LAB MANUAL

On

ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING LAB

COURSE CODE (22CDPC55)

(III- B. Tech. – I– Semester)

Submitted to

DEPARTMENT OF COMPUTER SCIENCE& ENGINEERING (DATA SCIENCE)

By

Mrs INDUMATHI V
(Asst. Professor, Dept. of AI & DS)

CMR INSTITUTE OF TECHNOLOGY
Kandlakoya(V), Medchal Road, Hyderabad – 501 401
Ph. No. 08418-222042, 22106 Fax No. 08418-222106

(2024-25)

CONTENTS

CMR INSTITUTE OF TECHNOLOGY

VISION: To create world class technocrats for societal needs

MISSION: Impart global quality technical education assessing learning environment through

- Innovative Research & Development
- Eco system for better Industry institute interaction
- Capacity building among stakeholders

QUALITY POLICY: Strive for global Professional excellence in pursuit of key – Stake holders

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING:

COMPUTER SCIENCE AND ENGINEERING (DATA SCIENCE)

Vision: Develop competent software professionals, researchers' and entrepreneurs to serve global society.

Mission: The department of **Computer Science and Engineering (Data Science)** is committed to

- Create technocrats with proficiency in design and code for software development
- Adapt contemporary technologies by lifelong learning and face challenges in IT and ITES sectors
- Quench the thirst of knowledge in higher education, employment, R&D and entrepreneurship

I. PROGRAMME EDUCATIONAL OBJECTIVES (PEO's)

PEO1: Graduate will be capable of practicing principles of computer science & engineering, mathematics and scientific investigation to solve the problems that are appropriate to the discipline. [PO's:1,2,3,4,5,7,8,9,10,11and12] [PSO's:1and2]

PEO2:Graduate will profess in Data Science applications that lead to professional, career and research advancement. [PO's: 1,2,3,4,5,6,7,8,9,10 and12][PSO's:1, 2and3]

PEO3: Graduate exhibits professional ethics, communication skills,teamworkandadaptstochangingenvironmentsofengineering and technology by engaging in lifelong learning.[PO's:1,2,3,4,5,6,7,8,9,10,11and12][PSO's:2and3]

II. PROGRAMME OUTCOMES (PO's)

1. **Engineering knowledge**: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems. **[PEO's: 1,2 and 3]**

2. **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. **[PEO's: 1,2 and 3]**

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**. [PEO's: 1,2 and 3]**

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. **[PEO's: 1,2 and 3]**

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. **[PEO's: 1,2 and 3]**

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. **[PEO's: 2 and 3]**

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. **[PEO's: 1,2 and 3]**

8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice. **[PEO's: 1,2 and 3]**

9. **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings. **[PEO's: 1,2 and 3]**

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. **[PEO's: 1,2 and 3]**

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. **[PEO's: 1 and 3]**

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. **[PEO's: 1,2 and 3]**

ARTIFICIAL INTELLIGENCE AND MACHINELEARNINGLAB

III-B.Tech.-I-Sem. **L TPC**
SubjectCode: 22CDPC55 **- - 2 1**

CourseOutcomes:
Uponcompletion ofthecourse,thestudentswillbeableto:

1. Illustratevarious search technique.
2. Solvereal-timeproblemsusinggraphtheory.
3. Usetechniquesofknowledgerepresentationandprobabilisticreasoning.
4. Designvarioussupervisedlearningalgorithms.
5. Implementvariousunsupervisedlearningalgorithms.

LISTOFEXPERIMENTS

1. Write a program to implement BFS and DFS Traversal.
2. Write a program to implement A* Search.
3. Write a program to implement Travelling Salesman Problem and Graph Coloring Problem.
4. Write a program to implement Knowledge Representation.
5. Write a program to implement Bayesian Network.
6. Write a program to implement Hidden Markov Model.
7. Write a program to implement Regression algorithm
8. Write a program to implement decision tree based ID3 algorithm.
9. Write a program to implement K-Means Clustering algorithm.
10. Write a program to implement K-Nearest Neighbor algorithm (K-NN).
11. Write a program to implement Back Propagation Algorithm.
12. Write a program to implement Support Vector Machine.

Micro-Projects: Student must submit are port onone of the following Micro–Projects before commencement of second internal examination.

1. ArtificialIntelligencefor RecordsManagement.
2. Efficient,ScalableProcessingofPatientDatausingArtificialIntelligence.
3. Smart BikeShareProgramsusingArtificialIntelligence.
4. AutomaticDocumentClassificationusingBayesiantheorem.
5. ArtificialIntelligenceine-Commerce.
6. DiagnosecropdiseasewithMachineLearning.
7. Developasystemtoanalyzebuyingbehaviorofacustomer.
8. Develop asystemtostudysentimentofusers ontwitter.
9. Develop apredictivemodeltostudytheemployeesatisfactioninanorganization.
10. Developapredictivemodeltostudytherainfallofyoursociety.

Reference:

1.ArtificialIntelligenceandMachineLearningLab Manual,Dept.ofCSE(DS),CMRIT,Hyd.

2. STUDENT ENTRY BEHAVIOR OR PRE-REQUISITES

- Students should have basic knowledge on Data Structures and Data Mining
- Students should have basic knowledge on Design and Analysis of Algorithms.
- Student should have knowledge on Python.

These prerequisites are taken by the students during the first two years. However during the initial sessions the topics are reviewed.

3. COURSE OBJECTIVES

Course Objectives	Course Objective Statements
Objective - 1	Explain the concepts of artificial intelligence and Machine Learning
Objective – 2	Adapt various probabilistic reasoning approaches
Objective – 3	Illustrate various search algorithms , Classification and Clustering techniques
Objective – 4	Elaborate Hidden Markov Model in AI and Back Propagation Algorithm in ML
Objective – 5	Perceive various reinforcement learning approaches

4. COURSE OUTCOMES

COs	Uponcompletionofcoursethestudents willbeableto	PO4	PO5	PO9	PSO2
CO1	illustratevarioussearchtechniques	3	3	3	3
CO2	solvereal-timeproblemsusinggraphtheory	3	3	3	3
CO3	usetechniquesofknowledgerepresentationandprobabilisticreasoning	3	3	3	3
CO4	designvarioussupervisedlearningalgorithms	3	3	3	3
CO5	implementvariousunsupervisedlearningalgorithms	3	3	3	3

Course Mapping

Course Name	PEO1	PEO2	PEO3	PSO1	PSO2	PSO3
Artificial Intelligence and Machine LearningLab	√	√	√	√	√	√

Course Name	Po_1	Po_2	Po_3	Po_4	Po_5	Po_6	Po_7	Po_8	Po_9	Po_{10}	Po_{11}	Po_{12}
Artificial Intelligence and Machine LearningLab				√	√				√			

5. MAPPING OF COURSE WITH PEOS-PSOS-POS

Program Educational Objectives (PEOs)

Sl. No.	PEOs Name	Program Education Objective Statements
1	PEO - 1	Impart profound knowledge in humanities and basic sciences along with core engineering concepts for practical understanding & project development.[PO's: 1,2,3,4,5,7,8,9,10,11 and 12] [PSO's: 1 and 2]
2	PEO – 2	Enrich analytical skills and Industry-based modern technical skills in core and interdisciplinary areas for accomplishing research, higher education, entrepreneurship and to succeed in various engineering positions globally. [PO's: 1,2,3,4,5,6,7,8,9,10 and 12] [PSO's: 1, 2 and 3]
3	PEO – 3	Infuse life-long learning, professional ethics, responsibilities and adaptation to innovation along with effective communication skills with a sense of social awareness. [PO's: 1,2,3,4,5,6,7,8,9,10,11 and 12] [PSO's: 2 and 3]

Program Specific Objectives (PSOs)

Sl. No.	PSOs Name	Program Specific Objective Statements
1	PSO - 1	Use mathematical abstractions and Algorithmic design along with open source programming tools to solve complexities involved in efficient programming.[PO:1,2,3,4 and 5] & [PEO:1 and 2]
2	PSO – 2	Ensure programming & documentation skills for each individual student in relevant subjects i.e., C, C^{++}, Java, DBMS, Web Technologies (Development), Linux, Data Warehousing & Data Mining and on Testing Tools.[PO:1,2,3,4,5,10 and 11] & [PEO:1,2 and 3]
3	PSO – 3	Ensure employability and career development skills through Industry oriented mini & major projects, internship, industry visits, seminars and workshops. [PO:6,7,8,9,10,11 and 12] & [PEO:1,2 and 3]

Program Outcomes (POs)

PO Name	Graduate Attributes	PO Statements
PO1	Engineering knowledge	Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems. [PEO's: 1,2 and 3] [PSO's: 1,2 and 3]
PO 2	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. [PEO's: 1,2 and 3] [PSO's: 1,2 and 3]
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. [PEO's: 1,2 and 3] [PSO's: 1,2 and 3]
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. [PEO's: 1,2 and 3] [PSO's: 1,2 and 3]
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. [PEO's: 1,2 and 3] [PSO's: 1,2 and 3]
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. [PEO's: 2 and 3]
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. [PEO's: 1,2 and 3]
PO 8	Ethics	Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice. [PEO's: 1,2 and 3] [PSO's: 2 and 3]
PO 9	Individual and team work	Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings. [PEO's: 1,2 and 3] [PSO's: 3]
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. [PEO's: 1,2 and 3] [PSO's: 2 and 3]
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. [PEO's: 1 and 3] [PSO's: 2 and 3]
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. [PEO's: 1,2 and 3] [PSO's: 1,2 and 3]

MAPPINGOFCOURSEOUTCOMESWITHPEO'S,PO'S
(Nocorrelation:0;Low:1;Medium:2;High:3)

Course Outcomes	PEO1	PEO2	PEO3	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
CO-1	3	3	3	0	0	0	3	3	0	0	0	3	0	0	0
CO–2	3	3	3	0	0	0	3	3	0	0	0	3	0	0	0
CO–3	3	3	3	0	0	0	3	3	0	0	0	3	0	0	0
CO–4	3	3	3	0	0	0	3	3	0	0	0	3	0	0	0
CO–5	3	3	3	0	0	0	3	3	0	0	0	3	0	0	0

6. Mapping Of Course Outcomes With PEOs

(Ticking & Correlation - 2 Tables)

No	Course Outcomes	PEO1	PEO2	PEO3
1	CO - 1	√	√	√
2	CO – 2	√	√	√
3	CO – 3	√	√	√
4	CO – 4	√	√	√
5	CO – 5	√	√	√

7. Mapping Of Course Outcomes With PSOs

(Ticking & Correlation - 2 Tables)

No	Course Outcomes	PSO1	PSO2	PSO3
1	CO - 1	√	√	√
2	CO – 2	√	√	√
3	CO – 3	√	√	√
4	CO – 4	√	√	√
5	CO – 5	√	√	√

No	Course Outcomes	PSO1	PSO2	PSO3	Avg
1	CO - 1	3	3	2	2.67
2	CO – 2	3	3	2	2.67
3	CO – 3	3	3	3	3.00
4	CO – 4	2	3	1	2.00
5	CO – 5	2	3	1	2.00
	Avg	2.6	3	1.8	2.46

8. Mapping Of Course Outcomes With POs

(Ticking & Correlation - 2 Tables)

No	Course Outcomes	PO_1	PO_2	PO_3	PO_4	PO_5	PO_6	PO_7	PO_8	PO_9	PO_{10}	PO_{11}	PO_{12}
1	CO - 1				√	√				√			
2	CO – 2				√	√				√			
3	CO – 3				√	√				√			
4	CO – 4				√	√				√			
5	CO – 5				√	√				√			

No	Course Outcomes	PO_1	PO_2	PO_3	PO_4	PO_5	PO_6	PO_7	PO_8	PO_9	PO_{10}	PO_{11}	PO_{12}	Avg
1	CO - 1				3	3				3				2.8
2	CO – 2				3	3				3				2.6
3	CO – 3				3	3				3				2.6
4	CO – 4				3	3				3				1.4
5	CO – 5				3	3				3				1.4
	Avg				3	3				3				1.96

9. Direct Course Assessment

(As mentioned in following table of 10 parameters, of which consider only the parameters required for this courses)

No	Description	Targeted Performance	Actual Performance	Remarks	Course Attainment
1	Internal Marks(25)	80% of Students(182 Students) should Secure 60% of Internal Marks i.e., 15 Marks			
2	External Marks(50)	80% of Students(182 Students) should Secure 70% of External Marks i.e., 35 Marks			
3	Clearing of Subject	A minimum of 95% of Students(216 Students) should clear this course in first attempt			
4	Getting First Class	90% of Students(205 Students) should Secure I Class Marks i.e., 45 Marks in my course			
5	Distinction	80% of Students (182 Students) should secure First Class With Distinction i.e., 53 Marks in my course			
6	Outstanding Performance	60% of Students (137 Students) should secure 80% and above Marks. i.e., 60 Marks in my course			

10. Indirect Course Assessment

(As mentioned-strong (3), moderate (2), weak (1) & no comment (0))

Mission Statement of CSE
• Impart fundamentals through state of art technologies for research and career in Computer Science & Engineering. • Create value-based, socially committed professionals for anticipating and satisfyingfast changing societal requirements. • Foster continuous self learning abilities through regular interaction with various stake holders for holistic development.

Correlation of Mission Elements with Mission Statement of CSE Department related to the Course (only Ticking given by faculty)

No	Mission Elements	Strong	Moderate	Weak	No Comment
M-1	Impart Fundamentals	√			
M-2	State Of Art Technologies	√			
M-3	Research & Career Development	√			
M-4	Value based Socially Committed Professional	√			
M-5	Anticipating & Satisfying Industry Trends		√		
M-6	Changing Societal Requirements			√	
M-7	Foster Continuous Learning	√			
M-8	Self Learning Abilities	√			
M-9	Interaction with stakeholders	√			
M-10	Holistic Development		√		

Indirect Course Assessment through Student Satisfaction Survey

(Note for *:Parameters used for course teaching like

a: Classroom teaching b: Simulations c:labs d: Mini_Projects
e: Major Projects f: Conferences g: professional activities
h: Technical Clubs i: Guest Lectures j: Workshops k: Technical Fests l:Tutorials
m:NPTLs n:Digital Library o: Industrial Visits
p: software Tools q: Internship/training r:Technical Seminars
s: NSS t: NSS u: sports etc.

No	Question Based on PEO/ PO/PSO/CO	Parameters (a /b /c…/)*	Strong (3)	Moderate (2)	Weak (1)	No comment (0)
1	Did the course impart fundamentals through interactive learning and contribute to core competence?					
2	Did the course provide the required knowledge to foster continuous learning?					
3	Whether the syllabus content anticipates & satisfies the industry and societal needs?					
4	Whether the course focuses on value based education to be a socially committed professional?					
5	Rate the role of the facilitator in mentoring and promoting the self learning abilities to excel academically and professionally?					
6	Rate the methodology adopted and techniques used in teaching learning processes?					
7	Rate the course in applying sciences & engineering fundamentals in providing research based conclusions with the help of modern tools?					
8	Did the course have any scope to design, develop and test a system or component?					
9	Rate the scope of this course in addressing cultural, legal, health, environment and safety issues?					
10	Scope of applying management fundamentals to demonstrate effective technical project presentations & report writing?					
	Total					
	Average					
Total Average			2.52			

11. Overall Course Assessment

(80% Direct + 20% Indirect, if any)

No	Assessment Type	Weightage	Attainment Level
1	Direct-Assignment, Quiz, Subjective, University Exams, Results, Bench Marks	0.8	
2	Indirect-Surveys-Questionnaire	0.2	
	Overall		

AI & ML LAB Course Attainment level:

12. Pi diagrams, Bar charts, Histograms

(For representing previous results, if any)

WT Pass % for Last 4 Academic Years	Appeared	Passed	Pass%

INDEX

EXPERIMENT NO: 1

AIM :Write a program to implement BFS and DFS Traversal.

ALGORITHM:

DFS (Depth first search) starts with a given node and explores the first unexplored node it comes across before returning to itself again and exploring its remaining nodes (e.g: if the parent node 1 has 2 children 2, 3 the DFS method will explore 2 and its children nodes before exploring 3. It will print self before exploring its children (so 1->(2,3) will print 1,2,3))

BFS (Breadth first search) works down a tree in a top-to-bottom manner (e.g: a graph with parent 1 and children 2, 3 will print level 1 first (1) then level 2 (2, 3) and then level 3 (the children of nodes 2 and 3). The level of a given node is determined by the highest level it could appear on (e.g: if 2 is a child of an item on level 1 and level 4, it would be printed as if it were a level 2 item)

SOURCE CODE:

```
from collections import defaultdict

class Graph():
   def __init__(self):
self.value = defaultdict(list)

   def addConnection(self, parent, child):
self.value[parent].append(child)

   def DFS(self, start):
      visited = [start]
      stack = [start]
print(start, end = " ")
      while stack:
         s = stack[-1]
         if any([item for item in self.value[s] if item not in visited]):
            for item in [item for item in self.value[s] if item not in visited]:
stack.append(item)
visited.append(item)
print(item, end= " ")
                  break
         else:
stack.pop()

   def BFS(self, start):
      visited = [start]
      queue = [start]
      while queue:
         x = queue.pop(0)
print(x, end= " ")
         for item in self.value[x]:
            if item not in visited:
queue.append(item)
```

```
visited.append(item)

#Build the graph
g=Graph()
g.addConnection(1,4)
g.addConnection(1,2)
g.addConnection(2,3)
g.addConnection(2,6)
g.addConnection(4,5)
g.addConnection(4,7)
g.addConnection(7,96)

#Explore the graph
g.DFS(1)
print("\n")
g.BFS(1)
```

OUTPUT:

```
DFS: 145796236
BFS: 142573696
```

EXPERIMENT NO: 2

AIM : Write a program to implement A*Search.

ALGORITHM:

1: Firstly, Place the starting node into OPEN and find its f (n) value.
2: Then remove the node from OPEN, having the smallest f (n) value. If it is a goal node, then stop and return to success.
3: Else remove the node from OPEN, and find all its successors.
4: Find the f (n) value of all the successors, place them into OPEN, and place the removed node into CLOSE.
5: Goto Step-2.
6: Exit.

SOURCE CODE:

```
def aStarAlgo(start_node, stop_node):
open_set = set(start_node)
closed_set = set()
   g = {}              #store distance from starting node
   parents = {}        # parents contains an adjacency map of all nodes
   #distance of starting node from itself is zero
   g[start_node] = 0
   #start_node is root node i.e it has no parent nodes
   #sostart_node is set to its own parent node
   parents[start_node] = start_node
   while len(open_set) > 0:
      n = None
      #node with lowest f() is found
      for v in open_set:
         if n == None or g[v] + heuristic(v) < g[n] + heuristic(n):
            n = v
      if n == stop_node or Graph_nodes[n] == None:
         pass
      else:
         for (m, weight) in get_neighbors(n):
            #nodes 'm' not in first and last set are added to first
            #n is set its parent
            if m not in open_set and m not in closed_set:
open_set.add(m)
               parents[m] = n
               g[m] = g[n] + weight
            #for each node m,compare its distance from start i.e g(m) to the
            #from start through n node
            else:
               if g[m] > g[n] + weight:
                  #update g(m)
                  g[m] = g[n] + weight
                  #change parent of m to n
                  parents[m] = n
                  #if m in closed set,remove and add to open
```

```
                    if m in closed_set:
closed_set.remove(m)
open_set.add(m)
        if n == None:
print('Path does not exist!')
            return None

        # if the current node is the stop_node
        # then we begin reconstructin the path from it to the start_node
        if n == stop_node:
            path = []
            while parents[n] != n:
path.append(n)
                n = parents[n]
path.append(start_node)
path.reverse()
print('Path found: {}'.format(path))
            return path
        # remove n from the open_list, and add it to closed_list
        # because all of his neighbors were inspected
open_set.remove(n)
closed_set.add(n)
print('Path does not exist!')
    return None

#define fuction to return neighbor and its distance
#from the passed node
def get_neighbors(v):
    if v in Graph_nodes:
        return Graph_nodes[v]
    else:
        return None
def heuristic(n):
H_dist = {
        'A': 11,
        'B': 6,
        'C': 5,
        'D': 7,
        'E': 3,
        'F': 6,
        'G': 5,
        'H': 3,
        'I': 1,
        'J': 0
    }
    return H_dist[n]
Graph_nodes = {
    'A': [('B', 6), ('F', 3)],
    'B': [('A', 6), ('C', 3), ('D', 2)],
    'C': [('B', 3), ('D', 1), ('E', 5)],
    'D': [('B', 2), ('C', 1), ('E', 8)],
```

```
    'E': [('C', 5), ('D', 8), ('I', 5), ('J', 5)],
    'F': [('A', 3), ('G', 1), ('H', 7)],
    'G': [('F', 1), ('I', 3)],
    'H': [('F', 7), ('I', 2)],
    'I': [('E', 5), ('G', 3), ('H', 2), ('J', 3)],
  }

aStarAlgo('A', 'J')
```

OUTPUT:

```
Path found: ['A', 'F', 'G', 'I', 'J']
            ['A', 'F', 'G', 'I', 'J']
```

EXPERIMENT NO: 3

AIM:Write a program to implement Travelling Salesman Problem and Graph Coloring Problem.

ALGORITHM – Travelling saleman problem

- Consider city 1 as the starting and ending point.
- Generate all (n-1)! Permutations of cities.
- Calculate cost of every permutation and keep track of minimum cost permutation.
- Return the permutation with minimum cost.

ALGORITHM – Graph coloring problem

- Color first vertex with first color.
- Do following for remaining V-1 vertices.
  ….. **a)** Consider the currently picked vertex and color it with the lowest numbered color that has not been used on any previously colored vertices adjacent to it. If all previously used colors appear on vertices adjacent to v, assign a new color to it.

SOURCE CODE- Travelling saleman problem

```
from sys import maxsize
from itertools import permutations
V = 4
# implementation of traveling Salesman Problem
def travellingSalesmanProblem(graph, s):
    # store all vertex apart from source vertex
    vertex = []
    for i in range(V):
        if i != s:
            vertex.append(i)
    # store minimum weight Hamiltonian Cycle
    min_path = maxsize
    next_permutation=permutations(vertex)
    for i in next_permutation:
        # store current Path weight(cost)
        current_pathweight = 0
        # compute current path weight
        k = s
        for j in i:
            current_pathweight += graph[k][j]
            k = j
        current_pathweight += graph[k][s]
        # update minimum
        min_path = min(min_path, current_pathweight)
    return min_path
```

```
 # Driver Code
if __name__ == "__main__":
    # matrix representation of graph
    graph = [[0, 10, 15, 20], [10, 0, 35, 25],
          [15, 35, 0, 30], [20, 25, 30, 0]]
    s = 0
    print(travellingSalesmanProblem(graph, s))
```

OUTPUT- Travelling saleman problem

80

SOURCE CODE:

```
#!/usr/bin/env python
# coding: utf-8
# In[6]:
colors = ['Red', 'Blue', 'Green', 'Yellow', 'Black']
states = ['Andhra', 'Karnataka', 'TamilNadu', 'Kerala']
neighbors = {}
neighbors['Andhra'] = ['Karnataka', 'TamilNadu']
neighbors['Karnataka'] = ['Andhra', 'TamilNadu', 'Kerala']
neighbors['TamilNadu'] = ['Andhra', 'Karnataka', 'Kerala']
neighbors['Kerala'] = ['Karnataka', 'TamilNadu']
colors_of_states = {}
def promising(state, color):
    for neighbor in neighbors.get(state):
        color_of_neighbor = colors_of_states.get(neighbor)
        if color_of_neighbor == color:
            return False
    return True
def get_color_for_state(state):
    for color in colors:
        if promising(state, color):
            return color

def main():
    for state in states:
        colors_of_states[state] = get_color_for_state(state)
    print (colors_of_states)
main()
# In[ ]:
 # In[ ]:
```

OUTPUT – Graph coloring peroblem.

{'Andhra': 'Red', 'Karnataka': 'Blue', 'TamilNadu': 'Green', 'Kerala': 'Red'}

EXPERIMENT NO: 4

AIM :Write a program to implement Knowledge Representation.

ALGORITHM:

Let's start with a Harry Potter example. Consider the following sentences:

1. If it didn't rain, Harry visited Hagrid today.

2. Harry visited Hagrid or Dumbledore today, but not both.

3. Harry visited Dumbledore today.

Based on these three sentences, we can answer the question "did it rain today?", even though none of the individual sentences tell us anything about whether it is raining today. Here is how we can go about it: looking at sentence 3, we know Harry visited Dumbledore. Looking at sentence 2, we know Harry visited either Dumbledore or Hagrid, and thus we can conclude **4. Harry did not visit Hagrid.**

Now, looking at sentence 1, we understand that if it didn't rain, Harry would have visited Hagrid. However, knowing sentence 4, we know that this is not the case. Therefore, we can conclude **5. It rained today.**

SOURCE CODE:

```
# Here we import everything from logic.py
from logic import *
rain = Symbol("rain")     # Rain is the symbol for rain
hagrid = Symbol("hagrid")
dumbledore = Symbol("dumpledore")
# we create here a logical sentence using logical connectives
logical_sentence = And(rain, hagrid)
# we can't directly print the logical sentence so we use a formula function
print(logical_sentence.formula())
# we create a implication logic here
implication_logic = Implication(Not(rain), hagrid)
print(implication_logic.formula())
"""We make decision from knowledge base " ,all of the statement will be true then knowlegde base will be true"'
knowledge_base = And(
Implication(Not(rain), hagrid),
Or(hagrid, dumbledore),
Not(And(hagrid, dumbledore)),
```

```
hagrid
)
print(knowledge_base.formula())
''' Here we use the model_check function and here two arguments are knoledge_base and another is it rain today or not. Then we print it'''
print(model_check(knowledge_base, rain))
```

OUTPUT:

Rain ^ hagrid

(¬ rain =>hagrid

((¬ rain) =>hagrid ^ (hagrid v dumledore) ^ (¬(hagrid ^ dumbledore)) ^ dumbledore

true

EXPERIMENT NO: 5

AIMWrite a program to implement Bayesian Network.

ALGORITHM:

A Bayesian network, Bayes network, belief network, decision network, Bayes(ian) model or probabilistic directed acyclic graphical model is a probabilistic graphical model (a type of statistical model) that represents a set of variables and their conditional dependencies via a directed acyclic graph (DAG). Bayesian networks are ideal for taking an event that occurred and predicting the likelihood that any one of several possible known causes was the contributing factor. For example, a Bayesian network could represent the probabilistic relationships between diseases and symptoms. Given symptoms, the network can be used to compute the probabilities of the presence of various diseases.

GraphicalModel:

Formally,Bayesiannetworksaredirectedacyclicgraphs(DAGs)whosenodesrepresentvariablesin the Bayesian sense:theymaybeobservablequantities, latentvariables,unknownparametersorhypotheses.Edges represent conditionaldependencies;nodes thatare notconnected (no path connectsone node to another) represent variables that are conditionally independent of each other. Each node isassociatedwitha probabilityfunction thattakes,asinput,aparticularset of values for thenode's parent variables, and gives (as output) the probability (or probability distribution, if applicable) ofthevariable representedbythe node.

SOURCE CODE:

```
import numpy
from pomegranate import *
guest = DiscreteDistribution({'A': 1./3, 'B': 1./3, 'C': 1./3})
prize = DiscreteDistribution({'A': 1./3, 'B': 1./3, 'C': 1./3})


monty = ConditionalProbabilityTable(
[[ 'A', 'A', 'A', 0.0 ],
[ 'A', 'A', 'B', 0.5 ],
[ 'A', 'A', 'C', 0.5 ],
[ 'A', 'B', 'A', 0.0 ],
[ 'A', 'B', 'B', 0.0 ],
[ 'A', 'B', 'C', 1.0 ],
[ 'A', 'C', 'A', 0.0 ],
[ 'A', 'C', 'B', 1.0 ],
[ 'A', 'C', 'C', 0.0 ],
[ 'B', 'A', 'A', 0.0 ],
[ 'B', 'A', 'B', 0.0 ],
[ 'B', 'A', 'C', 1.0 ],
[ 'B', 'B', 'A', 0.5 ],
```

```
[ 'B', 'B', 'B', 0.0 ],
[ 'B', 'B', 'C', 0.5 ],
[ 'B', 'C', 'A', 1.0 ],
[ 'B', 'C', 'B', 0.0 ],
[ 'B', 'C', 'C', 0.0 ],
[ 'C', 'A', 'A', 0.0 ],
[ 'C', 'A', 'B', 1.0 ],
[ 'C', 'A', 'C', 0.0 ],
[ 'C', 'B', 'A', 1.0 ],
[ 'C', 'B', 'B', 0.0 ],
[ 'C', 'B', 'C', 0.0 ],
[ 'C', 'C', 'A', 0.5 ],
[ 'C', 'C', 'B', 0.5 ],
[ 'C', 'C', 'C', 0.0 ]], [guest, prize])


s1 = State(guest, name="guest")
s2 = State(prize, name="prize")
s3 = State(monty, name="monty")
model = BayesianNetwork("Monty Hall Problem")
model.add_states(s1, s2, s3)
model.add_edge(s1, s3)
model.add_edge(s2, s3)
model.bake()
print(model.probability([['A', 'B', 'C'],['A','A','C'],['A','C','C']]))
print(model.predict([['A',None,'C'],['A','A',None],[None,'B','A']]))
```

EXPERIMENT NO: 6

AIM :Write a program to implement Hidden Markov Model.

ALGORITHM:

The Hidden Markov Model (HMM) algorithm can be implemented using the following steps:

Step 1: Define the state space and observation space
The state space is the set of all possible hidden states, and the observation space is the set of all possible observations.
Step 2: Define the initial state distribution
This is the probability distribution over the initial state.
Step 3: Define the state transition probabilities
These are the probabilities of transitioning from one state to another. This forms the transition matrix, which describes the probability of moving from one state to another.
Step 4: Define the observation likelihoods:
These are the probabilities of generating each observation from each state. This forms the emission matrix, which describes the probability of generating each observation from each state.
Step 5: Train the model
The parameters of the state transition probabilities and the observation likelihoods are estimated using the Baum-Welch algorithm, or the forward-backward algorithm. This is done by iteratively updating the parameters until convergence.
Step 6: Decode the most likely sequence of hidden states
Given the observed data, the Viterbi algorithm is used to compute the most likely sequence of hidden states. This can be used to predict future observations, classify sequences, or detect patterns in sequential data.
Step 7: Evaluate the model
The performance of the HMM can be evaluated using various metrics, such as accuracy, precision, recall, or F1 score.

SOURCE CODE:

```
import numpy as np
import itertools
import pandas as pd
# create state space and initial state probabilities
states = ['sleeping', 'eating', 'pooping']

hidden_states = ['healthy', 'sick']
pi = [0.5, 0.5]
state_space = pd.Series(pi, index=hidden_states, name='states')
print(state_space)



a_df = pd.DataFrame(columns=hidden_states, index=hidden_states)
a_df.loc[hidden_states[0]] = [0.7, 0.3]
a_df.loc[hidden_states[1]] = [0.4, 0.6]
```

```
print(a_df)

observable_states = states

b_df = pd.DataFrame(columns=observable_states, index=hidden_states)
b_df.loc[hidden_states[0]] = [0.2, 0.6, 0.2]
b_df.loc[hidden_states[1]] = [0.4, 0.1, 0.5]

print(b_df)

def HMM(obsq,a_df,b_df,pi,states,hidden_states):
    hidst=list(itertools.product(hidden_states,repeat=len(obsq)))
    print(hidst)
    sum=0
    for k in hidst:
                prod=1
                for j in range(len(k)):
                        c=0
                        for i in obsq:
                                if c==0:
                                        prod*=a_df[i][k[j]]*pi[hidden_states.index(k[j])]
                                        c=1
                                else:
                                        prod*=b_df[k[j]][k[j-1]]*a_df[i][k[j]]
                sum+=prod
                c=0
    return sum
def vertibi(obsq,a_df,b_df,pi,states,hidden_states):
    sum=0
    hidst=list(itertools.product(hidden_states,repeat=len(obsq)))
    for k in hidst:
                sum1=0
                prod=1
                for j in range(len(k)):
                        c=0
                        for i in obsq:
                                if c==0:
                                        prod*=a_df[i][k[j]]*pi[hidden_states.index(k[j])]
                                        c=1
                                else:
                                        prod*=b_df[k[j]][k[j-1]]*a_df[i][k[j]]
                c=0
                sum1+=prod
                if(sum1>sum):
                        sum=sum1
                        hs=k
    return sum,hs
```

```
    obsq=['pooping','pooping','pooping']
    print(HMM(obsq,b_df,a_df,pi,states,hidden_states))
print(vertibi(obsq,b_df,a_df,pi,states,hidden_states))
```

OUTPUT:

```
healthy    0.5
sick       0.5
Name: states, dtype: float64
       healthy sick
healthy     0.7  0.3
sick        0.4  0.6
       sleeping eating pooping
healthy      0.2    0.6     0.2
sick         0.4    0.1     0.5
[('healthy', 'healthy', 'healthy'), ('healthy', 'healthy', 'sick'), ('healthy', 'sick', 'healthy'), ('healthy', 'sick', 'sick'), ('sick', 'healthy', 'healthy'), ('sick', 'healthy', 'sick'), ('sick', 'sick', 'healthy'), ('sick', 'sick', 'sick')]
1.1662322536e-05
(1.1390624999999999e-05, ('sick', 'sick', 'sick'))
```

EXPERIMENT :7

AIM :Write a program to implement Regression algorithm

ALGORITHM:

STEP 1:Linear Assumption. Linear regression assumes that the relationship between your input and output is linear. It does not support anything else. This may be obvious, but it is good to remember when you have a lot of attributes. You may need to transform data to make the relationship linear (e.g. log transform for an exponential relationship).
STEP 2: Remove Noise. Linear regression assumes that your input and output variables are not noisy. Consider using data cleaning operations that let you better expose and clarify the signal in your data. This is most important for the output variable and you want to remove outliers in the output variable (y) if possible.
STEP 3:Remove Collinearity. Linear regression will over-fit your data when you have highly correlated input variables. Consider calculating pairwise correlations for your input data and removing the most correlated.
STEP 4:Gaussian Distributions. Linear regression will make more reliable predictions if your input and output variables have a Gaussian distribution. You may get some benefit using transforms (e.g. log or BoxCox) on you variables to make their distribution more Gaussian looking.
STEP 5:Rescale Inputs: Linear regression will often make more reliable predictions if you rescale input variables using standardization or normalization.

SOURCE CODE:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

dataset = pd.read_csv('Salary_Data.csv')
dataset.head()

# data preprocessing
X = dataset.iloc[:, :-1].values  #independent variable array
y = dataset.iloc[:,1].values  #dependent variable vector

# splitting the dataset
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=1/3,random_state=0)

# fitting the regression model
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train,y_train) #actually produces the linear eqn for the data

# predicting the test set results
y_pred = regressor.predict(X_test)
```

```
y_pred

y_test

# visualizing the results
#plot for the TRAIN

plt.scatter(X_train, y_train, color='red') # plotting the observation line
plt.plot(X_train, regressor.predict(X_train), color='blue') # plotting the regression line
plt.title("Salary vs Experience (Training set)") # stating the title of the graph

plt.xlabel("Years of experience") # adding the name of x-axis
plt.ylabel("Salaries") # adding the name of y-axis
plt.show() # specifies end of graph

#plot for the TEST

plt.scatter(X_test, y_test, color='red')
plt.plot(X_train, regressor.predict(X_train), color='blue') # plotting the regression line
plt.title("Salary vs Experience (Testing set)")

plt.xlabel("Years of experience")
plt.ylabel("Salaries")
plt.show()
```

OUTPUT:

EXPERIMENT NO: 8

AIM :Write a program to demonstrate the working of the decision tree based ID3 algorithm. Usean appropriate data set for building the decision tree and apply this knowledge to classify a newsample.

Description:

- ID3 algorithm is a basic algorithm that learns decision trees by constructing them topdown, beginning with the question "which attribute should be tested at the root of thetree".
- To answer this question, each instance attribute is evaluated using a statistical test todetermine how well it alone classifies the training examples. The best attribute is selectedandusedas thetestatthe rootnodeofthetree.
- A descendant of the root node is then created for each possible value of this attribute, andthe training examples are sorted to the appropriate descendant node (i.e., down the branchcorrespondingtotheexample'svalueforthis attribute).
- The entire process is then repeated using the training examples associated with eachdescendantnodetoselect the bestattributetotestatthatpointinthetree.
- A simplified version of the algorithm, specialized to learning boolean-valued functions(i.e.,conceptlearning),is describedbelow.

Algorithm:

ID3(Examples,Target
Attribute,Attributes)Input:Examplesarethetr
ainingexamples.
Targetattributeistheattributewhosevalueistobepredictedbythetree.Attributes isalistofother attributesthatmaybetestedbythelearneddecisiontree.
Output:ReturnsadecisiontreethatcorrectlyclassifiesthegivenExamplesMethod
:
1. CreateaRoot node forthetree
2. IfallExamplesare positive,Returnthesingle-nodetree Root,withlabel= +
3. IfallExamples are negative,Returnthesingle-nodetree Root,withlabel=-
4. IfAttributesisempty,
Return the single-node tree Root, with label = most common value ofTargetAttributeinExamples
Else
A←theattributefromAttributesthatbestclassifiesExamplesThe decisionattributeforRoot←A
Foreachpossiblevalue, vi, ofA,
Add a new tree branch below Root, corresponding to the test A = viLetExamplesvibethesubsetofExamplesthathavevaluevi for A
Else IfExamplesvi isemptyThenbelowthisnewbranchadd aleafnodewith label=mostcommonvalueofTargetAttribute inExamples
{A})
End BelowthisnewbranchaddthesubtreeID3(Examplesvi,TargetAttribute,Attributes–

5. ReturnRoot

Sourcecode:

```
import pandas as pd
df=pd.read_csv("lab2dataset.csv")

#Calculate Entropy
def entropy(probs):
    import math
    return sum(-prob*math.log(prob,2) for prob in probs)

# Calculating the Probability of Positive and negative examples
def entropy_of_list(a_list):
    from collections import Counter
cnt = Counter (x for x in a_list)
num_instances =len(a_list)
    probs=[x/num_instances for x in cnt.values()]
    return entropy(probs)

total_entropy= entropy_of_list(df['PlayTime'])

def information_gain(df,split_attribute_name,    target_attribute_name, trace=0):
df_split =df.groupby(split_attribute_name)
    for name,group in df_split:
    #print("Name", name)
    #print("Group",group)
    nobs=len(df.index)*1.0
    # Calculating Entropy of an attribute and probability part of formula
df_agg_ent=df_split.agg({target_attribute_name:          [entropy_of_list,lambda x: len(x)/nobs
     #print("df_agg_ent", df_agg_ent)
     #print(df_agg_ent.columns)
     #calculate information gain
avg_info=sum(df_agg_ent['entropy_of_list'] *     df_agg_ent['<lambda_0>'])
old_entropy=entropy_of_list(df[target_attribute_name])
return old_entropy-avg_

def id3DT(df, target_attribute_name, attribute_names, default_class=None):
     from collections import Counter
cnt = Counter(x for x in df[target_attribute_name])
     if len(cnt)==1:
          return next(iter(cnt))
elifdf.empty or (not attribute_names):
          return default_class
      else:
```

```
default_class =max(cnt.keys())
        #print("attributes_names:",attribute_names)
gainz=[information_gain(df,attr, target_attribute_name) for attr in attribute_names]
index_of_max=gainz.index(max(gainz))
best_attr=attribute_names[index_of_max]
        tree={best_attr:{}}
remaining_attributes_names=[i for i in attribute_names if i != best_attr]
        for attr_val, data_subset in df.groupby(best_attr):
subtree=id3DT(data_subset,target_attribute_name,remaining_attributes_names,default)
            tree[best_attr][attr_val]=subtree
return tree

attribute_names=list(df.columns)
attribute_names.remove('PlayTime')
from pprint import pprint
tree= id3DT(df,'PlayTime',attribute_names)
print("The Resultant Decision Tree is ")
pprint(tree)
attribute=next(iter(tree))
print("Best Attribute: \n", attribute)
print("Tree Keys\n ", tree[attribute].keys())

    def classify(instance, tree, default=None):
      attribute=next(iter(tree))
      print("Key:",tree.keys())
      print("Attribute",attribute)
      if instance[attribute] in tree[attribute].keys():
        result=tree[attribute][instance[attribute]]
    print("Instance Attribute:",instance[attribute], "TreeKeys:",tree[attribute].keys())
        if isinstance(result,dict):
          return classify(instance,result)
        else:
          return result
      else:
        return default

    tree1={'Outlook':['Rainy','Sunny'],'Temperature':['Mild','Hot'],'Humidity':['Normal','High'],'Windy':['Weak','Stron
    g']}
    df2=pd.DataFrame(tree1)
    df2['Predicted']=df2.apply(classify,axis=1, args=(tree,'No'))
    print(df2)
```

Output:

```
Key: dict_keys(['Outlook'])
Attribute Outlook
Key: dict_keys(['Outlook'])
Attribute Outlook
Instance Attribute: Sunny TreeKeys: dict_keys(['Overcast', 'Rain', 'Sunny'])
Key: dict_keys(['Temperature'])
Attribute Temperature
Instance Attribute: Hot TreeKeys: dict_keys(['Cool', 'Hot', 'Mild'])
  Outlook Temperature Humidity   Windy Predicted
0   Rainy        Mild   Normal    Weak        No
1   Sunny         Hot     High  Strong        No
```

EXPERIMENT-8

AIM: ImplementtheKmeansclustering Algorithms

ALGORITHM:

KmeansClustering

- Thealgorithmwillcategorizetheitemsintokgroupsofsimilarity.Tocalculatethatsimilarity,we willuse theEuclideandistance asmeasurement.
- Thealgorithmworksas follows:
  1. Firstweinitializekpoints,calledmeans,randomly.
  2. Wecategorizeeachitemtoitsclosestmeanandweupdatethemean'scoordinates,whichare the averagesoftheitemscategorizedinthatmeansofar.
  3. Werepeattheprocessforagivennumberofiterationsandattheend,wehaveourclusters.
- The "points" mentioned above are called means,because they hold the mean values oftheitems categorizedinit.To initialize thesemeans,we have alot of options.Anintuitivemethod is toinitialize the means atrandom items in the data set.Anothermethod is to initialize the means at random values between the boundaries of the data set(if for a feature x the items have values in [0,3], we will initialize the means with valuesforxat[0,3]).
- Pseudocode:

1. Initializekmeanswithrandomvalues
2. Foragivennumberofiterations:Ite ratethroughitems:

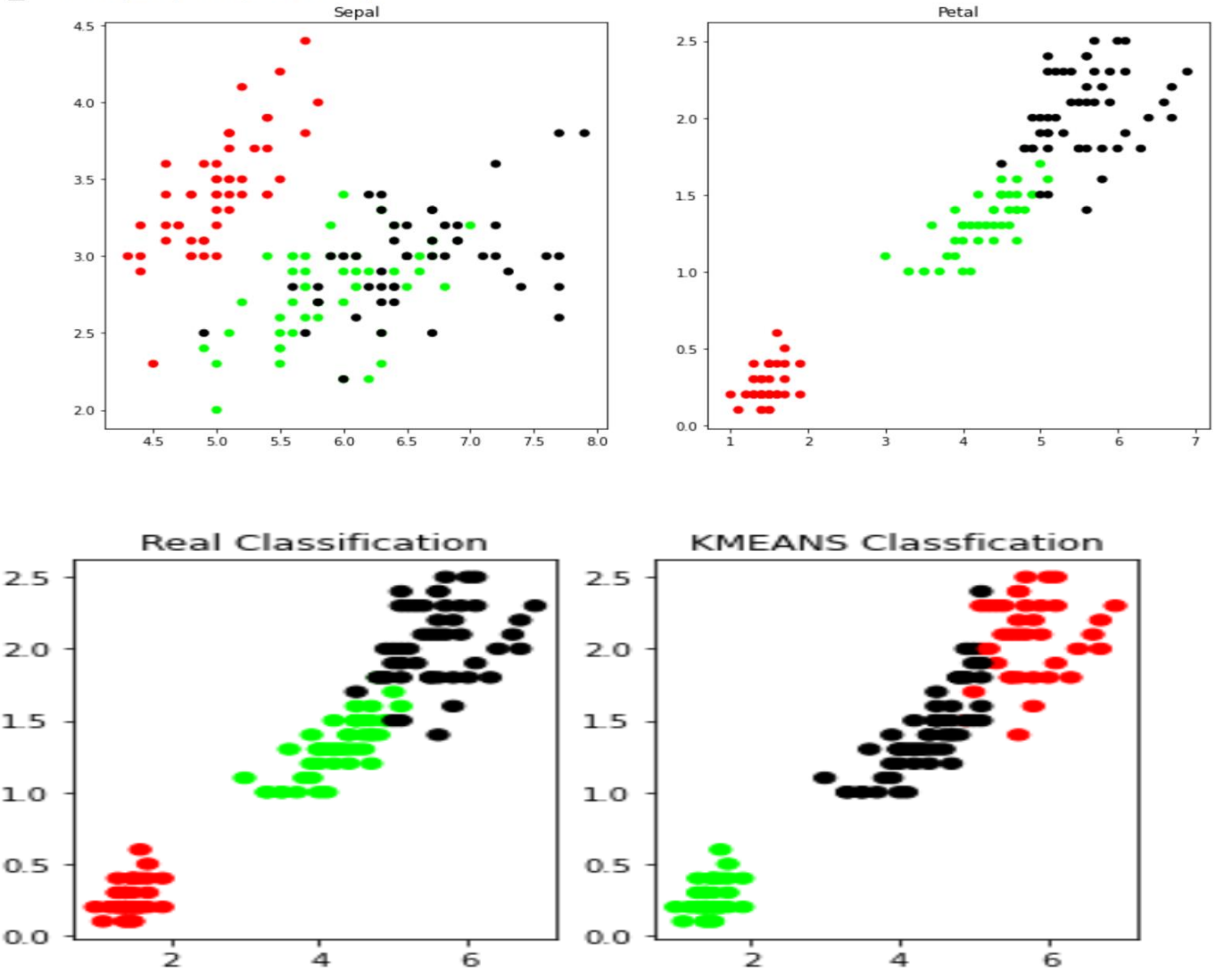
Find the mean closest to the itemAssignitemtomean Updatemean

Sourcecode:

```
import matplotlib.pyplot as plt
from sklearn import datasets
from sklearn.cluster import KMeans
import sklearn.metrics as sm
import pandas as pd
import numpy as np

iris =datasets.load_iris()
X=pd.DataFrame(iris.data)
X.columns=['Sepal_Length','Sepal_Width', 'Petal_length', 'Petal_Width']
y=pd.DataFrame(iris.target)
y.columns=['target']
plt.figure(figsize=(14,7))
colormap=np.array(['red','lime','black'])
```

```
plt.subplot(1,2,1)
plt.scatter(X.Sepal_Length,X.Sepal_Width,c=colormap[y.target],s=40)
plt.title('Sepal')
plt.subplot(1,2,2)
plt.scatter(X.Petal_length,X.Petal_Width,c=colormap[y.target],s=40)
plt.title('Petal')
model=KMeans(n_clusters=3)
model.fit(X)
print(model.labels_)
plt.subplot(1,2,1)
plt.scatter(X.Petal_length,X.Petal_Width,c=colormap[y.target],s=40)
plt.title('Real Classification')
plt.subplot(1,2,2)
plt.scatter(X.Petal_length,X.Petal_Width,c=colormap[model.labels_],s=40)
plt.title( 'KMEANS Classfication')
```

```
Text(0.5, 1.0, 'Petal')
```



EXPERIMENT-10

AIM: Writeaprogramtoimplementk-NearestNeighboralgorithmtoclassifytheirisdataset.

DESCRIPTION:

- K-Nearest Neighbors is one of the most basic yet essential classification algorithms inMachineLearning.Itbelongstothesupervisedlearningdomainandfindsintenseapplication inpatternrecognition,data miningandintrusiondetection.
- It is widely disposable in real-life scenarios since it is non-parametric, meaning, it doesnotmakeanyunderlyingassumptions aboutthe distributionofdata.

ALGORITHM:

Input:Let mbethenumberoftraining datasamples.Letpbeanunknownpoint.Method:

1. Storethetrainingsamplesinanarrayofdatapointsarr[].Thismeanseachelementofthisarrayrepresents atuple(x,y).
2. fori=0tom
   CalculateEuclideandistanced(arr[i],p).
3. Make set S of K smallest distances obtained. Each of these distances corresponds to an alreadyclassifieddatapoint.
4. ReturnthemajoritylabelamongS.

SOURCECODE:

```
import sklearn
import pandas as pd
from sklearn.datasets import load_iris
iris=load_iris()
print(iris.keys())
df=pd.DataFrame(iris['data'])
print(df)
print(iris['target_names'])
print(iris['feature_names'])
print(iris['target'])

X=df
y=iris['target']
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.33,random_state=42)
from sklearn.neighbors import KNeighborsClassifier
knn=KNeighborsClassifier(n_neighbors=3)
knn.fit(X_train,y_train)
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
y_pred=knn.predict(X_test)
cm=confusion_matrix(y_test,y_pred)
```

```
print(cm)
print("Correct prediction", accuracy_score(y_test,y_pred))
print("Wrong prediction", (1-accuracy_score(y_test,y_pred)))
y_testtrain=knn.predict(X_train)
cm1=confusion_matrix(y_train,y_testtrain)
print(cm1)
```

Output:

```
dict_keys(['data', 'target', 'frame', 'target_names', 'DESCR', 'feature_names', 'filename', 'data_module'])
     0    1    2    3
0    5.1  3.5  1.4  0.2
1    4.9  3.0  1.4  0.2
2    4.7  3.2  1.3  0.2
3    4.6  3.1  1.5  0.2
4    5.0  3.6  1.4  0.2


145  6.7  3.0  5.2  2.3
146  6.3  2.5  5.0  1.9
147  6.5  3.0  5.2  2.0
148  6.2  3.4  5.4  2.3
149  5.9  3.0  5.1  1.8
[150 rows x 4 columns]
['setosa' 'versicolor' 'virginica']
['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)']
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2]
[[19  0  0]
 [ 0 15  0]
 [ 0  1 15]]
Correct prediction 0.98
Wrong prediction 0.020000000000000018
[[31  0  0]
 [ 0 33  2]
 [ 0  2 32]]
```

EXPERIMENT -11

AIM: ImplementingBackpropagationalgorithmandtestthesameusingappropriatedatasets.

DESCRIPTION:

- Artificial neural networks (ANNs) provide a general, practical method for learning realvalued,discrete-valued,andvector-valuedfunctionsfromexamples.
- Algorithms such as BACKPROPAGATION gradient descent to tune network parameterstobestfita trainingsetofinput-outputpairs.
- ANN learning is robust to errors in the training data and has been successfully applied toproblemssuchasinterpretingvisualscenes,speechrecognition,andlearningrobotcontrolstrategies.

ALGORITHM:

1. Createafeed-forwardnetworkwithniinputs,nhiddenhiddenunits, andnoutoutputunits.
2. Initializeeachwito some smallrandomvalue(e.g.,between-.05 and .05).
3. Untiltheterminationconditionismet, do
   Foreachtrainingexample<(x1,…xn),t>,do
   // Propagatetheinputforwardthroughthenetwork:
   a. Inputtheinstance(x1,..,xn)to then/w &computethen/w outputsok foreveryunit
   //Propagatethe errorsbackwardthroughthe network:
   b. Foreachoutputunitk,calculateitserrorterm
   c. Foreachhiddenunith,calculateitserrorterm
   d. For eachnetworkweightwi,jdo;

SOURCECODE:

```
importnumpyas np
x=np.array(([2,9],[1,5],[3,6]),dtype=float)print (x)
y=np.array(([92],[86],[89]),dtype=float)

#Normalization of
datasetx=x/np.max(x,axis=0
)
y=y/100
print(x)
print(y)
defsigmoid(x):
   return(1/(1+np.exp(-x)))

def
   derivatives_sigmoid(x):retur
```

```
    nx*(1-x)
  epoch=10

  00lr=0.01

  input_layer_neurons=2hi
  dden_layer_neurons=2o
  utput_neurons=1

  wh=np.random.uniform(size=(input_layer_neurons,hidden_layer_neurons))bh=n
  p.random.uniform(size=(1,hidden_layer_neurons))
  wout=np.random.uniform(size=(hidden_layer_neurons,output_neurons)

bout=np.random.uniform(size=(1,output_neurons))
  for i in
     range(epoch):hinp1=n
     p.dot(x,wh)hinp=hinp
     1+bh
     hlayer_act=sigmoid(hinp)
     outinp1=np.dot(hlayer_act,wout)ou
     tinp=outinp1+bout
     output=sigmoid(outinp)
     EO=(y-output)
     # back Propagation
     outgrad=derivatives_sigmoid(output)d_
     output=EO*outgrad
     EH=d_output.dot(wout.T)
     hiddengrad=derivatives_sigmoid(hlayer_act)d_h
     iddenlayer=EH*hiddengrad
     #change of weight at each layer
     wout+=hlayer_act.T.dot(d_output)*lr
     bout += np.sum(d_output,axis=0,keepdims=True)
     *lrwh+=x.T.dot(d_hiddenlayer)*lr
     bh +=np.sum(d_hiddenlayer, axis=0,keepdims=True)
     *lr#output after each epoch
     #print("-----------Epoch-",i+1,"Starts---------------------------- ")
     #print("Input:\n"+ str(x))
     #print("ActualOutput: \n"+ str(y))
     #print("Predicted Output: \n" ,output)
     #print("Error:\n"+str(EO))
     #print("-----------Epoch-",i+1,"Ends ------------------------ \n")
```

```
print("Actualouput"+str(y))
print("Predicted
Output"+str(output))print("Error"+str(EO)
)
```

Output:

```
Actual Output:
[[0.92]
 [0.86]
 [0.89]]
Predicted Output:
 [[0.86627165]
 [0.8576674 ]
 [0.86966079]]
Error:
[[0.05372835]
 [0.0023326 ]
 [0.02033921]]
```

EXPERIMENT-12

AIM: Writeaprogramto do sentimentanalysisoflivetweets.

DESCRIPTION:

TextBlob: textblob is the **python** library for

processing textual **data**.**We follow these3majorsteps**

inourprogram:

1. Authorize**twitter**APIclient.
2. **Make**aGETrequest to**Twitter**API to fetch**tweets**for aparticular query.
3. Parsethe**tweets**. Classifyeach**tweet** aspositive,negativeorneutral.

SOURCECODE:

```
import tweepy
import pandas as pd
import numpy as np
from IPython.display import display
import matplotlib.pyplot as plt
import seaborn as sns
def
    twit
    ter_
    setu
    p():
    """
    Utility function to setup the Twitter's
    APIwith our access keys provided.
    """
    # Authentication and access using keys:
    auth = tweepy.OAuthHandler("API Key", "API
    Secret")auth.set_access_token("AccessKey","Acces
    sToken")

    # Return API with authentication:
    api=tweepy.API(auth)
    returnapi

# We create an extractor object:
extractor=twitter_setup()

# We create a tweet list as follows:
tweets = extractor.user_timeline(screen_name="Username of Twitter account",
count=200)print("Numberof tweets extracted:{}.\n".format(len(tweets)))

# We print the most recent 5 tweets:
print("13recenttweets:\n")
for tweet in
    tweets[:1
```

```
3]:print(t
weet.text
)
print()
```

```
# We create a pandas dataframe as follows:https://t.co/cGS4gdrBYH
data=pd.DataFrame(data=[tweet.textfortweetintweets],columns=['Tweets'])
# We display the first 10 elements of the dataframe:
display(data.head(5))
```

```
Internal methods of a single tweet object:
print(dir(tweets[0]))
```

```
# We print info from the first tweet:
print(tweets[0].id)
print(tweets[0].cre
ated_at)print(tweet
s[0].source)
print(tweets[0].favorite_
count)print(tweets[0].ret
weet_count)print(tweets
[0].geo)
print(tweets[0].coordinates)print(tweets[0].entities
```

```
# We add relevant data:
data['len']= np.array([len(tweet.text) for tweet in tweets])data['ID']
                    =np.array([tweet.idfortweetintweets])
data['Date']    =    np.array([tweet.created_at    for    tweet    in
tweets])data['Source']=np.array([tweet.sourcefortweetintweets])
data['Likes']= np.array([tweet.favorite_count for tweet in tweets])data['RTs']
                      =np.array([tweet.retweet_countfortweetintweets])
```

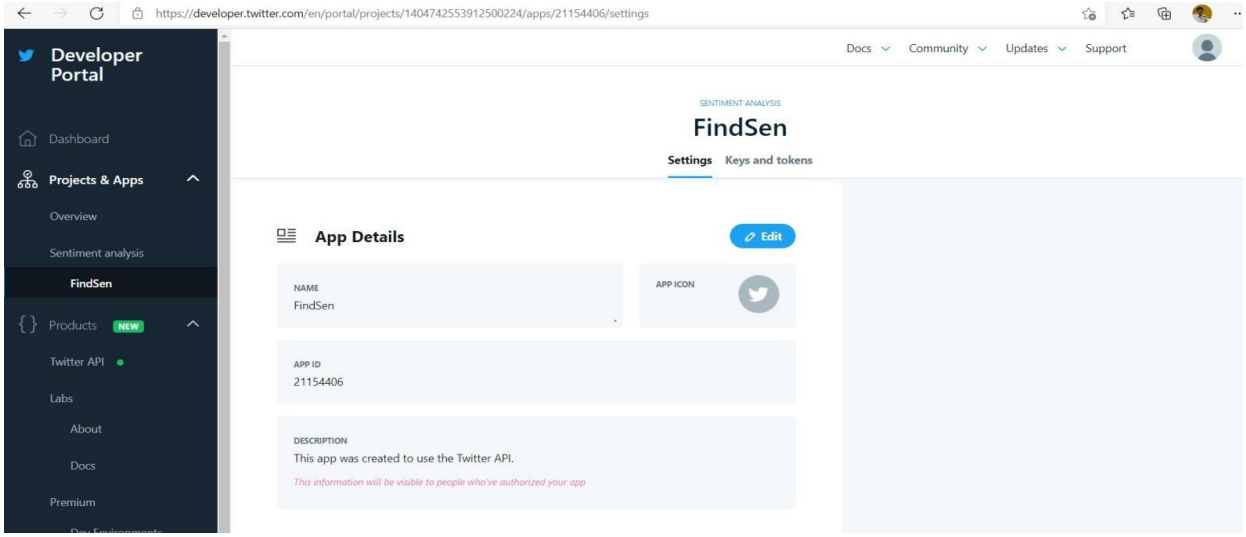
```
# Display of first 10 elements from dataframe:
display(data.head(10))
mean=np.mean(data['len'])
print("Thelenght'saverageintweets:{}".format(mean))
```

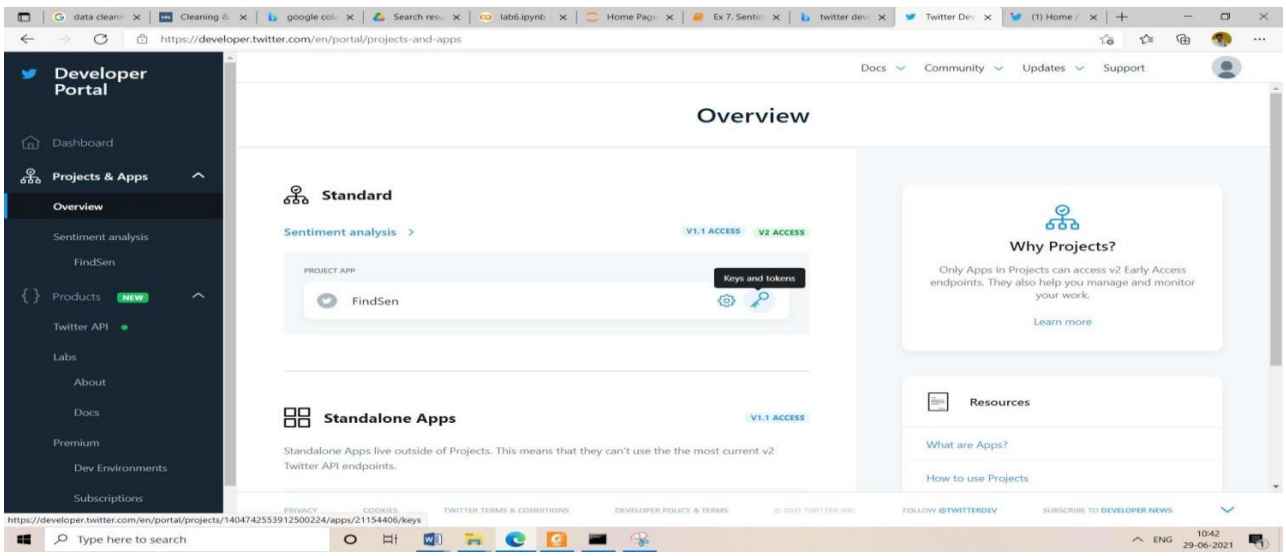
```
# We extract the tweet with more FAVs and more RTs:
fav_max                =
np.max(data['Likes'])rt_
max=np.max(data['RTs']
)
fav      =      data[data.Likes      ==
fav_max].index[0]rt
      =data[data.RTs==rt_max].index[
0]
# Max FAVs:
print("The          tweet          with          more          likes          is:
\n{}".format(data['Tweets'][fav]))print("Numberof likes: {}".format(fav_max))
print("{}characters.\n".format(data['len'][fav]))
# Max RTs:
print("The          tweet          with          more          retweets          is:
\n{}".format(data['Tweets'][rt]))print("Numberof retweets: {}".format(rt_max))
print("{}characters.\n".format(data['len'][rt]))
tlen  =  pd.Series(data=data['len'].values,  index=data['Date'])tfav  =
pd.Series(data=data['Likes'].values,
index=data['Date'])tret=pd.Series(data=data['RTs'].values,index=dat
a['Date'])
```

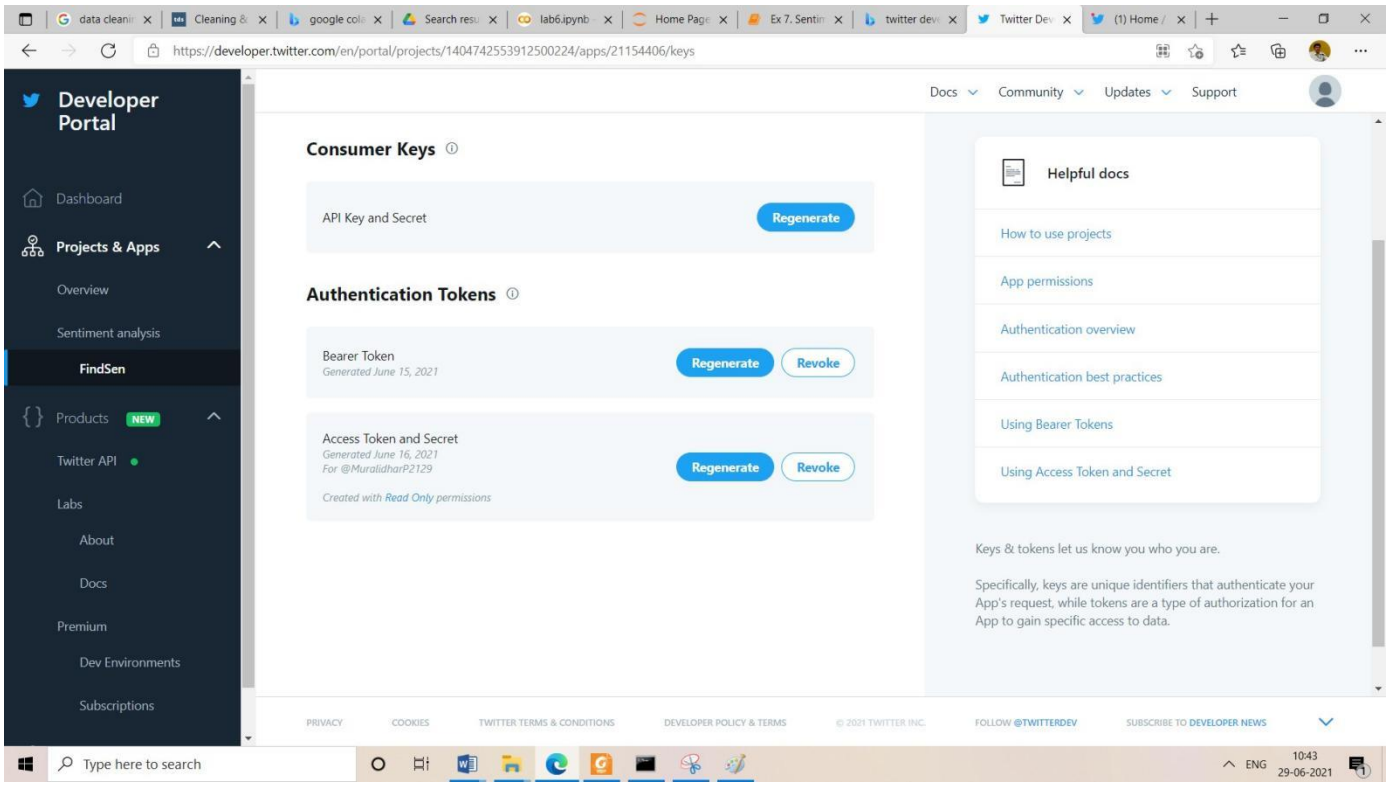
```
# Lenghts along time:
tlen.plot(figsize=(16,4),color='r');

tfav.plot(figsize=(16,4),label="Likes",legend=True)
tret.plot(figsize=(16,4),label="Retweets",legend=True);

# We obtain all possible sources:
sources=[]
forsourceindata['Source']:
    if source not in
        sources:sour
        ces.append(
        source)
# We print sources list:
print("Creationofcontentsources:")
forsourceinsources:
    print("*{}".format(source))
# We create a numpy vector mapped to labels:
percent=np.zeros(len(sources))
forsourceindata['Source']:
    forindexinrange(len(sources)):
        if source ==
            sources[index]
            :percent[index]
            +=1
            pass

percent/=100
# Pie chart:
pie_chart=pd.Series(percent,index=sources,name='Sources')
pie_chart.plot.pie(fontsize=11,autopct='%.2f',figsize=(6,6));
fromtextblobimportTextBlob
importre
def
    clean_t
    weet(tw
    eet):
'''
    Utility function to clean the text in a tweet by removinglinks
    and special characters using regex.
    '''
    return".join(re.sub("(@[A-Za-z0-9]+)|([^0-9A-Za-z\t])|(\w+:\/\/\S+)","",tweet).s
def
    analize_sentime
    nt(tweet):
  '''
    Utility function to classify the polarity of a
    tweetusing textblob.
    '''
    analysis=TextBlob(clean_tweet(tweet))
    ifanalysis.sentiment.polarity>0:
        return1
    elifanalysis.sentiment.polarity==0:
        return0
    else:
```

```
            return-1
# We create a column with the result of the analysis:
data['SA']=np.array([analize_sentiment(tweet)fortweetindata['Tweets']])
# We display the updated dataframe with the new column:
display(data.head(10))
# We construct lists with classified tweets:
pos_tweets = [ tweet for index, tweet in enumerate(data['Tweets']) if data['SA'][index] > 0
neu_tweets = [ tweet for index, tweet in enumerate(data['Tweets']) if data['SA'][index] == 0
neg_tweets = [ tweet for index, tweet in enumerate(data['Tweets']) if data['SA'][index] < 0
print(pos_tweets,neu_tweets, neg_tweets)
# We print percentages:
print("Percentage of positive tweets: {}%".format(len(pos_tweets)*100/len(data['Tweets'])))
print("Percentage of neutral tweets: {}%".format(len(neu_tweets)*100/len(data['Tweets'])))
print("Percentagedenegativetweets:{}%".format(len(negtweets)*100/len(_data['Tweets'])))_
```

Output:







Number of tweets extracted: 13.

13 recent tweets:

Youngster's are making Indian powerful

Indian Covid Vaccine became much popular in the world

Yes! The All-Party Parliamentary Group on Coronavirus (@AppgCoronavirus) is now discussing the needed scope & str… https://t.co/f3dxLJupGm (https://t.co/f3dxLJupGm)

How can this mob of aggressive anti-lockdown protesters think it’s OK to pursue & threaten BBC journalist… https://t.co/JM67tEAwpy (https://t.co/JM67tEAwpy)

Akshay Kumar's #BellBottom to release in theatres... After weeks of stimulation over its release plan, the makers h…https://t.co/AyJxA87Xx2(https://t.co/AyJxA87Xx2)

Than
k you
so so
much
@aks

hayk
umar
, I guess other will follow suit and start announcing dates and we can prepare…https://t.co/7rnQJOGjYn(https://t.co/7rnQJOGjYn)

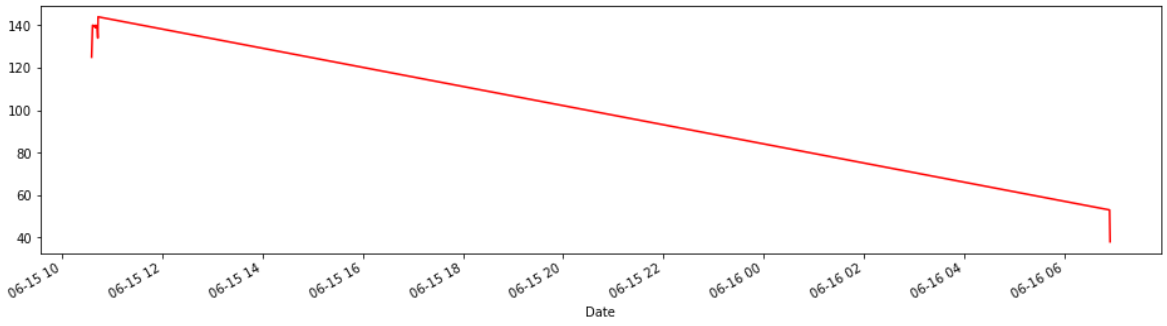
0	Youngster'saremakingIndianpowerful
1	IndianCovidVaccinebecamemuchpopularinth...
2	Yes!TheAll-PartyParliamentaryGrouponCoro...
3	How can this mob of aggressive anti-lockdown p...
4	Akshay Kumar's #BellBottom to release in theat...

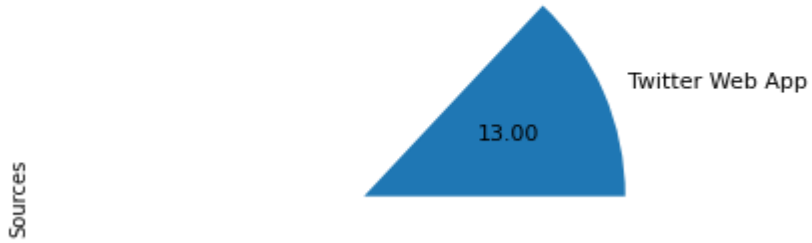
1405056277709946881
2021-06-16 06:54:55
{'hashtags': [], 'symbols': [], 'user_mentions': [], 'urls': []}
Twitter web App 0

The lenght's average in tweets: 123.92307692307692

The tweet with more likes is:
Youngster's are making
Indian powerfulNumber of
likes: 0
38 characters.

The tweet with more retweets is:
Youngster's are making
Indian powerfulNumber of
retweets: 0
38 characters.





Percentage of positive tweets:
46.15384615384615%Percentage of
neutral tweets: 46.15384615384615%
Percentage de negative tweets: 7.6923076923076925%