NEELESH GANGOPADHYAY
1BM18CS058
6B

ML- RECORD

LAB1: FIND S ALGORITHM

Dataset:

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

Code:

[1]:
```
import pandas as pd
import numpy as np

#to read the data in the csv file
data = pd.read_csv("data.csv")
print(data,"n")

#making an array of all the attributes
d = np.array(data)[:,:-1]
print("n The attributes are: ",d)

#segragating the target that has positive and negative examples
target = np.array(data)[:,-1]
print("n The target is: ",target)

#training function to implement find-s algorithm
def train(c,t):
    for i, val in enumerate(t):
        if val == "Yes":
            specific_hypothesis = c[i].copy()
            break

    for i, val in enumerate(c):
        if t[i] == "Yes":
            for x in range(len(specific_hypothesis)):
                if val[x] != specific_hypothesis[x]:
                    specific_hypothesis[x] = '?'
                else:
                    pass

    return specific_hypothesis

#obtaining the final hypothesis
print("n The final hypothesis is:",train(d,target))
```

NEELESH GANGOPADHYAY
1BM18CS058
6B

Output:

```
#obtaining the final hypothesis
print("n The final hypothesis is:",train(d,target))
```

```
    Time Weather Temperature Company Humidity   Wind Goes
0  Morning   Sunny       Warm     Yes     Mild Strong  Yes
1  Evening   Rainy       Cold      No     Mild Normal   No
2  Morning   Sunny    Moderate     Yes   Normal Normal  Yes
3  Evening   Sunny       Cold     Yes     High Strong  Yes n
n The attributes are:  [['Morning' 'Sunny' 'Warm' 'Yes' 'Mild' 'Strong']
 ['Evening' 'Rainy' 'Cold' 'No' 'Mild' 'Normal']
 ['Morning' 'Sunny' 'Moderate' 'Yes' 'Normal' 'Normal']
 ['Evening' 'Sunny' 'Cold' 'Yes' 'High' 'Strong']]
n The target is:  ['Yes' 'No' 'Yes' 'Yes']
n The final hypothesis is: ['?' 'Sunny' '?' 'Yes' '?' '?']
```

In [ ]:

LAB2 : CANDIDATE ELIMINATION ALGORITHM

Dataset:

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

NEELESH GANGOPADHYAY
1BM18CS058
6R

In [4]:
```
import numpy as np
import pandas as pd

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):
        print("For Loop Starts")
        if target[i] == "yes":
            print("If instance is Positive ")
            for x in range(len(specific_h)):
                if h[x]!= specific_h[x]:
                    specific_h[x] ='?'
                    general_h[x][x] ='?'

        if target[i] == "no":
            print("If instance is Negative ")
            for x in range(len(specific_h)):
                if h[x]!= specific_h[x]:
                    general_h[x][x] = specific_h[x]
                else:
                    general_h[x][x] = '?'

        print(" steps of Candidate Elimination Algorithm",i+1)
        print(specific_h)
        print(general_h)
        print("\n")
        print("\n")

    indices = [i for i, val in enumerate(general_h) if val == ['?', '?', '?', '?', '?', '?']]
    for i in indices:
        general_h.remove(['?', '?', '?', '?', '?', '?'])
    return specific_h, general_h

s_final, g_final = learn(concepts, target)

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

Output:

```
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']
[['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?',
'?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?']]
For Loop Starts
If instance is Positive
 steps of Candidate Elimination Algorithm 1
['sunny' 'warm' 'normal' 'strong' 'warm' 'same']
[['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?',
'?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?']]



For Loop Starts
If instance is Positive
 steps of Candidate Elimination Algorithm 2
['sunny' 'warm' '?' 'strong' 'warm' 'same']
[['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?',
'?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?']]



For Loop Starts
If instance is Negative
 steps of Candidate Elimination Algorithm 3
['sunny' 'warm' '?' 'strong' 'warm' 'same']
[['sunny', '?', '?', '?', '?', '?'], ['?', 'warm', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', 'same']]



For Loop Starts
If instance is Positive
 steps of Candidate Elimination Algorithm 4
['sunny' 'warm' '?' 'strong' '?' '?']
[['sunny', '?', '?', '?', '?', '?'], ['?', 'warm', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?']]



Final Specific_h:
['sunny' 'warm' '?' 'strong' '?' '?']
Final General_h:
[['sunny', '?', '?', '?', '?', '?'], ['?', 'warm', '?', '?', '?', '?']]
```

NEELESH GANGOPADHYAY
1BM18CS058
6B

LAB3:DECISION TREE

Dataset:

Id3.csv:

We can make this file beautiful and searchable if this error is corr

```
1  Outlook,Temperature,Humidity,Wind,Answer
2  sunny,hot,highweak,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
```

Id3_test.csv:

	Outlook	Temperature	Humidity	Wind
2	rain	cool	normal	strong
3	sunny	mild	normal	strong

NEELESH GANGOPADHYAY
1BM18CS058
6B

In [3]:
```
import math
import csv
def load_csv(filename):
    lines=csv.reader(open(filename,"r"));
    dataset = list(lines)
    headers = dataset.pop(0)
    return dataset,headers

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

def subtables(data,col,delete):
    dic={}
    coldata=[row[col] for row in data]
    attr=list(set(coldata))

    counts=[0]*len(attr)
    r=len(data)
    c=len(data[0])
    for x in range(len(attr)):
        for y in range(r):
            if data[y][col]==attr[x]:
                counts[x]+=1

    for x in range(len(attr)):
        dic[attr[x]]=[[0 for i in range(c)] for j in range(counts[x])]
        pos=0
        for y in range(r):
            if data[y][col]==attr[x]:
                if delete:
                    del data[y][col]
                dic[attr[x]][pos]=data[y]
                pos+=1
    return attr,dic

def entropy(S):
    attr=list(set(S))
    if len(attr)==1:
        return 0

    counts=[0,0]
    for i in range(2):
        counts[i]=sum([1 for x in S if attr[i]==x])/(len(S)*1.0)

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

```
def compute_gain(data,col):
    attr,dic = subtables(data,col,delete=False)

    total_size=len(data)
    entropies=[0]*len(attr)
    ratio=[0]*len(attr)

    total_entropy=entropy([row[-1] for row in data])
    for x in range(len(attr)):
        ratio[x]=len(dic[attr[x]])/(total_size*1.0)
        entropies[x]=entropy([row[-1] for row in dic[attr[x]]])
        total_entropy-=ratio[x]*entropies[x]
    return total_entropy

def build_tree(data,features):
    lastcol=[row[-1] for row in data]
    if(len(set(lastcol)))==1:
        node=Node("")
        node.answer=lastcol[0]
        return node

    n=len(data[0])-1
    gains=[0]*n
    for col in range(n):
        gains[col]=compute_gain(data,col)
    split=gains.index(max(gains))
    node=Node(features[split])
    fea = features[:split]+features[split+1:]


    attr,dic=subtables(data,split,delete=True)

    for x in range(len(attr)):
        child=build_tree(dic[attr[x]],fea)
        node.children.append((attr[x],child))
    return node

def print_tree(node,level):
    if node.answer!="":
        print("  "*level,node.answer)
        return

    print("  "*level,node.attribute)
    for value,n in node.children:
        print("  "*(level+1),value)
        print_tree(n,level+2)
```

NEELESH GANGOPADHYAY
1BM18CS058
6B

```
def classify(node,x_test,features):
    if node.answer!="":
        print(node.answer)
        return
    pos=features.index(node.attribute)
    for value, n in node.children:
        if x_test[pos]==value:
            classify(n,x_test,features)

'''Main program'''
dataset,features=load_csv("id3.csv")
node1=build_tree(dataset,features)

print("The decision tree for the dataset using ID3 algorithm is")
print_tree(node1,0)
testdata,features=load_csv("id3_test.csv")

for xtest in testdata:
    print("The test instance:",xtest)
    print("The label for test instance:",end="   ")
    classify(node1,xtest,features)
```

Output:

```
The decision tree for the dataset using ID3 algorithm is
 Outlook
   overcast
     yes
   sunny
     Humidity
       normal
         yes
       highweak
         no
       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
```

In [ ]:

NEELESH GANGOPADHYAY
1BM18CS058
6B

LAB4:NAIVE BAYESIAN CLASSIFIER

Example 1:

Dataset:

1	6	148	72	35	0	33.6	0.627	50	1
2	1	85	66	29	0	26.6	0.351	31	0
3	8	183	64	0	0	23.3	0.672	32	1
4	1	89	66	23	94	28.1	0.167	21	0
5	0	137	40	35	168	43.1	2.288	33	1
6	5	116	74	0	0	25.6	0.201	30	0
7	3	78	50	32	88	31	0.248	26	1
8	10	115	0	0	0	35.3	0.134	29	0
9	2	197	70	45	543	30.5	0.158	53	1
10	8	125	96	0	0	0	0.232	54	1
11	4	110	92	0	0	37.6	0.191	30	0
12	10	168	74	0	0	38	0.537	34	1
13	10	139	80	0	0	27.1	1.441	57	0
14	1	189	60	23	846	30.1	0.398	59	1
15	5	166	72	19	175	25.8	0.587	51	1
16	7	100	0	0	0	30	0.484	32	1
17	0	118	84	47	230	45.8	0.551	31	1
18	7	107	74	0	0	29.6	0.254	31	1
19	1	103	30	38	83	43.3	0.183	33	0
20	1	115	70	30	96	34.6	0.529	32	1
21	3	126	88	41	235	39.3	0.704	27	0
22	8	99	84	0	0	35.4	0.388	50	0

In [1]:
```
import csv
import random
import math

def loadcsv(filename):
        lines = csv.reader(open(filename, "r"));
        dataset = list(lines)
        for i in range(len(dataset)):
       #converting strings into numbers for processing
                dataset[i] = [float(x) for x in dataset[i]]

        return dataset

def splitdataset(dataset, splitratio):
    #67% training size
        trainsize = int(len(dataset) * splitratio);
        trainset = []
        copy = list(dataset);
        while len(trainset) < trainsize:
#generate indices for the dataset list randomly to pick ele for training data
                index = random.randrange(len(copy));
                trainset.append(copy.pop(index))
        return [trainset, copy]

def separatebyclass(dataset):
        separated = {} #dictionary of classes 1 and 0
#creates a dictionary of classes 1 and 0 where the values are
#the instances belonging to each class
        for i in range(len(dataset)):
                vector = dataset[i]
                if (vector[-1] not in separated):
                        separated[vector[-1]] = []
                separated[vector[-1]].append(vector)
        return separated

def mean(numbers):
        return sum(numbers)/float(len(numbers))

def stdev(numbers):
        avg = mean(numbers)
        variance = sum([pow(x-avg,2) for x in numbers])/float(len(numbers)-1)
        return math.sqrt(variance)
```

NEELESH GANGOPADHYAY
1BM18CS058
6B

```
def summarize(dataset): #creates a dictionary of classes
        summaries = [(mean(attribute), stdev(attribute)) for attribute in zip(*dataset)];
        del summaries[-1] #excluding labels +ve or -ve
        return summaries

def summarizebyclass(dataset):
        separated = separatebyclass(dataset);
    #print(separated)
        summaries = {}
        for classvalue, instances in separated.items():
#for key,value in dic.items()
#summaries is a dic of tuples(mean,std) for each class value
                summaries[classvalue] = summarize(instances) #summarize is used to cal to mean and std
        return summaries

def calculateprobability(x, mean, stdev):
        exponent = math.exp(-(math.pow(x-mean,2)/(2*math.pow(stdev,2))))
        return (1 / (math.sqrt(2*math.pi) * stdev)) * exponent

def calculateclassprobabilities(summaries, inputvector):
        probabilities = {} # probabilities contains the all prob of all class of test data
        for classvalue, classsummaries in summaries.items():#class and attribute information as mean and sd
                probabilities[classvalue] = 1
                for i in range(len(classsummaries)):
                        mean, stdev = classsummaries[i] #take mean and sd of every attribute for class 0 and 1 seperaely
                        x = inputvector[i] #testvector's first attribute
                        probabilities[classvalue] *= calculateprobability(x, mean, stdev);#use normal dist
        return probabilities

def predict(summaries, inputvector): #training and test data is passed
        probabilities = calculateclassprobabilities(summaries, inputvector)
        bestLabel, bestProb = None, -1
        for classvalue, probability in probabilities.items():#assigns that class which has he highest prob
                if bestLabel is None or probability > bestProb:
                        bestProb = probability
                        bestLabel = classvalue
        return bestLabel

def getpredictions(summaries, testset):
        predictions = []
        for i in range(len(testset)):
                result = predict(summaries, testset[i])
                predictions.append(result)
        return predictions
```

```
def getaccuracy(testset, predictions):
        correct = 0
        for i in range(len(testset)):
                if testset[i][-1] == predictions[i]:
                        correct += 1
        return (correct/float(len(testset))) * 100.0

def main():
        filename = 'naivedata.csv'
        splitratio = 0.67
        dataset = loadcsv(filename);

        trainingset, testset = splitdataset(dataset, splitratio)
        print('Split {0} rows into train={1} and test={2} rows'.format(len(dataset), len(trainingset), len(testset)))
        # prepare model
        summaries = summarizebyclass(trainingset);
        #print(summaries)
    # test model
        predictions = getpredictions(summaries, testset) #find the predictions of test data with the training data
        accuracy = getaccuracy(testset, predictions)
        print('Accuracy of the classifier is : {0}%'.format(accuracy))

main()
```

```
Split 768 rows into train=514 and test=254 rows
Accuracy of the classifier is : 74.01574803149606%
```

NEELESH GANGOPADHYAY
1BM18CS058
6B

Example 2:

Dataset:

1	63	1	1	145	233	1	2	150	0	2.3	3	0	6	0
2	67	1	4	160	286	0	2	108	1	1.5	2	3	3	2
3	67	1	4	120	229	0	2	129	1	2.6	2	2	7	1
4	37	1	3	130	250	0	0	187	0	3.5	3	0	3	0
5	41	0	2	130	204	0	2	172	0	1.4	1	0	3	0
6	56	1	2	120	236	0	0	178	0	0.8	1	0	3	0
7	62	0	4	140	268	0	2	160	0	3.6	3	2	3	3
8	57	0	4	120	354	0	0	163	1	0.6	1	0	3	0
9	63	1	4	130	254	0	2	147	0	1.4	2	1	7	2
10	53	1	4	140	203	1	2	155	1	3.1	3	0	7	1
11	57	1	4	140	192	0	0	148	0	0.4	2	0	6	0
12	56	0	2	140	294	0	2	153	0	1.3	2	0	3	0
13	56	1	3	130	256	1	2	142	1	0.6	2	1	6	2
14	44	1	2	120	263	0	0	173	0	0	1	0	7	0
15	52	1	3	172	199	1	0	162	0	0.5	1	0	7	0
16	57	1	3	150	168	0	0	174	0	1.6	1	0	3	0
17	48	1	2	110	229	0	0	168	0	1	3	0	7	1
18	54	1	4	140	239	0	0	160	0	1.2	1	0	3	0
19	48	0	3	130	275	0	0	139	0	0.2	1	0	3	0
20	49	1	2	130	266	0	0	171	0	0.6	1	0	3	0
21	64	1	1	110	211	0	2	144	1	1.8	2	0	3	0

In [1]:
```
# Naive Bayes On The Iris Dataset
from csv import reader
from random import seed
from random import randrange
from math import sqrt
from math import exp
from math import pi
```

In [2]:
```
# Load a CSV file
def load_csv(filename):
	dataset = list()
	with open(filename, 'r') as file:
		csv_reader = reader(file)
		for row in csv_reader:
			if not row:
				continue
			dataset.append(row)
	return dataset
```

In [3]:
```
# Convert string column to float
def str_column_to_float(dataset, column):
	for row in dataset:
		row[column] = float(row[column].strip())
```

In [4]:
```
# Convert string column to integer
def str_column_to_int(dataset, column):
	class_values = [row[column] for row in dataset]
	unique = set(class_values)
	lookup = dict()
	for i, value in enumerate(unique):
		lookup[value] = i
	for row in dataset:
		row[column] = lookup[row[column]]
	return lookup
```

In [5]:
```
# Split a dataset into k folds
def cross_validation_split(dataset, n_folds):
	dataset_split = list()
	dataset_copy = list(dataset)
	fold_size = int(len(dataset) / n_folds)
	for _ in range(n_folds):
		fold = list()
		while len(fold) < fold_size:
			index = randrange(len(dataset_copy))
			fold.append(dataset_copy.pop(index))
		dataset_split.append(fold)
	return dataset_split
```

NEELESH GANGOPADHYAY
1BM18CS058
6B

In [6]:
```
# Calculate accuracy percentage
def accuracy_metric(actual, predicted):
        correct = 0
        for i in range(len(actual)):
                if actual[i] == predicted[i]:
                        correct += 1
        return correct / float(len(actual)) * 100.0
```

In [7]:
```
# Evaluate an algorithm using a cross validation split
def evaluate_algorithm(dataset, algorithm, n_folds, *args):
        folds = cross_validation_split(dataset, n_folds)
        scores = list()
        for fold in folds:
                train_set = list(folds)
                train_set.remove(fold)
                train_set = sum(train_set, [])
                test_set = list()
                for row in fold:
                        row_copy = list(row)
                        test_set.append(row_copy)
                        row_copy[-1] = None
                predicted = algorithm(train_set, test_set, *args)
                actual = [row[-1] for row in fold]
                accuracy = accuracy_metric(actual, predicted)
                scores.append(accuracy)
        return scores
```

In [8]:
```
# Split the dataset by class values, returns a dictionary
def separate_by_class(dataset):
        separated = dict()
        for i in range(len(dataset)):
                vector = dataset[i]
                class_value = vector[-1]
                if (class_value not in separated):
                        separated[class_value] = list()
                separated[class_value].append(vector)
        return separated
```

In [9]:
```
# Calculate the mean of a list of numbers
def mean(numbers):
        return sum(numbers)/float(len(numbers))
```

In [10]:
```
# Calculate the standard deviation of a list of numbers
def stdev(numbers):
        avg = mean(numbers)
        variance = sum([(x-avg)**2 for x in numbers]) / float(len(numbers)-1)
        return sqrt(variance)
```

In [11]:
```
# Calculate the mean, stdev and count for each column in a dataset
def summarize_dataset(dataset):
        summaries = [(mean(column), stdev(column), len(column)) for column in zip(*dataset)]
        del(summaries[-1])
        return summaries
```

In [12]:
```
# Split dataset by class then calculate statistics for each row
def summarize_by_class(dataset):
        separated = separate_by_class(dataset)
        summaries = dict()
        for class_value, rows in separated.items():
                summaries[class_value] = summarize_dataset(rows)
        return summaries
```

In [13]:
```
# Calculate the Gaussian probability distribution function for x
def calculate_probability(x, mean, stdev):
        exponent = exp(-((x-mean)**2 / (2 * stdev**2 )))
        return (1 / (sqrt(2 * pi) * stdev)) * exponent
```

In [14]:
```
# Calculate the probabilities of predicting each class for a given row
def calculate_class_probabilities(summaries, row):
        total_rows = sum([summaries[label][0][2] for label in summaries])
        probabilities = dict()
        for class_value, class_summaries in summaries.items():
                probabilities[class_value] = summaries[class_value][0][2]/float(total_rows)
                for i in range(len(class_summaries)):
                        mean, stdev, _ = class_summaries[i]
                        probabilities[class_value] *= calculate_probability(row[i], mean, stdev)
        return probabilities
```

In [15]:
```
# Predict the class for a given row
def predict(summaries, row):
        probabilities = calculate_class_probabilities(summaries, row)
        best_label, best_prob = None, -1
        for class_value, probability in probabilities.items():
                if best_label is None or probability > best_prob:
                        best_prob = probability
                        best_label = class_value
        return best_label
```

In [16]:
```
# Naive Bayes Algorithm
def naive_bayes(train, test):
        summarize = summarize_by_class(train)
        predictions = list()
        for row in test:
                output = predict(summarize, row)
                predictions.append(output)
        return(predictions)
```

NEELESH GANGOPADHYAY
1BM18CS058
6B

In [17]:
```
# Test Naive Bayes on Iris Dataset
seed(1)
filename = 'Iris.csv'
dataset = load_csv(filename)
for i in range(len(dataset[0])-1):
        str_column_to_float(dataset, i)
# convert class column to integers
str_column_to_int(dataset, len(dataset[0])-1)
# evaluate algorithm
n_folds = 5
scores = evaluate_algorithm(dataset, naive_bayes, n_folds)
print('Scores: %s' % scores)
print('Mean Accuracy: %.3f%%' % (sum(scores)/float(len(scores))))
```

```
Scores: [93.33333333333333, 96.66666666666667, 100.0, 93.33333333333333, 93.33333333333333]
Mean Accuracy: 95.333%
```

In [18]:
```
# fit model
model = summarize_by_class(dataset)
# define a new record
row = [5.7,2.9,4.2,1.3]
# predict the label
label = predict(model, row)
print('Data=%s, Predicted: %s' % (row, label))
```

```
Data=[5.7, 2.9, 4.2, 1.3], Predicted: 1
```

In [ ]:

LAB5:BAYESIAN NETWORK

Example 1:

Dataset:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	heartdisease
2	63	1	1	145	233	1	2	150	0	2.3	3	0	6	0
3	67	1	4	160	286	0	2	108	1	1.5	2	3	3	2
4	67	1	4	120	229	0	2	129	1	2.6	2	2	7	1
5	37	1	3	130	250	0	0	187	0	3.5	3	0	3	0
6	41	0	2	130	204	0	2	172	0	1.4	1	0	3	0
7	56	1	2	120	236	0	0	178	0	0.8	1	0	3	0
8	62	0	4	140	268	0	2	160	0	3.6	3	2	3	3
9	57	0	4	120	354	0	0	163	1	0.6	1	0	3	0
10	63	1	4	130	254	0	2	147	0	1.4	2	1	7	2
11	53	1	4	140	203	1	2	155	1	3.1	3	0	7	1
12	57	1	4	140	192	0	0	148	0	0.4	2	0	6	0
13	56	0	2	140	294	0	2	153	0	1.3	2	0	3	0
14	56	1	3	130	256	1	2	142	1	0.6	2	1	6	2
15	44	1	2	120	263	0	0	173	0	0	1	0	7	0
16	52	1	3	172	199	1	0	162	0	0.5	1	0	7	0
17	57	1	3	150	168	0	0	174	0	1.6	1	0	3	0
18	48	1	2	110	229	0	0	168	0	1	3	0	7	1
19	54	1	4	140	239	0	0	160	0	1.2	1	0	3	0
20	48	0	3	130	275	0	0	139	0	0.2	1	0	3	0
21	49	1	2	130	266	0	0	171	0	0.6	1	0	3	0
22	64	1	1	110	211	0	2	144	1	1.8	2	0	3	0
23	58	0	1	150	283	1	2	162	0	1	1	0	3	0
24	58	1	2	120	284	0	2	160	0	1.8	2	0	3	1

NEELESH GANGOPADHYAY
1BM18CS058
6B

In [6]:
```
# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd
import pgmpy as pgmpy
from pgmpy.estimators import MaximumLikelihoodEstimator
from pgmpy.models import BayesianModel
from pgmpy.inference import VariableElimination

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that gets preserved as output when you create a version using "Sav
e & Run All"
# You can also write temporary files to /kaggle/temp/, but they won't be saved outside of the current session
```

```
/kaggle/input/bayesiannetwork/heart.csv
```

In [5]:
```
pip install pgmpy
```

```
Collecting pgmpy
  Downloading pgmpy-0.1.14-py3-none-any.whl (331 kB)
     |████████████████████████████████| 331 kB 4.4 MB/s eta 0:00:01
Requirement already satisfied: joblib in /opt/conda/lib/python3.7/site-packages (from pgmpy) (1.0.1)
Requirement already satisfied: pyparsing in /opt/conda/lib/python3.7/site-packages (from pgmpy) (2.4.7)
Requirement already satisfied: scipy in /opt/conda/lib/python3.7/site-packages (from pgmpy) (1.5.4)
Requirement already satisfied: tqdm in /opt/conda/lib/python3.7/site-packages (from pgmpy) (4.59.0)
Requirement already satisfied: statsmodels in /opt/conda/lib/python3.7/site-packages (from pgmpy) (0.12.2)
Requirement already satisfied: pandas in /opt/conda/lib/python3.7/site-packages (from pgmpy) (1.2.3)
Requirement already satisfied: numpy in /opt/conda/lib/python3.7/site-packages (from pgmpy) (1.19.5)
Requirement already satisfied: networkx in /opt/conda/lib/python3.7/site-packages (from pgmpy) (2.5)
Requirement already satisfied: scikit-learn in /opt/conda/lib/python3.7/site-packages (from pgmpy) (0.24.1)
Requirement already satisfied: torch in /opt/conda/lib/python3.7/site-packages (from pgmpy) (1.7.0)
Requirement already satisfied: decorator>=4.3.0 in /opt/conda/lib/python3.7/site-packages (from networkx->pgmpy) (4.4.2)
Requirement already satisfied: python-dateutil>=2.7.3 in /opt/conda/lib/python3.7/site-packages (from pandas->pgmpy) (2.8.1)
Requirement already satisfied: pytz>=2017.3 in /opt/conda/lib/python3.7/site-packages (from pandas->pgmpy) (2021.1)
Requirement already satisfied: six>=1.5 in /opt/conda/lib/python3.7/site-packages (from python-dateutil>=2.7.3->pandas->pgmpy) (1.15.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in /opt/conda/lib/python3.7/site-packages (from scikit-learn->pgmpy) (2.1.0)
Requirement already satisfied: patsy>=0.5 in /opt/conda/lib/python3.7/site-packages (from statsmodels->pgmpy) (0.5.1)
Requirement already satisfied: future in /opt/conda/lib/python3.7/site-packages (from torch->pgmpy) (0.18.2)
Requirement already satisfied: typing_extensions in /opt/conda/lib/python3.7/site-packages (from torch->pgmpy) (3.7.4.3)
Requirement already satisfied: dataclasses in /opt/conda/lib/python3.7/site-packages (from torch->pgmpy) (0.6)
Installing collected packages: pgmpy
Successfully installed pgmpy-0.1.14
Note: you may need to restart the kernel to use updated packages.
```

In [0]:
```
#read Cleveland Heart Disease data
heartDisease = pd.read_csv("/kaggle/input/bayesiannetwork/heart.csv")
heartDisease = heartDisease.replace('?',np.nan)
#display the data
print('Sample instances from the dataset are given below')
print(heartDisease.head())
#display the Attributes names and datatyes
print('\n Attributes and datatypes')
print(heartDisease.dtypes)
#Creat Model- Bayesian Network
model = BayesianModel([('age','heartdisease'),('sex','heartdisease'),('exang','heartdisease'),('cp','heartdisease'),('heartdisease','rest
ecg'),('heartdisease','chol')])
#Learning CPDs using Maximum Likelihood Estimators
print('\n Learning CPD using Maximum likelihood estimators')
model.fit(heartDisease,estimator=MaximumLikelihoodEstimator)
# Inferencing with Bayesian Network
print('\n Inferencing with Bayesian Networks:')
HeartDiseasetest_infer = VariableElimination(model)
#computing the Probability of HeartDisease given restecg
print('\n 1.Probability of HeartDisease given evidence= restecg :1')
q1=HeartDiseasetest_infer.query(variables=['heartdisease'],evidence={'restecg':1})
print(q1)
#computing the Probability of HeartDisease given cp
print('\n 2.Probability of HeartDisease given evidence= cp:2 ')
q2=HeartDiseasetest_infer.query(variables=['heartdisease'],evidence={'cp':2})
print(q2)
```

```
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

 Learning CPD using Maximum likelihood estimators
```

NEELESH GANGOPADHYAY
1BM18CS058
6B

```
Finding Elimination Order: :   0%|          | 0/5 [00:00<?, ?it/s]
  0%|          | 0/5 [00:00<?, ?it/s]
Eliminating: age:   0%|          | 0/5 [00:00<?, ?it/s]
Eliminating: chol:   0%|          | 0/5 [00:00<?, ?it/s]
Eliminating: cp:   0%|          | 0/5 [00:00<?, ?it/s]
Eliminating: sex:   0%|          | 0/5 [00:00<?, ?it/s]
Eliminating: exang: 100%|██████████| 5/5 [00:00<00:00, 189.65it/s]
Finding Elimination Order: : 100%|██████████| 5/5 [00:00<00:00, 132.81it/s]
Finding Elimination Order: :   0%|          | 0/5 [00:00<?, ?it/s]
  0%|          | 0/5 [00:00<?, ?it/s]
Eliminating: age:   0%|          | 0/5 [00:00<?, ?it/s]
Eliminating: chol:   0%|          | 0/5 [00:00<?, ?it/s]
Eliminating: restecg:   0%|          | 0/5 [00:00<?, ?it/s]
Eliminating: sex:   0%|          | 0/5 [00:00<?, ?it/s]
Eliminating: exang: 100%|██████████| 5/5 [00:00<00:00, 230.00it/s]
 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 |
+-----------------+---------------------+

 2.Probability of HeartDisease given evidence= cp:2
+-----------------+---------------------+
| heartdisease    |   phi(heartdisease) |
+=================+=====================+
| heartdisease(0) |              0.3610 |
+-----------------+---------------------+
| heartdisease(1) |              0.2159 |
+-----------------+---------------------+
| heartdisease(2) |              0.1373 |
+-----------------+---------------------+
| heartdisease(3) |              0.1537 |
+-----------------+---------------------+
| heartdisease(4) |              0.1321 |
+-----------------+---------------------+
```

Example 2:

In [5]:
```
from pgmpy.models import BayesianModel
from pgmpy.factors.discrete import TabularCPD
from pgmpy.inference import VariableElimination
```

In [3]:
```
pip install pgmpy
```

```
Collecting pgmpy
  Downloading pgmpy-0.1.14-py3-none-any.whl (331 kB)
     |████████████████████████████████| 331 kB 412 kB/s eta 0:00:01
Requirement already satisfied: numpy in /opt/conda/lib/python3.7/site-packages (from pgmpy) (1.19.5)
Requirement already satisfied: pyparsing in /opt/conda/lib/python3.7/site-packages (from pgmpy) (2.4.7)
Requirement already satisfied: networkx in /opt/conda/lib/python3.7/site-packages (from pgmpy) (2.5)
Requirement already satisfied: torch in /opt/conda/lib/python3.7/site-packages (from pgmpy) (1.7.0)
Requirement already satisfied: joblib in /opt/conda/lib/python3.7/site-packages (from pgmpy) (1.0.1)
Requirement already satisfied: scipy in /opt/conda/lib/python3.7/site-packages (from pgmpy) (1.5.4)
Requirement already satisfied: tqdm in /opt/conda/lib/python3.7/site-packages (from pgmpy) (4.56.2)
Requirement already satisfied: pandas in /opt/conda/lib/python3.7/site-packages (from pgmpy) (1.2.2)
Requirement already satisfied: statsmodels in /opt/conda/lib/python3.7/site-packages (from pgmpy) (0.12.2)
Requirement already satisfied: scikit-learn in /opt/conda/lib/python3.7/site-packages (from pgmpy) (0.24.1)
Requirement already satisfied: decorator>=4.3.0 in /opt/conda/lib/python3.7/site-packages (from networkx->pgmpy) (4.4.2)
Requirement already satisfied: python-dateutil>=2.7.3 in /opt/conda/lib/python3.7/site-packages (from pandas->pgmpy) (2.8.1)
Requirement already satisfied: pytz>=2017.3 in /opt/conda/lib/python3.7/site-packages (from pandas->pgmpy) (2021.1)
Requirement already satisfied: six>=1.5 in /opt/conda/lib/python3.7/site-packages (from python-dateutil>=2.7.3->pandas->pgmpy) (1.15.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in /opt/conda/lib/python3.7/site-packages (from scikit-learn->pgmpy) (2.1.0)
Requirement already satisfied: patsy>=0.5 in /opt/conda/lib/python3.7/site-packages (from statsmodels->pgmpy) (0.5.1)
Requirement already satisfied: future in /opt/conda/lib/python3.7/site-packages (from torch->pgmpy) (0.18.2)
Requirement already satisfied: typing_extensions in /opt/conda/lib/python3.7/site-packages (from torch->pgmpy) (3.7.4.3)
Requirement already satisfied: dataclasses in /opt/conda/lib/python3.7/site-packages (from torch->pgmpy) (0.6)
Installing collected packages: pgmpy
Successfully installed pgmpy-0.1.14
Note: you may need to restart the kernel to use updated packages.
```

In [8]:
```
cancer_model = BayesianModel([('Pollution','Cancer'),
                              ('Smoker','Cancer'),
                              ('Cancer','Xray'),
                              ('Cancer','Dyspnoea')])
print('Bayesian network nodes are:')
print("\t",cancer_model.nodes())
print('Bayesian network 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])
```

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

NEELESH GANGOPADHYAY
1BM18CS058
6B

In [9]:
```
cancer_model.add_cpds(cpd_poll,cpd_smoke,cpd_cancer,cpd_xray,cpd_dysp)
print('Model generated by adding cpts(cpds)')
print('Checking correctness of model:',end='')
print(cancer_model.check_model())
```

```
Model generated by adding cpts(cpds)
Checking correctness of model:True
```

In [10]:
```
print('All local depencies are as follows')
cancer_model.get_independencies()
```

```
All local depencies are as follows
```

Out[10]:
```
(Pollution ⊥ Smoker)
(Pollution ⊥ Dyspnoea, Xray | Cancer)
(Pollution ⊥ Xray | Dyspnoea, Cancer)
(Pollution ⊥ Dyspnoea | Cancer, Xray)
(Pollution ⊥ Dyspnoea, Xray | Cancer, Smoker)
(Pollution ⊥ Xray | Dyspnoea, Cancer, Smoker)
(Pollution ⊥ Dyspnoea | Cancer, Xray, Smoker)
(Smoker ⊥ Pollution)
(Smoker ⊥ Dyspnoea, Xray | Cancer)
(Smoker ⊥ Xray | Dyspnoea, Cancer)
(Smoker ⊥ Dyspnoea, Xray | Pollution, Cancer)
(Smoker ⊥ Dyspnoea | Cancer, Xray)
(Smoker ⊥ Xray | Dyspnoea, Pollution, Cancer)
(Smoker ⊥ Dyspnoea | Pollution, Cancer, Xray)
(Xray ⊥ Dyspnoea, Pollution, Smoker | Cancer)
(Xray ⊥ Pollution, Smoker | Dyspnoea, Cancer)
(Xray ⊥ Dyspnoea, Smoker | Pollution, Cancer)
(Xray ⊥ Dyspnoea, Pollution | Cancer, Smoker)
(Xray ⊥ Smoker | Dyspnoea, Pollution, Cancer)
(Xray ⊥ Pollution | Dyspnoea, Cancer, Smoker)
(Xray ⊥ Dyspnoea | Pollution, Cancer, Smoker)
(Dyspnoea ⊥ Pollution, Xray, Smoker | Cancer)
(Dyspnoea ⊥ Xray, Smoker | Pollution, Cancer)
(Dyspnoea ⊥ Pollution, Smoker | Cancer, Xray)
(Dyspnoea ⊥ Pollution, Xray | Cancer, Smoker)
(Dyspnoea ⊥ Smoker | Pollution, Cancer, Xray)
(Dyspnoea ⊥ Xray | Pollution, Cancer, Smoker)
(Dyspnoea ⊥ Pollution | Cancer, Xray, Smoker)
```

In [12]:
```
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'))
print(cancer_model.get_cpds('Dyspnoea'))
```

```
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) |
+---------+----------+----------+
| Xray(0) | 0.9      | 0.2      |
+---------+----------+----------+
| Xray(1) | 0.1      | 0.8      |
+---------+----------+----------+
+------------+-----------+----------+
| Cancer     | Cancer(0) | Cancer(1) |
+------------+-----------+----------+
| Dyspnoea(0) | 0.65     | 0.3      |
+------------+-----------+----------+
| Dyspnoea(1) | 0.35     | 0.7      |
+------------+-----------+----------+
```

In [14]:
```
cancer_infer=VariableElimination(cancer_model)
print('\n Inferencing with bayesian network')
print("\n Probability of Cancer given smoker")
q=cancer_infer.query(variables=['Cancer'],evidence={'Smoker':1})
print(q)

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

NEELESH GANGOPADHYAY
1BM18CS058
6B

```
Finding Elimination Order: :   0%|          | 0/3 [00:00<?, ?it/s]
  0%|          | 0/3 [00:00<?, ?it/s]
Eliminating: Dyspnoea:   0%|          | 0/3 [00:00<?, ?it/s]
Eliminating: Pollution:   0%|          | 0/3 [00:00<?, ?it/s]
Eliminating: Xray: 100%|██████████| 3/3 [00:00<00:00, 359.52it/s]

  0%|          | 0/2 [00:00<?, ?it/s]
Finding Elimination Order: :   0%|          | 0/2 [00:00<?, ?it/s]

  0%|          | 0/2 [00:00<?, ?it/s]

Eliminating: Dyspnoea:   0%|          | 0/2 [00:00<?, ?it/s]

Eliminating: Xray: 100%|██████████| 2/2 [00:00<00:00, 333.49it/s]A
 Inferencing with bayesian network

 Probability of Cancer given smoker
+-----------+---------------+
| Cancer    |   phi(Cancer) |
+===========+===============+
| Cancer(0) |        0.0029 |
+-----------+---------------+
| Cancer(1) |        0.9971 |
+-----------+---------------+

 Probability of Cancer given smoker,pollution
+-----------+---------------+
| Cancer    |   phi(Cancer) |
+===========+===============+
| Cancer(0) |        0.0200 |
+-----------+---------------+
| Cancer(1) |        0.9800 |
+-----------+---------------+
```

THANK YOU