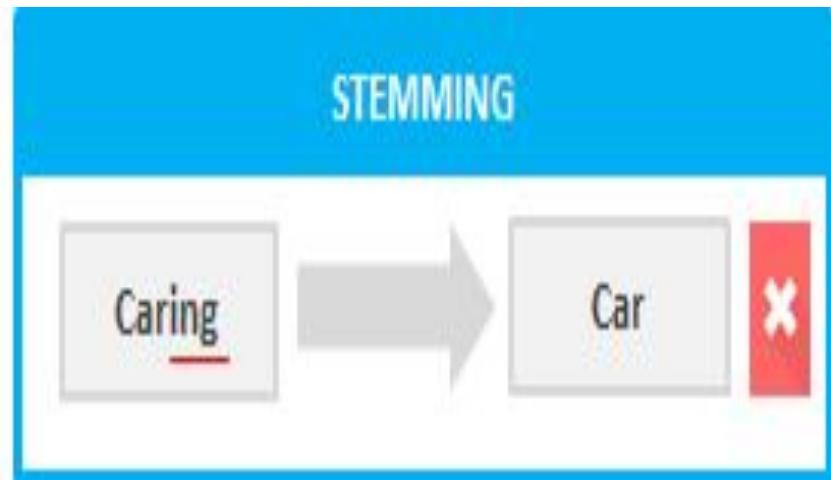
Process of slicing the end or the beginning of words with the intention of removing affixes



Lemmatization

Has the objective of reducing a word to its base form and grouping together different forms of the same word.

hence standardizing words with similar meaning to their root.



POS Tagging

Part of Speech (hereby referred to as POS) Tags are useful for building parse trees, which are used in building NERs (most named entities are Nouns) and extracting relations between words.

Tag	Meaning	English Examples
ADJ	adjective	<i>new, good, high, special, big, local</i>
ADP	adposition	<i>on, of, at, with, by, into, under</i>
ADV	adverb	<i>really, already, still, early, now</i>
CONJ	conjunction	<i>and, or, but, if, while, although</i>
DET	determiner, article	<i>the, a, some, most, every, no, which</i>
NOUN	noun	<i>year, home, costs, time, Africa</i>
NUM	numeral	<i>twenty-four, fourth, 1991, 14:24</i>
PRT	particle	<i>at, on, out, over per, that, up, with</i>
PRON	pronoun	<i>he, their, her, its, my, I, us</i>
VERB	verb	<i>is, say, told, given, playing, would</i>
.	punctuation marks	<i>., ; !</i>
X	other	<i>ersatz, esprit, dunno, gr8, univeristy</i>

Topic Modeling

Is as a method for uncovering hidden structures in sets of texts or documents.

This technique is based on the assumptions that each document consists of a mixture of topics and that each topic consists of a set of words, which means that if we can spot these hidden topics we can unlock the meaning of our texts.

Latent Dirichlet Allocation (LDA) is probably the most commonly used unsupervised learning method that discovers different topics underlying a collection of documents. In

In **unsupervised learning** methods like this one, there is no output variable to guide the learning process and data is explored by algorithms to find patterns.

LDA

1. Assigning each word to a random topic, where the user defines the number of topics it wishes to uncover. You don't define the topics themselves (you define just the number of topics) and the algorithm will map all documents to the topics in a way that words in each document are mostly captured by those imaginary topics.
2. The algorithm goes through each word iteratively and reassigns the word to a topic taking into considerations the probability that the word belongs to a topic, and the probability that the document will be generated by a topic. These probabilities are calculated multiple times, until the convergence of the algorithm.