B.M.S. COLLEGE OF ENGINEERING BENGALURU
Autonomous Institute, Affiliated to VTU

Lab Record

MACHINE LEARNING

Submitted in partial fulfillment for the 6th Semester Laboratory

Bachelor of Technology
in
Computer Science and Engineering

Submitted by:

Arjun A.S

1BM18CS019

Department of Computer Science and Engineering
B.M.S. College of Engineering
Bull Temple Road, Basavanagudi, Bangalore 560 019
Mar-June 2021

B.M.S. COLLEGE OF ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the Machine Learning (20CS6PCMAL) laboratory has been carried out by Arjun A.S (1BM18CS019) during the 6th Semester Mar-June-2021.

Signature of the Faculty Incharge:

Prof. Saritha A.N
Assistant Professor
Department of Computer Science and Engineering
B.M.S. College of Engineering, Bangalore

Table of Contents

Program 1:
Implement and demonstrate the FIND-S algorithm for finding the most specific hypothesis based on a given set of training data samples.

<u>Code:</u>

```
#Program to implement find S algorithm

#most general hypothesis [?,?,?]

import pandas as pd
import numpy as np


data=pd.read_csv("data.csv")
print(data)
H=np.array(data)[:,:-1]
t=np.array(data)[:,-1]
print("The attributes are : ")
print(H)
print("The target is : ")
print(t)

h=["*","*"] #most specific hypothesis
#H=[["rainy","Normal"],["sunny","Normal"],["cloudy","Normal"]] #data set
#t=["yes","yes","no"] #values for dataset
def training_example(H,t):
    for z,x in list(enumerate(H)):
        if t[z]=="yes":
            for i in range(len(x)):
                if h[i]=="*" and x[i]:
                    h[i]=x[i]
                elif h[i]!= x[i] and h[i]!="?":
                        h[i]="?"

    return h



print("The Hypothesis is : ")
print(training_example(H,t))
```

Dataset:

Weather	Humidity	Goes
Rainy	Normal	yes
Sunny	Normal	yes
Cloudy	Normal	No

Output:

```
The attributes are :
[['Rainy' 'Normal']
 ['Sunny' 'Normal']
 ['Cloudy' 'Normal']]
The target is :
['yes' 'yes' 'No']


The Hypothesis is :
['?', 'Normal']
```

Program 2:
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.

<u>Code:</u>

```
import csv
data=[]
print("The Data Set is: ")
with open("enjoysport.csv",'r') as csvfile :
          fdata = csv.reader(csvfile)
          for x in fdata:
                data.append(x)
                print(x)
num_att = len(data[0]) - 1
print('--------------------')
S = ['0']*num_att    #Specific Boundary
G = ['?']*num_att  #General Boundary
print("S0= ",S)
print("G0= ",G)
temp = []

for i in range(0, num_att):
    S[i] = data[1][i]
print('--------------------')

for i in range(1, len(data)):
    if data[i][num_att] == 'Yes':
            for j in range(0, num_att):
                if S[j]!=data[i][j]:
                    S[j]='?'
            for j in range(0, num_att):
                for k in range(0, len(temp)):
                    if temp[k][j]!=S[j] and temp[k][j]!='?':
                        del temp[k]
    if data[i][num_att] == 'No':
        for j in range(0, num_att):
            if data [i][j]!=S[j] and S[j]!='?':
                 G[j] = S[j]
                 temp.append(G)
                 G = ['?']*num_att


    print("S",i,"= ",S)

    if len(temp)==0:
             print ("G",i," =",G)
    else:
             print("G",i," =",temp)
    print("-----------------")
```

Dataset:

Sky	Airtemp	Humidity	Wind	Water	Forecast	Enjoysport
Sunny	Warm	Normal	Strong	Warm	Same	Yes
Sunny	Warm	High	Strong	Warm	Same	Yes
Rainy	Cold	High	Strong	Warm	Change	No
Sunny	Warm	High	Strong	Cool	Change	Yes

Output:

```
The Data Set is:
['Sky', 'Airtemp', 'Humidity', 'Wind', 'Water', 'Forecast', 'Enjoysport']
['Sunny', 'Warm', 'Normal', 'Strong', 'Warm', 'Same', 'Yes']
['Sunny', 'Warm', 'High', 'Strong', 'Warm', 'Same', 'Yes']
['Rainy', 'Cold', 'High', 'Strong', 'Warm', 'Change', 'No']
['Sunny', 'Warm', 'High', 'Strong', 'Cool', 'Change', 'Yes']
--------------------
S0= ['0', '0', '0', '0', '0', '0']
G0= ['?', '?', '?', '?', '?', '?']
--------------------
S 1 = ['Sunny', 'Warm', 'Normal', 'Strong', 'Warm', 'Same']
G 1 = ['?', '?', '?', '?', '?', '?']
-----------------
S 2 = ['Sunny', 'Warm', '?', 'Strong', 'Warm', 'Same']
G 2 = ['?', '?', '?', '?', '?', '?']
-----------------
S 3 = ['Sunny', 'Warm', '?', 'Strong', 'Warm', 'Same']
G 3 = [['Sunny', '?', '?', '?', '?', '?'], ['?', 'Warm', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', 'Same']]
-----------------
S 4 = ['Sunny', 'Warm', '?', 'Strong', '?', '?']
G 4 = [['Sunny', '?', '?', '?', '?', '?'], ['?', 'Warm', '?', '?', '?', '?']]
-----------------
```

Program 3:
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.

Code:

```
import pandas as pd
import math
import numpy as np


data=pd.read_csv("dataset_tennis.csv")
print(data)
features = [feat for feat in data]
features.remove("answer")

class Node:
    def __init__(self):
        self.children = []
        self.value = ""
        self.isLeaf = False
        self.pred = ""

def entropy(examples):
    positive = 0.0
    negative = 0.0
    for _, row in examples.iterrows():
        if row["answer"] == "yes":
            positive += 1
        else:
            negative += 1
    if positive == 0.0 or negative == 0.0:
        return 0.0
    else:
        p = positive / (positive + negative)
        n = negative / (positive + negative)
        return -(p * math.log(p, 2) + n * math.log(n, 2))  #formula to calcu
late Entropy

def gain_calc(examples, attr):  #Function to Calculate Gain for a given attr
ibute
    uniq = np.unique(examples[attr])

    gain = entropy(examples)

    for u in uniq:
        subdata = examples[examples[attr] == u]

        sub_e = entropy(subdata)
        gain -= (float(len(subdata)) / float(len(examples))) * sub_e
```

```python
    return gain
def ID3(examples, attrs):
    root = Node()

    max_gain = 0
    max_feat = ""
    for feature in attrs:

        gain = gain_calc(examples, feature)
        if gain > max_gain:
            max_gain = gain
            max_feat = feature
    root.value = max_feat

    uniq = np.unique(examples[max_feat])

    for u in uniq:

        subdata = examples[examples[max_feat] == u]

        if entropy(subdata) == 0.0:
            newNode = Node()
            newNode.isLeaf = True
            newNode.value = u
            newNode.pred = np.unique(subdata["answer"])
            root.children.append(newNode)
        else:
            dummyNode = Node()
            dummyNode.value = u
            new_attrs = attrs.copy()
            new_attrs.remove(max_feat)
            child = ID3(subdata, new_attrs)
            dummyNode.children.append(child)
            root.children.append(dummyNode)
    return root

def printTree(root: Node, depth=0):
    for i in range(depth):
        print("\t", end="")
    print(root.value,"", end="")
    if root.isLeaf:
        print("-> ", root.pred)
    print()

    for child in root.children:
        printTree(child, depth + 1)

root = ID3(data, features)
printTree(root)
```

Dataset:

	A	B	C	D	E
1	Outlook	Temperature	Humidity	Wind	answer
2	sunny	hot	high	weak	no
3	sunny	hot	high	strong	no
4	overcast	hot	high	weak	yes
5	rain	mild	high	weak	yes
6	rain	cool	normal	weak	yes
7	rain	cool	normal	strong	no
8	overcast	cool	normal	strong	yes
9	sunny	mild	high	weak	no
10	sunny	cool	normal	weak	yes
11	rain	mild	normal	weak	yes
12	sunny	mild	normal	strong	yes
13	overcast	mild	high	strong	yes
14	overcast	hot	normal	weak	yes
15	rain	mild	high	strong	no

Output:

```
Outlook
        overcast ->  ['yes']

        rain
                Wind
                        strong ->  ['no']

                        weak ->  ['yes']

        sunny
                Humidity
                        high ->  ['no']

                        normal ->  ['yes']
```

Program 4:

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.

Code:

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn import metrics

#This program takes certain parameters from patients and identifies whether
they have diabetes are not
dataf = pd.read_csv("./bayes_classifier_data.csv")
feature_col_names = ['num_preg', 'glucose_conc', 'diastolic_bp', 'thickness'
, 'insulin', 'bmi', 'diab_pred', 'age']
predicted_class_names = ['diabetes']

X = dataf[feature_col_names].values
y = dataf[predicted_class_names].values

print(dataf.head)

xtrain,xtest,ytrain,ytest=train_test_split(X,y,test_size=0.33)

print ('\nThe total number of Training Data:',ytrain.shape)
print ('The total number of Test Data:',ytest.shape)
classif = GaussianNB().fit(xtrain,ytrain.ravel())

predicted = classif.predict(xtest)

predictTestData= classif.predict([[5,148,72,35,0,32.6,0.543,50]])

print('\nConfusion matrix')
print(metrics.confusion_matrix(ytest,predicted))

print('\nAccuracy of the classifier:',metrics.accuracy_score(ytest,predicted
))

print('The value of Precision:', metrics.precision_score(ytest,predicted))

print('The value of Recall:', metrics.recall_score(ytest,predicted))

print("Predicted Value for individual Test Data:", predictTestData)
```

Dataset:

num_preg	glucose_conc	diastolic_bp	thickness	insulin	bmi	diab_pred	age	diabetes
6	148	72	35	0	33.6	0.627	50	1
1	85	66	29	0	26.6	0.351	31	0
8	183	64	0	0	23.3	0.672	32	1
1	89	66	23	94	28.1	0.167	21	0
0	137	40	35	168	43.1	2.288	33	1
5	116	74	0	0	25.6	0.201	30	0
3	78	50	32	88	31	0.248	26	1
10	115	0	0	0	35.3	0.134	29	0
2	197	70	45	543	30.5	0.158	53	1

The entire dataset can be found in this link:
https://github.com/ArjunAS861/ML_1BM18CS019/blob/main/ML_Lab4/bayes_classifier_data.csv

Output:

```
<bound method NDFrame.head of      num_preg  glucose_conc  diastolic_bp  thickness  insulin   bmi  \
0           6           148            72         35        0  33.6
1           1            85            66         29        0  26.6
2           8           183            64          0        0  23.3
3           1            89            66         23       94  28.1
4           0           137            40         35      168  43.1
..        ...           ...           ...        ...      ...   ...
763        10           101            76         48      180  32.9
764         2           122            70         27        0  36.8
765         5           121            72         23      112  26.2
766         1           126            60          0        0  30.1
767         1            93            70         31        0  30.4

     diab_pred  age  diabetes
0        0.627   50         1
1        0.351   31         0
2        0.672   32         1
3        0.167   21         0
4        2.288   33         1
..         ...  ...       ...
763      0.171   63         0
764      0.340   27         0
765      0.245   30         0
766      0.349   47         1
767      0.315   23         0

[768 rows x 9 columns]>

The total number of Training Data: (514, 1)
The total number of Test Data: (254, 1)

Confusion matrix
[[147  29]
 [ 34  44]]

Accuracy of the classifier: 0.7519685039370079
The value of Precision: 0.6027397260273972
The value of Recall: 0.5641025641025641
Predicted Value for individual Test Data: [1]
```

Program 5:
Write a program to construct a Bayesian network considering training data. Use this model to make predictions.

Code:

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
from pgmpy.estimators import MaximumLikelihoodEstimator
from pgmpy.models import BayesianModel
from pgmpy.inference import VariableElimination

heart_Disease = pd.read_csv('/home/bayes_heart_disease_dataset.csv')
heart_Disease = heart_Disease.replace('?',np.nan)

print('Sample instances from the dataset are given below')
print(heart_Disease.head())

print('\n Attributes and datatypes')
print(heart_Disease.dtypes)

model= BayesianModel([('age','heartdisease'),('sex','heartdisease'),('exang'
,'heartdisease'),('cp','heartdisease'),('heartdisease','restecg'),('heartdis
ease','chol')])
print('\nLearning CPD using Maximum likelihood estimators')
model.fit(heart_Disease,estimator=MaximumLikelihoodEstimator)

print('\n Inferencing with Bayesian Network:')
Heart_Disease_test_infer = VariableElimination(model)

print('\n 1. Probability of HeartDisease given evidence= restecg')
q1=Heart_Disease_test_infer.query(variables=['heartdisease'],evidence={'rest
ecg':1})
print(q1)

print('\n 2. Probability of HeartDisease given evidence= cp ')
q2=Heart_Disease_test_infer.query(variables=['heartdisease'],evidence={'cp':
2})
print(q2)
```

Dataset:

age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	heartdisease
63	1	1	145	233	1	2	150	0	2.3	3	0	6	0
67	1	4	160	286	0	2	108	1	1.5	2	3	3	2
67	1	4	120	229	0	2	129	1	2.6	2	2	7	1
37	1	3	130	250	0	0	187	0	3.5	3	0	3	0
41	0	2	130	204	0	2	172	0	1.4	1	0	3	0
56	1	2	120	236	0	0	178	0	0.8	1	0	3	0
62	0	4	140	268	0	2	160	0	3.6	3	2	3	3
57	0	4	120	354	0	0	163	1	0.6	1	0	3	0
63	1	4	130	254	0	2	147	0	1.4	2	1	7	2

The entire dataset can be found in this link:
https://github.com/ArjunAS861/ML_1BM18CS019/blob/main/ML_Lab6/bayes_heart_disease_dataset.csv

Output:

```
-                    -

 Attributes and datatypes
age                  int64
sex                  int64
cp                   int64
trestbps             int64
chol                 int64
fbs                  int64
restecg              int64
thalach              int64
exang                int64
oldpeak            float64
slope                int64
ca                  object
thal                object
heartdisease         int64
dtype: object
```

Fig 5.1 Attributes

```
Learning CPD using Maximum likelihood estimators
Finding Elimination Order: : 100%|██████████| 5/5 [00:00<00:00, 740.49it/s]
Eliminating: age: 100%|██████████| 5/5 [00:00<00:00, 75.39it/s]
 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

Fig 5.2 Query 1

```
 2. Probability of HeartDisease given evidence= cp
Finding Elimination Order: : 100%|██████████| 5/5 [00:00<00:00, 565.80it/s]
Eliminating: age: 100%|██████████| 5/5 [00:00<00:00, 121.08it/s]
```

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

Fig 5.3 Query 2

Program 6:
Apply k-Means algorithm to cluster a set of data stored in a .CSV file.

Code:

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
from sklearn import datasets
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.cluster import KMeans
from sklearn.metrics import accuracy_score, confusion_matrix

iris = datasets.load_iris()
print(iris['data'])
print(iris['target'])


sepal_X = iris.data[:, :2]
petal_X = iris.data[:, 2:]
y = iris.target
categories = len(iris.target_names)
fig, axes = plt.subplots(1, 2, figsize=(16,8))
axes[0].scatter(sepal_X[:, 0], sepal_X[:, 1], c=y, cmap='gist_rainbow', edge
color='k', s=150)
axes[1].scatter(petal_X[:, 0], petal_X[:, 1], c=y, cmap='gist_rainbow', edge
color='k', s=150)
axes[0].set_xlabel('Sepal length', fontsize=18)
axes[0].set_ylabel('Sepal width', fontsize=18)
axes[1].set_xlabel('Petal length', fontsize=18)
axes[1].set_ylabel('Petal width', fontsize=18)
axes[0].tick_params(direction='in', length=10, width=5, colors='k', labelsiz
e=20)
axes[1].tick_params(direction='in', length=10, width=5, colors='k', labelsiz
e=20)
axes[0].set_title('Sepal', fontsize=18)
axes[1].set_title('Petal', fontsize=18)
plt.show()


model_sepal = KMeans(n_clusters=3)
model_sepal.fit(sepal_X)
model_petal = KMeans(n_clusters=3)
model_petal.fit(petal_X)

def plot_centers(sepal_centers, petal_centers):
    plt.scatter([point[0] for point in sepal_centers], [point[1] for point i
n sepal_centers])
    plt.title('Sepal KMeans Centers')
    plt.show()

    plt.scatter([point[0] for point in petal_centers], [point[1] for point i
n petal_centers])
```

```
    plt.title('Petal KMeans Centers')
    plt.show()

def plot_actualvpredicted(X, y, predicted, part):
    fig, axes = plt.subplots(1, 2, figsize=(16,8))
    axes[0].scatter(X[:, 0], X[:, 1], c=y, cmap='gist_rainbow', edgecolor='k
', s=150)
    axes[1].scatter(X[:, 0], X[:, 1], c=predicted, cmap='jet', edgecolor='k'
, s=150)
    axes[0].set_xlabel(f'{part} length', fontsize=18)
    axes[0].set_ylabel(f'{part} width', fontsize=18)
    axes[1].set_xlabel(f'{part} length', fontsize=18)
    axes[1].set_ylabel(f'{part} width', fontsize=18)
    axes[0].tick_params(direction='in', length=10, width=5, colors='k', labe
lsize=20)
    axes[1].tick_params(direction='in', length=10, width=5, colors='k', labe
lsize=20)
    axes[0].set_title('Actual', fontsize=18)
    axes[1].set_title('Predicted', fontsize=18)
    plt.show()

def plot_confusion(accuracy, confusion, part):
    print(f'{part} Accuracy: {accuracy}')

    fig, ax = plt.subplots()
    im = ax.imshow(confusion)

    ax.set_xticks(range(categories))
    ax.set_yticks(range(categories))
    ax.set_xticklabels(iris.target_names)
    ax.set_yticklabels(iris.target_names)

    plt.setp(ax.get_xticklabels(), rotation=45, ha="right",
             rotation_mode="anchor")

    for i in range(categories):
        for j in range(categories):
            text = ax.text(j, i, confusion[i, j],
                           ha="center", va="center", color="w")

    ax.set_title(f"{part} Confusion Matrix (Actual / Predicted)")
    fig.tight_layout()
    plt.show()



model_sepal = KMeans(n_clusters=3)
model_sepal.fit(sepal_X)
model_petal = KMeans(n_clusters=3)
model_petal.fit(petal_X)

plot_actualvpredicted(sepal_X, y, sepal_labels, 'Sepal')
plot_actualvpredicted(petal_X, y, petal_labels, 'Petal')
```

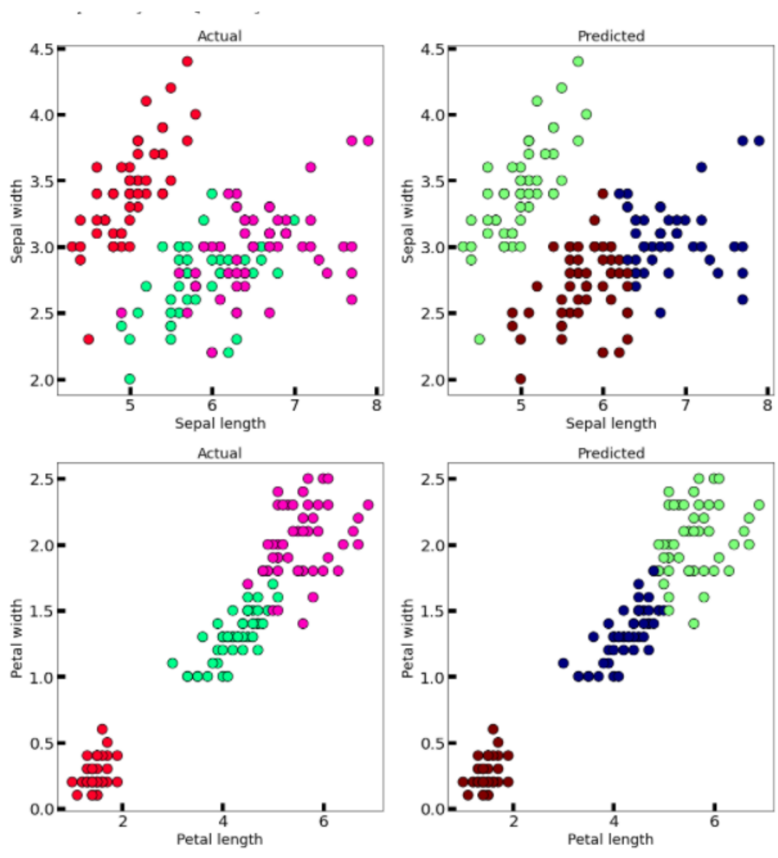
```
sepal_labels = model_sepal.labels_ petal_labels = model_petal.labels_ print(
sepal_labels, petal_labels, sep='\n')


for n_clusters in range(1, 4):
    print(f'\n\n\n\n============{n_clusters} clusters=================')

    model_sepal = KMeans(n_clusters=n_clusters)
    model_sepal.fit(sepal_X)
    model_petal = KMeans(n_clusters=n_clusters)
    model_petal.fit(petal_X)

    sepal_centers = model_sepal.cluster_centers_
    petal_centers = model_petal.cluster_centers_

    plot_centers(sepal_centers, petal_centers)

    sepal_labels = model_sepal.labels_
    petal_labels = model_petal.labels_

    plot_actualvpredicted(sepal_X, y, sepal_labels, 'Sepal')
    plot_actualvpredicted(petal_X, y, petal_labels, 'Petal')

    sepal_accuracy = accuracy_score(y, sepal_labels)
    petal_accuracy = accuracy_score(y, petal_labels)
    sepal_confusion = confusion_matrix(y, sepal_labels)
    petal_confusion = confusion_matrix(y, petal_labels)

    plot_confusion(sepal_accuracy, sepal_confusion, 'Sepal')
    plot_confusion(petal_accuracy, petal_confusion, 'Petal')
```

Dataset:

5.1	3.5	1.4	0.2	Iris-setosa
4.9	3	1.4	0.2	Iris-setosa
4.7	3.2	1.3	0.2	Iris-setosa
4.6	3.1	1.5	0.2	Iris-setosa
5	3.6	1.4	0.2	Iris-setosa
5.4	3.9	1.7	0.4	Iris-setosa
4.6	3.4	1.4	0.3	Iris-setosa
5	3.4	1.5	0.2	Iris-setosa
4.4	2.9	1.4	0.2	Iris-setosa
4.9	3.1	1.5	0.1	Iris-setosa

The entire dataset can be found in this link:
https://github.com/ArjunAS861/ML_1BM18CS019/blob/main/ML_Prog7/iris.csv

Output:



Program 7:

Apply EM algorithm to cluster a set of data stored in a .CSV file. Compare the results of k-Means algorithm and EM algorithm.

Code:

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

from sklearn.cluster import KMeans
from sklearn.mixture import GaussianMixture
import sklearn.metrics as metrics

names = ['Sepal_Length','Sepal_Width','Petal_Length','Petal_Width', 'Class']

dataset = pd.read_csv("iris.csv", names=names)
print(dataset.head())
X = dataset.iloc[:, :-1]

label = {'Iris-setosa': 0,'Iris-versicolor': 1, 'Iris-virginica': 2}

y = [label[c] for c in dataset.iloc[:, -1]]

plt.figure(figsize=(14,7))
colormap=np.array(['red','lime','black'])

# REAL PLOT
plt.subplot(1,3,1)
plt.title('Real')
plt.scatter(X.Petal_Length,X.Petal_Width,c=colormap[y])

# K-PLOT
model=KMeans(n_clusters=3, random_state=0).fit(X)
plt.subplot(1,3,2)
plt.title('KMeans')
plt.scatter(X.Petal_Length,X.Petal_Width,c=colormap[model.labels_])

print('The accuracy score of K-Mean: ',metrics.accuracy_score(y, model.label
s_))
print('The Confusion matrixof K-Mean:\n',metrics.confusion_matrix(y, model.l
abels_))

# GMM PLOT
gmm=GaussianMixture(n_components=3, random_state=0).fit(X)
y_cluster_gmm=gmm.predict(X)
plt.subplot(1,3,3)
plt.title('GMM Classification')
plt.scatter(X.Petal_Length,X.Petal_Width,c=colormap[y_cluster_gmm])

print('The accuracy score of EM: ',metrics.accuracy_score(y, y_cluster_gmm))
print('The Confusion matrix of EM:\n ',metrics.confusion_matrix(y, y_cluster
_gmm))
```

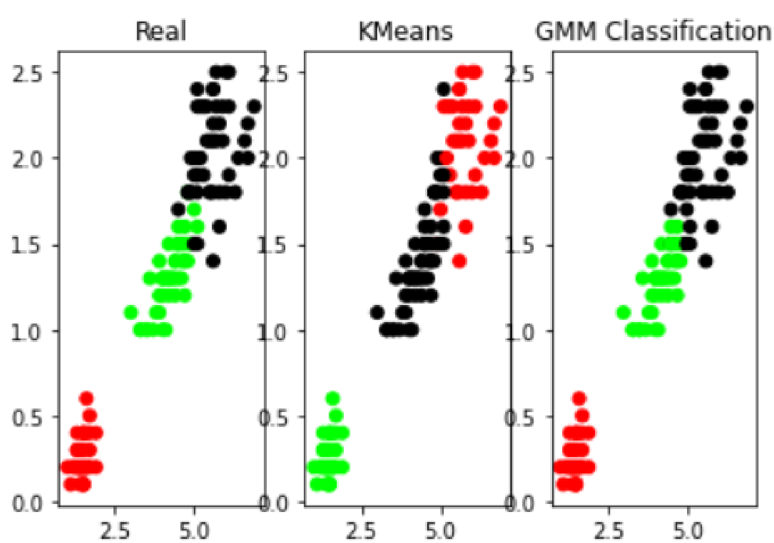
Dataset:

1	5.1	3.5	1.4	0.2	Iris-setosa
2	4.9	3	1.4	0.2	Iris-setosa
3	4.7	3.2	1.3	0.2	Iris-setosa
4	4.6	3.1	1.5	0.2	Iris-setosa
5	5	3.6	1.4	0.2	Iris-setosa
6	5.4	3.9	1.7	0.4	Iris-setosa
7	4.6	3.4	1.4	0.3	Iris-setosa
8	5	3.4	1.5	0.2	Iris-setosa
9	4.4	2.9	1.4	0.2	Iris-setosa
10	4.9	3.1	1.5	0.1	Iris-setosa

The entire dataset can be found in this link:
https://github.com/ArjunAS861/ML_1BM18CS019/blob/main/ML_Prog7/iris.csv

Output:

```
The accuracy score of K-Mean:  0.09333333333333334
The Confusion matrixof K-Mean:
 [[ 0 50  0]
 [ 2  0 48]
 [36  0 14]]
The accuracy score of EM:  0.9666666666666667
The Confusion matrix of EM:
  [[50  0  0]
 [ 0 45  5]
 [ 0  0 50]]
```



Program 8:
Write a program to implement k-Nearest Neighbour algorithm to classify the iris data set. Print both correct and wrong predictions.

Code:

```
import numpy as np
import pandas as pd
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn import metrics

names = ['sepal-length', 'sepal-width', 'petal-length', 'petal-width', 'Clas
s']

# Read dataset to pandas dataframe

dataset = pd.read_csv("iris.csv", names=names)
X = dataset.iloc[:, :-1]
y = dataset.iloc[:, -1]
print(X.head())
Xtrain, Xtest, ytrain, ytest = train_test_split(X, y, test_size=0.10)

classifier = KNeighborsClassifier(n_neighbors=5).fit(Xtrain, ytrain)

ypred = classifier.predict(Xtest)

i = 0
print ("\n---------------------------------------------------------------")
print ('%-25s %-25s %-25s' % ('Original Label', 'Predicted Label', 'Correct/
Wrong'))
print ("-----------------------------------------------------------------")
for label in ytest:
    print ('%-25s %-25s' % (label, ypred[i]), end="")
    if (label == ypred[i]):
        print (' %-25s' % ('Correct'))
    else:
        print (' %-25s' % ('Wrong'))
    i = i + 1
print ("-----------------------------------------------------------------")
print("\nConfusion Matrix:\n",metrics.confusion_matrix(ytest, ypred))
print ("-----------------------------------------------------------------")
print("\nClassification Report:\n",metrics.classification_report(ytest, ypre
d))
print ("-----------------------------------------------------------------")
print('Accuracy of the classifer is %0.2f' % metrics.accuracy_score(ytest,yp
red))
print ("-----------------------------------------------------------------")
```

Dataset:

1	5.1	3.5	1.4	0.2	Iris-setosa
2	4.9	3	1.4	0.2	Iris-setosa
3	4.7	3.2	1.3	0.2	Iris-setosa
4	4.6	3.1	1.5	0.2	Iris-setosa
5	5	3.6	1.4	0.2	Iris-setosa
6	5.4	3.9	1.7	0.4	Iris-setosa
7	4.6	3.4	1.4	0.3	Iris-setosa
8	5	3.4	1.5	0.2	Iris-setosa
9	4.4	2.9	1.4	0.2	Iris-setosa
10	4.9	3.1	1.5	0.1	Iris-setosa

The entire dataset can be found in this link:
https://github.com/ArjunAS861/ML_1BM18CS019/blob/main/ML_Prog7/iris.csv

Output:

```
   sepal-length  sepal-width  petal-length  petal-width
0           5.1          3.5           1.4          0.2
1           4.9          3.0           1.4          0.2
2           4.7          3.2           1.3          0.2
3           4.6          3.1           1.5          0.2
4           5.0          3.6           1.4          0.2

-----------------------------------------------------------------
Original Label          Predicted Label          Correct/Wrong
-----------------------------------------------------------------
Iris-virginica          Iris-virginica           Correct
Iris-versicolor         Iris-versicolor          Correct
Iris-virginica          Iris-virginica           Correct
Iris-setosa             Iris-setosa              Correct
Iris-versicolor         Iris-versicolor          Correct
Iris-versicolor         Iris-versicolor          Correct
Iris-versicolor         Iris-versicolor          Correct
Iris-versicolor         Iris-versicolor          Correct
Iris-setosa             Iris-setosa              Correct
Iris-virginica          Iris-virginica           Correct
Iris-virginica          Iris-virginica           Correct
Iris-versicolor         Iris-versicolor          Correct
Iris-setosa             Iris-setosa              Correct
Iris-virginica          Iris-virginica           Correct
Iris-virginica          Iris-virginica           Correct
-----------------------------------------------------------------
```

Fig 8.1 Output accuracy

```
Confusion Matrix:
 [[3 0 0]
 [0 6 0]
 [0 0 6]]
-------------------------------------------------------------------------

Classification Report:
                  precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00         3
Iris-versicolor       1.00      1.00      1.00         6
 Iris-virginica       1.00      1.00      1.00         6

       accuracy                           1.00        15
      macro avg       1.00      1.00      1.00        15
   weighted avg       1.00      1.00      1.00        15

-------------------------------------------------------------------------
Accuracy of the classifer is 1.00
-------------------------------------------------------------------------
```

Fig 8.2 Output accuracy

Program 9:
Implement the Linear Regression algorithm in order to fit data points. Select appropriate data set for your experiment and draw graphs.

<u>Code:</u>

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression


dataset = pd.read_csv('salary_data.csv')
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, 1].values

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=1/3, ran
dom_state=0)


# Fitting Simple Linear Regression to the Training set

regressor = LinearRegression()
regressor.fit(X_train, y_train)


# Predicting the Test set results
y_pred = regressor.predict(X_test)


# Visualizing the Training set results
viz_train = plt
viz_train.scatter(X_train, y_train, color='red')
viz_train.plot(X_train, regressor.predict(X_train), color='blue')
viz_train.title('Salary VS Experience (Training set)')
viz_train.xlabel('Year of Experience')
viz_train.ylabel('Salary')
viz_train.show()

# Visualizing the Test set results
viz_test = plt
viz_test.scatter(X_test, y_test, color='red')
viz_test.plot(X_train, regressor.predict(X_train), color='blue')
viz_test.title('Salary VS Experience (Test set)')
viz_test.xlabel('Year of Experience')
viz_test.ylabel('Salary')
viz_test.show()
```

<u>Dataset:</u>

YearsExperience	Salary
1.1	39343
1.3	46205
1.5	37731
2.0	43525
2.2	39891
2.9	56642
3.0	60150
3.2	54445
3.2	64445

The entire dataset can be found in this link:
https://github.com/ArjunAS861/ML_1BM18CS019/blob/main/ML_Prog9/salary_data.csv

<u>Output:</u>

Program 10:
Implement the non-parametric Locally Weighted Regression algorithm in order to fit data points. Select appropriate data set for your experiment and draw graphs.

Code:

```
import numpy as np
from bokeh.plotting import figure, show, output_notebook
from bokeh.layouts import gridplot
from bokeh.io import push_notebook

def local_regression(x0, X, Y, tau):# add bias term
 x0 = np.r_[1, x0] # Add one to avoid the loss in information
 X = np.c_[np.ones(len(X)), X]

 # fit model: normal equations with kernel
 xw = X.T * radial_kernel(x0, X, tau) # XTranspose * W

 beta = np.linalg.pinv(xw @ X) @ xw @ Y #@ Matrix Multiplication or Dot Prod
uct


 # predict value
 return x0 @ beta # @ Matrix Multiplication or Dot Product for prediction
def radial_kernel(x0, X, tau):
 return np.exp(np.sum((X - x0) ** 2, axis=1) / (-2 * tau * tau))
# Weight or Radial Kernal Bias Function

n = 1000
# generate dataset
X = np.linspace(-3, 3, num=n)
print("The Data Set ( 10 Samples) X :\n",X[1:10])
Y = np.log(np.abs(X ** 2 - 1) + .5)
print("The Fitting Curve Data Set (10 Samples) Y :\n",Y[1:10])
# jitter X
X += np.random.normal(scale=.1, size=n)
print("Normalised (10 Samples) X :\n",X[1:10])

domain = np.linspace(-3, 3, num=300)
print(" Xo Domain Space(10 Samples) :\n",domain[1:10])

def plot_lwr(tau):
 # prediction through regression
 prediction = [local_regression(x0, X, Y, tau) for x0 in domain]
 plot = figure(plot_width=400, plot_height=400)
 plot.title.text='tau=%g' % tau
 plot.scatter(X, Y, alpha=.3)
 plot.line(domain, prediction, line_width=2, color='red')
 return plot

show(gridplot([
 [plot_lwr(10.), plot_lwr(1.)],
```

```
[plot_lwr(0.1), plot_lwr(0.01)]]))
```

Dataset:

```
The Data Set ( 10 Samples) X :
          0
0 -2.993994
1 -2.987988
2 -2.981982
3 -2.975976
4 -2.969970
5 -2.963964
6 -2.957958
7 -2.951952
8 -2.945946
```

Fig 10.1 Dataset

Output:

```
The Data Set ( 10 Samples) X :
          0
0 -2.993994
1 -2.987988
2 -2.981982
3 -2.975976
4 -2.969970
5 -2.963964
6 -2.957958
7 -2.951952
8 -2.945946
The Fitting Curve Data Set (10 Samples) Y :
 [2.13582188 2.13156806 2.12730467 2.12303166 2.11874898 2.11445659
 2.11015444 2.10584249 2.10152068]
Normalised (10 Samples) X :
 [-2.76468351 -2.8892122  -2.90138442 -2.90382197 -3.03699931 -3.0209778
 -2.95438513 -3.0644698  -3.10680507]
 Xo Domain Space(10 Samples) :
 [-2.97993311 -2.95986622 -2.93979933 -2.91973244 -2.89966555 -2.87959866
 -2.85953177 -2.83946488 -2.81939799]
```

Fig 10.2 Output

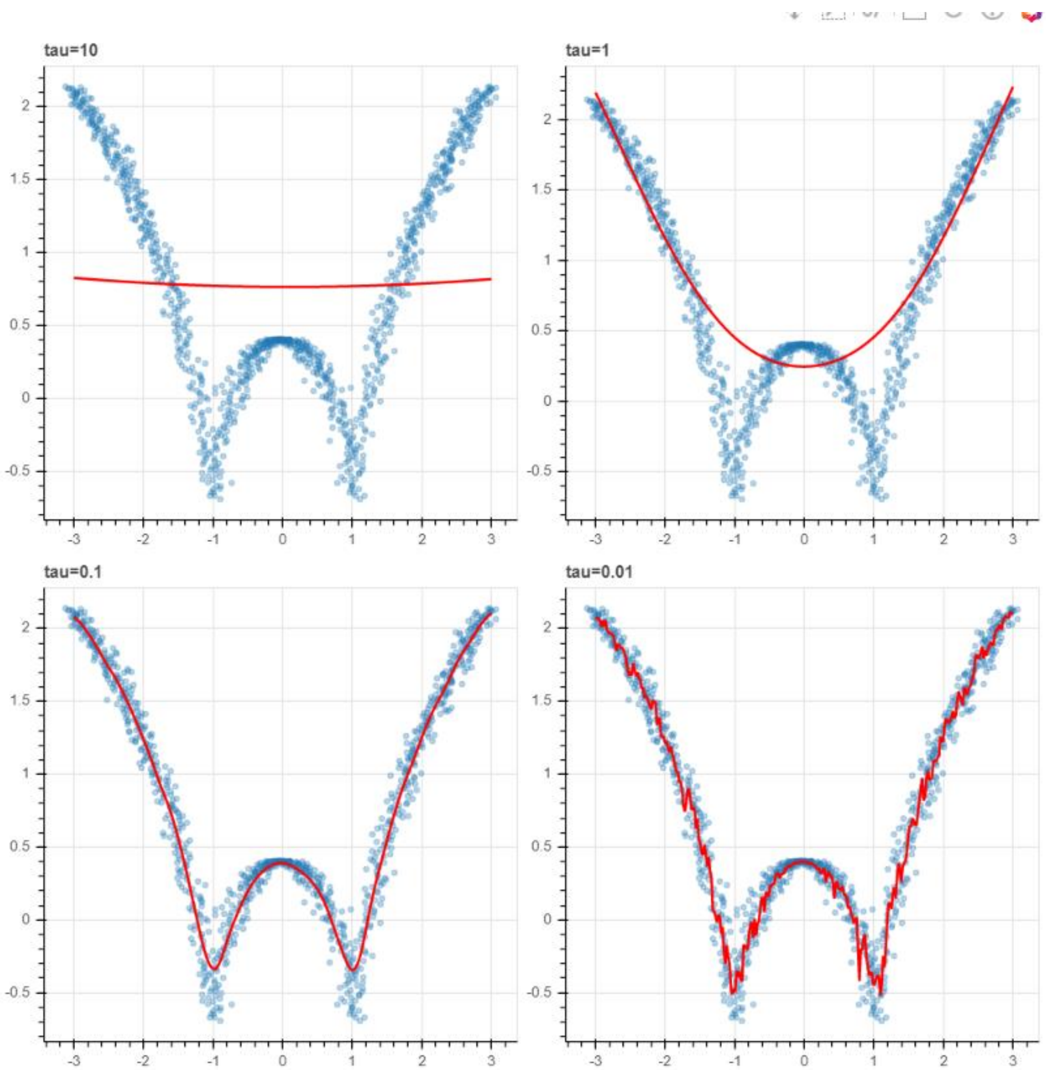


Fig 10.3 Output graph