

***TABLE OF CONTENTS***

|  |  |  |
| --- | --- | --- |
| ***SL.***  ***NO.*** | ***PARTICULARS*** | ***PAGE***  ***NO*** |
| ***1*** | Vision and Mission of Institution | **3** |
| ***2*** | Vision and Mission of the Department | **4** |
| ***3*** | PEO, PSO, PO | **5** |
| ***4*** | Course Details   * Course Objectives * Syllabus * Course Outcomes | **6** |
| ***5*** | Python and Packages Installation | **9** |
| ***6*** | **Lab Syllabus Programs** | **12** |
| 1. Implement A\* Search algorithm. |
| 2. Implement AO\* Search algorithm. | **15** |
| 3. For a given set of training data examples stored in a .CSV file, implement and demonstrate  the Candidate-Elimination algorithm to output a description of the set of all hypotheses consistent with the training examples. | **19** |
| 4. Write a program to demonstrate the working of the decision tree based ID3 algorithm.  Use an appropriate data set for building the decision tree and apply this knowledge to classify a new sample. | **20** |
| 5. Build an Artificial Neural Network by implementing the Backpropagation algorithm and test the same using appropriate data sets. | **22** |
| 6. Write a program to implement the naïve Bayesian classifier for a sample training data set stored as a .CSV file. Compute the accuracy of the classifier, considering  few test data sets. | **24** |
| 7. Apply EM algorithm to cluster a set of data stored in a .CSV file. Use the same data set or clustering using k-Means algorithm. Compare the results of these two algorithms and  Comment on the quality of clustering. You can add Java/Python ML  library classes/API in the program | **26** |
| 8. Write a program to implement k-Nearest Neighbors algorithm to classify the iris data set. Print both correct and wrong predictions. Java/Python ML library classes can be  used for this problem. | **28** |
| 9. Implement the non-parametric Locally Weighted Regression algorithm in order to fit data points. Select appropriate data set for your experiment and draw graphs | **38** |
| ***7*** | Viva Question | **40** |
| ***8*** | Content Beyond Syllabus | **47** |

**Vision of the Institute**

***“To be a leader in imparting value based Technical Education and Research for the benefit of society”.***

**Mission of the Institute**

|  |  |
| --- | --- |
| **M1** | To Provide State-of-the-art Infrastructure Facilities |
| **M2** | To Implement modern Pedagogical Methods in delivering the Academic Programs with Experienced and Committed Faculty |
| **M3** | To Create a Vibrant Ambience that promotes Learning, Research, Invention and Innovation |
| **M4** | To Undertake Skill Development Programmes for Academic Institutions and Industries |
| **M5** | To Enhance Industry Institute Interaction through Collaborative Research and Consultancy |
| **M6** | To relentlessly Pursue Professional Excellence with Ethical and Moral Values |

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

***VISION***

***“Be a premier department in the field of Computer Science & Engineering to meet the technological challenges of the society”.***

***Mission of the Department***

|  |  |
| --- | --- |
| **M1** | To provide the State-of-the-art Infrastructure Facilities |
| **M2** | To provide the exposure to the latest tools in the area of Computer Hardware and Software. |
| **M3** | To strive for Academic excellence through Research in Computer Science and Engineering with creative Teaching –Learning pedagogy |
| **M4** | To establish Industry-Institute Interaction and make Students ready for the Industrial environment. |
| **M5** | To transform Students into Entrepreneurial, Technically competent, Socially responsible and Ethical computer science professional. |

**Program Educational Objectives (PEOs) of Department**

**After course completion CSE graduates will be able to:**

|  |  |
| --- | --- |
| **PEO 1** | Graduates acquire advanced knowledge of Computer Science Engineering and excel in leadership roles to serve the society. |
| **PEO 2** | Graduates of the program will apply Computer Science and Engineering and excel in leadership computer science professional. |
| **PEO 3** | Graduates adapt Value-Based Proficiency in solving real time problems. |

**Program Specific Outcomes (PSO)**

|  |  |
| --- | --- |
| **PSO 1** | **Professional Skills:** Ability of applying the Computing Concepts, Data Structure, Computer Networks and Suitable Algorithm. |
| **PSO 2** | **Software Skills:** Ability to build Software Engineering System with Development Life Cycle by using analytical knowledge in Computer Science and Engineering and applying modern methodologies |

**Program Outcomes (POs)**

|  |  |
| --- | --- |
| **PO1** | **Engineering knowledge**: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems. |
| **PO2** | **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. |
| **PO3** | **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. |
| **PO4** | **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 |
| **PO5** | **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. |
| **PO6** | **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. |
| **PO7** | **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. |
| **PO8** | **Ethics**: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice. |
| **PO9** | **Individual and team work**: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings. |
| **PO10** | **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. |
| **PO11** | **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. |
| **PO12** | **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. |

**Course Details**

**ARTIFICIAL INTELLIGENT AND MACHINE LEARNING LABORATORY**

[As per Choice Based Credit System (CBCS) scheme]

**SEMESTER – VII Subject Code 18CSL76**

**Course objectives:**

This course will enable students to

1 Implement and demonstrate AI and ML algorithms .

2. Evaluate different algorithms.

**Description (If any):**

1. The programs can be implemented in either JAVA or Python.

2. For Problems 1 to 6 and 10, programs are to be developed without using the built-in classes or APIs

of Java/Python.

3. Data sets can be taken from standard repositories (https://archive.ics.uci.edu/ml/datasets.html) or constructed by the students.

**Lab Experiments:**

|  |  |
| --- | --- |
| **1** | Implement A\* Search algorithm. |
| **2** | Implement AO\* Search algorithm |
| **3** | For a given set of training data examples stored in a .CSV file, implement and demonstrate the Candidate-Elimination algorithm to output a description of the set of all hypotheses consistent with the training examples |
| **4** | Write a program to demonstrate the working of the decision tree based ID3 algorithm. Use an appropriate data set for building the decision tree and apply this knowledge to classify a new sample. |
| **5** | Build an Artificial Neural Network by implementing the Backpropagation algorithm and test the same using appropriate data sets. |
| **6** | Write a program to implement the naïve Bayesian classifier for a sample training data set stored as a .CSV file. Compute the accuracy of the classifier, considering few test data sets. |
| **7** | Apply EM algorithm to cluster a set of data stored in a .CSV file. Use the same data set for clustering using k-Means algorithm. Compare the results of these two algorithms and comment on the quality of clustering. You can add Java/Python ML library classes/API in the program |
| **8** | Write a program to implement k-Nearest Neighbour algorithm to classify the iris data set. Print both correct and wrong predictions. Java/Python ML library classes can be used for this problem. |
| **9** | Implement the non-parametric Locally Weighted Regression algorithm in order to fit data points. Select appropriate data set for your experiment and draw graphs |

**Course outcomes:**

The students should be able to:

|  |  |
| --- | --- |
| **CO No.** | **Statement** |
| C476.1 | Make use of BFS Control strategy to implement the A\* and AO\* algorithms. |
| C476.2 | Build supervised classification models to predict the outcome. |
| C476.3 | Analyze & compare the unsupervised machine learning algorithms. |
| C476.4 | Identify and apply Machine Learning algorithms to solve real world problems. |

**Conduction of Practical Examination:**

All laboratory experiments are to be included for practical examination.

Students are allowed to pick one experiment from the lot.

Strictly follow the instructions as printed on the cover page of answer script

Marks distribution: Procedure + Conduction + Viva:**15 + 70 +15 (100)**

**Strength of CO Mapping to PO/PSOs with Justification**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **COs** | **PO1** | **PO2** | **PO3** | **PO4** | **PO5** | **PO6** | **PO7** | **PO8** | **PO9** | **PO10** | **PO11** | **PO12** | **PSO1** | **PSO2** |
| **C476.1** | 2 | - | 1 | - | - | - | - | 1 | - | - | - | 1 | 2 | - |
| **C476.2** | 2 | 1 | 2 | 1 | 3 | 1 | - | 2 | 1 | 1 | - | 2 | 2 | 1- |
| **C476.3** | 2 | 1 | 2 | 1 | 3 | 1 | - | 1 | 1 | 1 | - | 2 | 2 | 1 |
| **Average** | **2** | **1** | **1.67** | **1** | **3** | **1** | **-** | **1.3** | **1** | **1** | **-** | **1.67** | **2** | **1** |

**Jupyter Notebook**

# Install Jupyter Notebook in Windows

Jupyter Notebook is an open-source web application that allows to create and share documents that contain live code, equations, visualizations, and narrative text. Uses include data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more.

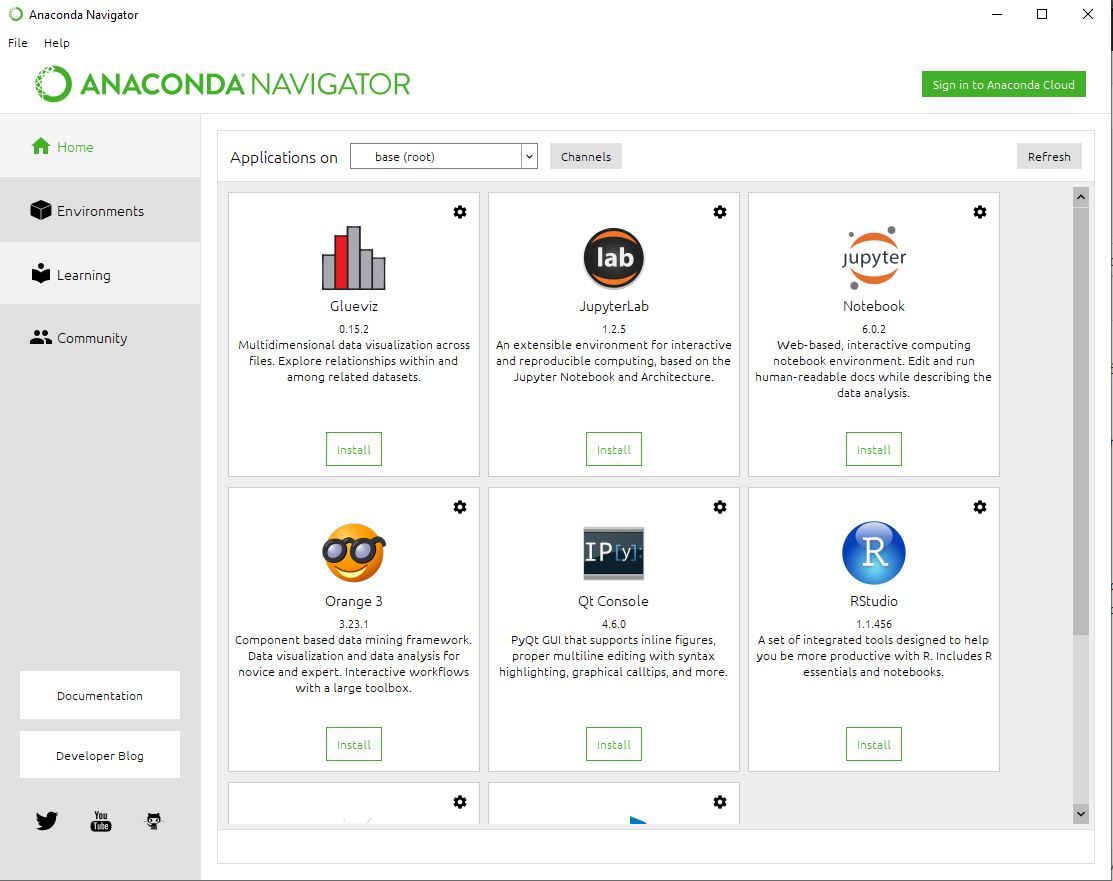
Jupyter has support for over 40 different programming languages and [Python](https://www.geeksforgeeks.org/python-language-introduction/) is one of them. Python is a requirement (Python 3.3 or greater, or Python 2.7) for installing the Jupyter Notebook itself.

Jupyter Notebook can be installed by using either of the two ways described below:

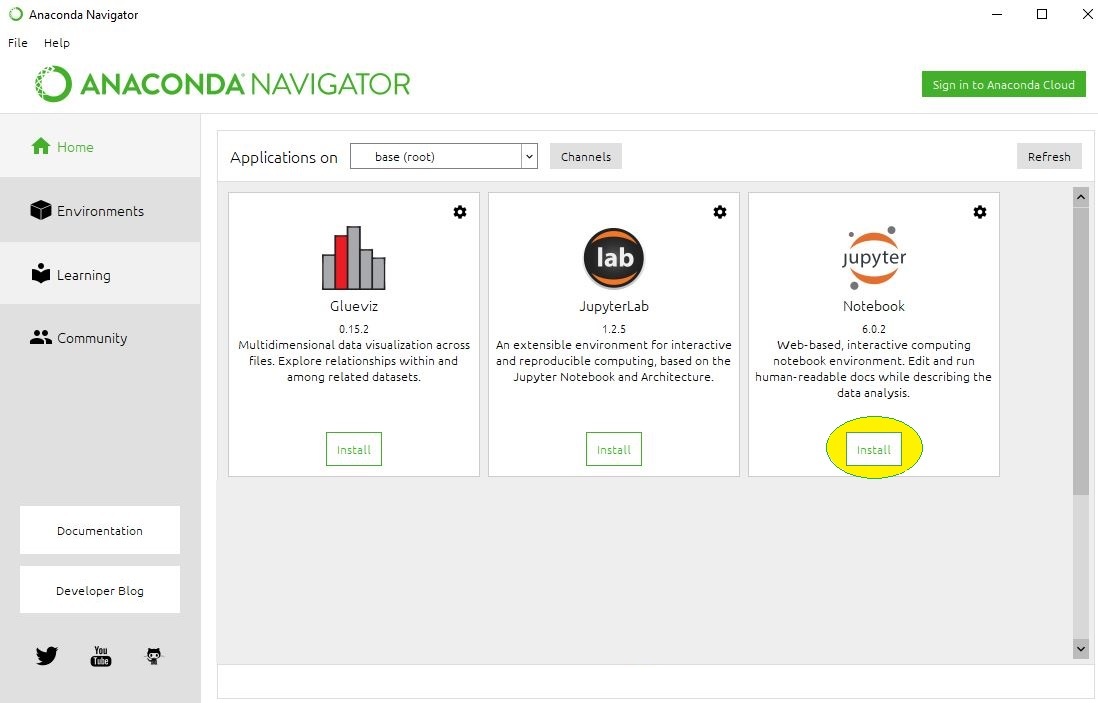
* **Using Anaconda:**  
  Install Python and Jupyter using the Anaconda Distribution, which includes Python, the Jupyter Notebook, and other commonly used packages for scientific computing and data science.
* **Using PIP:**  
  Install Jupyter using the **PIP package manager** used to install and manage software packages/libraries written in Python. Installing Jupyter Notebook using Anaconda

Anaconda is an open-source software that contains Jupyter, spyder, etc that are used for large data processing, data analytics, heavy scientific computing. Anaconda works for R and python programming language. Spyder(sub-application of Anaconda) is used for python. Opencv for python will work in spyder. Package versions are managed by the package management system called conda.

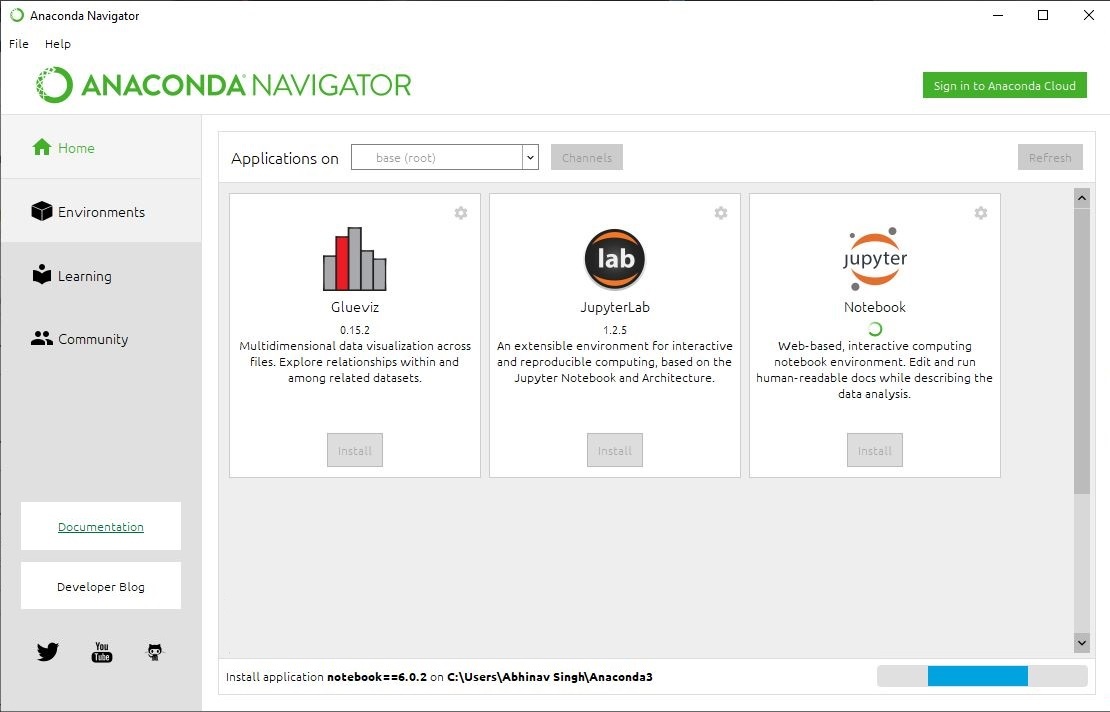
**Launch Anaconda Navigator:**



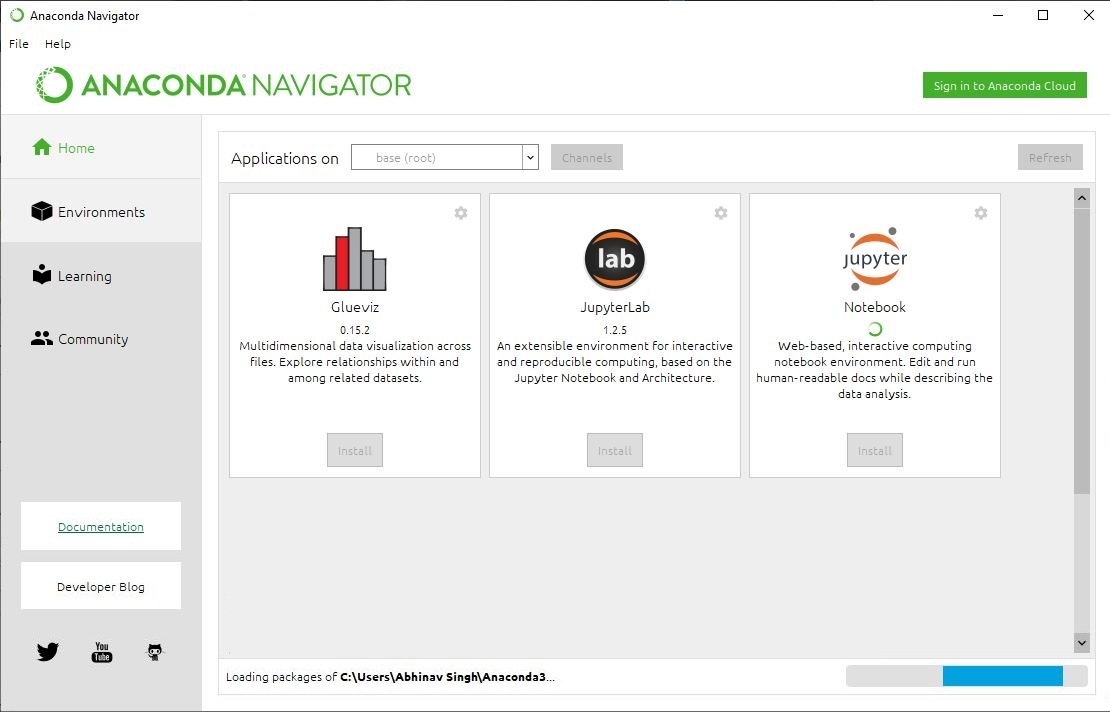
**Click on the Install Jupyter Notebook Button:**



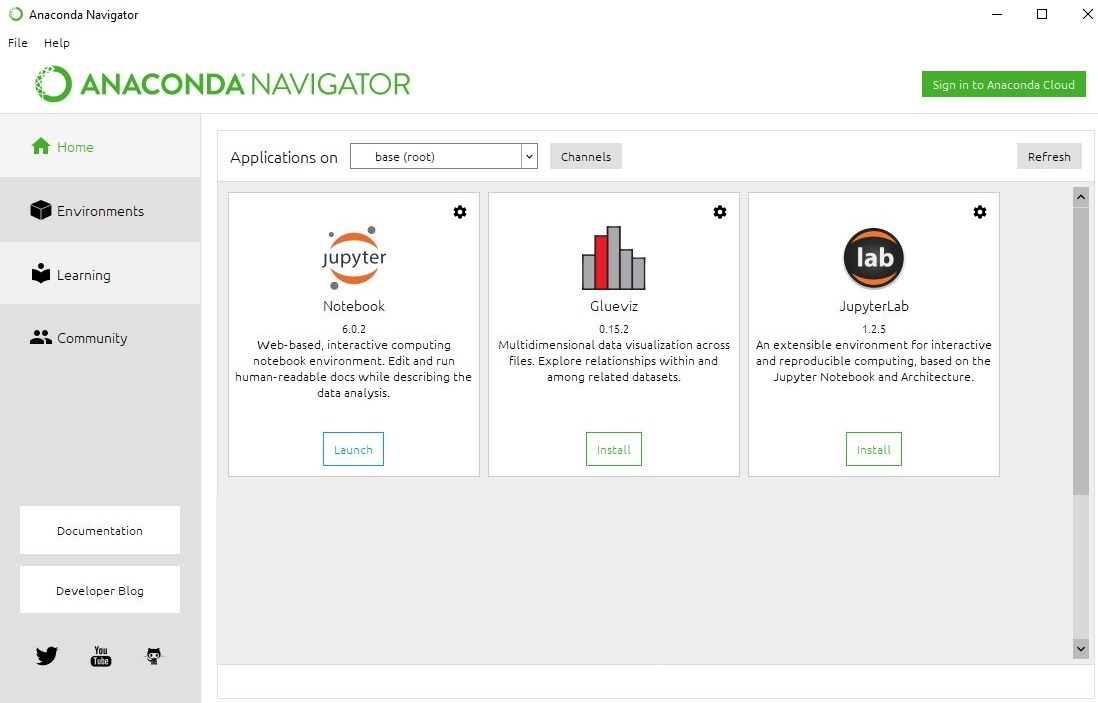
**Beginning the Installation**



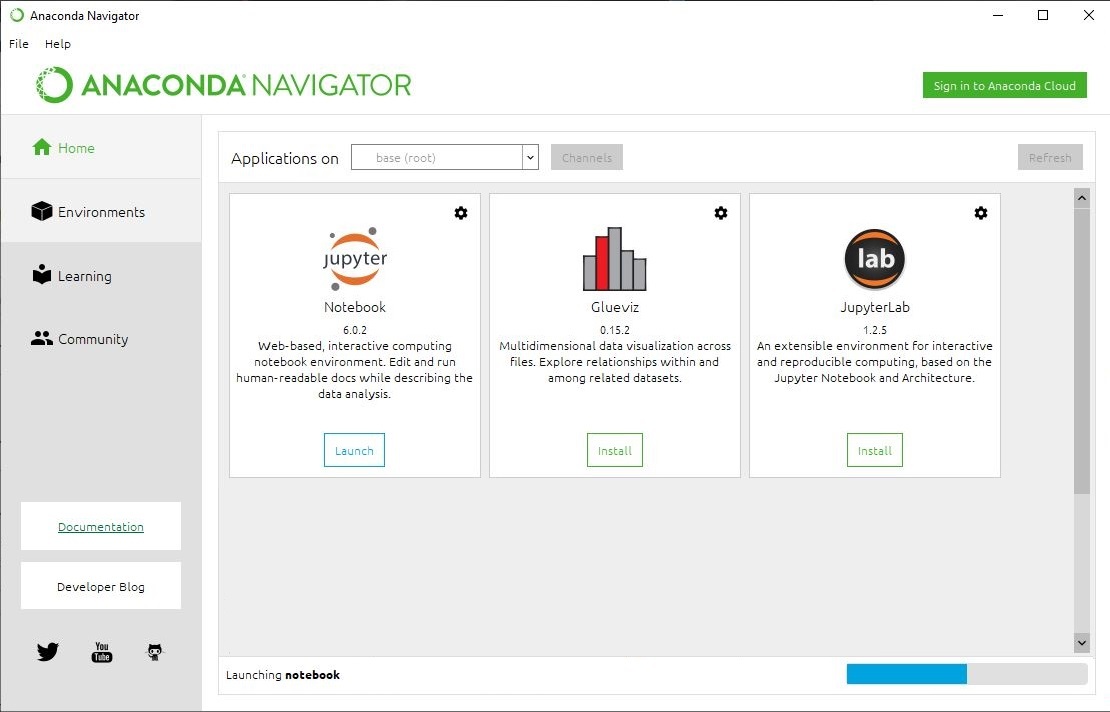
**Loading Packages:**

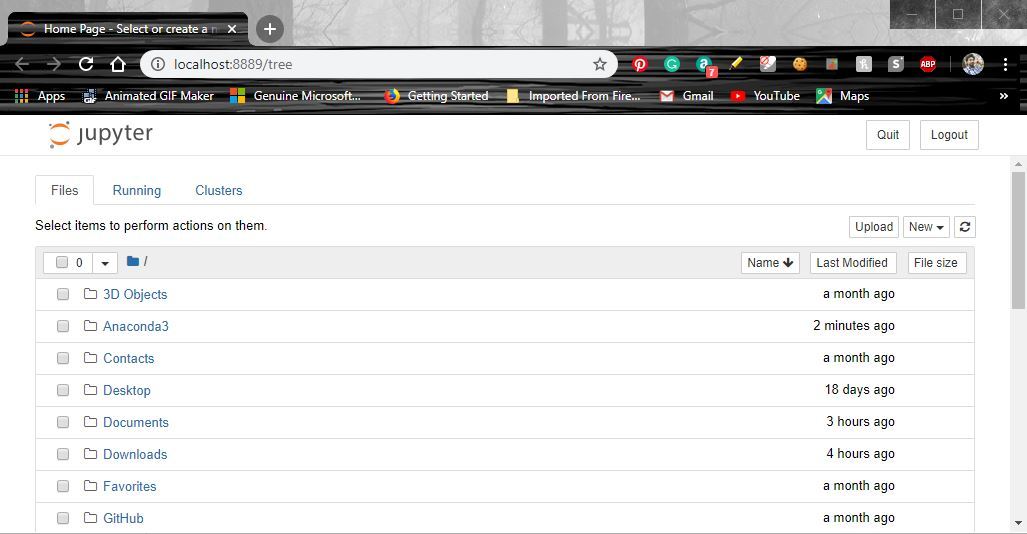


**Finished Installation:**



**Launching Jupyter:**





**Program-1=A\***

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

#ditance of starting node from itself is zero

g[start\_node] = 0

#start\_node is root node i.e it has no parent nodes

#so start\_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

#for simplicity we ll consider heuristic distances given

#and this function returns heuristic distance for all nodes

def heuristic(n):

H\_dist = {

'A': 11,

'B': 6,

'C': 99,

'D': 1,

'E': 7,

'G': 0,

}

return H\_dist[n]

#Describe your graph here

Graph\_nodes = {

'A': [('B', 2), ('E', 3)],

'B': [('C', 1),('G', 9)],

'C': None,

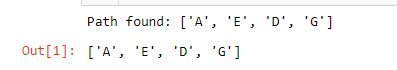
'E': [('D', 6)],

'D': [('G', 1)],

}

aStarAlgo('A', 'G')

Output:



**Program-2 AO\***

**class Graph:**

def \_\_init\_\_(self, graph, heuristicNodeList, startNode): #instantiate graph object with graph topology, heuristic values, start node

self.graph = graph

self.H=heuristicNodeList

self.start=startNode

self.parent={}

self.status={}

self.solutionGraph={}

def applyAOStar(self): # starts a recursive AO\* algorithm

self.aoStar(self.start, False)

def getNeighbors(self, v): # gets the Neighbors of a given node

return self.graph.get(v,'')

def getStatus(self,v): # return the status of a given node

return self.status.get(v,0)

def setStatus(self,v, val): # set the status of a given node

self.status[v]=val

def getHeuristicNodeValue(self, n):

return self.H.get(n,0) # always return the heuristic value of a given node

def setHeuristicNodeValue(self, n, value):

self.H[n]=value # set the revised heuristic value of a given node

def printSolution(self):

print("FOR GRAPH SOLUTION, TRAVERSE THE GRAPH FROM THE START NODE:",self.start)

print("------------------------------------------------------------")

print(self.solutionGraph)

print("------------------------------------------------------------")

def computeMinimumCostChildNodes(self, v): # Computes the Minimum Cost of child nodes of a given node v

minimumCost=0

costToChildNodeListDict={}

costToChildNodeListDict[minimumCost]=[]

flag=True

for nodeInfoTupleList in self.getNeighbors(v): # iterate over all the set of child node/s

cost=0

nodeList=[]

for c, weight in nodeInfoTupleList:

cost=cost+self.getHeuristicNodeValue(c)+weight

nodeList.append(c)

if flag==True: # initialize Minimum Cost with the cost of first set of child node/s

minimumCost=cost

costToChildNodeListDict[minimumCost]=nodeList # set the Minimum Cost child node/s

flag=False

else: # checking the Minimum Cost nodes with the current Minimum Cost

if minimumCost>cost:

minimumCost=cost

costToChildNodeListDict[minimumCost]=nodeList # set the Minimum Cost child node/s

return minimumCost, costToChildNodeListDict[minimumCost] # return Minimum Cost and Minimum Cost child node/s

def aoStar(self, v, backTracking): # AO\* algorithm for a start node and backTracking status flag

print("HEURISTIC VALUES :", self.H)

print("SOLUTION GRAPH :", self.solutionGraph)

print("PROCESSING NODE :", v)

print("-----------------------------------------------------------------------------------------")

if self.getStatus(v) >= 0: # if status node v >= 0, compute Minimum Cost nodes of v

minimumCost, childNodeList = self.computeMinimumCostChildNodes(v)

print(minimumCost, childNodeList)

self.setHeuristicNodeValue(v, minimumCost)

self.setStatus(v,len(childNodeList))

solved=True # check the Minimum Cost nodes of v are solved

for childNode in childNodeList:

self.parent[childNode]=v

if self.getStatus(childNode)!=-1:

solved=solved & False

if solved==True: # if the Minimum Cost nodes of v are solved, set the current node status as solved(-1)

self.setStatus(v,-1)

self.solutionGraph[v]=childNodeList # update the solution graph with the solved nodes which may be a part of solution

if v!=self.start: # check the current node is the start node for backtracking the current node value

self.aoStar(self.parent[v], True) # backtracking the current node value with backtracking status set to true

if backTracking==False: # check the current call is not for backtracking

for childNode in childNodeList: # for each Minimum Cost child node

self.setStatus(childNode,0) # set the status of child node to 0(needs exploration)

self.aoStar(childNode, False) # Minimum Cost child node is further explored with backtracking status as false

print ("Graph - 1")

h1 = {'A': 1, 'B': 6, 'C': 2, 'D': 12, 'E': 2, 'F': 1, 'G': 5, 'H': 7, 'I': 7, 'J': 1}

graph1 = {

'A': [[('B', 1), ('C', 1)], [('D', 1)]],

'B': [[('G', 1)], [('H', 1)]],

'C': [[('J', 1)]],

'D': [[('E', 1), ('F', 1)]],

'G': [[('I', 1)]]

}

G1= Graph(graph1, h1, 'A')

G1.applyAOStar()

G1.printSolution()

Output:

OUTPUT:

Graph - 1

HEURISTIC VALUES : {'A': 1, 'B': 6, 'C': 2, 'D': 12, 'E': 2, 'F': 1, 'G': 5, 'H': 7, 'I': 7, 'J': 1}

SOLUTION GRAPH : {}

PROCESSING NODE : A

-----------------------------------------------------------------------------------------

10 ['B', 'C']

HEURISTIC VALUES : {'A': 10, 'B': 6, 'C': 2, 'D': 12, 'E': 2, 'F': 1, 'G': 5, 'H': 7, 'I': 7, 'J': 1}

SOLUTION GRAPH : {}

PROCESSING NODE : B

-----------------------------------------------------------------------------------------

6 ['G']

HEURISTIC VALUES : {'A': 10, 'B': 6, 'C': 2, 'D': 12, 'E': 2, 'F': 1, 'G': 5, 'H': 7, 'I': 7, 'J': 1}

SOLUTION GRAPH : {}

PROCESSING NODE : A

-----------------------------------------------------------------------------------------

10 ['B', 'C']

HEURISTIC VALUES : {'A': 10, 'B': 6, 'C': 2, 'D': 12, 'E': 2, 'F': 1, 'G': 5, 'H': 7, 'I': 7, 'J': 1}

SOLUTION GRAPH : {}

PROCESSING NODE : G

-----------------------------------------------------------------------------------------

8 ['I']

HEURISTIC VALUES : {'A': 10, 'B': 6, 'C': 2, 'D': 12, 'E': 2, 'F': 1, 'G': 8, 'H': 7, 'I': 7, 'J': 1}

SOLUTION GRAPH : {}

PROCESSING NODE : B

-----------------------------------------------------------------------------------------

8 ['H']

HEURISTIC VALUES : {'A': 10, 'B': 8, 'C': 2, 'D': 12, 'E': 2, 'F': 1, 'G': 8, 'H': 7, 'I': 7, 'J': 1}

SOLUTION GRAPH : {}

PROCESSING NODE : A

-----------------------------------------------------------------------------------------

12 ['B', 'C']

HEURISTIC VALUES : {'A': 12, 'B': 8, 'C': 2, 'D': 12, 'E': 2, 'F': 1, 'G': 8, 'H': 7, 'I': 7, 'J': 1}

SOLUTION GRAPH : {}

PROCESSING NODE : I

-----------------------------------------------------------------------------------------

0 []

HEURISTIC VALUES : {'A': 12, 'B': 8, 'C': 2, 'D': 12, 'E': 2, 'F': 1, 'G': 8, 'H': 7, 'I': 0, 'J': 1}

SOLUTION GRAPH : {'I': []}

PROCESSING NODE : G

-----------------------------------------------------------------------------------------

1 ['I']

HEURISTIC VALUES : {'A': 12, 'B': 8, 'C': 2, 'D': 12, 'E': 2, 'F': 1, 'G': 1, 'H': 7, 'I': 0, 'J': 1}

SOLUTION GRAPH : {'I': [], 'G': ['I']}

PROCESSING NODE : B

-----------------------------------------------------------------------------------------

2 ['G']

HEURISTIC VALUES : {'A': 12, 'B': 2, 'C': 2, 'D': 12, 'E': 2, 'F': 1, 'G': 1, 'H': 7, 'I': 0, 'J': 1}

SOLUTION GRAPH : {'I': [], 'G': ['I'], 'B': ['G']}

PROCESSING NODE : A

-----------------------------------------------------------------------------------------

6 ['B', 'C']

HEURISTIC VALUES : {'A': 6, 'B': 2, 'C': 2, 'D': 12, 'E': 2, 'F': 1, 'G': 1, 'H': 7, 'I': 0, 'J': 1}

SOLUTION GRAPH : {'I': [], 'G': ['I'], 'B': ['G']}

PROCESSING NODE : C

-----------------------------------------------------------------------------------------

2 ['J']

HEURISTIC VALUES : {'A': 6, 'B': 2, 'C': 2, 'D': 12, 'E': 2, 'F': 1, 'G': 1, 'H': 7, 'I': 0, 'J': 1}

SOLUTION GRAPH : {'I': [], 'G': ['I'], 'B': ['G']}

PROCESSING NODE : A

-----------------------------------------------------------------------------------------

6 ['B', 'C']

HEURISTIC VALUES : {'A': 6, 'B': 2, 'C': 2, 'D': 12, 'E': 2, 'F': 1, 'G': 1, 'H': 7, 'I': 0, 'J': 1}

SOLUTION GRAPH : {'I': [], 'G': ['I'], 'B': ['G']}

PROCESSING NODE : J

-----------------------------------------------------------------------------------------

0 []

HEURISTIC VALUES : {'A': 6, 'B': 2, 'C': 2, 'D': 12, 'E': 2, 'F': 1, 'G': 1, 'H': 7, 'I': 0, 'J': 0}

SOLUTION GRAPH : {'I': [], 'G': ['I'], 'B': ['G'], 'J': []}

PROCESSING NODE : C

-----------------------------------------------------------------------------------------

1 ['J']

HEURISTIC VALUES : {'A': 6, 'B': 2, 'C': 1, 'D': 12, 'E': 2, 'F': 1, 'G': 1, 'H': 7, 'I': 0, 'J': 0}

SOLUTION GRAPH : {'I': [], 'G': ['I'], 'B': ['G'], 'J': [], 'C': ['J']}

PROCESSING NODE : A

-----------------------------------------------------------------------------------------

5 ['B', 'C']

FOR GRAPH SOLUTION, TRAVERSE THE GRAPH FROM THE START NODE: A

------------------------------------------------------------

{'I': [], 'G': ['I'], 'B': ['G'], 'J': [], 'C': ['J'], 'A': ['B', 'C']}

Program-3-CEA

import csv

hypo=[]

data=[]

temp=[]

gen=[]

sef=[]

with open('data21\_shape.csv') as csv\_file:

fd = csv.reader(csv\_file)

print("\nThe given training examples are:")

for line in fd:

print(line)

temp.append(line)

if line[-1]== "Y":

data.append(line)

print("\nThe positive examples are: Enjoy swimming");

for line in data:

print(line);

row= len(data);

col=len(data[0]);

print("\nThe final specific output......................");

for j in range(col-1):

hypo.append(data[0][j]);

for i in range(row):

for j in range(col-1):

#print('hypo[j]',hypo[j])

#print('data[i][j]',data[i][j])

if (hypo[j]!=data[i][j]):

hypo[j]='?'

print(hypo)

print("\nThe final Genralize output..................");

row=len(temp)

for i in range(row):

if temp[i][-1]=="N":

for j in range(col-1):

if temp[i][j] !=hypo[j] and hypo[j] != '?':

gen = ['?','?','?','?','?','?']

gen[j]=hypo[j]

sef.append(gen)

print(sef)

**Program-4-ID3**

import csv

import pprint

from math import \*

lines=list(csv.reader(open('data3.csv','r')))

data=lines.pop(0)

print(data)

print()

print(lines)

def entropy(pos,neg):

if pos==0 or neg==0:

return 0

tot=pos+neg

return -pos/tot\*log(pos/tot,2)-neg/tot\*log(neg/tot,2)

def gain(lines,attr,pos,neg):

d,E,acu={},entropy(pos,neg),0

for i in lines:

if i[attr] not in d:

d[i[attr]]={}

d[i[attr]][i[-1]]=1+d[i[attr]].get(i[-1],0)

for i in d:

tot=d[i].get('yes',0)+d[i].get('no',0)

acu+=tot/(pos+neg)\*entropy(d[i].get('yes',0),d[i].get('no',0))

return E-acu

def build(lines,data):

pos=len([x for x in lines if x[-1]=='yes'])

sz=len(lines[0])-1

neg=len(lines)-pos

if neg==0 or pos==0:

return 'yes' if neg==0 else 'no'

root=max([[gain(lines,i,pos,neg),i]for i in range(sz)])[1]

fin,res={},{}

uniq\_attr=set([x[root] for x in lines])

print(">>>",uniq\_attr)

for i in uniq\_attr:

res[i]=build([x[:root]+x[root+1:] for x in lines if x[root]==i],data[:root]+data[root+1:])

fin[data[root]]=res

return fin

tree=build(lines,data)

pprint.pprint(tree)

def classify(instance,tree,default=None):

attribute=next(iter(tree))

if instance[attribute] in tree[attribute].keys():

result=tree[attribute][instance[attribute]]

if isinstance(result,dict):

return classify(instance,result)

else:

return result

else:

return default

import pandas as pd

df\_new=pd.read\_csv('data3\_test.csv')

df\_new['predicted']=df\_new.apply(classify,axis=1 ,args=(tree,'?'))

print(df\_new)

**Output**

['Outlook', 'Temperature', 'Humidity', 'Wind', 'Target']

[['sunny', 'hot', 'high', 'weak', 'no'], ['sunny', 'hot', 'high', 'strong', 'no'], ['overcast', 'hot', 'high', 'weak', 'yes'], ['rain', 'mild', 'high', 'weak', 'yes'], ['rain', 'cool', 'normal', 'weak', 'yes'], ['rain', 'cool', 'normal', 'strong', 'no'], ['overcast', 'cool', 'normal', 'strong', 'yes'], ['sunny', 'mild', 'high', 'weak', 'no'], ['sunny', 'cool', 'normal', 'weak', 'yes'], ['rain', 'mild', 'normal', 'weak', 'yes'], ['sunny', 'mild', 'normal', 'strong', 'yes'], ['overcast', 'mild', 'high', 'strong', 'yes'], ['overcast', 'hot', 'normal', 'weak', 'yes'], ['rain', 'mild', 'high', 'strong', 'no']]

>>> {'overcast', 'sunny', 'rain'}

>>> {'normal', 'high'}

>>> {'weak', 'strong'}

{'Outlook': {'overcast': 'yes',

'rain': {'Wind': {'strong': 'no', 'weak': 'yes'}},

'sunny': {'Humidity': {'high': 'no', 'normal': 'yes'}}}}

Outlook Temperature Humidity Wind predicted

0 rain mild normal strong no



1

​

**Program-5**

import numpy as np

# X = (hours sleeping, hours studying), y = score on test

X = np.array(([2, 9], [1, 5], [3, 6]), dtype=float)

print('\*\*\*\*',np.amax(X, axis=0))

y = np.array(([92], [86], [89]), dtype=float)

# scale units

X = X/np.amax(X, axis=0) # maximum of X array

y = y/100 # max test score is 100

print('=======X',X)

print('++++++y',y)

class Neural\_Network(object):

def \_\_init\_\_(self):

#parameters

self.inputSize = 2

self.outputSize = 1

self.hiddenSize = 3

#weights

self.W1 = np.random.randn(self.inputSize, self.hiddenSize) # (3x2) weight matrix from input to hidden layer

self.W2 = np.random.randn(self.hiddenSize, self.outputSize) # (3x1) weight matrix from hidden to output layer

def forward(self, X):

#forward propagation through our network

self.z = np.dot(X, self.W1) # dot product of X (input) and first set of 3x2 weights

self.z2 = self.sigmoid(self.z) # activation function

self.z3 = np.dot(self.z2, self.W2) # dot product of hidden layer (z2) and second set of 3x1 weights

o = self.sigmoid(self.z3) # final activation function

return o

def sigmoid(self, s):

# activation function

return 1/(1+np.exp(-s))

def sigmoidPrime(self, s):

#derivative of sigmoid

return s \* (1 - s)

def backward(self, X, y, o):

# backward propgate through the network

self.o\_error = y - o # error in output

self.o\_delta = self.o\_error\*self.sigmoidPrime(o) # applying derivative of sigmoid to error

self.z2\_error = self.o\_delta.dot(self.W2.T) # z2 error: how much our hidden layer weights contributed to output error

self.z2\_delta = self.z2\_error\*self.sigmoidPrime(self.z2) # applying derivative of sigmoid to z2 error

self.W1 += l\_rate\*X.T.dot(self.z2\_delta) # adjusting first set (input --> hidden) weights

self.W2 += l\_rate\*self.z2.T.dot(self.o\_delta) # adjusting second set (hidden --> output) weights

def train (self, X, y):

o = self.forward(X)

self.backward(X, y, o)

NN = Neural\_Network()

l\_rate=0.1

for i in range(10): # trains the NN 1,000 times

print("Epoch:",i)

print("l\_rate",l\_rate)

print ("Input: \n",str(X))

print ("Actual Output: \n",str(y) )

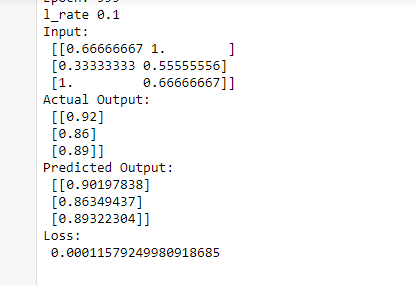
print ("Predicted Output: \n",str(NN.forward(X)) )

print ("Loss: \n",str(np.mean(np.square(y - NN.forward(X))))) # mean sum squared loss

print ("\n")

NN.train(X, y)

**OutPUT-**



**Program 6: Naïve Bayes Classifier**

import math

import statistics as st

from sklearn.model\_selection import train\_test\_split

import pandas as pd

def summarizeByClass(x\_tr,y\_tr):

separated = {} # Create a dictionary with labels as keys 1 and 0

for i in range(len(x\_train)):

x, y = x\_tr[i],y\_tr[i]

if (y not in separated):

separated[y] = []

separated[y].append(x)

summary = {} # to store mean and std of +ve and -ve instances

for lbl, subset in separated.items():

summary[lbl] = [ (st.mean(attribute), st.stdev(attribute))

for attribute in zip(\*subset)]; #zip(\*res) transposes a matrix (2-d array/list)

return summary

#For continuous attributes p is estimated using Gaussion distribution

def estimateProbability(x, mean, stdev):

exponent = math.exp(-(math.pow(x-mean,2)/(2\*math.pow(stdev,2))))

return (1 / (math.sqrt(2\*math.pi) \* stdev)) \* exponent

def predict(summaries, testVector):

bestLabel, bestProb = None, -1

p = {}

for lbl, mean\_std in summaries.items():

#class and attribute information as mean and sd

p[lbl] = 1

for i in range(len(mean\_std)):

mean, stdev = mean\_std[i]

x = testVector[i]

p[lbl] \*= estimateProbability(x, mean, stdev);

#assigns that class which has he highest prob

if bestLabel is None or p[lbl] > bestProb:

bestProb = p[lbl]

bestLabel = lbl

return bestLabel

def do\_classification\_compute\_accuracy(summaries, test\_x, test\_y):

correct = 0

for i in range(len(test\_x)):

result = predict(summaries, test\_x[i])

if result == test\_y[i]:

correct = correct + 1

accuracy = (correct/float(len(test\_x))) \* 100.0

return accuracy

# Main program

df=pd.read\_csv('ConceptLearning.csv',header=None)

cols = [0,1,2,3]

df\_x = df[df.columns[cols]]

df\_y = df[df.columns[4]]

X = df\_x.values.tolist()

Y = df\_y.values.tolist()

x\_train, x\_test, y\_train, y\_test = train\_test\_split(X,Y,test\_size=0.10)

print('Dataset loaded...')

print('Total instances available :',len(X))

print('Total attributes present :',len(X[0]))

print("First Five instances of dataset:")

for i in range(5):

print(i+1 , ':' , X[i])

print('\nDataset is split into training and testing set.')

print('Training examples = {0} \nTesting examples = {1}'.format

(len(x\_train), len(x\_test)))

summaries = summarizeByClass(x\_train,y\_train);

accuracy = do\_classification\_compute\_accuracy(summaries,x\_test,y\_test)

print('\nAccuracy of the Naive Baysian Classifier is :', accuracy)

**Output:**

Dataset loaded...

Total instances available : 16

Total attributes present : 4

First Five instances of dataset:

1 : [1, 1, 1, 1]

2 : [1, 1, 1, 2]

3 : [2, 1, 1, 2]

4 : [3, 2, 1, 1]

5 : [3, 3, 2, 1]

Dataset is split into training and testing set.

Training examples = 14

Testing examples = 2

Accuracy of the Naive Baysian Classifier is : 100.0

**Program 7: EM and Kmeans**

import matplotlib.pyplot as plt

from sklearn import datasets

from sklearn.cluster import KMeans

import pandas as pd

import numpy as np

# import some data to play with

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 = ['Targets']

# Build the K Means Model

model = KMeans(n\_clusters=4)

model.fit(X) # model.labels\_ : Gives cluster no for which samples belongs to

# # Visualise the clustering results

plt.figure(figsize=(14,14))

colormap = np.array(['red', 'lime', 'black', 'blue'])

# Plot the Original Classifications using Petal features

plt.subplot(2, 2, 1)

plt.scatter(X.Petal\_Length, X.Petal\_Width, c=colormap[y.Targets], s=40)

plt.title('Real Clusters')

plt.xlabel('Petal Length')

plt.ylabel('Petal Width')

# Plot the Models Classifications

plt.subplot(2, 2, 2)

plt.scatter(X.Petal\_Length, X.Petal\_Width, c=colormap[model.labels\_], s=40)

plt.title('K-Means Clustering')

plt.xlabel('Petal Length')

plt.ylabel('Petal Width')

# General EM for GMM

from sklearn import preprocessing

# transform your data such that its distribution will have a

# mean value 0 and standard deviation of 1.

scaler = preprocessing.StandardScaler()

scaler.fit(X)

xsa = scaler.transform(X)

xs = pd.DataFrame(xsa, columns = X.columns)

from sklearn.mixture import GaussianMixture

gmm = GaussianMixture(n\_components=4)

gmm.fit(xs)

gmm\_y = gmm.predict(xs)

plt.subplot(2, 2, 3)

plt.scatter(X.Petal\_Length, X.Petal\_Width, c=colormap[gmm\_y], s=40)

plt.title('GMM Clustering')

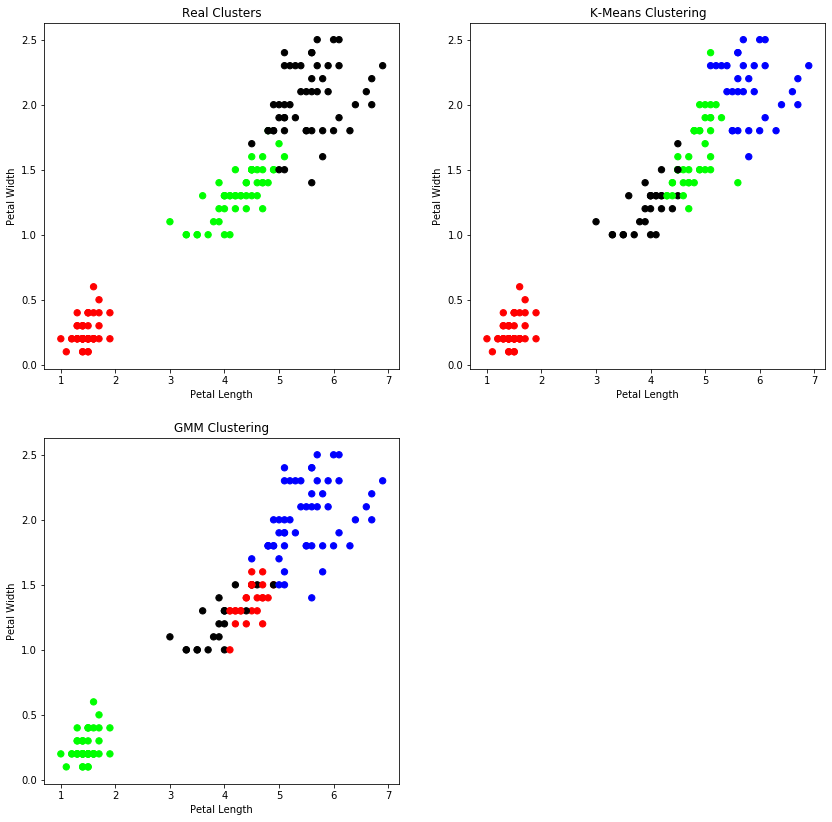
plt.xlabel('Petal Length')

plt.ylabel('Petal Width')

print('Observation: The GMM using EM algorithm based clustering matched the true labels more closely than the Kmeans.')

**Output:**

Observation: The GMM using EM algorithm based clustering matched the true labels more closely than the Kmeans.



**Program 8: KNN**

from sklearn.datasets import load\_iris

from sklearn.neighbors import KNeighborsClassifier

import numpy as np

from sklearn.model\_selection import train\_test\_split

iris\_dataset=load\_iris()

print("\nIRIS FEATURES \TARGET NAMES: \n",iris\_dataset.target\_names)

for i in range(len(iris\_dataset.target\_names)):

print("\n[{0}]:[{1}]".format(i,iris\_dataset.target\_names[i]))

print("\nIRIS DATA :\n",iris\_dataset["data"])

X\_train,X\_test,y\_train,y\_test=train\_test\_split(iris\_dataset["data"],iris\_dataset["target"],random\_state=0)

print("\nTarget :\n",iris\_dataset["target"])

print("\nX TRAIN \n",X\_train)

print("\nX TEST \n",X\_test)

print("\nY TRAIN \n",y\_train)

print("\nY TEST \n",y\_test)

kn=KNeighborsClassifier(n\_neighbors=1)

kn.fit(X\_train,y\_train)

x\_new=np.array([[5,2.9,1,0.2]])

print("\nXNEW \n",x\_new)

prediction=kn.predict(x\_new)

print("\nPredicted target value: {}\n".format(prediction))

print("\nPredicted feature name:{}\n".format(iris\_dataset["target\_names"][prediction]))

i=1

x=X\_test[i]

x\_new=np.array([x])

print("\nXNEW \n",x\_new)

for i in range(len(X\_test)):

x=X\_test[i]

x\_new=np.array([x])

prediction=kn.predict(x\_new)

print("\n Actual:[{0}][{1}] \t,Predicted:{2}{3}".format(y\_test[i],iris\_dataset["target\_names"][y\_test[i]],prediction,iris\_dataset["target\_names"][prediction]))

print("\nTEST SCORE[ACCURACY]: {:.2f}\n".format(kn.score(X\_test,y\_test)))

**Output:**

IRIS FEATURES \TARGET NAMES:

['setosa' 'versicolor' 'virginica']

[0]:[setosa]

[1]:[versicolor]

[2]:[virginica]

IRIS DATA :

[[5.1 3.5 1.4 0.2]

[4.9 3. 1.4 0.2]

[4.7 3.2 1.3 0.2]

[4.6 3.1 1.5 0.2]

[5. 3.6 1.4 0.2]

[5.4 3.9 1.7 0.4]

[4.6 3.4 1.4 0.3]

[5. 3.4 1.5 0.2]

[4.4 2.9 1.4 0.2]

[4.9 3.1 1.5 0.1]

[5.4 3.7 1.5 0.2]

[4.8 3.4 1.6 0.2]

[4.8 3. 1.4 0.1]

[4.3 3. 1.1 0.1]

[5.8 4. 1.2 0.2]

[5.7 4.4 1.5 0.4]

[5.4 3.9 1.3 0.4]

[5.1 3.5 1.4 0.3]

[5.7 3.8 1.7 0.3]

[5.1 3.8 1.5 0.3]

[5.4 3.4 1.7 0.2]

[5.1 3.7 1.5 0.4]

[4.6 3.6 1. 0.2]

[5.1 3.3 1.7 0.5]

[4.8 3.4 1.9 0.2]

[5. 3. 1.6 0.2]

[5. 3.4 1.6 0.4]

[5.2 3.5 1.5 0.2]

[5.2 3.4 1.4 0.2]

[4.7 3.2 1.6 0.2]

[4.8 3.1 1.6 0.2]

[5.4 3.4 1.5 0.4]

[5.2 4.1 1.5 0.1]

[5.5 4.2 1.4 0.2]

[4.9 3.1 1.5 0.2]

[5. 3.2 1.2 0.2]

[5.5 3.5 1.3 0.2]

[4.9 3.6 1.4 0.1]

[4.4 3. 1.3 0.2]

[5.1 3.4 1.5 0.2]

[5. 3.5 1.3 0.3]

[4.5 2.3 1.3 0.3]

[4.4 3.2 1.3 0.2]

[5. 3.5 1.6 0.6]

[5.1 3.8 1.9 0.4]

[4.8 3. 1.4 0.3]

[5.1 3.8 1.6 0.2]

[4.6 3.2 1.4 0.2]

[5.3 3.7 1.5 0.2]

[5. 3.3 1.4 0.2]

[7. 3.2 4.7 1.4]

[6.4 3.2 4.5 1.5]

[6.9 3.1 4.9 1.5]

[5.5 2.3 4. 1.3]

[6.5 2.8 4.6 1.5]

[5.7 2.8 4.5 1.3]

[6.3 3.3 4.7 1.6]

[4.9 2.4 3.3 1. ]

[6.6 2.9 4.6 1.3]

[5.2 2.7 3.9 1.4]

[5. 2. 3.5 1. ]

[5.9 3. 4.2 1.5]

[6. 2.2 4. 1. ]

[6.1 2.9 4.7 1.4]

[5.6 2.9 3.6 1.3]

[6.7 3.1 4.4 1.4]

[5.6 3. 4.5 1.5]

[5.8 2.7 4.1 1. ]

[6.2 2.2 4.5 1.5]

[5.6 2.5 3.9 1.1]

[5.9 3.2 4.8 1.8]

[6.1 2.8 4. 1.3]

[6.3 2.5 4.9 1.5]

[6.1 2.8 4.7 1.2]

[6.4 2.9 4.3 1.3]

[6.6 3. 4.4 1.4]

[6.8 2.8 4.8 1.4]

[6.7 3. 5. 1.7]

[6. 2.9 4.5 1.5]

[5.7 2.6 3.5 1. ]

[5.5 2.4 3.8 1.1]

[5.5 2.4 3.7 1. ]

[5.8 2.7 3.9 1.2]

[6. 2.7 5.1 1.6]

[5.4 3. 4.5 1.5]

[6. 3.4 4.5 1.6]

[6.7 3.1 4.7 1.5]

[6.3 2.3 4.4 1.3]

[5.6 3. 4.1 1.3]

[5.5 2.5 4. 1.3]

[5.5 2.6 4.4 1.2]

[6.1 3. 4.6 1.4]

[5.8 2.6 4. 1.2]

[5. 2.3 3.3 1. ]

[5.6 2.7 4.2 1.3]

[5.7 3. 4.2 1.2]

[5.7 2.9 4.2 1.3]

[6.2 2.9 4.3 1.3]

[5.1 2.5 3. 1.1]

[5.7 2.8 4.1 1.3]

[6.3 3.3 6. 2.5]

[5.8 2.7 5.1 1.9]

[7.1 3. 5.9 2.1]

[6.3 2.9 5.6 1.8]

[6.5 3. 5.8 2.2]

[7.6 3. 6.6 2.1]

[4.9 2.5 4.5 1.7]

[7.3 2.9 6.3 1.8]

[6.7 2.5 5.8 1.8]

[7.2 3.6 6.1 2.5]

[6.5 3.2 5.1 2. ]

[6.4 2.7 5.3 1.9]

[6.8 3. 5.5 2.1]

[5.7 2.5 5. 2. ]

[5.8 2.8 5.1 2.4]

[6.4 3.2 5.3 2.3]

[6.5 3. 5.5 1.8]

[7.7 3.8 6.7 2.2]

[7.7 2.6 6.9 2.3]

[6. 2.2 5. 1.5]

[6.9 3.2 5.7 2.3]

[5.6 2.8 4.9 2. ]

[7.7 2.8 6.7 2. ]

[6.3 2.7 4.9 1.8]

[6.7 3.3 5.7 2.1]

[7.2 3.2 6. 1.8]

[6.2 2.8 4.8 1.8]

[6.1 3. 4.9 1.8]

[6.4 2.8 5.6 2.1]

[7.2 3. 5.8 1.6]

[7.4 2.8 6.1 1.9]

[7.9 3.8 6.4 2. ]

[6.4 2.8 5.6 2.2]

[6.3 2.8 5.1 1.5]

[6.1 2.6 5.6 1.4]

[7.7 3. 6.1 2.3]

[6.3 3.4 5.6 2.4]

[6.4 3.1 5.5 1.8]

[6. 3. 4.8 1.8]

[6.9 3.1 5.4 2.1]

[6.7 3.1 5.6 2.4]

[6.9 3.1 5.1 2.3]

[5.8 2.7 5.1 1.9]

[6.8 3.2 5.9 2.3]

[6.7 3.3 5.7 2.5]

[6.7 3. 5.2 2.3]

[6.3 2.5 5. 1.9]

[6.5 3. 5.2 2. ]

[6.2 3.4 5.4 2.3]

[5.9 3. 5.1 1.8]]

Target :

[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 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 2]

X TRAIN

[[5.9 3. 4.2 1.5]

[5.8 2.6 4. 1.2]

[6.8 3. 5.5 2.1]

[4.7 3.2 1.3 0.2]

[6.9 3.1 5.1 2.3]

[5. 3.5 1.6 0.6]

[5.4 3.7 1.5 0.2]

[5. 2. 3.5 1. ]

[6.5 3. 5.5 1.8]

[6.7 3.3 5.7 2.5]

[6. 2.2 5. 1.5]

[6.7 2.5 5.8 1.8]

[5.6 2.5 3.9 1.1]

[7.7 3. 6.1 2.3]

[6.3 3.3 4.7 1.6]

[5.5 2.4 3.8 1.1]

[6.3 2.7 4.9 1.8]

[6.3 2.8 5.1 1.5]

[4.9 2.5 4.5 1.7]

[6.3 2.5 5. 1.9]

[7. 3.2 4.7 1.4]

[6.5 3. 5.2 2. ]

[6. 3.4 4.5 1.6]

[4.8 3.1 1.6 0.2]

[5.8 2.7 5.1 1.9]

[5.6 2.7 4.2 1.3]

[5.6 2.9 3.6 1.3]

[5.5 2.5 4. 1.3]

[6.1 3. 4.6 1.4]

[7.2 3.2 6. 1.8]

[5.3 3.7 1.5 0.2]

[4.3 3. 1.1 0.1]

[6.4 2.7 5.3 1.9]

[5.7 3. 4.2 1.2]

[5.4 3.4 1.7 0.2]

[5.7 4.4 1.5 0.4]

[6.9 3.1 4.9 1.5]

[4.6 3.1 1.5 0.2]

[5.9 3. 5.1 1.8]

[5.1 2.5 3. 1.1]

[4.6 3.4 1.4 0.3]

[6.2 2.2 4.5 1.5]

[7.2 3.6 6.1 2.5]

[5.7 2.9 4.2 1.3]

[4.8 3. 1.4 0.1]

[7.1 3. 5.9 2.1]

[6.9 3.2 5.7 2.3]

[6.5 3. 5.8 2.2]

[6.4 2.8 5.6 2.1]

[5.1 3.8 1.6 0.2]

[4.8 3.4 1.6 0.2]

[6.5 3.2 5.1 2. ]

[6.7 3.3 5.7 2.1]

[4.5 2.3 1.3 0.3]

[6.2 3.4 5.4 2.3]

[4.9 3. 1.4 0.2]

[5.7 2.5 5. 2. ]

[6.9 3.1 5.4 2.1]

[4.4 3.2 1.3 0.2]

[5. 3.6 1.4 0.2]

[7.2 3. 5.8 1.6]

[5.1 3.5 1.4 0.3]

[4.4 3. 1.3 0.2]

[5.4 3.9 1.7 0.4]

[5.5 2.3 4. 1.3]

[6.8 3.2 5.9 2.3]

[7.6 3. 6.6 2.1]

[5.1 3.5 1.4 0.2]

[4.9 3.1 1.5 0.2]

[5.2 3.4 1.4 0.2]

[5.7 2.8 4.5 1.3]

[6.6 3. 4.4 1.4]

[5. 3.2 1.2 0.2]

[5.1 3.3 1.7 0.5]

[6.4 2.9 4.3 1.3]

[5.4 3.4 1.5 0.4]

[7.7 2.6 6.9 2.3]

[4.9 2.4 3.3 1. ]

[7.9 3.8 6.4 2. ]

[6.7 3.1 4.4 1.4]

[5.2 4.1 1.5 0.1]

[6. 3. 4.8 1.8]

[5.8 4. 1.2 0.2]

[7.7 2.8 6.7 2. ]

[5.1 3.8 1.5 0.3]

[4.7 3.2 1.6 0.2]

[7.4 2.8 6.1 1.9]

[5. 3.3 1.4 0.2]

[6.3 3.4 5.6 2.4]

[5.7 2.8 4.1 1.3]

[5.8 2.7 3.9 1.2]

[5.7 2.6 3.5 1. ]

[6.4 3.2 5.3 2.3]

[6.7 3. 5.2 2.3]

[6.3 2.5 4.9 1.5]

[6.7 3. 5. 1.7]

[5. 3. 1.6 0.2]

[5.5 2.4 3.7 1. ]

[6.7 3.1 5.6 2.4]

[5.8 2.7 5.1 1.9]

[5.1 3.4 1.5 0.2]

[6.6 2.9 4.6 1.3]

[5.6 3. 4.1 1.3]

[5.9 3.2 4.8 1.8]

[6.3 2.3 4.4 1.3]

[5.5 3.5 1.3 0.2]

[5.1 3.7 1.5 0.4]

[4.9 3.1 1.5 0.1]

[6.3 2.9 5.6 1.8]

[5.8 2.7 4.1 1. ]

[7.7 3.8 6.7 2.2]

[4.6 3.2 1.4 0.2]]

X TEST

[[5.8 2.8 5.1 2.4]

[6. 2.2 4. 1. ]

[5.5 4.2 1.4 0.2]

[7.3 2.9 6.3 1.8]

[5. 3.4 1.5 0.2]

[6.3 3.3 6. 2.5]

[5. 3.5 1.3 0.3]

[6.7 3.1 4.7 1.5]

[6.8 2.8 4.8 1.4]

[6.1 2.8 4. 1.3]

[6.1 2.6 5.6 1.4]

[6.4 3.2 4.5 1.5]

[6.1 2.8 4.7 1.2]

[6.5 2.8 4.6 1.5]

[6.1 2.9 4.7 1.4]

[4.9 3.6 1.4 0.1]

[6. 2.9 4.5 1.5]

[5.5 2.6 4.4 1.2]

[4.8 3. 1.4 0.3]

[5.4 3.9 1.3 0.4]

[5.6 2.8 4.9 2. ]

[5.6 3. 4.5 1.5]

[4.8 3.4 1.9 0.2]

[4.4 2.9 1.4 0.2]

[6.2 2.8 4.8 1.8]

[4.6 3.6 1. 0.2]

[5.1 3.8 1.9 0.4]

[6.2 2.9 4.3 1.3]

[5. 2.3 3.3 1. ]

[5. 3.4 1.6 0.4]

[6.4 3.1 5.5 1.8]

[5.4 3. 4.5 1.5]

[5.2 3.5 1.5 0.2]

[6.1 3. 4.9 1.8]

[6.4 2.8 5.6 2.2]

[5.2 2.7 3.9 1.4]

[5.7 3.8 1.7 0.3]

[6. 2.7 5.1 1.6]]

Y TRAIN

[1 1 2 0 2 0 0 1 2 2 2 2 1 2 1 1 2 2 2 2 1 2 1 0 2 1 1 1 1 2 0 0 2 1 0 0 1

0 2 1 0 1 2 1 0 2 2 2 2 0 0 2 2 0 2 0 2 2 0 0 2 0 0 0 1 2 2 0 0 0 1 1 0 0

1 0 2 1 2 1 0 2 0 2 0 0 2 0 2 1 1 1 2 2 1 1 0 1 2 2 0 1 1 1 1 0 0 0 2 1 2

0]

Y TEST

[2 1 0 2 0 2 0 1 1 1 2 1 1 1 1 0 1 1 0 0 2 1 0 0 2 0 0 1 1 0 2 1 0 2 2 1 0

1]

XNEW

[[5. 2.9 1. 0.2]]

Predicted target value: [0]

Predicted feature name:['setosa']

XNEW

[[6. 2.2 4. 1. ]]

Actual:[2][virginica] ,Predicted:[2]['virginica']

Actual:[1][versicolor] ,Predicted:[1]['versicolor']

Actual:[0][setosa] ,Predicted:[0]['setosa']

Actual:[2][virginica] ,Predicted:[2]['virginica']

Actual:[0][setosa] ,Predicted:[0]['setosa']

Actual:[2][virginica] ,Predicted:[2]['virginica']

Actual:[0][setosa] ,Predicted:[0]['setosa']

Actual:[1][versicolor] ,Predicted:[1]['versicolor']

Actual:[1][versicolor] ,Predicted:[1]['versicolor']

Actual:[1][versicolor] ,Predicted:[1]['versicolor']

Actual:[2][virginica] ,Predicted:[2]['virginica']

Actual:[1][versicolor] ,Predicted:[1]['versicolor']

Actual:[1][versicolor] ,Predicted:[1]['versicolor']

Actual:[1][versicolor] ,Predicted:[1]['versicolor']

Actual:[1][versicolor] ,Predicted:[1]['versicolor']

Actual:[0][setosa] ,Predicted:[0]['setosa']

Actual:[1][versicolor] ,Predicted:[1]['versicolor']

Actual:[1][versicolor] ,Predicted:[1]['versicolor']

Actual:[0][setosa] ,Predicted:[0]['setosa']

Actual:[0][setosa] ,Predicted:[0]['setosa']

Actual:[2][virginica] ,Predicted:[2]['virginica']

Actual:[1][versicolor] ,Predicted:[1]['versicolor']

Actual:[0][setosa] ,Predicted:[0]['setosa']

Actual:[0][setosa] ,Predicted:[0]['setosa']

Actual:[2][virginica] ,Predicted:[2]['virginica']

Actual:[0][setosa] ,Predicted:[0]['setosa']

Actual:[0][setosa] ,Predicted:[0]['setosa']

Actual:[1][versicolor] ,Predicted:[1]['versicolor']

Actual:[1][versicolor] ,Predicted:[1]['versicolor']

Actual:[0][setosa] ,Predicted:[0]['setosa']

Actual:[2][virginica] ,Predicted:[2]['virginica']

Actual:[1][versicolor] ,Predicted:[1]['versicolor']

Actual:[0][setosa] ,Predicted:[0]['setosa']

Actual:[2][virginica] ,Predicted:[2]['virginica']

Actual:[2][virginica] ,Predicted:[2]['virginica']

Actual:[1][versicolor] ,Predicted:[1]['versicolor']

Actual:[0][setosa] ,Predicted:[0]['setosa']

Actual:[1][versicolor] ,Predicted:[2]['virginica']

TEST SCORE[ACCURACY]: 0.97

**Program 9:Locally Weighted Regression**

import matplotlib.pyplot as pl

from math import ceil

import numpy as np

from scipy import linalg

def lowess(x, y, f=2. / 3., iter=3):

n = len(x)

r = int(ceil(f \* n))

h = [np.sort(np.abs(x - x[i]))[r] for i in range(n)]

w = np.clip(np.abs((x[:, None] - x[None, :]) / h), 0.0, 1.0)

w = (1 - w \*\* 3) \*\* 3

yest = np.zeros(n)

delta = np.ones(n)

for iteration in range(iter):

for i in range(n):

weights = delta \* w[:, i]

b = np.array([np.sum(weights \* y), np.sum(weights \* y \* x)])

A = np.array([[np.sum(weights), np.sum(weights \* x)],

[np.sum(weights \* x), np.sum(weights \* x \* x)]])

beta = linalg.solve(A, b)

yest[i] = beta[0] + beta[1] \* x[i]

residuals = y - yest

s = np.median(np.abs(residuals))

delta = np.clip(residuals / (6.0 \* s), -1, 1)

delta = (1 - delta \*\* 2) \*\* 2

return yest

if \_\_name\_\_ == '\_\_main\_\_':

import math

n = 100

x = np.linspace(0, 2 \* math.pi, n)

y = np.sin(x) + 0.3 \* np.random.randn(n)

f = 0.25

yest = lowess(x, y, f=f, iter=3)

import pylab as pl

pl.clf()

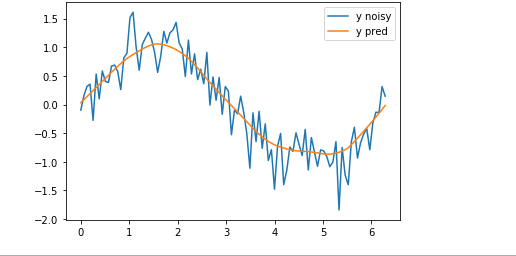
pl.plot(x, y, label='y noisy')

pl.plot(x, yest, label='y pred')

pl.legend()

pl.show()

**Output:**



**VIVA QUESTIONS**

**1) What is Machine learning?**

Machine learning is a branch of computer science which deals with system programming in order to automatically learn and improve with experience. For example: Robots are programed so that they can perform the task based on data they gather from sensors. It automatically learns programs from data.

**2) Mention the difference between Data Mining and Machine learning?**

Machine learning relates with the study, design and development of the algorithms that give computers the capability to learn without being explicitly programmed. While, data mining can be defined as the process in which the unstructured data tries to extract knowledge or unknown interesting patterns. During this process machine learning algorithms are used.

**3) What is ‘Overfitting’ in Machine learning?**

In machine learning, when a statistical model describes random error or noise instead of underlying relationship ‘overfitting’ occurs. When a model is excessively complex, overfitting is normally observed, because of having too many parameters with respect to the number of training data types. The model exhibits poor performance which has been overfit.

**4) Why overfitting happens?**

The possibility of overfitting exists as the criteria used for training the model is not the same as the criteria used to judge the efficacy of a model.

**5) How can you avoid overfitting ?**

By using a lot of data overfitting can be avoided, overfitting happens relatively as you have a small dataset, and you try to learn from it. But if you have a small database and you are forced to come with a model based on that. In such situation, you can use a technique known as cross validation. In this method the dataset splits into two section, testing and training datasets, the testing dataset will only test the model while, in training dataset, the datapoints will come up with the model.

In this technique, a model is usually given a dataset of a known data on which training (training data set) is run and a dataset of unknown data against which the model is tested. The idea of cross validation is to define a dataset to “test” the model in the training phase.

**6) What is inductive machine learning?**

The inductive machine learning involves the process of learning by examples, where a system, from a set of observed instances tries to induce a general rule.

**7) What are the five popular algorithms of Machine Learning?**

a) Decision Trees

b) Neural Networks (back propagation)

c) Probabilistic networks

d) Nearest Neighbor

e) Support vector machines

**8) What are the different Algorithm techniques in Machine Learning?**

The different types of techniques in Machine Learning are

a) Supervised Learning

b) Unsupervised Learning

c) Semi-supervised Learning

d) Reinforcement Learning

e) Transduction

f) Learning to Learn

**9) What are the three stages to build the hypotheses or model in machine learning?**

a) Model building

b) Model testing

c) Applying the model

**10) What is the standard approach to supervised learning?**

The standard approach to supervised learning is to split the set of example into the training set and the test.

**11) What is ‘Training set’ and ‘Test set’?**

In various areas of information science like machine learning, a set of data is used to discover the potentially predictive relationship known as ‘Training Set’. Training set is an examples given to the learner, while Test set is used to test the accuracy of the hypotheses generated by the learner, and it is the set of example held back from the learner. Training set are distinct from Test set.

**12) List down various approaches for machine learning?**

The different approaches in Machine Learning are

a) Concept Vs Classification Learning

b) Symbolic Vs Statistical Learning

c) Inductive Vs Analytical Learning

**13) What is not Machine Learning?**

a) Artificial Intelligence

b) Rule based inference

**14) Explain what is the function of ‘Unsupervised Learning’?**

a) Find clusters of the data

b) Find low-dimensional representations of the data

c) Find interesting directions in data

d) Interesting coordinates and correlations

e) Find novel observations/ database cleaning

**15) Explain what is the function of ‘Supervised Learning’?**

a) Classifications

b) Speech recognition

c) Regression

d) Predict time series

e) Annotate strings

**16) What is algorithm independent machine learning?**

Machine learning in where mathematical foundations is independent of any particular classifier or learning algorithm is referred as algorithm independent machine learning?

**17) What is the difference between artificial learning and machine learning?**

Designing and developing algorithms according to the behaviours based on empirical data are known as Machine Learning. While artificial intelligence in addition to machine learning, it also covers other aspects like knowledge representation, natural language processing, planning, robotics etc.

**18) What is classifier in machine learning?**

A classifier in a Machine Learning is a system that inputs a vector of discrete or continuous feature values and outputs a single discrete value, the class.

**19) What are the advantages of Naive Bayes?**

In Naïve Bayes classifier will converge quicker than discriminative models like logistic regression, so you need less training data. The main advantage is that it can’t learn interactions between features.

**20) In what areas Pattern Recognition is used?**

Pattern Recognition can be used in

a) Computer Vision

b) Speech Recognition

c) Data Mining

d) Statistics

e) Informal Retrieval

f) Bio-Informatics

21) What is Genetic Programming?

Genetic programming is one of the two techniques used in machine learning. The model is based on the testing and selecting the best choice among a set of results.

**22) What is Inductive Logic Programming in Machine Learning?**

Inductive Logic Programming (ILP) is a subfield of machine learning which uses logical programming representing background knowledge and examples.

**23) What is Model Selection in Machine Learning?**

The process of selecting models among different mathematical models, which are used to describe the same data set is known as Model Selection. Model selection is applied to the fields of statistics, machine learning and data mining

**24) What are the two methods used for the calibration in Supervised Learning?**

The two methods used for predicting good probabilities in Supervised Learning are

a) Platt Calibration

b) Isotonic Regression

These methods are designed for binary classification, and it is not trivial.

**25) Which method is frequently used to prevent overfitting?**

When there is sufficient data ‘Isotonic Regression’ is used to prevent an overfitting issue.

**26) What is the difference between heuristic for rule learning and heuristics for decision trees?**

The difference is that the heuristics for decision trees evaluate the average quality of a number of disjointed sets while rule learners only evaluate the quality of the set of instances that is covered with the candidate rule.

**27) What is Perceptron in Machine Learning?**

In Machine Learning, Perceptron is an algorithm for supervised classification of the input into one of several possible non-binary outputs.

**28) Explain the two components of Bayesian logic program?**

Bayesian logic program consists of two components. The first component is a logical one ; it consists of a set of Bayesian Clauses, which captures the qualitative structure of the domain. The second component is a quantitative one, it encodes the quantitative information about the domain.

**29) What are Bayesian Networks (BN) ?**

Bayesian Network is used to represent the graphical model for probability relationship among a set of variables .

**30) Why instance based learning algorithm sometimes referred as Lazy learning algorithm?**

Instance based learning algorithm is also referred as Lazy learning algorithm as they delay the induction or generalization process until classification is performed.

**31) What are the two classification methods that SVM ( Support Vector Machine) can handle?**

a) Combining binary classifiers

b) Modifying binary to incorporate multiclass learning

**32) What is ensemble learning?**

To solve a particular computational program, multiple models such as classifiers or experts are strategically generated and combined. This process is known as ensemble learning.

**33) Why ensemble learning is used?**

Ensemble learning is used to improve the classification, prediction, function approximation etc of a model.

**34) When to use ensemble learning?**

Ensemble learning is used when you build component classifiers that are more accurate and independent from each other.

**35) What are the two paradigms of ensemble methods?**

The two paradigms of ensemble methods are

a) Sequential ensemble methods

b) Parallel ensemble methods

**36) What is the general principle of an ensemble method and what is bagging and boosting in ensemble method?**

The general principle of an ensemble method is to combine the predictions of several models built with a given learning algorithm in order to improve robustness over a single model. Bagging is a method in ensemble for improving unstable estimation or classification schemes. While boosting method are used sequentially to reduce the bias of the combined model. Boosting and Bagging both can reduce errors by reducing the variance term.

**37) What is bias-variance decomposition of classification error in ensemble method?**

The expected error of a learning algorithm can be decomposed into bias and variance. A bias term measures how closely the average classifier produced by the learning algorithm matches the target function. The variance term measures how much the learning algorithm’s prediction fluctuates for different training sets.

**38) What is an Incremental Learning algorithm in ensemble?**

Incremental learning method is the ability of an algorithm to learn from new data that may be available after classifier has already been generated from already available dataset.

**39) What is PCA, KPCA and ICA used for?**

PCA (Principal Components Analysis), KPCA ( Kernel based Principal Component Analysis) and ICA ( Independent Component Analysis) are important feature extraction techniques used for dimensionality reduction.

**40) What is dimension reduction in Machine Learning?**

In Machine Learning and statistics, dimension reduction is the process of reducing the number of random variables under considerations and can be divided into feature selection and feature extraction

**41) What are support vector machines?**

Support vector machines are supervised learning algorithms used for classification and regression analysis.

**42) What are the components of relational evaluation techniques?**

The important components of relational evaluation techniques are

a) Data Acquisition

b) Ground Truth Acquisition

c) Cross Validation Technique

d) Query Type

e) Scoring Metric

f) Significance Test

**43) What are the different methods for Sequential Supervised Learning?**

The different methods to solve Sequential Supervised Learning problems are

a) Sliding-window methods

b) Recurrent sliding windows

c) Hidden Markow models

d) Maximum entropy Markow models

e) Conditional random fields

f) Graph transformer networks

**44) What are the areas in robotics and information processing where sequential prediction problem arises?**

The areas in robotics and information processing where sequential prediction problem arises are

a) Imitation Learning

b) Structured prediction

c) Model based reinforcement learning

**45) What is batch statistical learning?**

Statistical learning techniques allow learning a function or predictor from a set of observed data that can make predictions about unseen or future data. These techniques provide guarantees on the performance of the learned predictor on the future unseen data based on a statistical assumption on the data generating process.

**46) What is PAC Learning?**

PAC (Probably Approximately Correct) learning is a learning framework that has been introduced to analyze learning algorithms and their statistical efficiency.

**47) What different categories can you categorize in the sequence learning process?**

a) Sequence prediction

b) Sequence generation

c) Sequence recognition

d) Sequential decision

**48) What is sequence learning?**

Sequence learning is a method of teaching and learning in a logical manner.

**49) What are two techniques of Machine Learning ?**

The two techniques of Machine Learning are

a) Genetic Programming

b) Inductive Learning

**50) Give a popular application of machine learning that you see on day to day basis?**

The recommendation engine implemented by major ecommerce websites uses Machine Learning

**CONTENT BEYOND SYLLABUS**

**Program 1: Load a dataset for machine learning locally.**

import pandas as pd

#reading csv file

data = pd.read\_csv('weather.csv')

# shape of dataset

print("Shape:", data.shape)

# column names

print("\nFeatures:", data.columns)

#storing the feature matrix (X) and response vector (y)

X = data[data.columns[:-1]]

y = data[data.columns[-1]]

# printing first 5 rows of feature matrix

print("\nFeature matrix:\n", X.head(10))

# printing first 5 values of response vector

print("\nResponse vector:\n", y.head(10))

**Output**

Shape: (14, 5)

Features: Index(['Outlook', 'Temperature', 'Humidity', 'Windy', 'Play'], dtype='object')

Feature matrix:

Outlook Temperature Humidity Windy

0 overcast hot high False

1 overcast cool normal True

2 overcast mild high True

3 overcast hot normal False

4 rainy mild high False

5 rainy cool normal False

6 rainy cool normal True

7 rainy mild normal False

8 rainy mild high True

9 sunny hot high False

Response vector:

0 yes

1 yes

2 yes

3 yes

4 yes

5 yes

6 no

7 yes

8 no

9 no

Name: Play, dtype: object

**Program 2: Program to draw Box and whisker plot on iris dataset.**

#Load libraries

import pandas

from pandas.tools.plotting import scatter\_matrix

import matplotlib.pyplot as plt

#Load dataset

url = "https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"

names = ['sepal-length', 'sepal-width', 'petal-length', 'petal-width', 'class']

dataset = pandas.read\_csv(url, names=names)

#shape

print(dataset.shape)

#head

print(dataset.head(20))

#descriptions

print(dataset.describe())

#class distribution

print(dataset.groupby('class').size())

#box and whisker plots

dataset.plot(kind='box', subplots=True, layout=(2,2), sharex=False, sharey=False)

plt.show()

#histograms

dataset.hist()

plt.show()

**Output**

(150, 5)

sepal-length sepal-width ... petal-width class

0 5.1 3.5 ... 0.2 Iris-setosa

1 4.9 3.0 ... 0.2 Iris-setosa

2 4.7 3.2 ... 0.2 Iris-setosa

3 4.6 3.1 ... 0.2 Iris-setosa

4 5.0 3.6 ... 0.2 Iris-setosa

5 5.4 3.9 ... 0.4 Iris-setosa

6 4.6 3.4 ... 0.3 Iris-setosa

7 5.0 3.4 ... 0.2 Iris-setosa

8 4.4 2.9 ... 0.2 Iris-setosa

9 4.9 3.1 ... 0.1 Iris-setosa

10 5.4 3.7 ... 0.2 Iris-setosa

11 4.8 3.4 ... 0.2 Iris-setosa

12 4.8 3.0 ... 0.1 Iris-setosa

13 4.3 3.0 ... 0.1 Iris-setosa

14 5.8 4.0 ... 0.2 Iris-setosa

15 5.7 4.4 ... 0.4 Iris-setosa

16 5.4 3.9 ... 0.4 Iris-setosa

17 5.1 3.5 ... 0.3 Iris-setosa

18 5.7 3.8 ... 0.3 Iris-setosa

19 5.1 3.8 ... 0.3 Iris-setosa

[20 rows x 5 columns]

sepal-length sepal-width petal-length petal-width

count 150.000000 150.000000 150.000000 150.000000

mean 5.843333 3.054000 3.758667 1.198667

std 0.828066 0.433594 1.764420 0.763161

min 4.300000 2.000000 1.000000 0.100000

25% 5.100000 2.800000 1.600000 0.300000

50% 5.800000 3.000000 4.350000 1.300000

75% 6.400000 3.300000 5.100000 1.800000

max 7.900000 4.400000 6.900000 2.500000

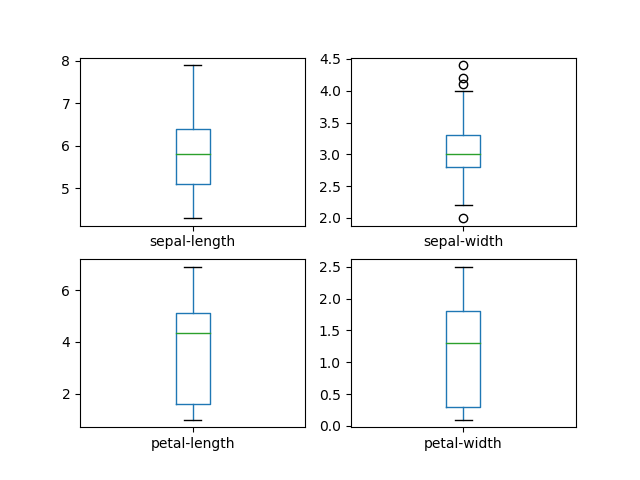
class

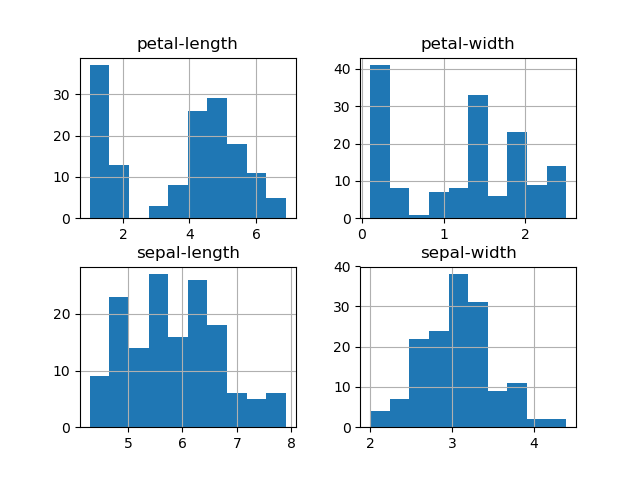
Iris-setosa 50

Iris-versicolor 50

Iris-virginica 50

dtype: int64





**Program 3: Python program to draw sigmoid function**

import numpy as np

import matplotlib.pyplot as plt

z = np.arange(-2, 2, 0.01);

sigmoid = 1/(1+np.exp(-z));

fig = plt.figure('Cost function convergence')

plt.plot(z,sigmoid)

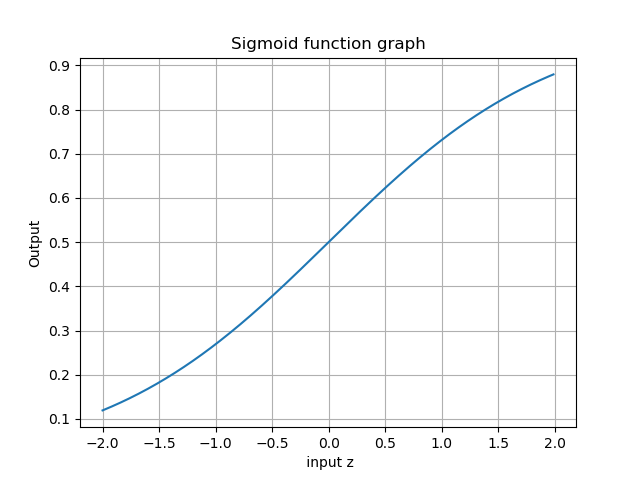
plt.grid(True)

plt.xlabel(' input z')

plt.ylabel('Output')

plt.title('Sigmoid function graph')

plt.show()



**Program 4: Simple Artificial neural network implementation in python**

from numpy import \*

class NeuralNet(object):

def \_\_init\_\_(self):

# Generate random numbers

random.seed(1)

# Assign random weights to a 3 x 1 matrix,

self.synaptic\_weights = 2 \* random.random((3, 1)) - 1

# The Sigmoid function

def \_\_sigmoid(self, x):

return 1 / (1 + exp(-x))

# The derivative of the Sigmoid function.

# This is the gradient of the Sigmoid curve.

def \_\_sigmoid\_derivative(self, x):

return x \* (1 - x)

# Train the neural network and adjust the weights each time.

def train(self, inputs, outputs, training\_iterations):

for iteration in range(training\_iterations):

# Pass the training set through the network.

output = self.learn(inputs)

# Calculate the error

error = outputs - output

# Adjust the weights by a factor

factor = dot(inputs.T, error \* self.\_\_sigmoid\_derivative(output))

self.synaptic\_weights += factor

# The neural network thinks.

def learn(self, inputs):

return self.\_\_sigmoid(dot(inputs, self.synaptic\_weights))

if \_\_name\_\_ == "\_\_main\_\_":

#Initialize

neural\_network = NeuralNet()

# The training set.

inputs = array([[0, 1, 1], [1, 0, 0], [1, 0, 1]])

outputs = array([[1, 0, 1]]).T

# Train the neural network

neural\_network.train(inputs, outputs, 10000)

# Test the neural network with a test example.

print (neural\_network.learn(array([1, 0, 1])))

**Output:**

[0.9897704]

**Program 5: Support vector Machine (SVM) on iris dataset.**

import numpy as np

import matplotlib.pyplot as plt

from sklearn import svm, datasets

# import some data to play with

iris = datasets.load\_iris()

X = iris.data[:, :2] # we only take the first two features. We could

# avoid this ugly slicing by using a two-dim dataset

y = iris.target

# we create an instance of SVM and fit out data. We do not scale our

# data since we want to plot the support vectors

C = 1.0 # SVM regularization parameter

svc = svm.SVC(kernel='linear', C=1,gamma=0.1).fit(X, y)

# create a mesh to plot in

x\_min, x\_max = X[:, 0].min() - 1, X[:, 0].max() + 1

y\_min, y\_max = X[:, 1].min() - 1, X[:, 1].max() + 1

h = (x\_max / x\_min)/100

xx, yy = np.meshgrid(np.arange(x\_min, x\_max, h),

np.arange(y\_min, y\_max, h))

plt.subplot(1, 1, 1)

Z = svc.predict(np.c\_[xx.ravel(), yy.ravel()])

Z = Z.reshape(xx.shape)

plt.contourf(xx, yy, Z, cmap=plt.cm.Paired, alpha=0.3)

plt.scatter(X[:, 0], X[:, 1], c=y, cmap=plt.cm.Paired)

plt.xlabel('Sepal length')

plt.ylabel('Sepal width')

plt.xlim(xx.min(), xx.max())

plt.title('SVC with linear kernel')

plt.show()

**OUTPUT**

