**VISVESVARAYA TECHNOLOGICAL UNIVERSITY**

**“JnanaSangama”, Belgaum -590014, Karnataka.**

****

**LAB REPORT**

**on**

**MACHINE LEARNING**

***Submitted by***

**NEHA B CHADAGA (1BM19CS098)**

***in partial fulfillment for the award of the degree of***

**BACHELOR OF ENGINEERING**

***in***

**COMPUTER SCIENCE AND ENGINEERING**



**B.M.S. COLLEGE OF ENGINEERING**

**(Autonomous Institution under VTU)**

**BENGALURU-560019**

**May-2022 to July-2022**

**B. M. S. College of Engineering,**

**Bull Temple Road, Bangalore 560019**

(Affiliated To Visvesvaraya Technological University, Belgaum)

**Department of Computer Science and Engineering**



**CERTIFICATE**

This is to certify that the Lab work entitled “MACHINE LEARNING” carried out by **NEHA B CHADAGA (1BM19CS098),** who is bonafide student of **B. M. S. College of Engineering.** It is in partial fulfillment for the award of **Bachelor of Engineering in Computer Science and Engineering** of the Visvesvaraya Technological University, Belgaum during the year 2022. The Lab report has been approved as it satisfies the academic requirements in respect of a **Course Title - (Course code)** work prescribed for the said degree.

Name of the Lab-Incharge               **Dr. Jyothi S Nayak**

Designation Professor and Head

Department of CSE Department of CSE

BMSCE, Bengaluru BMSCE, Bengaluru

`

**Index Sheet**

|  |  |  |
| --- | --- | --- |
| **Sl. No.** | **Experiment Title** | **Page No.** |
| **1** | **FIND-S Algorithm** |  |
| **2** | **Candidate Elimination Algorithm** |  |
| **3** | **Decision tree based ID3 Algorithm** |  |
| **4** | **Linear Regression** |  |
| **5** | **Naive Bayesian Classifier** |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**Course Outcome**

|  |  |
| --- | --- |
|  |  |

## EXPERIMENT 1

1

**CODE:**

import csv

import pandas as pd

import numpy as np

data = pd.read\_csv("Desktop/data.csv")

print(data,"\n")

#array of all the attributes

d = np.array(data)[:,:-1]

print("\n The attributes are: ",d)

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

print("\n The target is: ",target)

def findS(c,t):

for i, val in enumerate(t):

if val == "Yes":

specific\_hypothesis = c[i].copy()

break

for i, val in enumerate(c):

if t[i] == "Yes":

for x in range(len(specific\_hypothesis)):

if val[x] != specific\_hypothesis[x]:

specific\_hypothesis[x] = '?'

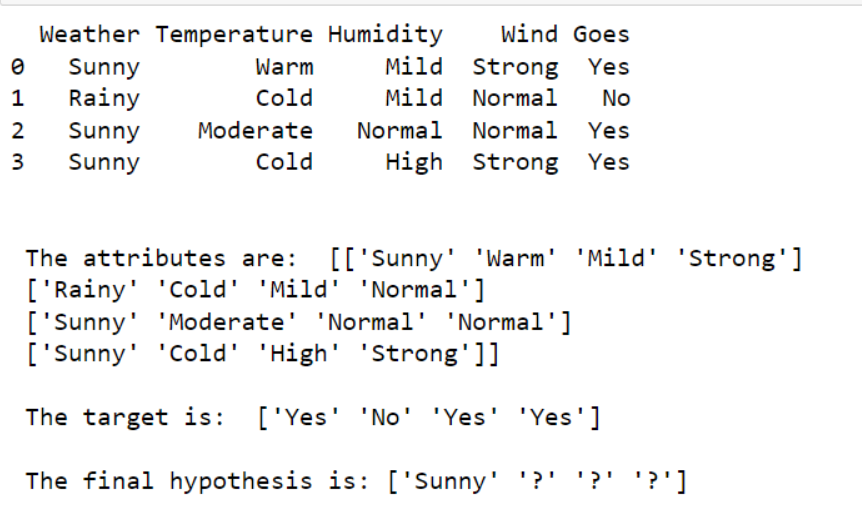
else:

pass

return specific\_hypothesis

print("\n The final hypothesis is:",findS(d,target))

**OUTPUT:**

****

## EXPERIMENT 2

2

**CODE:**

import numpy as np

import pandas as pd

data = pd.read\_csv('Desktop/shape.csv')

concepts = np.array(data.iloc[:,0:-1])

print("\nInstances are:\n",concepts)

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

print("\nTarget Values are: ",target)

def learn(concepts, target):

specific\_h = concepts[0].copy()

print("\nInitialization of specific\_h and genearal\_h")

print("\nSpecific Boundary: ", specific\_h)

general\_h = [["?" for i in range(len(specific\_h))] for i in range(len(specific\_h))]

print("\nGeneric Boundary: ",general\_h)

for i, h in enumerate(concepts):

print("\nInstance", i+1 , "is ", h)

if target[i] == "yes":

print("Instance is Positive ")

for x in range(len(specific\_h)):

if h[x]!= specific\_h[x]:

specific\_h[x] ='?'

general\_h[x][x] ='?'

if target[i] == "no":

print("Instance is Negative ")

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("Specific Bundary after ", i+1, "Instance is ", specific\_h)

print("Generic Boundary after ", i+1, "Instance is ", general\_h)

print("\n")

indices = [i for i, val in enumerate(general\_h) if val == ['?', '?', '?', '?', '?', '?']]

for i in indices:

general\_h.remove(['?', '?', '?', '?', '?', '?'])

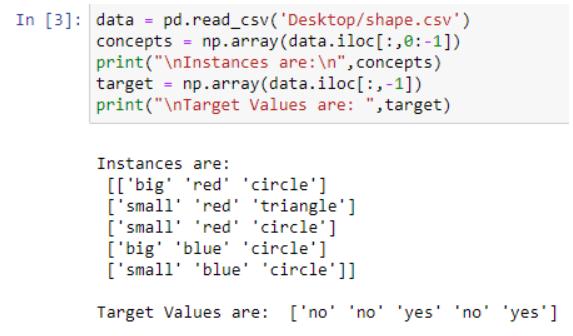
return specific\_h, general\_h

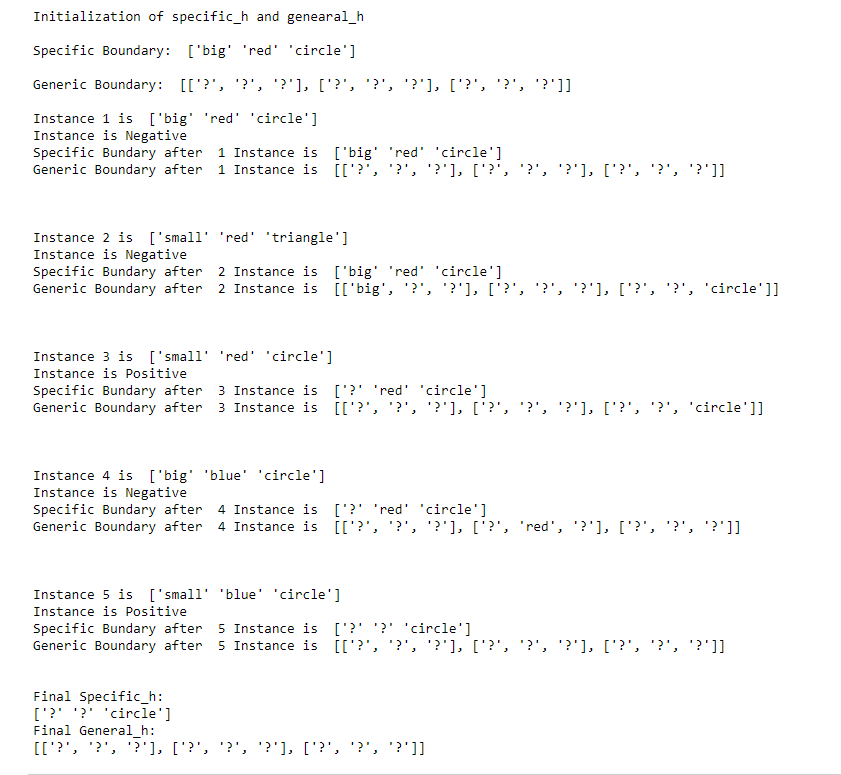
s\_final, g\_final = learn(concepts, target)

print("Final Specific\_h: ", s\_final, sep="\n")

print("Final General\_h: ", g\_final, sep="\n")

**OUTPUT:**

****

****

## EXPERIMENT 3

3

**CODE:**

WITHOUT ALGO:

import csv

def load\_csv(filename):

lines=csv.reader(open(filename,"r"));

dataset = list(lines)

headers = dataset.pop(0)

return dataset,headers

class Node:

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

self.attribute=attribute

self.children=[]

self.answer=""

def subtables(data,col,delete):

dic={}

coldata=[row[col] for row in data]

attr=list(set(coldata))

counts=[0]\*len(attr)

r=len(data)

c=len(data[0])

for x in range(len(attr)):

for y in range(r):

if data[y][col]==attr[x]:

counts[x]+=1

for x in range(len(attr)):

dic[attr[x]]=[[0 for i in range(c)] for j in range(counts[x])]

pos=0

for y in range(r):

if data[y][col]==attr[x]:

if delete:

del data[y][col]

dic[attr[x]][pos]=data[y]

pos+=1

return attr,dic

def entropy(S):

attr=list(set(S))

if len(attr)==1:

return 0

counts=[0,0]

for i in range(2):

counts[i]=sum([1 for x in S if attr[i]==x])/(len(S)\*1.0)

sums=0

for cnt in counts:

sums+=-1\*cnt\*math.log(cnt,2)

return sums

def compute\_gain(data,col):

attr,dic = subtables(data,col,delete=False)

total\_size=len(data)

entropies=[0]\*len(attr)

ratio=[0]\*len(attr)

total\_entropy=entropy([row[-1] for row in data])

for x in range(len(attr)):

ratio[x]=len(dic[attr[x]])/(total\_size\*1.0)

entropies[x]=entropy([row[-1] for row in dic[attr[x]]])

total\_entropy-=ratio[x]\*entropies[x]

return total\_entropy

def build\_tree(data,features):

lastcol=[row[-1] for row in data]

if(len(set(lastcol)))==1:

node=Node("")

node.answer=lastcol[0]

return node

n=len(data[0])-1

gains=[0]\*n

for col in range(n):

gains[col]=compute\_gain(data,col)

split=gains.index(max(gains))

node=Node(features[split])

fea = features[:split]+features[split+1:]

attr,dic=subtables(data,split,delete=True)

for x in range(len(attr)):

child=build\_tree(dic[attr[x]],fea)

node.children.append((attr[x],child))

return node

def print\_tree(node,level):

if node.answer!="":

print(" "\*level,node.answer)

return

print(" "\*level,node.attribute)

for value,n in node.children:

print(" "\*(level+1),value)

print\_tree(n,level+2)

def classify(node,x\_test,features):

if node.answer!="":

print(node.answer)

return

pos=features.index(node.attribute)

for value, n in node.children:

if x\_test[pos]==value:

classify(n,x\_test,features)

'''Main program'''

dataset,features=load\_csv("data.csv")

node1=build\_tree(dataset,features)

print("The decision tree for the dataset using ID3 algorithm is")

print\_tree(node1,0)

testdata,features=load\_csv("test.csv")

for xtest in testdata:

print("The test instance:",xtest)

print("The label for test instance:",end=" ")

classify(node1,xtest,features)

WITH ALGO:

import numpy as np

import pandas as pd

import math

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

print(data)

def countPosNeg(data):

pos = data.iloc[:,-1:].value\_counts()['yes']

neg = len(data) - pos

return pos, neg

def calcEntropy(pos, neg):

entropy = -(pos/(pos+neg))\*math.log2(pos/(pos+neg)) -(neg/(pos+neg))\*math.log2(neg/(pos+neg))

return entropy

def calcAverageInformation(data):

# iterate through each attribute (col)

attribs = data.iloc[:0,:-1].columns.values

print(attribs)

for attrib in attribs:

# get possible values

values = data[attrib].unique()

valueEntropies = pd.DataFrame(0, columns=['p','n','entropy'], index=values)

print()

print(attrib)

print(valueEntropies)

# iterate through whole dataframe

for i in data.index:

print(data['Answer'][i])

if data['Answer'][i] == 'yes':

valueEntropies[data[attrib]]['p'] += 1

elif data['Answer'][i] == 'no':

valueEntropies[data[attrib]]['n'] += 1

for value in valueEntropies:

value['entropy'] = calcEntropy(value['p'], value['n'])

print(valueEntropies)

return 10

calcAverageInformation(data)

def calcGain(entropy, avg\_info):

return entropy - avg\_info

# data for the total dataset

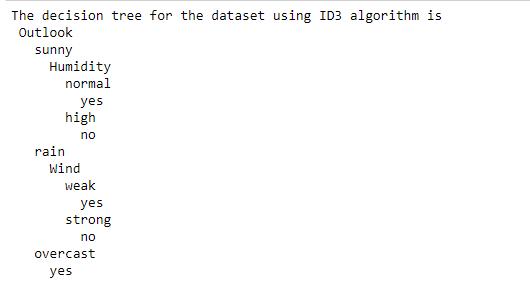
tot\_pos, tot\_neg = countPosNeg(data)

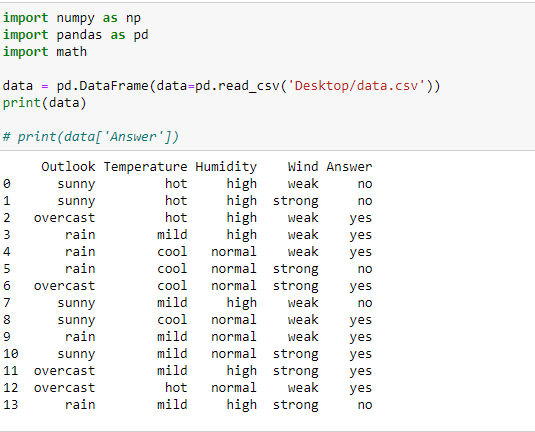
tot\_entropy = calcEntropy(tot\_pos, tot\_neg)

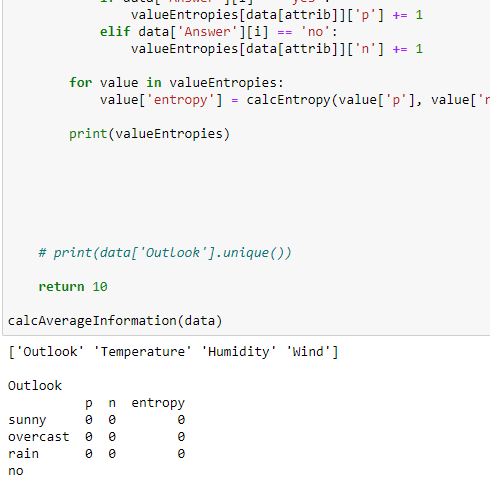
print(tot\_entropy)

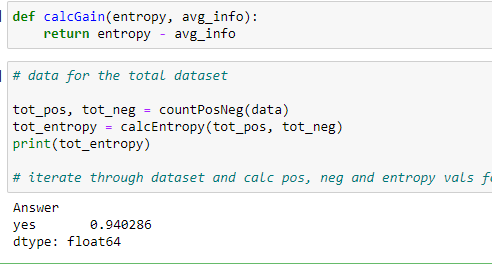
# iterate through dataset and calc pos, neg and entropy vals for each column

**OUTPUT:**

****

****

****

****

## EXPERIMENT 4

4

**CODE:**

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

dataset = pd.read\_csv('salary\_data.csv')

X = dataset.iloc[:, :-1].values

y = dataset.iloc[:, 1].values

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

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=1/3, random\_state=0)

# Fitting Simple Linear Regression to the Training set

from sklearn.linear\_model import LinearRegression

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')

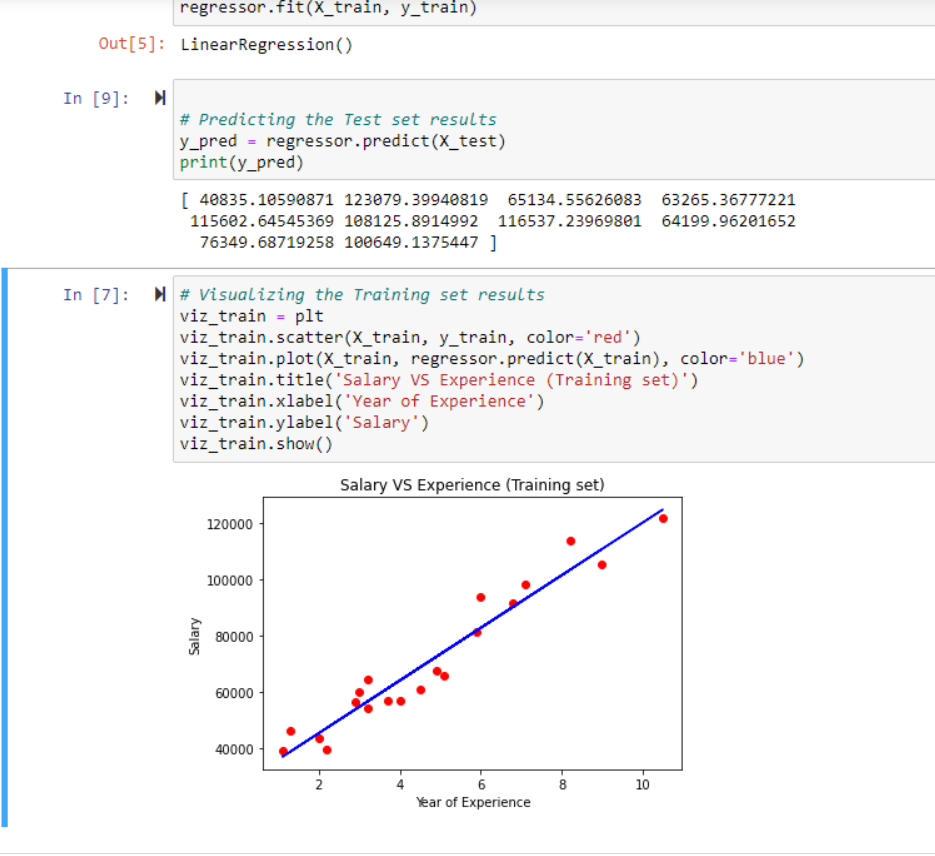
viz\_test.title('Salary VS Experience (Test set)')

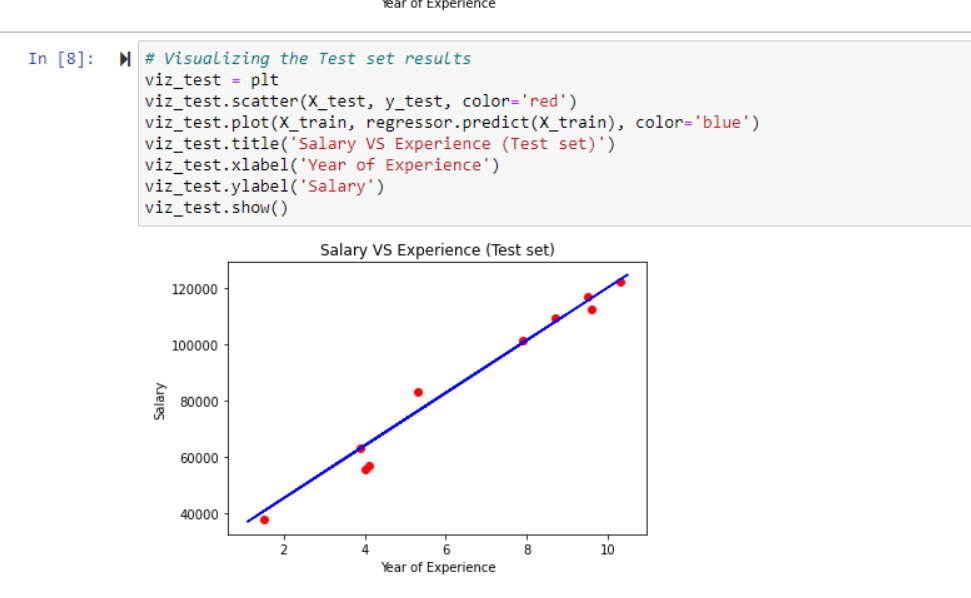
viz\_test.xlabel('Year of Experience')

viz\_test.ylabel('Salary')

viz\_test.show()

**OUTPUT:**

****

****

## EXPERIMENT 5

5

**CODE:**

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("Downloads/data.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

y = df[predicted\_class\_names].values

print(df.head)

xtrain,xtest,ytrain,ytest=train\_test\_split(X,y,test\_size=0.40)

print ('\n the total number of Training Data :',ytrain.shape)

print ('\n the total number of Test Data :',ytest.shape)

clf = GaussianNB().fit(xtrain,ytrain.ravel())

predicted = clf.predict(xtest)

predictTestData= clf.predict([[6,148,72,35,0,33.6,0.627,50]])

print('\n Confusion matrix')

print(metrics.confusion\_matrix(ytest,predicted))

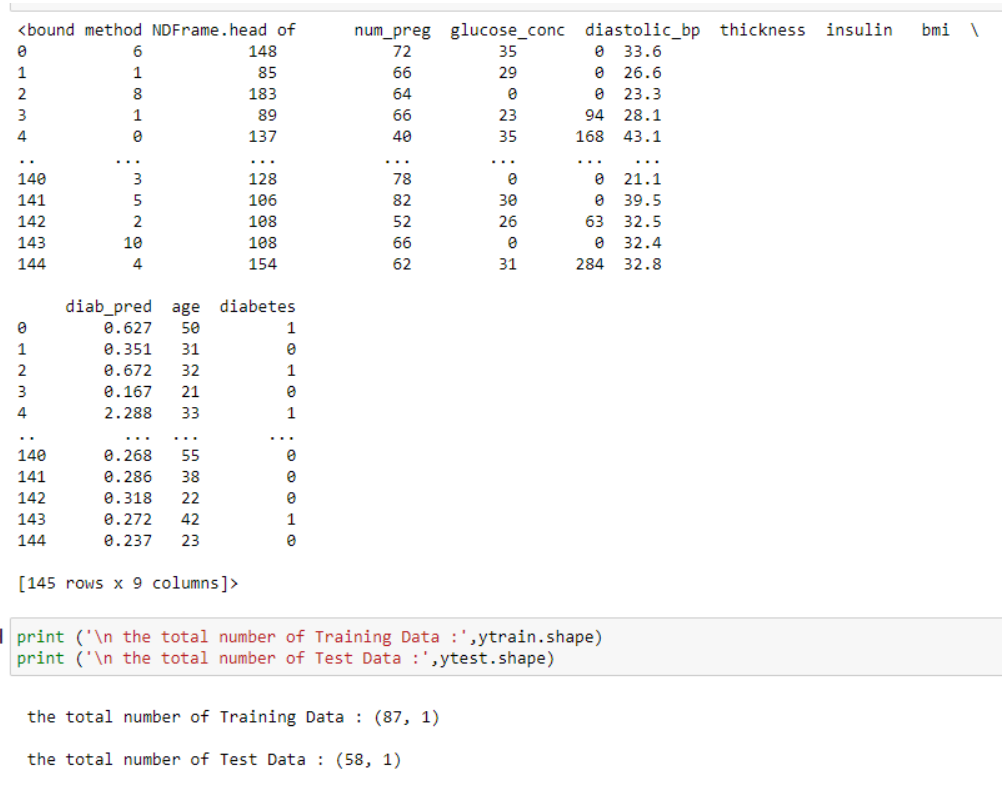
print('\n Accuracy of the classifier is',metrics.accuracy\_score(ytest,predicted))

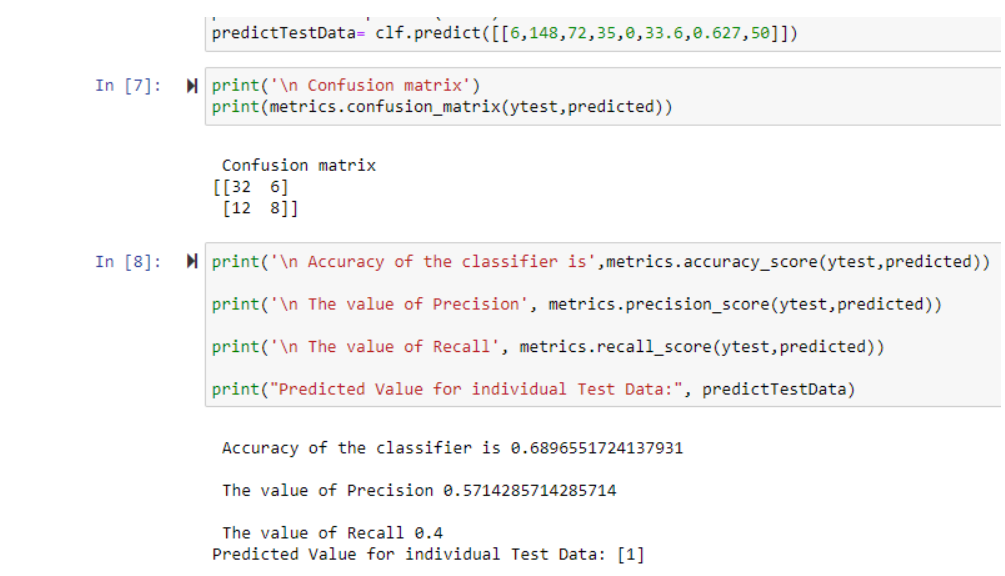
print('\n The value of Precision', metrics.precision\_score(ytest,predicted))

print('\n The value of Recall', metrics.recall\_score(ytest,predicted))

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

**OUTPUT:**

****

****