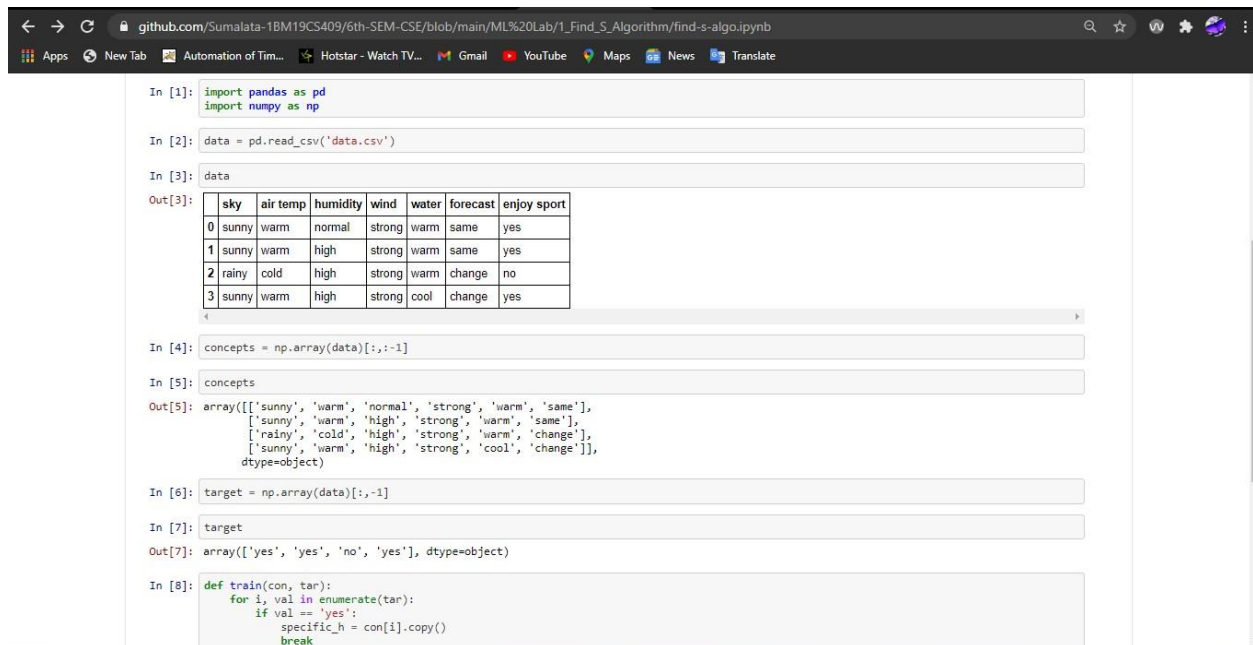


1. Fins – S algorithm

```
import pandas as pd
import numpy as np
data = pd.read_csv('data.csv')
data
concepts = np.array(data)[:,-1]
concepts
target = np.array(data)[:,-1]
target
def train(con, tar):
    for i, val in enumerate(tar):
        if val == 'yes':
            specific_h = con[i].copy()
            break

    for i, val in enumerate(con):
        if tar[i] == 'yes':
            for x in range(len(specific_h)):
                if val[x] != specific_h[x]:
                    specific_h[x] = '?'
            else:
                pass
    return specific_h
print(train(concepts, target))
```

Output



```
github.com/Sumalata-1BM19CS409/6th-SEM-CSE/blob/main/ML%20Lab/1_Find_S_Algorithm/find-s-algo.ipynb
In [1]: import pandas as pd
import numpy as np

In [2]: data = pd.read_csv('data.csv')

In [3]: data
Out[3]:
```

	sky	air temp	humidity	wind	water	forecast	enjoy sport
0	sunny	warm	normal	strong	warm	same	yes
1	sunny	warm	high	strong	warm	same	yes
2	rainy	cold	high	strong	warm	change	no
3	sunny	warm	high	strong	cool	change	yes

```
In [4]: concepts = np.array(data)[:,-1]

In [5]: concepts
Out[5]: array([[ 'sunny', 'warm', 'normal', 'strong', 'warm', 'same'],
 [ 'sunny', 'warm', 'high', 'strong', 'warm', 'same'],
 [ 'rainy', 'cold', 'high', 'strong', 'warm', 'change'],
 [ 'sunny', 'warm', 'high', 'strong', 'cool', 'change']],
 dtype=object)

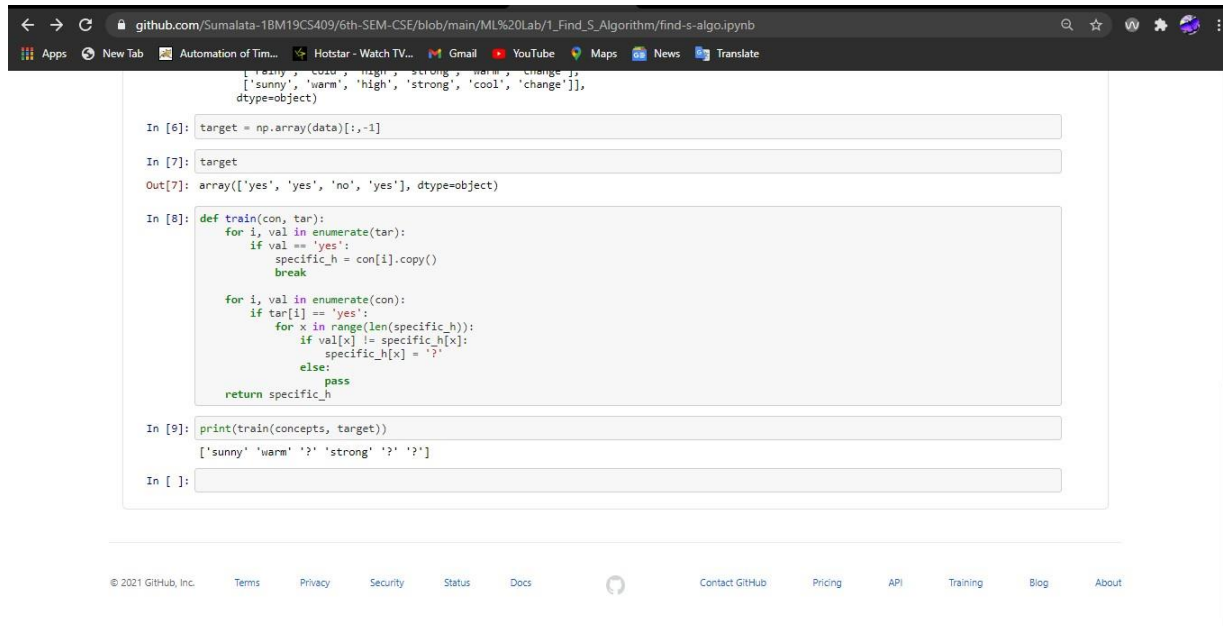
In [6]: target = np.array(data)[:,-1]

In [7]: target
Out[7]: array(['yes', 'yes', 'no', 'yes'], dtype=object)

In [8]: def train(con, tar):
        for i, val in enumerate(tar):
            if val == 'yes':
                specific_h = con[i].copy()
                break
```

Machine Learning Lab Report

Sowmya D U
1BM19CS408



The screenshot displays a Jupyter Notebook interface within a web browser. The browser's address bar shows the URL: `github.com/Sumalata-1BM19CS409/6th-SEM-CSE/blob/main/ML%20Lab/1_Find_S_Algorithm/find-s-algo.ipynb`. The notebook contains the following code and output:

```
[['sunny', 'cool', 'high', 'strong', 'warm', 'change'],  
 ['sunny', 'warm', 'high', 'strong', 'cool', 'change']],  
 dtype=object)  
  
In [6]: target = np.array(data)[1,-1]  
  
In [7]: target  
Out[7]: array(['yes', 'yes', 'no', 'yes'], dtype=object)  
  
In [8]: def train(con, tar):  
        for i, val in enumerate(tar):  
            if val == 'yes':  
                specific_h = con[i].copy()  
                break  
  
        for i, val in enumerate(con):  
            if tar[i] == 'yes':  
                for x in range(len(specific_h)):  
                    if val[x] != specific_h[x]:  
                        specific_h[x] = '?'  
                else:  
                    pass  
        return specific_h  
  
In [9]: print(train(concepts, target))  
['sunny' 'warm' '?' 'strong' '?' '?']  
  
In [ ]:
```

The footer of the browser window shows the GitHub logo and the text: © 2021 GitHub, Inc. Terms Privacy Security Status Docs Contact GitHub Pricing API Training Blog About.

2. Candidate elimination algorithm

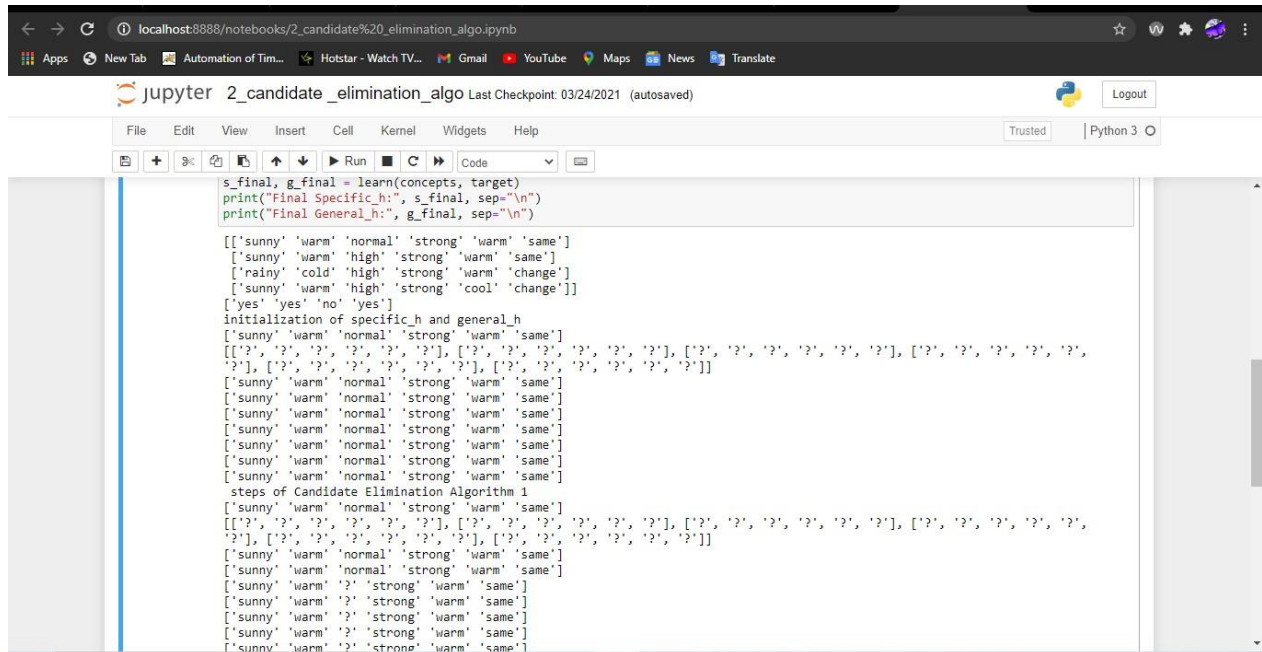
```
import numpy as np
import pandas as pd
data = pd.DataFrame(data=pd.read_csv('enjoysport.csv'))
concepts = np.array(data.iloc[:,0:-1])
print(concepts)
target = np.array(data.iloc[:,-1])
print(target)

def learn(concepts, target):
    specific_h = concepts[0].copy()
    print("initialization of specific_h and general_h")
    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
s_final, g_final = learn(concepts, target)
print("Final Specific_h:", s_final, sep="\n")
print("Final General_h:", g_final, sep="\n")
```

Machine Learning Lab Report

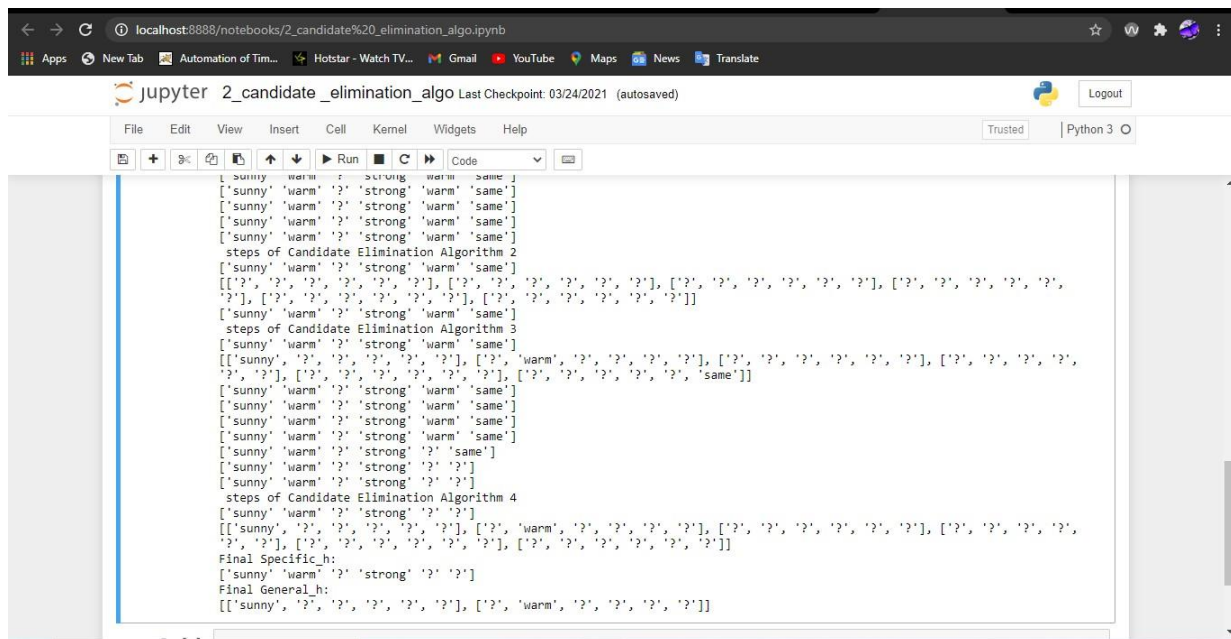
Sowmya D U
IBM19CS408

Output



```
s_final, g_final = learn(concepts, target)
print("Final Specific_h:", s_final, sep="\n")
print("Final General_h:", g_final, sep="\n")

[['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']
Initialization of specific_h and general_h
['sunny' 'warm' 'normal' 'strong' 'warm' 'same']
[['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?']]
['sunny' 'warm' 'normal' 'strong' 'warm' 'same']
['sunny' 'warm' 'normal' 'strong' 'warm' 'same']
['sunny' 'warm' 'normal' 'strong' 'warm' 'same']
['sunny' 'warm' 'normal' 'strong' 'warm' 'same']
['sunny' 'warm' 'normal' 'strong' 'warm' 'same']
['sunny' 'warm' 'normal' 'strong' 'warm' 'same']
steps of Candidate Elimination Algorithm 1
['sunny' 'warm' 'normal' 'strong' 'warm' 'same']
[['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?']]
['sunny' 'warm' 'normal' 'strong' 'warm' 'same']
['sunny' 'warm' 'normal' 'strong' 'warm' 'same']
['sunny' 'warm' '?' 'strong' 'warm' 'same']
['sunny' 'warm' '?' 'strong' 'warm' 'same']
['sunny' 'warm' '?' 'strong' 'warm' 'same']
['sunny' 'warm' '?' 'strong' 'warm' 'same']
['sunny' 'warm' '?' 'strong' 'warm' 'same']
```



```
['sunny' 'warm' '?' 'strong' 'warm' 'same']
['sunny' 'warm' '?' 'strong' 'warm' 'same']
['sunny' 'warm' '?' 'strong' 'warm' 'same']
['sunny' 'warm' '?' 'strong' 'warm' 'same']
steps of Candidate Elimination Algorithm 2
['sunny' 'warm' '?' 'strong' 'warm' 'same']
[['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?']]
['sunny' 'warm' '?' 'strong' 'warm' 'same']
steps of Candidate Elimination Algorithm 3
['sunny' 'warm' '?' 'strong' 'warm' 'same']
[['sunny', '?', '?', '?', '?', '?'], ['?', 'warm', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?']]
['sunny' 'warm' '?' 'strong' 'warm' 'same']
['sunny' 'warm' '?' 'strong' 'warm' 'same']
['sunny' 'warm' '?' 'strong' 'warm' 'same']
['sunny' 'warm' '?' 'strong' 'warm' 'same']
['sunny' 'warm' '?' 'strong' 'warm' 'same']
['sunny' 'warm' '?' 'strong' 'warm' 'same']
steps of Candidate Elimination Algorithm 4
['sunny' 'warm' '?' 'strong' 'warm' 'same']
[['sunny', '?', '?', '?', '?', '?'], ['?', 'warm', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?']]
Final Specific_h:
['sunny' 'warm' '?' 'strong' 'warm' 'same']
Final General_h:
[['sunny', '?', '?', '?', '?', '?'], ['?', 'warm', '?', '?', '?', '?']]
```

3. ID3 algorithm

```
import math
import csv
```

In [2]:

```
def load_csv(filename):
    lines = csv.reader(open(filename,"r"));
    dataset = list(lines)
    headers = dataset.pop(0)
    return dataset, headers
```

In [3]:

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

In [4]:

```
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
```

In [5]:

```
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)
```

Machine Learning Lab Report

Sowmya D U
1BM19CS408

```
sums = 0
for cnt in counts:
    sums += -1 * cnt * math.log(cnt, 2)
return sums
```

In [6]:

```
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
```

In [7]:

```
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
```

In [8]:

```
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)
```

In [9]:

Machine Learning Lab Report

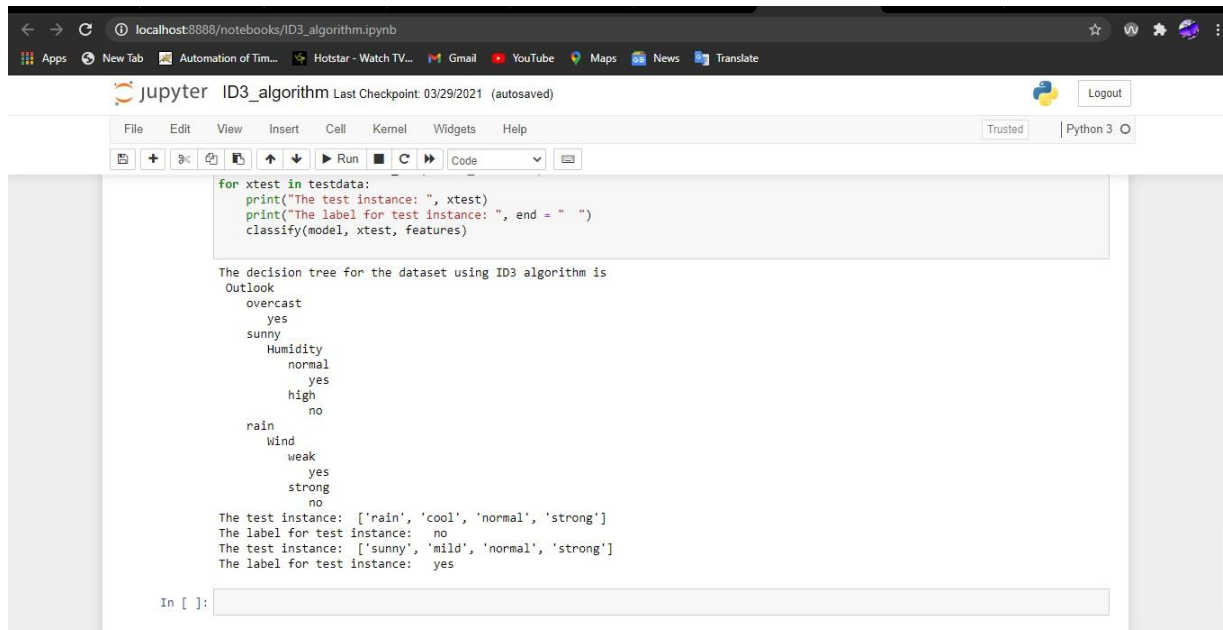
Sowmya D U
1BM19CS408

```
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)
```

In [10]:

```
"""Main Program"""  
dataset, features = load_csv("data3.csv")  
model = build_tree(dataset, features)  
  
print("The decision tree for the dataset using ID3 algorithm is")  
print_tree(model, 0)  
testdata, features = load_csv("data3_test.csv")  
for xtest in testdata:  
    print("The test instance: ", xtest)  
    print("The label for test instance: ", end = " ")  
    classify(model, xtest, features)
```

Output



```
for xtest in testdata:  
    print("The test instance: ", xtest)  
    print("The label for test instance: ", end = " ")  
    classify(model, xtest, features)
```

The decision tree for the dataset using ID3 algorithm is

```
Outlook  
  overcast  
  yes  
  sunny  
    Humidity  
    normal  
    yes  
    high  
    no  
  rain  
    wind  
    weak  
    yes  
    strong  
    no
```

The test instance: ['rain', 'cool', 'normal', 'strong']
The label for test instance: no
The test instance: ['sunny', 'mild', 'normal', 'strong']
The label for test instance: yes

4. Naïve Bayesian algorithm

```
import pandas as pd
from sklearn import tree
from sklearn.preprocessing import LabelEncoder
from sklearn.naive_bayes import GaussianNB
```

```
data = pd.read_csv('tennisdata.csv')
print("The first 5 values of data is :\n",data.head())
```

In [2]:

```
X = data.iloc[:, :-1]
print("\nThe First 5 values of train data is\n",X.head())
```

In [3]:

```
y = data.iloc[:, -1]
print("\nThe first 5 values of Train output is\n",y.head())
```

In [4]:

```
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())
```

In [5]:

```
le_PlayTennis = LabelEncoder()
y = le_PlayTennis.fit_transform(y)
print("\nNow the Train output is\n",y)
```

In [6]:

```
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))
```


Machine Learning Lab Report

Sowmya D U
1BM19CS408

Output

```
localhost:8888/notebooks/naive_bayesian.ipynb
jupyter naive_bayesian Last Checkpoint: Last Sunday at 6:42 AM (autosaved)
Python 3

In [1]: import pandas as pd
        from sklearn import tree
        from sklearn.preprocessing import LabelEncoder
        from sklearn.naive_bayes import GaussianNB

        data = pd.read_csv('tennisdata.csv')
        print("The first 5 values of data is :\n",data.head())

        The first 5 values of data is :
           Outlook Temperature Humidity Windy PlayTennis
0      Sunny      Hot      High  False      No
1      Sunny      Hot      High   True      No
2  Overcast      Hot      High  False      Yes
3     Rainy     Mild      High  False      Yes
4     Rainy     Cool     Normal  False      Yes

In [2]: X = data.iloc[:, :-1]
        print("\nThe First 5 values of train data is\n",X.head())

        The First 5 values of train data is
           Outlook Temperature Humidity Windy
0      Sunny      Hot      High  False
1      Sunny      Hot      High   True
2  Overcast      Hot      High  False
3     Rainy     Mild      High  False
4     Rainy     Cool     Normal  False
```

```
localhost:8888/notebooks/naive_bayesian.ipynb
jupyter naive_bayesian Last Checkpoint: Last Sunday at 6:42 AM (autosaved)
Python 3

In [3]: y = data.iloc[:, -1]
        print("\nThe first 5 values of Train output is\n",y.head())

        The first 5 values of Train output is
0      No
1      No
2      Yes
3      Yes
4      Yes
Name: PlayTennis, dtype: object

In [4]: 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)

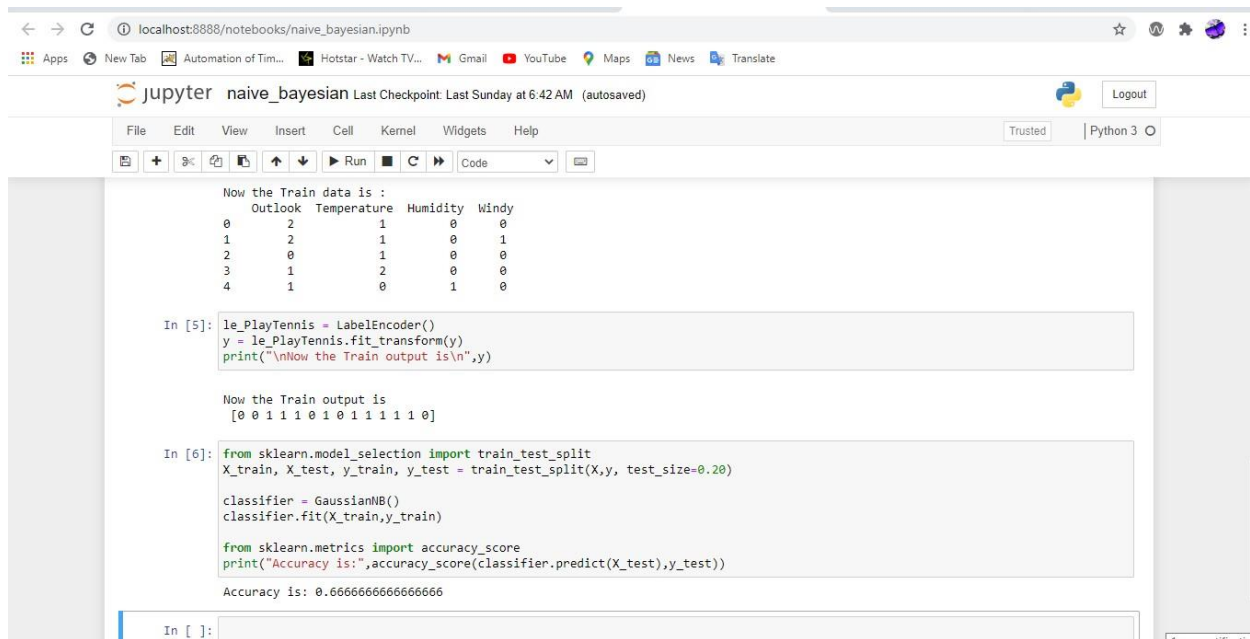
        le_Windy = LabelEncoder()
        X.Windy = le_Windy.fit_transform(X.Windy)

        print("\nNow the Train data is :\n",X.head())

        Now the Train data is :
           Outlook Temperature Humidity Windy
```

Machine Learning Lab Report

Sowmya D U
1BM19CS408



```
localhost:8888/notebooks/naive_bayesian.ipynb
jupyter naive_bayesian Last Checkpoint: Last Sunday at 6:42 AM (autosaved)
File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

Now the Train data is :
  Outlook  Temperature  Humidity  Windy
0         2           1         0       0
1         2           1         0       1
2         0           1         0       0
3         1           2         0       0
4         1           0         1       0

In [5]: le_PlayTennis = LabelEncoder()
        y = le_PlayTennis.fit_transform(y)
        print("\nNow the Train output is\n",y)

Now the Train output is
[0 0 1 1 1 0 1 0 1 1 1 1 1 0]

In [6]: 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))

Accuracy is: 0.6666666666666666

In [ ]:
```

5. Bayesian Network Classifier algorithm

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

In [3]:

```
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']
```

In [4]:

```
X = df[feature_col_names].values # these are factors for the prediction
y = df[predicted_class_names].values # this is what we want to predict
```

In [5]:

```
#splitting the dataset into train and test data
print(df.head)
xtrain,xtest,ytrain,ytest=train_test_split(X,y,test_size=0.33)
```

In [6]:

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

In [7]:

```
# Training Naive Bayes (NB) classifier on training data.
```

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

In [8]:

```
#printing Confusion matrix, accuracy, Precision and Recall
```

```
print('\n Confusion matrix')
print(metrics.confusion_matrix(ytest,predicted))
```

In [9]:

```
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)
```

Machine Learning Lab Report

Sowmya D U
1BM19CS408

Output

```
localhost:8888/notebooks/Bayesian_network.ipynb#
jupyter Bayesian_network Last Checkpoint: 7 minutes ago (autosaved)
Python 3

In [1]: import pandas as pd
        from sklearn.model_selection import train_test_split
        from sklearn.naive_bayes import GaussianNB
        from sklearn import metrics

In [3]: 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']

In [4]: X = df[feature_col_names].values # these are factors for the prediction
        y = df[predicted_class_names].values # this is what we want to predict

In [5]: #splitting the dataset into train and test data
        print(df.head)
        xtrain,xtest,ytrain,ytest=train_test_split(X,y,test_size=0.33)

<bound method NDFrame.head of          num_preg  glucose_conc  diastolic_bp  thickness  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           82           70         31         0  30.4
```

```
localhost:8888/notebooks/Bayesian_network.ipynb#
jupyter Bayesian_network Last Checkpoint: 8 minutes ago (autosaved)
Python 3

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]>

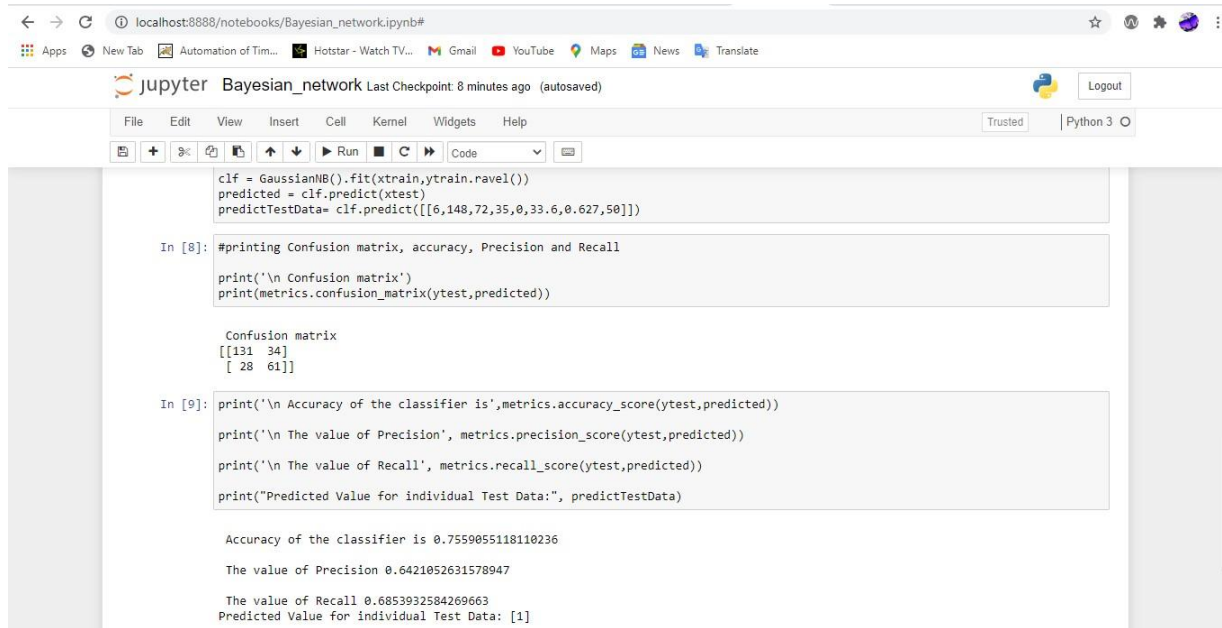
In [6]: print ('\n the total number of Training Data :',ytrain.shape)
        print ('\n the total number of Test Data :',ytest.shape)

        the total number of Training Data : (514, 1)
        the total number of Test Data : (254, 1)

In [7]: # Training Naive Bayes (NB) classifier on training data.
        clf = GaussianNB().fit(xtrain,ytrain.ravel())
        predicted = clf.predict(xtest)
        predictTestData= clf.predict([[6,148,72,35,0,33.6,0.627,50]])
```

Machine Learning Lab Report

Sowmya D U
1BM19CS408



The screenshot displays a Jupyter Notebook titled "Bayesian_network" running on a local host. The interface includes a standard web browser at the top with the address bar showing "localhost:8888/notebooks/Bayesian_network.ipynb#". Below the browser, the Jupyter logo and title are visible, along with a "Logout" button. The notebook's menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help) and a toolbar with icons for file operations and execution are present. The main area contains two code cells. The first cell defines a Gaussian Naive Bayes classifier, fits it to training data, and predicts on test data. The second cell prints the confusion matrix, accuracy, precision, and recall scores, and displays the predicted value for a specific test data point. The output of the second cell shows a confusion matrix, accuracy of approximately 0.755, precision of approximately 0.642, recall of approximately 0.685, and a predicted value of [1].

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

In [8]: #printing Confusion matrix, accuracy, Precision and Recall
print('\n Confusion matrix')
print(metrics.confusion_matrix(ytest,predicted))

Confusion matrix
[[131  34]
 [ 28  61]]

In [9]: 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)

Accuracy of the classifier is 0.7559055118110236
The value of Precision 0.6421052631578947
The value of Recall 0.6853932584269663
Predicted Value for individual Test Data: [1]
```

6. Bayesian Network using cancer dataset

```
from pgmpy.models import BayesianModel
from pgmpy.factors.discrete import TabularCPD
from pgmpy.inference import VariableElimination

cancer_model=BayesianModel([('Pollution','Cancer'),('Smoker','Cancer'),('Cancer','Xray'),('Cancer','Dyspnoea')
])
print('Bayesian network models are :')
print('\t',cancer_model.nodes())
print('Bayesian edges are:')
print('\t',cancer_model.edges())

cpd_poll = TabularCPD(variable='Pollution', variable_card=2,
                      values=[[0.9], [0.1]])
cpd_smoke = TabularCPD(variable='Smoker', variable_card=2,
                      values=[[0.3], [0.7]])
cpd_cancer = TabularCPD(variable='Cancer', variable_card=2,
                      values=[[0.03, 0.05, 0.001, 0.02],
                             [0.97, 0.95, 0.999, 0.98]],
                      evidence=['Smoker', 'Pollution'],
                      evidence_card=[2, 2])
cpd_xray = TabularCPD(variable='Xray', variable_card=2,
                      values=[[0.9, 0.2], [0.1, 0.8]],
                      evidence=['Cancer'], evidence_card=[2])
cpd_dysp = TabularCPD(variable='Dyspnoea', variable_card=2,
                      values=[[0.65, 0.3], [0.35, 0.7]],
                      evidence=['Cancer'], evidence_card=[2])

# Associating the parameters with the model structure.
cancer_model.add_cpds(cpd_poll, cpd_smoke, cpd_cancer, cpd_xray, cpd_dysp)

# Checking if the cpds are valid for the model.
cancer_model.check_model()

cancer_infer=VariableElimination(cancer_model)

print('All local independecies are as follows')
cancer_model.get_independencies()
print('Displaying CPDs')
print(cancer_model.get_cpds('Pollution'))
```

Machine Learning Lab Report

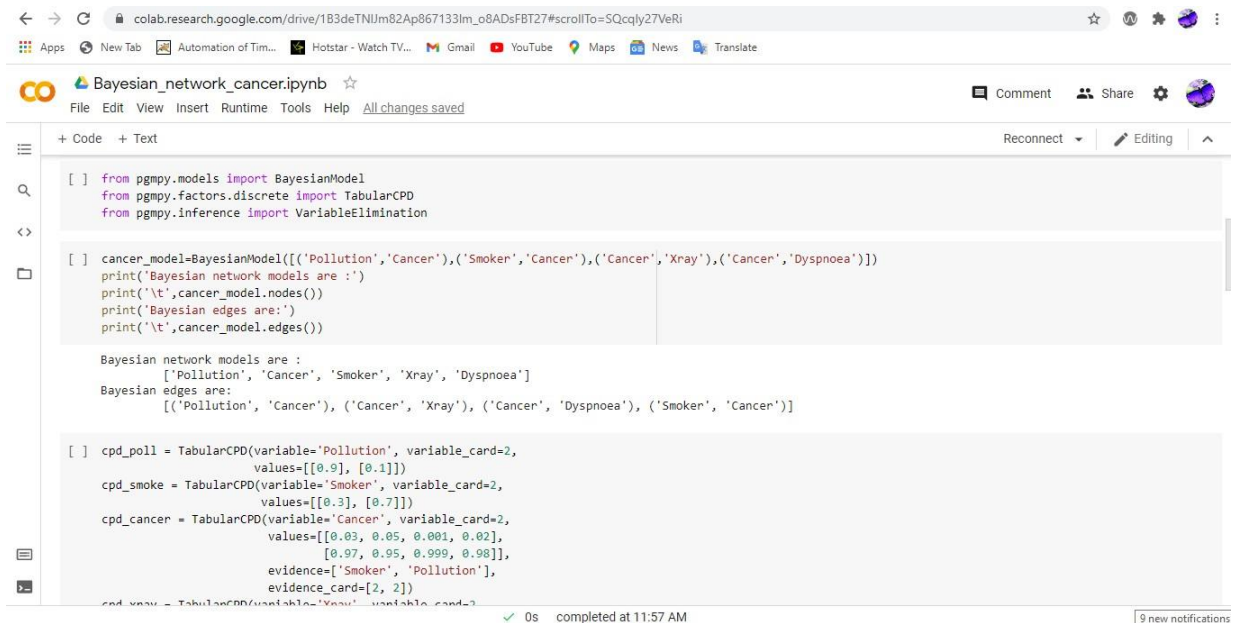
Sowmya D U
1BM19CS408

```
print(cancer_model.get_cpds('Smoker'))
print(cancer_model.get_cpds('Cancer'))
print(cancer_model.get_cpds('Xray'))
print(cancer_model.get_cpds('Dyspnoea'))

print("\n Probablity of Cancer given smoker')
q=cancer_infer.query(variables=['Cancer'],evidence={'Smoker':1})
print(q)

print("\n Probablity of Cancer given smoker, pollution')
q=cancer_infer.query(variables=['Cancer'],evidence={'Smoker':1,'Pollution':1})
print(q)
```

Output



The screenshot shows a Jupyter Notebook titled 'Bayesian_network_cancer.ipynb' in a Google Colab environment. The code defines a Bayesian network model with four nodes: 'Pollution', 'Cancer', 'Smoker', and 'Dyspnoea'. It then prints the nodes and edges of the model. The edges are: ('Pollution', 'Cancer'), ('Cancer', 'Xray'), ('Cancer', 'Dyspnoea'), and ('Smoker', 'Cancer'). The notebook also defines three conditional probability distributions (CPDs) for the nodes: 'Pollution' (values [0.9, 0.1]), 'Smoker' (values [0.3, 0.7]), and 'Cancer' (values [0.03, 0.05, 0.001, 0.02, 0.97, 0.95, 0.999, 0.98]). The notebook is running, and the output shows the model structure and the CPDs.

```
[ ] from pgmpy.models import BayesianModel
    from pgmpy.factors.discrete import TabularCPD
    from pgmpy.inference import VariableElimination

[ ] cancer_model=BayesianModel([('Pollution','Cancer'),('Smoker','Cancer'),('Cancer','Xray'),('Cancer','Dyspnoea')])
    print("Bayesian network models are :")
    print("\t",cancer_model.nodes())
    print("Bayesian edges are:")
    print("\t",cancer_model.edges())

Bayesian network models are :
['Pollution', 'Cancer', 'Smoker', 'Xray', 'Dyspnoea']
Bayesian edges are:
[('Pollution', 'Cancer'), ('Cancer', 'Xray'), ('Cancer', 'Dyspnoea'), ('Smoker', 'Cancer')]

[ ] cpd_poll = TabularCPD(variable='Pollution', variable_card=2,
    values=[[0.9], [0.1]])
    cpd_smoke = TabularCPD(variable='Smoker', variable_card=2,
    values=[[0.3], [0.7]])
    cpd_cancer = TabularCPD(variable='Cancer', variable_card=2,
    values=[[0.03, 0.05, 0.001, 0.02],
    [0.97, 0.95, 0.999, 0.98]],
    evidence=['Smoker', 'Pollution'],
    evidence_card=[2, 2])
    cpd_xray = TabularCPD(variable='Xray', variable_card=2,
```

Machine Learning Lab Report

Sowmya D U
IBM19CS408

```
colab.research.google.com/drive/1B3deTNUm82Ap867133lm_o8ADsFBT27#scrollTo=SQcqly27VeRi
Apps New Tab Automation of Tim... Hotstar - Watch TV... Gmail YouTube Maps News Translate
Bayesian_network_cancer.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Reconnect Editing
+ Code + Text
[ ] evidence_card=[2, 2])
cpd_xray = TabularCPD(variable='Xray', variable_card=2,
                      values=[[0.9, 0.2], [0.1, 0.8]],
                      evidence=['Cancer'], evidence_card=[2])
cpd_dysp = TabularCPD(variable='Dyspnoea', variable_card=2,
                      values=[[0.65, 0.3], [0.35, 0.7]],
                      evidence=['Cancer'], evidence_card=[2])

[ ] # Associating the parameters with the model structure.
cancer_model.add_cpds(cpd_poll, cpd_smoke, cpd_cancer, cpd_xray, cpd_dysp)

# Checking if the cpds are valid for the model.
cancer_model.check_model()

True

[ ] cancer_infer=VariableElimination(cancer_model)

[ ] print('All local independencies are as follows')
cancer_model.get_independencies()
print('Displaying CPDs')
print(cancer_model.get_cpds('Pollution'))
print(cancer_model.get_cpds('Smoker'))
print(cancer_model.get_cpds('Cancer'))
print(cancer_model.get_cpds('Xray'))

0s completed at 11:57 AM
```

```
colab.research.google.com/drive/1B3deTNUm82Ap867133lm_o8ADsFBT27#scrollTo=SQcqly27VeRi
Apps New Tab Automation of Tim... Hotstar - Watch TV... Gmail YouTube Maps News Translate
Bayesian_network_cancer.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Reconnect Editing
+ Code + Text
[ ] print(cancer_model.get_cpds('Smoker'))
print(cancer_model.get_cpds('Cancer'))
print(cancer_model.get_cpds('Xray'))
print(cancer_model.get_cpds('Dyspnoea'))

All local independencies are as follows
Displaying CPDs
+-----+
| Pollution(0) | 0.9 |
+-----+
| Pollution(1) | 0.1 |
+-----+
+-----+
| Smoker(0) | 0.3 |
+-----+
| Smoker(1) | 0.7 |
+-----+
+-----+
| Smoker | Smoker(0) | Smoker(0) | Smoker(1) | Smoker(1) |
+-----+
| Pollution | Pollution(0) | Pollution(1) | Pollution(0) | Pollution(1) |
+-----+
| Cancer(0) | 0.03 | 0.05 | 0.001 | 0.02 |
+-----+
| Cancer(1) | 0.97 | 0.95 | 0.999 | 0.98 |
+-----+
+-----+
| Cancer | Cancer(0) | Cancer(1) |
+-----+
```


Machine Learning Lab Report

Sowmya D U
1BM19CS408

The image displays two screenshots of a Jupyter Notebook interface, likely Google Colab, showing the execution of a Bayesian network inference script. The notebook is titled "Bayesian_network_cancer.ipynb".

Top Screenshot:

- The code cell shows a query for the probability of Cancer given Smoker status, with evidence set to 'Smoker':1.
- The output displays a table of probabilities for different pollution levels (0 and 1) and cancer status (0 and 1).
- The table structure is as follows:

	Pollution	Pollution(0)	Pollution(1)	Pollution(0)	Pollution(1)
Cancer(0)	0.03	0.05	0.001	0.02	
Cancer(1)	0.97	0.95	0.999	0.98	
- Below this, another table shows probabilities for Xray status (0 and 1) and cancer status (0 and 1).

	Cancer	Cancer(0)	Cancer(1)
Xray(0)	0.9	0.2	
Xray(1)	0.1	0.8	
- A third table shows probabilities for Dyspnoea status (0 and 1) and cancer status (0 and 1).

	Cancer	Cancer(0)	Cancer(1)
Dyspnoea(0)	0.65	0.3	
Dyspnoea(1)	0.35	0.7	
- The final output shows the probability of Cancer given smoker status, calculated using the query.
- The execution time is 0s, completed at 11:57 AM.

Bottom Screenshot:

- The code cell shows a query for the probability of Cancer given Smoker status, with evidence set to 'Smoker':1 and 'Pollution':1.
- The output displays a table of probabilities for different pollution levels (0 and 1) and cancer status (0 and 1).
- The table structure is as follows:

	Cancer	phi(Cancer)
Cancer(0)	0.0029	
Cancer(1)	0.9971	
- Below this, another table shows probabilities for Xray status (0 and 1) and cancer status (0 and 1).

	Cancer	phi(Cancer)
Xray(0)	0.0200	
Xray(1)	0.9800	
- The final output shows the probability of Cancer given smoker status, calculated using the query.
- The execution time is 0s, completed at 11:57 AM.