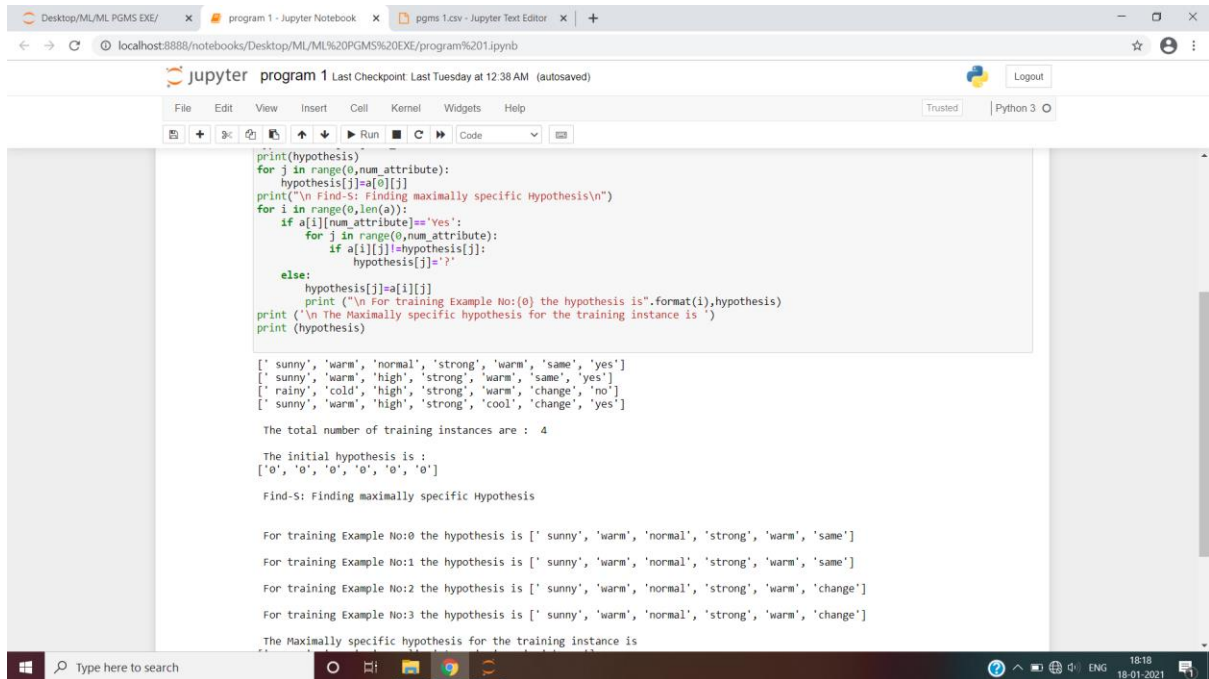


ML PROGRAMS OUTPUT

Program1 output



The screenshot shows a Jupyter Notebook interface with a single code cell. The code implements a function to find a maximally specific hypothesis from a set of training examples. The output of the code is displayed below the code cell.

```
print(hypothesis)
for j in range(0,num_attribute):
    hypothesis[j]=a[0][j]
print("\n Find-S: Finding maximally specific Hypothesis\n")
for i in range(0,len(a)):
    if a[i][num_attribute]!='Yes':
        for j in range(0,num_attribute):
            if a[i][j]!=hypothesis[j]:
                hypothesis[j]='?'
        else:
            hypothesis[j]=a[i][j]
print("\n For training Example No:0 the hypothesis is".format(i),hypothesis)
print("\n The Maximally specific hypothesis for the training instance is ")
print (hypothesis)

[' 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']

The total number of training instances are : 4

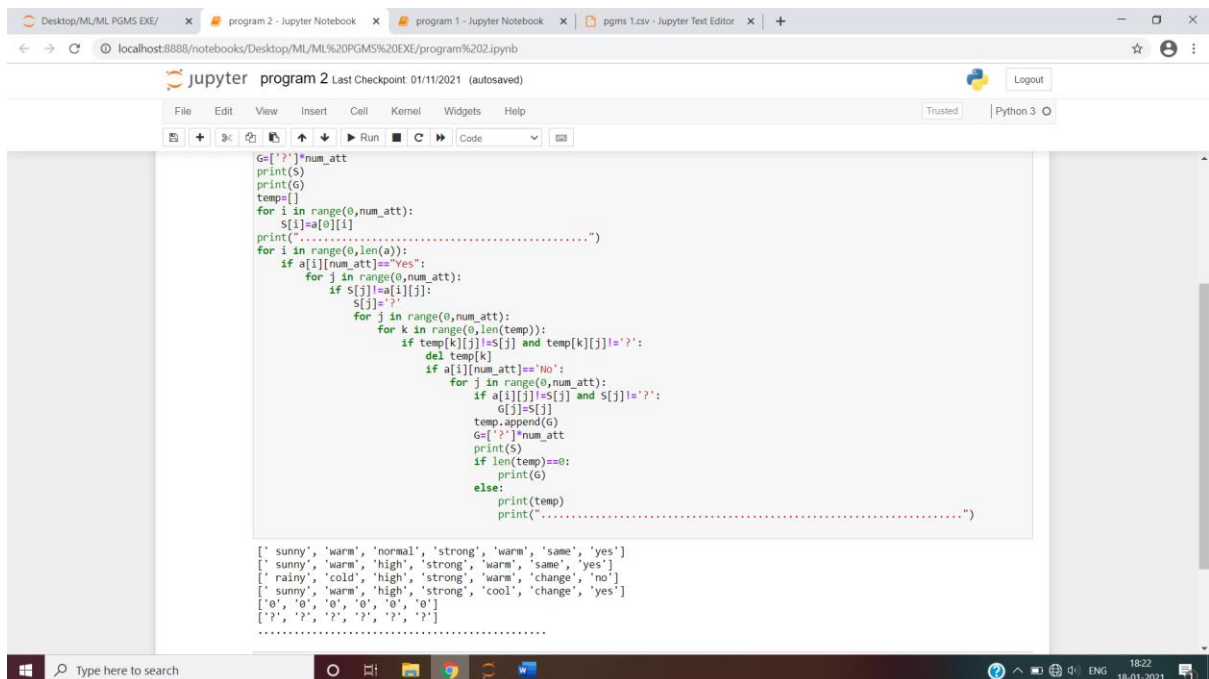
The initial hypothesis is :
['?', '?', '?', '?', '?', '?']

Find-S: Finding maximally specific Hypothesis

For training Example No:0 the hypothesis is [' sunny', 'warm', 'normal', 'strong', 'warm', 'same']
For training Example No:1 the hypothesis is [' sunny', 'warm', 'normal', 'strong', 'warm', 'same']
For training Example No:2 the hypothesis is [' sunny', 'warm', 'normal', 'strong', 'warm', 'change']
For training Example No:3 the hypothesis is [' sunny', 'warm', 'normal', 'strong', 'warm', 'change']

The Maximally specific hypothesis for the training instance is
```

Program2 output

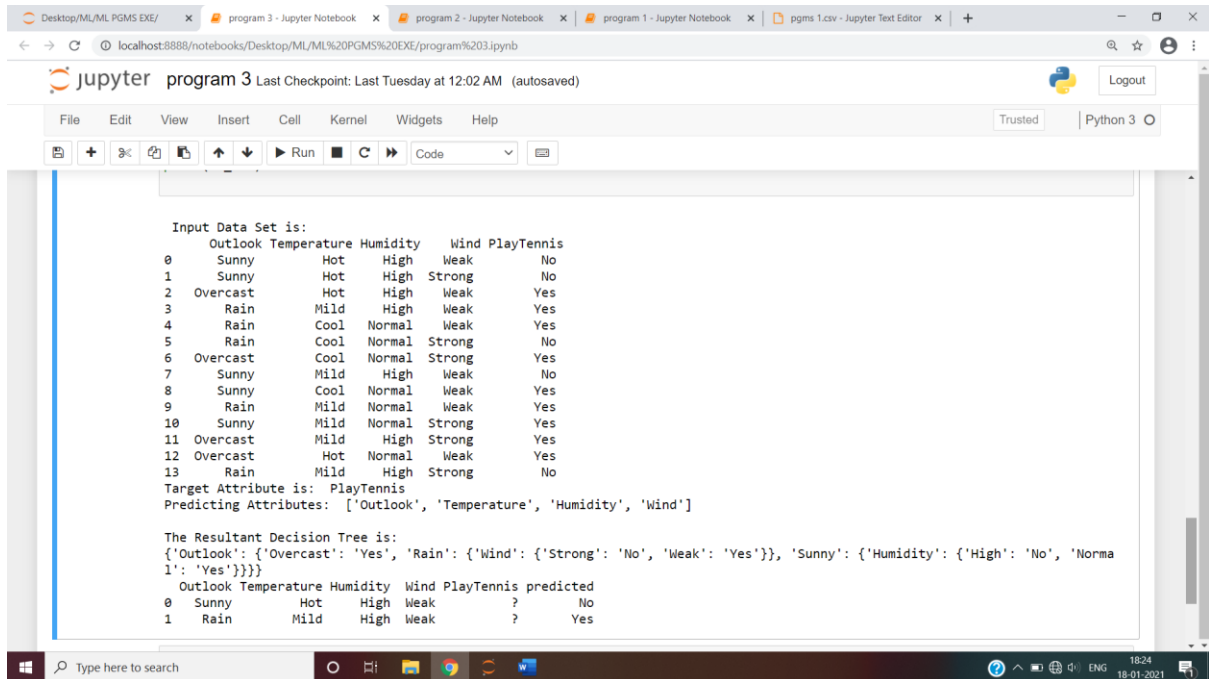


The screenshot shows a Jupyter Notebook interface with a single code cell. The code implements a function to find a maximally specific hypothesis from a set of training examples. The output of the code is displayed below the code cell.

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

[' 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']
['?', '?', '?', '?', '?', '?']
['?', '?', '?', '?', '?', '?']
.....
```

Program3 output

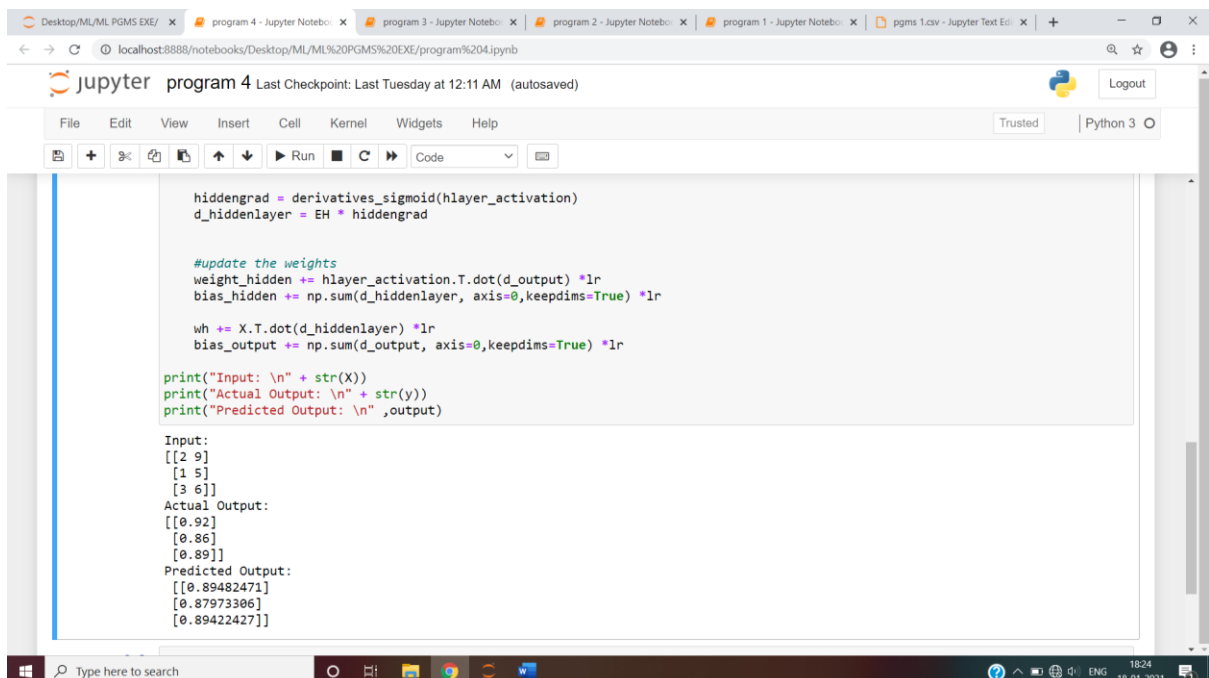


The screenshot shows a Jupyter Notebook interface with a single code cell. The output of the cell is a text-based decision tree for the 'PlayTennis' target attribute. The input data set is a 14x6 table with columns: Outlook, Temperature, Humidity, Wind, and PlayTennis. The decision tree logic is as follows: If Outlook is 'Overcast', the prediction is 'Yes'. If Outlook is 'Sunny', check Humidity: if 'High', prediction is 'No'; if 'Normal' or 'Mild', prediction is 'Yes'. If Outlook is 'Rain', check Wind: if 'Strong', prediction is 'No'; if 'Weak', prediction is 'Yes'. The resultant decision tree is summarized in a table below.

```
Input Data Set is:
  Outlook Temperature Humidity Wind PlayTennis
0 Sunny Hot High Weak No
1 Sunny Hot High Strong No
2 Overcast Hot High Weak Yes
3 Rain Mild High Weak Yes
4 Rain Cool Normal Weak Yes
5 Rain Cool Normal Strong No
6 Overcast Cool Normal Strong Yes
7 Sunny Mild High Weak No
8 Sunny Cool Normal Weak Yes
9 Rain Mild Normal Weak Yes
10 Sunny Mild Normal Strong Yes
11 Overcast Mild High Strong Yes
12 Overcast Hot Normal Weak Yes
13 Rain Mild High Strong No
Target Attribute is: PlayTennis
Predicting Attributes: ['Outlook', 'Temperature', 'Humidity', 'Wind']

The Resultant Decision Tree is:
{'Outlook': {'Overcast': 'Yes', 'Rain': {'Wind': {'Strong': 'No', 'Weak': 'Yes'}}, 'Sunny': {'Humidity': {'High': 'No', 'Normal': 'Yes'}}}}
  Outlook Temperature Humidity Wind PlayTennis predicted
0 Sunny Hot High Weak ? No
1 Rain Mild High Weak ? Yes
```

Program4 output



The screenshot shows a Jupyter Notebook interface with a single code cell. The output displays the calculations for a neural network layer, including hidden layer derivatives, weight updates, and final input/output values.

```
hiddengrad = derivatives_sigmoid(hlayer_activation)
d_hiddenlayer = EH * hiddengrad

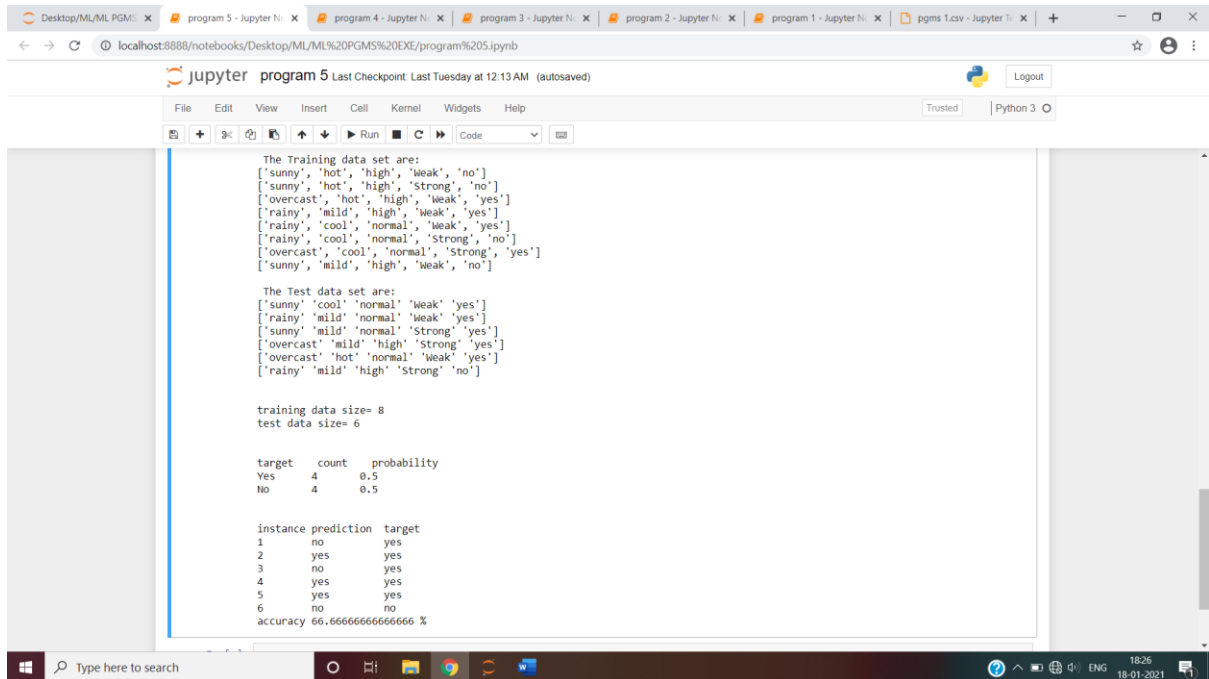
#update the weights
weight_hidden += hlayer_activation.T.dot(d_output) *lr
bias_hidden += np.sum(d_hiddenlayer, axis=0, keepdims=True) *lr

wh += X.T.dot(d_hiddenlayer) *lr
bias_output += np.sum(d_output, axis=0, keepdims=True) *lr

print("Input: \n" + str(X))
print("Actual Output: \n" + str(y))
print("Predicted Output: \n", output)

Input:
[[2 9]
 [1 5]
 [3 6]]
Actual Output:
[[0.92]
 [0.86]
 [0.89]]
Predicted Output:
[[0.89482471]
 [0.87973306]
 [0.89422427]]
```

Program5 output



The screenshot shows a Jupyter Notebook titled 'program 5' with the following output:

```
The Training data set are:
['sunny', 'hot', 'high', 'weak', 'no']
['sunny', 'hot', 'high', 'strong', 'no']
['overcast', 'hot', 'high', 'weak', 'yes']
['rainy', 'mild', 'high', 'weak', 'yes']
['rainy', 'cool', 'normal', 'weak', 'yes']
['rainy', 'cool', 'normal', 'strong', 'no']
['overcast', 'cool', 'normal', 'strong', 'yes']
['sunny', 'mild', 'high', 'weak', 'no']

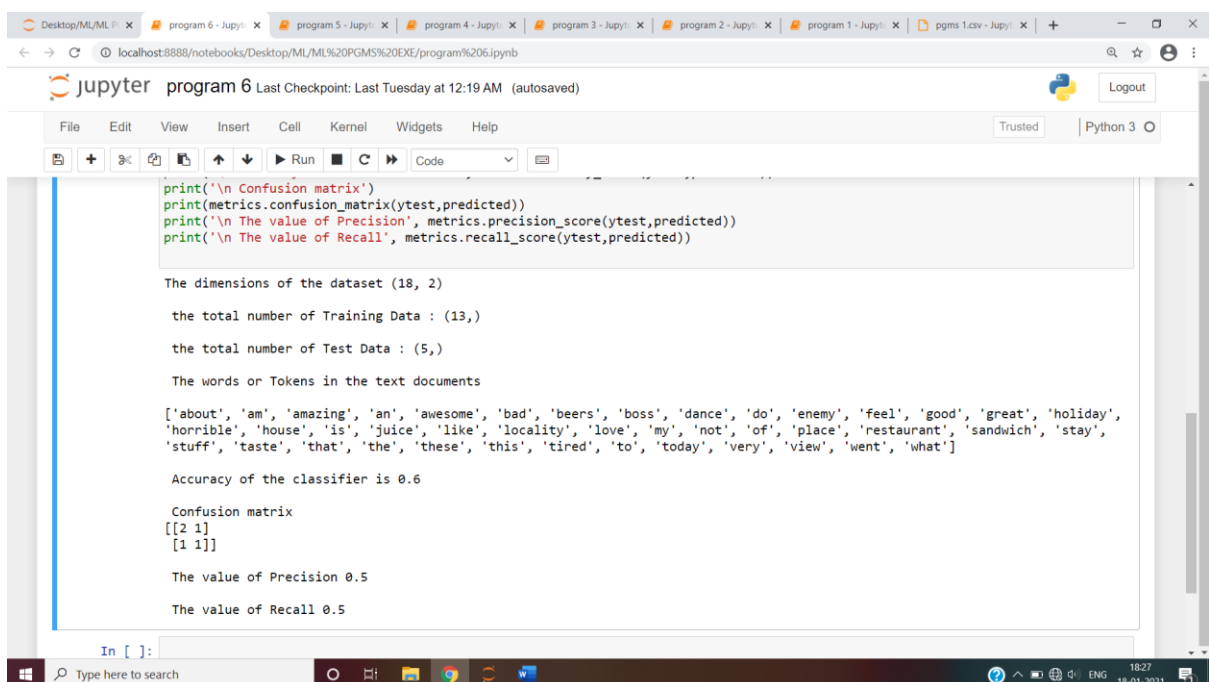
The Test data set are:
['sunny', 'cool', 'normal', 'weak', 'yes']
['rainy', 'mild', 'normal', 'weak', 'yes']
['sunny', 'mild', 'normal', 'strong', 'yes']
['overcast', 'mild', 'high', 'strong', 'yes']
['overcast', 'hot', 'normal', 'weak', 'yes']
['rainy', 'mild', 'high', 'strong', 'no']

training data size= 8
test data size= 6

target    count    probability
Yes       4         0.5
No        4         0.5

instance prediction target
1         no         yes
2         yes         yes
3         no         yes
4         yes         yes
5         yes         yes
6         no         no
accuracy 66.66666666666666 %
```

Program6 output



The screenshot shows a Jupyter Notebook titled 'program 6' with the following output:

```
print("\n Confusion matrix")
print(metrics.confusion_matrix(ytest,predicted))
print("\n The value of Precision", metrics.precision_score(ytest,predicted))
print("\n The value of Recall", metrics.recall_score(ytest,predicted))

The dimensions of the dataset (18, 2)

the total number of Training Data : (13,)

the total number of Test Data : (5,)

The words or Tokens in the text documents
['about', 'am', 'amazing', 'an', 'awesome', 'bad', 'beers', 'boss', 'dance', 'do', 'enemy', 'feel', 'good', 'great', 'holiday',
'horrible', 'house', 'is', 'juice', 'like', 'locality', 'love', 'my', 'not', 'of', 'place', 'restaurant', 'sandwich', 'stay',
'stuff', 'taste', 'that', 'the', 'these', 'this', 'tired', 'to', 'today', 'very', 'view', 'went', 'what']

Accuracy of the classifier is 0.6

Confusion matrix
[[2 1]
 [1 1]]

The value of Precision 0.5

The value of Recall 0.5
```

Program7 output

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	object
thal	object
heartdisease	int64
dtype:	object

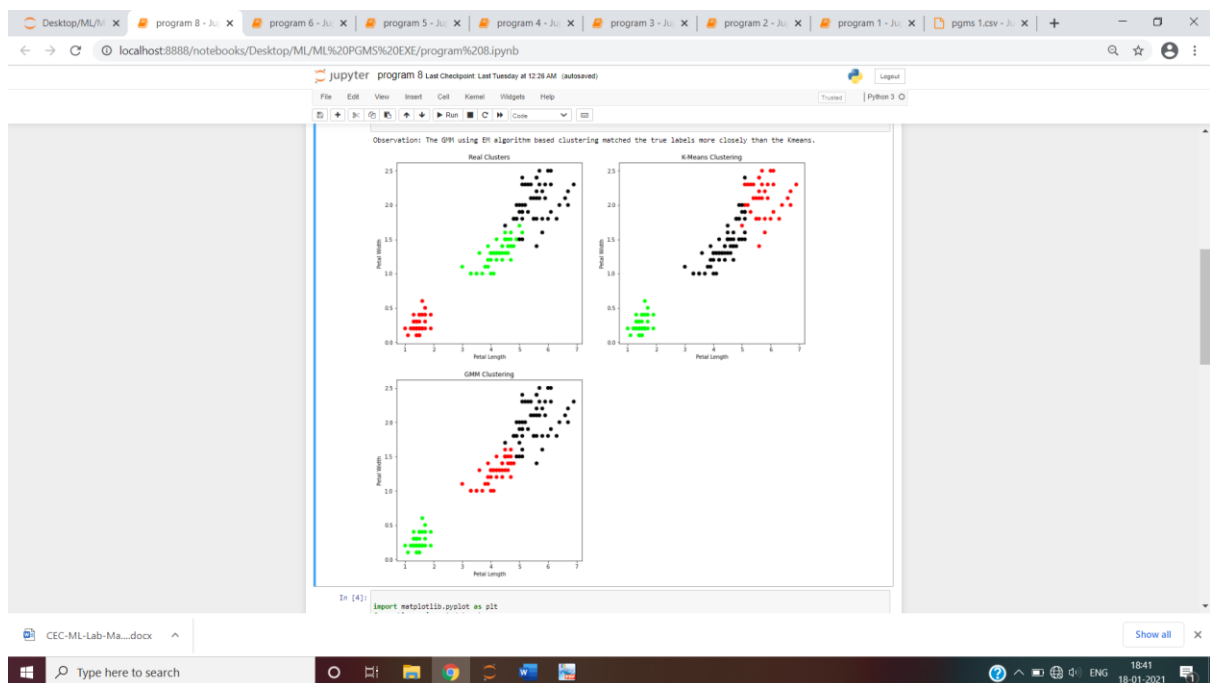
Inferencing with Bayesian Network:

1.Probability of HeartDisease given evidence=restecg :1

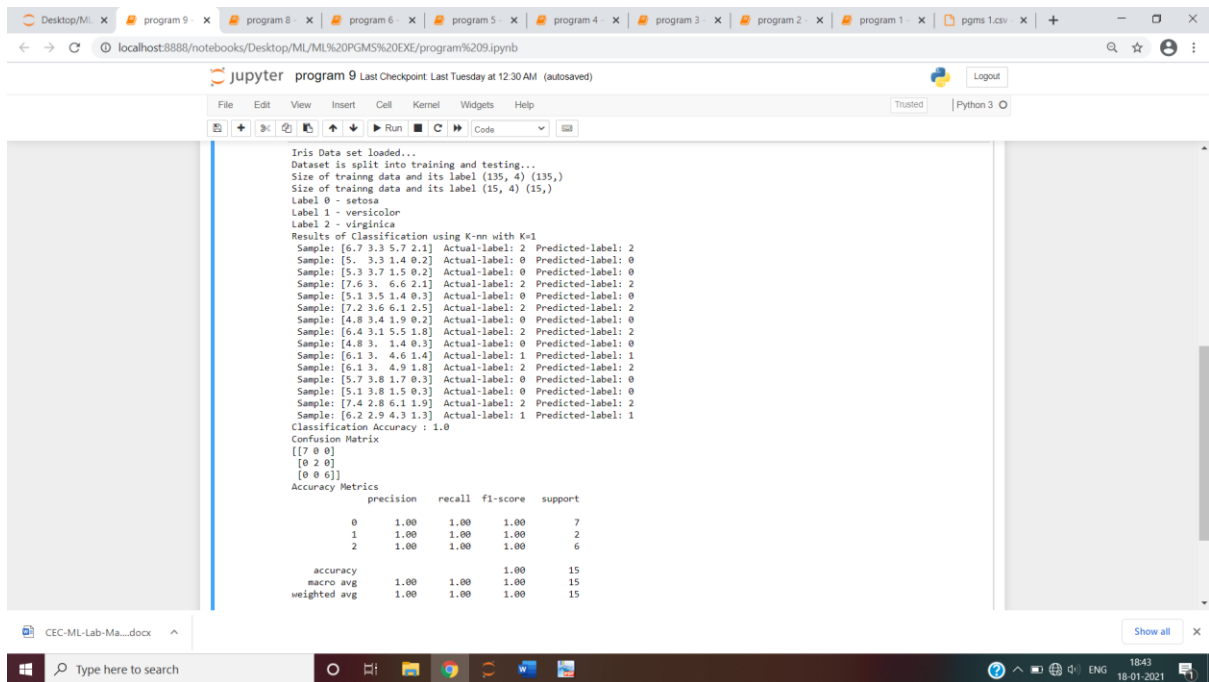
heartdisease	phi(heartdisease)
heartdisease(0)	0.1012
heartdisease(1)	0.0000
heartdisease(2)	0.2392
heartdisease(3)	0.2015
heartdisease(4)	0.4581

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

Program8 output



Program9 output



Program10 output

