EX. NO: 2

DATE :

IMPLEMENTATION OF DECISION TREE IN ID3 ALGORITHM

AIM:

To build Decision tree in ID3 algorithm to classify a new sample using python.

ALGORITHM:

1. Observe the dataset. Import the necessary basic python libraries.
2. Read the dataset.
3. Calculate the Entropy of the whole dataset.
4. Calculate the Entropy of the filtered dataset.
5. Calculate the Information gain for the feature(outlook).
6. Finding the most informative feature (feature with highest information gain).
7. Adding a node to the tree.
8. Perform ID3 algorithm and generate a tree.
9. Finding unique classes of the label.
10. Predicting from the tree.
11. Evaluating the test dataset.
12. Checking the test dataset.

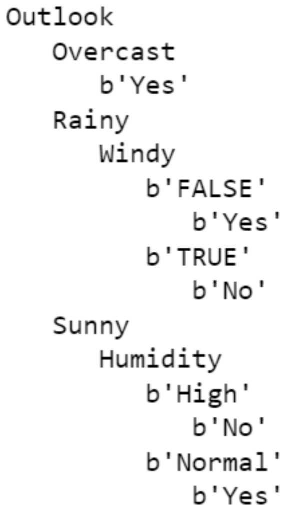
PROGRAM:

```
import numpy as np

import math

import csv

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)
class Node:
    def__init_(self, attribute):
        self.attribute = attribute
        self.children = []
        self.answer = ""

    def__str_(self):
        return self.attribute
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
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
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
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
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)
metadata, traindata = read_data("E:\BALA\AI\Lab programs\pgms\Tennisdata.csv")
data = np.array(traindata)
```

node = create_node(data, metadata)

print_tree(node, 0)

OUTPUT:



```
Outlook
   Overcast
      b'Yes'
   Rainy
      Windy
         b'FALSE'
            b'Yes'
         b'TRUE'
            b'No'
   Sunny
      Humidity
         b'High'
            b'No'
         b'Normal'
            b'Yes'
```

RESULT:

Thus the program to implement decision tree based ID3 algorithm using python was executed and verified successfully.

EX. NO: 3

DATE :

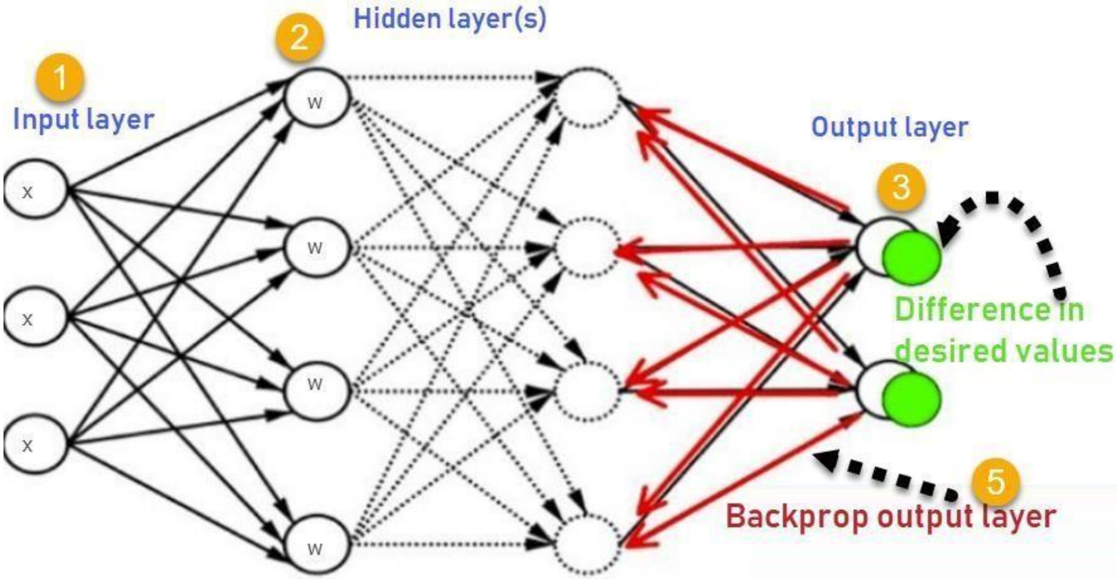
IMPLEMENTATION OF BACK PROPAGATION ALGORITHM TO BUILD AN ARTIFICIAL NEURAL NETWORK

AIM:

To implement the Back Propagation algorithm to build an Artificial Neural Network.

ALGORITHM:

1. Inputs X, arrive through the preconnected path.
2. Input is modeled using real weights W. The weights are usually randomly selected.
3. Calculate the output for every neuron from the input layer, to the hidden layers, to the output layer.
4. Calculate the error in the outputs
5. Travel back from the output layer to the hidden layer to adjust the weights such that the errors is decreased. Keep repeating the process until the desired output is achieved.



PROGRAM:

```
from math import exp

from random import seed

from random import random

# Initialize a network

def initialize_network(n_inputs, n_hidden, n_outputs):
    network = list()
    hidden_layer = [{'weights':[random() for i in range(n_inputs + 1)]} for i in range(n_hidden)]
    network.append(hidden_layer)
    output_layer = [{'weights':[random() for i in range(n_hidden + 1)]} for i in range(n_outputs)]
    network.append(output_layer)
    return network
# Calculate neuron activation for an input
def activate(weights, inputs):
    activation = weights[-1]
    for i in range(len(weights)-1):
        activation += weights[i] * inputs[i]
    return activation
# Transfer neuron activation
def transfer(activation):
    return 1.0 / (1.0 + exp(-activation))
# Forward propagate input to a network output
def forward_propagate(network, row):
    inputs = row
    for layer in network:
        new_inputs =  []
        for neuron in layer:
            activation = activate(neuron['weights'], inputs)
            neuron['output'] = transfer(activation)
            new_inputs.append(neuron['output'])
        inputs = new_inputs
    return inputs
# Calculate the derivative of an neuron output
```

```
def transfer_derivative(output):
    return output * (1.0 - output)
# Backpropagate error and store in neurons
def backward_propagate_error(network, expected):
    for i in reversed(range(len(network))):
        layer = network[i]
        errors = list()
        if i != len(network)-1:
            for j in range(len(layer)):
                error = 0.0
                for neuron in network[i + 1]:
                    error += (neuron['weights'][j] * neuron['delta'])
                errors.append(error)
        else:
            for j in range(len(layer)):
                neuron = layer[j]
                errors.append(neuron['output'] - expected[j])
        for j in range(len(layer)):
            neuron = layer[j]
            neuron['delta'] = errors[j] * transfer_derivative(neuron['output'])
# Update network weights with error
def update_weights(network, row, l_rate):
    for i in range(len(network)):
        inputs = row[:-1]
        if i != 0:
            inputs = [neuron['output'] for neuron in network[i - 1]]
        for neuron in network[i]:
            for j in range(len(inputs)):
                neuron['weights'][j] -= l_rate * neuron['delta'] * inputs[j]
            neuron['weights'][-1] -= l_rate * neuron['delta']
# Train a network for a fixed number of epochs
def train_network(network, train, l_rate, n_epoch, n_outputs):
    for epoch in range(n_epoch):
        sum_error = 0
        for row in train:
            outputs = forward_propagate(network, row)
            expected = [0 for i in range(n_outputs)]
            expected[row[-1]] = 1
            sum_error += sum([(expected[i]-outputs[i])**2 for i in range(len(expected))])
            backward_propagate_error(network, expected)
            update_weights(network, row, l_rate)
        print('>epoch=%d, lrate=%.3f, error=%.3f' % (epoch, l_rate, sum_error))
# Test training backprop algorithm
seed(1)
dataset = [[2.7810836,2.550537003,0],
    [1.465489372,2.362125076,0],
    [3.396561688,4.400293529,0],
    [1.38807019,1.850220317,0],
    [3.06407232,3.005305973,0],
    [7.627531214,2.759262235,1],
    [5.332441248,2.088626775,1],
    [6.922596716,1.77106367,1],
    [8.675418651,-0.242068655,1],
    [7.673756466,3.508563011,1]]
n_inputs = len(dataset[0]) - 1
n_outputs = len(set([row[-1] for row in dataset]))
```

```
network = initialize_network(n_inputs, 2, n_outputs)
train_network(network, dataset, 0.5, 20, n_outputs)
for layer in network:
        print(layer)
```

OUTPUT:

```
>epoch=0, lrate=0.500, error=6.350
>epoch=1, lrate=0.500, error=5.531
>epoch=2, lrate=0.500, error=5.221
>epoch=3, lrate=0.500, error=4.951
>epoch=4, lrate=0.500, error=4.519
>epoch=5, lrate=0.500, error=4.173
>epoch=6, lrate=0.500, error=3.835
>epoch=7, lrate=0.500, error=3.506
>epoch=8, lrate=0.500, error=3.192
>epoch=9, lrate=0.500, error=2.898
>epoch=10, lrate=0.500, error=2.626
>epoch=11, lrate=0.500, error=2.377
>epoch=12, lrate=0.500, error=2.153
>epoch=13, lrate=0.500, error=1.953
>epoch=14, lrate=0.500, error=1.774
>epoch=15, lrate=0.500, error=1.614
>epoch=16, lrate=0.500, error=1.472
>epoch=17, lrate=0.500, error=1.346
>epoch=18, lrate=0.500, error=1.233
>epoch=19, lrate=0.500, error=1.132
[{'weights': [-1.4688375095432327, 1.850887325439514, 1.0858178629550297], 'output': 0.029
980305604426185, 'delta': 0.0059546604162323625}, {'weights': [0.37711098142462157, -0.06
25909894552989, 0.2765123702642716], 'output': 0.9456229000211323, 'delta': -0.0026279652
850863837}]
[{'weights': [2.515394649397849, -0.3391927502445985, -0.9671565426390275], 'output': 0.23
648794202357587, 'delta': 0.04270059278364587}, {'weights': [-2.5584149848484263, 1.00364
22106209202, 0.42383086467582715], 'output': 0.7790535202438367, 'delta': -0.038031325964
37354}]
```

RESULT:

Thus the Back propagation algorithm to build an Artificial Neural networks was implemented successfully.

EX.NO: 4

DATE:

IMPLEMENTATION OF NAÏVE BAYESIAN CLASSIFIER FOR A SAMPLE TRAINING DATASET AND TO COMPUTE ACCURACY

AIM:

To implement Naïve Bayesian classifier for Tennis data set and to compute the accuracy with few datasets.

ALGORITHM:

1. Convert the data set into a frequency table.
2. Create likelihood table by finding the probabilities like overcast probability = 0.29 and probability of plating is 0.64.
3. Now, use Naive Bayesian equation to calculate the posterior probability for each class. The class with the highest posterior probability is the outcome of prediction.

   Problem: Players will play if weather is sunny. Is this statement is correct?

   We can solve it using above discussed method of posterior probability.

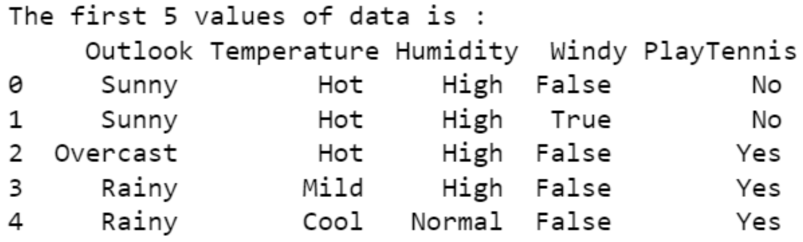
   P(Yes | Sunny) = P(Sunny | Yes) * P(Yes) / P (Sunny)

   Here we have P (Sunny |Yes) = 3/9 = 0.33, P(Sunny) = 5/14 = 0.36, P(Yes)= 9/14 = 0.64

   Now, P (Yes | Sunny) = 0.33 * 0.64 / 0.36 = 0.60, which has higher probability.
4. Exit.

PROGRAM:

```
import pandas as pd
from sklearn import tree
from sklearn.preprocessing import LabelEncoder
from sklearn.naive_bayes import GaussianNB
data = pd.read_csv("E:\BALA\AI\Lab programs\pgms\Tennis.csv")
print("The first 5 values of data is :\n",data.head())
```

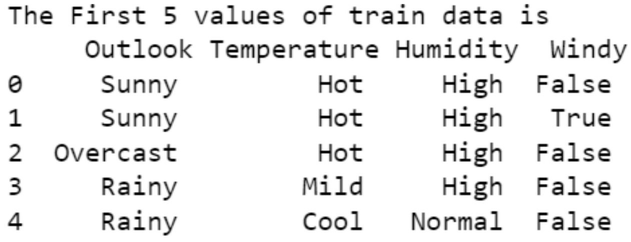


```
The first 5 values of data is :
      Outlook Temperature Humidity  Windy PlayTennis
0      Sunny         Hot     High  False         No
1      Sunny         Hot     High   True         No
2   Overcast         Hot     High  False        Yes
3      Rainy        Mild     High  False        Yes
4  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

# obtain Train data and Train output

X = data.iloc[:,:-1]

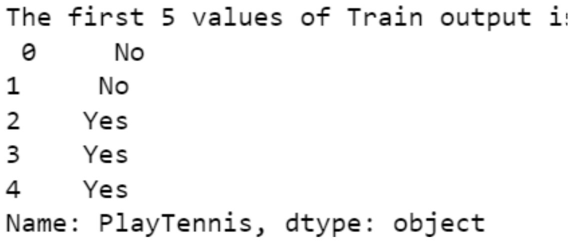
print("\nThe First 5 values of train data is\n",X.head())



```
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
```

y = data.iloc[:,-1]

print("\nThe first 5 values of Train output is\n",y.head())



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

# 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)

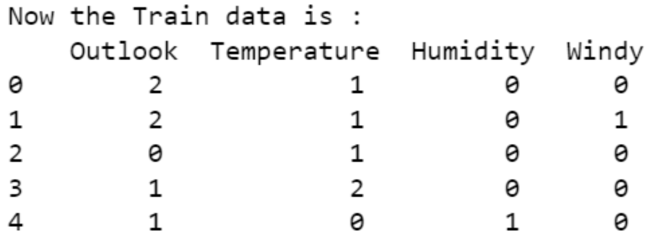
le_Humidity = LabelEncoder()

X.Humidity = le_Humidity.fit_transform(X.Humidity)

le_Windy = LabelEncoder()

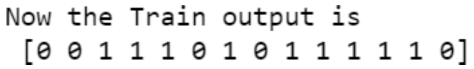
X.Windy = le_Windy.fit_transform(X.Windy)

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



```
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
```

```
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]
```

```
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))
```

OUTPUT:

```
Accuracy is: 0.6666666666666666
```

RESULT:

Thus the program to implement Naïve Bayesian classifier to compute the accuracy with few datasets using python was executed and verified successfully.

EX. NO: 5

DATE :

IMPLEMENTATION OF NAÏVE BAYESIAN CLASSIFIER MODEL TO CLASSIFY A SET OF DOCUMENTS AND TO MEASURE THE ACCURACY, PRECISION, AND RECALL

AIM:

To classify a set of documents using Naïve Bayesian classifier and to measure the accuracy and precision

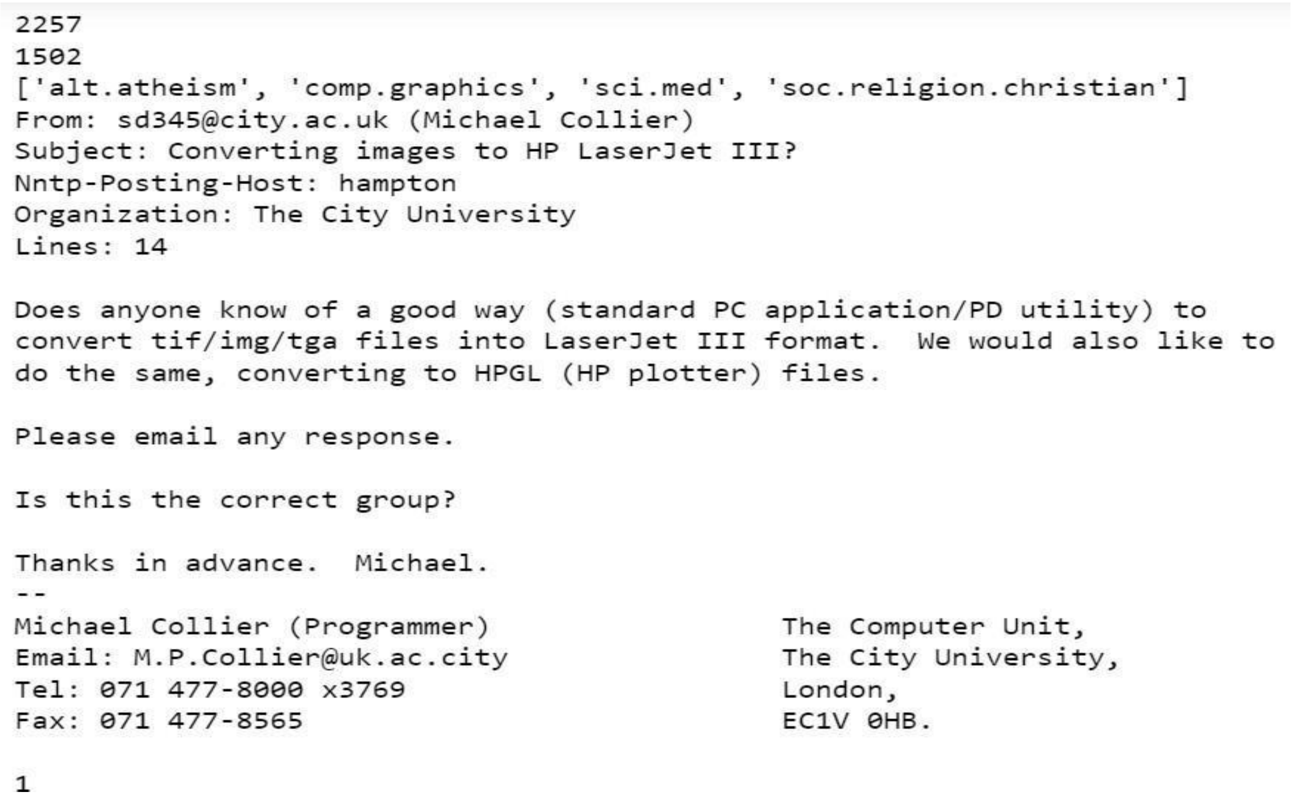
ALGORITHM:

1. Import basic libraries.
2. Importing the dataset.
3. Data preprocessing.
4. Training the model.
5. Testing and evaluation of the model.
6. Visualizing the model.

PROGRAM:

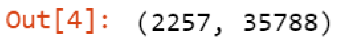
```
from sklearn.datasets import fetch_20newsgroups

from sklearn.metrics import confusion_matrix

from sklearn.metrics import classification_report

import numpy as np

categories = ['alt.atheism', 'soc.religion.christian','comp.graphics', 'sci.med']

twenty_train = fetch_20newsgroups(subset='train',categories=categories,shuffle=True)

twenty_test = fetch_20newsgroups(subset='test',categories=categories,shuffle=True)

print(len(twenty_train.data))

print(len(twenty_test.data))

print(twenty_train.target_names)
print("\n".join(twenty_train.data[0].split("\n")))

print(twenty_train.target[0])
```

OUTPUT:



```
2257
1502
['alt.atheism', 'comp.graphics', 'sci.med', 'soc.religion.christian']
From: sd345@city.ac.uk (Michael Collier)
Subject: Converting images to HP LaserJet III?
Nntp-Posting-Host: hampton
Organization: The City University
Lines: 14

Does anyone know of a good way (standard PC application/PD utility) to
convert tif/img/tga files into LaserJet III format.  We would also like to
do the same, converting to HPGL (HP plotter) files.

Please email any response.

Is this the correct group?

Thanks in advance.  Michael.
--
Michael Collier (Programmer)                 The Computer Unit,
Email: M.P.Collier@uk.ac.city                The City University,
Tel: 071 477-8000 x3769                      London,
Fax: 071 477-8565                            EC1V 0HB.

1
```

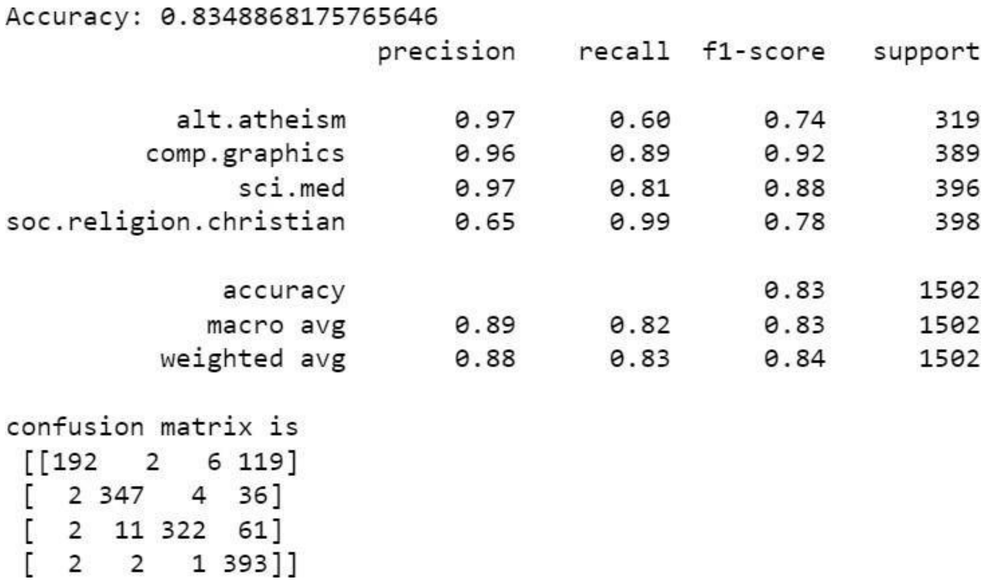
```
from sklearn.feature_extraction.text import CountVectorizer

count_vect = CountVectorizer()

X_train_tf = count_vect.fit_transform(twenty_train.data)

from sklearn.feature_extraction.text import TfidfTransformer

tfidf_transformer = TfidfTransformer()

X_train_tfidf = tfidf_transformer.fit_transform(X_train_tf)

X_train_tfidf.shape
```



```
Out[4]: (2257, 35788)
```

```
from sklearn.naive_bayes import MultinomialNB

from sklearn.metrics import accuracy_score
from sklearn import metrics
mod = MultinomialNB()
mod.fit(X_train_tfidf, twenty_train.target)
X_test_tf = count_vect.transform(twenty_test.data)
X_test_tfidf = tfidf_transformer.transform(X_test_tf)
predicted = mod.predict(X_test_tfidf)
print("Accuracy:", accuracy_score(twenty_test.target, predicted))
print(classification_report(twenty_test.target,predicted,target_names=twenty_test.target_names))
print("confusion matrix is \n",metrics.confusion_matrix(twenty_test.target, predicted))
```

OUTPUT:



```
Accuracy: 0.8348868175765646
                        precision    recall  f1-score   support

           alt.atheism       0.97      0.60      0.74       319
         comp.graphics       0.96      0.89      0.92       389
               sci.med       0.97      0.81      0.88       396
soc.religion.christian       0.65      0.99      0.78       398

              accuracy                           0.83      1502
             macro avg       0.89      0.82      0.83      1502
          weighted avg       0.88      0.83      0.84      1502

confusion matrix is
 [[192   2   6 119]
 [  2 347   4  36]
 [  2  11 322  61]
 [  2   2   1 393]]
```

RESULT:

Thus the accuracy and precision was measured by Naïve Bayesian classifier model.

EX. NO: 6

DATE :

CONSTRUCTION OF A BAYESIAN NETWORK TO DIAGNOSE CORONA INFECTION USING STANDARD WHO DATA SET

AIM:

To construct a Bayesian network to diagnose corona infection using WHO data set.

ALGORITHM:

This Naive Bayes is broken down into 5 parts:

1: Separate by Class.

2: Summarize Dataset.

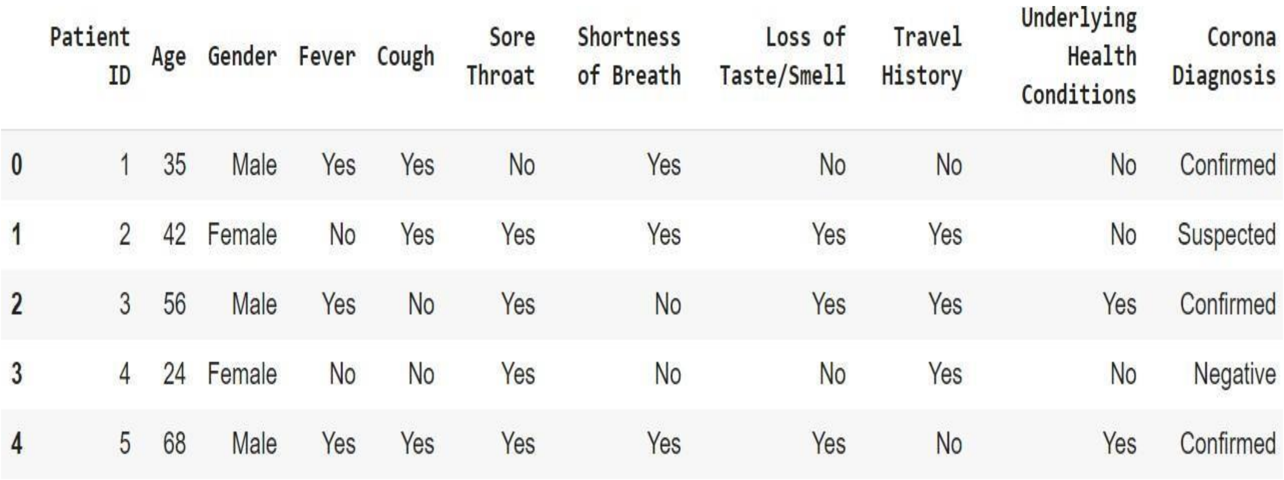
3: Summarize Data by Class.

4: Gaussian Probability Density Function.

5: Class Probabilities.

PROGRAM

```
import pandas as pd
covid_19_data=pd.read_csv("/content/corona.csv")
covid_19_data
```



	Patient ID	Age	Gender	Fever	Cough	Sore Throat	Shortness of Breath	Loss of Taste/Smell	Travel History	Underlying Health Conditions	Corona Diagnosis
0	1	35	Male	Yes	Yes	No	Yes	No	No	No	Confirmed
1	2	42	Female	No	Yes	Yes	Yes	Yes	Yes	No	Suspected
2	3	56	Male	Yes	No	Yes	No	Yes	Yes	Yes	Confirmed
3	4	24	Female	No	No	Yes	No	No	Yes	No	Negative
4	5	68	Male	Yes	Yes	Yes	Yes	Yes	No	Yes	Confirmed

import warnings

warnings.filterwarnings("ignore",category=FutureWarning)

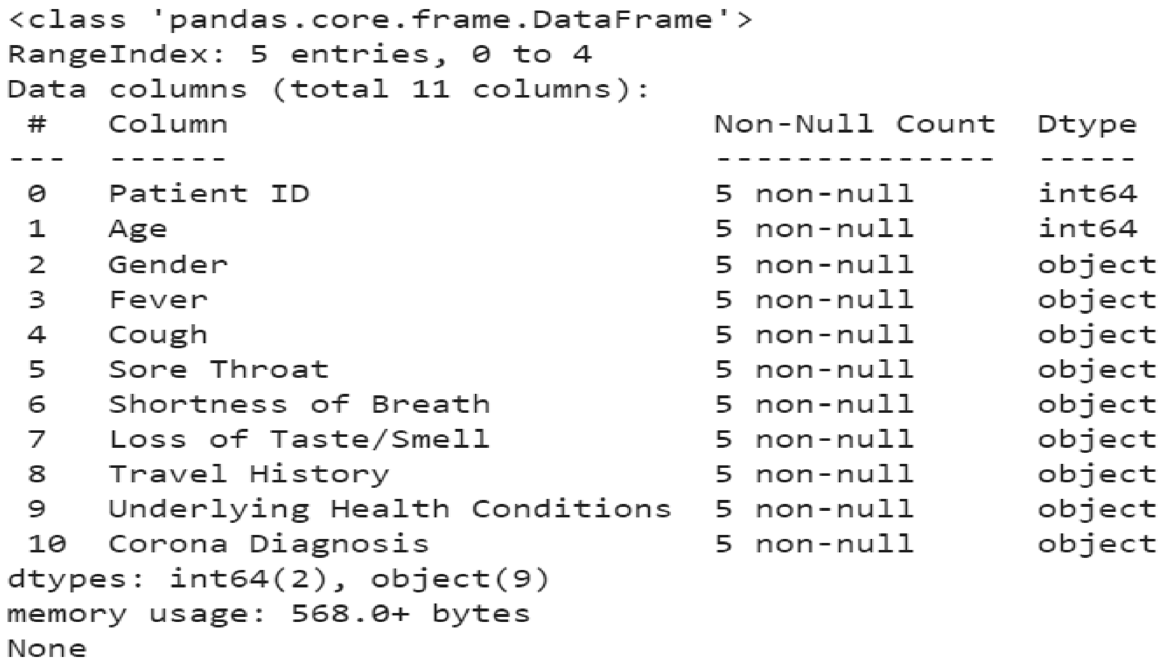
covid_19_data=pd.read_csv("/content/corona.csv")

print(f'The shape of the dataframe is {covid_19_data.shape}')

print()

The shape of the dataframe is (5, 11)

print(covid_19_data.info())



```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5 entries, 0 to 4
Data columns (total 11 columns):
 #   Column                       Non-Null Count  Dtype 
---  ------                       --------------  ----- 
 0   Patient ID                   5 non-null      int64 
 1   Age                          5 non-null      int64 
 2   Gender                       5 non-null      object
 3   Fever                        5 non-null      object
 4   Cough                        5 non-null      object
 5   Sore Throat                  5 non-null      object
 6   Shortness of Breath          5 non-null      object
 7   Loss of Taste/Smell          5 non-null      object
 8   Travel History               5 non-null      object
 9   Underlying Health Conditions  5 non-null      object
 10  Corona Diagnosis             5 non-null      object
dtypes: int64(2), object(9)
memory usage: 568.0+ bytes
None
```

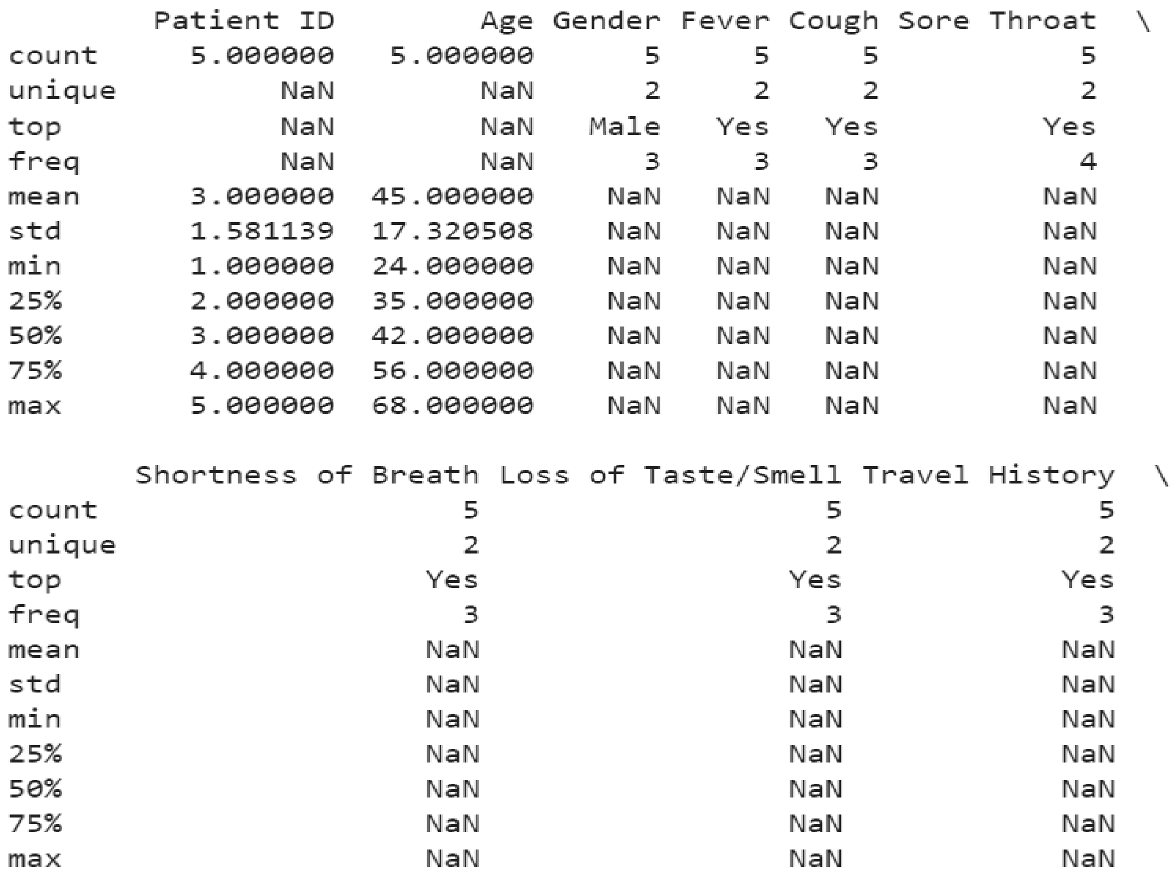
print()

import numpy as np

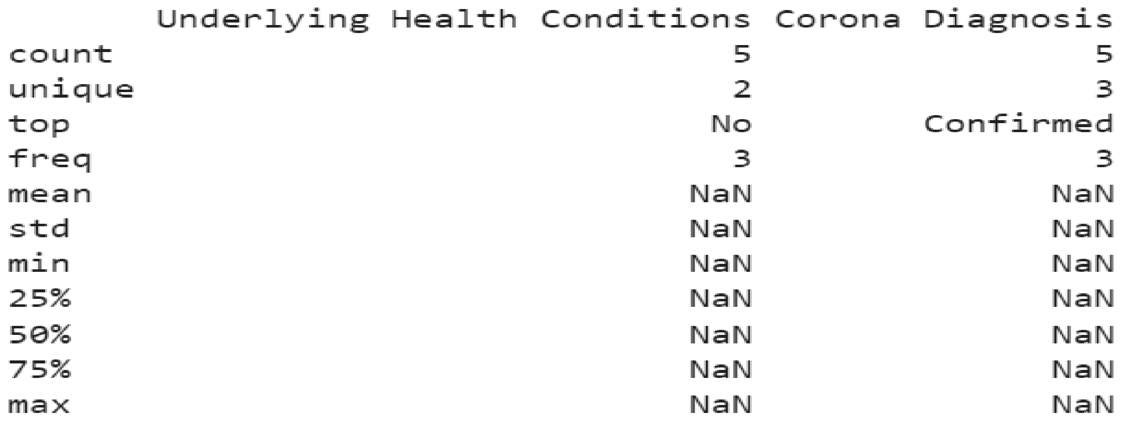
covid_19_data.replace(to_replace='?',value=np.NaN,inplace=True)

print(covid_19_data.describe(include='all'))

print()

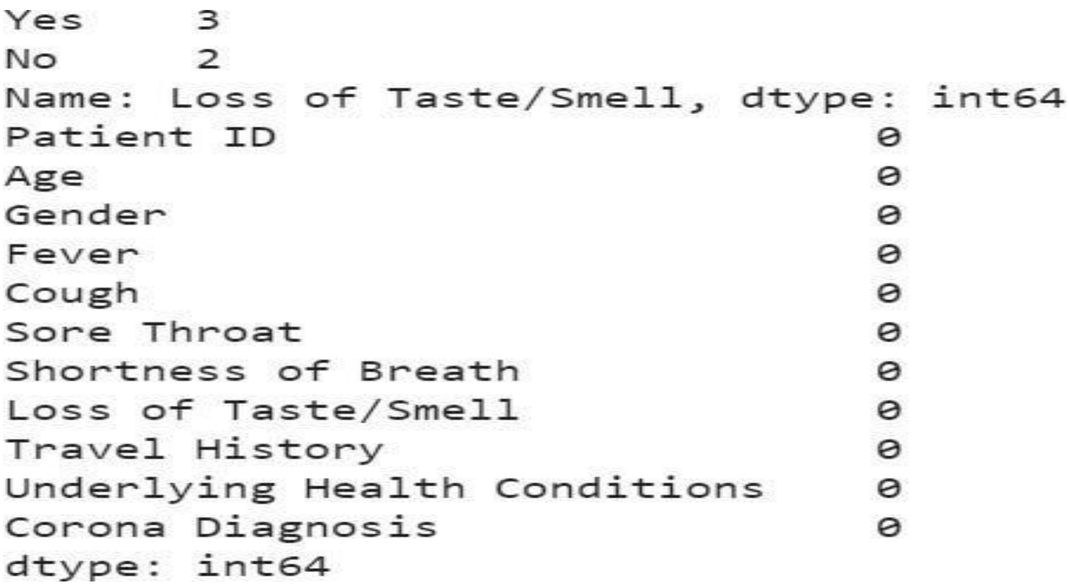


```
       Patient ID        Age Gender Fever Cough Sore Throat  \
count    5.000000   5.000000      5     5     5           5
unique        NaN        NaN      2     2     2           2
top           NaN        NaN   Male   Yes   Yes         Yes
freq          NaN        NaN      3     3     3           4
mean     3.000000  45.000000    NaN   NaN   NaN         NaN
std      1.581139  17.320508    NaN   NaN   NaN         NaN
min      1.000000  24.000000    NaN   NaN   NaN         NaN
25%      2.000000  35.000000    NaN   NaN   NaN         NaN
50%      3.000000  42.000000    NaN   NaN   NaN         NaN
75%      4.000000  56.000000    NaN   NaN   NaN         NaN
max      5.000000  68.000000    NaN   NaN   NaN         NaN

       Shortness of Breath Loss of Taste/Smell Travel History  \
count                    5                   5              5
unique                   2                   2              2
top                    Yes                 Yes            Yes
freq                     3                   3              3
mean                   NaN                 NaN            NaN
std                    NaN                 NaN            NaN
min                    NaN                 NaN            NaN
25%                    NaN                 NaN            NaN
50%                    NaN                 NaN            NaN
75%                    NaN                 NaN            NaN
max                    NaN                 NaN            NaN
```



```
       Underlying Health Conditions Corona Diagnosis
count                             5                5
unique                            2                3
top                              No        Confirmed
freq                              3                3
mean                            NaN              NaN
std                             NaN              NaN
min                             NaN              NaN
25%                             NaN              NaN
50%                             NaN              NaN
75%                             NaN              NaN
max                             NaN              NaN
```

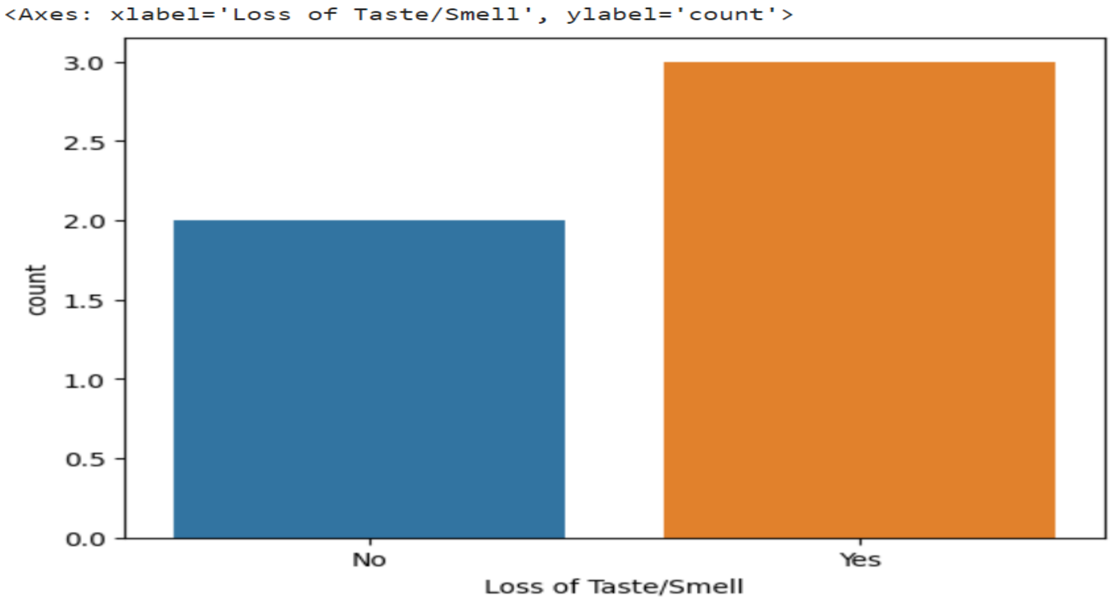
```
print(covid_19_data["Loss of Taste/Smell"].value_counts())
print(covid_19_data.isnull().sum())
```



```
Yes     3
No      2
Name: Loss of Taste/Smell, dtype: int64
Patient ID                          0
Age                                 0
Gender                              0
Fever                               0
Cough                               0
Sore Throat                         0
Shortness of Breath                 0
Loss of Taste/Smell                 0
Travel History                      0
Underlying Health Conditions        0
Corona Diagnosis                    0
dtype: int64
```

import seaborn as sns

sns.countplot(x="Loss of Taste/Smell",data=covid_19_data,linewidth=3)



```
<Axes: xlabel='Loss of Taste/Smell', ylabel='count'>
```

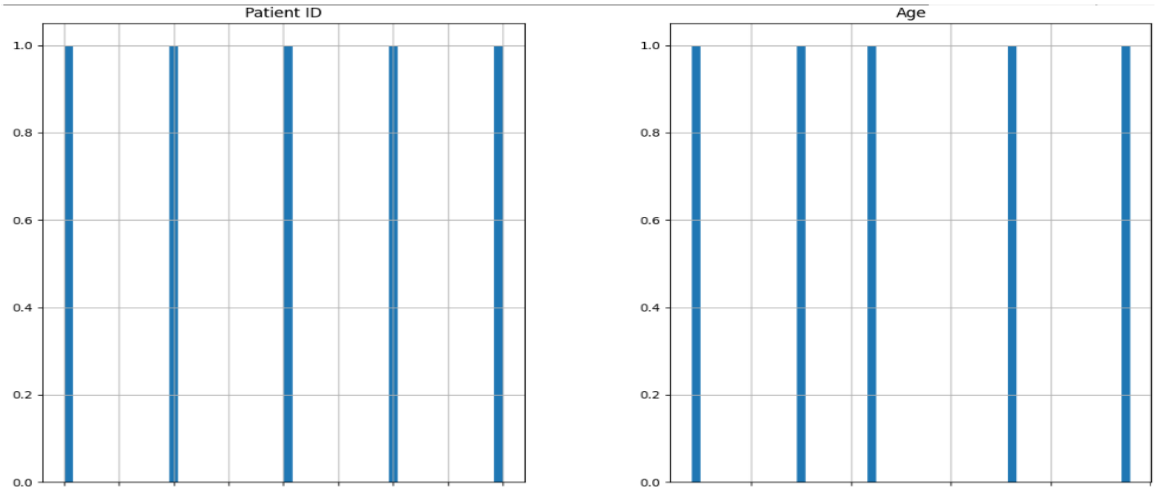
import matplotlib.pyplot as plt

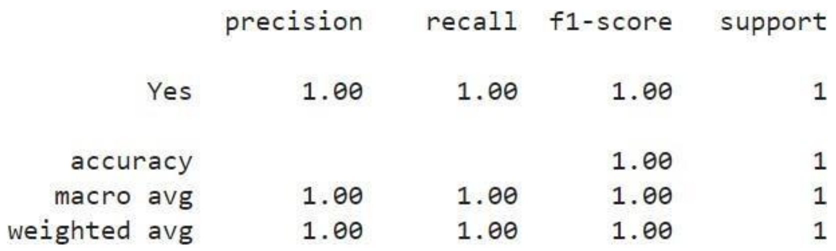
covid_19_data[['Patient ID','Age','Gender','Fever','Cough','Sore Throat']].hist(bins=50,figsize=(15,8))

plt.show()

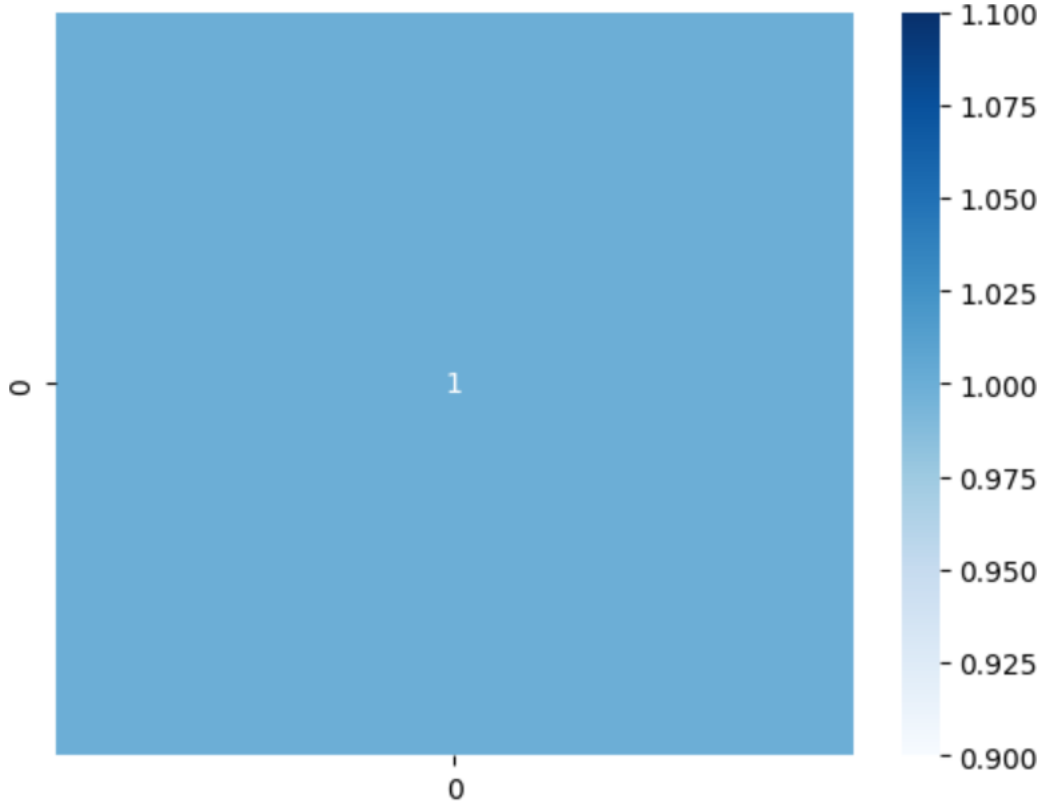
```
covid_19_data['Patient ID'].fillna(covid_19_data['Sore Throat'].mode()[0],inplace=True)

X=covid_19_data.drop(['Travel History'],axis=1)
y=covid_19_data.Cough
X=X[['Patient ID','Age']]
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2)
from sklearn.naive_bayes import GaussianNB
NB_classifier=GaussianNB()
NB_classifier.fit(X_test,y_test)
y_predict=NB_classifier.predict(X_test)
from sklearn.metrics import confusion_matrix
cm=confusion_matrix(y_test,y_predict)
sns.heatmap(cm,annot=True,cmap='Blues')
from sklearn.metrics import classification_report
print(classification_report(y_test,y_predict))
```

OUTPUT:





```
              precision    recall  f1-score   support

         Yes       1.00      1.00      1.00         1

    accuracy                           1.00         1
   macro avg       1.00      1.00      1.00         1
weighted avg       1.00      1.00      1.00         1
```



RESULT:

Thus the program to diagnose corona infection using Bayesian network was successfully implemented using python.