#### VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.



## LAB REPORT on

# MACHINE LEARNING (20CS6PCMAL)

Submitted by

SAMPREETH (1BM19CS142)

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" was carried out by SAMPREETH P (1BM19CS142), who is a bona fide 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 the course MACHINE LEARNING (20CS6PCMAL) work prescribed for the said degree.

Name of the Lab-In charge Designation Department of CSE BMSCE, Bengaluru **SARITHA A N**Assistant Professor
Department of CSE
BMSCE, Bengaluru

## **Index Sheet**

SI.	Experiment Title	Page No.
No.		
1.	FIND-S :- Implement and demonstrate the FIND-S algorithm for	2
	finding the most specific hypothesis based on a given set of	
	training data samples.	
2.	CANDIDATE ELIMINATION :- For a given set of training data	9
	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.	
3.	<u>ID3</u> :- Write a program to demonstrate the working of the	18
	decision tree based ID3 algorithm. Use an appropriate data set for	
	building the decision tree and apply this knowledge to classify a	
	new sample.	
4.a.	NAÏVE BAYESIAN CLASSIFIER :- Write a program to implement the	31
	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	
4.b.	NAÏVE BAYESIAN CLASSIFIER (Without packages) :- Write a	39
	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.(Without packages)	
5.	<u>LINEAR REGRESSION</u> :- Implement the Linear Regression	54
	algorithm in order to fit data points. Select appropriate data set	
	for your experiment and draw graphs.	

### **Course Outcome :-**

#### At the end of the course the student will be able to

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.

#### Lab Program -1:-

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

```
+*In[1]:*+
[source, ipython3]
import csv
hypo = ['%','%','%','%','%','%'];
with open(r'C:\Users\Admin\OneDrive\Desktop\6th sem\ML\lab-ml\lab 1\finds.csv') as csv_file:
  readcsv = csv.reader(csv_file, delimiter=',')
  print(readcsv)
  data = []
  print("\nThe given training examples are:")
  for row in readcsv:
    print(row)
    if row[len(row)-1].upper() == "YES":
      data.append(row)
+*Out[1]:*+
<_csv.reader object at 0x0000013B7E4DFD60>
```

```
The given training examples are:
['sky', 'air temp', 'humidity', 'wind', 'water', 'forecast', 'enjoy sport']
['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']
+*In[2]:*+
[source, ipython3]
print("\nThe positive examples are:");
for x in data:
  print(x);
print("\n");
+*Out[2]:*+
The positive examples are:
['sunny', 'warm', 'normal', 'strong', 'warm', 'same', 'yes']
['sunny', 'warm', 'high', 'strong', 'warm', 'same', 'yes']
['sunny', 'warm', 'high', 'strong', 'cool', 'change', 'yes']
```

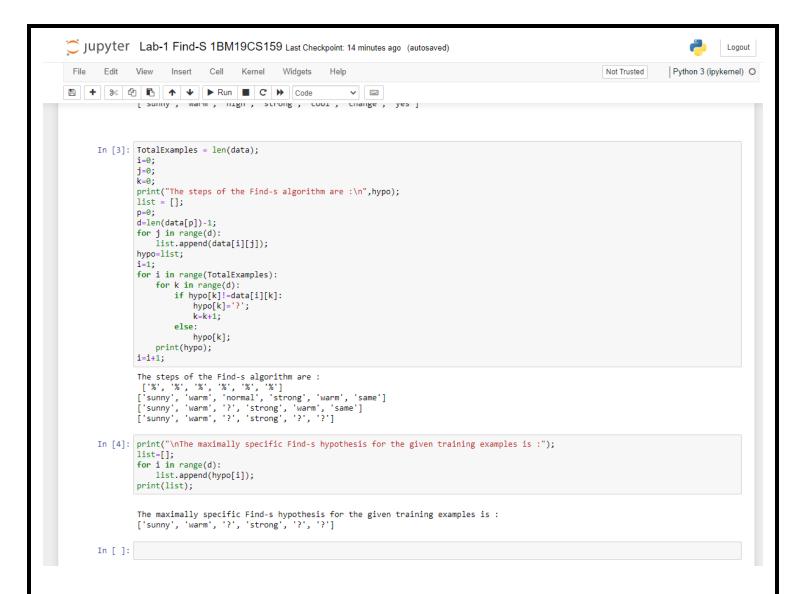
```
+*In[3]:*+
[source, ipython3]
TotalExamples = len(data);
i=0;
j=0;
k=0;
print("The steps of the Find-s algorithm are :\n",hypo);
list = [];
p=0;
d=len(data[p])-1;
for j in range(d):
  list.append(data[i][j]);
hypo=list;
i=1;
for i in range(TotalExamples):
  for k in range(d):
    if hypo[k]!=data[i][k]:
      hypo[k]='?';
      k=k+1;
    else:
      hypo[k];
  print(hypo);
i=i+1;
```

```
+*Out[3]:*+
The steps of the Find-s algorithm are:
['%', '%', '%', '%', '%', '%']
['sunny', 'warm', 'normal', 'strong', 'warm', 'same']
['sunny', 'warm', '?', 'strong', 'warm', 'same']
['sunny', 'warm', '?', 'strong', '?', '?']
+*In[4]:*+
[source, ipython3]
print("\nThe maximally specific Find-s hypothesis for the given training examples is :");
list=[];
for i in range(d):
  list.append(hypo[i]);
print(list);
+*Out[4]:*+
The maximally specific Find-s hypothesis for the given training examples is :
['sunny', 'warm', '?', 'strong', '?', '?']
```

```
+*In[]:*+
[source, ipython3]
----
```

#### Output screenshots :-

```
Jupyter Lab-1 Find-S Last Checkpoint: 12 minutes ago (autosaved)
                                                                                                                                                                                                      Logout
 File Edit View Insert Cell Kernel Widgets Help
                                                                                                                                                                  Not Trusted
                                                                                                                                                                                     Python 3 (ipykernel) O
In [1]: import csv
hypo = ['%','%','%','%','%','%'];
                    with open(r'C:\Users\Admin\OneDrive\Desktop\6th sem\ML\lab-ml\lab 1\finds.csv') as csv_file:
                         readcsv = csv.reader(csv_file, delimiter=',')
                         print(readcsv)
                         data = []
print("\nThe given training examples are:")
                          for row in readcsv:
                               print(row)
                                if row[len(row)-1].upper() == "YES":
                                     data.append(row)
                     <_csv.reader object at 0x0000013B7E4DFD60>
                    The given training examples are:
['sky', 'air temp', 'humidity', 'wind', 'water', 'forecast', 'enjoy sport']
['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']
        In [2]: print("\nThe positive examples are:");
                    for x in data:
                         print(x);
                    print("\n");
                     The positive examples are:
                    ['sunny', 'warm', 'normal', 'strong', 'warm', 'same', 'yes']
['sunny', 'warm', 'high', 'strong', 'warm', 'same', 'yes']
['sunny', 'warm', 'high', 'strong', 'cool', 'change', 'yes']
```



-								
4	Α	В	С	D	Е	F	G	Н
1	sky	air temp	humidity	wind	water	forecast	enjoy sport	
2	sunny	warm	normal	strong	warm	same	yes	
3	sunny	warm	high	strong	warm	same	yes	
4	rainy	cold	high	strong	warm	change	no	
5	sunny	warm	high	strong	cool	change	yes	
6								
7								
0								

#### Lab 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.

```
+*In[7]:*+
[source, ipython3]
import numpy as np
import pandas as pd
+*In[10]:*+
[source, ipython3]
# Loading Data from a CSV File
data = pd.DataFrame(data=pd.read_csv(r'C:\Users\Admin\OneDrive\Desktop\6th
sem\ML\lab-ml\lab 2\trainingdata.csv'))
print(data)
+*Out[10]:*+
```

```
sky airtemp humidity wind water forecast enjoySport
O Sunny Warm Normal Strong Warm
                                                 Yes
                                        Same
1 Sunny Warm High Strong Warm
                                                Yes
2 Rainy Cold
               High Strong Warm Change
                                               No
3 Sunny Warm High Strong Cool Change
                                               Yes
+*In[11]:*+
[source, ipython3]
# Separating concept features from Target
concepts = np.array(data.iloc[:,0:-1])
print(concepts)
+*Out[11]:*+
[['Sunny' 'Warm' 'Normal' 'Strong' 'Warm' 'Same']
['Sunny' 'Warm' 'High' 'Strong' 'Warm' 'Same']
['Rainy' 'Cold' 'High' 'Strong' 'Warm' 'Change']
['Sunny' 'Warm' 'High' 'Strong' 'Cool' 'Change']]
```

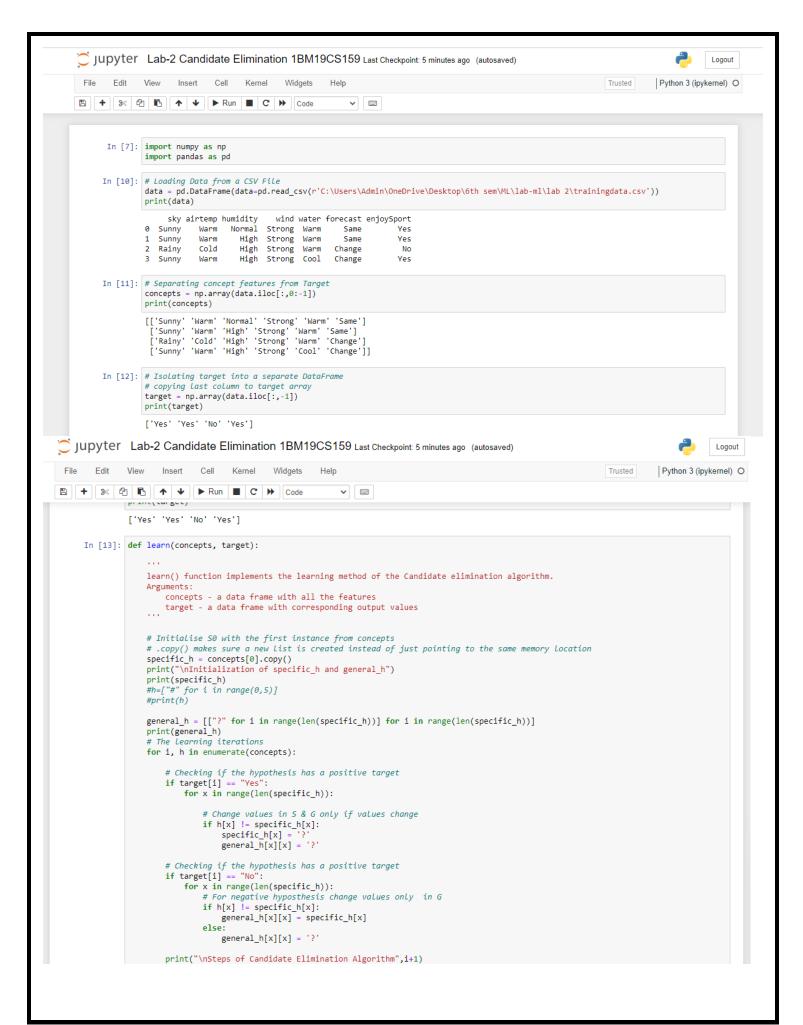
```
+*In[12]:*+
[source, ipython3]
# Isolating target into a separate DataFrame
# copying last column to target array
target = np.array(data.iloc[:,-1])
print(target)
+*Out[12]:*+
['Yes' 'Yes' 'No' 'Yes']
+*In[13]:*+
[source, ipython3]
def learn(concepts, target):
  111
  learn() function implements the learning method of the Candidate elimination algorithm.
  Arguments:
    concepts - a data frame with all the features
```

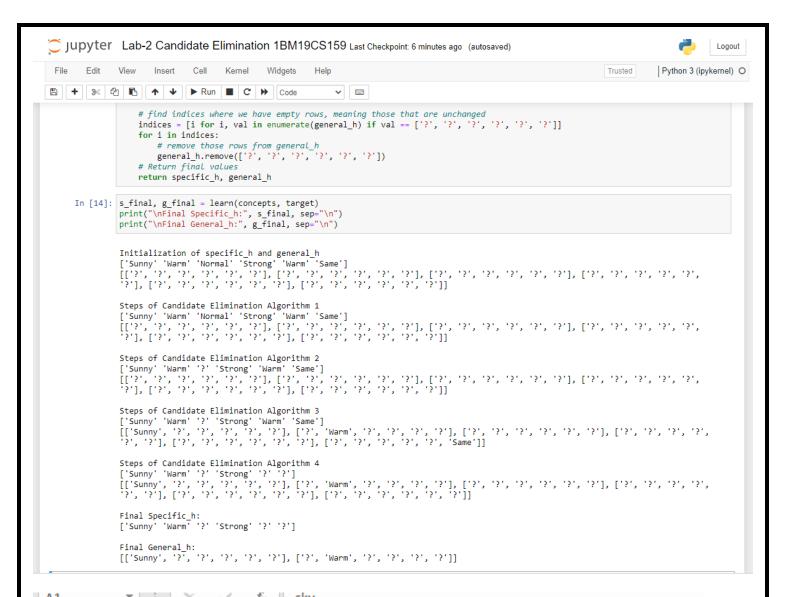
```
target - a data frame with corresponding output values
  111
  # Initialise SO with the first instance from concepts
  #.copy() makes sure a new list is created instead of just pointing to the same memory
location
  specific_h = concepts[0].copy()
  print("\nInitialization of specific_h and general_h")
  print(specific_h)
  #h=["#" for i in range(0,5)]
  #print(h)
  general h = [["?" for i in range(len(specific h))] for i in range(len(specific h))]
  print(general_h)
  # The learning iterations
  for i, h in enumerate(concepts):
    # Checking if the hypothesis has a positive target
    if target[i] == "Yes":
      for x in range(len(specific_h)):
         # Change values in S & G only if values change
         if h[x] != specific_h[x]:
           specific h[x] = '?'
           general_h[x][x] = '?'
    # Checking if the hypothesis has a positive target
```

```
if target[i] == "No":
       for x in range(len(specific h)):
         # For negative hyposthesis change values only in G
         if h[x] != specific h[x]:
           general_h[x][x] = specific_h[x]
         else:
           general_h[x][x] = '?'
    print("\nSteps of Candidate Elimination Algorithm",i+1)
    print(specific_h)
    print(general_h)
  # find indices where we have empty rows, meaning those that are unchanged
  indices = [i for i, val in enumerate(general_h) if val == ['?', '?', '?', '?', '?', '?']]
  for i in indices:
    # remove those rows from general_h
    general_h.remove(['?', '?', '?', '?', '?', '?'])
  # Return final values
  return specific_h, general_h
+*In[14]:*+
[source, ipython3]
s_final, g_final = learn(concepts, target)
```

```
print("\nFinal Specific h:", s final, sep="\n")
print("\nFinal General h:", g final, sep="\n")
+*Out[14]:*+
Initialization of specific_h and general_h
['Sunny' 'Warm' 'Normal' 'Strong' 'Warm' 'Same']
'?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?']
Steps of Candidate Elimination Algorithm 1
['Sunny' 'Warm' 'Normal' 'Strong' 'Warm' 'Same']
'?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?']]
Steps of Candidate Elimination Algorithm 2
['Sunny' 'Warm' '?' 'Strong' 'Warm' 'Same']
'?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?']]
Steps of Candidate Elimination Algorithm 3
['Sunny' 'Warm' '?' 'Strong' 'Warm' 'Same']
'?'], ['?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', 'Same']]
```

```
Steps of Candidate Elimination Algorithm 4
['Sunny' 'Warm' '?' 'Strong' '?' '?']
'?'], ['?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?']]
Final Specific_h:
['Sunny' 'Warm' '?' 'Strong' '?' '?']
Final General_h:
[['Sunny', '?', '?', '?', '?'], ['?', 'Warm', '?', '?', '?', '?']]
+*In[]:*+
[source, ipython3]
Output screenshots:-
```





4	Α	В	С	D	E	F	G	Н	1	J
1	sky	airtemp	humidity	wind	water	forecast	enjoySpor	t		
2	Sunny	Warm	Normal	Strong	Warm	Same	Yes			
3	Sunny	Warm	High	Strong	Warm	Same	Yes			
4	Rainy	Cold	High	Strong	Warm	Change	No			
5	Sunny	Warm	High	Strong	Cool	Change	Yes			
6										
7										
8										
9										

#### Lab 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.

```
+*In[1]:*+
[source, ipython3]
import numpy as np
import math
import csv
+*In[2]:*+
[source, ipython3]
def read_data(filename):
  with open(filename, 'r') as csvfile:
    datareader = csv.reader(csvfile, delimiter=',')
    headers = next(datareader)
    metadata = []
    traindata = []
    for name in headers:
      metadata.append(name)
```

```
for row in datareader:
       traindata.append(row)
  return (metadata, traindata)
+*In[5]:*+
[source, ipython3]
class Node:
  def __init__(self, attribute):
    self.attribute = attribute
    self.children = []
    self.answer = ""
  def __str__(self):
    return self.attribute
+*In[6]:*+
[source, ipython3]
def subtables(data, col, delete):
  dict = \{\}
```

```
items = np.unique(data[:, col])
  count = np.zeros((items.shape[0], 1), dtype=np.int32)
  for x in range(items.shape[0]):
    for y in range(data.shape[0]):
      if data[y, col] == items[x]:
         count[x] += 1
  for x in range(items.shape[0]):
    dict[items[x]] = np.empty((int(count[x]), data.shape[1]), dtype="|S32")
    pos = 0
    for y in range(data.shape[0]):
      if data[y, col] == items[x]:
         dict[items[x]][pos] = data[y]
         pos += 1
    if delete:
       dict[items[x]] = np.delete(dict[items[x]], col, 1)
  return items, dict
+*In[7]:*+
[source, ipython3]
def entropy(S):
```

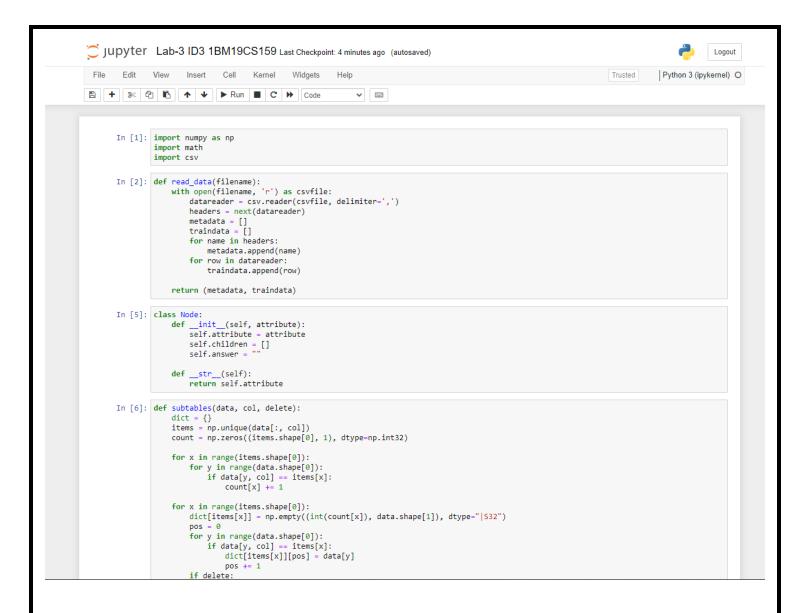
```
items = np.unique(S)
  if items.size == 1:
    return 0
  counts = np.zeros((items.shape[0], 1))
  sums = 0
  for x in range(items.shape[0]):
    counts[x] = sum(S == items[x]) / (S.size * 1.0)
  for count in counts:
    sums += -1 * count * math.log(count, 2)
  return sums
+*In[8]:*+
[source, ipython3]
def gain_ratio(data, col):
  items, dict = subtables(data, col, delete=False)
  total_size = data.shape[0]
  entropies = np.zeros((items.shape[0], 1))
  intrinsic = np.zeros((items.shape[0], 1))
```

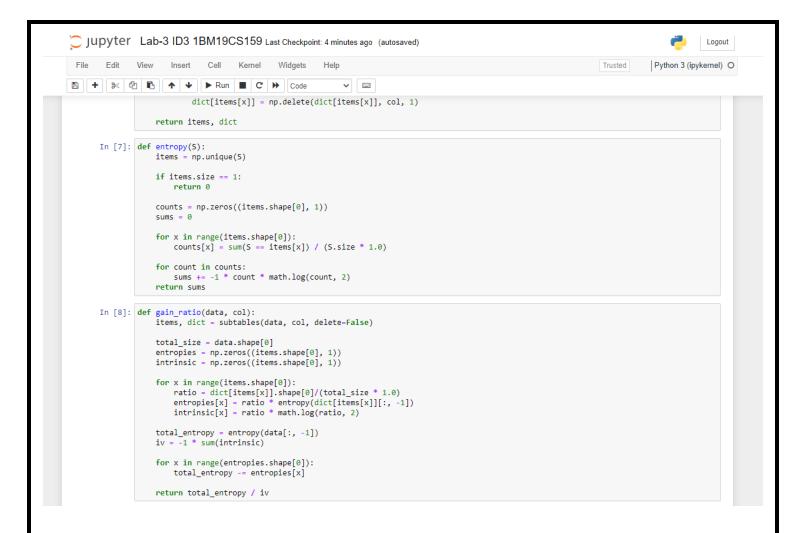
```
for x in range(items.shape[0]):
    ratio = dict[items[x]].shape[0]/(total_size * 1.0)
    entropies[x] = ratio * entropy(dict[items[x]][:, -1])
    intrinsic[x] = ratio * math.log(ratio, 2)
  total_entropy = entropy(data[:, -1])
  iv = -1 * sum(intrinsic)
  for x in range(entropies.shape[0]):
    total_entropy -= entropies[x]
  return total_entropy / iv
+*In[9]:*+
[source, ipython3]
def create_node(data, metadata):
  if (np.unique(data[:, -1])).shape[0] == 1:
    node = Node("")
    node.answer = np.unique(data[:, -1])[0]
    return node
  gains = np.zeros((data.shape[1] - 1, 1))
```

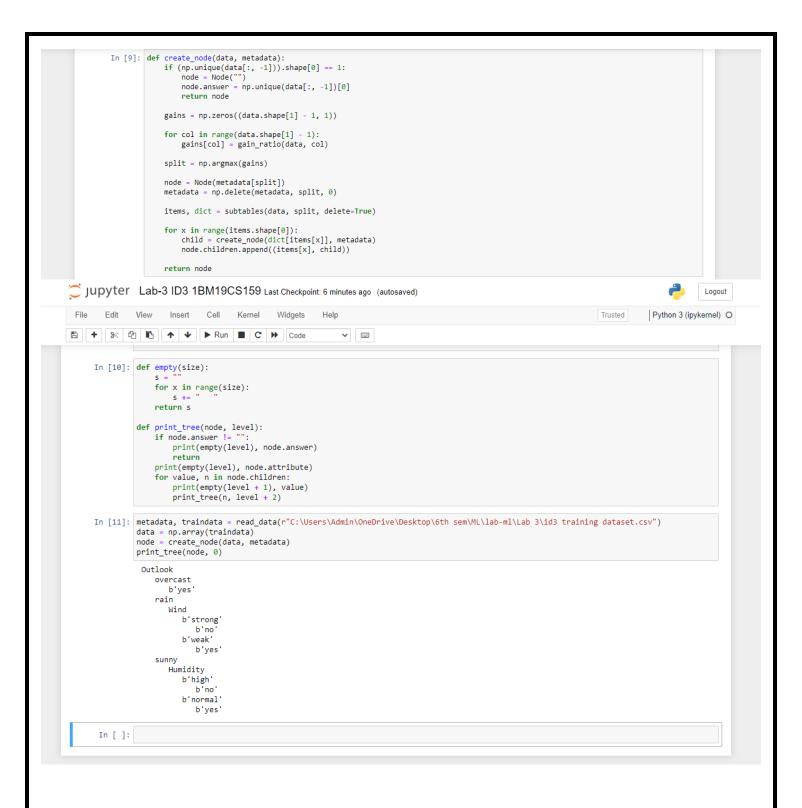
```
for col in range(data.shape[1] - 1):
    gains[col] = gain_ratio(data, col)
  split = np.argmax(gains)
  node = Node(metadata[split])
  metadata = np.delete(metadata, split, 0)
  items, dict = subtables(data, split, delete=True)
  for x in range(items.shape[0]):
    child = create_node(dict[items[x]], metadata)
    node.children.append((items[x], child))
  return node
+*In[10]:*+
[source, ipython3]
def empty(size):
  s = ""
  for x in range(size):
    s += " "
```

```
return s
def print_tree(node, level):
  if node.answer != "":
    print(empty(level), node.answer)
    return
  print(empty(level), node.attribute)
  for value, n in node.children:
    print(empty(level + 1), value)
    print_tree(n, level + 2)
+*In[11]:*+
[source, ipython3]
metadata, traindata = read_data(r"C:\Users\Admin\OneDrive\Desktop\6th sem\ML\label{eq:metadata})
ml\Lab 3\id3 training dataset.csv")
data = np.array(traindata)
node = create_node(data, metadata)
print_tree(node, 0)
+*Out[11]:*+
Outlook
```

```
overcast
   b'yes'
  rain
   Wind
     b'strong'
       b'no'
     b'weak'
       b'yes'
  sunny
   Humidity
     b'high'
       b'no'
     b'normal'
       b'yes'
+*In[]:*+
[source, ipython3]
Output screenshots :-
```







A	A1 * : × ✓ f <sub>x</sub> Outlook									
4	Α	В	С	D	E	F	G			
1	Outlook	Temperat	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					
16										
17										
18										

#### Lab Program -4.a.:-

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

```
+*In[1]:*+
[source, ipython3]
# import necessary libarities
import pandas as pd
from sklearn import tree
from sklearn.preprocessing import LabelEncoder
from sklearn.naive_bayes import GaussianNB
# load data from CSV
data = pd.read csv(r"C:\Users\Admin\OneDrive\Desktop\6th sem\ML\lab-ml\Lab 4\Naive
Bayesian classifier training dataset.csv")
print("THe first 5 values of data is :\n",data.head())
+*Out[1]:*+
THe first 5 values of data is:
  Outlook Temperature Humidity Windy PlayTennis
```

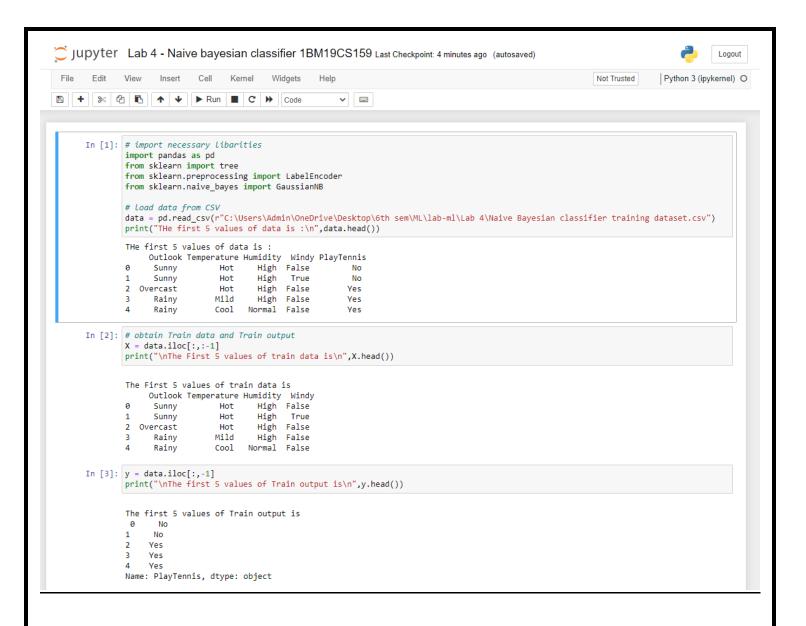
```
Sunny
              Hot
                    High False
0
                                   No
                    High True
   Sunny
1
              Hot
                                   No
2 Overcast
               Hot
                     High False
                                    Yes
                    High False
    Rainy
             Mild
3
                                  Yes
    Rainy
             Cool Normal False
                                    Yes
+*In[2]:*+
[source, ipython3]
# obtain Train data and Train output
X = data.iloc[:,:-1]
print("\nThe First 5 values of train data is\n",X.head())
+*Out[2]:*+
The First 5 values of train data is
  Outlook Temperature Humidity Windy
   Sunny
                    High False
              Hot
0
                    High True
   Sunny
              Hot
2 Overcast
               Hot High False
             Mild
                    High False
    Rainy
3
```

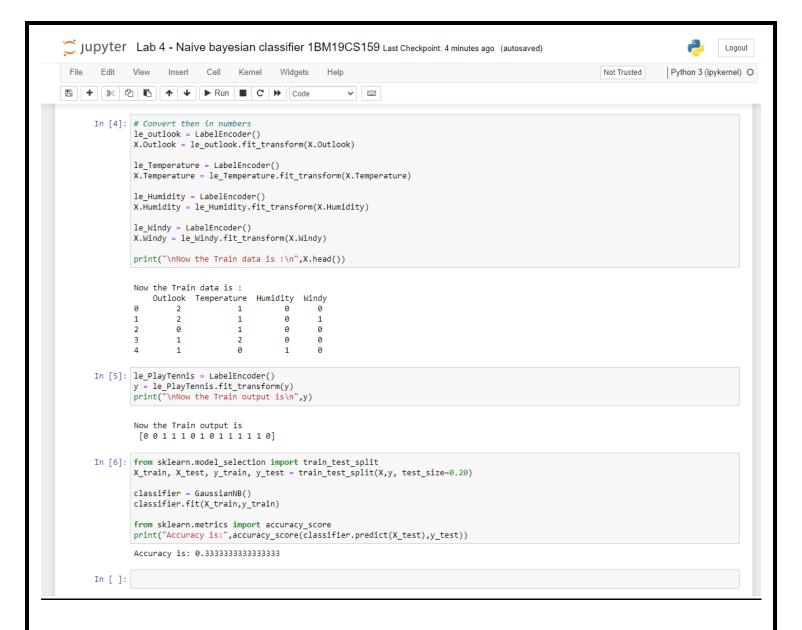
```
Rainy Cool Normal False
+*In[3]:*+
[source, ipython3]
y = data.iloc[:,-1]
print("\nThe first 5 values of Train output is\n",y.head())
+*Out[3]:*+
The first 5 values of Train output is
    No
0
    No
  Yes
  Yes
4 Yes
Name: PlayTennis, dtype: object
+*In[4]:*+
```

```
[source, ipython3]
# Convert then in numbers
le_outlook = LabelEncoder()
X.Outlook = le_outlook.fit_transform(X.Outlook)
le_Temperature = LabelEncoder()
X.Temperature = le_Temperature.fit_transform(X.Temperature)
le_Humidity = LabelEncoder()
X.Humidity = le_Humidity.fit_transform(X.Humidity)
le_Windy = LabelEncoder()
X.Windy = le_Windy.fit_transform(X.Windy)
print("\nNow the Train data is :\n",X.head())
+*Out[4]:*+
Now the Train data is:
  Outlook Temperature Humidity Windy
     2
             1
                       0
0
     2
1
             1
                   0
                       1
```

```
0
2
     0
              1
                    0
3
     1
              2
                    0
                         0
     1
                    1
                         0
              0
+*In[5]:*+
[source, ipython3]
le_PlayTennis = LabelEncoder()
y = le_PlayTennis.fit_transform(y)
print("\nNow the Train output is\n",y)
+*Out[5]:*+
Now the Train output is
[0\ 0\ 1\ 1\ 1\ 0\ 1\ 0\ 1\ 1\ 1\ 1\ 1\ 0]
+*In[6]:*+
[source, ipython3]
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.20)
classifier = GaussianNB()
classifier.fit(X_train,y_train)
from sklearn.metrics import accuracy_score
print("Accuracy is:",accuracy_score(classifier.predict(X_test),y_test))
+*Out[6]:*+
Accuracy is: 0.33333333333333333
+*In[]:*+
[source, ipython3]
Output screenshots:-
```





A1	L	<b>-</b>	×	f <sub>x</sub> Ou	tlook	
4	Α	В	С	D	Е	F
1	Outlook	Temperat	Humidity	Windy	PlayTennis	
2	Sunny	Hot	High	FALSE	No	
3	Sunny	Hot	High	TRUE	No	
4	Overcast	Hot	High	FALSE	Yes	
5	Rainy	Mild	High	FALSE	Yes	
6	Rainy	Cool	Normal	FALSE	Yes	
7	Rainy	Cool	Normal	TRUE	No	
8	Overcast	Cool	Normal	TRUE	Yes	
9	Sunny	Mild	High	FALSE	No	
10	Sunny	Cool	Normal	FALSE	Yes	
11	Rainy	Mild	Normal	FALSE	Yes	
12	Sunny	Mild	Normal	TRUE	Yes	
13	Overcast	Mild	High	TRUE	Yes	
14	Overcast	Hot	Normal	FALSE	Yes	
15	Rainy	Mild	High	TRUE	No	
16						
17						
18						
10						

#### Lab Program -4.b.:-

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 (without packages).

# Source code and output :-

classes.append(dataset[i][-1])

```
+*In[1]:*+
[source, ipython3]
import math
import csv
import random
+*In[2]:*+
[source, ipython3]
# This make sures that the dataset is in an ordered format. If we have some arbirary names in
that column it difficult to deal with that.
def encode_class(dataset):
 classes=[]
 for i in range(len(dataset)):
  if dataset[i][-1] not in classes:
```

```
# Looping across the classes which we have derived above. This will make sure that we have
definitive classes (numeric) and not arbitrary
 for i in range(len(classes)):
  # Looping across all rows of dataset
  for j in range(len(dataset)):
   if dataset[j][-1] == classes[i]:
    dataset[j][-1]=i
 return dataset
+*In[3]:*+
[source, ipython3]
# Splitting the data between training set and testing set. Normally its a general understanding
the training:testing=7:3
def train test split(dataset,ratio):
 test num=int(ratio*len(dataset))
 train=list(dataset)
 test=[]
 for i in range(test num):
  rand=random.randrange(len(train))
  test.append(train.pop(rand))
 return train, test
```

```
+*In[4]:*+
[source, ipython3]
# Now depending on resultant value (last column values), we need to group the rows. It will be
usefult for calculating mean and std_dev
def groupUnderClass(train):
 dict={}
 for row in train:
  if row[-1] not in dict:
   dict[row[-1]]=[]
  dict[row[-1]].append(row)
 return dict
+*In[5]:*+
[source, ipython3]
# Standard formulae (just by-heart)
def mean(val):
 return sum(val)/float(len(val)) #Obvious
def stdDev(val):
```

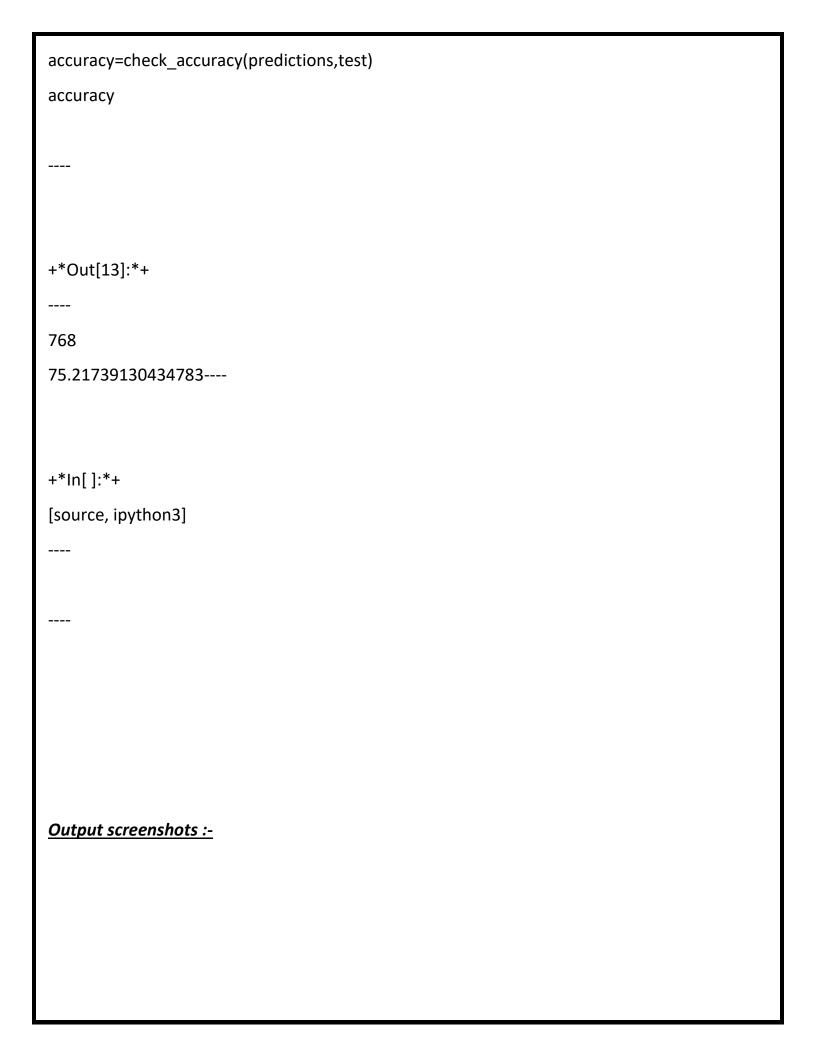
```
avg=mean(val)
 variance=sum([pow(x-avg,2) for x in val])/float(len(val)-1) # Especially this one
 return math.sqrt(variance)
+*In[6]:*+
[source, ipython3]
# We will calculte the mean and std dev with respect to each attribute. Important while
calculating gaussian probablity
def meanStdDev(instances):
 info=[(mean(x),stdDev(x)) for x in zip(*instances)] # Here we are taking complete column's
values of all instances.
 del info[-1]
 return info
+*In[7]:*+
[source, ipython3]
# As explained earlier why e need to group. We will be calculating the mean and std dev with
respect each class.
```

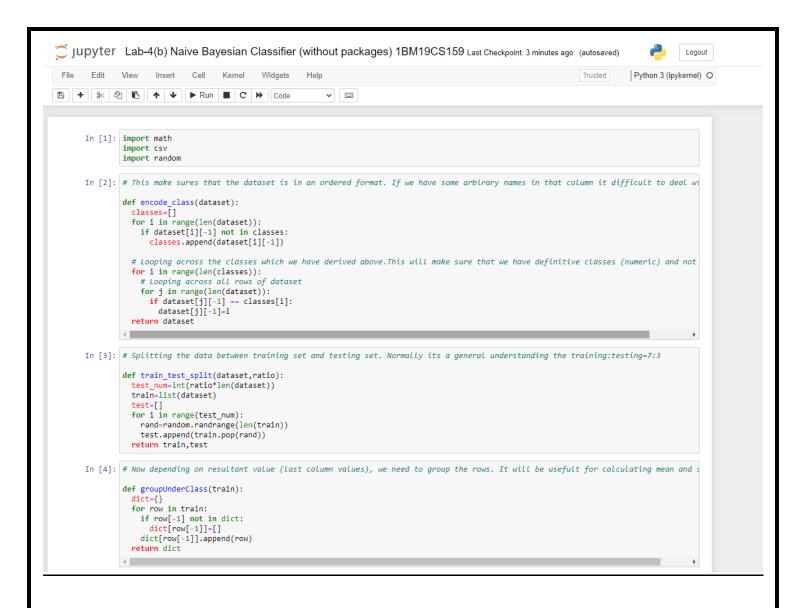
```
def MeanAndStdDevForClass(train):
 info={}
 dictionary=groupUnderClass(train)
 # print(dictionary)
 for key, value in dictionary.items():
  # dictionary[key]=meanStdDev(value)
  info[key]=meanStdDev(value) #Here value stands for a complete group.
 return info
+*In[8]:*+
[source, ipython3]
# Its a formula by heart (no choice)
def calculateGaussianProbablity(x,mean,std_dev):
 expo = math.exp(-(math.pow(x - mean, 2) / (2 * math.pow(std_dev, 2))))
 return (1 / (math.sqrt(2 * math.pi) * std_dev)) * expo
+*In[9]:*+
[source, ipython3]
```

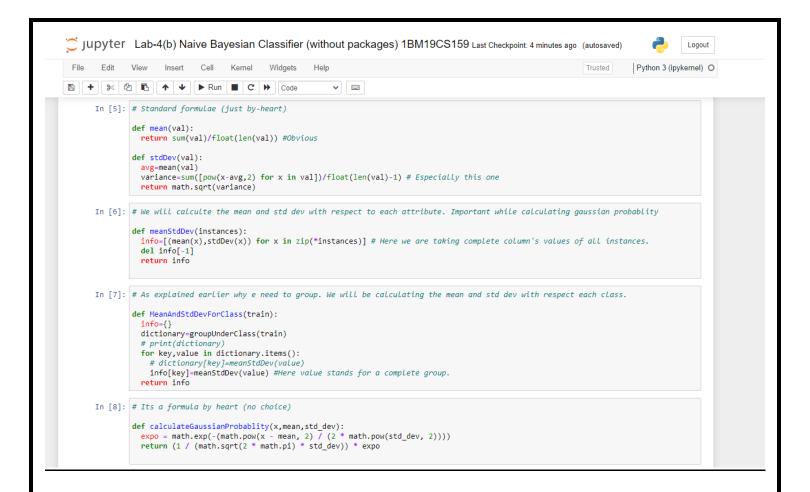
```
# After calculating mean and std dev w.r.t training data now its time to check if the logic will
work on testing data
def calculateClassProbablities(info,ele):
 probablities={}
 for key, summaries in info.items(): # Info contains the groupName (key) and list of
(mean,std_dev) for each attribute of that group
  probablities[key]=1
  for i in range(len(summaries)): #Loop across all attributes
   mean,std_dev=summaries[i]
   x=ele[i] # Testing data's one instance's attribute value.
   probablities[key] *= calculateGaussianProbablity(x, mean, std_dev)
 return probablities
+*In[10]:*+
[source, ipython3]
def predict(info,ele):
 probablities=calculateClassProbablities(info,ele) # returns a dictionary of probablities for each
group
 bestLabel,bestProb=None,-1
 # Consider group name whichever gives you the highest probablities for this instance of
testing data
```

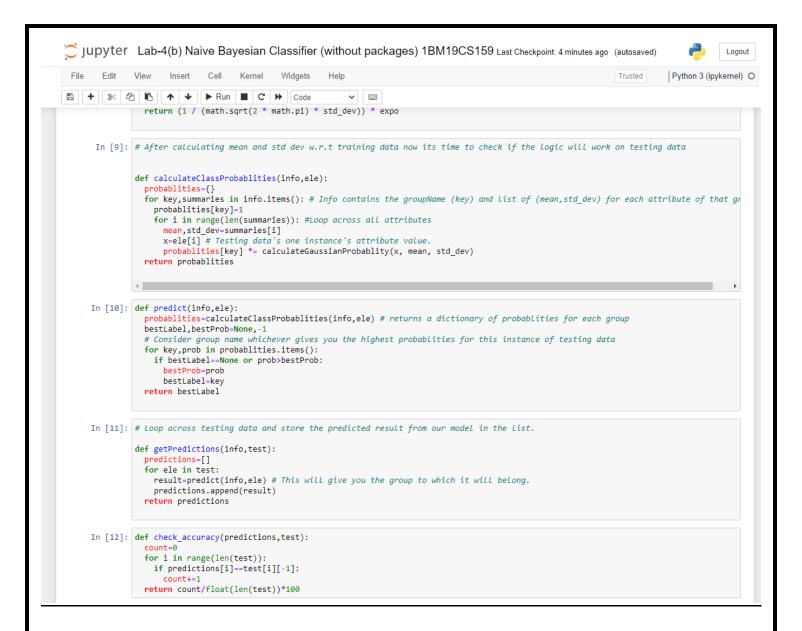
```
for key,prob in probablities.items():
  if bestLabel==None or prob>bestProb:
   bestProb=prob
   bestLabel=key
 return bestLabel
+*In[11]:*+
[source, ipython3]
# Loop across testing data and store the predicted result from our model in the list.
def getPredictions(info,test):
 predictions=[]
 for ele in test:
  result=predict(info,ele) # This will give you the group to which it will belong.
  predictions.append(result)
 return predictions
+*In[12]:*+
[source, ipython3]
```

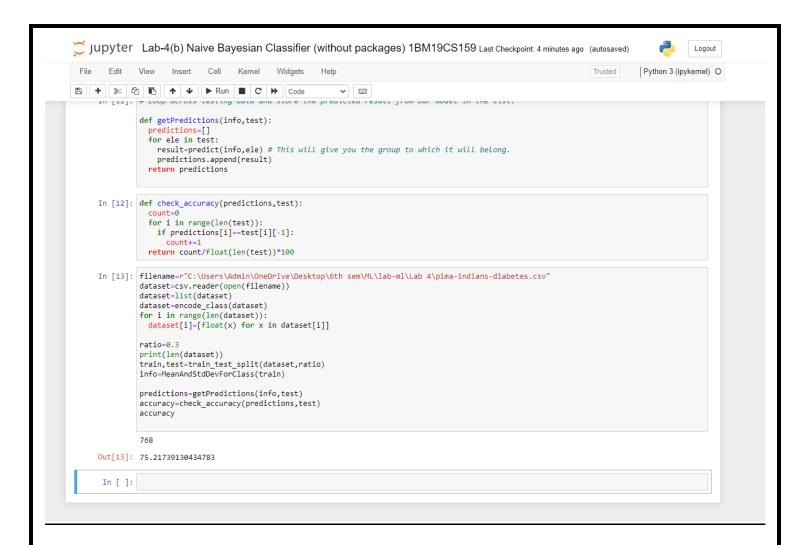
```
def check_accuracy(predictions,test):
 count=0
 for i in range(len(test)):
  if predictions[i]==test[i][-1]:
   count+=1
 return count/float(len(test))*100
+*In[13]:*+
[source, ipython3]
filename=r"C:\Users\Admin\OneDrive\Desktop\6th sem\ML\lab-ml\Lab 4\pima-indians-
diabetes.csv"
dataset=csv.reader(open(filename))
dataset=list(dataset)
dataset=encode_class(dataset)
for i in range(len(dataset)):
 dataset[i]=[float(x) for x in dataset[i]]
ratio=0.3
print(len(dataset))
train,test=train_test_split(dataset,ratio)
info=MeanAndStdDevForClass(train)
predictions=getPredictions(info,test)
```

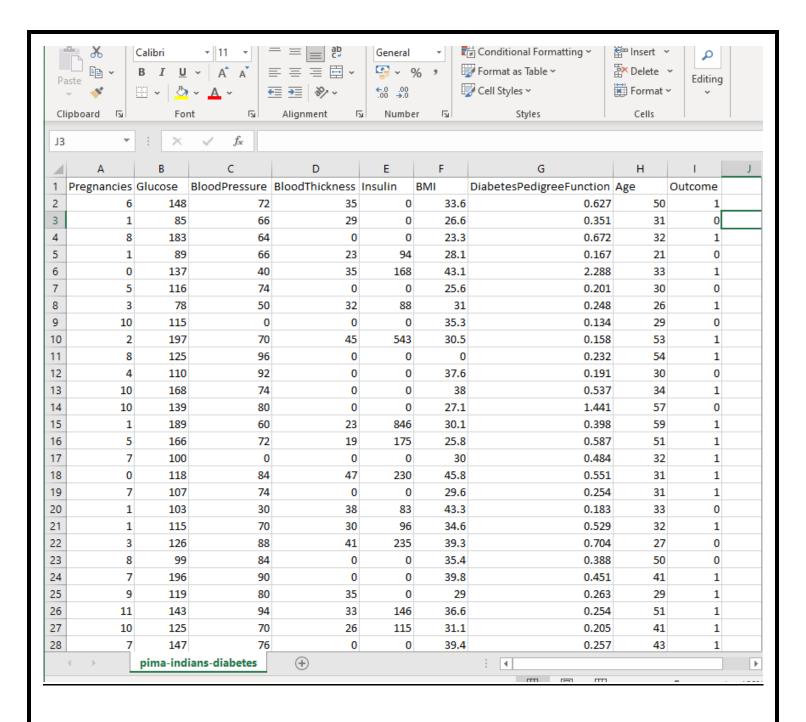












## Lab Program -5.:-

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

## Source code and output :-

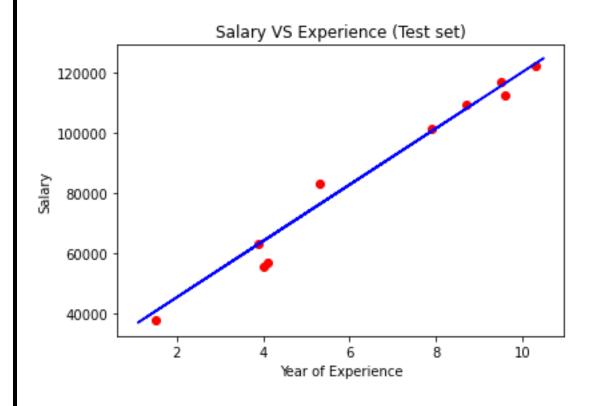
```
+*In[1]:*+
[source, ipython3]
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
+*In[11]:*+
[source, ipython3]
dataset = pd.read\_csv(r"C:\Users\Admin\OneDrive\Desktop\6th\ sem\ML\lab-ml\Lab\ 5\Lr-ml\Lab\ 5\Lr-ml\ 5\L
Salary Dataset.csv")
X = dataset.iloc[:,:-1].values
y = dataset.iloc[:, 1].values
+*In[13]:*+
```

```
[source, ipython3]
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)
+*In[14]:*+
[source, ipython3]
# Fitting Simple Linear Regression to the Training set
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train, y_train)
+*Out[14]:*+
----LinearRegression()----
+*In[15]:*+
[source, ipython3]
# Predicting the Test set results
y_pred = regressor.predict(X_test)
```

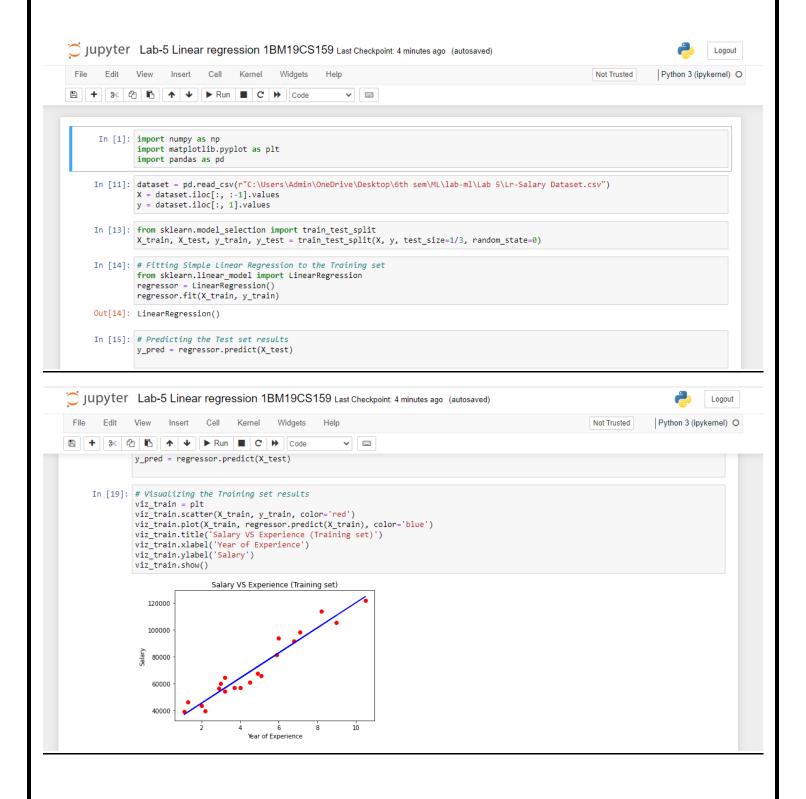
```
+*In[19]:*+
[source, ipython3]
# Visualizing the Training set results
viz_train = plt
viz_train.scatter(X_train, y_train, color='red')
viz_train.plot(X_train, regressor.predict(X_train), color='blue')
viz_train.title('Salary VS Experience (Training set)')
viz_train.xlabel('Year of Experience')
viz_train.ylabel('Salary')
viz_train.show()
+*Out[19]:*+
![png](output_5_0.png)
+*In[17]:*+
[source, ipython3]
```

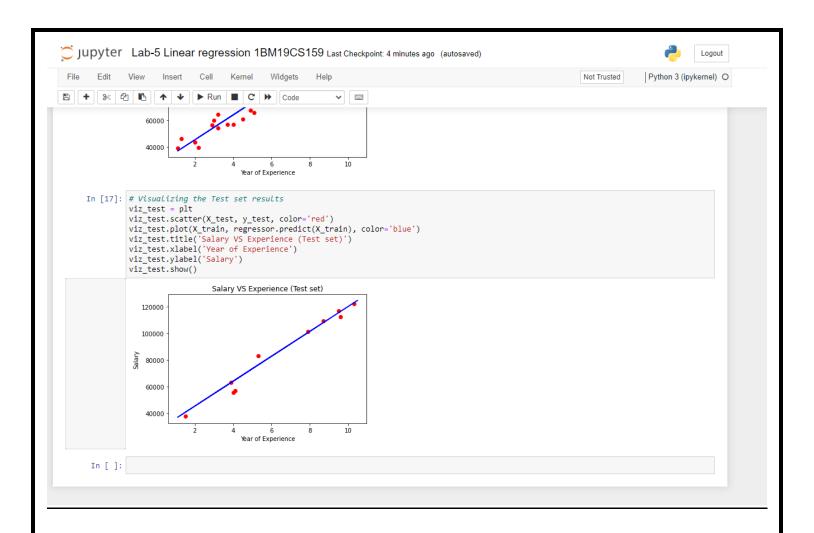
```
# Visualizing the Test set results
viz_test = plt
viz_test.scatter(X_test, y_test, color='red')
viz_test.plot(X_train, regressor.predict(X_train), color='blue')
viz_test.title('Salary VS Experience (Test set)')
viz_test.xlabel('Year of Experience')
viz_test.ylabel('Salary')
viz_test.show()
+*Out[17]:*+
![png](output_6_0.png)
+*In[]:*+
[source, ipython3]
```





#### Output screenshots :-





		format.							
	A1	L * :	× ✓	fx	Year				
ľ	4	А	В	С					
	1	YearsExperience	Salary						
	2	1.1	39343						
	3	1.3	46205						
	4	1.5	37731						
П	5	2	43525						
	6	2.2	39891						
	7	2.9	56642						
	8	3	60150						
	9	3.2	54445						
	10	3.2	64445						
	11	3.7	57189						
	12	3.9	63218						
8	13	4	55794						
	14	4	56957						
п	15	4.1	57081						
	16	4.5	61111						
	17	4.9	67938						
	18	5.1	66029						
	19	5.3	83088						
	20	5.9	81363						
	21	6	93940						
	22	6.8	91738						
	23	7.1	98273						
	24	7.9	101302						
	25	8.2	113812						
	Lr-Salary Dataset +								