MACHINE LEARNING LABORATORY 18AIL66

LABORATORY MANUAL

VI Semester
B.E. CSE
(Data Science)
(Academic Year: 2022-23)

Dr. Navaneeth Bhaskar
Associate Professor
Department of CSE (Data Science)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING (DATA SCIENCE)

SAHYADRI

College of Engineering &
Management Adyar, Mangaluru
- 575007

SAHYADRI
COLLEGE OF ENGINEERING & MANAGEMENT
An Autonomous Institution

Vision

To be a premier institution in Technology and Management by fostering excellence in education, innovation, incubation and values to inspire and empower the young minds.

Mission

M1. Creating an academic ambience to impart holistic education focusing on individual growth, integrity, ethical values and social responsibility.

M2. Develop skill based learning through industry-institution interaction to enhance competency and promote entrepreneurship.

M3. Fostering innovation and creativity through competitive environment with state-of-the- art infrastructure.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING (DATA SCIENCE)

Vision

To be a center of excellence in Data Science and Engineering through the interactive teaching-learning process, research, and innovation.

Mission

M1. Creating competitive ambience to enhance the innovative and experiential learning process through state of the art infrastructure.

M2. Grooming young minds through industry-institute interactions to solve societal issues and inculcate affinity towards research and entrepreneurship.

M3. Promoting teamwork and leadership qualities through inter-disciplinary activities in diversified areas of data science and engineering.

Program Educational Objectives (PEOs):

PEO1: Possess theoretical and practical knowledge to identify, scrutinize, formulate and solve challenging problems related to dynamically evolving data science.

PEO2: Inculcate core competency, professionalism and ethics to cater industrial needs and to solve societal problems.

PEO3: Engage in Lifelong learning and stay intact to the transformation in technologies and pursue research.

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.

Program Specific Outcomes (PSOs):

PSO1: Exhibit competency and skills in distributed computing, information security, cyber security, data analytics, and machine learning.

PSO2: Able to provide sustainable solution to implement and validate data scienceprojects.

COURSE OUTCOMES

COs	Description	Bloom's Level
CO1	Demonstrate Machine learning algorithms for finding the hypothesis	CL3
CO2	Demonstrate data pre-processing techniques on an appropriate dataset	CL3
CO3	Implement ML algorithms to classify a new test sample	CL3
CO4	Demonstrate the performance parameters of the classifiers	CL3
CO5	Implement a Bayesian network considering medical data	CL3

Contents

Program 1:

Aim: Implement and demonstrate the Find-S algorithm for finding the most specific hypothesis based on a given set of training data samples. Read the training data from a .CSV file and show the output for test cases.

PROGRAM:

```
import csv
import pandas as
pd a = []
d=pd.read_csv("enjoysport.csv")
print(d)
with open('enjoysport.csv', 'r') as
        csvfile: for row in
        csv.reader(csvfile):
                a.append(row)
print("\n The total number of training instances are :
",len(a)) num_attribute = len(a[0])-1
print("\n The initial hypothesis is :
") hypothesis = ['0']*num_attribute
print(hypothesis)
for i in range(0, len(a)):
        if a[i][num_attribute] == 'yes':
                for j in range(0, num_attribute):
                        if hypothesis[j] == '0' or hypothesis[j] ==
                                a[i][j]: hypothesis[j] = a[i][j]
                        else:
                                hypothesis[j] = '?'
        print("\n The hypothesis for the training instance {} is :\n"
.format(i+1),hypothesis) print("\n The Maximally specific hypothesis for the training
instance is ") print(hypothesis)
```

OUTPUT:

```
    Sky    airtemp  humidity   wind  water     forcast    enjoysport
0  sunny    warm     normal     strong warm      same        yes
1  sunny    warm     high       strong warm      same        yes
2  rainy     cold     high       strong warm      change      no
3        sunny        warm       high             strong cool
change      yes The total number of training instances are:  5
The initial hypothesis is:
['0', '0', '0', '0', '0', '0']
The hypothesis for the training instance 1
is: ['0', '0', '0', '0', '0', '0']
The hypothesis for the training instance 2 is:
['sunny', 'warm', 'normal', 'strong', 'warm',
```

'same'] The hypothesis for the training instance
3 is: ['sunny', 'warm', '?', 'strong', 'warm', 'same']

The hypothesis for the training instance 4
is: ['sunny', 'warm', '?', 'strong', 'warm',
'same'] The hypothesis for the training
instance 5 is: ['sunny', 'warm', '?', 'strong',
'?', '?']
The Maximally specific hypothesis for the training instance
is ['sunny', 'warm', '?', 'strong', '?', '?']

Program 2:

Aim: For a given set of training data examples stored in a .CSV file, implement and demonstrate Candidate Elimination algorithm. Output a description of the set of all hypotheses consistent with the training examples.

PROGRAM:

```
import numpy as
np import pandas
as pd
data =
pd.DataFrame(data=pd.read_csv('enjoysport.csv'))
concepts = np.array(data.iloc[:,0:-1])
print(concepts)
target =
np.array(data.iloc[:,-1])
print(target)
def learn(concepts, target):
    specific_h = concepts[0].copy()
    print("initialization of specific_h and
    general_h") print(specific_h)
    general_h = [["?" for i in range(len(specific_h))] for i in
    range(len(specific_h))] print(general_h)
    for i, h in
        enumerate(concepts): if
        target[i] == "yes":
            for x in range(len(specific_h)):
                if h[x]!=
                    specific_h[x]:
                    specific_h[x] ='?'
                    general_h[x][x] ='?'
                print(specific_h
        ) print(specific_h)
        if target[i] == "no":
            for x in
                range(len(specific_h)): if
                h[x]!= specific_h[x]:
                    general_h[x][x] =
                specific_h[x] else:
                    general_h[x][x] = '?'
        print(" steps of Candidate Elimination
        Algorithm",i+1) print(specific_h)
        print(general_h)
    indices = [i for i, val in enumerate(general_h) if val == ['?', '?', '?', '?', '?',
    '?']] for i in indices:
```

```
        general_h.remove(['?', '?', '?', '?', '?',
    '?']) return specific_h, general_h
s_final, g_final = learn(concepts, target)
print("Final Specific_h:", s_final,
sep="\n") print("Final General_h:",
g_final, sep="\n")
```

OUTPUT:

```
[['sunny' 'warm' 'normal' 'strong' 'warm' 'same']
['sunny' 'warm' 'high' 'strong' 'warm' 'same']
['rainy' 'cold' 'high' 'strong' 'warm' 'change']
['sunny' 'warm' 'high' 'strong' 'cool'
'change']] ['yes' 'yes' 'no' 'yes']
initialization of specific_h and general_h
['sunny' 'warm' 'normal' 'strong' 'warm' 'same']
[['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'],  ['?', '?', '?', '?', '?', '?'],
['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?']]
['sunny' 'warm' 'normal' 'strong' 'warm' 'same']
['sunny' 'warm' 'normal' 'strong' 'warm' 'same']
['sunny' 'warm' 'normal' 'strong' 'warm' 'same']
['sunny' 'warm' 'normal' 'strong' 'warm' 'same']
['sunny' 'warm' 'normal' 'strong' 'warm' 'same']
['sunny' 'warm' 'normal' 'strong' 'warm' 'same']
['sunny' 'warm' 'normal' 'strong' 'warm'
'same'] steps of Candidate Elimination
Algorithm 1 ['sunny' 'warm' 'normal' 'strong'
'warm' 'same']
[['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'],
['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?']]
['sunny' 'warm' 'normal' 'strong' 'warm' 'same']
['sunny' 'warm' 'normal' 'strong' 'warm' 'same']
['sunny' 'warm' '?' 'strong' 'warm' 'same']
['sunny' 'warm' '?' 'strong' 'warm' 'same']
['sunny' 'warm' '?' 'strong' 'warm' 'same']
['sunny' 'warm' '?' 'strong' 'warm' 'same']
['sunny' 'warm' '?' 'strong' 'warm' 'same']
steps of Candidate Elimination Algorithm 2
['sunny' 'warm' '?' 'strong' 'warm' 'same']
[['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'],
['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?']]
['sunny' 'warm' '?' 'strong' 'warm' 'same']
steps of Candidate Elimination Algorithm
3 ['sunny' 'warm' '?' 'strong' 'warm' 'same']
[['sunny', '?', '?', '?', '?', '?'], ['?', 'warm', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'],
['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', 'same']]
['sunny' 'warm' '?' 'strong' 'warm' 'same']
['sunny' 'warm' '?' 'strong' 'warm' 'same']
['sunny' 'warm' '?' 'strong' 'warm' 'same']
['sunny' 'warm' '?' 'strong' 'warm' 'same']
['sunny' 'warm' '?' 'strong' '?' 'same']
['sunny' 'warm' '?' 'strong' '?' '?']
['sunny' 'warm' '?' 'strong' '?' '?']
steps of Candidate Elimination Algorithm
4 ['sunny' 'warm' '?' 'strong' '?' '?']
[['sunny', '?', '?', '?', '?', '?'], ['?', 'warm', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'],
['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?']]
Final Specific_h: ['sunny' 'warm' '?' 'strong' '?' '?']
Final General_h: [['sunny', '?', '?', '?', '?', '?'], ['?', 'warm', '?', '?', '?', '?']]
```

Program 3:

Aim: Demonstrate Preprocessing (Data Cleaning, Integration and Transformation) activity on suitable data: For example: Identify and Delete Rows that Contain Duplicate Data by considering an appropriate dataset. Identify and Delete Columns that contain a Single value by considering an appropriate dataset.

PROGRAM:

```
import pandas as pd
from sklearn.preprocessing import
MinMaxScaler # Load the dataset
data =
pd.read_csv('student.csv') #
printing original dataset
print(data)
# Data cleaning
data = data.dropna() # Remove rows with missing
values # Remove duplicate names
data =
data.drop_duplicates(subset='Name') #
print the dataset after cleaning
print("\n",data)
# Data transformation ie, Scale numerical
features scaler = MinMaxScaler()
scaled_data = scaler.fit_transform(data[['Age',
'GPA']]) data[['Age', 'GPA']] = scaled_data
# Data integration
# Combining columns 'Name' and 'Grade' into a new column
'Student_Info' data['Student_Info'] = data['Name'] + ' (' + data['Grade'] + ')'
# Print the preprocessed
data print("\n",data.head())
```

OUTPUT:

```
      Name Age Gender Grade GPA
0     Ethan 20.0   male     A 8.0
1      Liam 21.0   male     A 8.1
2      Liam 20.0   male     B 6.0
3     Grace 21.0 female      A 9.0
4    Wilson 20.0   male     A 9.1
5     Emily NaN female       A 8.3
6  Mitchell 22.0   male NaN 7.7
7  Benjamin 20.0   male     B NaN
8    Olivia NaN female       A 8.0
9    Sophia 18.0 female      A 8.1
10  Jackson 19.0   male     B 7.5
11   Wilson 21.0   male     A 8.9
12    Lucas NaN    male     B 7.0
```

13 Ava 21.0 female A 8.4

	Name	Age	Gender	Grade	GPA
0	Ethan	20.0	male	A	8.0
1	Liam	21.0	male	A	8.1
3	Grace	21.0	female	A	9.0
4	Wilson	20.0	male	A	9.1
9	Sophia	18.0	female	A	8.1
10	Jackson	19.0	male	B	7.5
13	Ava	21.0	female	A	8.4

	Name	Age	Gender	Grade	GPA	Student_Info
0	Ethan	0.666667	male	A	0.3125	Ethan (A)
1	Liam	1.000000	male	A	0.3750	Liam (A)
3	Grace	1.000000	female	A	0.9375	Grace (A)
4	Wilson	0.666667	male	A	1.0000	Wilson (A)
9	Sophia	0.000000	female	A	0.3750	Sophia (A)

Program 4:

Aim: 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.

PROGRAM:

```
import numpy as
np import pandas
as pd
from sklearn import metrics
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import
train_test_split from sklearn.preprocessing import
OneHotEncoder # reading the dataset
dataset = pd.read_csv('PlayTennis.csv')
features = ['Outlook', 'Temperature', 'Humidity',
'Wind'] X = dataset[features]
Y = dataset.PlayTennis
encoder = OneHotEncoder(sparse_output=False, handle_unknown='ignore')
X_encoded = pd.DataFrame(encoder.fit_transform(X),
columns=encoder.get_feature_names_out(featur es))
# splitting the dataset
X_train, X_test, y_train, y_test = train_test_split(X_encoded, Y, test_size=0.30,
random_state=100) # building the decision tree
dtree = DecisionTreeClassifier(criterion="entropy",
random_state=100) dtree.fit(X_train, y_train)
y_pred = dtree.predict(X_test)
# classifying the new instance based on the training data
def classify_new_instance(outlook, temperature, humidity, wind,
    encoder): instance = [[outlook, temperature, humidity, wind]]
    instance_df = pd.DataFrame(instance, columns=features)
    instance_encoded = encoder.transform(instance_df)
    feature_names =
    encoder.get_feature_names_out(features)
    instance_encoded_df = pd.DataFrame(instance_encoded,
    columns=feature_names) prediction = dtree.predict(instance_encoded_df)
    return prediction[0]
# predicting the class of new instance
pred = classify_new_instance("Rain","Mild","High","Strong",
encoder=encoder) print("Prediction:", pred)
print("Accuracy:", metrics.accuracy_score(y_test, y_pred))
```

OUTPUT:

Prediction:
No Accuracy:
0.6

Program 5:

Aim: Demonstrate the working of the Random Forest algorithm. Use an appropriate data set for building and apply this knowledge to classify a new sample.

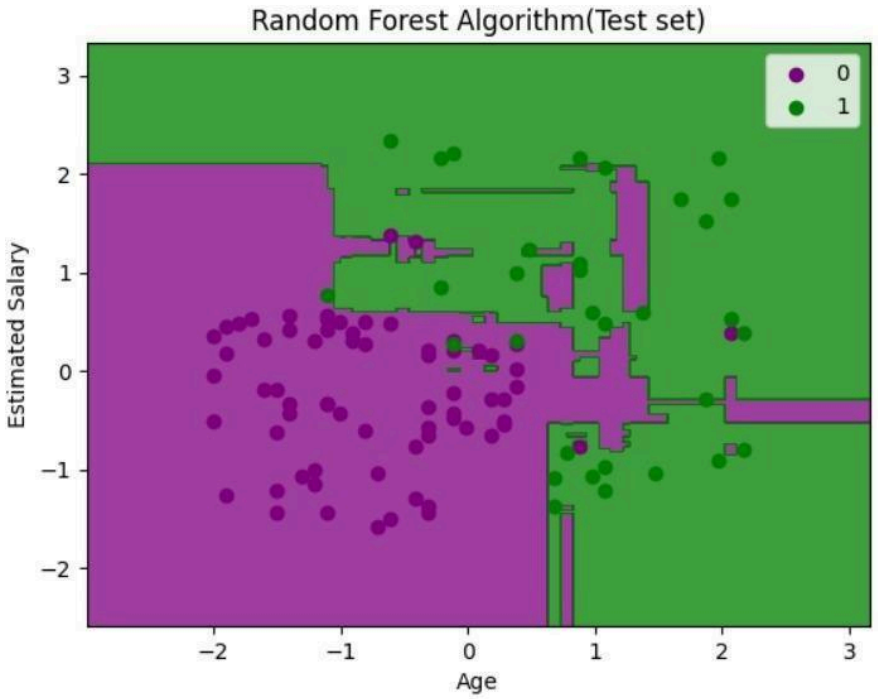
PROGRAM:

```
# importing
libraries import
numpy as np
import matplotlib.pyplot as
plt import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import
RandomForestClassifier from sklearn.metrics import
confusion_matrix
from matplotlib.colors import
ListedColormap #importing datasets
data_set= pd.read_csv('User_data.csv')
#Extracting Independent and dependent
Variable x= data_set.iloc[:, [2,3]].values
y= data_set.iloc[:, 4].values
# Splitting the dataset into training and test set.
x_train, x_test, y_train, y_test= train_test_split(x, y, test_size= 0.25,
random_state=0) #feature Scaling
st_x= StandardScaler()
x_train=
st_x.fit_transform(x_train) x_test=
st_x.transform(x_test)
#Fitting Decision tree classifier to the training set
classifier= RandomForestClassifier(n_estimators= 10,
criterion="entropy") classifier.fit(x_train, y_train)
#Predicting the test set result
y_pred=
classifier.predict(x_test)
#Creating the Confusion matrix
cm= confusion_matrix(y_test,
y_pred) #Visulaizing the test set
result
x_set, y_set = x_test, y_test
x1, x2 = np.meshgrid(np.arange(start = x_set[:, 0].min() - 1, stop = x_set[:, 0].max() +
1, step =0.01),
np.arange(start = x_set[:, 1].min() - 1, stop = x_set[:, 1].max() + 1, step = 0.01))
plt.contourf(x1, x2, classifier.predict(np.array([x1.ravel(),
```

```
x2.ravel()]).T).reshape(x1. shape), alpha = 0.75, cmap =
ListedColormap(('purple','green' )))
plt.xlim(x1.min(), x1.max())
plt.ylim(x2.min(), x2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(x_set[y_set == j, 0], x_set[y_set == j, 1],
        c = ListedColormap(('purple', 'green'))(i), label = j)
```

```
# plotting the random forest
plt.title('Random Forest Algorithm(Test set)')
plt.xlabel('Age')
plt.ylabel('Estimated
Salary') plt.legend()
plt.show()
```

OUTPUT:



Program 6:

Aim: Implement the Naive 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.model_selection import
train_test_split from sklearn.naive_bayes import
GaussianNB
from sklearn import metrics
df = pd.read_csv("pima_indian.csv")
feature_col_names = ['num_preg', 'glucose_conc', 'diastolic_bp', 'thickness', 'insulin', 'bmi',
'diab_pred', 'age']
predicted_class_names =
['diabetes']
X = df[feature_col_names].values # these are factors for the prediction
y = df[predicted_class_names].values # this is what we want to predict
#splitting the dataset into train and test data
xtrain,xtest,ytrain,ytest=train_test_split(X,y,test_size=0.33)
print ('\n the total number of Training
Data:',ytrain.shape) print ('\n the total number of Test
Data:',ytest.shape)
# Training Naive Bayes (NB) classifier on training
data. clf = GaussianNB().fit(xtrain,ytrain.ravel())
predicted = clf.predict(xtest)
predictTestData=
clf.predict([[6,148,72,35,0,33.6,0.627,50]]) #printing
Confusion matrix, accuracy, Precision and Recall print('\n
Confusion matrix')
print(metrics.confusion_matrix(ytest,predicted))
print('\n Accuracy of the classifier
is',metrics.accuracy_score(ytest,predicted)) print('\n The value of Precision',
metrics.precision_score(ytest,predicted)) print('\n The value of Recall',
metrics.recall_score(ytest,predicted)) print("Predicted Value for individual
Test Data:", predictTestData)
```

OUTPUT:

```
the total number of Training Data: (514,
1) the total number of Test Data: (254, 1)
Confusion
matrix [[147 23]
[ 39 45]]
```

Accuracy of the classifier is
0.7559055118110236 The value of Precision
0.6617647058823529
The value of Recall 0.5357142857142857
Predicted Value for individual Test Data: [1]

Program 7:

Aim: demonstate the text classifier using naive bayes classifier algorithm

PROGRAM:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import
CountVectorizer from sklearn.naive_bayes import
MultinomialNB
from sklearn import metrics
msg=pd.read_csv("naivetext.csv",names=['message','label']
) msg['labelnum']=msg.label.map({'pos':1,'neg':0})
X=msg.messag
e
y=msg.labelnu
m
#splitting the dataset into train and test data
xtrain,xtest,ytrain,ytest=train_test_split(X,y
)
#output of the words or Tokens in the text
documents count_vect = CountVectorizer()
xtrain_dtm =
count_vect.fit_transform(xtrain)
xtest_dtm=count_vect.transform(xtest)
print('\n The words or Tokens in the text documents \n')
# if get_feature_names_out() gives error then replace it with
get_feature_names() print(count_vect.get_feature_names_out())
df=pd.DataFrame(xtrain_dtm.toarray(),
columns=count_vect.get_feature_names_out())
# Training Naive Bayes (NB) classifier on training
data. clf = MultinomialNB().fit(xtrain_dtm,ytrain)
predicted = clf.predict(xtest_dtm)
#printing accuracy, Confusion matrix, Precision and Recall
print('\n Accuracy of the classifier
is',metrics.accuracy_score(ytest,predicted)) print('\n Confusion matrix')
print(metrics.confusion_matrix(ytest,predicted))
print('\n The value of Precision',
metrics.precision_score(ytest,predicted)) print('\n The value of Recall',
metrics.recall_score(ytest,predicted))
```

OUTPUT:

```
The words or Tokens in the text documents
['about' 'am' 'an' 'and' 'awesome' 'bad' 'beers' 'best' 'boss' 'can'
```

'dance' 'deal' 'do' 'enemy' 'feel' 'good' 'horrible' 'house' 'is'
'juice' 'like' 'locality' 'love' 'my' 'not' 'of' 'place' 'restaurant'
'sandwich' 'sick' 'stay' 'taste' 'that' 'the' 'these' 'this' 'tired' 'to'
'today'
'very' 'view' 'went' 'what' 'with'
'work'] Accuracy of the classifier is
1.0 Confusion matrix

[[2 0]
[0 3]]
The value of Precision
1.0 The value of Recall
1.0

Program 8:

Aim: Construct a Bayesian network considering medical data. Use this model to demonstrate the diagnosis of heart patients using standard Heart Disease Data Set.

PROGRAM:

```
import numpy as
np import pandas
as pd import csv
from pgmpy.estimators import
MaximumLikelihoodEstimator from pgmpy.models import
BayesianModel
from pgmpy.inference import
VariableElimination heartDisease =
pd.read_csv('heart.csv') heartDisease =
heartDisease.replace('?',np.nan)
model=
BayesianModel([('age','heartdisease'),('sex','heartdisease'),('exang','heartdisease'),('cp','heartdise
ase'),('heartdisease','restecg'),('heartdisease','chol')])
print('\nLearning CPD using Maximum likelihood estimators')
model.fit(heartDisease,estimator=MaximumLikelihoodEstimator)
print('\n Inferencing with Bayesian Network:')
HeartDiseasetest_infer = VariableElimination(model)
print('\n 1. Probability of HeartDisease given evidence= restecg')
q1=HeartDiseasetest_infer.query(variables=['heartdisease'],evidence={'restecg':1}
) print(q1)
print('\n 2. Probability of HeartDisease given evidence= cp ')
q2=HeartDiseasetest_infer.query(variables=['heartdisease'],evidence={'cp':2}
) print(q2)
```

OUTPUT:

Learning CPD using Maximum likelihood
estimators Inferencing with Bayesian Network:

1. Probability of HeartDisease given evidence= restecg

heartdisease	phi(heartdisease)
heartdisease(0)	0.1012
heartdisease(1)	0.0000
heartdisease(2)	0.2392
heartdisease(3)	0.2015

| heartdisease(4) | 0.4581 |
+_____+_____+

2. Probability of HeartDisease given evidence= cp

heartdisease	phi(heartdisease)
heartdisease(0)	0.3610
heartdisease(1)	0.2159
heartdisease(2)	0.1373
heartdisease(3)	0.1537
heartdisease(4)	0.1321

Program 9:

Aim: Demonstrate the working of EM algorithm to cluster a set of data stored in a .CSV file.

PROGRAM:

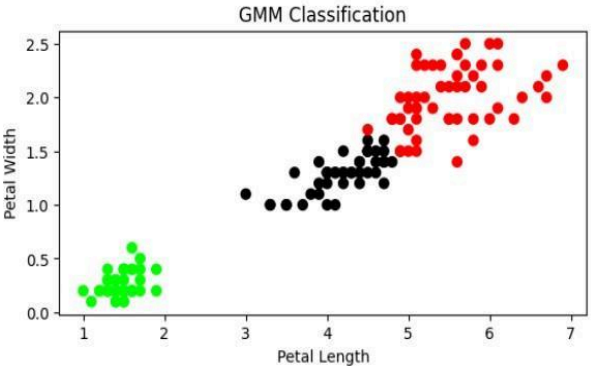
```
import matplotlib.pyplot as
plt from sklearn import
datasets
from sklearn.cluster import
KMeans import sklearn.metrics as
sm
import pandas as
pd import numpy
as np
from sklearn import preprocessing
from sklearn.mixture import
GaussianMixture iris = datasets.load_iris()
X = pd.DataFrame(iris.data)
X.columns =
['Sepal_Length','Sepal_Width','Petal_Length','Petal_Width'] y =
pd.DataFrame(iris.target)
y.columns = ['Targets']
model =
KMeans(n_clusters=3)
model.fit(X)
plt.figure(figsize=(14,7))
colormap = np.array(['red', 'lime',
'black']) # Plot the Original
Classifications plt.subplot(1, 2, 1)
plt.scatter(X.Petal_Length, X.Petal_Width, c=colormap[y.Targets], s=40)
plt.title('Real Classification')
plt.xlabel('Petal
Length')
plt.ylabel('Petal Width')
# Plot the Models
Classifications plt.subplot(1, 2,
2)
plt.scatter(X.Petal_Length, X.Petal_Width, c=colormap[model.labels_],
s=40) plt.title('K Mean Classification')
plt.xlabel('Petal Length')
plt.ylabel('Petal Width')
print('The accuracy score of K-Mean: ',sm.accuracy_score(y, model.labels_))
print('The Confusion matrix of K-Mean: ',sm.confusion_matrix(y,
model.labels_)) scaler = preprocessing.StandardScaler()
scaler.fit(X)
```

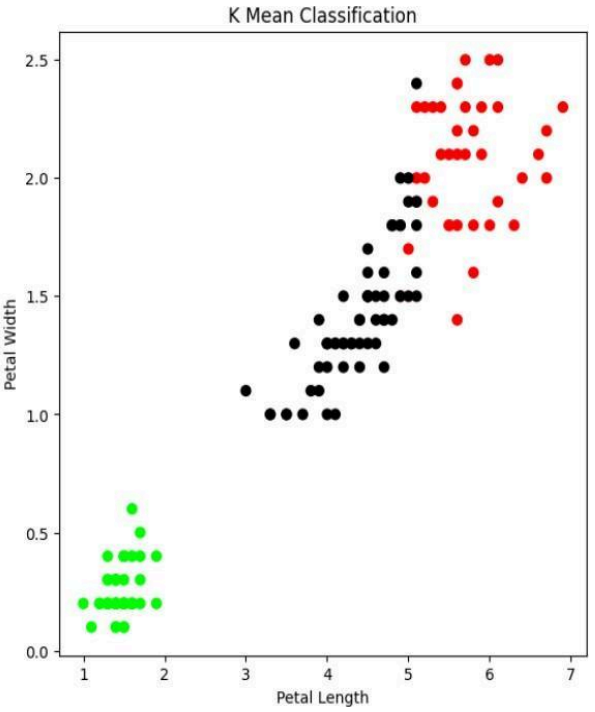
```
xsa = scaler.transform(X)
xs = pd.DataFrame(xsa, columns =
X.columns) gmm =
GaussianMixture(n_components=3)
gmm.fit(xs)
y_gmm = gmm.predict(xs)
plt.subplot(2, 2, 3)
plt.scatter(X.Petal_Length, X.Petal_Width, c=colormap[y_gmm], s=40)
```

```
plt.title('GMM
Classification')
plt.xlabel('Petal Length')
plt.ylabel('Petal Width')
print('The accuracy score of EM: ',sm.accuracy_score(y, y_gmm))
print('The Confusion matrix of EM: ',sm.confusion_matrix(y,
y_gmm))
```

OUTPUT:

```
The accuracy score of K-Mean: 0.09333333333333334
The Confusion matrix of K-Mean:  [[ 0 50 0]
 [ 2 0 48]
 [36 0 14]]
The accuracy score of EM:  0.0
The Confusion matrix of EM:  [[ 0 50  0]
 [ 5 0 45]
 [50 0 0]]
```





Program 10:

Aim: Demonstrate the working of SVM classifier for a suitable data set.

PROGRAM:

```
import numpy as np
import matplotlib.pyplot as
plt from sklearn import
datasets
from sklearn.model_selection import
train_test_split from sklearn.svm import SVC
from sklearn.metrics import
accuracy_score # Load the dataset
(example: Iris dataset) iris =
datasets.load_iris()
X = iris.data[:, :2] # Consider only the first two features for simplicity
y = iris.target
# Select only two classes for binary
classification X = X[y != 2]
y = y[y != 2]
# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X,
y, test_size=0.2, random_state=42)
# Create an SVM classifier
svm_classifier = SVC(kernel='linear')
# Train the classifier on the training
data svm_classifier.fit(X_train, y_train)
# Make predictions on the testing data
y_pred = svm_classifier.predict(X_test)
# Calculate the accuracy of the
classifier
accuracy = accuracy_score(y_test,
y_pred) print("Accuracy:", accuracy)
# Plot the decision boundary and support vectors
plt.scatter(X[:, 0], X[:, 1], c=y, cmap=plt.cm.Paired,
edgecolors='k') ax = plt.gca()
xlim =
ax.get_xlim() ylim
= ax.get_ylim()
# Create a meshgrid to plot the decision
boundary xx = np.linspace(xlim[0], xlim[1], 30)
yy = np.linspace(ylim[0], ylim[1],
30) YY, XX = np.meshgrid(yy, xx)
xy = np.vstack([XX.ravel(), YY.ravel()]).T
```

```
Z =
svm_classifier.decision_function(xy).reshape(XX.shape) #
Plot the decision boundary and margins
ax.contour(XX, YY, Z, colors='k', levels=[-1, 0, 1], alpha=0.5, linestyles=['--', '-', '--'])
ax.scatter(svm_classifier.support_vectors_[:, 0], svm_classifier.support_vectors_[:,
1], s=100, linewidth=1, facecolors='none', edgecolors='k')
```

```
plt.xlabel('Feature 1')
plt.ylabel('Feature 2')
plt.title('SVM Binary
Classifier') plt.show()
```

OUTPUT:

```
Accuracy: 1.0
```