# VISVESVARAYA TECHNOLOGICAL UNIVERSITY

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

**LAB REPORT**

**on**

# MACHINE LEARNING (20CS6PCMAL)

***Submitted by***

**MALLIKA PRASAD(1BM19CS081)**

***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 “LAB COURSE **MACHINE LEARNING**” carried out by **MALLIKA PRASAD (1BM19CS081),** 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 Machine Learning **- (20CS6PCMAL)** work prescribed for the said degree.

**Dr G.R.Asha**

Assistant Professor
Department of CSE
BMSCE, Bengaluru

**Dr. Jyothi S Nayak**
Professor and Head
Department of CSE
BMSCE, Bengaluru

`

## Index Sheet

| Sl. No. | Experiment Title | Page No. |
|---|---|---|
| 1 | FIND S | 4 |
| 2 | CANDIDATE ELIMINATION | 5 |
| 3 | DECISION TREE | 7 |
| 4 | NAIVE BAYES | 10 |
| 5 | LINEAR REGRESSION | 11 |
| | | |
| | | |

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

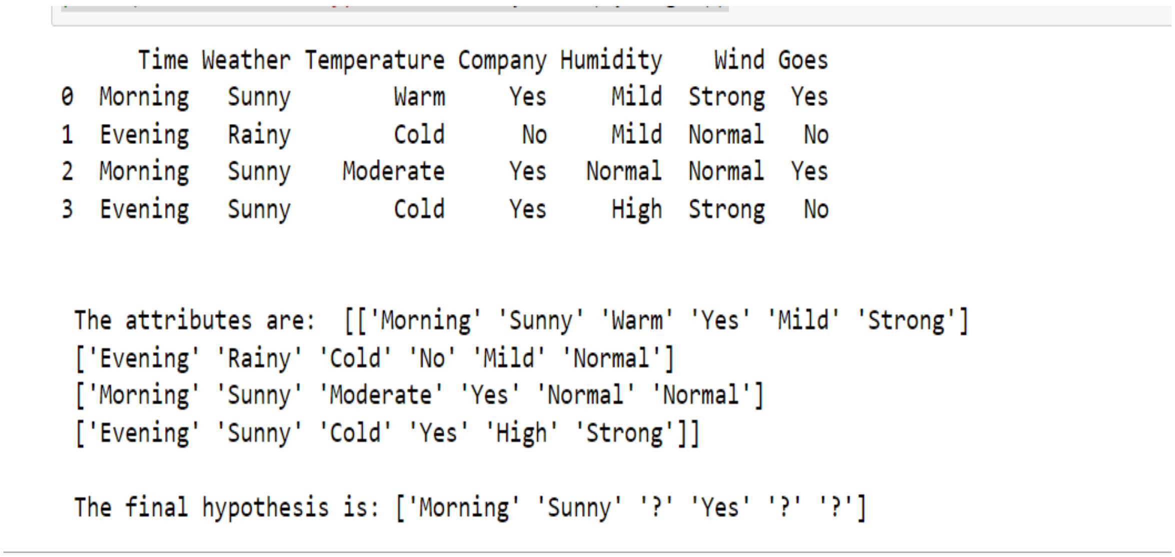
## Course Outcome

| | |
|---|---|
| CO1 | Ability to apply the different learning algorithms. |
| CO2 | Ability to analyze the learning techniques for given dataset |
| CO3 | Ability to design a model using machine learning to solve a problem. |
| CO4 | Ability to conduct practical experiments to solve problems using appropriate machine learning Techniques. |

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

```
import
pandas
as pd
        import numpy as np



        data = pd.read_csv("data - Sheet1.csv")

        print(data,"\n")
```

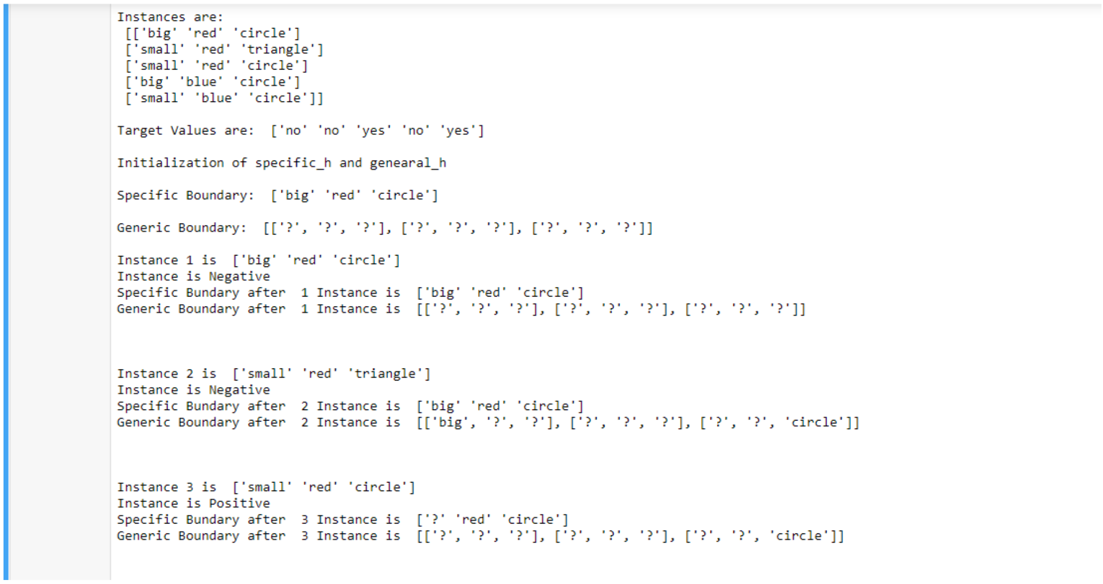
```
d = np.array(data)[:,:-1]

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


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


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))
```



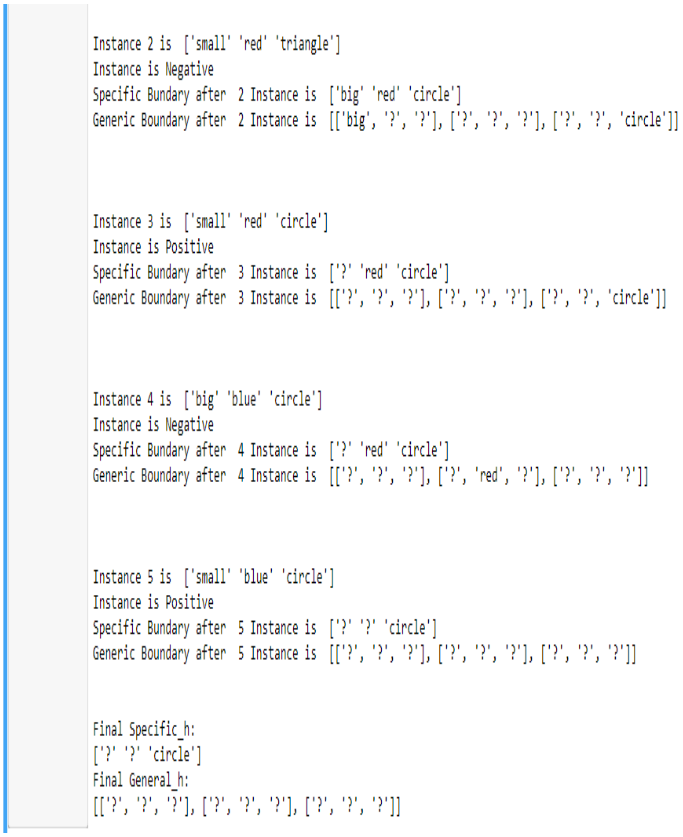
```
      Time Weather Temperature Company Humidity    Wind Goes
0  Morning   Sunny        Warm     Yes     Mild  Strong  Yes
1  Evening   Rainy        Cold      No     Mild  Normal   No
2  Morning   Sunny    Moderate     Yes   Normal  Normal  Yes
3  Evening   Sunny        Cold     Yes     High  Strong   No


 The attributes are:  [['Morning' 'Sunny' 'Warm' 'Yes' 'Mild' 'Strong']
 ['Evening' 'Rainy' 'Cold' 'No' 'Mild' 'Normal']
 ['Morning' 'Sunny' 'Moderate' 'Yes' 'Normal' 'Normal']
 ['Evening' 'Sunny' 'Cold' 'Yes' 'High' 'Strong']]

 The final hypothesis is: ['Morning' 'Sunny' '?' 'Yes' '?' '?']
```

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

```
import
numpy
as np
        import pandas as pd
        data = pd.read_csv('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")
```



```
Instances are:
 [['big' 'red' 'circle']
 ['small' 'red' 'triangle']
 ['small' 'red' 'circle']
 ['big' 'blue' 'circle']
 ['small' 'blue' 'circle']]

Target Values are:  ['no' 'no' 'yes' 'no' 'yes']

Initialization of specific_h and genearal_h

Specific Boundary:  ['big' 'red' 'circle']

Generic Boundary:  [['?', '?', '?'], ['?', '?', '?'], ['?', '?', '?']]

Instance 1 is  ['big' 'red' 'circle']
Instance is Negative
Specific Bundary after  1 Instance is  ['big' 'red' 'circle']
Generic Boundary after  1 Instance is  [['?', '?', '?'], ['?', '?', '?'], ['?', '?', '?']]



Instance 2 is  ['small' 'red' 'triangle']
Instance is Negative
Specific Bundary after  2 Instance is  ['big' 'red' 'circle']
Generic Boundary after  2 Instance is  [['big', '?', '?'], ['?', '?', '?'], ['?', '?', 'circle']]



Instance 3 is  ['small' 'red' 'circle']
Instance is Positive
Specific Bundary after  3 Instance is  ['?' 'red' 'circle']
Generic Boundary after  3 Instance is  [['?', '?', '?'], ['?', '?', '?'], ['?', '?', 'circle']]
```



```
Instance 2 is  ['small' 'red' 'triangle']
Instance is Negative
Specific Bundary after  2 Instance is  ['big' 'red' 'circle']
Generic Boundary after  2 Instance is  [['big', '?', '?'], ['?', '?', '?'], ['?', '?', 'circle']]



Instance 3 is  ['small' 'red' 'circle']
Instance is Positive
Specific Bundary after  3 Instance is  ['?' 'red' 'circle']
Generic Boundary after  3 Instance is  [['?', '?', '?'], ['?', '?', '?'], ['?', '?', 'circle']]



Instance 4 is  ['big' 'blue' 'circle']
Instance is Negative
Specific Bundary after  4 Instance is  ['?' 'red' 'circle']
Generic Boundary after  4 Instance is  [['?', '?', '?'], ['?', 'red', '?'], ['?', '?', '?']]



Instance 5 is  ['small' 'blue' 'circle']
Instance is Positive
Specific Bundary after  5 Instance is  ['?' '?' 'circle']
Generic Boundary after  5 Instance is  [['?', '?', '?'], ['?', '?', '?'], ['?', '?', '?']]


Final Specific_h:
['?' '?' 'circle']
Final General_h:
[['?', '?', '?'], ['?', '?', '?'], ['?', '?', '?']]
```

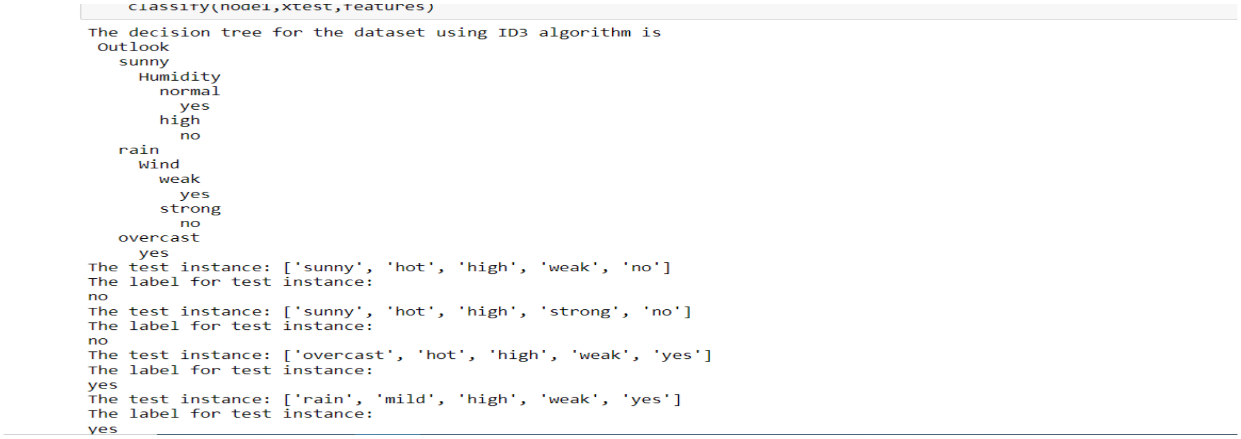
3.Write a program to demonstrate the working of the decision tree based ID3 algorithm. Use an appropriate data set for building the decision tree and apply this knowledge to classify a new sample.

```
import
math
        import csv
        def load_csv(filename):
        lines=csv.reader(open(“data.csv”,"r"))
        dataset = list(lines)
        headers = dataset.pop(0)
        return dataset,headers

        class Node:
```

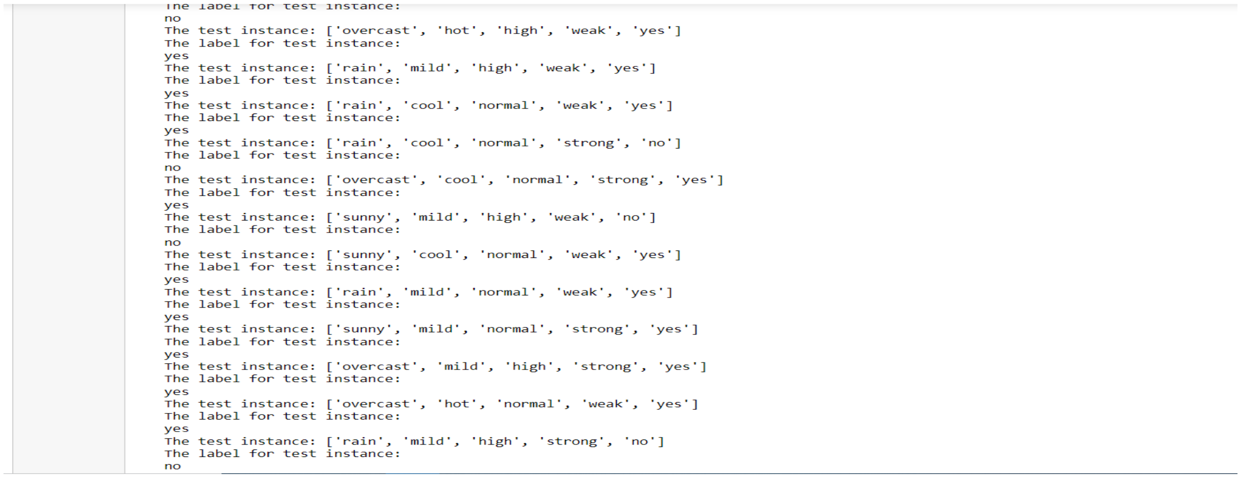
```
def __init__(self,attribute):
self.attribute=attribute
self.children=[]
self.answer=""

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("id3.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:")
classify(node1,xtest,features)
```



```
    classify(node1,xtest,features)
The decision tree for the dataset using ID3 algorithm is
 Outlook
   sunny
     Humidity
       normal
         yes
       high
         no
   rain
     Wind
       weak
         yes
       strong
         no
   overcast
     yes
The test instance: ['sunny', 'hot', 'high', 'weak', 'no']
The label for test instance:
no
The test instance: ['sunny', 'hot', 'high', 'strong', 'no']
The label for test instance:
no
The test instance: ['overcast', 'hot', 'high', 'weak', 'yes']
The label for test instance:
yes
The test instance: ['rain', 'mild', 'high', 'weak', 'yes']
The label for test instance:
yes
```

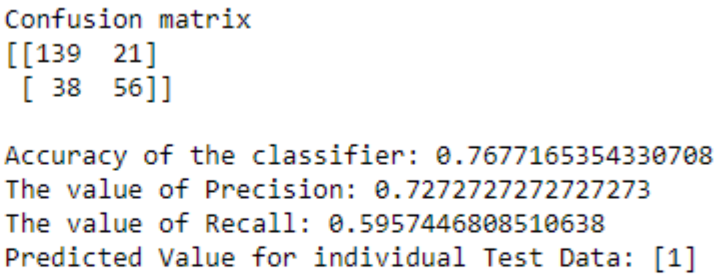


```
The label for test instance:
no
The test instance: ['overcast', 'hot', 'high', 'weak', 'yes']
The label for test instance:
yes
The test instance: ['rain', 'mild', 'high', 'weak', 'yes']
The label for test instance:
yes
The test instance: ['rain', 'cool', 'normal', 'weak', 'yes']
The label for test instance:
yes
The test instance: ['rain', 'cool', 'normal', 'strong', 'no']
The label for test instance:
no
The test instance: ['overcast', 'cool', 'normal', 'strong', 'yes']
The label for test instance:
yes
The test instance: ['sunny', 'mild', 'high', 'weak', 'no']
The label for test instance:
no
The test instance: ['sunny', 'cool', 'normal', 'weak', 'yes']
The label for test instance:
yes
The test instance: ['rain', 'mild', 'normal', 'weak', 'yes']
The label for test instance:
yes
The test instance: ['sunny', 'mild', 'normal', 'strong', 'yes']
The label for test instance:
yes
The test instance: ['overcast', 'mild', 'high', 'strong', 'yes']
The label for test instance:
yes
The test instance: ['overcast', 'hot', 'normal', 'weak', 'yes']
The label for test instance:
yes
The test instance: ['rain', 'mild', 'high', 'strong', 'no']
The label for test instance:
no
```

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

```
import
numpy
as np

        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

y = df[predicted_class_names].values

print(df.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)

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

predicted = clf.predict(xtest)

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

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)
```

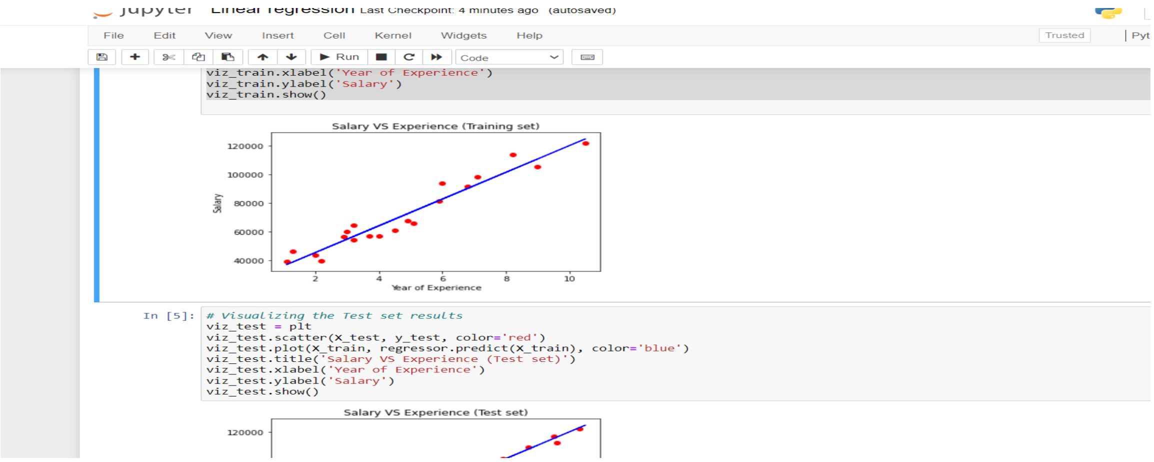


```
Confusion matrix
[[139  21]
 [ 38  56]]

Accuracy of the classifier: 0.7677165354330708
The value of Precision: 0.7272727272727273
The value of Recall: 0.5957446808510638
Predicted Value for individual Test Data: [1]
```

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

```
import
```

```
import matplotlib.pyplot as plt
import pandas as pd
dataset = pd.read_csv('salary.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)
LinearRegression()
# 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()
```



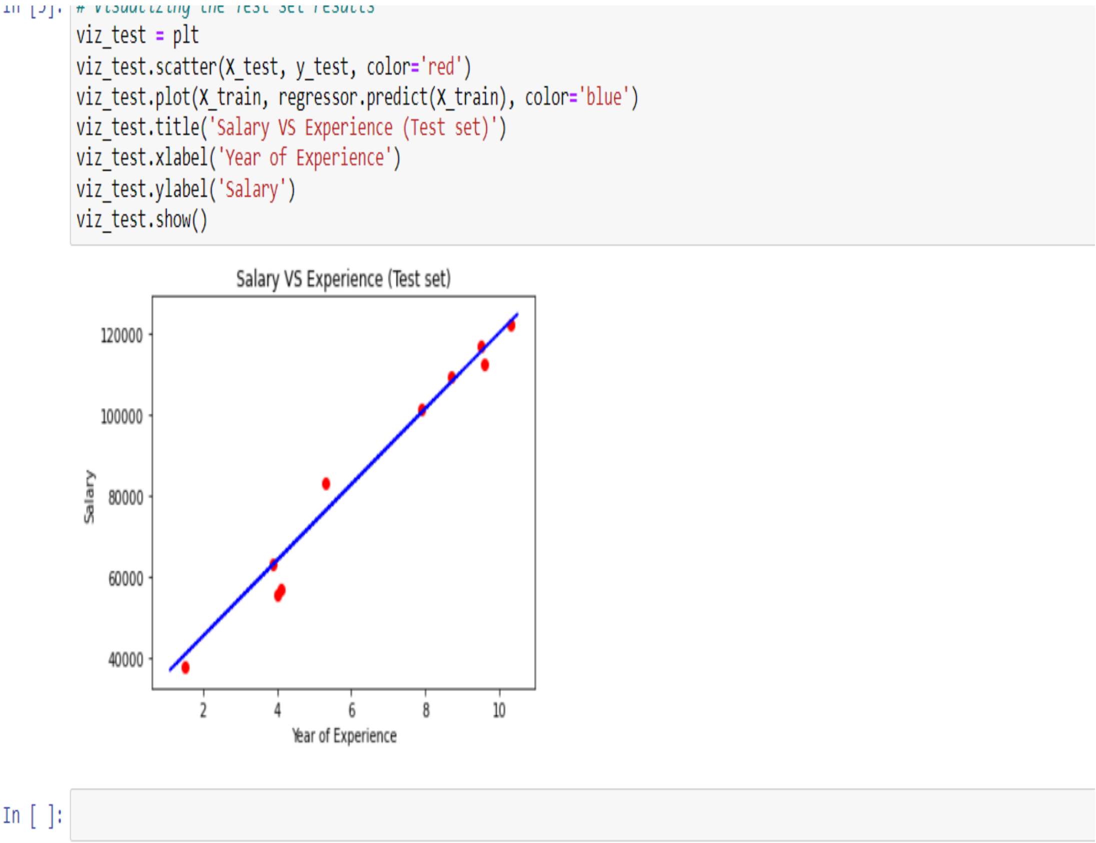
jupyter Linear regression Last Checkpoint: 4 minutes ago (autosaved)
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Trusted
Pyt
Run
Code

```
viz_train.xlabel('Year of Experience')
viz_train.ylabel('Salary')
viz_train.show()
```

Salary VS Experience (Training set)
120000
100000
80000
60000
40000
Salary
2
4
6
8
10
Year of Experience

```
In [5]: # 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()
```

Salary VS Experience (Test set)
120000



```
In [5]: # 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()
```

```
In [ ]:
```