DEPARTMENT OF COMPUTER AND COMMUNICATION ENGINEERING

VISION AND MISSION OF THE INSTITUTION

VISION

Global Leadership in Higher Education and Human Development

MISSION

- Be the most preferred university for innovative and interdisciplinary learning
- Foster academic, research, and professional excellence in all domains
- Transform young minds into competent professionals with good human values.

VISION AND MISSION OF THE CCE DEPARTMENT

VISION

Inculcate academic excellence and research aptitude for global competency and leadership

MISION

- Nurture the technical skillset of students to enable them to create an innovative solution
- Sharpen the problem-solving skills through project-based learning.
- Serve society by inspiring young minds for research and innovation with ethical values.

AI & Machine Learning Lab (CC3230)

Introduction: This course is designed to study and design the AI search algorithms and ML algorithms that allow computers to automatically learn from data or experience, how to improve their performance at some tasks (object classification etc). Students will also learn the fundamental methodology for how to design and analyse machine learning systems.

Course Outcomes: At the end of the course, students will be able to:

[CC3230.1]: Use Python/ Jupyter Notebook to implement AI/ML algorithms. (L3)
[CC3230.2]: Build Python programs to implement AI heuristic search techniques. (L3)
[CC3230.3]: Build ML models for publicly available datasets. (L3)
[CC3230.4]: Compare performance of ML models on performance metrics for improving employment prospects. (L4)

PROGRAM OUTCOMES AND PROGRAM SPECIFIC OUTCOMES

[PO.1]. **Engineering knowledge**: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems

[PO.2]. **Problem analysis**: Identify, formulate, research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences

[PO.3]. **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.

[PO.4]. **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.

[PO.5]. **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.

[PO.6]. **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.

[PO.7]. **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.

[PO.8]. **Ethics**: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practices.

[PO.9]. **Individual and team work**: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

[PO.10]. **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.

[PO.11]. **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 multidisciplinary environments

[PO.12]. **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 Specific Outcomes (PSOs)

[PSO.1] Imbibe the basic concepts and applications of computer-based Communication or networking, information sharing, signal processing, web-based systems, smart devices, and communication technology.

[PSO.2] Investigate prominent areas in the field of Computer and Communication Engineering to provide feasible solutions.

[PSO.3] Apply the contextual knowledge in the field of Computing and Communication to assess social, health, safety, and security issues relevant to the professional engineering practice.

Assessment Plan:

Criteria	Description	Maximum Marks
Internal Assessment (Formative)	Fortnightly evaluation (Record + execution + viva)	50
	Mini project	10
End Term Exam (Summative)	End Term Exam	40
	Total	100
Attendance (Formative)	A minimum of 75% Attendance is required to be maintained by a student to be qualified for taking up the End Semester examination. The allowance of 25% includes all types of leaves including medical leaves.	

DEPARTMENT OF COMPUTER AND COMMUNICATION ENGINEERING

LIST OF EXPERIMENTS

S. No.	Name of the Experiment
1 & 2	Introduction to Python
3	Write a program to implement hill climbing search algorithm
4	Write a program to implement A* search algorithm
5	Write a program to solve some real-world problem using constraint satisfaction
6 & 7	Write a program to Implement Simple Linear and Logistic Regression.
8	Write a program to implement the Bayes Classifier and SVM Classifier.
9	Write a program to implement Decision Tree Algorithm
10	Write a program to implement k-Nearest Neighbours
11	Write a program to implement k-means algorithm
12	Write a Program to implement Principal Component Analysis for dimensionality reduction
13	Write programs to Implement the Perceptron Algorithm Write a program to implement the Backpropagation Algorithm.
14	Mini Project

Program 1 & 2

Introduction to Python

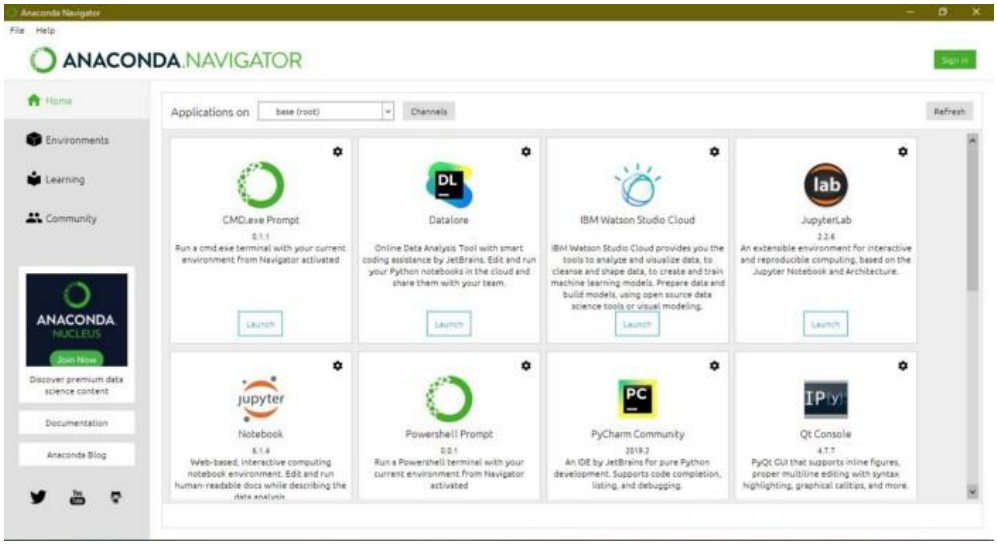
AIM: Brief introduction about fundamentals of Python programming to make student familiarize with basic building blocks of Python.

Description: Python is a high-level, interpreted, general-purpose programming language. Being a general-purpose language, it can be used to build almost any type of application with the right tools/libraries. Additionally, python supports objects, modules, threads, exception-handling, and automatic memory management which help in modelling real-world problems and building applications to solve these problems. It was created by Guido van Rossum and released in 1991.

How to install Python:

1. Install Python using Anaconda navigator: Anaconda Navigator is a desktop GUI interface that comes with Anaconda Individual Edition.

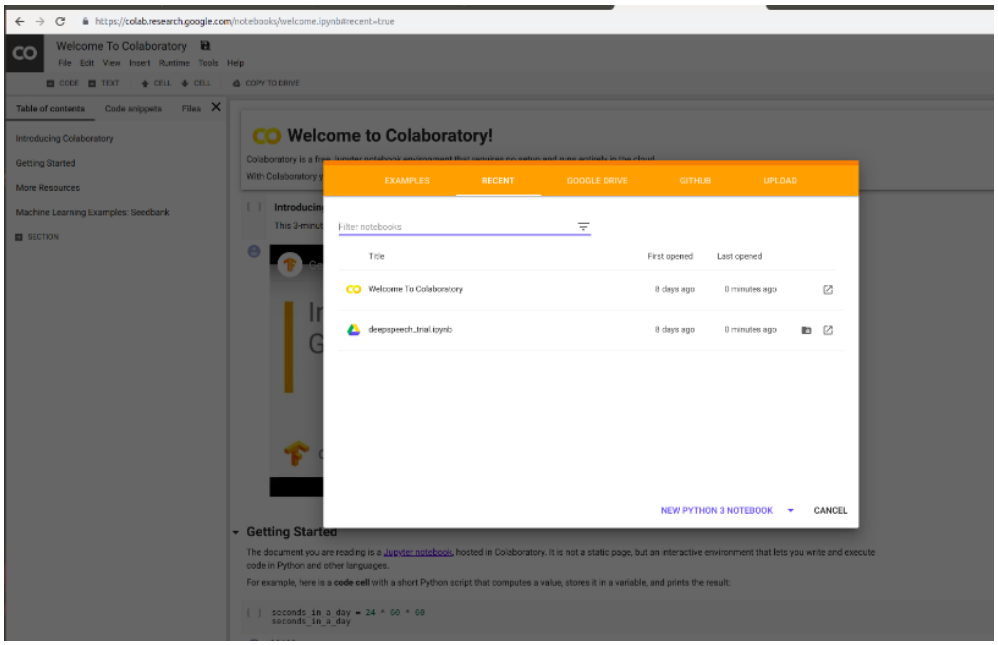
Step 1: Visit the official site for anaconda https://www.anaconda.com/
Step 2: Click on Product then select Anaconda Distribution (Open Source)
Step 3: Click on the download button
Step 4: Select your OS and Download a package
Step 5: After the installer gets downloaded, open it
Step 6: Start the installation in your system
Step 7: Kudos! ANACONDA has been successfully installed in your system.



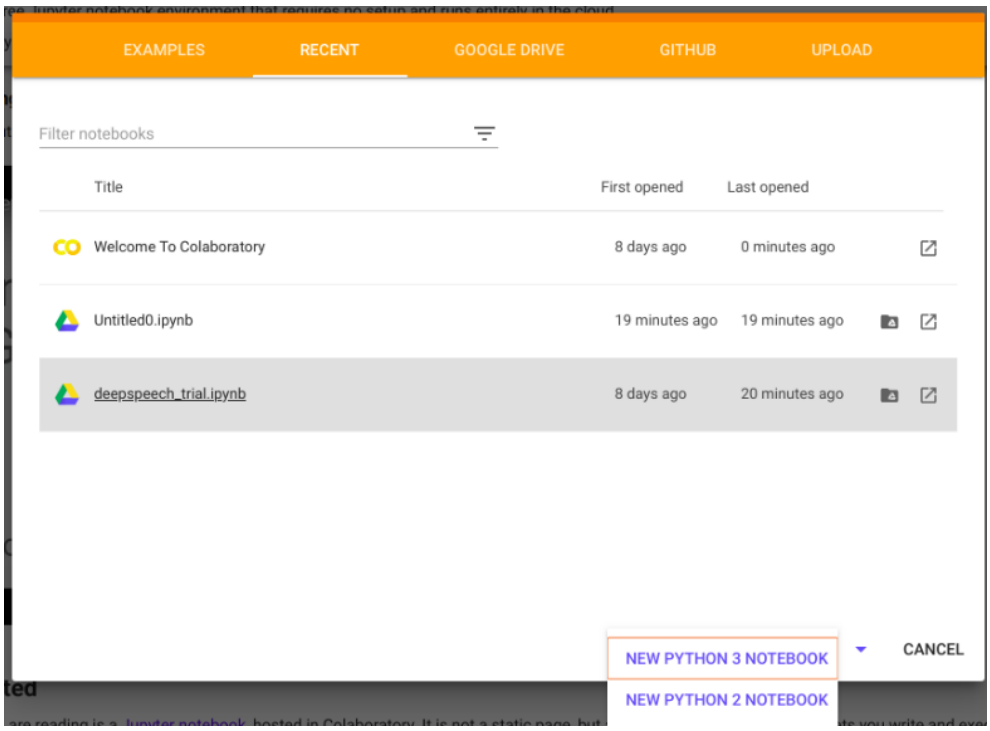
Happy Coding (click on jupyter notebook to start programming)

2. **Run Python Code on Google Colab:** Google provides Jupyter Notebook like interface to run Python code on online Virtual Machines.

Step 1: Open https://colab.research.google.com/

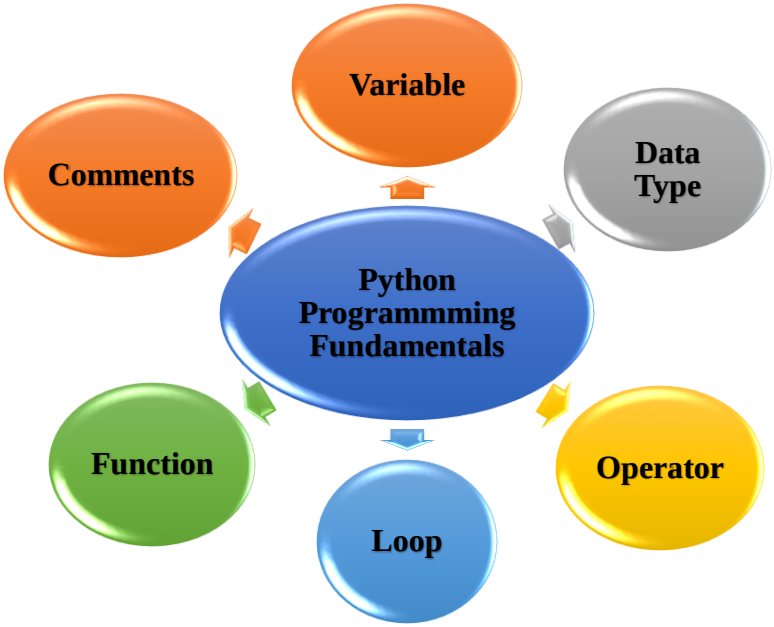


Step 2: Select New Python3 Notebook



Step 3: Start Typing code into the code cells.

Python Fundamentals



1. Variable/ Identifier

Variable is a name that is used to refer to memory location. Python variable is also known as an identifier and used to hold value. In Python, we don't need to specify the type of variable because Python is a infer language and smart enough to get variable type.

1.1 Naming Convention

- The first character of the variable must be an alphabet or underscore (_).
- All the characters except the first character may be an alphabet of lower-case(a-z), upper-case (A-Z), underscore, or digit (0-9).
- Identifier name must not contain any white-space, or special character (!, @, #, %, ^, &, *).
- Identifier name must not be similar to any keyword defined in the language.
- Identifier names are case sensitive; for example, my name, and MyName is not the same.

Examples of valid identifiers: a123, _n, n_9, etc.
Examples of invalid identifiers: 1a, n%4, n 9, etc.


Python is the highly object-oriented programming language; that's why every data item belongs to a specific type of class.

1. 1 Write a program in python to declare a variable

```
Number1 = 10
print(Number1)
Number2 = 32.5
print(Number2)
Name ="python"
print(Name)
```

OUTPUT

```
10
32.5
python
```

1.2 Program to get input from User

```
name = input("Enter your name: ")
# user entered the name 'Manipal'
print("hello", name)
```

OUTPUT

```
Enter your name: Manipal
Hello Manipal
```

1.3 Program to get input from user

```
# accepting integer from the user the return type of input() function is string , so
#we need to convert the input to integer
num1 = int(input("Enter num1: "))
num2 = int(input("Enter num2: "))
num3 = num1 * num2
print("Product is: ", num3)
```

OUTPUT

```
Enter num1:23
Enter num2:45
Product is:1035
```

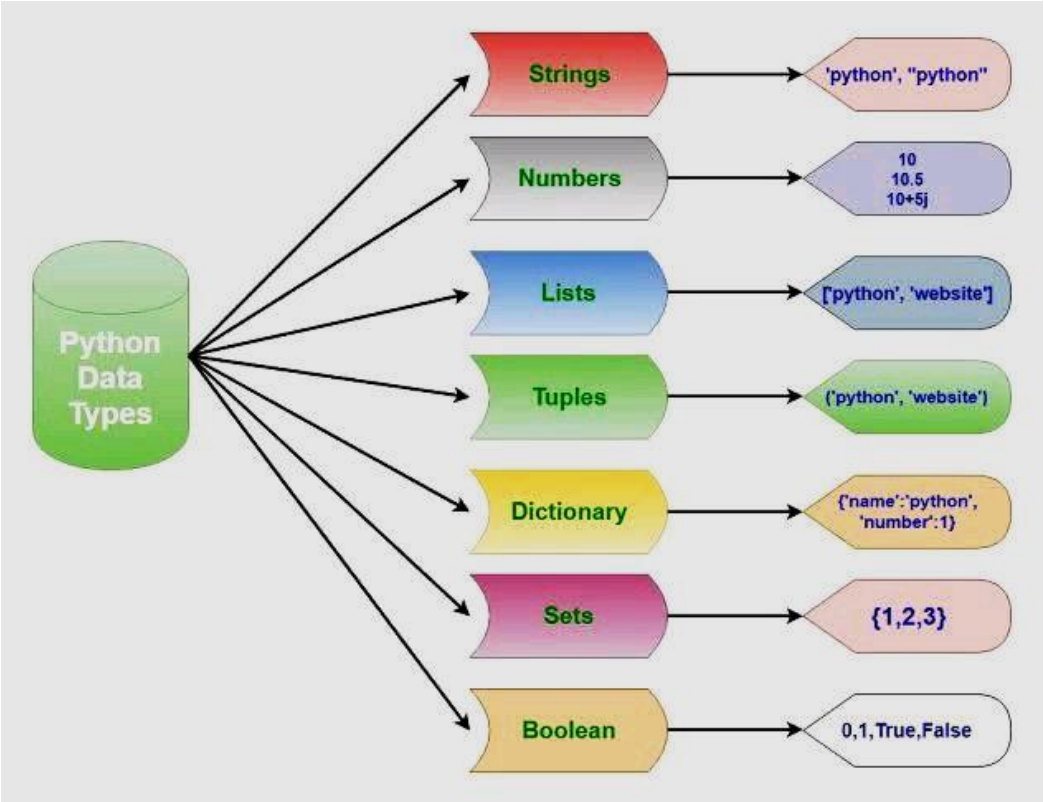
Python allows you to assign a single value to several variables simultaneously. For example a=b=c=1

Here, an integer object is created with the value 1, and all three variables are assigned to the same memory location. You can also assign multiple objects to multiple variables. For example –

a,b,c=1,2,“manipal”

Here, two integer objects with values 1 and 2 are assigned to variables a and b respectively, and one string object with the value "john" is assigned to the variable c.

2. Data Type



2.1 Strings

The string can be defined as the sequence of characters represented in the quotation marks. In Python, we can use single, double, or triple quotes to define a string.

1.4 Program to demonstrate String datatype in Python

```
str1 = 'Manipal University' #string str1
str2 = ' B.TECH CCE' #string str2
print (str1[0:2]) #printing first two character using slice operator
print (str1[4]) #printing 4th character of the string
print (str1*2) #printing the string twice
print (str1 + str2) #printing the concatenation of str1 and str2
```

OUTPUT

```
Ma
P
```

Manipal University Manipal University

Manipal University B.TECH CCE
h

2.2 Numbers

Number stores numeric values. The integer, float, and complex values belong to a Python Numbers datatype.

1.5 Program to demonstrate Number datatype in Python

```
a = 5
print("The type of a", type(a))
b = 40.5
print("The type of b", type(b))
c = 1+3j
print("The type of c", type(c))
```

OUTPUT

```
The type of a <class 'int'>
The type of b <class 'float'>
The type of c <class 'complex'>
```

2.3 List

Python Lists are like arrays in C. However, the list can contain data of different types. The items stored in the list are separated with a comma (,) and enclosed within square brackets []. list1 = [1, "hi", "Python", 2]

1.6 Program to demonstrate List datatype in Python

```
list1  = [1, "Manipal", "University", 2]
#Printing the list1
print (list1)
# List slicing
print (list1[3:])
# List slicing
print (list1[0:2])
# List Concatenation using + operator
print (list1 + list1)
# List repetition using * operator
print (list1 * 3)
```

OUTPUT

```
[1, 'Manipal', 'University', 2]
[2]
```

[1, 'Manipal']
[1, 'Manipal', 'University', 2, 1, 'Manipal', 'University', 2]
[1, 'Manipal', 'University', 2, 1, 'Manipal', 'University', 2, 1, 'Manipal', 'University', 2]

2.4 Tuples

A tuple is like the list in many ways. Like lists, tuples also contain the collection of the items of different data types. The items of the tuple are separated with a comma (,) and enclosed in parentheses (). A tuple is a read-only data structure as we can't modify the size and value of the items of a tuple. tup = ("hi", "Python", 2)

1.7 Program to demonstrate Tuples datatype in Python

```
tup  = ("Manipal", "University", 2)
# Checking type of tup
print (type(tup))
#Printing the tuple
print (tup)
# Tuple slicing
print (tup[1:])
print (tup[0:1])
# Tuple concatenation using + operator
print (tup + tup)
# Tuple repatation using * operator
print (tup * 3)
# Adding value to tup. It will throw an error.
tup[2] = "jaipur"
```

OUTPUT

```
<class 'tuple'>
('Manipal', 'University', 2)
('University', 2)
('Manipal',)
('Manipal', 'University', 2, 'Manipal', 'University', 2)
('Manipal', 'University', 2, 'Manipal', 'University', 2, 'Manipal', 'University', 2)
TypeError: 'tuple' object does not support item assignment
```

2.5 Dictionary

Dictionary is an unordered set of a key-value pair of items. It is like an associative array or a hash table where each key stores a specific value. Key can hold any primitive data type, whereas value is an arbitrary Python object. The items in the dictionary are separated with the comma (,) and enclosed in the curly braces {}.
d = {1:'Jimmy', 2:'Alex', 3:'john', 4:'mike'}

1.8 Program to demonstrate Dictionary datatype in Python

```
d = {1:'Jaipur', 2:'Ajmer', 3:'Jodhpur', 4:'Udaipur'}
```

```
# Printing dictionary
print (d)
# Accesing value using keys
print("1st name is "+d[1])
print("2nd name is "+ d[4])
print (d.keys())
print (d.values())
```

OUTPUT

```
{1: 'Jaipur', 2: 'Ajmer', 3: 'Jodhpur', 4: 'Udaipur'}
1st name is Jaipur
2nd name is Udaipur
dict_keys([1, 2, 3, 4])
dict_values(['Jaipur', 'Ajmer', 'Jodhpur', 'Udaipur'])
```

2.6 Sets

Python Set is the unordered collection of the data type. It is iterable, mutable (can modify after creation), and has unique elements. In set, the order of the elements is undefined; it may return the changed sequence of the element.

1.9 Program to demonstrate Set datatype in Python

```
# Creating Empty set
set1 = set()
set2 = {'James', 2, 3,'Python'}
#Printing Set value
print(set2)
# Adding element to the set
set2.add(10)
print(set2)
#Removing element from the set
set2.remove(2)
print(set2)
```

OUTPUT

```
{'James', 'Python', 2, 3}
{'Python', 2, 3, 10, 'James'}
{'Python', 3, 10, 'James'}
```

2.7 Boolean

Boolean type provides two built-in values, True and False. These values are used to determine the given statement true or false. It denotes by the class bool. True can be represented by any non-zero value or 'T' whereas false can be represented by the 0 or 'F'.

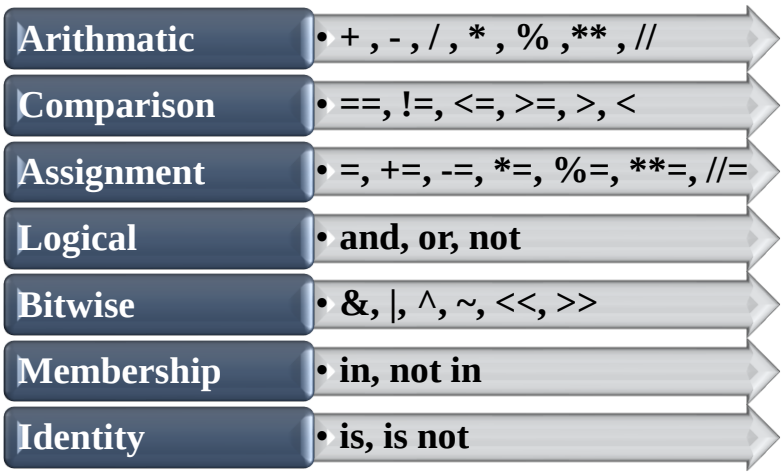
1.10 Program to demonstrate Boolean datatype in Python

```
print(type(True))
print(type(False))
print(false)
```

OUTPUT

```
<class 'bool'>
<class 'bool'>
NameError: name 'false' is not defined
```

3. Operator



3.1 Arithmetic Operator

1.11 Program to demonstrate Arithmetic Operator in python

```
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
```

3.2 Comparison Operator

1.12 Program to demonstrate comparison operator in python

```
x = 10
y = 12
print('x > y is',x>y)
print('x < y is',x<y)
print('x == y is',x==y)
print('x != y is',x!=y)
print('x >= y is',x>=y)
print('x <= y is',x<=y)
```

OUTPUT

```
x > y is False
x < y is True
x == y is False
x != y is True
x >= y is False
x <= y is True
```

3.3 Assignment Operator

1.13 Program to demonstrate Assignment operator in python

```
a = 10
# Assign value
b = a
print(b)
# Add and assign value
b += a
print(b)
# Subtract and assign value
b -= a
print(b)
# multiply and assign
b *= a
```

```
print(b)
# bitwise lishift operator
b <<= a
print(b)
```

OUTPUT

```
10
20
10
100
102400
```

3.4 Logical Operator

1.14 Program to demonstrate Logical Operator in python

```
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
```

3.5 Bitwise Operator

1.15 Program to demonstrate Bitwise Operator in python

```
a = 10
b = 4
# Print bitwise AND operation
print(a & b)
 # Print bitwise OR operation
print(a | b)
 # Print bitwise NOT operation
print(~a)
 # print bitwise XOR operation
print(a ^ b)
 # print bitwise right shift operation
print(a >> 2)
 # print bitwise left shift operation
print(a << 2)
```

OUTPUT

```
0
```

14

-11

14

2

40

3.6 Membership Operator

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

1.16 Program to demonstrate Membership Operator in python

```
x = 24
y = 20
list = [10, 20, 30, 40, 50]
if (x not in list):
   print("x is NOT present in given list")
else:
   print("x is present in given list")
if (y in list):
   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

3.7 Identity Operator

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

1.17 Program to demonstrate Identity Operator in python

```
a = 10
b = 20
c = a
print(a is not b)
print(a is c)
```

OUTPUT

TRUE
TRUE

4. Loop

4.1 if-else-if

Pyhton if -else statement is used for decision making

SYNTAX

```
if test expression:
   Body of if
elif test expression:
   Body of elif
else:
   Body of else
```

1.18 Program to demonstrate decision making using if else in python

'''In this program, we check if the number is positive or negative or zero and display an appropriate message'''

```
num = 3.4
# Try these two variations as well:
# num = 0
# num = -4.5
if num > 0:
   print("Positive number")
elif num == 0:
   print("Zero")
else:
   print("Negative number")
```

OUTPUT

Positive number

4.2 FOR loop

The for loop in Python is used to iterate over a sequence.

SYNTAX

```
for val in sequence:
   loop body
```

1.19 Program to demonstrate for loop in python

```
# Program to find the sum of all numbers stored in a list
```

```
# List of numbers
numbers = [6, 5, 3, 8, 4, 2, 5, 4, 11]
# variable to store the sum
sum = 0
# iterate over the list
for val in numbers:
        sum = sum+val
print("The sum is", sum)
```

OUTPUT

The sum is 48

We can generate a sequence of numbers using range() function. range(10) will generate numbers from 0 to 9 (10 numbers). We can also define the start, stop and step size as range(start, stop,step_size). step_size defaults to 1 if not provided.

1.20 Program to demonstrate range function in python

```
print(range(10))
print(list(range(10)))
print(list(range(2, 8)))
print(list(range(2, 20, 3)))
```

OUTPUT

```
range(0, 10)
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
[2, 3, 4, 5, 6, 7]
[2, 5, 8, 11, 14, 17]
```

We can use the range() function in for loops to iterate through a sequence of numbers.

1.21 Program to demonstrate use of range function with for loop

```
genre = ['pop', 'rock', 'jazz']
# iterate over the list using index
for i in range(len(genre)):
   print("I like", genre[i])
```

OUTPUT

```
I like pop
I like rock
I like jazz
```

4.3 While loop

The while loop in Python is used to iterate over a block of code as long as the test expression (condition) is true.

SYNTAX

```
while test_expression:
   Body of while
```

1.22 Program to demonstrate while loop in python

 # Program to add natural numbers up to n sum = 1+2+3+...+n

```
# To take input from the user,
# n = int(input("Enter n: "))
n = 10
# initialize sum and counter
sum = 0
i = 1
while i <= n:
    sum = sum + i
    i = i+1    # update counter
# print the sum
print("The sum is", sum)
```

OUTPUT

```
Enter n: 10
The sum is 55
```

5. Function

In Python, a function is a group of related statements that performs a specific task.

SYNTAX

```
def function_name(parameters):
    """docstring"""
    statement(s)
```

Above shown is a function definition that consists of the following components.

- Keyword def that marks the start of the function header.
- A function name to uniquely identify the function. Function naming follows the same rules of writing identifiers in Python.
- Parameters (arguments) through which we pass values to a function. They are optional.
- A colon (:) to mark the end of the function header.
- Optional documentation string (docstring) to describe what the function does.
- One or more valid python statements that make up the function body. Statements must have the same indentation level (usually 4 spaces).
- An optional return statement to return a value from the function.

1.23 Program to demonstrate use of function in python

```
def add_numbers(x,y):
  sum = x + y
  return sum
num1 = 5
num2 = 6
print("The sum is", add_numbers(num1, num2))
```

OUTPUT

```
Enter a number: 2.4
Enter another number: 6.5
The sum is 8.9
```

6. Comments

- Comments can be used to explain Python code.
- Comments can be used to make the code more readable.
- Comments can be used to prevent execution when testing code.

Comments starts with a #, and Python will ignore them:

1.24 Program to demonstrate single line comment in python

```
#This is a comment
print("Hello, World!")
```

OUTPUT

```
HELLO, World
```

Python does not really have a syntax for multi line comments.
To add a multiline comment you could insert a # for each line

1.25 Program to demonstrate multi line comment in python

```
#This is a comment
#written in
#more than just one line
print("Hello, World!")
```

OUTPUT

```
Hello, World
```

Program 3

Program to implement hill climbing search algorithm

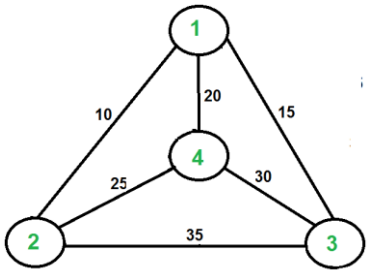
AIM: Write a Python code to implement Hill Climbing Search Algorithm specifically to solve Travelling salesman problem.

Description: Search algorithms are algorithms that help in solving search problems. A search problem consists of a search space, start state, and goal state. Search algorithms help the AI agents to attain the goal state through the assessment of scenarios and alternatives. The algorithms provide search solutions through a sequence of actions that transform the initial state to the goal state. Without these algorithms, AI machines and applications cannot implement search functions and find viable solutions.

What is Travelling Salesman Problem

In the Travelling salesman problem, we have a salesman who needs to visit a number of cities exactly once, after which he returns to the first city. The distances between each pair of cities are known, and we need to find the shortest route.

EXAMPLE: Given a set of cities {1,2,3,4} and the distance between every pair of cities, the problem is to find the shortest possible route that visits every city exactly once and returns to the starting point.



A TSP tour in the graph is 1-2-4-3-1. The cost of the tour is 10+25+30+15 which is 80.

Here we will solve this problem using Hill Climbing Algorithm

What is Hill Climbing Algorithm

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

Algorithm for Simple Hill Climbing:

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 step2.

Step 5: Exit.

3.1 Python Code to implement hill climbing algorithm to solve travelling salesman problem

```
import random
def randomSolution(tsp):
cities = list(range(len(tsp)))
solution = []

for i in range(len(tsp)):
randomCity = cities[random.randint(0, len(cities) - 1)]
solution.append(randomCity)
cities.remove(randomCity)

return solution

def routeLength(tsp, solution):
routeLength = 0
for i in range(len(solution)):
routeLength += tsp[solution[i - 1]][solution[i]]
return routeLength

def getNeighbours(solution):
neighbours = []
for i in range(len(solution)):
for j in range(i + 1, len(solution)):
neighbour = solution.copy()
neighbour[i] = solution[j]
neighbour[j] = solution[i]
neighbours.append(neighbour)
return neighbours

def getBestNeighbour(tsp, neighbours):
bestRouteLength = routeLength(tsp, neighbours[0])
bestNeighbour = neighbours[0]
for neighbour in neighbours:
currentRouteLength = routeLength(tsp, neighbour)
if currentRouteLength < bestRouteLength:
bestRouteLength = currentRouteLength
bestNeighbour = neighbour
return bestNeighbour, bestRouteLength

def hillClimbing(tsp):
currentSolution = randomSolution(tsp)
currentRouteLength = routeLength(tsp, currentSolution)
neighbours = getNeighbours(currentSolution)
```

```
bestNeighbour, bestNeighbourRouteLength = getBestNeighbour(tsp, neighbours)

while bestNeighbourRouteLength < currentRouteLength:
currentSolution = bestNeighbour
currentRouteLength = bestNeighbourRouteLength
neighbours = getNeighbours(currentSolution)
bestNeighbour, bestNeighbourRouteLength = getBestNeighbour(tsp, neighbours)

return currentSolution, currentRouteLength

def main():
tsp = [
[0, 400, 500, 300],
[400, 0, 300, 500],
[500, 300, 0, 400],
[300, 500, 400, 0]
]

print(hillClimbing(tsp))
if __name__ == "__main__":
main()
```

OUTPUT: ([2, 3, 0, 1], 1400)

Program 4

A* search algorithm

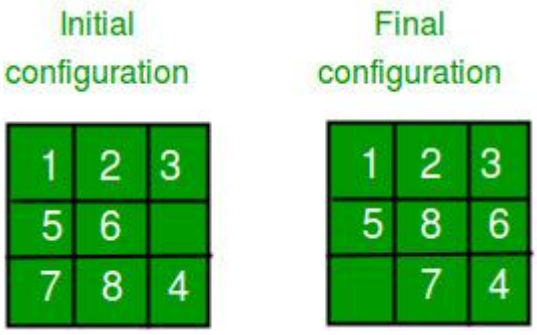
AIM: Write a Python code to implement A* Search Algorithm specifically to solve 3X3- 8 Puzzle problem.

Description: A* search algorithm is an uninformed search algorithm which works on the concept of heuristic function.

What is 8- Puzzle Problem

Given a 3×3 board with 8 tiles (every tile has one number from 1 to 8) and one empty space. The objective is to place the numbers on tiles to match the final configuration using the empty space. We can slide four adjacent (left, right, above, and below) tiles into the empty space.

For example,



We also know the eight puzzle problem by the name of N puzzle problem or sliding puzzle problem. N-puzzle that consists of N tiles (N+1 titles with an empty tile) where N can be 8, 15, 24 and so on.

What is A* Algorithm

A* search algorithm finds the shortest path through the search space using the heuristic function. In A* algorithm heuristic function f(n) can be denoted as the combination of g(n)+h(n) where g(n): cost to reach n^{th} node from start node and h(n): cost to reach destination node from current or n^{th} node.

Algorithm for A* search algorithm:

Step1: Place the starting node in the OPEN list.
Step 2: Check if the OPEN list is empty or not, if the list is empty then return failure and stops.
Step 3: Select the node from the OPEN list which has the smallest value of evaluation function (g+h), if node n is goal node then return success and stop, otherwise

Step 4: Expand node n and generate all of its successors, and put n into the closed list. For each successor n', check whether n' is already in the OPEN or CLOSED list, if not then compute evaluation function for n' and place into Open list.
Step 5: Else if node n' is already in OPEN and CLOSED, then it should be attached to the back pointer which reflects the lowest g(n') value.
Step 6: Return to Step 2.

4.1 Python Code to implement A* search algorithm to solve 8-puzzle problem

```
# swapping of tiles
def move(ar,p,st):
rh=99999
store_st=st.copy()
for i in range(len(ar)):
dupl_st=st.copy()

temp=dupl_st[p]
dupl_st[p]=dupl_st[arr[i]]
dupl_st[arr[i]]=temp

tmp_rh=count(dupl_st)

if tmp_rh<rh:
rh=tmp_rh
store_st=dupl_st.copy()
return store_st,tmp_rh

#print in format function
def print_in_format(matrix):
for i in range(9):
if i%3==0 and i>0:
print("")
print(str(matrix[i])+" ", end= " ")

# function to count number of misplaced tiles
def count(s):
c=0
ideal=[1,2,3,
4,5,6,
7,8,0]
for i in range(9):
if s[i]!=0 and s[i]!=ideal[i]:
c=c+1
return c

# create the problem space

start=[1,2,3,
0,5,6,
4,7,8]
```

```
h=count(start)
level=1
print("\n.....Level" + str(level)+".....")
print_in_format(start)
print("\n Heuristic Value (misplace tiles): " +str(h))

# define moves
while h>0:
pos=int(start.index(0))
level+=1
if pos==0:
arr=[1,3]
start,h=move(arr,pos,start)
elif pos==1:
arr=[0,2,4]
start,h=move(arr,pos,start)
elif pos==2:
arr=[1,5]
start,h=move(arr,pos,start)
elif pos==3:
arr=[0,4,6]
start,h=move(arr,pos,start)
elif pos==4:
arr=[1,3,5,7]
start,h=move(arr,pos,start)
elif pos==5:
arr=[2,4,8]
start,h=move(arr,pos,start)
elif pos==6:
arr=[3,7]
start,h=move(arr,pos,start)
elif pos==7:
arr=[4,6,8]
start,h=move(arr,pos,start)
elif pos==8:
arr=[5,7]
start,h=move(arr,pos,start)
print("\n.....Level " +str(level)+".....")
print_in_format(start)
print("\nHeuristic value (misplaced tiles): " +str(h))
```

OUTPUT:

```
..................Level1.....................
1  2  3
0  5  6
4  7  8
 Heuristic Value (misplace tiles): 3

...........Level 2..............
1  2  3
4  5  6
0  7  8
Heuristic value (misplaced tiles): 2

...........Level 3..............
1  2  3
4  5  6
7  0  8
Heuristic value (misplaced tiles): 1

...........Level 4..............
1  2  3
4  5  6
7  8  0
Heuristic value (misplaced tiles): 0
```

Program 5

Constraint Satisfaction Problem

AIM: Write a Python code to solve some algebraic relations using constraint satisfaction.

Description: Constraint satisfaction is a technique where a problem is solved when its values satisfy certain constraints or rules of the problem. Constraint satisfaction depends on three components, namely:
X: It is a set of variables.
D: It is a set of domains where the variables reside. There is a specific domain for each variable.
C: It is a set of constraints which are followed by the set of variables.
In constraint satisfaction, domains are the spaces where the variables reside, following the problem specific constraints. These are the three main elements of a constraint satisfaction technique. The constraint value consists of a pair of {scope, rel}. The scope is a tuple of variables which participate in the constraint and rel is a relation which includes a list of values which the variables can take to satisfy the constraints of the problem.

5.1 Python Code to implement Constraint Satisfaction Problem to solve an algebraic equation given below

```
# Find the value of x and y such that x=[1,2,3] and value of y can be from 0 to 9 and x+y>=10

import constraint
from constraint import *
problem=Problem()
problem.addVariable("x",[1,2,3])
problem.addVariable("y",[0,1,2,3,4,5,6,7,8,9])
def my_constraint(x,y):
if x+y>=10:
return True
problem.addConstraint(my_constraint,['x','y'])
s=problem.getSolutions()
for i in s:
print(i)
```

OUTPUT

```
{'x': 3, 'y': 9}
{'x': 3, 'y': 8}
{'x': 3, 'y': 7}
{'x': 2, 'y': 9}
{'x': 2, 'y': 8}
{'x': 1, 'y': 9}
```

Program 6 & 7

Linear Regression & Logistic Regression

AIM: Write a Python code to implement linear and logistic regression concept for binary and multiclass classification.

Description:

Linear Regression

Linear Regression is a supervised Machine Learning model in which the model finds the best fit linear line between the independent and dependent variable i.e it finds the linear relationship between the dependent and independent variable. Linear Regression is of two types: Simple and Multiple. Simple Linear Regression is where only one independent variable is present and the model has to find the linear relationship of it with the dependent variable Whereas, In Multiple Linear Regression there are more than one independent variables for the model to find the relationship.

Equation of Simple Linear Regression,

$$y = b_o + b_1 x$$

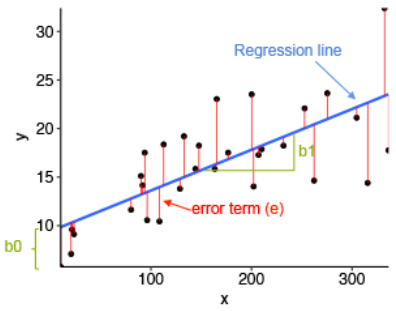
where bo is the intercept, b1 is coefficient or slope, x is the independent variable and y is the dependent variable.

Equation of Multiple Linear Regression,

$$y = b_o + b_1 x_1 + b_2 x_2 + b_3 x_3 \ldots . + b_n x_n$$

where bo is the intercept, b1,b2,b3,b4…,bn are coefficients or slopes of the independent variables x1,x2,x3,x4…,xn and y is the dependent variable.

Error is the difference between the actual value and Predicted value and the goal is to reduce this difference



In the above diagram,

- x is our dependent variable which is plotted on the x-axis and y is the dependent variable which is plotted on the y-axis.
- Black dots are the data points i.e the actual values.
- bo is the intercept which is 10 and b1 is the slope of the x variable.
- The blue line is the best fit line predicted by the model i.e the predicted values lie on the blue line.
- The vertical distance between the data point and the regression line is known as error or residual. Each data point has one residual and the sum of all the differences is known as the Sum of Residuals/Errors.

6.1 Program to demonstrate Linear Regression (single variable) using python

For the given dataset find out the price for the homes whose area is 3300 Sq feet and 5000 Sq feet.

Area (Sq feet)	Price
2600	550000
3000	565000
3200	610000
3600	680000
4000	725000

This dataset can be downloaded from the given link
https://raw.githubusercontent.com/codebasics/py/master/ML/1_linear_reg/homeprices.csv

Source Code

```
# import libraries
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import linear_model

# access database from github/kaggle
url=(r'https://raw.githubusercontent.com/codebasics/py/master/ML/1_linear_reg/homeprices.csv')

# read database from the specified url
df=pd.read_csv(url)

# visualize the database through scatterplot
df
plt.xlabel('area')
plt.ylabel('price')
plt.scatter(df.area,df.price,color='red',marker='+')

#drop price column from the database
new_df=df.drop('price',axis='columns')
```

```
new_df
price=df.price
price

# create linear regression
reg=linear_model.LinearRegression()
reg.fit(new_df,price)
reg.predict([[3300]])
reg.predict([[5000]])
```

OUTPUT:

```
628715.75342466
859554.79452055
```

6.2 Program to demonstrate Linear Regression (multiple variable) using python

For the given dataset find out the price for the homes whose area is

a. 3000 Sq feet area, 3 bedroom, 40 year old
b. 2500 Sq feet area, 4 bedroom, 5 year old

Area (Sq feet)	Bedroom	Age	Price
2600	3	20	550000
3000	4	25	565000
3200		28	610000
3600	3	30	680000
4000	5	8	725000

This dataset can be downloaded from the given link
https://raw.githubusercontent.com/codebasics/py/master/ML/2_linear_reg_multivariate/homeprices.csv

Source Code

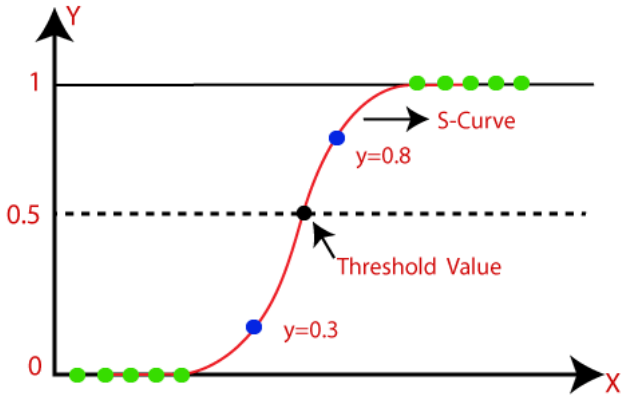
```
# import libraries
import pandas as pd
from sklearn import linear_model

#read dataset
url=(r'https://raw.githubusercontent.com/codebasics/py/master/ML/2_linear_reg_multivariate/homeprices.csv')
df=pd.read_csv(url)
df
df.bedrooms.median()
df.bedrooms=df.bedrooms.fillna(0)
df
reg=linear_model.LinearRegression()
reg.fit(df.drop('price',axis='columns'),df.price)
reg.predict([[3000, 3, 40]])
reg.predict([[2500,4,5]])
```

OUTPUT:
498408.25158031
578876.03748933

Logistic Regression

- Logistic regression is a type of Supervised Learning technique. It is used for predicting the categorical dependent variable using a given set of independent variables.
- Logistic regression predicts the output of a categorical dependent variable. Therefore the outcome must be a categorical or discrete value. It can be either Yes or No, 0 or 1, true or False, etc. but instead of giving the exact value as 0 and 1, **it gives the probabilistic values which lie between 0 and 1**.
- Logistic Regression is much similar to the Linear Regression except that how they are used. Linear Regression is used for solving Regression problems, whereas **Logistic regression is used for solving the classification problems**.
- In Logistic regression, instead of fitting a regression line, we fit an "S" shaped logistic function, which predicts two maximum values (0 or 1).
- The curve from the logistic function indicates the likelihood of something such as whether the cells are cancerous or not, a mouse is obese or not based on its weight, etc.
- Logistic regression uses the concept of predictive modeling as regression; therefore, it is called logistic regression, but is used to classify samples; Therefore, it falls under the classification algorithm.

The below image is showing the logistic function:



Logistic Function (Sigmoid Function)

- The sigmoid function is a mathematical function used to map the predicted values to probabilities.
- It maps any real value into another value within a range of 0 and 1.
- The value of the logistic regression must be between 0 and 1, which cannot go beyond this limit, so it forms a curve like the "S" form. The S-form curve is called the Sigmoid function or the logistic function.
- In logistic regression, we use the concept of the threshold value, which defines the probability of either 0 or 1. Such as values above the threshold value tends to 1, and a value below the threshold values tends to 0.

Logistic Regression Equation:

The Logistic regression equation can be obtained from the Linear Regression equation. The mathematical steps to get Logistic Regression equations are given below:

Equation of the straight line can be written as:

$$y= b_0 + b_1x_1 + b_2x_2 + b_3x_3 + \cdots + b_nx_n$$

In Logistic Regression y can be between 0 and 1 only, so for this divide the above equation by (1-y):

$$\frac{y}{1-y}\text{; 0 for y= 0, and infinity for y=1}$$

But we need range between -[infinity] to +[infinity], then take logarithm of the equation it will become:

$$log\left[\frac{y}{1-y}\right] = b_0 + b_1x_1 + b_2x_2 + b_3x_3 + \cdots + b_nx_n$$

The above equation is the final equation for Logistic Regression.

Type of Logistic Regression

On the basis of the categories, Logistic Regression can be classified into two types:

Binomial: In binomial Logistic regression, there can be only two possible types of the dependent variables, such as 0 or 1, Pass or Fail, etc.

Multinomial: In multinomial Logistic regression, there can be 3 or more possible unordered types of the dependent variable, such as "cat", "dogs", or "sheep"

7.1 Program to implement Logistic Regression (Binary Class classification) in pyhton

For the given dataset Predict that if a person would buy life insurance based on his age. The dataset can be downloaded from the given link
py/insurance_data.csv at master · codebasics/py · GitHub

Source Code

```
# import required libraries
import pandas as pd
import matplotlib.pyplot as plt
```

```
# read dataset
df=pd.read_csv(r'C:\Users\punamk\Desktop\insurancedata.csv')
df

# plot dataset
plt.scatter(df.age,df.bought_insurance,color='blue',marker='*')
plt.xlabel('age')
plt.ylabel('bought_insurance')

# split the dataset in to training and testing (80:20 ratio)
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(df[['age']],df.bought_insurance,train_s
ize=0.8)

# implement logistic regression
from sklearn.linear_model import LogisticRegression
model=LogisticRegression()

# train the model
model.fit(x_train,y_train)

# make prediction and get the output
y=model.predict(x_test)
y
```

OUTPUT

```
1, 0, 1, 0, 1, 0
```

7.2 Program to implement Logistic Regression (Multi Class classification) in phyton.

Identify a particular digit from the digit dataset. Dataset is already present in sklearn library

Source Code

```
# import required packages
import matplotlib.pyplot as plt
from sklearn.datasets import load_digits
digits=load_digits()

# check description of dataset and print some random images
dir(digits)
digits.data[1]
plt.gray()
for i in range(5):
plt.matshow(digits.images[i])

# split the dataset in to 80:20 training and testing ratio
from sklearn.model_selection import train_test_split
```

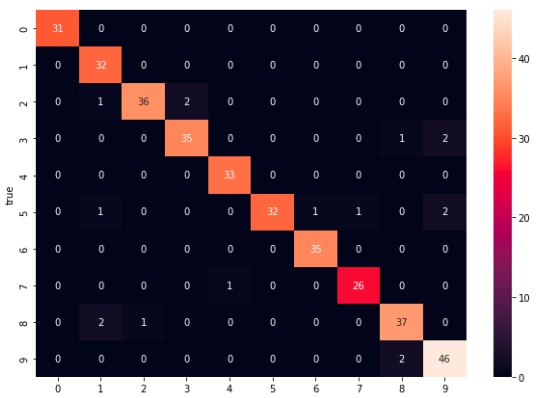
```
x_train,x_test,y_train,y_test=train_test_split(digits.data,digits.target,test_size=0.2)

# import and implement logistic regression
from sklearn.linear_model import LogisticRegression
model=LogisticRegression(max_iter=10000)

# train the model
model.fit(x_train,y_train)

# check overall model performance
model.score(x_test,y_test)

# predict some random outputs
y_predicted=model.predict(x_test)

# create confusion matrix to analyze the performance of the model
from sklearn.metrics import confusion_matrix
cm=confusion_matrix(y_test,y_predicted)
cm

# create heatmap
import seaborn as sn
plt.figure(figsize=(10,7))
sn.heatmap(cm,annot=True)
plt.xlabel('predicted')
plt.ylabel('true')
```

OUTPUT:

0.95277777

```
array([[31,  0,  0,  0,  0,  0,  0,  0,  0,  0],
       [ 0, 32,  0,  0,  0,  0,  0,  0,  0,  0],
       [ 0,  1, 36,  2,  0,  0,  0,  0,  0,  0],
       [ 0,  0,  0, 35,  0,  0,  0,  0,  1,  2],
       [ 0,  0,  0,  0, 33,  0,  0,  0,  0,  0],
       [ 0,  1,  0,  0,  0, 32,  1,  1,  0,  2],
       [ 0,  0,  0,  0,  0,  0, 35,  0,  0,  0],
       [ 0,  0,  0,  0,  1,  0,  0, 26,  0,  0],
       [ 0,  2,  1,  0,  0,  0,  0,  0, 37,  0],
       [ 0,  0,  0,  0,  0,  0,  0,  0,  2, 46]], dtype=int64)
```

Out[28]: Text(69.0, 0.5, 'true')



Program 8

Baye's and SVM Classifier

AIM: Write a Python code to implement Baye's and SVM Classifier for classification.

Description:

Naïve Bayes Classifier Algorithm

Naïve Bayes algorithm is a supervised learning algorithm, which is based on Bayes theorem and used for solving classification problems. It is a probabilistic classifier, which means it predicts on the basis of the probability of an object.

Why is it called Naïve Bayes?

The Naïve Bayes algorithm is comprised of two words Naïve and Bayes, Which can be described as:

Naïve: It is called Naïve because it assumes that the occurrence of a certain feature is independent of the occurrence of other features. Such as if the fruit is identified on the bases of color, shape, and taste, then red, spherical, and sweet fruit is recognized as an apple. Hence each feature individually contributes to identify that it is an apple without depending on each other.

Bayes: It is called Bayes because it depends on the principle of Bayes' Theorem.

Bayes' Theorem:

Bayes' theorem is also known as **Bayes' Rule** or **Bayes' law**, which is used to determine the probability of a hypothesis with prior knowledge. It depends on the conditional probability.

The formula for Bayes' theorem is given as:

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

Where,
P(A|B) is Posterior probability: Probability of hypothesis A on the observed event B.
P(B|A) is Likelihood probability: Probability of the evidence given that the probability of a hypothesis is true.
P(A) is Prior Probability: Probability of hypothesis before observing the evidence.
P(B) is Marginal Probability: Probability of Evidence.

8.1 Program to demonstrate implementation of naïve Bayes Theorem in python

Based on sex, age and fare find out the probability of survival of passenger in the titanic ship. Download the dataset from the given link

https://raw.githubusercontent.com/codebasics/py/master/ML/14_naive_bayes/titanic.csv

Source Code

```
#import all packages, read and visualize dataset
url=(r'https://raw.githubusercontent.com/codebasics/py/master/ML/14_naive_bayes
/titanic.csv')
import pandas as pd
titanicdf=pd.read_csv(url)
titanicdf.head()
df1=titanicdf
df1.head()
df1.drop(['PassengerId',",'],axis='columns',inplace=True)
df1.head()
target=df1.Survived
df1

# check for null values
df1.columns[df1.isna().any()]
df1.Age[:10]
df1.Age =df1.Age.fillna(df1.Age.mean())
df1.head()
df1.drop(['Sex'],axis='columns',inplace=True)
df1.head()

# split the dataset in to 80:20 training and testing ratio
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(df1,target,test_size=0.3)

# use guassiannb from sklearn library
from sklearn.naive_bayes import GaussianNB
model = GaussianNB()

#train the model
model.fit(X_train,y_train)
model.score(X_test,y_test)

#analyze the performance of the model
X_test[0:10]
y_test[0:10]
model.predict(X_test[0:10])
```

OUTPUT

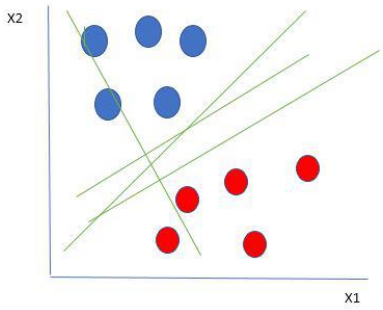
1, 0, 1, 0, 0, 0, 1, 0, 1, 0

SVM Classification algorithm

Support Vector Machine (SVM) is a supervised machine learning algorithm used for both classification and regression. The objective of SVM algorithm is to find a hyperplane in an N-dimensional space that distinctly classifies the data points. The

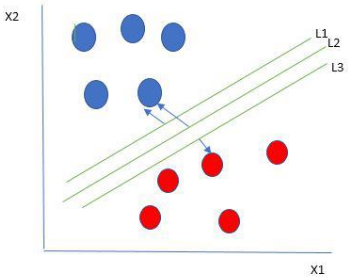
dimension of the hyperplane depends upon the number of features. If the number of input features is two, then the hyperplane is just a line. If the number of input features is three, then the hyperplane becomes a 2-D plane. It becomes difficult to imagine when the number of features exceeds three.

Let's consider two independent variables x1, x2 and one dependent variable which is either a blue circle or a red circle.



From the figure above its very clear that there are multiple lines (our hyperplane here is a line because we are considering only two input features x1, x2) that segregates our data points or does a classification between red and blue circles. So how do we choose the best line or in general the best hyperplane that segregates our data points.

Selecting the best hyper-plane:
One reasonable choice as the best hyperplane is the one that represents the largest separation or margin between the two classes.



So we choose the hyperplane whose distance from it to the nearest data point on each side is maximized. If such a hyperplane exists it is known as the maximum-margin hyperplane/hard margin. So from the above figure, we choose L2.

SVM Kernel:
The SVM kernel is a function that takes low dimensional input space and transforms it into higher-dimensional space, ie it converts not separable problem to separable problem. It is mostly useful in non-linear separation problems. Simply put the kernel, it does some extremely complex data transformations then finds out the process to separate the data based on the labels or outputs defined.

8.2 Program to demonstrate SVM for classification in python

Identify a particular digit from the digit dataset. Dataset is already present in sklearn library

Source Code

```
Import digits dataset from sklearn library
import pandas as pd
from sklearn.datasets import load_digits
digits = load_digits()

Store your dataset with the name df in the form of dataframe
df = pd.DataFrame(digits.data,digits.target)
##

#Print first 5 rows of your dataset
df.head()


#Keep your target variable aside with the variable name ‘target’
df['target'] = digits.target
df.head(20)


#Find out if there is any nan values in your dataframe
If yes thn replace that nan value with the mean value of the column

df.columns[df.isna().any()]
Index([], dtype='object')

#Split your dataset in to 70 : 30 ratio
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(df.drop('target',axis='columns'),
df.target, test_size=0.3)

# Import SVC from sklearn.svm
from sklearn.svm import SVC
rbf_model = SVC(kernel='rbf')

# train your model
rbf_model.fit(X_train, y_train)
SVC()

# check performance score of your model for rbf and linear kernel
rbf_model.score(X_test,y_test)
```

OUTPUT

```
0.9925925925925926
```

Program 9

Decision Tree classifier

AIM: Write a Python code to implement Decision tree classifier.

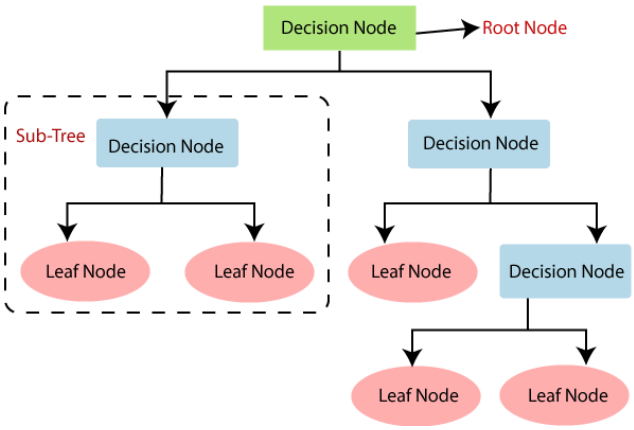
Description:

➢ Decision Tree is a Supervised learning technique that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome.

➢ In a Decision tree, there are two nodes, which are the Decision Node and Leaf Node. Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches.

➢ In order to build a tree, we use the CART algorithm, which stands for Classification and Regression Tree algorithm.

General structure of a decision tree:



Decision Tree Terminologies

Root Node: Root node is from where the decision tree starts. It represents the entire dataset, which further gets divided into two or more homogeneous sets.
Leaf Node: Leaf nodes are the final output node, and the tree cannot be segregated further after getting a leaf node.
Splitting: Splitting is the process of dividing the decision node/root node into sub-nodes according to the given conditions.
Branch/Sub Tree: A tree formed by splitting the tree.
Pruning: Pruning is the process of removing the unwanted branches from the tree.

Parent/Child node: The root node of the tree is called the parent node, and other nodes are called the child nodes.

Decision Tree algorithm

Step-1: Begin the tree with the root node, says S, which contains the complete dataset.
Step-2: Find the best attribute in the dataset using **Attribute Selection Measure (ASM)** which are Information Gain, Ginni Index and Entropy.
Step-3: Divide the S into subsets that contains possible values for the best attributes.
Step-4: Generate the decision tree node, which contains the best attribute.
Step-5: Recursively make new decision trees using the subsets of the dataset created in step -3. Continue this process until a stage is reached where you cannot further classify the nodes and called the final node as a leaf node.

9.1 Program to demonstrate Decision tree classification in python.

For the given dataset, based on company, job and degree predict whether the employee salary is more than 100k?
Download the dataset from the link given below
https://raw.githubusercontent.com/codebasics/py/master/ML/9_decision_tree/salaries.csv

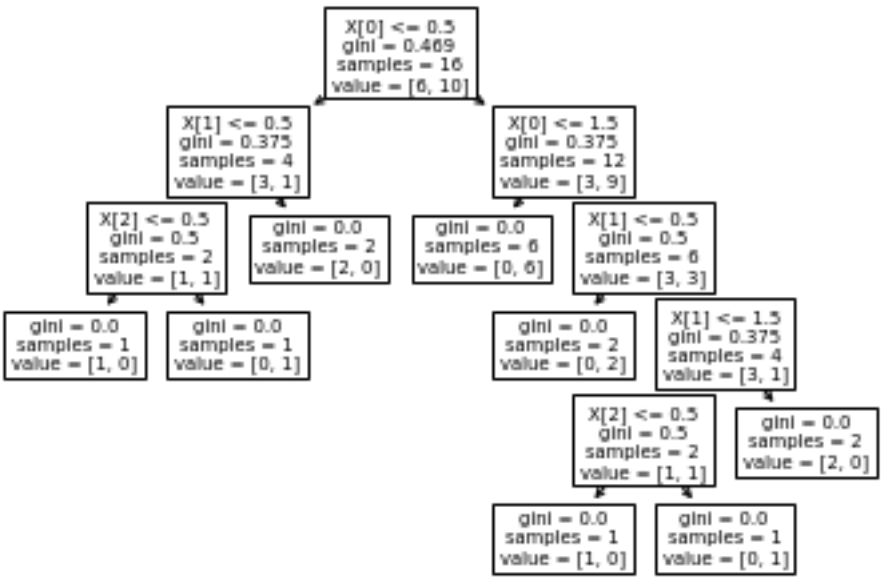
Source Code

```
# import package and read the dataset
import pandas as pd
url=(r'https://raw.githubusercontent.com/codebasics/py/master/ML/9_decision_tree/salaries.csv')
df=pd.read_csv(url)

#print first 5 rows of your dataset
df.head()

# drop salary_more_then_100k column
inputs = df.drop('salary_more_then_100k',axis='columns')

# keep it aside in a target varaible
target = df['salary_more_then_100k']

# label encoding
from sklearn.preprocessing import LabelEncoder
le_company = LabelEncoder()
le_job = LabelEncoder()
le_degree = LabelEncoder()
inputs['company_n'] = le_company.fit_transform(inputs['company'])
inputs['job_n'] = le_job.fit_transform(inputs['job'])
inputs['degree_n'] = le_degree.fit_transform(inputs['degree'])
inputs_n = inputs.drop(['company','job','degree'],axis='columns')
```

```
inputs_n

# implement decision tree
from sklearn import tree
model = tree.DecisionTreeClassifier()

# fit/ train the model
model.fit(inputs_n, target)

# analyze the performance of the model
model.score(inputs_n,target)

# print decision tree in graphical format
from sklearn import tree
tree.plot_tree(model)
```

OUTPUT

1.0



Program 10

K-Nearest Neighbour

AIM: Write a Python code to implement K-Nearest Neighbour classifier.

Description:

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. It is also called a lazy 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.

Algorithm

Step1: Select the number K of the neighbors
Step2: Calculate the Euclidean distance of K number of neighbors
Step3: Take the K nearest neighbors as per the calculated Euclidean distance.
Step4: Among these k neighbors, count the number of the data points in each category.
Step5: Assign the new data points to that category for which the number of the neighbor is maximum.
Step6: Our model is ready.

10.1 Program to demonstrate implementation of K-Nearest neighbour algorithm in python.

Build your very own k – Nearest Neighbor classifier to classify data from the IRIS dataset of scikit-learn.

Source Code

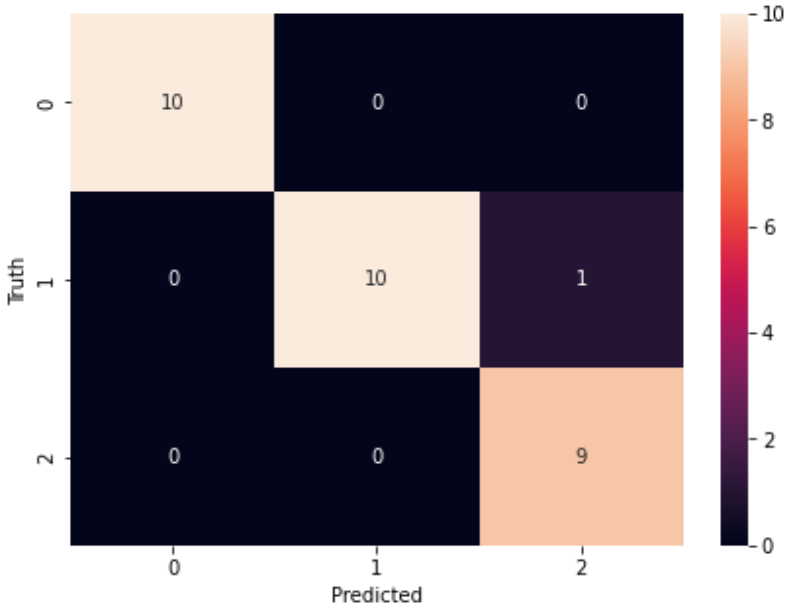
```
# import packages
import pandas as pd
from sklearn.datasets import load_iris
iris=load_iris()

# Analyze the dataset
dir(iris)
iris.frame
iris.feature_names

# convert dataset in to dataframe
df=pd.DataFrame(iris.data,columns=iris.feature_names)

#print row number 9 to 12 from the dataframe
df[9:12]
```

```
df['target']=iris.target
df
df[df.target==2].head()
df['flowername']=df.target.apply(lambda x: iris.target_names[x])
df
from sklearn.model_selection import train_test_split
x=df.drop(['target','flowername'],axis='columns')
y=df.target

# split dataset in to 80:20 ratio for training and testing
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2)
len(x_test)

# implement k -neighbour classifier
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=20)

# train the model
knn.fit(x_train, y_train)
KNeighborsClassifier(n_neighbors=20)

# Calculate score of the model
knn.score(x_test, y_test)

# some random prediction
knn.predict([[4.8,3.0,1.5,0.3]])

# create confusion matrix
from sklearn.metrics import confusion_matrix
y_pred = knn.predict(x_test)
cm = confusion_matrix(y_test, y_pred)
cm

# create heatmap
import matplotlib.pyplot as plt
import seaborn as sn
plt.figure(figsize=(7,5))
sn.heatmap(cm, annot=True)
plt.xlabel('Predicted')
plt.ylabel('Truth')

#print classification report
from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred))
```

OUTPUT

```
0.9666666666666667
array([0])
```

array([[10, 0, 0],
       [0, 10, 1],
       [0, 0, 9]], dtype=int64)



[

	precision	recall	f1-score	support
0	1.00	1.00	1.00	10
1	1.00	0.91	0.95	11
2	0.90	1.00	0.95	9
accuracy			0.97	30
macro avg	0.97	0.97	0.97	30
weighted avg	0.97	0.97	0.97	30

Program 11

K-Means Algorithm

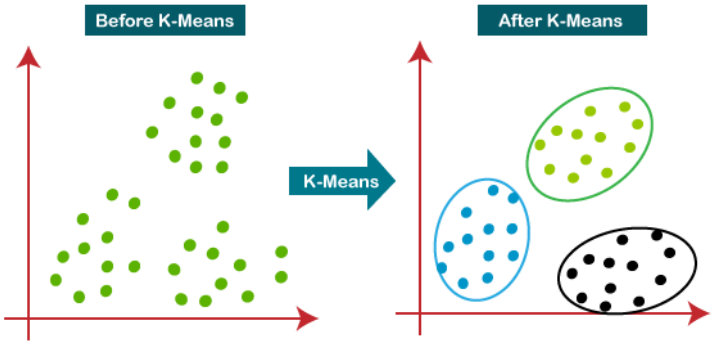
AIM: Write a Python code to implement Decision tree classifier.

Description:

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 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.

Algorithm

Step 1 Select the number K to decide the number of clusters.
Step 2 Select random K points or centroids. (It can be other 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 datapoint 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.



Elbow Method
The Elbow method is one of the most popular ways to find the optimal number of clusters. This method uses the concept of WCSS value. WCSS stands for Within Cluster Sum of Squares, which defines the total variations within a cluster. The formula to calculate the value of WCSS (for 3 clusters) is given below:

WCSS= ∑Pi in Cluster1 distance(Pi C1)2 +∑Pi in Cluster2distance(Pi C2)2+∑Pi in CLuster3 distance(Pi C3)2

In the above formula of WCSS,

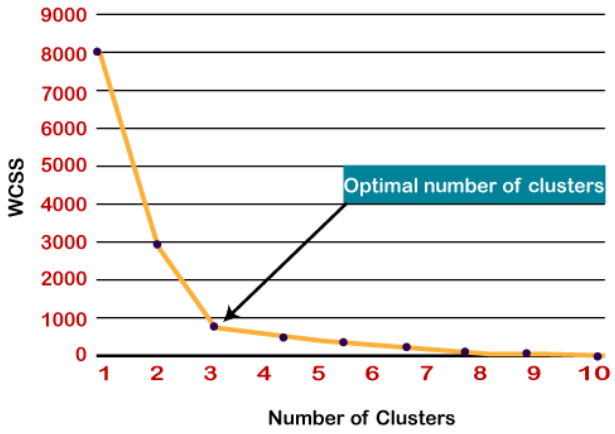
∑Pi in Cluster1 distance(Pi C1)2: It is the sum of the square of the distances between each data point and its centroid within a cluster1 and the same for the other two terms.

To measure the distance between data points and centroid, we can use any method such as Euclidean distance or Manhattan distance.

Steps to Implement Elbow method
To find the optimal value of clusters, the elbow method follows the below steps:

1. It executes the K-means clustering on a given dataset for different K values (ranges from 1-10).
2. For each value of K, calculates the WCSS value.
3. Plots a curve between calculated WCSS values and the number of clusters K.
4. The sharp point of bend or a point of the plot looks like an arm, then that point is considered as the best value of K.
5. Since the graph shows the sharp bend, which looks like an elbow, hence it is known as the elbow method.

The graph for the elbow method looks like the below image:



11.1 Program to demonstrate K-Means clustering algorithm in python.

Clustering using K-means algorithm for the given dataset income. Dataset can be downloaded from the link given below
https://github.com/codebasics/py/blob/master/ML/13_kmeans/income.csv

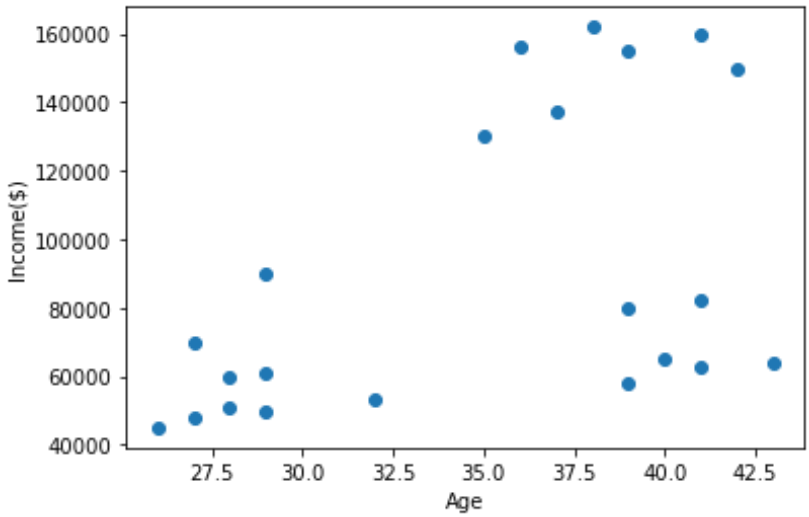
```
# import all the required packages
from sklearn.cluster import KMeans
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
from matplotlib import pyplot as plt
%matplotlib inline

#read dataset
```

```
url=("https://raw.githubusercontent.com/codebasics/py/master/ML/13_kmeans/income.
csv")

# store dataset in the form of dataframe
df = pd.read_csv(url)

#print first 5 rows of your dataset
df.head()

# visualize the dataset using scatter plot
plt.scatter(df.Age,df['Income($)'])
plt.xlabel('Age')
plt.ylabel('Income($)')

# create clusters for random value of k=3
km = KMeans(n_clusters=3)
y_predicted = km.fit_predict(df[['Age','Income($)']])
y_predicted
df['cluster']=y_predicted
df.head()

# analyze cluster centers
km.cluster_centers_

# draw clusters with cluster center
df1 = df[df.cluster==0]
df2 = df[df.cluster==1]
df3 = df[df.cluster==2]
plt.scatter(df1.Age,df1['Income($)'],color='green')
plt.scatter(df2.Age,df2['Income($)'],color='red')
plt.scatter(df3.Age,df3['Income($)'],color='black')
plt.scatter(km.cluster_centers_[:,0],km.cluster_centers_[:,1],color='purple',marker='*',la
bel='centroid')
plt.xlabel('Age')
plt.ylabel('Income ($)')
plt.legend()

# data normalization and visualization
scaler = MinMaxScaler()
scaler.fit(df[['Income($)']])
df['Income($)'] = scaler.transform(df[['Income($)']])
scaler.fit(df[['Age']])
df['Age'] = scaler.transform(df[['Age']])
df.head()
plt.scatter(df.Age,df['Income($)'])

km = KMeans(n_clusters=3)
y_predicted = km.fit_predict(df[['Age','Income($)']])
y_predicted
km.cluster_centers_
```

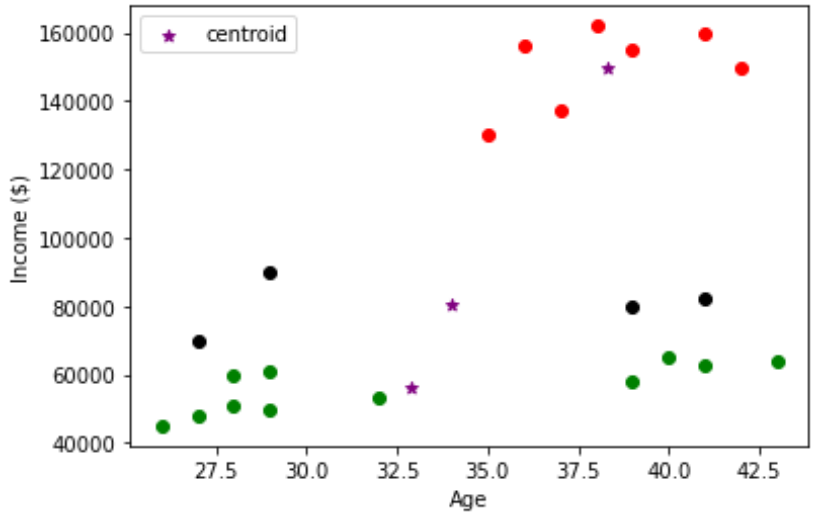
```
df['cluster']=y_predicted
df.head()
df1 = df[df.cluster==0]
df2 = df[df.cluster==1]
df3 = df[df.cluster==2]
plt.scatter(df1.Age,df1['Income($)'],color='green')
plt.scatter(df2.Age,df2['Income($)'],color='red')
plt.scatter(df3.Age,df3['Income($)'],color='black')
plt.scatter(km.cluster_centers_[:,0],km.cluster_centers_[:,1],color='purple',marker='*',la
bel='centroid')
plt.legend()

# calculate value of K using elbow method
sse = []
k_rng = range(1,10)
for k in k_rng:
    km = KMeans(n_clusters=k)
    km.fit(df[['Age','Income($)']])
    sse.append(km.inertia_)
plt.xlabel('K')
plt.ylabel('Sum of squared error')
plt.plot(k_rng,sse)
```

OUTPUT

```
<matplotlib.legend.Legend at 0x1aca98e40d0>
Text(0, 0.5, 'Income($)')
```



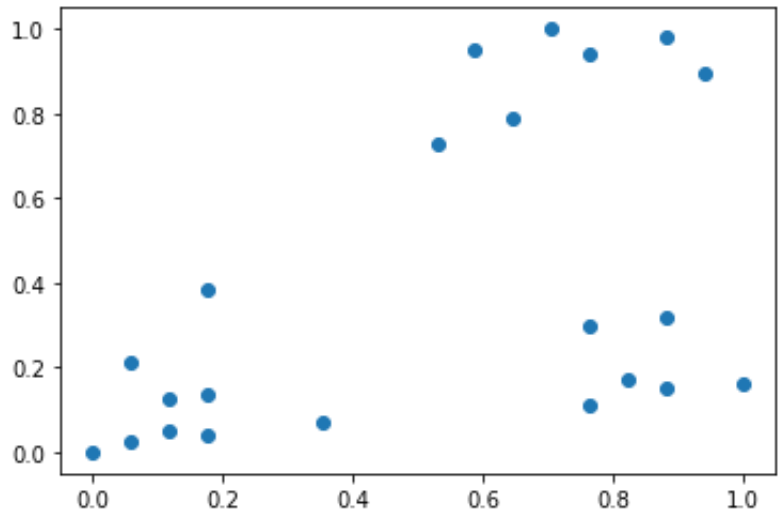

```
array([2, 2, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 2, 2, 0])

array([[3.29090909e+01, 5.61363636e+04],
       [3.82857143e+01, 1.50000000e+05],
       [3.40000000e+01, 8.05000000e+04]])
```



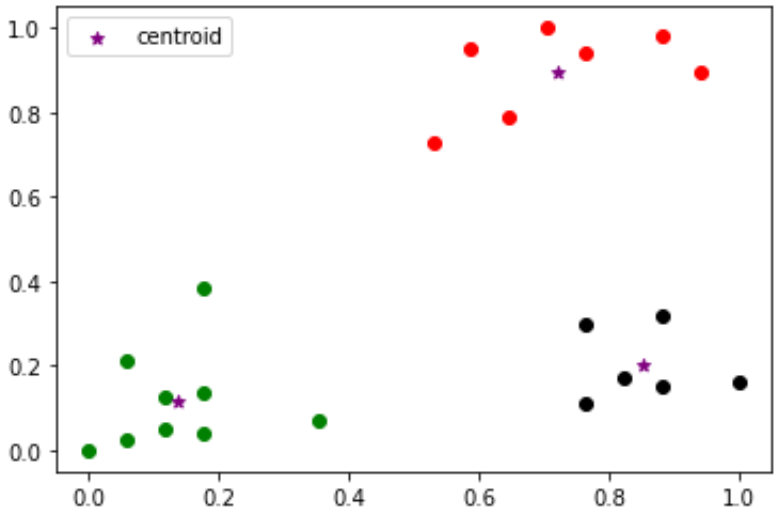
array([0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 2, 2, 2, 2, 2, 2])

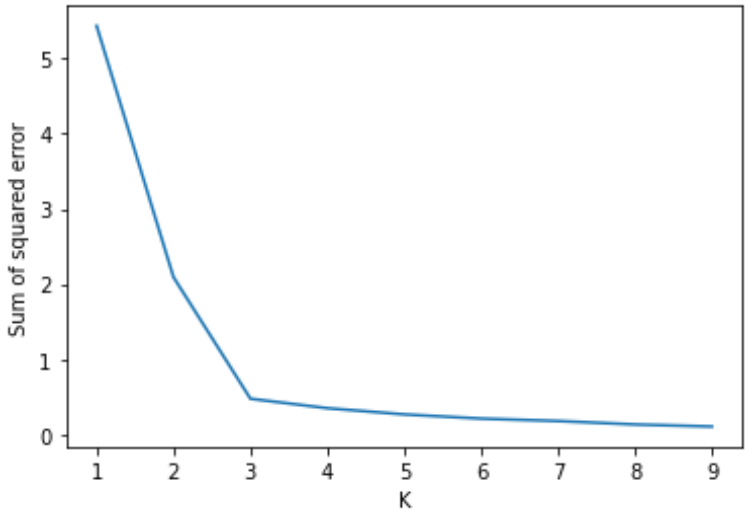
array([[0.1372549 , 0.11633428],
[0.72268908, 0.8974359],
[0.85294118, 0.2022792]])

<matplotlib.collections.PathCollection at 0x1aca997a520>



<matplotlib.legend.Legend at 0x1aca99e0ee0>


centroid
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0


Sum of squared error
5
4
3
2
1
0
1
2
3
4
5
6
7
8
9
K

Program 12

Principal Component Analysis for dimensionality reduction

AIM: Write a Python code to implement Principal component analysis for dimensionality reduction.

Description:

Principal Component Analysis is an unsupervised learning algorithm that is used for the dimensionality reduction in machine learning. It is a statistical process that converts the observations of correlated features into a set of linearly uncorrelated features with the help of orthogonal transformation. These new transformed features are called the Principal Components. PCA generally tries to find the lower-dimensional surface to project the high-dimensional data. Some common terms used in PCA algorithm:

Dimensionality: It is the number of features or variables present in the given dataset. More easily, it is the number of columns present in the dataset.

Correlation: It signifies that how strongly two variables are related to each other. Such as if one changes, the other variable also gets changed. The correlation value ranges from -1 to +1. Here, -1 occurs if variables are inversely proportional to each other, and +1 indicates that variables are directly proportional to each other.

Orthogonal: It defines that variables are not correlated to each other, and hence the correlation between the pair of variables is zero.

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

Covariance Matrix: A matrix containing the covariance between the pair of variables is called the Covariance Matrix.

Steps for PCA algorithm

1. Getting the dataset

Firstly, we need to take the input dataset and divide it into two subparts X and Y, where X is the training set, and Y is the validation set.

2. Representing data into a structure

Now we will represent our dataset into a structure. Such as we will represent the two-dimensional matrix of independent variable X. Here each row corresponds to the data items, and the column corresponds to the Features. The number of columns is the dimensions of the dataset.

3. Standardizing the data

In this step, we will standardize our dataset. Such as in a particular column, the features with high variance are more important compared to the features with lower variance.

If the importance of features is independent of the variance of the feature, then we will divide each data item in a column with the standard deviation of the column. Here we will name the matrix as Z.

4. Calculating the Covariance of Z

To calculate the covariance of Z, we will take the matrix Z, and will transpose it. After transpose, we will multiply it by Z. The output matrix will be the Covariance matrix of Z.

5. Calculating the Eigen Values and Eigen Vectors

Now we need to calculate the eigenvalues and eigenvectors for the resultant covariance matrix Z. Eigenvectors or the covariance matrix are the directions of the axes with high information. And the coefficients of these eigenvectors are defined as the eigenvalues.

6. Sorting the Eigen Vectors

In this step, we will take all the eigenvalues and will sort them in decreasing order, which means from largest to smallest. And simultaneously sort the eigenvectors accordingly in matrix P of eigenvalues. The resultant matrix will be named as P*.

7. Calculating the new features Or Principal Components

Here we will calculate the new features. To do this, we will multiply the P* matrix to the Z. In the resultant matrix Z*, each observation is the linear combination of original features. Each column of the Z* matrix is independent of each other.

8. Remove less or unimportant features from the new dataset.

The new feature set has occurred, so we will decide here what to keep and what to remove. It means, we will only keep the relevant or important features in the new dataset, and unimportant features will be removed out.

12.1 Program to demonstrate principal component analysis in python

Demonstrate the working of PCA for digit classification using digit dataset which is already stored in sklearn library

SOURCE CODE

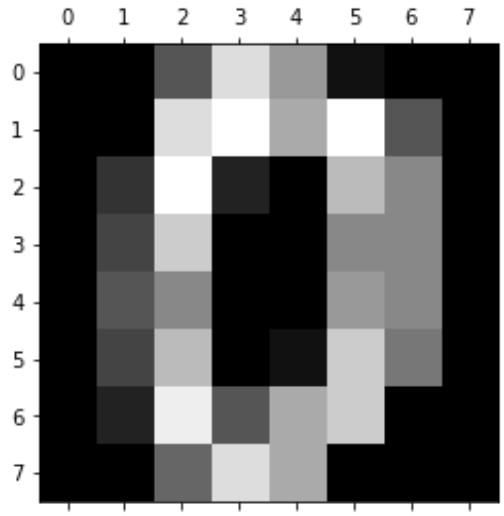
```
# import required libraries
from sklearn.datasets import load_digits
import pandas as pd
dataset = load_digits()

# use to check what the dataset is all about
dataset.keys()

#to shack shape of data [it contain 1797 sample and each sample contain 64 features]
dataset.data.shape


# first element of dataset which is in the form of 1 d array
dataset.data[0]


# for visualization using matplot lib convert 1 d array to 2 d array
dataset.data[0].reshape(8,8)


# data visualization
from matplotlib import pyplot as plt
%matplotlib inline
plt.gray()
plt.matshow(dataset.data[0].reshape(8,8))

# check the dataset target
dataset.target[:5]

# create dataframe
df = pd.DataFrame(dataset.data, columns=dataset.feature_names)
df.head()
X = df
y = dataset.target


# scaling the value using standardscaler [range is -
1 to 1]. you can use minmax scaler also
```

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
X_scaled


# splitting the dataset and use logistic regression
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_sta
te=30)
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
model.fit(X_train, y_train)
model.score(X_test, y_test)


# apply pca for retaining 95 percent useful features
from sklearn.decomposition import PCA
pca = PCA(0.95)
X_pca = pca.fit_transform(X)
X_pca.shape


# split new dataframe and train the model
X_train_pca, X_test_pca, y_train, y_test = train_test_split(X_pca, y, test_size=0.2, rand
om_state=30)
from sklearn.linear_model import LogisticRegression
model = LogisticRegression(max_iter=1000)
model.fit(X_train_pca, y_train)
model.score(X_test_pca, y_test)


#from the above scenario we can see that despite dropping lots of features we obtained
near by similar accuracy
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X)
X_pca.shape

X_train_pca, X_test_pca, y_train, y_test = train_test_split(X_pca, y, test_size=0.2, rand
om_state=30)
model = LogisticRegression(max_iter=1000)
model.fit(X_train_pca, y_train)
model.score(X_test_pca, y_test)
```

OUTPUT

```
dict_keys(['data', 'target', 'frame', 'feature_names', 'target_names', 'images', 'DESCR'])

(1797, 64)
```

```
array([ 0.,  0.,  5., 13.,  9.,  1.,  0.,  0.,  0.,  0., 13., 15., 10.,
       15.,  5.,  0.,  0.,  3., 15.,  2.,  0., 11.,  8.,  0.,  0.,  4.,
       12.,  0.,  0.,  8.,  8.,  0.,  0.,  5.,  8.,  0.,  0.,  9.,  8.,
        0.,  0.,  4., 11.,  0.,  1., 12.,  7.,  0.,  0.,  2., 14.,  5.,
       10., 12.,  0.,  0.,  0.,  0.,  6., 13., 10.,  0.,  0.,  0.])

array([[ 0.,  0.,  5., 13.,  9.,  1.,  0.,  0.],
       [ 0.,  0., 13., 15., 10., 15.,  5.,  0.],
       [ 0.,  3., 15.,  2.,  0., 11.,  8.,  0.],
       [ 0.,  4., 12.,  0.,  0.,  8.,  8.,  0.],
       [ 0.,  5.,  8.,  0.,  0.,  9.,  8.,  0.],
       [ 0.,  4., 11.,  0.,  1., 12.,  7.,  0.],
       [ 0.,  2., 14.,  5., 10., 12.,  0.,  0.],
       [ 0.,  0.,  6., 13., 10.,  0.,  0.,  0.]])

<matplotlib.image.AxesImage at 0x1ae666ed4c0>
<Figure size 432x288 with 0 Axes>
```




```
array([0, 1, 2, 3, 4])

array([[ 0.        , -0.33501649, -0.04308102, ..., -1.14664746,
        -0.5056698 ,          , -0.19600752],
       [ 0.        , -0.33501649, -1.09493684, ...,  0.54856067,
        -0.5056698 ,          , -0.19600752],
       [ 0.        , -0.33501649, -1.09493684, ...,  1.56568555,
         1.6951369 ,          , -0.19600752],
       ...,
       [ 0.        , -0.33501649, -0.88456568, ..., -0.12952258,
        -0.5056698 ,          , -0.19600752],
       [ 0.        , -0.33501649, -0.67419451, ...,  0.8876023 ,
        -0.5056698 ,          , -0.19600752],
       [ 0.        , -0.33501649,  1.00877481, ...,  0.8876023 ,
        -0.26113572, -0.19600752]])
```

0.9722222222222222

(1797, 29)

0.9694444444444444

(1797, 2)

0.6083333333333333

#We get less accuancy (~60%) as using only 2 components did not retain much of the fe
ature information.
#However in real life you will find many cases where using 2 or few PCA components
can still give you a pretty good accuracy

Program 13

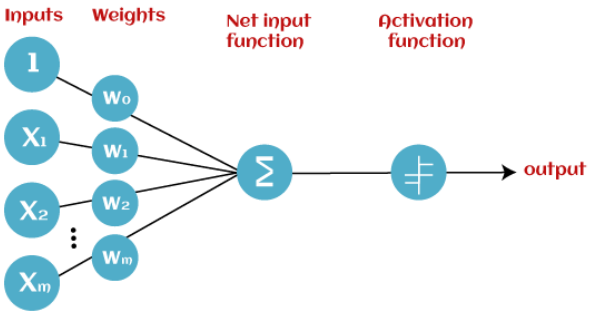
Perceptron and Backpropagation algorithm

AIM: Write a Python code to implement Perceptron and Backpropagation algorithm.

Description:

PERCEPTRON

Perceptron is Machine Learning algorithm for supervised learning of various binary classification tasks. Further, Perceptron is also understood as an Artificial Neuron or neural network unit that helps to detect certain input data computations in business intelligence. Hence, we can consider it as a single-layer neural network with four main parameters, i.e., input values, weights and Bias, net sum, and an activation function.



Input Nodes or Input Layer:
This is the primary component of Perceptron which accepts the initial data into the system for further processing. Each input node contains a real numerical value.

Wight and Bias:
Weight parameter represents the strength of the connection between units. This is another most important parameter of Perceptron components. Weight is directly proportional to the strength of the associated input neuron in deciding the output. Further, Bias can be considered as the line of intercept in a linear equation.

Activation Function:
These are the final and important components that help to determine whether the neuron will fire or not. Activation Function can be considered primarily as a step function. Types of Activation functions:

- Sign function
- Step function, and
- Sigmoid function

Perceptron model works in two important steps as follows:

Step 1 In the first step first, multiply all input values with corresponding weight values and then add them to determine the weighted sum. Mathematically, we can calculate the weighted sum as follows:

$$\sum wi*xi = x1*w1 + x2*w2 + \ldots wn*xn$$

Add a special term called bias 'b' to this weighted sum to improve the model's performance.

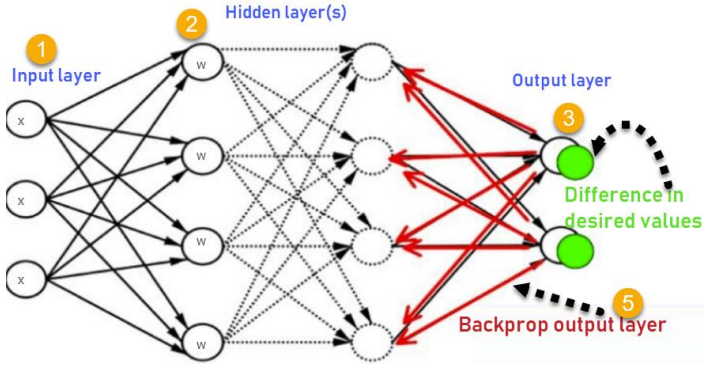
$$\sum wi*xi + b$$

Step 2 In the second step, an activation function is applied with the above-mentioned weighted sum, which gives us output either in binary form or a continuous value as follows:

$$Y = f(\sum wi*xi + b)$$

BACK PROPAGATION ALGORITHM

The Back propagation algorithm in neural network computes the gradient of the loss function for a single weight by the chain rule. It efficiently computes one layer at a time, unlike a native direct computation. It computes the gradient, but it does not define how the gradient is used. It generalizes the computation in the delta rule. Consider the following Back propagation neural network example diagram to understand:



Algorithm

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
   Error= Actual Output – Desired Output
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

13.1 Program to demonstrate back propagation algorithm in python.

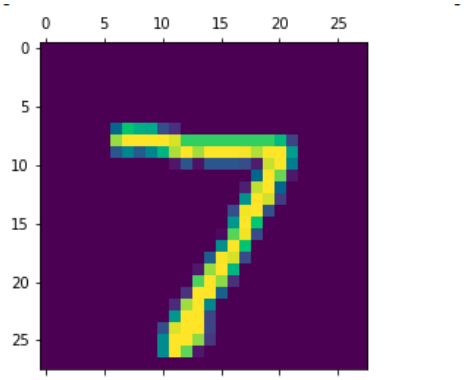
```
#Handwritten digits classification using neural network
#In this notebook we will classify handwritten digits using a simple
#neural network which has only input and output layers.
#We will than add a hidden layer and see how the performance of the model improve


# import all the required framework and libraries
import tensorflow as tf
from tensorflow import keras
import matplotlib.pyplot as plt
%matplotlib inline
import numpy as np

# load mnist dataset
(X_train, y_train) , (X_test, y_test) = keras.datasets.mnist.load_data()

#normalize the images
X_train = X_train / 255
X_test = X_test / 255

#reshape and flattened image array
X_train_flattened = X_train.reshape(len(X_train), 28*28)
X_test_flattened = X_test.reshape(len(X_test), 28*28)
X_train_flattened.shape

# create a neural network with one input and one output layer
model = keras.Sequential([
   keras.layers.Dense(10, input_shape=(784,), activation='sigmoid')
])

model.compile(optimizer='adam',
          loss='sparse_categorical_crossentropy',
          metrics=['accuracy'])

# train the neural network
model.fit(X_train_flattened, y_train, epochs=5)

#evaluate the performance of th neural network
model.evaluate(X_test_flattened, y_test)

# some predictions
y_predicted = model.predict(X_test_flattened)
y_predicted[0]

# display input image at particular index
plt.matshow(X_test[0])

#np.argmax finds a maximum element from an array and returns the index of it
np.argmax(y_predicted[0])
```

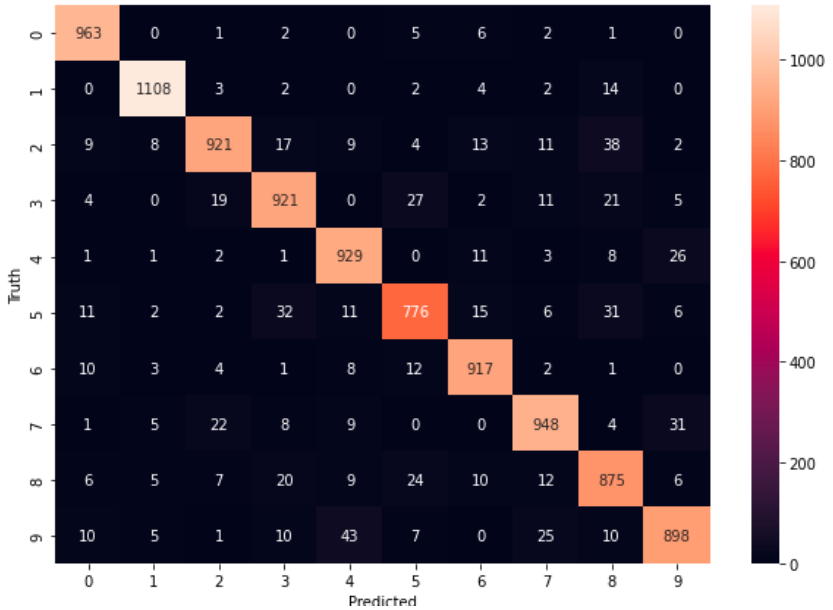
```
#predictions
y_predicted_labels = [np.argmax(i) for i in y_predicted]
y_predicted_labels[:5]

# create confusion matrix
cm = tf.math.confusion_matrix(labels=y_test,predictions=y_predicted_labels)
cm

#plot confusion matrix
import seaborn as sn
plt.figure(figsize = (10,7))
sn.heatmap(cm, annot=True, fmt='d')
plt.xlabel('Predicted')
plt.ylabel('Truth')

#add hidden layer in the neural network
Using hidden layer
model = keras.Sequential([
   keras.layers.Dense(100, input_shape=(784,), activation='relu'),
   keras.layers.Dense(10, activation='sigmoid')
])

model.compile(optimizer='adam',
          loss='sparse_categorical_crossentropy',
          metrics=['accuracy'])

model.fit(X_train_flattened, y_train, epochs=5)

#evlaute the performance of the new neural network with hidden layers
model.evaluate(X_test_flattened,y_test)

#predictions
y_predicted = model.predict(X_test_flattened)
y_predicted_labels = [np.argmax(i) for i in y_predicted]
#create confusion matrix
cm = tf.math.confusion_matrix(labels=y_test,predictions=y_predicted_labels)
#plot confusion matrix
plt.figure(figsize = (10,7))
sn.heatmap(cm, annot=True, fmt='d')
plt.xlabel('Predicted')
plt.ylabel('Truth')

#Using Flatten layer so that we don't have to call .reshape on input dataset
model = keras.Sequential([
   keras.layers.Flatten(input_shape=(28, 28)),
   keras.layers.Dense(100, activation='relu'),
   keras.layers.Dense(10, activation='sigmoid')
])

model.compile(optimizer='adam',
          loss='sparse_categorical_crossentropy',
          metrics=['accuracy'])
```

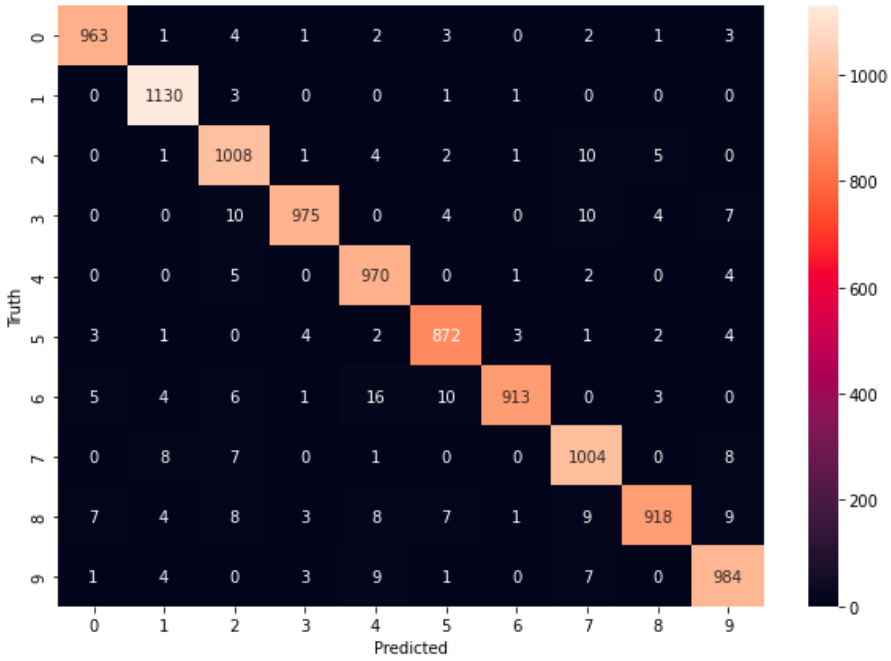
```
#train the model
model.fit(X_train, y_train, epochs=10)

# evalute the performance of the model
model.evaluate(X_test,y_test)
```

OUTPUT

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11493376/11490434 [==============================] - 0s 0us/step
11501568/11490434 [==============================] - 0s 0us/step
Epoch 1/5
1875/1875 [==============================] - 4s 2ms/step - loss: 0.4708 - accuracy: 0.8769
Epoch 2/5
1875/1875 [==============================] - 3s 1ms/step - loss: 0.3038 - accuracy: 0.9165
Epoch 3/5
1875/1875 [==============================] - 3s 1ms/step - loss: 0.2833 - accuracy: 0.9206
Epoch 4/5
1875/1875 [==============================] - 3s 1ms/step - loss: 0.2731 - accuracy: 0.9242
Epoch 5/5
1875/1875 [==============================] - 3s 1ms/step - loss: 0.2663 - accuracy: 0.9258
313/313 [==============================] - 0s 1ms/step - loss: 0.2694 - accuracy: 0.9256
Epoch 1/5
1875/1875 [==============================] - 4s 2ms/step - loss: 0.2772 - accuracy: 0.9211
Epoch 2/5
1875/1875 [==============================] - 3s 2ms/step - loss: 0.1252 - accuracy: 0.9632
Epoch 3/5
1875/1875 [==============================] - 3s 2ms/step - loss: 0.0870 - accuracy: 0.9739
Epoch 4/5
1875/1875 [==============================] - 3s 2ms/step - loss: 0.0671 - accuracy: 0.9795
Epoch 5/5
1875/1875 [==============================] - 3s 2ms/step - loss: 0.0538 - accuracy: 0.9831
313/313 [==============================] - 0s 1ms/step - loss: 0.0837 - accuracy: 0.9737
Epoch 1/10
1875/1875 [==============================] - 4s 2ms/step - loss: 0.2692 - accuracy: 0.9245
Epoch 2/10
1875/1875 [==============================] - 3s 2ms/step - loss: 0.1257 - accuracy: 0.9635
Epoch 3/10
1875/1875 [==============================] - 3s 2ms/step - loss: 0.0873 - accuracy: 0.9740
Epoch 4/10
1875/1875 [==============================] - 3s 2ms/step - loss: 0.0661 - accuracy: 0.9803
Epoch 5/10
1875/1875 [==============================] - 3s 2ms/step - loss: 0.0521 - accuracy: 0.9842
Epoch 6/10
1875/1875 [==============================] - 3s 2ms/step - loss: 0.0428 - accuracy: 0.9872
Epoch 7/10
1875/1875 [==============================] - 3s 2ms/step - loss: 0.0341 - accuracy: 0.9894
Epoch 8/10
1875/1875 [==============================] - 3s 2ms/step - loss: 0.0288 - accuracy: 0.9916
Epoch 9/10
1875/1875 [==============================] - 3s 2ms/step - loss: 0.0229 - accuracy: 0.9929
Epoch 10/10
1875/1875 [==============================] - 3s 2ms/step - loss: 0.0202 - accuracy: 0.9939
313/313 [==============================] - 0s 1ms/step - loss: 0.0815 - accuracy: 0.9779
[0.08149928599596024, 0.9779000282287598]
```


0 5 10 15 20 25
0 5 10 15 20 25


963 0 1 2 0 5 6 2 1 0
0 1108 3 2 0 2 4 2 14 0
9 8 921 17 9 4 13 11 38 2
4 0 19 921 0 27 2 11 21 5
1 1 2 1 929 0 11 3 8 26
11 2 2 32 11 776 15 6 31 6
10 3 4 1 8 12 917 2 1 0
1 5 22 8 9 0 0 948 4 31
6 5 7 20 9 24 10 12 875 6
10 5 1 10 43 7 0 25 10 898
Truth
0 1 2 3 4 5 6 7 8 9
Predicted
1000
800
600
400
200
0


963 1 4 1 2 3 0 2 1 3
0 1130 3 0 0 1 1 0 0 0
0 1 1008 1 4 2 1 10 5 0
0 0 10 975 0 4 0 10 4 7
0 0 5 0 970 0 1 2 0 4
3 1 0 4 2 872 3 1 2 4
5 4 6 1 16 10 913 0 3 0
0 8 7 0 1 0 0 1004 0 8
7 4 8 3 8 7 1 9 918 9
1 4 0 3 9 1 0 7 0 984
Truth
0 1 2 3 4 5 6 7 8 9
Predicted
1000
800
600
400
200
0

Program 14

MINI PROJECT

AIM: A mini, miniature project completed by the student in a short course of time.

Description: The student is supposed to learn the basic technical concepts, and the administration encompassing the project execution. Advantage of doing mini project is that Student will learn good time management as well as he/she will get enhanced Planning, organizing and communication skills.

List of Sample mini-Projects which can be take up into this lab are:

1. **Cartoonify Image with Machine Learning:** Transform images into its cartoon. Yes, the objective of this machine learning project is to CARTOONIFY the images. Thus, you will build a python application that will transform an image into its cartoon using machine learning libraries.
2. **Emojify – Create your own emoji:** The objective of this machine learning project is to classify human facial expressions and map them to emojis. You will build a convolution neural network to recognize facial emotions. Then you will map those emotions with the corresponding emojis or avatars
3. **Fake News Detection Project:** Fake news spreads like a wildfire and this is a big issue in this era. You can learn how to distinguish fake news from a real one. You can use supervised learning to implement a model like this.

   Dataset link:
   https://drive.google.com/file/d/1er9NJTLUA3qnRuyhfzuN0XUsoIC4a-_q/view

4. **Personality Prediction Project:** The Myers Briggs Type Indicator is a personality type system that divides a person into 16 distinct personalities based on introversion, intuition, thinking and perceiving capabilities. You can identify the personality of a person from the type of posts they put on social media.

   Dataset: https://www.kaggle.com/datasets/datasnaek/mbti-type

5. **Sign Language Recognition with Machine Learning:** A lot of research has been done to help people who are deaf and dumb. In this sign language recognition project, you create a sign detector that detects sign language. This can be very helpful for the deaf and dumb people in communicating with others.