SIDDHARTH INSTITUTE OF ENGINEERING & TECHNOLOGY

(AUTONOMOUS)

(Approved by AICTE, New Delhi& Affiliated to JNTUA, Ananthapuramu)

(Accredited by NBA for Civil, EEE, Mech., ECE & CSE

Accredited by NAAC with 'A+' Grade)

Puttur -517583, Tirupati District, AP.

Department of Computer Science and Engineering

(Common to CSM and CAD)

(20CS0902) ARTIFICIAL INTELLIGENCE LAB

II B.Tech -II Semester

Lab Observation Book

Academic Year : _____

Name : _____

Roll. Number : _____

Year & Branch : _____

Specialization : _____

SIDDHARTH INSTITUTE OF ENGINEERING & TECHNOLOGY

(AUTONOMOUS)
(Approved by AICTE, New Delhi& Affiliated to JNTUA, Ananthapuramu)
(Accredited by NBA for Civil, EEE, Mech., ECE & CSE
Accredited by NAAC with 'A' Grade)
Puttur -517583, Tirupati District, AP.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

INSTITUTE VISION

To emerge as one of the premier institutions through excellence in education and research, producing globally competent and ethically strong professionals and entrepreneurs

INSTITUTE MISSION

M1: Imparting high-quality technical and management education through the state-of-the- art resources.

M2: Creating an eco-system to conduct independent and collaborative research for the betterment of the society

M3: Promoting entrepreneurial skills and inculcating ethics for the socio-economic development of the nation.

DEPARTMENT VISION

To impart quality education and research in Computer Science and Engineering for producing technically competent and ethically strong IT professionals with contemporary knowledge

DEPARTMENT MISSION

M1: Achieving academic excellence in computer science through effective pedagogy, modern curriculum and state-of-art computing facilities.

M2: Encouraging innovative research in Computer Science and Engineering by collaborating with Industry and Premier Institutions to serve the nation.

M3: Empowering the students by inculcating professional behavior, strong ethical values and leadership abilities

SIDDHARTH INSTITUTE OF ENGINEERING & TECHNOLOGY

(AUTONOMOUS)
(Approved by AICTE, New Delhi& Affiliated to JNTUA, Ananthapuramu)
(Accredited by NBA for Civil, EEE, Mech., ECE & CSE
Accredited by NAAC with 'A' Grade)
Puttur -517583, Tirupati District, AP.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Program Outcomes

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.

SIDDHARTH INSTITUTE OF ENGINEERING & TECHNOLOGY
(AUTONOMOUS)
(Approved by AICTE, New Delhi& Affiliated to JNTUA, Ananthapuramu)
(Accredited by NBA for Civil, EEE, Mech., ECE & CSE
Accredited by NAAC with 'A' Grade)
Puttur -517583, Tirupati District, AP.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

PROGRAM EDUCATIONAL OBJECTIVES (PEOs)

PEO1: To provide software solutions for arising problems in diverse areas with strong knowledge in innovative technologies of computer science.

PEO2: To serve in IT industry as professionals and entrepreneurs or in pursuit of higher education and research.

PEO3: To attain professional etiquette, soft skills, leadership, ethical values meld with a commitment for lifelong learning.

PROGRAM SPECIFIC OUTCOMES (PSOs)

PSO1: **Analysis & Design:** Ability to design, develop and deploy customized applications in all applicable domains using various algorithms and programming languages.

PSO2: **Computational Logic:** Ability to visualize and configure computational need in terms of hardware and software to provide solutions for various complex applications.

PSO3: **Software Development:** Ability to apply standard procedures, tools and strategies for software development.

Do's:

1. Know the location of the fire extinguisher and the first aid box and how to use them in case of an emergency.

2. Read and understand how to carry out an activity thoroughly before coming to the laboratory.

3. Report fires or accidents to your lecturer/laboratory technician immediately.

4. Report any broken plugs or exposed electrical wires to your lecturer/laboratory technician immediately.

Don'ts:

1. Do not eat or drink in the laboratory.

2. Avoid stepping on electrical wires or any other computer cables.

3. Do not open the system unit casing or monitor casing particularly when the power is turned on. Some internal components hold electric voltages of up to 30000 volts, which can be fatal.

4. Do not insert metal objects such as clips, pins and needles into the computer casings. They may cause fire.

5. Do not remove anything from the computer laboratory without permission.

6. Do not touch, connect or disconnect any plug or cable without your lecturer/laboratory technician's permission.

7. Do not misbehave in the computer laboratory.

COURSE OBJECTIVES

The objective of the course is to

1. *Make use of Datasets in implementing the machine learning algorithms*
2. *Implement them a chine learning concepts and algorithms in any suitable language of choice.*

COURSE OUTCOMES

On successful completion of the course, the students will be able to

1. *Understand the implementation procedures for the machine learning algorithms.*
2. *Design Java/Python programs for various Learning algorithms.*
3. *Apply appropriate datasets to the Machine Learning algorithms.*
4. *Identify Machine Learning algorithms to solve real world problems.*
5. *Write Machine Learning algorithms to solve real world problems.*
6. *Implement different machine learning algorithms*

List of Experiments:

1	Write a program to demonstrate the working of the decision tree algorithm. Use an appropriate data set for building the decision tree and apply this knowledge to classify a new sample.
2	Write a program for implementing the Back propagation algorithm and test the same using appropriate datasets.
3	Write a program for implementing the classification using Multilayer perceptron.
4	Write a program to implement the naïve Bayesian classifier for a sample training dataset stored as a .CSV file. Compute the accuracy of the classifier, considering few test datasets
5	Assuming a set of documents that need to be classified, use the naïve Bayesian Classifier model to perform this task. Built-in Java classes/API can be used to write the program. Calculate the accuracy, precision, and recall for your dataset
6	Write a program to construct a Bayesian network considering medical data. Use this model to demonstrate the diagnosis of heart patients using standard Heart Disease DataSet. You can use Java/Python ML library classes/API.
7	Apply EM algorithm to cluster a set of data stored in a .CSV file. Use the same Dataset 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 Principle Component Analysis for Dimensionality Reduction.
9	Write a program to implement k-Nearest Neighbour algorithm to classify the iris dataset. Print both correct and wrong predictions. Java/Python ML library classes canbe used for this problem.
10	Implement the non-parametric Locally Weighted Regression algorithm in order to fit datapoints. Select appropriate data set for your experiment and draw graphs.

INDEX

Ex.No. 1	Implementation of Decision Tree Algorithm	Date:

Aim:

Write a program to demonstrate the working of the decision tree algorithm.

Program:

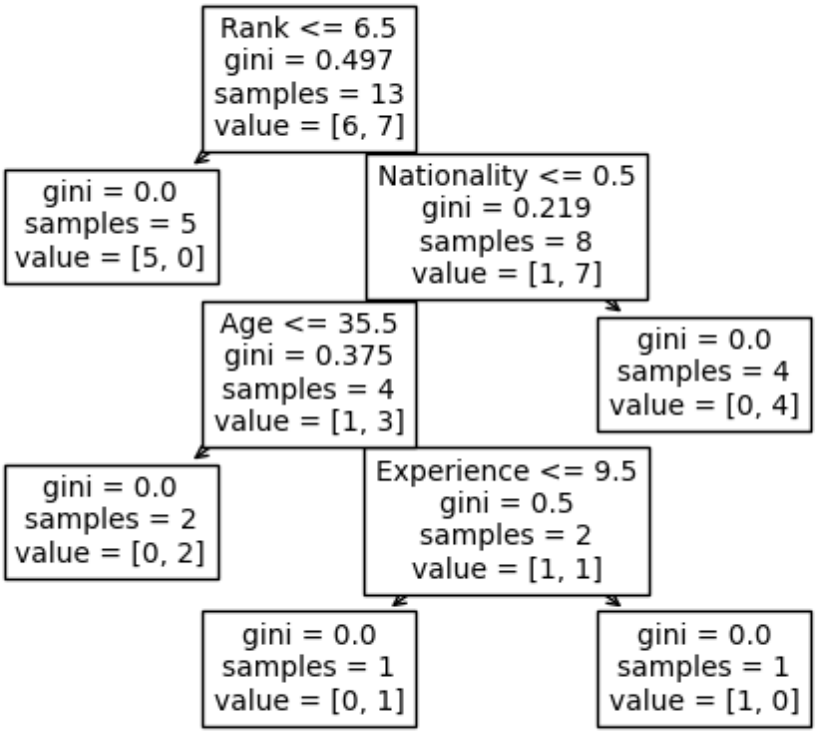
```
#Three lines to make our compiler able to draw:
import sys
import matplotlib
matplotlib.use('Agg')
import pandas
from sklearn import tree
from sklearn.tree import DecisionTreeClassifier
import matplotlib.pyplot as plt
df = pandas.read_csv("data.csv")
d = {'UK': 0, 'USA': 1, 'N': 2}
df['Nationality'] = df['Nationality'].map(d)
d = {'YES': 1, 'NO': 0}
df['Go'] = df['Go'].map(d)
features = ['Age', 'Experience', 'Rank', 'Nationality']
X = df[features]
y = df['Go']
```

Dataset:

Age	Experience	Rank	Nationality	Go
36	10	9	UK	NO
42	12	4	USA	NO
23	4	6	N	NO
52	4	4	USA	NO
43	21	8	USA	YES
44	14	5	UK	NO
66	3	7	N	YES
35	14	9	UK	YES
52	13	7	N	YES

35	**5**	**9**	**N**	**YES**
24	**3**	**5**	**USA**	**NO**
18	**3**	**7**	**UK**	**YES**
45	**9**	**9**	**UK**	**YES**

Output:



Result:

Ex.No. 2	Back Propagation Algorithm.	Date:

Aim:

Write a program for implementing the Back propagation algorithm and test the same using appropriate datasets.

Program:

```
import numpy as np
X = np.array(([2, 9], [1, 5], [3, 6]), dtype=float)
y = np.array(([92], [86], [89]), dtype=float)
X = X/np.amax(X,axis=0)
# maximum of X array longitudinally
y = y/100
#Sigmoid Function
def sigmoid (x): return 1/(1 + np.exp(-x))
#Derivative of Sigmoid Function
def derivatives_sigmoid(x): return x * (1 - x)
#Variable initialization
epoch=5000
#Setting training iterations
lr=0.1
#Setting learning rate
inputlayer_neurons = 2
#number of features in data set
hiddenlayer_neurons = 3
#number of hidden layers neurons
output_neurons = 1
#number of neurons at output layer
#weight and bias initialization
wh=np.random.uniform(size=(inputlayer_neurons,hiddenlayer_neurons))
bh=np.random.uniform(size=(1,hiddenlayer_neurons))
wout=np.random.uniform(size=(hiddenlayer_neurons,output_neurons))
bout=np.random.uniform(size=(1,output_neurons))
#draws a random range of numbers uniformly of dim x*y  for i in range(epoch):
#Forward Propogation
hinp1=np.dot(X,wh)
```

```
hinp=hinp1 + bh
hlayer_act = sigmoid(hinp)
outinp1=np.dot(hlayer_act,wout)
outinp= outinp1+ bout
output = sigmoid(outinp)
#Backpropagation
EO = y-output
outgrad = derivatives_sigmoid(output)
d_output = EO* outgrad
EH = d_output.dot(wout.T)
#how much hidden layer wts contributed to error
hiddengrad = derivatives_sigmoid(hlayer_act)
d_hiddenlayer = EH * hiddengrad
# dotproduct of nextlayererror and currentlayerop
wout += hlayer_act.T.dot(d_output) *lr
wh+= X.T.dot(d_hiddenlayer) *lr
print("Input: \n" + str(X))
print("Actual Output: \n" + str(y))
print("Predicted Output: \n" ,output)
```

Output:

Input:

```
[[0.66666667 1.        ]
 [0.33333333 0.55555556]
 [1.         0.66666667]]
```

Actual Output:

```
[[0.92]
 [0.86]
 [0.89]]
```

Predicted Output:

```
 [[0.92745804]
 [0.91954311]
 [0.92598481]]
```

Result:

Ex.No. 3	Classification using Multilayer perceptron	Date:

Aim:

Write a program for implementing the classification using Multilayer perceptron.

Program:

```
import numpy as np
import pandas as pd
import os
import copy
import time
import torch
import torch nn as nn
import cv2
import matplotlib.pyplot as plt
import copy
import time
import albumentations as A
import torch_optimizer as optim
from res_mlp_pytorch
import ResMLP
fromPIL import Image
from albumentations.pytorch
import ToTensorV2
from torch.utils.data
import Dataset, DataLoader
class FoodDataset(Dataset):
    def__init_(self, data_type=None, transforms=None):self.path =
    '../input/food5k/Food-5K/' + data_type + '/'self.images_name = os.listdir(self.path)
    self.transforms = transforms
def_len_(self):
        return len(self.images_name)
def getitem(self, idx):
        data = self.images_name[idx]
        label = data.split('_')[0]
```

```
        label = int(label)
        label = torch.tensor(label)
        image = cv2.imread(self.path + data)
        image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
        if self.transforms:aug =self.transforms(image=image)image = aug['image']
          return (image, label)
train_data = FoodDataset('training', A.Compose([ A.RandomResizedCrop(256, 256),
                              A.HorizontalFlip(), A.Normalize(),ToTensorV2()]))
val_data = FoodDataset('validation', A.Compose([ A.Resize(384, 384),
                             A.CenterCrop(256, 256), A.Normalize(), ToTensorV2()]))
dataloaders = {
                  'train': DataLoader(train_data, batch_size=32,
                  shuffle=True, num_workers=4),'val':
                  DataLoader(val_data, batch_size=32, shuffle=True,
                  num_workers=4), 'test': DataLoader(test_data,
                  batch_size=32, shuffle=True, num_workers=4)
                }
 dataset_sizes = {'train': len(train_data),'val': len(val_data),'test': len(test_data)
 def train_model(model, criterion, optimizer, epochs=1):
   since = 0.0
   best_model_wts = copy.deepcopy(model.state_dict())
    best_loss = 0.0
    best_acc = 0
   for ep in range(epochs): print(f"Epoch {ep}/{epochs-1}")print("-"*10)
   for phase in ['train', 'val']:
          if phase == 'train':model.train()
          else:
          model.eval()
    running_loss = 0.0
    running_corrects = 0

    for images, labels in dataloaders[phase]:
          images = images.to(device)
          labels = labels.to(device)
    optimizer.zero_grad() with torch.set_grad_enabled(phase == 'train'):
    outputs =model(images)
              _, preds = torch.max(outputs, 1)
              loss = criterion(output s, labels)
```

```
                if phase == 'train':loss.backward()
                optimizer.step()
                running_loss += loss.item() *images.size(0)
                 running_corrects +=torch.sum(preds == labels.data)

            epoch_loss = running_loss / dataset_sizes[phase]
            epoch_acc = running_corrects.double() / dataset_sizes[phase]
            print(f"{phase}  Loss:{epoch_loss:.4f} Acc:{epoch_acc:.4f}")
            if phase == 'val':
            if ep == 0:

            best_loss = epoch_loss
                best_model_wts=copy.deepcopy(model.state_dict())
                else:
                   if epoch_loss < best_loss:
                       best_loss = epoch_loss
                       best_acc = epoch_acc
                       best_model_wts = copy.deepcopy(model.state_dict())

        print()

        time_elapsed = time.time() - since
        print(f'Training complete in {time_elapsed // 60}m{time_elapsed % 60}s')
        print(f'Best val loss:{best_loss:.4f}')
        print(f'Best acc:{best_acc}')
        model.load_state_dict(best_model_wts)
        return model
        # Train The Model
       device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
       model = ResMLP(image_size=256, patch_size=16, dim=512, depth=12,
    num_classes=2)
    model = model.to(device)
    criterion = nn.CrossEntropyLoss()
    optimizer = optim.Lamb(model.parameters(), lr=0.005,weight_decay=0.2)
    best_model = train_model(model, criterion, optimizer, epochs=20)
```

Output:

Result:

Ex.No. 4	Naïve Bayesian Classifier	Date:

Aim:

To 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.

Program:

```
import pandas as pd
from sklearn import tree
from sklearn.preprocessing import LabelEncoder
from sklearn.naive_bayes import GaussianNB

data = pd.read_csv('tennisdata.csv')
print("The first 5 values of data is :\n",data.head())

X = data.iloc[:,:-1]
print("\nThe First 5 values of train data is\n",X.head())
y = data.iloc[:,-1]
print("\nThe first 5 values of Train output is\n",y.head())

le_outlook = LabelEncoder()
X.Outlook = le_outlook.fit_transform(X.Outlook)
le_Temperature = LabelEncoder()
X.Temperature = le_Temperature.fit_transform(X.Temperature)
le_Humidity = LabelEncoder()
X.Humidity = le_Humidity.fit_transform(X.Humidity)
le_Windy = LabelEncoder()
X.Windy = le_Windy.fit_transform(X.Windy)

print("\nNow the Train data is :\n",X.head())
le_PlayTennis = LabelEncoder()
y = le_PlayTennis.fit_transform(y)
print("\nNow the Train output is\n",y)

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.20)
classifier = GaussianNB()
classifier.fit(X_train,y_train)

from sklearn.metrics import accuracy_score
print("Accuracy is:",accuracy_score(classifier.predict(X_test),y_test))
```

Tennisdata.csv

Outlook	Temperature	Humidity	Windy	PlayTennis
Sunny	Hot	High	FALSE	No
Sunny	Hot	High	TRUE	No
Overcast	Hot	High	FALSE	Yes
Rainy	Mild	High	FALSE	Yes
Rainy	Cool	Normal	FALSE	Yes
Rainy	Cool	Normal	TRUE	No
Overcast	Cool	Normal	TRUE	Yes
Sunny	Mild	High	FALSE	No
Sunny	Cool	Normal	FALSE	Yes
Rainy	Mild	Normal	FALSE	Yes
Sunny	Mild	Normal	TRUE	Yes
Overcast	Mild	High	TRUE	Yes
Overcast	Hot	Normal	FALSE	Yes
Rainy	Mild	High	TRUE	No

Output:

The first 5 values of data is :

	Outlook	Temperature	Humidity	Windy	PlayTennis
0	Sunny	Hot	High	False	No
1	Sunny	Hot	High	True	No
2	Overcast	Hot	High	False	Yes
3	Rainy	Mild	High	False	Yes
4	Rainy	Cool	Normal	False	Yes

The First 5 values of train data is

	Outlook	Temperature	Humidity	Windy
0	Sunny	Hot	High	False
1	Sunny	Hot	High	True
2	Overcast	Hot	High	False
3	Rainy	Mild	High	False
4	Rainy	Cool	Normal	False

The first 5 values of Train output is

0 No

1 No

2 Yes

3 Yes

4 Yes

Name: PlayTennis, dtype: object

Now the Train data is :

	Outlook	Temperature	Humidity	Windy
0	2	1	0	0
1	2	1	0	1
2	0	1	0	0
3	1	2	0	0
4	1	0	1	0

Now the Train output is

[0 0 1 1 1 0 1 0 1 1 1 1 1 0]

Accuracy is: 0.3333333333333333

Result:

Ex.No. 5	Naïve Bayesian Classifier for Text Classification	Date:

Aim:

Assuming a set of documents that need to be classified, use the naïve Bayesian classifier model to perform this task. Built-in Java classes/API can be used to write the program. Calculate the accuracy, precision, and recall for your dataset.

Program:

```
import pandas as pd
msg = pd.read_csv(r"C:\Users\Sirisha\Desktop\ml\document.csv",
                                                    names=['message', 'label'])
print("Total Instances of Dataset: ", msg.shape[0])
msg['labelnum'] = msg.label.map({'pos': 1, 'neg': 0})

X = msg.message
y = msg.labelnum
from sklearn.model_selection import train_test_split
Xtrain, Xtest, ytrain, ytest = train_test_split(X, y)
from sklearn.feature_extraction.text import CountVectorizer

count_v = CountVectorizer()
Xtrain_dm = count_v.fit_transform(Xtrain)
Xtest_dm = count_v.transform(Xtest)

df =
pd.DataFrame(Xtrain_dm.toarray(),columns=count_v.get_feature_names_out())
print(df[0:5])

from sklearn.naive_bayes import MultinomialNB
clf = MultinomialNB()
clf.fit(Xtrain_dm, ytrain)
pred = clf.predict(Xtest_dm)

for doc, p in zip(Xtrain, pred):
   p = 'pos' if p == 1 else 'neg'
   print("%s -> %s" % (doc, p))
from sklearn.metrics import accuracy_score, confusion_matrix, precision_score,
recall_score
print('Accuracy Metrics: \n')
print('Accuracy: ', accuracy_score(ytest, pred))
print('Recall: ', recall_score(ytest, pred))
print('Precision: ', precision_score(ytest, pred))
print('Confusion Matrix: \n', confusion_matrix(ytest, pred))
```

Output:

```
Total Instances of Dataset:  17
   about  am  amazingplace  an  and  awesome  bad  beers  best  dance  ..
.  \
0      0   0             0   0    0        0    0      0     0      1  ..
.
1      0   0             0   0    0        0    0      0     0      0  ..
.
2      0   0             0   1    0        1    0      0     0      0  ..
.
3      0   0             0   0    0        0    0      0     0      0  ..
.
4      0   0             1   1    0        0    0      0     0      0  ..
.

   taste  that  these  this  thisis  tired  to  very  what  work
0      0     0      0     0       0      0   1     0     0     0
1      1     0      0     1       0      0   0     0     0     0
2      0     0      0     1       0      0   0     0     0     0
3      0     0      0     0       0      0   0     0     0     0
4      0     0      0     0       1      0   0     0     0     0

[5 rows x 42 columns]
I love to dance -> pos
I do not like he taste of this juice -> neg
This is an awesome place -> neg
Heis my sworn enemy -> neg
Thisis an amazingplace -> neg
Accuracy Metrics:

Accuracy:  0.8
Recall:  0.5
Precision:  1.0
Confusion Matrix:
 [[3 0]
 [1 1]]
```

Result:

Ex.No.6	Bayesian network considering Medical Dataset	Date:

Aim:

Write a program to construct a Bayesian network considering medical data. Use this model to demonstrate the diagnosis of heart patients using standard Heart Disease DataSet. You can use Java/Python ML library classes/API.

Program:

```
import pandas as pd
from pgmpy.estimators import MaximumLikelihoodEstimator
from pgmpy.models import BayesianModel
from pgmpy.inference import VariableElimination
data = pd.read_csv(r"C:\Users\Sirisha\Downloads\ds4.csv")
heart_disease = pd.DataFrame(data)
print(heart_disease)
model = BayesianModel([
   ('age', 'Lifestyle'),
   ('Gender', 'Lifestyle'),
   ('Family', 'heartdisease'),
   ('diet', 'cholestrol'),
   ('Lifestyle', 'diet'),
   ('cholestrol', 'heartdisease'),
   ('diet', 'cholestrol')
])
model.fit(heart_disease, estimator=MaximumLikelihoodEstimator)
HeartDisease_infer = VariableElimination(model)
print('For Age enter SuperSeniorCitizen:0, SeniorCitizen:1, MiddleAged:2, Youth:3, Teen:4')
print('For Gender enter Male:0, Female:1')
print('For Family History enter Yes:1, No:0')
print('For Diet enter High:0, Medium:1')
print('for LifeStyle enter Athlete:0, Active:1, Moderate:2, Sedentary:3')
print('for Cholesterol enter High:0, BorderLine:1, Normal:2')
q = HeartDisease_infer.query(variables=['heartdisease'], evidence={
   'age': int(input('Enter Age: ')),
   'Gender': int(input('Enter Gender: ')),
   'Family': int(input('Enter Family History: ')),
   'diet': int(input('Enter Diet: ')),
   'Lifestyle': int(input('Enter Lifestyle: ')),
   'cholestrol': int(input('Enter Cholestrol: '))
```

```
})

print(q)
```

Dataset:

age	Gender	Family	diet	Lifestyle	cholesterol	heartdisease
0	0	1	1	3	0	1
0	1	1	1	3	0	1
1	0	0	0	2	1	1
4	0	1	1	3	2	0
3	1	1	0	0	2	0
2	0	1	1	1	0	1
4	0	1	0	2	0	1
0	0	1	1	3	0	1
3	1	1	0	0	2	0
1	1	0	0	0	2	1
4	1	0	1	2	0	1
4	0	1	1	3	2	0
2	1	0	0	0	0	0
2	0	1	1	1	0	1
3	1	1	0	0	1	0
0	0	1	0	0	2	1
1	1	0	1	2	1	1
3	1	1	1	0	1	0
4	0	1	1	3	2	0

Output:

```
For Age enter SuperSeniorCitizen:0, SeniorCitizen:1, MiddleAged:2, You
th:3, Teen:4
For Gender enter Male:0, Female:1
For Family History enter Yes:1, No:0
For Diet enter High:0, Medium:1
for LifeStyle enter Athlete:0, Active:1, Moderate:2, Sedentary:3
for Cholesterol enter High:0, BorderLine:1, Normal:2
Enter Age: 2
Enter Gender: 1
Enter Family History: 0
Enter Diet: 1
Enter Lifestyle: 2
Enter Cholestrol: 1

+----------------+--------------------+
| heartdisease   |   phi(heartdisease) |
+================+====================+
| heartdisease(0) |             0.5000 |
+----------------+--------------------+
| heartdisease(1) |             0.5000 |
+----------------+--------------------+
```

Result:

Ex.No. 7	Expectation Maximization Algorithm	Date:

Aim:

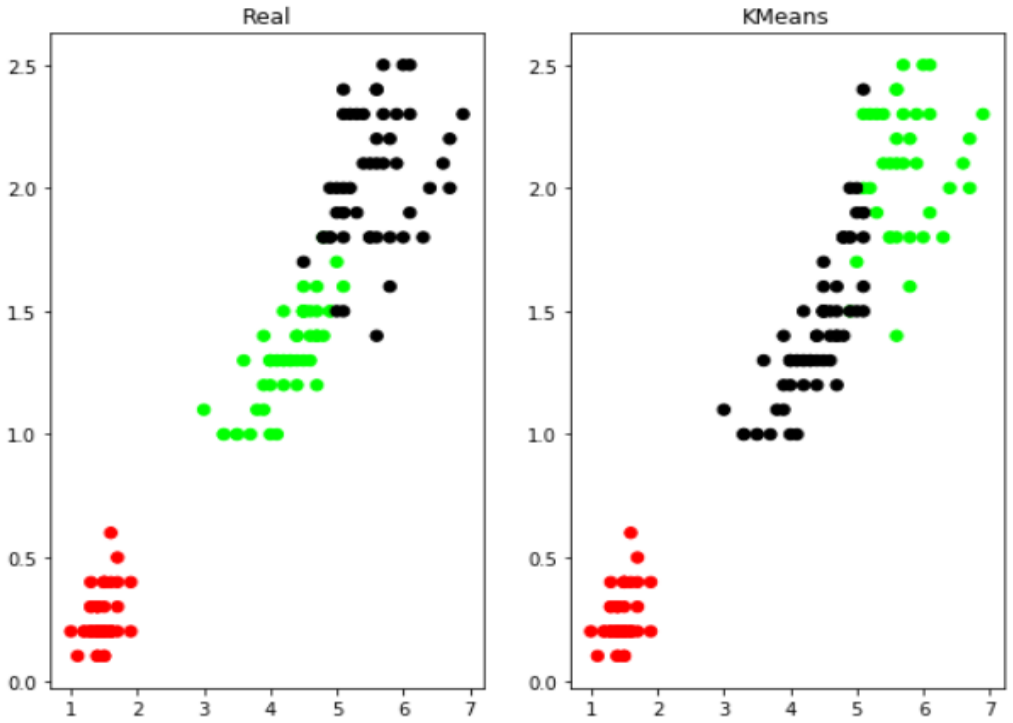
Apply EM algorithm to cluster a set of data stored in a .CSV file. Use the same Dataset for clustering using k-Means algorithm. Compare the results of these two algorithms and comment on the quality of clustering. You can add Java/Python ML library classes / API in the program.

Program:

```
from sklearn.cluster import KMeans
from sklearn import preprocessing
from sklearn.mixture import GaussianMixture
from sklearn.datasets import load_iris
import sklearn.metrics as sm
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
dataset=load_iris()
# print(dataset)
X=pd.DataFrame(dataset.data)
X.columns=['Sepal_Length','Sepal_Width','Petal_Length','Petal_Width']
y=pd.DataFrame(dataset.target)
y.columns=['Targets']
# print(X)
plt.figure(figsize=(14,7))
colormap=np.array(['red','lime','black'])
# REAL PLOT
plt.subplot(1,3,1)
plt.scatter(X.Petal_Length,X.Petal_Width,c=colormap[y.Targets],s=40)
plt.title('Real')
# K-PLOT
plt.subplot(1,3,2)
model=KMeans(n_clusters=3)
model.fit(X)
```

```
predY=np.choose(model.labels_,[0,1,2]).astype(np.int64)
plt.scatter(X.Petal_Length,X.Petal_Width,c=colormap[predY],s=40)
plt.title('KMeans')
```

Output:



Result:

Ex.No. 8	Principle Component Analysis for Dimensionality Reduction.	Date:

Aim:

Write a program to implement Principle Component Analysis for Dimensionality Reduction.

Program:

```
from numpy import array
from numpy import mean
from numpy import cov
from numpy.linalg import eig

# define a small 3×2 matrix
matrix = array([[5, 6], [8, 10], [12, 18]])
print("original Matrix: ")
print(matrix)

# calculate the mean of each column
Mean_col = mean(matrix.T, axis=1)
print("Mean of each column: ")
print(Mean_col)

# center columns by subtracting column means
Centre_col = matrix - Mean_col
print("Covariance Matrix: ")
print(Centre_col)

# calculate covariance matrix of centered matrix
cov_matrix = cov(Centre_col.T)
print(cov_matrix)

# eigendecomposition of covariance matrix
values, vectors = eig(cov_matrix)
print("Eigen vectors: ",vectors)
print("Eigen values: ",values)

# project data on the new axes
projected_data = vectors.T.dot(Centre_col.T)
print(projected_data.T)
```

Output:

```
original Matrix:
[[ 5  6]
 [ 8 10]
 [12 18]]
Mean of each column:
[ 8.33333333 11.33333333]
Covariance Matrix:
[[-3.33333333 -5.33333333]
 [-0.33333333 -1.33333333]
 [ 3.66666667  6.66666667]]
[[12.33333333 21.33333333]
 [21.33333333 37.33333333]]
Eigen vectors:  [[-0.86762506 -0.49721902]
 [ 0.49721902 -0.86762506]]
Eigen values:  [ 0.10761573 49.55905094]
[[ 0.24024879  6.28473039]
 [-0.37375033  1.32257309]
 [ 0.13350154 -7.60730348]]
```

Result:

Ex.No. 9	k-Nearest Neighbor algorithm	Date:

Aim:

Write a program to implement k-Nearest Neighbor algorithm to classify the iris dataset. Print both correct and wrong predictions. Java/Python ML library classes can be used for this problem.

Program:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
import seaborn as sns
df = pd.read_csv(r"C:\Users\Sirisha\Desktop\iris_flower_dataset.csv" )
df.head()
X = df.drop("species", axis=1)
X.head()
y = df["species"]
y.head()
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
classifier = KNeighborsClassifier(n_neighbors=5)
classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)
y_pred
acc = accuracy_score(y_test, y_pred)
print("Accuracy:", acc)
classifier2 = KNeighborsClassifier(n_neighbors=50)
classifier2.fit(X_train, y_train)
y_pred2 = classifier2.predict(X_test)
y_pred2
acc2 = accuracy_score(y_test, y_pred2)
print("Accuracy:", acc2)
cm = confusion_matrix(y_test, y_pred)
print(cm)
sns.heatmap(cm, annot=True)
cm2 = confusion_matrix(y_test, y_pred2)
print(cm2)
sns.heatmap(cm2, annot=True)
```

Output:

```
Accuracy: 1.0
Accuracy: 0.9666666666666667
[[10  0  0]
 [ 0  9  0]
 [ 0  0 11]]
```

```
[[10  0  0]
 [ 0  9  0]
 [ 0  1 10]]
<Axes: >
```

Result:

Ex.No.10	Locally Weighted Regression algorithm	Date:

Aim:

Implement the non-parametric Locally Weighted Regression algorithm in order to fit datapoints. Select appropriate data set for your experiment and draw graphs.

Program:

```
from numpy import *
import operator
from os import listdir
import matplotlib
import matplotlib.pyplot as plt
import pandas as pd
import numpy.linalg
from scipy.stats.stats import pearsonr
def kernel(point,xmat, k):
 m,n = shape(xmat)
 weights = mat(eye((m)))
 for j in range(m):
 diff = point - X[j]
 weights[j,j] = exp(diff*diff.T/(-2.0*k**2))
 return weights
def localWeight(point,xmat,ymat,k):
 wei = kernel(point,xmat,k)
 W = (X.T*(wei*X)).I*(X.T*(wei*ymat.T))
 return W
def localWeightRegression(xmat,ymat,k):
 m,n = shape(xmat)
 ypred = zeros(m)
 for i in range(m):
 ypred[i] = xmat[i]*localWeight(xmat[i],xmat,ymat,k)
 return ypred
# load data points
data = pd.read_csv(r"C:\Users\Sirisha\Desktop\ml\tips.csv")
bill = array(data.total_bill)
tip = array(data.tip)
#preparing and add 1 in bill
mbill = mat(bill)
```

```
mtip = mat(tip)
m= shape(mbill)[1]
one = mat(ones(m))
X= hstack((one.T,mbill.T))
#set k here
ypred = localWeightRegression(X,mtip,0.2)
SortIndex = X[:,1].argsort(0)
xsort = X[SortIndex][:,0]
fig = plt.figure()
ax = fig.add_subplot(1,1,1)
ax.scatter(bill,tip, color='green')
ax.plot(xsort[:,1],ypred[SortIndex], color = 'red', linewidth=5)
plt.xlabel('Total bill')
plt.ylabel('Tip')
plt.show();
```

A Sample Dataset:

total_bill	tip	sex	smoker	day	time	size
16.99	1.01	Female	No	Sun	Dinner	2
10.34	1.66	Male	No	Sun	Dinner	3
21.01	3.5	Male	No	Sun	Dinner	3
23.68	3.31	Male	No	Sun	Dinner	2
24.59	3.61	Female	No	Sun	Dinner	4
25.29	4.71	Male	No	Sun	Dinner	4
8.77	2	Male	No	Sun	Dinner	2
26.88	3.12	Male	No	Sun	Dinner	4
15.04	1.96	Male	No	Sun	Dinner	2
14.78	3.23	Male	No	Sun	Dinner	2
10.27	1.71	Male	No	Sun	Dinner	2
35.26	5	Female	No	Sun	Dinner	4
15.42	1.57	Male	No	Sun	Dinner	2
18.43	3	Male	No	Sun	Dinner	4
14.83	3.02	Female	No	Sun	Dinner	2
21.58	3.92	Male	No	Sun	Dinner	2
10.33	1.67	Female	No	Sun	Dinner	3
16.29	3.71	Male	No	Sun	Dinner	3
16.97	3.5	Female	No	Sun	Dinner	3
20.65	3.35	Male	No	Sat	Dinner	3
17.92	4.08	Male	No	Sat	Dinner	2
20.29	2.75	Female	No	Sat	Dinner	2
15.77	2.23	Female	No	Sat	Dinner	2
39.42	7.58	Male	No	Sat	Dinner	4
19.82	3.18	Male	No	Sat	Dinner	2
17.81	2.34	Male	No	Sat	Dinner	4

13.37	2	Male	No	Sat	Dinner	2
12.69	2	Male	No	Sat	Dinner	2
21.7	4.3	Male	No	Sat	Dinner	2
19.65	3	Female	No	Sat	Dinner	2
9.55	1.45	Male	No	Sat	Dinner	2
18.35	2.5	Male	No	Sat	Dinner	4
15.06	3	Female	No	Sat	Dinner	2
20.69	2.45	Female	No	Sat	Dinner	4
17.78	3.27	Male	No	Sat	Dinner	2
24.06	3.6	Male	No	Sat	Dinner	3
16.31	2	Male	No	Sat	Dinner	3
16.93	3.07	Female	No	Sat	Dinner	3
18.69	2.31	Male	No	Sat	Dinner	3
31.27	5	Male	No	Sat	Dinner	3
16.04	2.24	Male	No	Sat	Dinner	3
17.46	2.54	Male	No	Sun	Dinner	2
13.94	3.06	Male	No	Sun	Dinner	2
9.68	1.32	Male	No	Sun	Dinner	2
30.4	5.6	Male	No	Sun	Dinner	4
18.29	3	Male	No	Sun	Dinner	2
22.23	5	Male	No	Sun	Dinner	2
32.4	6	Male	No	Sun	Dinner	4
28.55	2.05	Male	No	Sun	Dinner	3
18.04	3	Male	No	Sun	Dinner	2
12.54	2.5	Male	No	Sun	Dinner	2
10.29	2.6	Female	No	Sun	Dinner	2
34.81	5.2	Female	No	Sun	Dinner	4
9.94	1.56	Male	No	Sun	Dinner	2
25.56	4.34	Male	No	Sun	Dinner	4
19.49	3.51	Male	No	Sun	Dinner	2
38.01	3	Male	Yes	Sat	Dinner	4
26.41	1.5	Female	No	Sat	Dinner	2
11.24	1.76	Male	Yes	Sat	Dinner	2
48.27	6.73	Male	No	Sat	Dinner	4
20.29	3.21	Male	Yes	Sat	Dinner	2
13.81	2	Male	Yes	Sat	Dinner	2
11.02	1.98	Male	Yes	Sat	Dinner	2
18.29	3.76	Male	Yes	Sat	Dinner	4
17.59	2.64	Male	No	Sat	Dinner	3
20.08	3.15	Male	No	Sat	Dinner	3
16.45	2.47	Female	No	Sat	Dinner	2
3.07	1	Female	Yes	Sat	Dinner	1
20.23	2.01	Male	No	Sat	Dinner	2
15.01	2.09	Male	Yes	Sat	Dinner	2
12.02	1.97	Male	No	Sat	Dinner	2
17.07	3	Female	No	Sat	Dinner	3
26.86	3.14	Female	Yes	Sat	Dinner	2

Output:

Result: