



# MALLA REDDY COLLEGE OF ENGINEERING & TECHNOLOGY

(AUTONOMOUS INSTITUTION-UGC, GOVT. OF INDIA)

B.Tech  
**Aeronautical  
Engineering**

## Department of AERONAUTICAL ENGINEERING



## ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING MANUAL

Prepared by:

**LSUSHMA**

Associate Professor

Department of ANE

[sushmalukkani@mrcet.ac.in](mailto:sushmalukkani@mrcet.ac.in)

# ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING MANUAL



**B.TECH(R-22 Regulation)  
(III YEAR – II SEM)  
(2023-24)**

**DEPARTMENT OF AERONAUTICAL ENGINEERING**



**MALLAREDDY COLLEGE OF ENGINEERING & TECHNOLOGY**

**(Autonomous Institution – UGC, Govt. of India)**

Recognized under 2(f) and 12(B) of UGC Act 1956

(Affiliated to JNTUH, Hyderabad, Approved by AICTE - Accredited by NBA & NAAC – 'A' Grade - ISO 9001:2015 Certified)  
Maisammaguda, Dhulapally (Post Via. Hakimpet), Secunderabad – 500100, Telangana State, India



## **MALLAREDDY COLLEGE OF ENGINEERING & TECHNOLOGY**

**(Autonomous Institution – UGC, Govt. of India)**

**Affiliated to JNTU H. Approved by AICTE, NBA – Tier 1 & NAAC – ‘A’ Grade ISO 9001:2015 Certified)**

**Maisammaguda, Dhulapally (Post Via. Hakimpet), Secunderabad – 500100, Telangana State, India**

# **ARTIFICIAL INTELLIGENCE & MACHINE LEARNING**

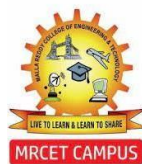
## **B.TECH III YEAR – I SEM**

NAME \_\_\_\_\_

ROLLNO: \_\_\_\_\_ BRANCH \_\_\_\_\_

YEAR: \_\_\_\_\_ SEM: \_\_\_\_\_





**MALLAREDDY COLLEGE OF ENGINEERING & TECHNOLOGY**

**(Autonomous Institution – UGC, Govt. of India)**

**Affiliated to JNTU H. Approved by AICTE, NBA-Tier 1 & NAAC – 'A' Grade ISO 9001:2015 Certified)**

**Maisammaguda, Dhulapally (Post Via. Hakimpet), Secunderabad – 500100, Telangana State, India**

# CERTIFICATE

Certified that this is the Bonafide Record of the work done  
by Mr./Ms. \_\_\_\_\_ bearing  
Roll No. \_\_\_\_\_ of B.Tech III Year  
\_\_\_\_\_ Semester for the Academic year 2022-2023  
in \_\_\_\_\_

Date:

Faculty In-charge

HOD

Internal Examiner

External Examiner

## INDEX

S.No	Date	Title	Page No	Faculty Sign

**INDEX**

S.No	Date	Title	Page No	Faculty Sign

## **Department of AERONAUTICAL ENGINEERING**

### **Vision**

- Department of Aeronautical Engineering aims to be an indispensable source in Aeronautical Engineering which has a zeal to provide the value driven platform for the students to acquire knowledge and empower themselves to shoulder higher responsibility in building a strong nation..

### **Mission**

- The primary mission of the department is to promote engineering education and research. To strive consistently to provide quality education, keeping in pace with time and technology. Department passions to integrate the intellectual, spiritual, ethical, and social development of the students for shaping them into dynamic engineers.

### **QUALITY POLICY**

- Impart up-to date knowledge to the students in Aeronautical area to make them quality engineers. Make the students experience the applications on quality equipment and tools. Provide systems, resources, and training opportunities to achieve continuous improvement. Maintain global standards in education, training, and services.

## PROGRAM OUTCOMES (PO's)

Engineering Graduates will be able to:

- Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
- Problem analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
- Design / development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal and environmental considerations.
- Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
- Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
- The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
- Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.



- Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice. Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
- Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
- Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multi disciplinary environments.
- Life- long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

### **PROGRAM EDUCATIONAL OBJECTIVES–Aeronautical Engineering**

- PEO1 (PROFESSIONALISM & CITIZENSHIP): To create and sustain a community of learning in which students acquire knowledge and learn to apply it professionally with due consideration for ethical, ecological and economic issues.
- PEO2 (TECHNICAL ACCOMPLISHMENTS): To provide knowledge based services to satisfy the needs of society and the industry by providing hands on experience in various technologies in core field.
- PEO3 (INVENTION, INNOVATION AND CREATIVITY): To make the students to design, experiment, analyze, and interpret in the core field with the help of other multidisciplinary concepts wherever applicable.
- PEO4 (PROFESSIONAL DEVELOPMENT): To educate the students to disseminate research findings with good soft skills and become a successful entrepreneur.
- PEO5 (HUMAN RESOURCE DEVELOPMENT): To graduate the students in building national capabilities in technology, education and research

### **PROGRAM SPECIFIC OUTCOMES–Aeronautical Engineering**

- To mould students to become a professional with all necessary skills, personality and sound knowledge in basic and advance technological areas.
- To promote understanding of concepts and develop ability in design manufacture and maintenance of aircraft, aerospace vehicles and associated equipment and develop application capability of the concepts sciences to engineering design and processes.
- Understanding the current scenario in the field of aeronautics and acquire ability to apply knowledge of engineering, science and mathematics to design and conduct experiments in the field of Aeronautical Engineering.
- 4. To develop leadership skills in our students necessary to shape the social, intellectual, business and technical worlds.



MALLAREDDY COLLEGE OF ENGINEERING AND TECHNOLOGY  
III Year B.Tech.ECE-I Sem

L/T/P/C  
0/-/3/1.5

## (R20A0566) ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING LAB

### LAB OBJECTIVES:

1. Familiarity with the Prolog programming environment.
2. To introduce students to the basic concepts and techniques of Machine Learning.
3. To implement classification and clustering methods.
4. To become familiar with Dimensionality reduction Techniques.
5. Learning basic concepts of Prolog through illustrative examples and small exercises & Understanding list data structure in Prolog.

### STUDY OF PROLOG; WRITE THE FOLLOWING PROGRAMS USING PROLOG/PYTHON

- week-1.** Write a program to implement all set operations
- week-2.** Implementation of BFS for water jug problem using Python
- week-3.** Implementation of DFS for tic-tac-toe problem using Python
- week-4.** Solve 8-puzzle problem using best first search
- week-5.** Write a program to solve 8 queens problem
- week-6.** Implementation of Hill-climbing to solve 8-Puzzle Problem

## MACHINE LEARNING

### WEEK-1

#### Data Extraction, Wrangling

1. Loading different types of dataset in Python
2. Arranging the data

### WEEK-2

#### Data Visualization

1. Handling missing values
2. Plotting the graphs

### WEEK-3

#### Supervised Learning

Implementation of Linear Regression

## WEEK-4

Implementation of K-nearest Neighbor

## WEEK-5

### Unsupervised Learning

Implementing K-means Clustering

## WEEK-6

### Unsupervised Learning

Implementing Hierarchical Clustering

## LABORATORY OUTCOMES:

1. Apply various AI search algorithms (uninformed, informed, heuristic, constraint satisfaction,)
2. Understand the fundamentals of knowledge representation, inference using AI tools..
3. Solve the problems using various machine learning techniques
4. Design application using machine learning techniques

**LIST OF EXPERIMENTS**

<b>S.No</b>	<b>NAME OF EXPERIMENT</b>	<b>Page No</b>
	<b>ARTIFICIAL INTELLIGENCE</b>	
1	Program to execute all the Operators	16
2	Program to execute Breadth First Search for Water Jug Problem	25
3	Program to execute Depth First Search for TICTACTOE	33
4	Program to execute Best First Search for 8 Puzzle problem	36
5	Write a program to solve 8 queens problem	45
6	Implementation of Hill-climbing to solve 8-Puzzle Problem	51
	<b>MACHINE LEARNING</b>	
1	Program to Loading different types of datasets in Python and Arranging the data	58
2	Program Handling missing values and plotting graphs	65
3	Program to Implementation of Linear Regression	71
4	Program to Implementation of K-nearest Neighbor	77
5	Program to Implementing K-means Clustering	83
6	Program to Implementing Hierarchical Clustering	89

## INTRODUCTION

Python is developed by **Guido van Rossum**. Guido van Rossum started implementing Python in 1989. Python is a very simple programming language so even if you are new to programming, you can learn Python without facing any issues.

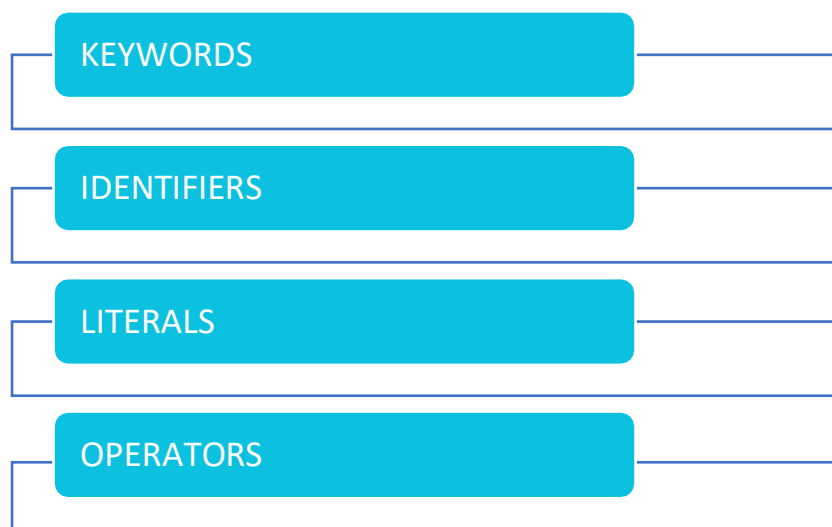
- Programming is the process of creating a set of instructions that tell a computer how to perform a task.

C, C++, MatLab, Python, JavaScript etc.

- Python is a powerful general purpose interactive, object-oriented high-level programming language.
- Easy language to learn as compared to others
- Few hours is very much sufficient and developer friendly
- Focuses on solving problems, skills, concepts and language-specific syntax
- It's used in web development, Data Science, creating software prototypes and much more.
  - i. Backend (or server-side) web and mobile app development
  - ii. Desktop app and software development
  - iii. Writing script files.

### Data Types

- In Python, every logical line of code is broken down into components known as tokens.



**Keywords:** Keywords are that have specific meanings and restrictions around how they should be used. Python keywords are special reserved words that have specific meanings and purposes and can't be used for anything but those specific purposes.

There are thirty-five keywords in Python

- Value Keywords: True, False, None
- Operator Keywords: and, or, not, in, is
- Control Flow Keywords: if, elif, else
- Iteration Keywords: for, while, break, continue, else
- Structure Keywords: def, class, with, as, pass, lambda
- Returning Keywords: return, yield
- Import Keywords: import, from, as
- Exception-Handling Keywords: try, except, raise, finally, else, assert
- Asynchronous Programming Keywords: async, await
- Variable Handling Keywords: del, global, nonlocal

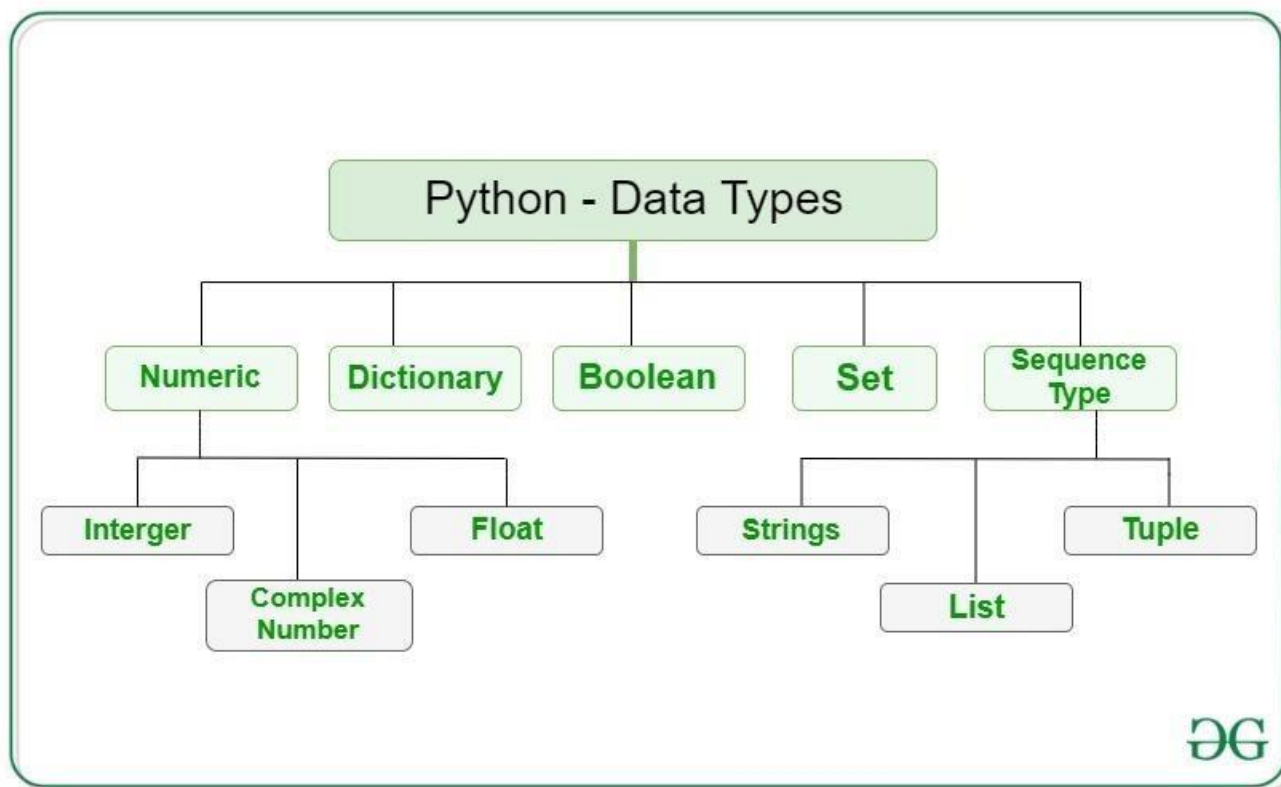
False	await	else	import	pass
None	break	except	in	raise
True	class	finally	is	return
and	continue	for	lambda	try
as	def	from	nonlocal	while
assert	del	global	not	with
async	elif	if	or	yield

**Identifier** is a name used to identify a variable, function, class, module, etc. The identifier is a combination of character digits and underscore. The identifier should start with a character or Underscore then use a digit. The characters are A-Z or a-z, an Underscore(\_), and digit(0-9). we should not use special characters(#, @, \$, %, !) in identifiers.



**Literals:** Literals in Python are defined as the raw data assigned to variables or constants while programming. We mainly have five types of literals which includes string literals, numeric literals, boolean literals, literal collections and a special literal

**Raw data given to variables are of 4 types**



Int	Long	Float	Complex
+ve or -ve numbers(whole numbers)	An unlimited string of integers followed by upper or lower case L	Real number with both integer and fraction. continuous numbers	Number with Real and imaginary
Without fractions or decimals	Ex: 233424243L	Ex: 213.4, -345.8	Ex: 3+4j
Ex: 100, 89, -90, -30			

String: String is set of characters and letters

Boolean: Logical

Python divides the operators in the following groups:

- Arithmetic operators
- Assignment operators
- Comparison operators
- Logical operators
- Identity operators



## EXPERIMENT 1

AIM: Write a program for executing all operators in Python

Software Used: Python

### Arithmetic Operators

Arithmetic operators are used to performing mathematical operations like addition, subtraction, multiplication, and division.

Operator	Description	Syntax
+	Addition: add two operands	$x + y$
-	Subtraction: subtract two operands	$x - y$
*	Multiplication: multiply two operands	$x * y$
/	Division(float): divide the first operand by the second	$x / y$
//	Division(floor): divide the first operand by the second	$x // y$
%	Modulus: return the remainder when the first operand is divided by the second	$x \% y$
**	Power: Return first raised to power second	$x ** y$

### Example: Arithmetic operators in Python

- Python3

```
#Examples of Arithmetic Operator
a=9
b=4

#Addition of numbers add
=a +b

#Subtraction of numbers sub
=a -b

#Multiplication of number
mul =a *b

#Division (float) of number
div1 =a /b
```

```
#Division (floor) of number
div2 =a //b

#Modulo of both number mod
=a %b

#Power
p=a**b

#print results
print(add)
print(sub)
print(mul)
print(div1)
print(div2)
print(mod)
print(p)
```

## Output

```
13
5
36
2.25
2
1
6561
```

## Comparison Operators

Comparison of Relational operators compares the values. It either returns **True** or **False** according to the condition.

Operator	Description	Syntax
>	Greater than: True if the left operand is greater than the right	x>y
<	Less than: True if the left operand is less than the right	x<y
==	Equal to: True if both operands are equal	x==y
!=	Not equal to: True if operands are not equal	x!=y
>=	Greater than or equal to: True if the left operand is greater than or equal to the right	x>=y
<=	Less than or equal to: True if the left operand is less than or equal to the right	x<=y

## Example: Comparison Operators in Python

- Python3

```
#Examples of Relational Operators
a=13
b=33

#a>b is False
print(a > b)

#a<b is True
print(a < b)

#a==b is False print(a
==b)

#a!=b is True
print(a !=b)

#a>=b is False print(a
>=b)

#a<=b is True
print(a <=b)
```

### Output

False

True

False

True

False

True

## Logical Operators

Logical operators perform **Logical AND**, **Logical OR**, and **Logical NOT** operations. It is used to combine conditional statements.

Operator	Description	Syntax
and	Logical AND: True if both the operands are true	x and y
or	Logical OR: True if either of the operands is true	x or y
not	Logical NOT: True if the operand is false	not x

## Example: Logical Operators in Python

- Python3

```
#Examples of Logical Operator
a=True
b=False
```

```
#Print a and b is False
print(a and b)
```

```
#Print a or b is True
print(a or b)
```

```
#Print not a is False
print(not a)
```

## Output

False

True

False

## Bitwise Operators

Bitwise operators act on bits and perform the bit-by-bit operations. These are used to operate on binary numbers.

Operator	Description	Syntax
&	Bitwise AND	x&y
	Bitwise OR	x y
~	Bitwise NOT	~x
^	Bitwise XOR	x^y
>>	Bitwise right shift	x>>
<<	Bitwise left shift	x<<

## Example: Bitwise Operators in Python

- Python3

```
#Examples of Bitwise operators
a = 10
```

```

b=4

#PrintbitwiseANDoperation print(a
& b)

#PrintbitwiseORoperation print(a |
b)

#PrintbitwiseNOToperation
print(~a)

#printbitwiseXORoperation print(a
^ b)

#printbitwiserightshiftoperation print(a
>> 2)

#printbitwiseleftshiftoperation print(a <<
2)

```

## Output

```

0
14
-11
14
2
40

```

## Assignment Operators

Assignment operators are used to assigning values to the variables.

Operator	Description	Syntax
=	Assign value of right side of expression to left side operand	x=y+z
+=	Add AND: Add right-side operand with left side operand and then assign to left operand	a+=b    a=a+b
-=	Subtract AND: Subtract right operand from left operand and then assign to left operand	a-=b    a=a-b
*=	Multiply AND: Multiply right operand with left operand and then assign to left operand	a*=b    a=a*b
/=	Divide AND: Divide left operand with right operand and then assign to left operand	a/=b    a=a/b
%=	Modulus AND: Takes modulus using left and right operands and assign the result to left operand	a%=b    a=a%b

Operator	Description	Syntax
//=	Divide(floor)AND:Divide left operand with right operand and then assign the value(floor) to left operand	a//=b    a=a//b
**=	ExponentAND:Calculate exponent(raise power) value using operands and assign value to left operand	a**=b    a=a**b
&=	Performs Bitwise AND on operands and assign value to left operand	a&=b    a=a&b
=	Performs Bitwise OR on operands and assign value to left operand	a =b    a=a b
^=	Performs Bitwise XOR on operands and assign value to left operand	a^=b    a=a^b
>>=	Performs Bitwise right shift on operands and assign value to left operand	a>>=b    a=a>>b
<<=	Performs Bitwise left shift on operands and assign value to left operand	a<<=b    a=a<<b

## Example: Assignment Operators in Python

- Python3

```
#Examples of Assignment Operators
a = 10

#Assign value b
b = a
print(b)

#Add and assign value b
b += a
print(b)

#Subtract and assign value b
b -= a
print(b)

#multiply and assign b
b *= a
print(b)
```



```
#bitwise left shift operator b
<<=a
print(b)
```

## Output

10

20

10

100

102400

## Identity Operators

**is** and **is not** are the identity operators both are used to check if two values are located on the same part of the memory. Two variables that are equal do not imply that they are identical.

```
is           True if the operands are identical
is not       True if the operands are not identical
```

### Example: Identity Operator

- Python3

```
a=10
b=20
c=a

print(a is not b)
print(a is c)
```

## O-utput

True

True

## Membership Operators

**in** and **not in** are the membership operators; used to test whether a value or variable is in a sequence.

```
in           True if value is found in the sequence
not in       True if value is not found in the sequence
```

### Example: Membership Operator

- Python3

```
#Python program to illustrate #
not 'in' operator
x=24
y=20
list=[10,20,30,40,50]

if(xnotinlist):
    print("x is NOT present in given list") else:
    print("x is present in given list")

if(yinlist):
    print("y is present in given list") else:
    print("y is NOT present in given list")
```

**Output**

x is NOT present in given list y  
is present in given list

**Precedence and Associativity of Operators**

**Precedence and Associativity of Operators:** Operator precedence and associativity determine the priorities of the operator.

**Operator Precedence**

This is used in an expression with more than one operator with different precedence to determine which operation to perform first.

**Example: Operator Precedence**

- Python3

```
#Examples of Operator Precedence #

Precedence of '+' & '*'
expr=10+20*30
print(expr)

#Precedence of 'or' & 'and' name
="Alex"
age=0

if name=="Alex" or name=="John" and age>=2:
    print("Hello! Welcome.")
else:
    print("Good Bye!!")
```

**Output**

610

Hello! Welcome.

## Operator Associativity

If an expression contains two or more operators with the same precedence then Operator Associativity is used to determine. It can either be Left to Right or from Right to Left.

### Example: Operator Associativity

- Python3

```
#Examples of Operator Associativity #
```

```
Left-right associativity
#100/10*10 is calculated as
#(100/10)*10 and not
#as 100/(10*10)
print(100/10*10)
```

```
#Left-right associativity #5-
2+3 is calculated as
#(5-2)+3 and not
#as 5-(2+3)
print(5-2+3)
```

```
#left-right associativity
print(5-(2+3))
```

```
#right-left associativity
#2**3**2 is calculated as
#2**(3**2) and not
#as (2**3)**2
print(2**3**2)
```

### Output

100.0

6

0

512

## EXPERIMENT NO 2

**Aim:** Execute code to perform Breadth First Search for Water Jug Problem

**Problem:**

Let our problem given is we have two empty jugs A and B to which we need to fill the water and is not marked. If Jug A holding capacity is of 3 Liters and Jug B capacity is 04 Liters. The condition or requirement mentioned is to fill 2 Liters of water in either of jug. How would you achieve this requirement?



Figure 1.1

*Hint: There can be several approaches. You need to pick the best solution.*

The steps you can perform to achieve this requirement are as follows.

- Empty a Jug.
- Fill a Jug.

- Pour water from one jug to the other until one of the jugs is holding required capacity of 2 liters.

## **Approach—Using Breadth-First Search (BFS)**

Breadth-first search is a graph traversal algorithm. It starts traversing the graph from the root node and explores all the neighboring nodes. Then, it selects the nearest node and explores all the unexplored nodes. The algorithm continues the same process for each of the nearest nodes until it finds the goal.

Here, we'll be defining the states in terms of  $(A, B)$ .

$A$ —Amount of water in Jug A

$B$ —Amount of water in Jug B

INITIAL STATE: As initially both jugs are empty—initial state would be  $(0, 0)$

FINAL STATE: As either of jug must have 2 litres

final states of the process could be either  $(0, 2)$  or  $(2, 0)$

The steps that we'll be performing in this approach are as follows.

1. Empty a Jug: The state transition could be  $(A, B) \rightarrow (0, B)$ . Let's assume that we empty the Jug A.
2. Fill a Jug: The state transition could be  $(0, 0) \rightarrow (A, 0)$ . Let's assume that we Fill Jug A.

3. Pour water from one jug to the other until one of the jugs is either empty or full:  
The state transition could be  $(A, B) \rightarrow (A-d, B+d)$ .

Steps:

```

00  EmptyBoth
30  Fill3litersJug
34  Fill4litersJug
04  Empty3litersJug
3 1  Fill3litersjugfrom4litersjug,So leftoverwillbe1literin4LiterJug 0 1 Empty
3 liters jug
10  Fillthe1literleftoverin3liters
1 4  nowfill4literjugagain,andpourthe2litersinto3liters(asitisfilled with 1
     liter already
32  sonowJugAhas3litersJugBhas2liter
02  Empty3liters,nowtheResultobtainedis1Jughas2liters
    
```

## Algorithm

- Initialise a queue to implement **BFS**.
- Since, initially, both the jugs are empty, insert the state  $\{0,0\}$  into the queue.
- Perform the following state, till the queue becomes empty:
  - Pop out the first element of the queue.
  - If the value of popped element is equal to **Z**, return True.
  - Let **X\_left** and **Y\_left** be the amount of water left in the jugs respectively.
  - Now perform the **fill** operation:
    - If the value of **X\_left**  $<$  **X**, insert  $\{\mathbf{X\_left}, \mathbf{Y}\}$  into the hashmap, since this state hasn't been visited and some water can still be poured in the jug.
    - If the value of **Y\_left**  $<$  **Y**, insert  $\{\mathbf{Y\_left}, \mathbf{X}\}$  into the hashmap, since this state hasn't been visited and some water can still be poured in the jug.
  - Perform the **empty** operation:

- If the state  $(0, Y\_left)$  isn't visited, insert it into the hashmap, since we can empty any of the jugs.
- Similarly, if the state  $(X\_left, 0)$  isn't visited, insert it into the hashmap, since we can empty any of the jugs.
- Perform the **transfer of water** operation:
  - $\min(X - X\_left, Y)$  can be poured from second jug to first jug. Therefore, in case  $-(X + \min(X - X\_left, Y), Y - \min(X - X\_left, Y))$  isn't visited, put it into hashmap.
  - $\min(X\_left, Y - Y\_left)$  can be poured from first jug to second jug. Therefore, in case  $-(X\_left - \min(X\_left, Y - Y\_left), Y + \min(X\_left, Y - Y\_left))$  isn't visited, put it into hashmap.
- Return False, since, it is not possible to measure Z litres.

Code: Python











## EXPERIMENT 3

Aim: Depth First Search for TICTACTOE

Software Used: Python

Problem:

Depth-first search is an algorithm that traverses a tree depth-first, meaning that it traverses the tree recursively, exhausting one branch completely before continuing to the next one.

Tic-tac-toe is a very popular game, so let's implement an automatic Tic-tac-toe game using Python. The game is automatically played by the program and hence, no user input is needed. Still, developing an automatic game will be lots of fun. Let's see how to do this. NumPy and random Python libraries are used to build this game. Instead of asking the user to put a mark on the board, the code randomly chooses a place on the board and puts the mark. It will display the board after each turn unless a player wins. If the game gets drawn, then it returns -1.

**Explanation:** `play_game()` is the main function, which performs the following tasks:

- Calls `create_board()` to create a 3x3 board and initializes with 0.
- For each player (1 or 2), calls the `random_place()` function to randomly choose a location on board and mark that location with the player number, alternatively.
- Print the board after each move.
- Evaluate the board after each move to check whether a row or column or diagonal has the same player number. If so, displays the winner's name. If after 9 moves, there is no winner then displays -1.

Code:











**EXPERIMENT NO 4**

AIM: Execute Best First Search for 8 Puzzle problem

Software Used: Python

Best First Search: The idea of **Best First Search** is to use an evaluation function to decide which adjacent is most promising and then explore.

Best First Search falls under the category of Heuristic Search or Informed Search.

**Method of Best First Search algorithm**

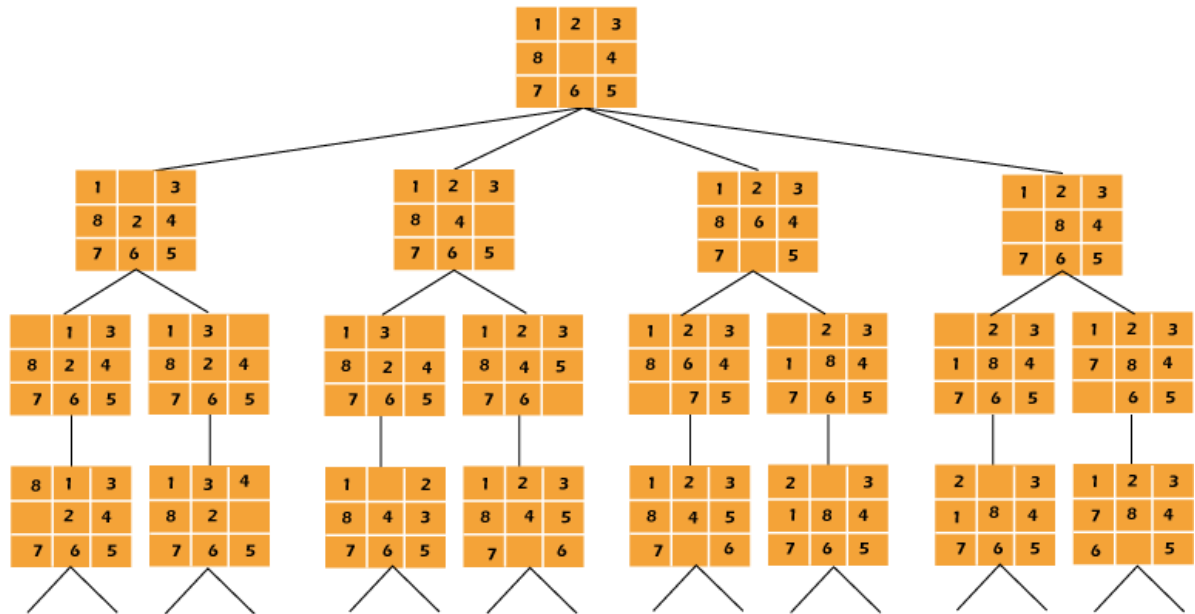
- Create two empty lists
- Start from the initial node and add it to the ordered open list
- Next the below steps are repeated until the final node or endpoint is reached
- If the open list is empty, exit the loop and return a False statement which says that the final node cannot be reached
- Select the top node in the open list and move it to the closed list while keeping track of the parent node
- If the node removed is the endpoint node, return a True statement meaning a path has been found and moving the node to the closed list
- However, if it is not the endpoint node, then list down all the neighboring nodes of it and add them to the open list
- According to the evaluation function, reorder the nodes.

This algorithm will traverse the shortest path first in the queue. The time complexity of the algorithm is  $O(n \cdot \log(n))$ .

Problem:

The 8 puzzle problem solution is to be solved. A **3 by 3 board with 8 tiles** (each tile has a number from 1 to 8) and a **single empty space** is provided. The goal is to **use the vacant space to arrange the numbers on the tiles** such that they match the final arrangement. **Four neighbouring** (left, right, above, and below) **tiles** can be moved into the available area.

For instance,













## EXPERIMENT 5

AIM: Write a program to solve 8 queens problem

Software Used: Python

### Problem:

In the N-Queens problem, we have N queens on an  $N \times N$  chessboard. Each queen occupies their own column and only their own column, but can move to any spot in that specific column.

On each iteration, **one** queen can move anywhere in their own column. The solution (or at least *goal* solution) is to reach a state (arrangement of queens on the board) where no queen intersects with another queen diagonally or horizontally (we obviously don't have to worry about vertical intersections :-)). There may not be a perfect solution, dependent upon the structure (what your **N** is) and the initial state (which row each queen occupies) of the problem.

To try and get to a *goal* state (i.e., no intersecting queens) given an initial state, we are going to look at using hill-climbing and simulated annealing algorithms (local search algorithms). We'll start off with an introduction to the code, then explain our hill-climbing and simulated annealing code.

A description of the notions of all terminologies used in the problem will be given and are as follows:-

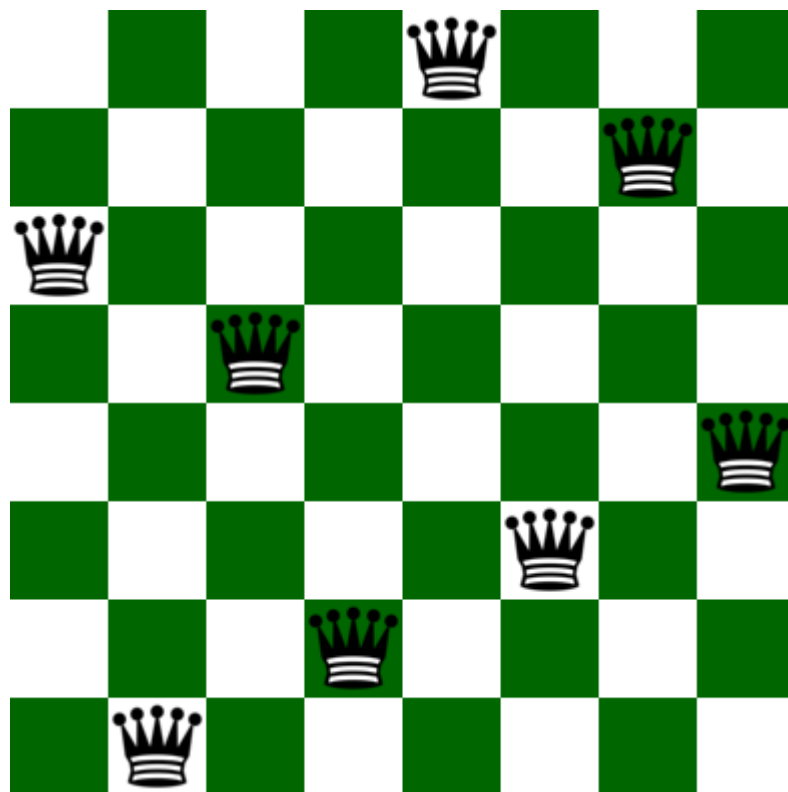
- **Notion of a State** – A state here in this context is any configuration of the N queens on the  $N \times N$  board. Also, in order to reduce the search space let's add an additional constraint that there can only be a single queen in a particular column. A state in the program is implemented using an array of length N, such that if **state[i]=j** then there is a queen at column index **i** and row index **j**.
- **Notion of Neighbours** – Neighbours of a state are other states with board configuration that differ from the current state's board configuration with respect to the position of only a single queen. This queen that differs a state from its neighbour may be displaced anywhere in the same column.
- **Optimisation function or Objective function** – We know that local search is an optimization algorithm that searches the local space to optimize a function that takes the state as input and gives some value as an output. The value of the objective function of a state here in this context is the number of pairs of queens attacking each other. Our goal here is to find a state with the minimum objective value. This function has a maximum value of  $N^2$  and a minimum value of 0.

### Algorithm:

1. Start with a random state (i.e., a random configuration of the board).



2. Scan through all possible neighbours of the current state and jump to the neighbour with the highest objective value, if found any. If there does not exist, a neighbour, with objective strictly higher than the current state but there exists one with equal then jump to any random neighbour(escaping shoulder and/or local optimum).
3. Repeat step 2, until a state whose objective is strictly higher than all it's neighbour's objectives, is found and then go to step 4.
4. The state thus found after the local search is either the local optimum or the global optimum. There is no way of escaping local optima but adding a random neighbour or a random restart each time a local optimum is encountered increases the chances of achieving global optimum(the solution to our problem).
5. Output the state and return.











## EXPERIMENT 6

AIM: Implementation of Hill-climbing to solve 8-Puzzle Problem Software

Used: Python

### Simple Hill Climb Algorithm

Simple hill climbing is the simplest way to implement a hill climbing algorithm. **It only evaluates the neighbor node state at a time and selects the first one which optimizes current cost and set it as a current state.** It only checks its one successor state, and if it finds better than the current state, then move else be in the same state.

#### features

- Less time consuming
- Less optimal solution and the
- solution is not guaranteed

### Steps involved in simple hill climbing algorithm

**Step 1:** Evaluate the initial state, if it is goal state then return success and Stop. **Step 2:**

Loop Until a solution is found or there is no new operator left to apply. **Step 3:**

Select and apply an operator to the current state.

**Step 4:** Check new state:

If it is goal state, then return success and quit.

Else if it is better than the current state then assign new state as a current state. Else if not better than the current state, then return to step 2.

**Step 5:** Exit.

Problem:













# MACHINE LEARNING

## EXPERIMENT 1

**AIM:** Loading different types of datasets in Python and Arranging the data

**Software Used:** Python

**Data science:** Data science combines math and statistics, specialized programming, advanced analytics, artificial intelligence (AI), and machine learning with specific subject matter expertise to uncover actionable insights hidden in an organization's data. These insights can be used to guide decision making and strategic planning.

There are **different ways of loading data using python**. The motivation behind writing this blog is when I searched about the same and got minimal sites that are shifting their focus from **CSV format** to any other available method. However, there is no doubt about the fact that **PD.read\_csv()** is one of the best ways of reading datasets using python (**pandas**-particularly). While working on a real-time project, we should know different methods of **loading and accessing the dataset** as no one knows what turns out to be handy at what time.

**There are different ways of loading data using python.**

Loading Data in Python Using 5 different methods

1. **Manually loading a file:** This is the first, most **popular**, and **least recommended way to load data as it requires many code parts** to read one tuple from the DataFrame. This way comes into the picture when the dataset **doesn't have any particular pattern to identify** or a **specific pattern**.
2. **I am using np.loadtxt:** One of the **NumPy methods** for loading different types of data though it is only supported when we have data in a specific format, i.e., **pattern recognizable**, unlike the manual way of reading the dataset.
3. **Using np.GenFromTxt:** This is another NumPy way to read the data, but this time it is much better than the **np.loadtxt()** method recognizes the column header's presence on its own, which the previous one cannot follow. Along with that, it can also detect the **right data type** for each column.
4. **Using PD.read\_csv:** Here is the most recommended and widely used method for **reading, writing, and manipulating** the dataset. It only deals with **CSV formatted data**, but the support of various parameters makes it a **gold mine for data analysts** to work with different sorts of data (they should have a specific format).
5. **Using pickle:** Last but not least, we will also use **a pickle** to read the dataset present in the **binary format**.

**Numpy:** Numpy is the core library for scientific computing in Python. It provides a high-performance multidimensional array object, and tools for working with these arrays.

- **Arrays:** A numpy array is a grid of values, all of the same type, and is indexed by a tuple of nonnegative integers. The number of dimensions is the *rank* of the array; the *shape* of an array is a tuple of integers giving the size of the array along each dimension.
- **SciPy:** Numpy provides a high-performance multidimensional array and basic tools to compute with and manipulate these arrays. SciPy builds on this, and provides a large number of functions that operate on numpy arrays and are useful for different types of scientific and engineering applications.
- **Matplotlib:** Matplotlib is a plotting library. In this section, we give a brief introduction to the `matplotlib.pyplot` module, which provides a plotting system similar to that of MATLAB.
- **Images:** You can use the `imshow` function to show images.

Learning pandas **sort methods** is a great way to start with or practice doing basic [data analysis using Python](#). Most commonly, data analysis is done with [spreadsheets](#), [SQL](#), or [pandas](#). One of the great things about using pandas is that it can handle a large amount of data and offers highly performant data manipulation capabilities.

- **Sorting Your DataFrame on Single Columns**
  - [Sorting by a Column in Ascending Order](#)
  - [Changing the Sort Order](#)
  - [Choosing a Sorting Algorithm](#)
- **Sorting Your DataFrame on Multiple Columns**
  - [Sorting by Multiple Columns in Ascending Order](#)
  - [Changing the Column Sort Order](#)
  - [Sorting by Multiple Columns in Descending Order](#)
  - [Sorting by Multiple Columns with Different Sort Orders](#)
- **Sorting Your DataFrame on Its Index**
  - [Sorting by Index in Ascending Order](#)
  - [Sorting by Index in Descending Order](#)
  - [Exploring Advanced Index-Sorting Concepts](#)
- **Sorting the Columns of Your DataFrame**
  - [Working With the DataFrame axis](#)
  - [Using Column Labels to Sort](#)
- **Working With Missing Data When Sorting in Pandas**
  - [Understanding the `na\_position` Parameter in `sort\_values\(\)`](#)
  - [Understanding the `na\_position` Parameter in `sort\_index\(\)`](#)
- **Using Sort Methods to Modify Your DataFrame**
  - [Using `sort\_values\(\)` In Place](#)
  - [Using `sort\_index\(\)` In Place](#)













## EXPERIMENT 2

**AIM:** Handling missing values and plotting graphs

**Software Used:** Python

Missing data is always a problem in real life scenarios. Areas like machine learning and data mining face severe issues in the accuracy of their model predictions because of poor quality of data caused by missing values. In these areas, missing value treatment is a major point of focus to make their models more accurate and valid.

**When and Why is Data Missed?**

Let us consider an online survey for a product. Many a times, people do not share all the information related to them. Few people share their experience, but not how long they are using the product; few people share how long they are using the product, their experience but not their contact information. Thus, in some or the other way a part of data is always missing, and this is very common in real time.

**Calculation with Missing:** Data None, is a Python singular object that is frequently used for missing data in Python programs. Because it is a Python object, None can only be used in arrays of the data type "object" (i.e., arrays of Python objects), and cannot be used in any other NumPy/Pandas array:

**Cleaning Missing Data:** The result of the `isna()` and `isnull()` methods is a Boolean check of whether or not each cell of the DataFrame has a missing value. In this way, if a value is absent from a certain cell, the function will return True; otherwise, it will return False (if the cell has a value).

**Dropping Missing Data:** You can choose to either ignore missing data or substitute values for it when handling missing data. As we can see at the bottom of the DataFrame output, this results in a clean DataFrame with no missing data.

**Replacing Missing Data:** You can choose to either ignore missing data or substitute values for it when handling missing data. Fortunately, the Pandas `fillna()` method may be used to replace missing values in a DataFrame with a value given by the user. Type the following to replace any missing values with the number 0 (i.e., the value of 0 is arbitrary and may be any other value of your choice):

### Important Functions for Handling Missing Data in Pandas

**isnull():** The `isnull()` method returns a DataFrame of boolean values that are True for NaN values when checking null values in a Pandas DataFrame.

**notnull():** The `notnull()` method returns a DataFrame of boolean values that are False for NaN values when checking for null values in a Pandas DataFrame.

**dropna():** We have used the `dropna()` method to remove null values from a DataFrame. This function removes rows and columns of datasets containing null values in several ways. We'll go through an illustration of dropping rows that have at least one null value.

**fillna():** By using the `fillna()`, `replace()`, and `interpolate()` functions, we may fill in any null values in a dataset by replacing NaN values with alternative values. The datasets of a DataFrame can be filled

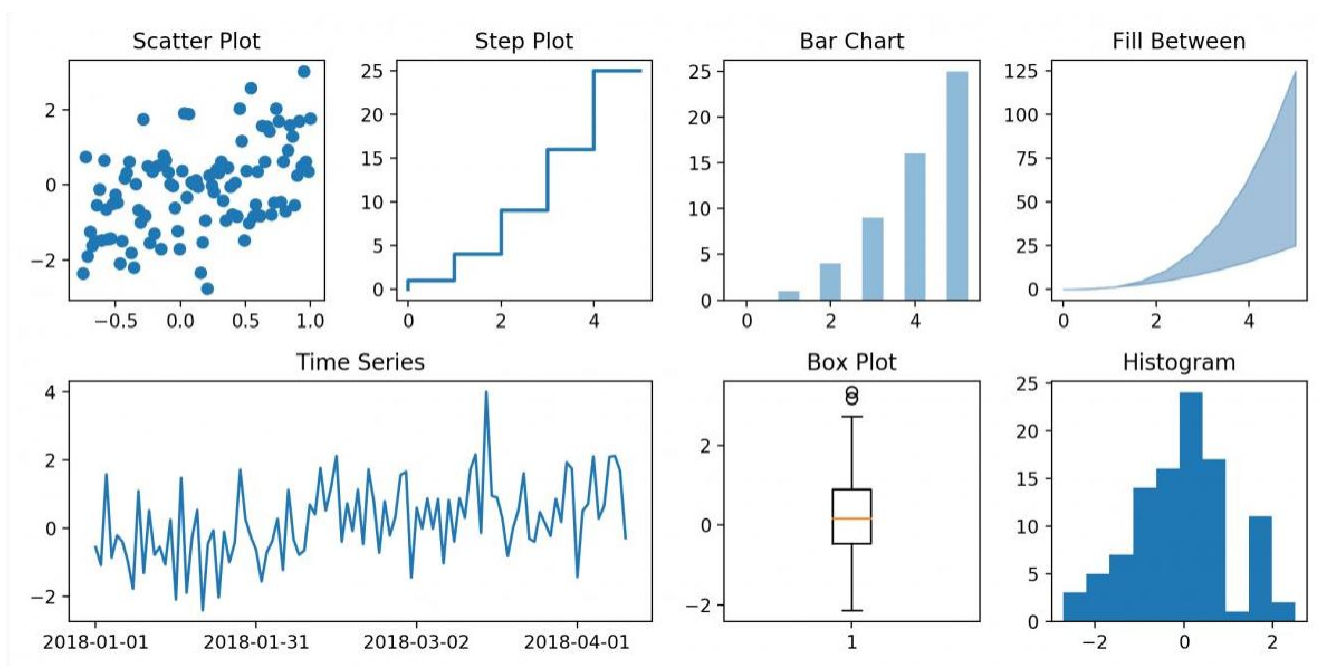
with null values thanks to all these functions. The Interpolate() method is mostly used to fill NA values in a dataframe, although it does so using various interpolation techniques rather than hard-coding the value. We'll begin by looking at an illustration of how to fill in a null value with a single value.

**replace():** Using the replace method we can not only replace or fill null values but any values specified as a function attribute. We specify the value to be replaced into `_replace` and then the new value in `value`.

**interpolate():** Pandas' ability to substitute missing values with those that make sense is another characteristic. The interpolate() function is employed. Pandas fills in the gaps beautifully by using the points' midpoints. Naturally, if this was curvilinear, a function would be fitted to it in order to discover a different technique to determine the average. We'll discuss the linear approach to interpolate the missing numbers. Keep in mind that the linear technique treats the data as equally spaced and disregards the index.

Visualization is the perfect form of representing data so that developers can understand what the data wants to express. It also gives a clear insight into the data demonstrating approaches. But not all data requires the same format of representation. That is where Matplotlib comes with different ways of generating visuals against data. Plotting methods allow for a handful of plot styles other than the default line plot. These methods can be provided as the kind keyword argument to **plot()**, and include:

- 'bar' or 'barh' for bar plots
- 'hist' for histogram
- 'box' for boxplot
- 'kde' or 'density' for density plots
- 'area' for area plots
- 'scatter' for scatter plots
- 'hexbin' for hexagonal bin plots
- 'pie' for pie plots













## EXPERIMENT 3

**AIM:** Implementation of Linear Regression

**Software used:** Python

Supervised learning is the area of Machine Learning where we have a set of independent variables which helps us to analyse the dependent variable and the relation between them. Whatever we want to predict is called as Dependent Variable, while variables that we use to predict are called as Independent Variables. Suppose we want to predict the age of the person based on the person's height and weight, then height and weight will be the independent variables, while age will be the dependent variable.

Linear regression is a statistical technique to describe relationships between dependent variables with a number of independent variables. This tutorial will discuss the basic concepts of linear regression as well as its application within Python.

In order to give an understanding of the basics of the concept of linear regression, we begin with the most basic form of linear regression, i.e., "Simple linear regression".

## Simple Linear Regression

Simple linear regression (SLR) is a method to predict a response using one feature. It is believed that both variables are linearly linked. Thus, we strive to find a linear equation that can predict an answer value ( $y$ ) as precisely as possible in relation to features or the independently derived variable ( $x$ ).

For simplification, we define:

$x$  as **feature vector**, i.e.,  $x = [x_1, x_2, x_3, \dots, x_n]$ ,

$y$  as **response vector**, i.e.,  $y = [y_1, y_2, y_3, \dots, y_n]$

for observations (in above example,  $n=10$ ).

It is one of the most popular Supervised Machine Learning algorithms in Python that maintains an observation of continuous features and based on it, predicts an outcome. It establishes a relationship between dependent and independent variables by fitting a best line. This **best fit line is represented by a linear equation  $Y = a * X + b$** , commonly called the regression line.

In this equation,


**Y** – Dependent variable

**a** – Slope

**X** – Independent variable

**b** – Intercept

The regression line is the line that **fits best in the equation to supply a relationship between the dependent and independent variables**. When it runs on a single variable or feature, we call it **simple linear regression** and when it runs on different variables, we call it **multiple linear regression**. This is often used to estimate the cost of houses, total sales or total number of calls based on continuous variables.

 The picture can't be displayed









## EXPERIMENT 4

**Aim:** Implementation of K-nearest Neighbor

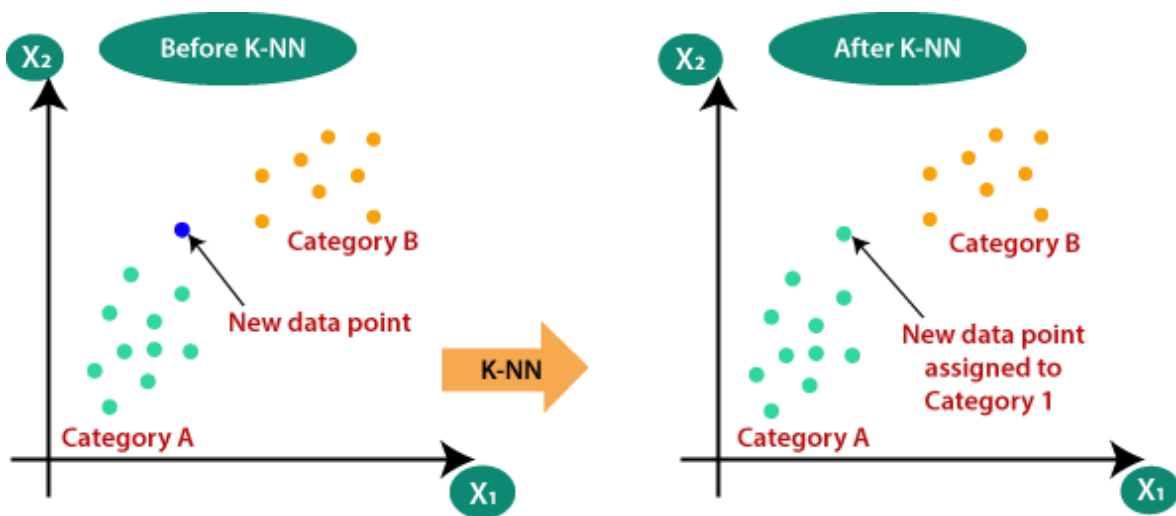
**Software Used:** Python

- K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning technique.
- K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories.
- K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm.
- K-NN algorithm can be used for Regression as well as for Classification but mostly it is used for the Classification problems.
- K-NN is **anon-parametric algorithm**, which means it does not make any assumption on underlying data.
- It is also called **alazy learner algorithm** because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset.
- KNN algorithm at the training phase just stores the dataset and when it gets new data, then it classifies that data into a category that is much similar to the new data.

The K-NN working can be explained on the basis of the below algorithm:

- **Step-1:** Select the number K of the neighbors
- **Step-2:** Calculate the Euclidean distance of **K number of neighbors**
- **Step-3:** Take the K nearest neighbors as per the calculated Euclidean distance.
- **Step-4:** Among these k neighbors, count the number of the data points in each category.
- **Step-5:** Assign the new data point to that category for which the number of the neighbor is maximum.
- **Step-6:** Our model is ready.













## **EXPERIMENT 5**

**Aim:** Implementing K-means Clustering

**Software Used:** Python

### **K-Means Clustering Algorithm**

K-Means Clustering is an unsupervised learning algorithm that is used to solve the clustering problems in machine learning or data science. In this topic, we will learn what is K-means clustering algorithm, how the algorithm works, along with the Python implementation of k-means clustering.

What is K-Means Algorithm?

K-Means Clustering is an Unsupervised Learning algorithm, which groups the unlabeled dataset into different clusters. Here K defines the number of pre-defined clusters that need to be created in the process, as if  $K=2$ , there will be two clusters, and for  $K=3$ , there will be three clusters, and so on.

It is an iterative algorithm that divides the unlabeled dataset into  $k$  different clusters in such a way that each dataset belongs only one group that has similar properties.

It allows us to cluster the data into different groups and a convenient way to discover the categories of groups in the unlabeled dataset on its own without the need for any training.

It is a centroid-based algorithm, where each cluster is associated with a centroid. The main aim of this algorithm is to minimize the sum of distances between the data point and their corresponding clusters.

The algorithm takes the unlabeled dataset as input, divides the dataset into  $k$ -number of clusters, and repeats the process until it does not find the best clusters. The value of  $k$  should be predetermined in this algorithm.

The k-means clustering algorithm mainly performs two tasks:

- Determine the best value for  $K$  center points or centroids by an iterative process.
- Assign each data point to its closest  $k$ -center. Those data points which are near to the particular  $k$ -center, create a cluster.

Hence each cluster has data points with some commonalities, and it is away from other clusters. The

below diagram explains the working of the K-means Clustering Algorithm:

How does the K-Means Algorithm Work?

The working of the K-Means algorithm is explained in the below steps:

**Step-1:** Select the number  $K$  to decide the number of clusters.

**Step-2:** Select random  $K$  points or centroids. (It can be either from the input dataset).

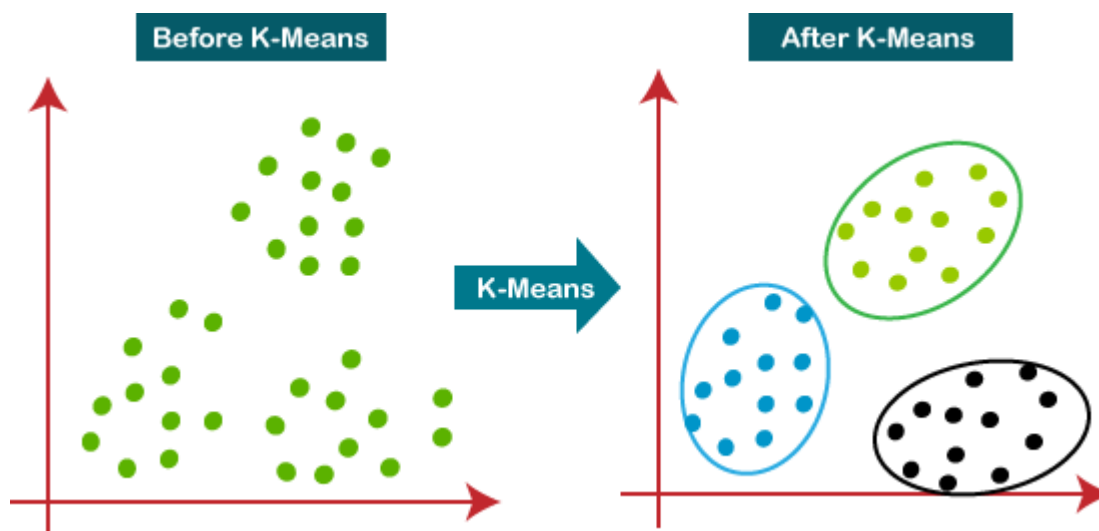
**Step-3:** Assign each data point to their closest centroid, which will form the predefined  $K$  clusters.

**Step-4:** Calculate the variance and place a new centroid of each cluster.

**Step-5:** Repeat the third steps, which means reassign each data point to the new closest centroid of each cluster.

**Step-6:** If any reassignment occurs, then go to step-4 else go to FINISH.

**Step-7:** The model is ready.



The other popularly used similarity measures are:-

1. **Cosine distance:** It determines the cosine of the angle between the point vectors of the two points in the  $n$ -dimensional space.
2. **Manhattan distance:** It computes the sum of the absolute differences between the coordinates of the two data points.
3. **Minkowski distance:** It is also known as the generalized distance metric. It can be used for both ordinal and quantitative variables.











## EXPERIMENT 6

**Aim:** Implementing Hierarchical Clustering

**Software Used:** Python

Implementing Hierarchical Clustering

Hierarchical clustering is another unsupervised machine learning algorithm, which is used to group the unlabeled datasets into a cluster and also known as **hierarchical cluster analysis** or HCA.

In this algorithm, we develop the hierarchy of clusters in the form of a tree, and this tree-shaped structure is known as the **dendrogram**.

Sometimes the results of K-means clustering and hierarchical clustering may look similar, but they both differ depending on how they work. As there is no requirement to predetermine the number of clusters as we did in the K-Means algorithm.

The hierarchical clustering technique has two approaches:

Play Video

1. **Agglomerative:** Agglomerative is a **bottom-up** approach, in which the algorithm starts with taking all data points as single clusters and merging them until one cluster is left.
2. **Divisive:** Divisive algorithm is the reverse of the agglomerative algorithm as it is a **top-down approach**.

### Why hierarchical clustering?

As we already have other clustering algorithms such as **K-Means Clustering**, then why we need hierarchical clustering? So, as we have seen in the K-means clustering that there are some challenges with this algorithm, which are a predetermined number of clusters, and it always tries to create the clusters of the same size. To solve these two challenges, we can opt for the hierarchical clustering algorithm because, in this algorithm, we don't need to have knowledge about the predefined number of clusters.

In this topic, we will discuss the Agglomerative Hierarchical clustering algorithm.

### Agglomerative Hierarchical clustering

The agglomerative hierarchical clustering algorithm is a popular example of HCA. To group the datasets into clusters, it follows the **bottom-up approach**. It means, this algorithm considers each dataset as a single cluster at the beginning, and then start combining the closest pair of clusters together. It does this until all the clusters are merged into a single cluster that contains all the datasets.

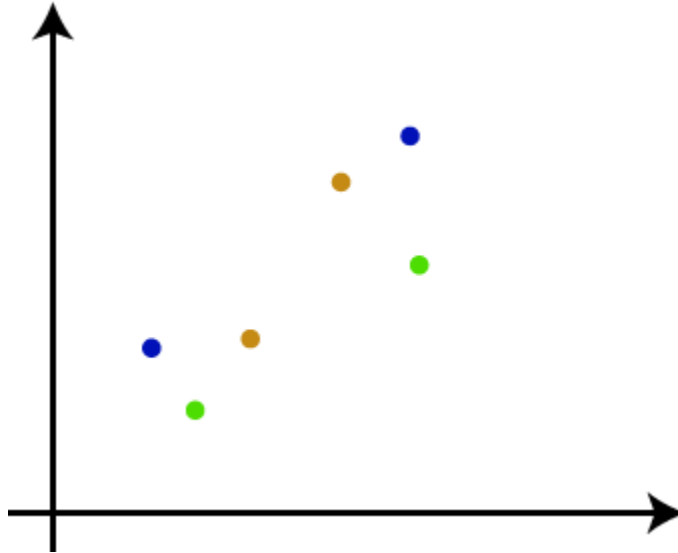
This hierarchy of clusters is represented in the form of the dendrogram.

## How the Agglomerative Hierarchical Clustering Work?

The working of the AH Algorithm can be explained using the below steps:

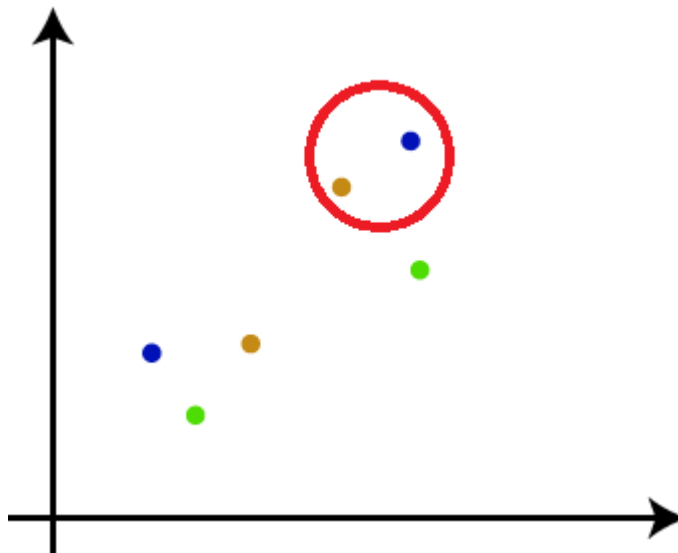
- **Step-1:** Create each data point as a single cluster. Let's say there are  $N$  data points, so the number of clusters will also be  $N$ .

- 



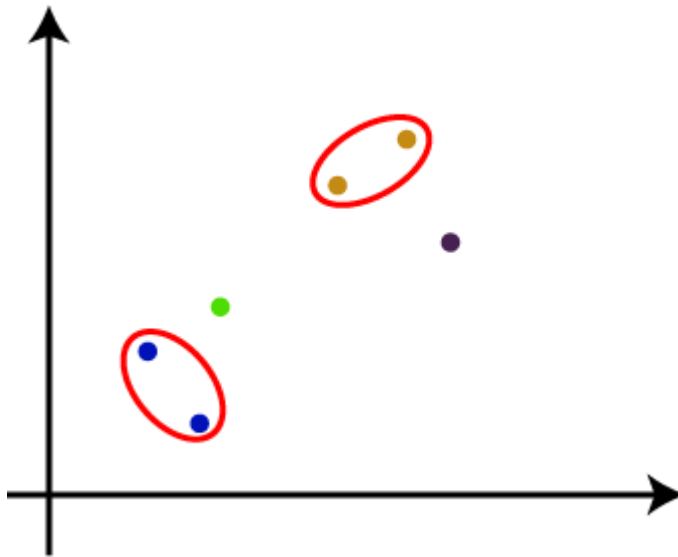
- **Step-2:** Take two closest data points or clusters and merge them to form one cluster. So, there will now be  $N-1$  clusters.

- 



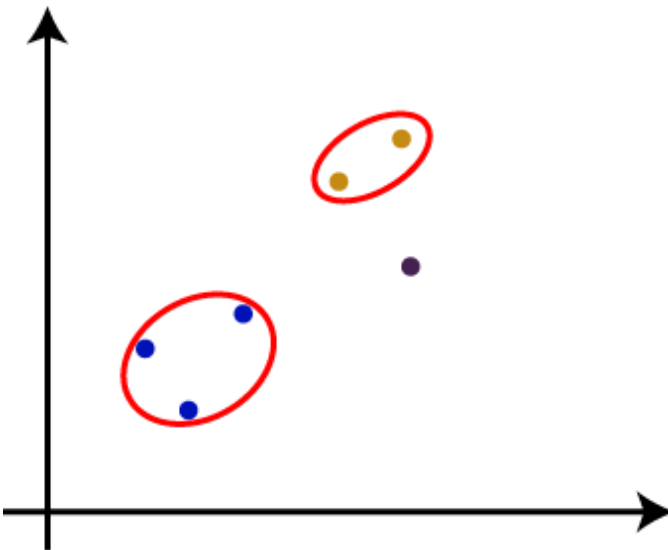
- **Step-3:** Again, take the two closest clusters and merge them together to form one cluster. There will be  $N-2$  clusters.

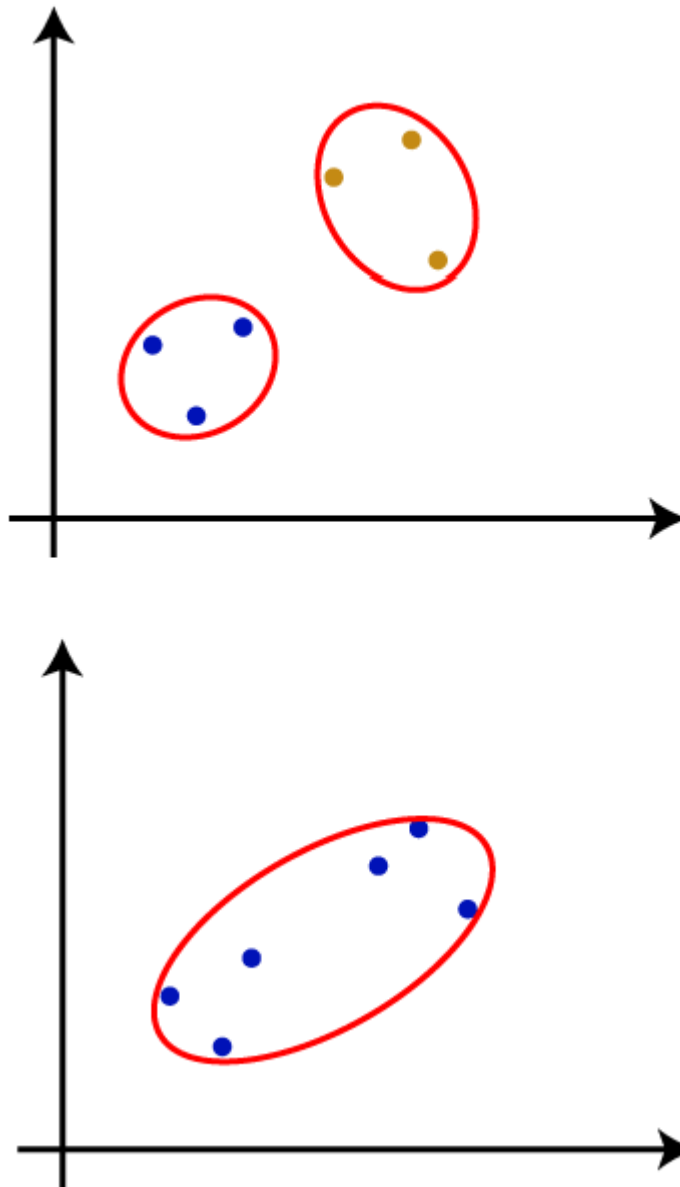
○



- **Step-4:** Repeat Step 3 until only one cluster left. So, we will get the following clusters. Consider the below images:

○



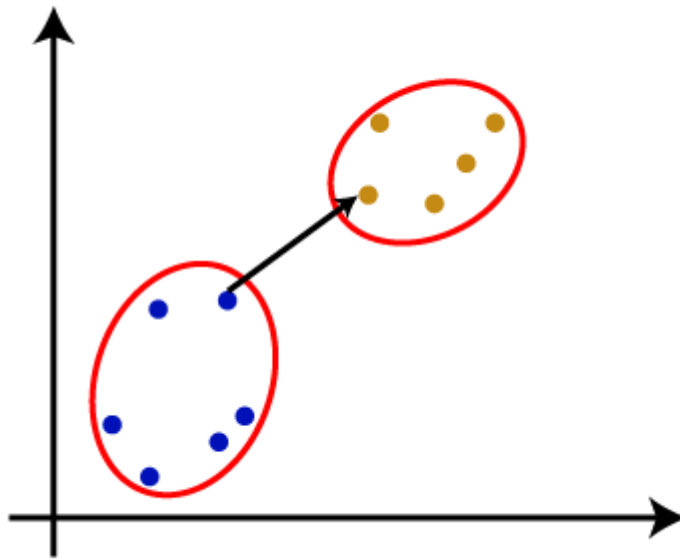


- **Step-5:** Once all the clusters are combined into one big cluster, develop the dendrogram to divide the clusters as per the problem.

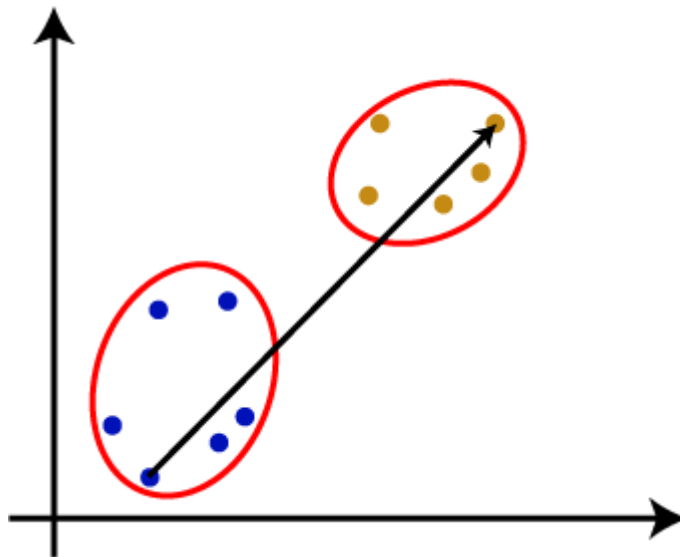
## Measure for the distance between two clusters

As we have seen, the **closest distance** between the two clusters is crucial for the hierarchical clustering. There are various ways to calculate the distance between two clusters, and these ways decide the rule for clustering. These measures are called **Linkage methods**. Some of the popular linkage methods are given below:

1. **Single Linkage:** It is the shortest distance between the closest points of the clusters. Consider the image:



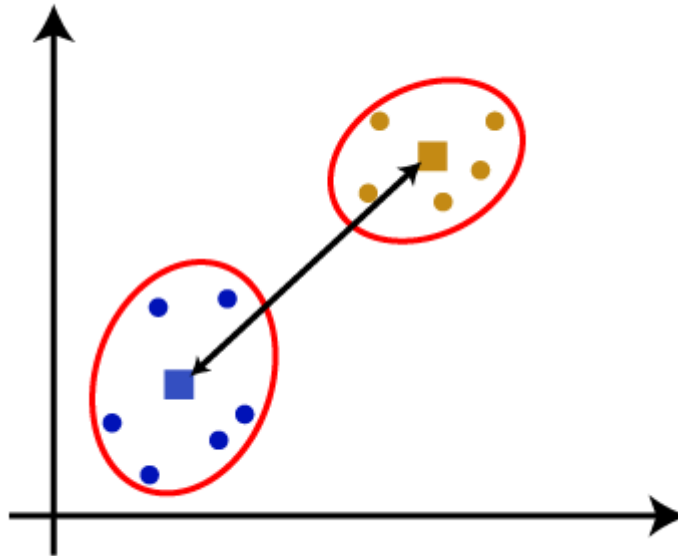
2. **Complete Linkage:** It is the farthest distance between the two points of two different clusters. It is one of the popular linkage methods as it forms tighter clusters than single-linkage.



3. **Average Linkage:** It is the linkage method in which the distance between each pair of datasets is added up and then divided by the total number of datasets to calculate the average distance between two clusters. It is also one of the most popular linkage methods.



4. **Centroid Linkage:** It is the linkage method in which the distance between the centroid of the clusters is calculated. Consider the below image:



From the above-given approaches, we can apply any of them according to the type of problem or business requirement.

## Working of Dendrogram in Hierarchical clustering

The dendrogram is a tree-like structure that is mainly used to store each step as a memory that the HC algorithm performs. In the dendrogram plot, the Y-axis shows the Euclidean distances between the data points, and the x-axis shows all the data points of the given dataset.







