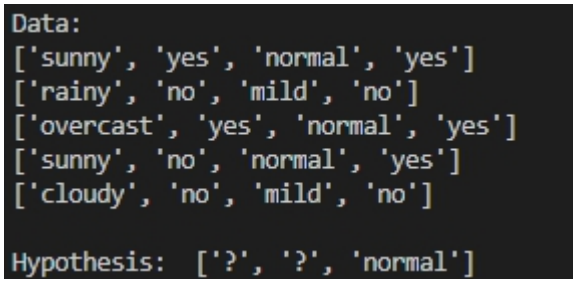
ML LAB-1 REPORT

PROGRAM-1
Date-10/03/2021

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

```
import csv

def updateHypothesis(x,h):
    if h==[]:
        return x
   for i in range(0,len(h)):
        if x[i].upper()!=h[i].upper():
            h[i] = '?'
    return h

if __name__ == "__main__":
    data = []
    h = []

    # reading csv file
    with open('data.csv', 'r') as file:
        reader = csv.reader(file)
        print("Data: ")
        for row in reader:
            data.append(row)
            print(row)

    if data:
        for x in data:
            if x[-1].upper()=="YES":
                x.pop() # removing last field
                h = updateHypothesis(x,h)

    print("\nHypothesis: ",h)
```

SCREENSHOTS



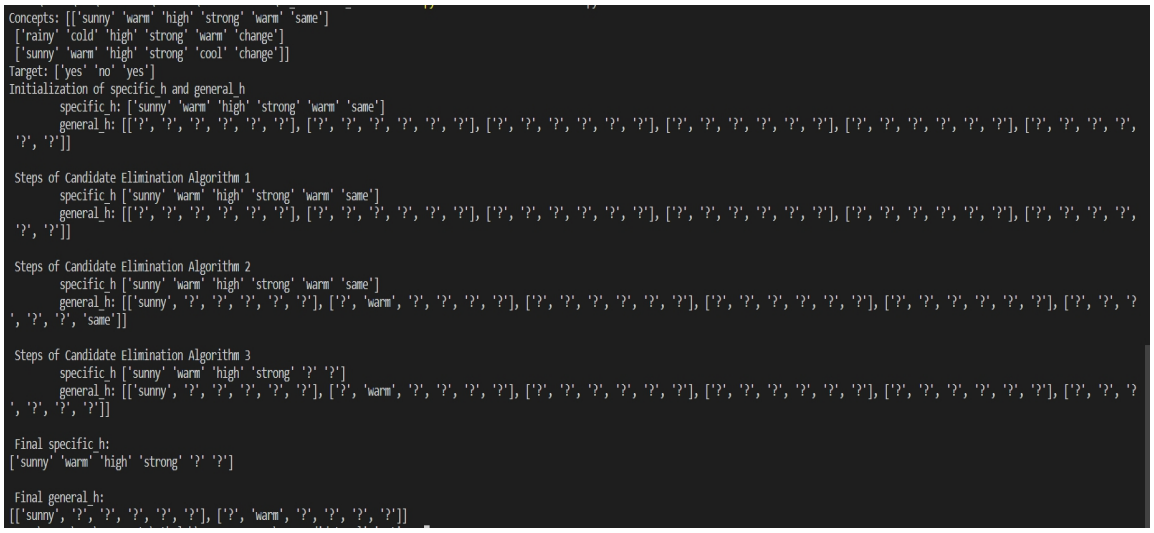
PROGRAM-2
Date-24/03/2021

For a given set of training data examples stored in a .CSV file, implement and demonstrate the Candidate-Elimination algorithm to output a description of the set of all hypotheses consistent with the training examples.

```
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:', concepts)
target = np.array(data.iloc[:,-1])
print('Target:', target)

def learn(concepts, target):
    print("Initialization of specific_h and general_h")

    specific_h = concepts[0].copy()
    print('\t specific_h:', specific_h)

    general_h = [["?" for i in range(len(specific_h))] for i in
range(len(specific_h))]
    print('\t general_h:', 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] ='?'
        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("\n Steps of Candidate Elimination Algorithm",i+1)
        print('\t specific_h', specific_h)
        print('\t general_h:', 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("\n Final specific_h:", s_final, sep="\n")
print("\n Final general_h:", g_final, sep="\n")
```

SCREENSHOTS



PROGRAM-3
Date-31/03/2021

Write a program to demonstrate the working of the decision tree based ID3 algorithm. Use an appropriate data set for building the decision tree and apply this knowledge to classify a new sample.

```
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)


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)
```

SCREENSHOTS

```
The decision tree for the dataset using ID3 algorithm is
 Outlook
   rain
     Wind
       strong
         no
       weak
         yes
   overcast
     yes
   sunny
     Humidity
       high
         no
       normal
         yes
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
```

PROGRAM-4
Date-21/04/2021

Write a program to implement the naive Bayesian classifier for a sample training data set stored as a .CSV file. Compute the accuracy of the classifier, considering few test data sets

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

df = pd.read_csv("dataset.csv")
feature_col_names = ['num_preg', 'glucose_conc', 'diastolic_bp',
'thickness', 'insulin', 'bmi', 'diab_pred', 'age']
predicted_class_names = ['diabetes']

X = df[feature_col_names].values
y = df[predicted_class_names].values

print(df.head)
xtrain,xtest,ytrain,ytest=train_test_split(X,y,test_size=0.40)

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

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

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

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)
```

SCREENSHOTS

```
<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
..        ...           ...           ...        ...      ...   ...
140         3           128            78          0        0  21.1
141         5           106            82         30        0  39.5
142         2           108            52         26       63  32.5
143        10           108            66          0        0  32.4
144         4           154            62         31      284  32.8

     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
..         ...  ...       ...
140      0.268   55         0
141      0.286   38         0
142      0.318   22         0
143      0.272   42         1
```

```
144      0.237   23          0

[145 rows x 9 columns]>

 the total number of Training Data : (87, 1)

 the total number of Test Data : (58, 1)

 Confusion matrix
[[28 10]
 [ 8 12]]

 Accuracy of the classifier is 0.6896551724137931

 The value of Precision 0.5454545454545454

 The value of Recall 0.6
Predicted Value for individual Test Data: [1]
```

```
 the total number of Training Data : (101, 1)

 the total number of Test Data : (44, 1)

 Confusion matrix
[[23  4]
 [ 6 11]]

 Accuracy of the classifier is 0.7727272727272727

 The value of Precision 0.7333333333333333

 The value of Recall 0.6470588235294118
Predicted Value for individual Test Data: [1]
```

```
 the total number of Training Data : (116, 1)

 the total number of Test Data : (29, 1)

 Confusion matrix
[[13  5]
 [ 3  8]]

 Accuracy of the classifier is 0.7241379310344828

 The value of Precision 0.6153846153846154

 The value of Recall 0.7272727272727273
Predicted Value for individual Test Data: [1]
```

PROGRAM-5
Date-28/04/2021
Write a program to construct a Bayesian network considering training data. Use this model to make predictions.

```
import numpy as np
import pandas as pd
import csv
from pgmpy.estimators import MaximumLikelihoodEstimator
from pgmpy.models import BayesianModel
from pgmpy.inference import VariableElimination

heartDisease = pd.read_csv('heart.csv')
heartDisease = heartDisease.replace('?',np.nan)

print('Sample instances from the dataset are given below')
print(heartDisease.head())

print('\n Attributes and datatypes')
print(heartDisease.dtypes)

model=BayesianModel([('age','heartdisease'),('sex','heartdisease'),('exang',
'heartdisease'),('cp','heartdisease'),('heartdisease','restecg'),('heartdisease','c
hol')])

print('\n Learning CPD using Maximum likelihood estimators')
model.fit(heartDisease,estimator=MaximumLikelihoodEstimator)

print('\n Inferencing with Bayesian Network:')
HeartDiseasetest_infer = VariableElimination(model)

print('\n 1.Probability of HeartDisease given evidence=restecg :1')
q1=HeartDiseasetest_infer.query(variables=['heartdisease'],evidence={'re
stecg':1})
print(q1)

print('\n 2.Probability of HeartDisease given evidence= cp:2 ')
q2=HeartDiseasetest_infer.query(variables=['heartdisease'],evidence={'cp
':2})
print(q2)
```

SCREENSHOTS

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               int64
thal             int64
heartdisease     int64
dtype: object
```

Learning CPD using Maximum likelihood estimators

```
Finding Elimination Order: : 100%|██████████| 5/5 [00:00<00:00, 720.37it/s]
Eliminating: age: 100%|██████████| 5/5 [00:00<00:00, 66.59it/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

```
Finding Elimination Order: : 100%|██████████| 5/5 [00:00<00:00, 839.60it/s]
Eliminating: age: 100%|██████████| 5/5 [00:00<00:00, 127.14it/s]
```

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