VISVESVARAYATECHNOLOGICALU NIVERSITY

"JnanaSangama",Belgaum-590014,Karnataka.

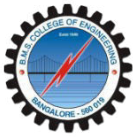
LABREPORT
on

MACHINELEARNING(20CS6PCMAL)

Submittedby

MOHDIRFAN(1BM20CS409)

inpartialfulfillmentfortheawardofthedegreeof
BACHELOROFENGINEERING
in
COMPUTERSCIENCEANDENGINEERING



B.M.S.COLLEGEOFENGINEERING
(AutonomousInstitutionunderVTU)
BENGALURU-560019

May-2022toJuly-2022

B.M.S.CollegeofEngineering,
BullTempleRoad,Bangalore560019
(AffiliatedToVisvesvarayaTechnologicalUniversity,Belgaum)
DepartmentofComputerScienceandEngineering

CERTIFICATE

This is to certifythat the Lab work entitled "**MACHINELEARNING**" carried out by**MOHDIRFAN(1BM20CS409)),**whoisbonafidestudentof**B.M.S.CollegeofEngineering.** Itisinpartialfulfillmentfortheawardof**BachelorofEngineeringinComputerScienceand Engineering** VisvesvarayaTechnological University, Belgaum during the year 2022. The Lab report has been approved as it satisfies the academic requirements in respect of a**MachineLearning- (20CS6PCMAL)**work prescribed for the said degree.

SarithaAN
AssistantProfessor
DepartmentofCSE
BMSCE,Bengaluru

Dr.JyothiSNayak
ProfessorandHead
DepartmentofCSE
BMSCE,Bengaluru

1.ImplementFINDSALGORITHM

In [5]:

```
import csv

a=[]
```

In [6]:

```
with open('enjoysports.csv','r') as csvfile:
    for row in csv.reader(csvfile):
        a.append(row)
    print(a)

print("\nThe Total Number of Training Instances are: ",len(a))
```

```
[['sunny', 'warm', 'normal', 'strong', 'warm', 'same', 'yes'], ['sunny', 'wa
rm', 'high', 'strong', 'warm', 'same', 'yes'], ['rainy', 'cold', 'high', 'st
rong', 'warm', 'change', 'no'], ['sunny', 'warm', 'high', 'strong', 'cool',
'change', 'yes']]

The Total Number of Training Instances are:  4
```

In [7]:

```
num_attribute = len(a[0])-1

print("\nThe 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]:
```

```
            if hypothesis[j] == '0' or hypothesis[j] == a[i][j]:
                hypothesis[j] = a[i][j]
            else:
                hypothesis[j] = '?'

    print("\nThe Hypothesis for the training instance {} is: \n".format(i+1),hypothesis)
```

```
The Initial Hypothesis is:
['0', '0', '0', '0', '0', '0']

The Hypothesis for the training instance 1 is:
 ['sunny', 'warm', 'normal', 'strong', 'warm', 'same']

The Hypothesis for the training instance 2 is:
 ['sunny', 'warm', '?', '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', '?', '?']
```

In [8]:

```
print("\nThe Maximally Specific Hypothesis for the training instance is ")
print(hypothesis)
```

```
The Maximally Specific Hypothesis for the training instance is
['sunny', 'warm', '?', 'strong', '?', '?']
```

2..ImplementCandidateeliminationAlgorithm

```
import numpy as np
import pandas as pd

data = pd.DataFrame(data = pd.read_csv('enjoysports.csv'))
concepts = np.array(data.iloc[:,0:-1])
print(concepts)

target = np.array(data.iloc[:,-1])
print(target)
```

```
[['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']
```

```
def learn(concepts, target):
    specific_h = concepts[0].copy()
    print("Initialization Of Specific Hypothesis & General Hypothesis : ")
    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
```

```
print("Final Specific Hypothesis: ",s_final, sep="\n")
print("Final General Hypothesis: ",g_final, sep="\n")
```

```
Final Specific Hypothesis:
['sunny' 'warm' '?' 'strong' '?' '?']
Final General Hypothesis:
[['sunny', '?', '?', '?', '?', '?'], ['?', 'warm', '?', '?', '?', '?']]
```

3.IMPLEMENTID-3ALGORITHM

In [ ]:
```
import numpy as np
```

In [ ]:
```
import pandas as pd
from sklearn.tree import DecisionTreeClassifier # Import Decision Tree Classifier
from sklearn.model_selection import train_test_split # Import train_test_split function
from sklearn import metrics #Import scikit-learn metrics module for accuracy calculation
```

In [4]:
```
col_names = ['pregnant', 'glucose', 'bp', 'skin', 'insulin', 'bmi','pedigree','age','label']
pima = pd.read_csv("/content/drive/MyDrive/diabetes.csv", header=None, names=col_names)
```

In [5]:
```
pima.head()
```

Out[5]:

	pregnant	glucose	bp	skin	insulin	bmi	pedigree	age	label
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

In [6]:
```
feature_cols = ['pregnant', 'insulin', 'bmi', 'age','glucose','bp','pedigree']
X = pima[feature_cols] # Features
y = pima.label # Target variable
```

In [7]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=2)
```

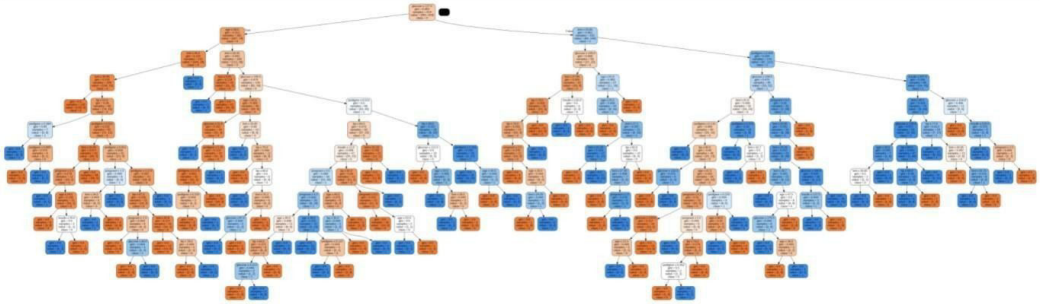
In [8]:
```
clf = DecisionTreeClassifier()
clf = clf.fit(X_train,y_train)
y_pred = clf.predict(X_test)
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

```
clf = clf.fit(X_train,y_train)
y_pred = clf.predict(X_test)
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.7467532467532467
```

In [9]:
```
from sklearn.tree import export_graphviz
from six import StringIO
from IPython.display import Image
import pydotplus

dot_data = StringIO()
export_graphviz(clf, out_file=dot_data,
                filled=True, rounded=True,
                special_characters=True,feature_names = feature_cols,class_names=['0','1'])
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
graph.write_png('diabetes.png')
Image(graph.create_png())
```

Out[9]:



In [ ]:
```
```

4.IMPLEMENTNAIVEBAYES

In [23]:

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

In [24]:

```
df = pd.read_csv("pima_indian.csv")
feature_col_names = ['num_preg', 'glucose_conc', 'diastolic_bp', 'thickness', 'insulin', 'b
predicted_class_names = ['diabetes']
```

In [25]:

```
X = df[feature_col_names].values
y = df[predicted_class_names].values
```

In [26]:

```
print(df.head)
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)
```

```
<bound method NDFrame.head of      num_preg  glucose_conc  diastolic_bp  thi
ckness  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)
```

In [27]:

```
clf = GaussianNB().fit(xtrain,ytrain.ravel())
predicted = clf.predict(xtest)
predictTestData= clf.predict([[6,148,72,35,0,33.6,0.627,50]])
```

In [28]:

```
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)
```

```
 Confusion matrix
[[139  20]
 [ 44  51]]

 Accuracy of the classifier is 0.7480314960629921

 The value of Precision 0.7183098591549296

 The value of Recall 0.5368421052631579
 Predicted Value for individual Test Data: [1]
```

5.IMPLEMENTLINEARREGRESSION

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

In [28]:
```
dataset = pd.read_csv('Salary_Data.csv')
dataset.head()
```

Out[28]:

	YearsExperience	Salary
0	1.1	39343.0
1	1.3	46205.0
2	1.5	37731.0
3	2.0	43525.0
4	2.2	39891.0

In [19]:
```
X = dataset.iloc[:, :-1].values
print(X)
```

```
<class 'numpy.ndarray'>
```

In [6]:
```
y = dataset.iloc[:, -1].values
```

In [10]:
```
dataset.head()
```

Out[10]:

	YearsExperience	Salary
0	1.1	39343.0
1	1.3	46205.0
2	1.5	37731.0
3	2.0	43525.0
4	2.2	39891.0

In [11]:
```
from sklearn.model_selection import train_test_split
```

In [12]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 1/3, random_state = 0)
```

In [14]:
```
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train, y_train)
```

Out[14]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

In [15]:
```
y_pred = regressor.prplt.scatter(X_train, y_train, color = 'red')
plt.plot(X_train, regressor.predict(X_train), color = 'blue')
plt.title('Salary vs Experience (Training set)')
plt.xlabel('Years of Experience')
plt.ylabel('Salary')
plt.show()edict(X_test)
```

In [16]:
```
pd.DataFrame(data={'Actuals': y_test, 'Predictions': y_pred})
```

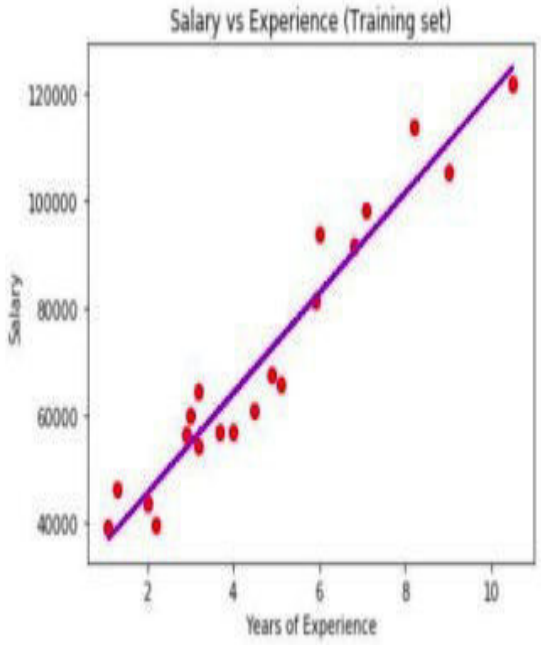
Out[16]:

	Actuals	Predictions
0	37731.0	40835.105909
1	122391.0	123079.399408
2	57081.0	65134.556261
3	63218.0	63265.367772
4	116969.0	115602.645454
5	109431.0	108125.891499
6	112635.0	116537.239698
7	55794.0	64199.962017
8	83088.0	76349.687193

7	55794.0	64199.962017
8	83088.0	76349.687193
9	101302.0	100649.137545

In [17]:

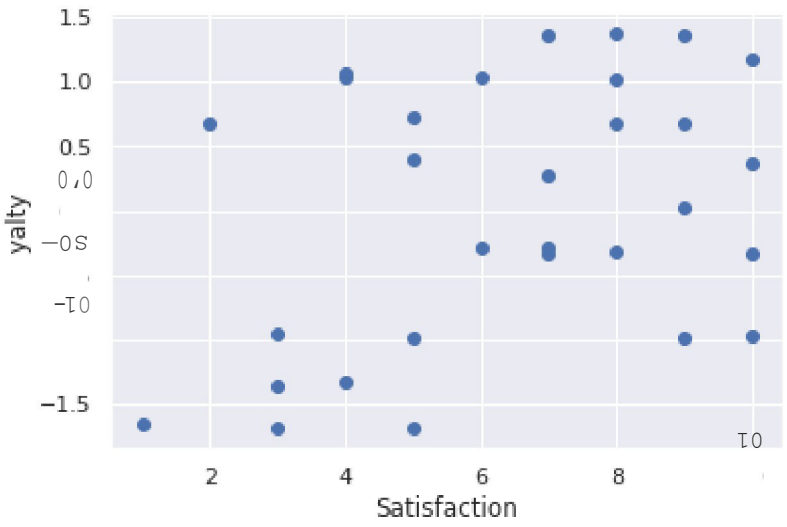
```
plt.scatter(X_train, y_train, color = 'red')
plt.plot(X_train, regressor.predict(X_train), color = 'blue')
plt.title('Salary vs Experience (Training set)')
plt.xlabel('Years of Experience')
plt.ylabel('Salary')
plt.show()
```

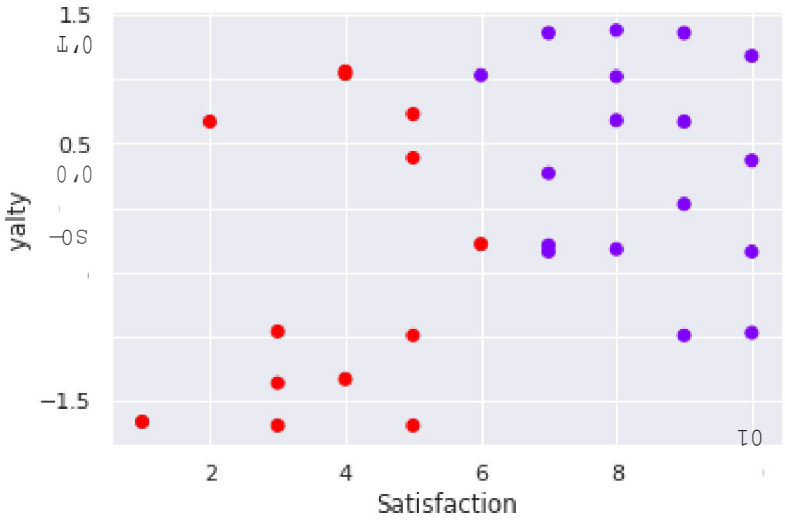


Program 6: K Means

```
import numpy as np import
pandas as pd
import matplotlib.pyplot as plt import
seaborn as sns
sns.set()
from sklearn.cluster import KMeans

data = pd.read_csv("../input/K-Means/Dataset.csv") data

plt.scatter(data['Satisfaction'],data['Loyalty']) plt.xlabel('Satisfaction')
plt.ylabel('Loyalty') plt.show()

x=data.copy() kmean=KMeans(2)
kmean.fit(x)

clusters=x.copy()
clusters['cluster_pred']=kmean.fit_predict(x)

plt.scatter(clusters['Satisfaction'],clusters['Loyalty'],c=clusters['cluster_pred'],cmap='rainbow') plt.xlabel('Satisfaction')
plt.ylabel('Loyalty')
plt.ylabel('Loyalty') plt.show()
```

Satisfaction	Loyalty
4	-1.33
6	-0.28
5	-0.99
7	-0.29
4	1.06
1	-1.66
10	-0.97
8	-0.32
8	1.02
8	0.68
10	-0.34
5	0.39
5	-1.69
2	0.67
7	0.27


1.5
1.0
0.5
yalty
2
4
6
8
Satisfaction


1.5
0.5
yalty
2
4
6
8
Satisfaction

Program 7: EM Algorithm

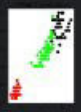
```
import matplotlib.pyplot as
plt from sklearn
importdatasets
from sklearn.cluster import
KMeans import sklearn.metrics
assm
import pandas as
pd import numpy
as np

iris = datasets.load_iris()

X = pd.DataFrame(iris.data)
X.columns = ['Sepal_Length','Sepal_Width','Petal_Length','Petal_Width']

y =
pd.DataFrame(iris.target)
y.columns = ['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 matrixof K-Mean:\n',sm.confusion_matrix(y,
model.labels_))
```

```
from sklearn import preprocessing
scaler =
preprocessing.StandardScaler()
scaler.fit(X)
xsa = scaler.transform(X)
xs = pd.DataFrame(xsa, columns = X.columns)
```

```
from sklearn.mixture import
GaussianMixture gmm =
GaussianMixture(n_components=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:\n',sm.confusion_matrix(y,
y_gmm))
```

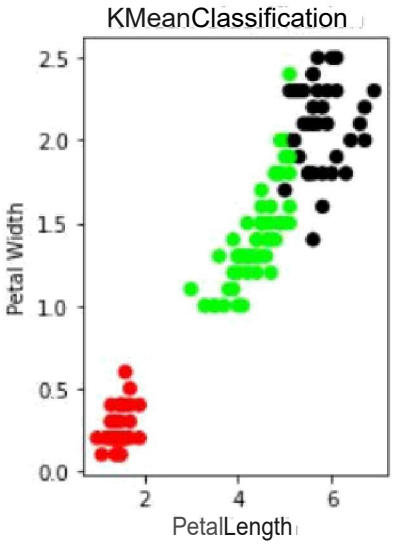
```
<Figure size 1008x504 with 0 Axes>
Text(0, 0.5, 'Petal Width')
```

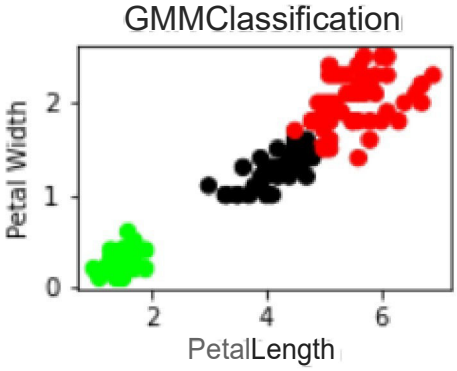
```
Text(0, 0.5, 'Petal Width')
```



```
The accuracy score of K-Mean:  0.8933333333333333
The Confusion matrixof K-Mean:
 [[50  0  0]
 [ 0 48  2]
 [ 0 14 36]]
Text(0, 0.5, 'Petal Width')
```

```
The accuracy score of EM:  0.0
The Confusion matrix of EM:
 [[ 0 50  0]
 [ 5  0 45]
 [50  0  0]]
```


KMeanClassification
Petal Width
PetalLength
0.0
0.5
1.0
1.5
2.0
2.5
2
4
6


GMMClassification
Petal Width
PetalLength
0
1
2
2
4
6

Program 8 : K Nearest Neighbor

```
from sklearn.model_selection import
train_test_split from sklearn.neighbors import
KNeighborsClassifier
from sklearn.metrics import classification_report,
confusion_matrix from sklearn import datasets

iris =
datasets.load_iris() X =
iris.data
Y = iris.target

print('sepal-length','sepal-width','petal-length','petal-
width')print(X)
print('target'
)print(Y)

x_train, x_test, y_train, y_test = train_test_split(X,Y,test_size=0.3)

classier =
KNeighborsClassifier(n_neighbors=5)
classier.fit(x_train, y_train)

y_pred=classier.predict(x_test)

print('Confusion Matrix')
print(confusion_matrix(y_test,y_pred)
)


print('Accuracy')
print(classification_report(y_test,y_pred)
)
```

```
target
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2]
Confusion Matrix
[[19  0  0]
 [ 0 16  1]
 [ 0  0  9]]
Accuracy
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        19
           1       1.00      0.94      0.97        17
           2       0.90      1.00      0.95         9

    accuracy                           0.98        45
   macro avg       0.97      0.98      0.97        45
weighted avg       0.98      0.98      0.98        45
```

Program 9: Bayesian Network

```
import numpy as
np import pandas
as pd import csv
import pgmpy
from pgmpy.estimators import
MaximumLikelihoodEstimator from pgmpy.models import
BayesianNetwork
from pgmpy.inference import VariableElimination

#read Cleveland Heart Disease data
heartDisease =
pd.read_csv('../input/heartdisease/heartDiseaseDataset.csv') heartDisease
= heartDisease.replace('?',np.nan)

#display the data
print('Sample instances from the dataset are given
below') print(heartDisease.head())

#display the Attributes names and
datatypes print('\n Attributes and
datatypes') print(heartDisease.dtypes)

#Create Model-Bayesian
Network model =
BayesianNetwork([('age','heartDisease'),('sex','heartDisease'),('exang','heartDisease'),('cp','heartDisease'),
(' restecg','heartDisease'),('heartDisease','chol')])

#Learning CPDs using Maximum Likelihood Estimators
print('\n Learning CPD using Maximum likelihood
estimators')
model.fit(heartDisease,estimator=MaximumLikelihoodEstimator)

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

#computing the Probability of heartDisease given restecg
print('\n 1.Probability of heartDisease given evidence= restecg :1')
q1=heartDiseasetest_infer.query(variables=['heartDisease'],evidence={'restecg':1})
print(q1)

#computing the Probability of heartDisease given cp
print('\n 2.Probability of heartDisease given evidence= cp:2 ')
q2=heartDiseasetest_infer.query(variables=['heartDisease'],evidence={'cp':2}
) print(q2)
```

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
53	1	4	140	203	1	2	155	1	3.1	3	0	7	1
57	1	4	140	192	0	0	148	0	0.4	2	0	6	0
56	0	2	140	294	0	2	153	0	1.3	2	0	3	0
56	1	3	130	256	1	2	142	1	0.6	2	1	6	2
44	1	2	120	263	0	0	173	0	0	1	0	7	0
52	1	3	172	199	1	0	162	0	0.5	1	0	7	0
57	1	3	150	168	0	0	174	0	1.6	1	0	3	0
48	1	2	110	229	0	0	168	0	1	3	0	7	1
54	1	4	140	239	0	0	160	0	1.2	1	0	3	0
48	0	3	130	275	0	0	139	0	0.2	1	0	3	0
49	1	2	130	266	0	0	171	0	0.6	1	0	3	0
64	1	1	110	211	0	2	144	1	1.8	2	0	3	0
58	0	1	150	283	1	2	162	0	1	1	0	3	0

```
Sample instances from the dataset are given below
   age  sex  cp  trestbps  chol  fbs  restecg  thalach  exang  oldpeak  slope  \
0   63    1   1       145   233    1        2      150      0      2.3      3
1   67    1   4       160   286    0        2      108      1      1.5      2
2   67    1   4       120   229    0        2      129      1      2.6      2
3   37    1   3       130   250    0        0      187      0      3.5      3
4   41    0   2       130   204    0        2      172      0      1.4      1

   ca  thal  heartDisease
0   0     6             0
1   3     3             2
2   2     7             1
3   0     3             0
4   0     3             0

 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                int64
thal              int64
heartDisease      int64
dtype: object
```

LearningCPDusingMaximumlikelihoodestimatorsInferencingwithBayesianNetvArk:

6. Probabilityofheart Diseasegivenevidence=restecg

Finding

EliminationOrder::100%Eliminating:exang:10096

4/4[00!00<00:00,81.79it/s]

:1 4/4[00:DOEOD:0064.63it/s]

heartDiseaseh phi(heartDisease)

eartDisease(B) he

artDisease(1)heart 8.1978

D1sea se(2)	0.1976
heartDisease(3)	0.1972 0.1976
heartDisease(4)	

7. Proba611ityofheartD1s ea se given ev dence=cp:2

FindingEliminationOrder::100%Eliminating:e

4/4[OO:OOH00:00,69.B 5lt/s]

xang:10096	0.2106

4/4[00:DOEOD:00 89.18it/s]

heartD1sea ph1heantDisease)

seheartDisease(B

)

~~heartDisease(1)~~	
~~heartDisease(2)~~	~~0.1633~~ 0.3138
~~heartD1sease(3)~~	~~6.1527~~ 0.2150
~~heartDisease(4)~~	0.1552

Program 10: Locally Weighted Regression

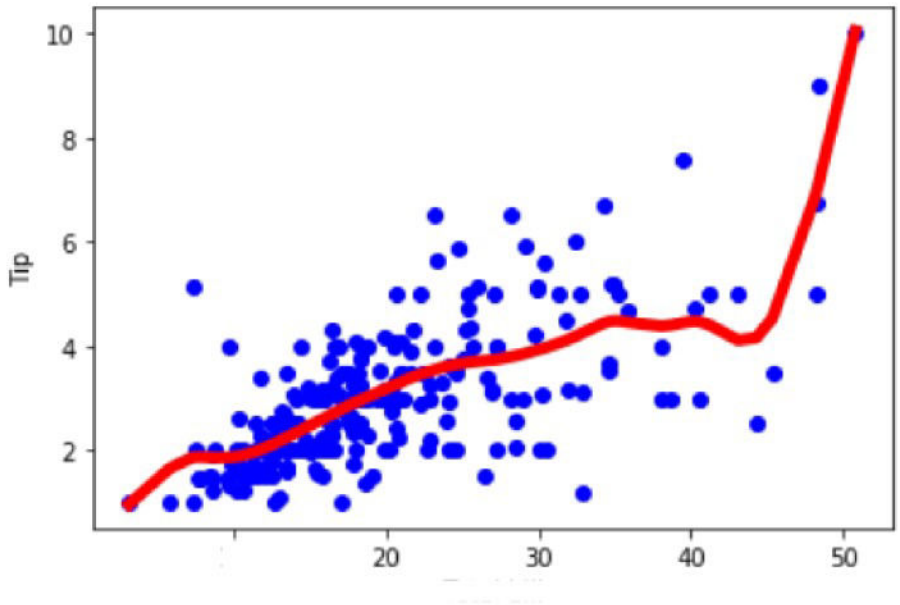
```
from numpy import *
from os import listdir
import matplotlib
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np1
import numpy.linalg as np
from scipy.stats.stats import pearsonr

def kernel(point,xmat, k):
    m,n = np1.shape(xmat)
    weights = np1.mat(np1.eye((m)))
    for j in range(m):
        diff = point - X[j]
        weights[j,j] = np1.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 = np1.shape(xmat)
    ypred = np1.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('../input/tipsdata/tips.csv')
bill = np1.array(data.total_bill)
tip = np1.array(data.tip)

#preparing and add 1 in bill
mbill = np1.mat(bill)
mtip = np1.mat(tip)
# mat is used to convert to n dimesiona to 2 dimensional array form m=
np1.shape(mbill)[1] # print(m) 244 data is stored in m
one = np1.mat(np1.ones(m))
X= np1.hstack((one.T,mbill.T)) # create a stack of bill from ONE
print(X)
```

```
#set k here
ypred = localWeightRegression(X,mtip,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='blue')
ax.plot(xsort[:,1],ypred[SortIndex], color = 'red', linewidth=5)
plt.xlabel('Total bill')
plt.ylabel('Tip')
plt.show();



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 Product 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])
```

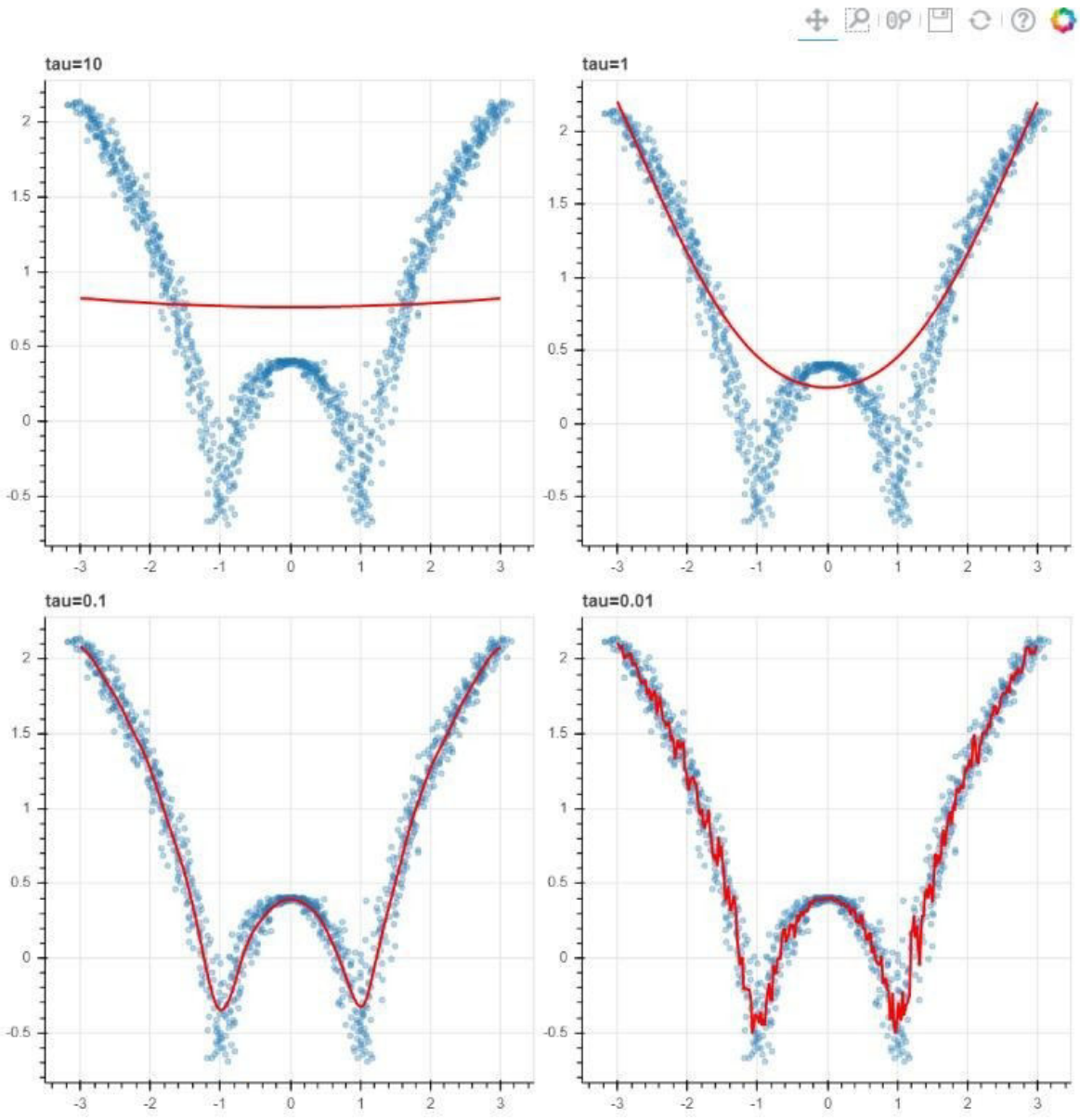
```
domain = np.linspace(-3, 3, num=300)
print(" Xo Domain Space(10 Samples) :\n",domain[1:10])

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)]]))

from numpy import *
from os import listdir
import matplotlib
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np1
import numpy.linalg as np
from scipy.stats.stats import pearsonr

def kernel(point,xmat, k):
    m,n = np1.shape(xmat)
    weights = np1.mat(np1.eye((m)))
    for j in range(m):
        diff = point - X[j]
        weights[j,j] = np1.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 = np1.shape(xmat)
    ypred = np1.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('../input/tipsdata/tips.csv')
bill = np1.array(data.total_bill)
tip = np1.array(data.tip)

#preparing and add 1 in bill
mbill = np1.mat(bill)
mtip = np1.mat(tip) # mat is used to convert to n dimesiona to 2 dimensional array form m=
np1.shape(mbill)[1]
# print(m) 244 data is stored in m
one = np1.mat(np1.ones(m))
X= np1.hstack((one.T,mbill.T)) # create a stack of bill from ONE
#print(X)
#set k here
ypred = localWeightRegression(X,mtip,0.3)
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();
```

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	2.0	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	5.0	Female	No	Sun	Dinner	4
15.42	1.57	Male	No	Sun	Dinner	2
18.43	3.0	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



Totalbill


tau=10
tau=1
tau=0.1
tau=0.01