[Note: Dataset will be provided during External lab Exam]

Machine learning Lab Manual

2019-20

Program 1:Find S Algorithm:

```
f = open('prg1.csv','r')
length = len(f.readline().split(','))
hypo = ['0']*(length-1)
print('Intital Hypo = ',hypo)
f.close()
f = open('prg1.csv','r')
count =1
for line in f:
    lst = line.split(',')
    for i in range(length-1):
        if(lst[-1] == 'yes\n'):
            if(hypo[i]!='0' and lst[i]!=hypo[i]):
                hypo[i]='?'
            else:
                hypo[i] = lst[i]
    print('Hypo ',hypo)
print('final hypo ',hypo)
```

/*prg1.csv*/

```
sunny,warm,normal,strong,warm,same,yes
sunny,warm,high,strong,warm,same,yes
rainy,cold,high,strong,warm,change,no
sunny,warm,high,strong,cool,change,yes
```

Program 2:Candidate elimination Algorithm:

```
f = open('prg1.csv','r')
length = len(f.readline().split(',')) -1
f.close()
f = open('prg1.csv','r')
shypo = ['0']*(length)
ghypo =['?']*(length)
print('Intital Specific hypothesis',shypo)
count = 1
print('Intital General hypothesis',ghypo)
ghypo.clear()
for line in f:
    lst = line.split(',')
    for i in range(length):
        if(lst[-1] == 'yes\n'):
            if shypo[i]!='0' and shypo[i]!=lst[i]:
                shypo[i] ='?'
            else:
                shypo[i] = lst[i]
        elif (lst[-1] == 'no\n'):
            if '0' in shypo:
                temp_lst = ['?']*i
                temp_lst += [lst[i]]
                temp_lst += ['?'] * (length-1-i)
                ghypo.append(temp_lst)
            elif shypo[i]!='?' and shypo[i]!=lst[i]:
                temp_lst = ['?']*i
                temp_lst = temp_lst + [shypo[i]]
                temp_lst = temp_lst + ['?'] * (length-1-i)
                if(temp_lst not in ghypo):
                    ghypo.append(temp_lst)
    print('SHYPO ',count ," ",shypo)
    print('GHYPO ',count ," ",ghypo)
    count+=1
f_ghypo = list()
for i in range(len(ghypo)):
    for j in range(len(ghypo[i])):
        if(ghypo[i][j]!='?' and ghypo[i][j]==shypo[j]):
            f_ghypo.append(ghypo[i])
print(f_ghypo)
```

/*prg1.csv*/

```
sunny,warm,normal,strong,warm,same,yes
sunny,warm,high,strong,warm,same,yes
rainy,cold,high,strong,warm,change,no
sunny,warm,high,strong,cool,change,yes
```

Program 3:ID3 Decision Tree

```
import numpy as np

import pandas as pd


def entropy(target_col):

    val,counts = np.unique(target_col,return_counts = True)

    ent = sum( (-counts[i]/np.sum(counts)) * np.log2( counts[i]/np.sum(counts) ) for i in range(len(val)))

    return ent


def infoGain(data,features,target):

    te = entropy(data[target])

    val,counts = np.unique(data[features],return_counts = True)

    eg = sum((counts[i]/sum(counts)) * entropy(data[data[features] == val[i]][target] ) for i in range(len(val)))

    InfoGain = te-eg

    return InfoGain

def ID3(data,features,target,pnode):

    if len(np.unique(data[target])) == 1:

        return np.unique(data[target])[0]

    elif len(features) == 0:

        return pnode

    else:

        pnode = np.unique(data[target])[np.argmax(np.unique(data[target])[1])]

        IG = [infoGain(data,f,target) for f in features]

        index = np.argmax(IG)

        col = features[index]
    tree = {col:{}}
    features = [f for f in features if f!=col]
    for val in np.unique(data[col]):
        sub_data = data[data[col]==val].dropna()
        subtree = ID3(sub_data,features,target,pnode)
        tree[col][val] = subtree
    return tree

data = pd.read_csv('PlayTennis.csv')
testData = data.sample(frac = 0.1)
data.drop(testData.index,inplace = True)
print(data)
target = 'PlayTennis'
features = data.columns[data.columns!=target]
tree = ID3(data,features,target,None)
print (tree)
test = testData.to_dict('records')[0]
print(test,'=>', test['PlayTennis'])
```

```
/*PlayTennis.csv*/
Outlook,Temperature,Humidity,Wind,PlayTennis
Sunny,Hot,High,Weak,No
Sunny,Hot,High,Strong,No
Overcast,Hot,High,Weak,Yes
Rain,Mild,High,Weak,Yes
Rain,Cool,Normal,Weak,Yes
Rain,Cool,Normal,Strong,No
Overcast,Cool,Normal,Strong,Yes
Sunny,Mild,High,Weak,No
Sunny,Cool,Normal,Weak,Yes
Rain,Mild,Normal,Weak,Yes
Sunny,Mild,Normal,Strong,Yes
Overcast,Mild,High,Strong,Yes
Overcast,Hot,Normal,Weak,Yes
Rain,Mild,High,Strong,No
```

Program 4:Back Propagation Algorithm:

```
import numpy as np # numpy is commonly used to process number array
X = np.array([[2,9], [3,6], [4,8]]) # Features ( Hrs Slept, Hrs Studied)
y = np.array([[92], [86], [89]]) # Labels(Marks obtained)
X = X/np.amax(X,axis=0) # Normalize
y = y/100
def sigmoid(x):
    return 1/(1 + np.exp(-x))
def sigmoid_grad(x):
    return x * (1 - x)
# Variable initialization
epoch=1000 #Setting training iterations
eta =0.1 #Setting learning rate (eta)
input_neurons = 2 #number of features in data set
hidden_neurons = 3 #number of hidden layers neurons
output_neurons = 1 #number of neurons at output layer
# Weight and bias - Random initialization
wh=np.random.uniform(size=(input_neurons,hidden_neurons)) # 2x3
bh=np.random.uniform(size=(1,hidden_neurons)) # 1x3
wout=np.random.uniform(size=(hidden_neurons,output_neurons)) # 1x1
bout=np.random.uniform(size=(1,output_neurons))
for i in range(epoch):
    #Forward Propogation
    h_ip=np.dot(X,wh) + bh # Dot product + bias
    h_act = sigmoid(h_ip) # Activation function
    o_ip=np.dot(h_act,wout) + bout
    output = sigmoid(o_ip)
    # Error at Output layer
    Eo = y-output # Error at o/p
    outgrad = sigmoid_grad(output)
    d_output = Eo* outgrad # Errj=Oj(1-Oj)(Tj-Oj)
    # Error at Hidden later
    Eh = np.dot(d_output,wout.T) # .T means transpose
    hiddengrad = sigmoid_grad(h_act) # How much hidden layer wts contributed to error
    d_hidden = Eh * hiddengrad

    wout += np.dot(h_act.T,d_output) *eta # Dotproduct of nextlayererror and currentlayerop
    wh += np.dot(X.T,d_hidden) *eta

print("Normalized Input: \n" ,X)
print("Actual Output: \n" ,y)
print("Predicted Output: \n" ,output)
```

Program 5:Bayesian Classifier

```
import pandas as pd
mush = pd.read_csv('mushrooms.csv')
target = 'class'
classes = mush[target].unique()
features = mush.columns[mush.columns!=target]
testData = mush.sample(frac=0.3)
mush.drop(testData.index,inplace = True)
first ={}
fourth ={}
for x in classes:
    mushcl = mush[mush[target]==x][features]
    tot = len(mushcl)
    second={}
    for col in mushcl.columns:
        third={}
        for val,cnt in mushcl[col].value_counts().iteritems():
            prob = cnt/tot
            third[val]=prob
            second[col]=third
    first[x]=second
    fourth[x]=len(mushcl)/len(mush)
def proabs(params):
    proab={}
    for x in classes:
        calc = fourth[x]
        for col, val in params.iteritems():
            try:
                calc = first[x][col][val]
            except KeyError:
                calc =0
        proab[x]=calc
    return proab
def maxx(params):
    proab = proabs(params)
    maxcl =''; maxv=0
    for col,val in proab.items():
        if(val>maxv):
            maxv=val
            maxcl=col
    return maxcl

b=[]
for i in mush.index:
    b.append(   maxx(mush.loc[i,features]) == mush.loc[i,target] )
print(sum(b),'correct of',len(b))
print('Accuracy =',sum(b)/len(b))
b=[]
for i in testData.index:
    b.append(   maxx(testData.loc[i,features]) == testData.loc[i,target] )
print(sum(b),'correct of',len(b))
print('Accuracy =',sum(b)/len(b))
```

/*mushrooms.csv*/

```
class,cap-shape,cap-surface,cap-color,bruises,odor,gill-attachment,gill-spacing,gill-size,gill-color,stalk-shape,stalk-root,stalk-surface-above-ring,stalk-surface-below-ring,stalk-color-above-ring,stalk-color-below-ring,veil-type,veil-color,ring-number,ring-type,spore-print-color,population,habitat

p,x,s,n,t,p,f,c,n,k,e,e,s,s,w,w,p,w,o,p,k,s,u

e,x,s,y,t,a,f,c,b,k,e,c,s,s,w,w,p,w,o,p,n,n,g

e,b,s,w,t,l,f,c,b,n,e,c,s,s,w,w,p,w,o,p,n,n,m

p,x,y,w,t,p,f,c,n,n,e,e,s,s,w,w,p,w,o,p,k,s,u

e,x,s,g,f,n,f,w,b,k,t,e,s,s,w,w,p,w,o,e,n,a,g

e,x,y,y,t,a,f,c,b,n,e,c,s,s,w,w,p,w,o,p,k,n,g

e,b,s,w,t,a,f,c,b,g,e,c,s,s,w,w,p,w,o,p,k,n,m

e,b,y,w,t,l,f,c,b,n,e,c,s,s,w,w,p,w,o,p,n,s,m

p,x,y,w,t,p,f,c,n,p,e,e,s,s,w,w,p,w,o,p,k,v,g

e,b,s,y,t,a,f,c,b,g,e,c,s,s,w,w,p,w,o,p,k,s,m

e,x,y,y,t,l,f,c,b,g,e,c,s,s,w,w,p,w,o,p,n,n,g

e,x,y,y,t,a,f,c,b,n,e,c,s,s,w,w,p,w,o,p,k,s,m

e,b,s,y,t,a,f,c,b,w,e,c,s,s,w,w,p,w,o,p,n,s,g

p,x,y,w,t,p,f,c,n,k,e,e,s,s,w,w,p,w,o,p,n,v,u

e,x,f,n,f,n,f,w,b,n,t,e,s,f,w,w,p,w,o,e,k,a,g

e,s,f,g,f,n,f,c,n,k,e,e,s,s,w,w,p,w,o,p,n,y,u

e,f,f,w,f,n,f,w,b,k,t,e,s,s,w,w,p,w,o,e,n,a,g

p,x,s,n,t,p,f,c,n,n,e,e,s,s,w,w,p,w,o,p,k,s,g

p,x,y,w,t,p,f,c,n,n,e,e,s,s,w,w,p,w,o,p,n,s,u

p,x,s,n,t,p,f,c,n,k,e,e,s,s,w,w,p,w,o,p,n,s,u

e,b,s,y,t,a,f,c,b,k,e,c,s,s,w,w,p,w,o,p,n,s,m

p,x,y,n,t,p,f,c,n,n,e,e,s,s,w,w,p,w,o,p,n,v,g

e,b,y,y,t,l,f,c,b,k,e,c,s,s,w,w,p,w,o,p,n,s,m
```

```
e,b,y,w,t,a,f,c,b,w,e,c,s,s,w,w,p,w,o,p,n,n,m
e,b,s,w,t,l,f,c,b,g,e,c,s,s,w,w,p,w,o,p,k,s,m
p,f,s,w,t,p,f,c,n,n,e,e,s,s,w,w,p,w,o,p,n,v,g
e,x,y,y,t,a,f,c,b,n,e,c,s,s,w,w,p,w,o,p,n,n,m
e,x,y,w,t,l,f,c,b,w,e,c,s,s,w,w,p,w,o,p,n,n,m
e,f,f,n,f,n,f,c,n,k,e,e,s,s,w,w,p,w,o,p,k,y,u
e,x,s,y,t,a,f,w,n,n,t,b,s,s,w,w,p,w,o,p,n,v,d
e,b,s,y,t,l,f,c,b,g,e,c,s,s,w,w,p,w,o,p,n,n,m
p,x,y,w,t,p,f,c,n,k,e,e,s,s,w,w,p,w,o,p,n,s,u
e,x,y,y,t,l,f,c,b,n,e,c,s,s,w,w,p,w,o,p,n,n,m
e,x,y,n,t,l,f,c,b,p,e,r,s,y,w,w,p,w,o,p,n,y,p
e,b,y,y,t,l,f,c,b,n,e,c,s,s,w,w,p,w,o,p,n,s,m
e,x,f,y,t,l,f,w,n,w,t,b,s,s,w,w,p,w,o,p,n,v,d
e,s,f,g,f,n,f,c,n,k,e,e,s,s,w,w,p,w,o,p,k,v,u
p,x,y,n,t,p,f,c,n,w,e,e,s,s,w,w,p,w,o,p,n,s,u
e,x,f,y,t,a,f,w,n,p,t,b,s,s,w,w,p,w,o,p,n,v,d
e,b,s,y,t,l,f,c,b,k,e,c,s,s,w,w,p,w,o,p,k,s,m
e,b,y,y,t,a,f,c,b,n,e,c,s,s,w,w,p,w,o,p,n,s,g
e,x,y,y,t,l,f,c,b,n,e,r,s,y,w,w,p,w,o,p,k,y,p
e,x,f,n,f,n,f,c,n,g,e,e,s,s,w,w,p,w,o,p,k,y,u
p,x,y,w,t,p,f,c,n,p,e,e,s,s,w,w,p,w,o,p,n,v,g
e,x,s,y,t,a,f,c,b,w,e,c,s,s,w,w,p,w,o,p,k,n,m
e,x,y,w,t,a,f,c,b,n,e,c,s,s,w,w,p,w,o,p,n,n,g
e,x,y,y,t,l,f,c,b,k,e,c,s,s,w,w,p,w,o,p,k,s,m
e,x,s,w,t,l,f,c,b,w,e,c,s,s,w,w,p,w,o,p,n,n,m
e,x,y,y,t,l,f,c,b,n,e,r,s,y,w,w,p,w,o,p,n,s,p
e,f,y,y,t,l,f,c,b,w,e,r,s,y,w,w,p,w,o,p,k,s,p
e,x,y,n,t,a,f,c,b,w,e,r,s,y,w,w,p,w,o,p,k,s,g
```

```
e,x,s,w,t,l,f,c,b,k,e,c,s,s,w,w,p,w,o,p,k,s,g

e,b,s,w,t,l,f,c,b,k,e,c,s,s,w,w,p,w,o,p,n,n,m

p,x,y,n,t,p,f,c,n,k,e,e,s,s,w,w,p,w,o,p,n,v,u

p,x,s,w,t,p,f,c,n,k,e,e,s,s,w,w,p,w,o,p,k,v,u

e,b,y,y,t,a,f,c,b,w,e,c,s,s,w,w,p,w,o,p,k,s,m

e,f,f,g,f,n,f,w,b,n,t,e,s,s,w,w,p,w,o,e,n,a,g

e,b,s,w,t,a,f,c,b,w,e,c,s,s,w,w,p,w,o,p,n,n,g

e,x,s,y,t,l,f,c,b,k,e,c,s,s,w,w,p,w,o,p,k,n,g

e,x,y,n,t,a,f,c,b,p,e,r,s,y,w,w,p,w,o,p,k,y,p

e,s,f,g,f,n,f,c,n,k,e,e,s,s,w,w,p,w,o,p,n,v,u

e,b,y,y,t,a,f,c,b,k,e,c,s,s,w,w,p,w,o,p,n,s,m

e,b,s,y,t,l,f,c,b,g,e,c,s,s,w,w,p,w,o,p,n,s,m

e,b,y,y,t,l,f,c,b,g,e,c,s,s,w,w,p,w,o,p,n,n,m

e,b,y,w,t,l,f,c,b,n,e,c,s,s,w,w,p,w,o,p,n,s,g

e,f,s,n,f,n,f,w,b,k,t,e,s,s,w,w,p,w,o,e,k,a,g

e,x,s,w,t,l,f,c,b,n,e,c,s,s,w,w,p,w,o,p,k,s,g

e,f,y,y,t,a,f,c,b,w,e,r,s,y,w,w,p,w,o,p,n,s,g

e,x,y,y,t,a,f,c,b,w,e,c,s,s,w,w,p,w,o,p,k,n,g

e,x,f,g,f,n,f,c,n,p,e,e,s,s,w,w,p,w,o,p,n,v,u

e,f,f,y,t,l,f,w,n,p,t,b,s,s,w,w,p,w,o,p,n,v,d

e,b,y,w,t,l,f,c,b,g,e,c,s,s,w,w,p,w,o,p,n,s,m

e,f,f,y,t,l,f,w,n,w,t,b,s,s,w,w,p,w,o,p,n,v,d

e,x,y,n,t,a,f,c,b,p,e,r,s,y,w,w,p,w,o,p,k,s,p

e,b,s,y,t,a,f,c,b,k,e,c,s,s,w,w,p,w,o,p,k,s,g

e,f,s,y,t,l,f,w,n,p,t,b,s,s,w,w,p,w,o,p,n,v,d

e,x,s,w,t,l,f,w,n,n,t,b,s,s,w,w,p,w,o,p,u,v,d

e,f,y,n,t,l,f,c,b,p,e,r,s,y,w,w,p,w,o,p,n,y,p

p,x,y,n,t,p,f,c,n,w,e,e,s,s,w,w,p,w,o,p,n,v,u
```

e,f,y,n,t,a,f,c,b,n,e,r,s,y,w,w,p,w,o,p,n,y,g

e,x,s,n,f,n,f,w,b,k,t,e,f,s,w,w,p,w,o,e,n,s,g

p,x,y,w,t,p,f,c,n,w,e,e,s,s,w,w,p,w,o,p,k,s,g

e,f,f,g,f,n,f,c,n,n,e,e,s,s,w,w,p,w,o,p,n,y,u

e,x,f,g,f,n,f,w,b,n,t,e,s,s,w,w,p,w,o,e,n,s,g

e,x,y,y,t,l,f,c,b,w,e,r,s,y,w,w,p,w,o,p,k,s,g

e,x,s,n,f,n,f,w,b,k,t,e,s,s,w,w,p,w,o,e,k,s,g

e,b,s,w,t,a,f,c,b,w,e,c,s,s,w,w,p,w,o,p,k,s,g

e,x,s,w,t,l,f,c,b,n,e,c,s,s,w,w,p,w,o,p,n,s,g

e,f,y,n,t,l,f,c,b,w,e,r,s,y,w,w,p,w,o,p,k,y,g

e,s,f,n,f,n,f,c,n,n,e,e,s,s,w,w,p,w,o,p,n,v,u

e,x,f,n,f,n,f,c,n,n,e,e,s,s,w,w,p,w,o,p,n,y,u

e,b,s,w,t,l,f,c,b,k,e,c,s,s,w,w,p,w,o,p,k,s,g

e,x,y,y,t,a,f,c,b,g,e,c,s,s,w,w,p,w,o,p,k,s,g

e,x,y,y,t,l,f,c,b,g,e,c,s,s,w,w,p,w,o,p,k,n,m

e,x,s,n,f,n,f,w,b,n,t,e,s,s,w,w,p,w,o,e,n,a,g

e,x,s,w,t,a,f,c,b,g,e,c,s,s,w,w,p,w,o,p,n,s,g

e,f,y,n,t,l,f,c,b,p,e,r,s,y,w,w,p,w,o,p,n,s,g

e,x,s,y,t,a,f,c,b,n,e,c,s,s,w,w,p,w,o,p,k,n,g

e,b,s,w,t,a,f,c,b,g,e,c,s,s,w,w,p,w,o,p,n,s,g

e,x,y,w,t,a,f,c,b,g,e,c,s,s,w,w,p,w,o,p,k,s,g

e,x,f,n,f,n,f,w,b,p,t,e,f,s,w,w,p,w,o,e,k,s,g

e,b,s,y,t,l,f,c,b,n,e,c,s,s,w,w,p,w,o,p,k,n,g

e,f,y,y,t,l,f,c,b,w,e,r,s,y,w,w,p,w,o,p,n,s,g

e,x,y,y,t,a,f,c,b,n,e,r,s,y,w,w,p,w,o,p,k,y,p

e,b,y,w,t,l,f,c,b,g,e,c,s,s,w,w,p,w,o,p,n,n,g

e,x,y,y,t,a,f,c,b,n,e,c,s,s,w,w,p,w,o,p,k,n,m

e,x,y,y,t,a,f,c,b,w,e,r,s,y,w,w,p,w,o,p,n,y,g

```
e,b,y,w,t,l,f,c,b,n,e,c,s,s,w,w,p,w,o,p,k,s,m
e,b,y,w,t,a,f,c,b,g,e,c,s,s,w,w,p,w,o,p,n,s,m
e,x,s,y,t,a,f,c,b,k,e,c,s,s,w,w,p,w,o,p,k,n,m
e,x,s,y,t,l,f,c,b,w,e,c,s,s,w,w,p,w,o,p,k,n,g
e,s,f,g,f,n,f,c,n,g,e,e,s,s,w,w,p,w,o,p,k,y,u
e,x,f,w,t,a,f,w,n,w,t,b,s,s,w,w,p,w,o,p,u,v,d
e,x,s,y,t,a,f,c,b,n,e,c,s,s,w,w,p,w,o,p,k,n,m
p,x,y,w,t,p,f,c,n,n,e,e,s,s,w,w,p,w,o,p,n,v,u
e,x,y,y,t,l,f,c,b,p,e,r,s,y,w,w,p,w,o,p,n,s,g
e,s,f,g,f,n,f,c,n,p,e,e,s,s,w,w,p,w,o,p,n,y,u
e,x,y,y,t,l,f,c,b,w,e,r,s,y,w,w,p,w,o,p,k,y,g
e,x,s,y,t,l,f,w,n,p,t,b,s,s,w,w,p,w,o,p,u,v,d
e,s,f,n,f,n,f,c,n,k,e,e,s,s,w,w,p,w,o,p,n,y,u
p,x,s,w,t,p,f,c,n,k,e,e,s,s,w,w,p,w,o,p,k,v,g
e,x,y,w,t,a,f,c,b,g,e,c,s,s,w,w,p,w,o,p,n,n,m
p,f,y,n,t,p,f,c,n,p,e,e,s,s,w,w,p,w,o,p,k,v,g
e,f,s,g,f,n,f,w,b,k,t,e,s,s,w,w,p,w,o,e,n,a,g
e,x,s,y,t,l,f,c,b,g,e,c,s,s,w,w,p,w,o,p,n,s,m
e,x,s,w,f,n,f,w,b,n,t,e,s,f,w,w,p,w,o,e,k,s,g
e,b,s,y,t,a,f,c,b,w,e,c,s,s,w,w,p,w,o,p,n,n,g
e,f,f,g,f,n,f,w,b,h,t,e,s,s,w,w,p,w,o,e,n,a,g
e,x,s,w,t,l,f,c,b,n,e,c,s,s,w,w,p,w,o,p,k,n,g
e,b,s,w,t,l,f,c,b,n,e,c,s,s,w,w,p,w,o,p,n,s,m
e,b,s,w,t,l,f,c,b,w,e,c,s,s,w,w,p,w,o,p,n,s,g
e,b,y,w,t,l,f,c,b,w,e,c,s,s,w,w,p,w,o,p,n,s,m
e,f,s,w,t,l,f,w,n,w,t,b,s,s,w,w,p,w,o,p,u,v,d
e,x,y,y,t,l,f,c,b,g,e,c,s,s,w,w,p,w,o,p,k,s,m
e,f,s,w,t,a,f,w,n,p,t,b,s,s,w,w,p,w,o,p,n,v,d
```

```
p,x,y,w,t,p,f,c,n,w,e,e,s,s,w,w,p,w,o,p,n,v,u

e,f,f,w,t,l,f,w,n,w,t,b,s,s,w,w,p,w,o,p,n,v,d

e,x,y,y,t,a,f,c,b,n,e,c,s,s,w,w,p,w,o,p,n,s,g

p,x,s,n,t,p,f,c,n,p,e,e,s,s,w,w,p,w,o,p,n,v,g

e,b,s,y,t,l,f,c,b,w,e,c,s,s,w,w,p,w,o,p,n,n,g

e,x,y,n,t,a,f,c,b,w,e,r,s,y,w,w,p,w,o,p,k,y,p

e,b,y,y,t,l,f,c,b,g,e,c,s,s,w,w,p,w,o,p,k,n,m

e,s,f,n,f,n,f,c,n,k,e,e,s,s,w,w,p,w,o,p,n,v,u

e,f,y,n,t,a,f,c,b,w,e,r,s,y,w,w,p,w,o,p,k,y,p

e,x,y,y,t,a,f,c,b,k,e,c,s,s,w,w,p,w,o,p,k,n,g

e,x,f,g,f,n,f,w,b,k,t,e,f,f,w,w,p,w,o,e,k,s,g

e,f,f,w,f,n,f,w,b,k,t,e,s,f,w,w,p,w,o,e,n,a,g

e,x,y,y,t,l,f,c,b,w,e,c,s,s,w,w,p,w,o,p,n,n,m

e,b,s,y,t,l,f,c,b,k,e,c,s,s,w,w,p,w,o,p,k,n,g

e,b,y,w,t,a,f,c,b,g,e,c,s,s,w,w,p,w,o,p,k,n,m

e,x,y,w,t,a,f,c,b,w,e,c,s,s,w,w,p,w,o,p,n,s,g

e,x,s,n,f,n,f,w,b,p,t,e,f,s,w,w,p,w,o,e,n,a,g

e,x,y,w,t,l,f,c,b,g,e,c,s,s,w,w,p,w,o,p,n,s,g

e,s,f,n,f,n,f,c,n,k,e,e,s,s,w,w,p,w,o,p,k,v,u

e,x,s,w,t,a,f,w,n,w,t,b,s,s,w,w,p,w,o,p,u,v,d

e,x,y,n,t,l,f,c,b,w,e,r,s,y,w,w,p,w,o,p,k,s,g

e,b,y,y,t,a,f,c,b,k,e,c,s,s,w,w,p,w,o,p,n,n,g

e,x,y,w,t,a,f,c,b,k,e,c,s,s,w,w,p,w,o,p,n,n,g

e,b,y,w,t,a,f,c,b,n,e,c,s,s,w,w,p,w,o,p,k,s,m

e,b,s,y,t,a,f,c,b,g,e,c,s,s,w,w,p,w,o,p,k,s,g

e,b,s,y,t,l,f,c,b,g,e,c,s,s,w,w,p,w,o,p,k,s,m

e,b,y,y,t,a,f,c,b,n,e,c,s,s,w,w,p,w,o,p,k,n,g

e,x,f,n,f,n,f,c,n,k,e,e,s,s,w,w,p,w,o,p,n,y,u
```

```
e,f,y,n,t,l,f,c,b,n,e,r,s,y,w,w,p,w,o,p,n,y,g

e,x,y,w,t,a,f,c,b,k,e,c,s,s,w,w,p,w,o,p,n,s,g

e,f,y,y,t,l,f,c,b,w,e,r,s,y,w,w,p,w,o,p,n,y,p

e,b,s,w,t,a,f,c,b,w,e,c,s,s,w,w,p,w,o,p,n,s,g

e,b,s,w,t,a,f,c,b,w,e,c,s,s,w,w,p,w,o,p,k,s,m

e,x,y,n,t,l,f,c,b,w,e,r,s,y,w,w,p,w,o,p,k,y,g

e,b,s,w,t,a,f,c,b,k,e,c,s,s,w,w,p,w,o,p,k,s,g

e,x,f,g,f,n,f,c,n,g,e,e,s,s,w,w,p,w,o,p,n,y,u

e,b,s,y,t,l,f,c,b,k,e,c,s,s,w,w,p,w,o,p,n,s,g

e,x,f,y,t,l,f,w,n,n,t,b,s,s,w,w,p,w,o,p,u,v,d

e,b,y,y,t,a,f,c,b,w,e,c,s,s,w,w,p,w,o,p,k,s,g

e,f,y,y,t,l,f,c,b,p,e,r,s,y,w,w,p,w,o,p,n,s,g

e,b,y,w,t,l,f,c,b,w,e,c,s,s,w,w,p,w,o,p,k,n,m

e,b,y,w,t,a,f,c,b,k,e,c,s,s,w,w,p,w,o,p,k,n,m

e,b,y,y,t,a,f,c,b,g,e,c,s,s,w,w,p,w,o,p,n,s,g

e,x,y,y,t,l,f,c,b,g,e,c,s,s,w,w,p,w,o,p,n,n,m

e,b,s,y,t,l,f,c,b,g,e,c,s,s,w,w,p,w,o,p,n,n,g

p,x,y,w,t,p,f,c,n,p,e,e,s,s,w,w,p,w,o,p,n,v,u

e,s,f,n,f,n,f,c,n,g,e,e,s,s,w,w,p,w,o,p,n,y,u

e,f,f,n,f,n,f,c,n,g,e,e,s,s,w,w,p,w,o,p,k,v,u

e,x,s,y,t,a,f,c,b,g,e,c,s,s,w,w,p,w,o,p,n,s,m

e,f,y,n,t,a,f,c,b,p,e,r,s,y,w,w,p,w,o,p,k,s,p

p,x,y,w,t,p,f,c,n,k,e,e,s,s,w,w,p,w,o,p,k,s,g

e,b,s,w,t,l,f,c,b,g,e,c,s,s,w,w,p,w,o,p,n,s,m

e,f,f,g,f,n,f,c,n,p,e,e,s,s,w,w,p,w,o,p,k,v,u

e,b,y,y,t,l,f,c,b,g,e,c,s,s,w,w,p,w,o,p,n,n,g

e,x,y,n,t,a,f,c,b,w,e,r,s,y,w,w,p,w,o,p,n,y,p

e,x,f,w,f,n,f,w,b,p,t,e,s,f,w,w,p,w,o,e,k,s,g
```

```
e,x,s,w,t,l,f,w,n,w,t,b,s,s,w,w,p,w,o,p,n,v,d
e,b,s,w,t,l,f,c,b,w,e,c,s,s,w,w,p,w,o,p,n,s,m
e,f,s,y,t,a,f,w,n,w,t,b,s,s,w,w,p,w,o,p,n,v,d
e,x,s,y,t,a,f,c,b,k,e,c,s,s,w,w,p,w,o,p,n,n,m
e,f,f,g,f,n,f,c,n,g,e,e,s,s,w,w,p,w,o,p,n,y,u
e,b,s,y,t,a,f,c,b,n,e,c,s,s,w,w,p,w,o,p,k,s,g
e,x,s,w,t,l,f,c,b,g,e,c,s,s,w,w,p,w,o,p,n,n,m
e,x,y,w,t,l,f,c,b,n,e,c,s,s,w,w,p,w,o,p,n,s,m
```

/*Data set is larger*/

Program 6:Bayesian Classifier for Text Classifier

```
import pandas as pd
msg=pd.read_csv('naive.csv',names=['message','label'])
print('The dimensions of the dataset',msg.shape)
msg['labelnum']=msg.label.map({'pos':1,'neg':0})
X=msg.message
y=msg.labelnum
print(X)
print(y)
#splitting the dataset into train and test data
from sklearn.model_selection import train_test_split
xtrain,xtest,ytrain,ytest=train_test_split(X,y)
print(xtest.shape)
print(xtrain.shape)
print(ytest.shape)
print(ytrain.shape)
#output of count vectoriser is a sparse matrix
from sklearn.feature_extraction.text import CountVectorizer
count_vect = CountVectorizer()
xtrain_dtm = count_vect.fit_transform(xtrain)
xtest_dtm=count_vect.transform(xtest)
print(count_vect.get_feature_names())

df=pd.DataFrame(xtrain_dtm.toarray(),columns=count_vect.get_feature_names())
print(df)#tabular representation
print(xtrain_dtm) #sparse matrix representation
# Training Naive Bayes (NB) classifier on training data.
```

```
from sklearn.naive_bayes import MultinomialNB
clf = MultinomialNB().fit(xtrain_dtm,ytrain)
predicted = clf.predict(xtest_dtm)
#printing accuracy metrics
from sklearn import metrics
print('Accuracy metrics')
print('Accuracy of the classifer is',metrics.accuracy_score(ytest,predicted))
print('Confusion matrix')
print(metrics.confusion_matrix(ytest,predicted))
print('Recall and Precison ')
print(metrics.recall_score(ytest,predicted))
print(metrics.precision_score(ytest,predicted))
```

/*naïve.csv*/

```
I love this sandwich,pos
This is an amazing place,pos
I feel very good about these beers,pos
This is my best work,pos
What an awesome view,pos
I do not like this restaurant,neg
I am tired of this stuff,neg
I can't deal with this,neg
He is my sworn enemy,neg
My boss is horrible,neg
This is an awesome place,pos
I do not like the taste of this juice,neg
I love to dance,pos
I am sick and tired of this place,neg
What a great holiday,pos
That is a bad locality to stay,neg
We will have good fun tomorrow,pos
I went to my enemy's house today,neg
```

Program 7:Bayesian Network For Heart Diseases

```
import pandas as pd

from pgmpy.estimators import BayesianEstimator

from pgmpy.models import BayesianModel

from pgmpy.inference import VariableElimination

f=open('data7_name.csv','r')

attributes= f.readline().split(',')

heartDisease=pd.read_csv('data7.csv',names=attributes)

print("\nAttributes and datatypes")

print(heartDisease.dtypes)

model=BayesianModel([('age','trestbps'),('age','fbs'),('sex','trestbps'),('exang','trestbps'),('trestbps','heartdisease'),('fbs','heartdisease')])

model.fit(heartDisease,BayesianEstimator)

HeartDisease_infer=VariableElimination(model)

print("\n 1. Probability heart disease given age=28")

q=HeartDisease_infer.query(['heartdisease'],{'age':28})

print(q['heartdisease'])

print("\n 2. Probability of heart disease for male")

q=HeartDisease_infer.query(['heartdisease'],{'sex':1})

print(q['heartdisease'])
```

/*data_7_name.csv*/

```
age,sex,cp,trestbps,chol,fbs,restecg,thalach,exang,oldpeak,slope,ca,thal,heartdisease
```

/*data7.csv*/

```
63,1,1,145,233,1,2,150,0,2.3,3,0,6,0
67,1,4,160,286,0,2,108,1,1.5,2,3,3,2
67,1,4,120,229,0,2,129,1,2.6,2,2,7,1
37,1,3,130,250,0,0,187,0,3.5,3,0,3,0
41,0,2,130,204,0,2,172,0,1.4,1,0,3,0
56,1,2,120,236,0,0,178,0,0.8,1,0,3,0
62,0,4,140,268,0,2,160,0,3.6,3,2,3,3
57,0,4,120,354,0,0,163,1,0.6,1,0,3,0
63,1,4,130,254,0,2,147,0,1.4,2,1,7,2
53,1,4,140,203,1,2,155,1,3.1,3,0,7,1
57,1,4,140,192,0,0,148,0,0.4,2,0,6,0
56,0,2,140,294,0,2,153,0,1.3,2,0,3,0
56,1,3,130,256,1,2,142,1,0.6,2,1,6,2
44,1,2,120,263,0,0,173,0,0,1,0,7,0
52,1,3,172,199,1,0,162,0,0.5,1,0,7,0
57,1,3,150,168,0,0,174,0,1.6,1,0,3,0
48,1,2,110,229,0,0,168,0,1,3,0,7,1
54,1,4,140,239,0,0,160,0,1.2,1,0,3,0
48,0,3,130,275,0,0,139,0,0.2,1,0,3,0
49,1,2,130,266,0,0,171,0,0.6,1,0,3,0
64,1,1,110,211,0,2,144,1,1.8,2,0,3,0
58,0,1,150,283,1,2,162,0,1,1,0,3,0
58,1,2,120,284,0,2,160,0,1.8,2,0,3,1
```

```
58,1,3,132,224,0,2,173,0,3.2,1,2,7,3

60,1,4,130,206,0,2,132,1,2.4,2,2,7,4

50,0,3,120,219,0,0,158,0,1.6,2,0,3,0

58,0,3,120,340,0,0,172,0,0,1,0,3,0

66,0,1,150,226,0,0,114,0,2.6,3,0,3,0

43,1,4,150,247,0,0,171,0,1.5,1,0,3,0

40,1,4,110,167,0,2,114,1,2,2,0,7,3

69,0,1,140,239,0,0,151,0,1.8,1,2,3,0

60,1,4,117,230,1,0,160,1,1.4,1,2,7,2

64,1,3,140,335,0,0,158,0,0,1,0,3,1

59,1,4,135,234,0,0,161,0,0.5,2,0,7,0

44,1,3,130,233,0,0,179,1,0.4,1,0,3,0

42,1,4,140,226,0,0,178,0,0,1,0,3,0

43,1,4,120,177,0,2,120,1,2.5,2,0,7,3

57,1,4,150,276,0,2,112,1,0.6,2,1,6,1

55,1,4,132,353,0,0,132,1,1.2,2,1,7,3

61,1,3,150,243,1,0,137,1,1,2,0,3,0

65,0,4,150,225,0,2,114,0,1,2,3,7,4

40,1,1,140,199,0,0,178,1,1.4,1,0,7,0

71,0,2,160,302,0,0,162,0,0.4,1,2,3,0

59,1,3,150,212,1,0,157,0,1.6,1,0,3,0

61,0,4,130,330,0,2,169,0,0,1,0,3,1

58,1,3,112,230,0,2,165,0,2.5,2,1,7,4

51,1,3,110,175,0,0,123,0,0.6,1,0,3,0

50,1,4,150,243,0,2,128,0,2.6,2,0,7,4

65,0,3,140,417,1,2,157,0,0.8,1,1,3,0

53,1,3,130,197,1,2,152,0,1.2,3,0,3,0

41,0,2,105,198,0,0,168,0,0,1,1,3,0
```

```
65,1,4,120,177,0,0,140,0,0.4,1,0,7,0

44,1,4,112,290,0,2,153,0,0,1,1,3,2

44,1,2,130,219,0,2,188,0,0,1,0,3,0

60,1,4,130,253,0,0,144,1,1.4,1,1,7,1

54,1,4,124,266,0,2,109,1,2.2,2,1,7,1

50,1,3,140,233,0,0,163,0,0.6,2,1,7,1

41,1,4,110,172,0,2,158,0,0,1,0,7,1

54,1,3,125,273,0,2,152,0,0.5,3,1,3,0

51,1,1,125,213,0,2,125,1,1.4,1,1,3,0

51,0,4,130,305,0,0,142,1,1.2,2,0,7,2

46,0,3,142,177,0,2,160,1,1.4,3,0,3,0

58,1,4,128,216,0,2,131,1,2.2,2,3,7,1

54,0,3,135,304,1,0,170,0,0,1,0,3,0

54,1,4,120,188,0,0,113,0,1.4,2,1,7,2

60,1,4,145,282,0,2,142,1,2.8,2,2,7,2

60,1,3,140,185,0,2,155,0,3,2,0,3,1

54,1,3,150,232,0,2,165,0,1.6,1,0,7,0

59,1,4,170,326,0,2,140,1,3.4,3,0,7,2

46,1,3,150,231,0,0,147,0,3.6,2,0,3,1

65,0,3,155,269,0,0,148,0,0.8,1,0,3,0

67,1,4,125,254,1,0,163,0,0.2,2,2,7,3

62,1,4,120,267,0,0,99,1,1.8,2,2,7,1

65,1,4,110,248,0,2,158,0,0.6,1,2,6,1

44,1,4,110,197,0,2,177,0,0,1,1,3,1

65,0,3,160,360,0,2,151,0,0.8,1,0,3,0

60,1,4,125,258,0,2,141,1,2.8,2,1,7,1

51,0,3,140,308,0,2,142,0,1.5,1,1,3,0
```

/*Sample dataset <Dataset is too large>*/

Program 8:K-Means Algorithm

```
from sklearn.cluster import KMeans
from sklearn.mixture import GaussianMixture
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
data = pd.read_csv('8-kmeansdata.csv')
f1 =data['Distance_Feature']
f2=data['Speeding_Feature']
X =np.array(list(zip(f1,f2)))
plt.scatter(f1,f2,color='black')
plt.show()
kmeans = KMeans(3).fit(X)
labels = kmeans.predict(X)
plt.scatter(f1,f2,c=labels)
plt.show()
gm = GaussianMixture(3).fit(X)
labels = gm.predict(X)
plt.scatter(f1,f2,c=labels)
plt.show()
```

/*8-kmeansdata.csv*/

```
Driver_ID,Distance_Feature,Speeding_Feature
3423311935,71.24,28
3423313212,52.53,25
3423313724,64.54,27
3423311373,55.69,22
3423310999,54.58,25
3423313857,41.91,10
3423312432,58.64,20
3423311434,52.02,8
3423311328,31.25,34
3423312488,44.31,19
3423311254,49.35,40
3423312943,58.07,45
3423312536,44.22,22
3423311542,55.73,19
3423312176,46.63,43
3423314176,52.97,32
3423314202,46.25,35
3423311346,51.55,27
3423310666,57.05,26
3423313527,58.45,30
3423312182,43.42,23
3423313590,55.68,37
3423312268,55.15,18
```

Program 9:kNN Algorithm:

```
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn import datasets
iris=datasets.load_iris()
print("Iris Data set loaded...")
x_train, x_test, y_train, y_test = train_test_split(iris.data,iris.target)
classifier = KNeighborsClassifier(3).fit(x_train, y_train)
y_pred=classifier.predict(x_test)
print("Results of Classification using K-nn with K=1 ")
for r in range(0,len(x_test)):
    print(" Sample:", str(x_test[r]), " Actual-label:", str(y_test[r]), " Predicted-label:",str(y_pred[r]))
print("Classification Accuracy :" , classifier.score(x_test,y_test));
```

Program 10:Linear Regression:

```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
def localWeigh(point,X,ymat,k):
    m,n = np.shape(X)
    weights = np.mat(np.eye(m))
    for i in range(m):
        diff = point - X[i]
        weights[i,i] = np.exp(diff*diff.T/(-2.0*k**2))
    W = (X.T *(weights*X)).I * (X.T*(weights*ymat.T))
    return W
def localWeightReg(X,ymat,k):
    m,n = np.shape(X)
    ypred = np.zeros(m)
    for i in range(m):
        ypred[i] = X[i] * localWeigh(X[i],X,ymat,k)
    return ypred
def plott(X,pred):
    sortIndex = X[:,1].argsort(0)
    xsort = X[sortIndex][:,0][:,1]
    ysort = pred[sortIndex]
    plt.scatter(x,y,color='green')
    plt.plot(xsort,ysort,color="red",linewidth=5)
    plt.xlabel('Total bill')
    plt.ylabel('Tips')
    plt.show()
```

```
data = pd.read_csv('data10.csv')
x=data['total_bill']
y = data['tip']
xmat = np.mat(x)
ymat = np.mat(y)
size = np.shape(xmat)[1]
ones = np.mat(np.ones(size))
X=np.hstack((ones.T,xmat.T))
pred = localWeightReg(X,ymat,3)
plott(X,pred)
```

/*data10.csv*/

```
total_bill,tip,sex,smoker,day,time,size,fraction
16.99,1.01,Female,No,Sun,Dinner,2,0.05944673337257211
10.34,1.66,Male,No,Sun,Dinner,3,0.16054158607350097
21.01,3.5,Male,No,Sun,Dinner,3,0.16658733936220846
23.68,3.31,Male,No,Sun,Dinner,2,0.1397804054054054
24.59,3.61,Female,No,Sun,Dinner,4,0.14680764538430255
25.29,4.71,Male,No,Sun,Dinner,4,0.18623962040332148
8.77,2.0,Male,No,Sun,Dinner,2,0.22805017103762829
26.88,3.12,Male,No,Sun,Dinner,4,0.11607142857142858
15.04,1.96,Male,No,Sun,Dinner,2,0.13031914893617022
14.78,3.23,Male,No,Sun,Dinner,2,0.2185385656292287
10.27,1.71,Male,No,Sun,Dinner,2,0.1665043816942551
35.26,5.0,Female,No,Sun,Dinner,4,0.14180374361883155
15.42,1.57,Male,No,Sun,Dinner,2,0.10181582360570687
18.43,3.0,Male,No,Sun,Dinner,4,0.16277807921866522
14.83,3.02,Female,No,Sun,Dinner,2,0.20364126770060686
21.58,3.92,Male,No,Sun,Dinner,2,0.18164967562557924
10.33,1.67,Female,No,Sun,Dinner,3,0.1616650532429816
16.29,3.71,Male,No,Sun,Dinner,3,0.22774708410067526
16.97,3.5,Female,No,Sun,Dinner,3,0.20624631703005306
20.65,3.35,Male,No,Sat,Dinner,3,0.16222760290556903
17.92,4.08,Male,No,Sat,Dinner,2,0.22767857142857142
20.29,2.75,Female,No,Sat,Dinner,2,0.13553474618038444
15.77,2.23,Female,No,Sat,Dinner,2,0.14140773620798985
39.42,7.58,Male,No,Sat,Dinner,4,0.19228817858954844
19.82,3.18,Male,No,Sat,Dinner,2,0.16044399596367306
```

```
17.81,2.34,Male,No,Sat,Dinner,4,0.13138686131386862
13.37,2.0,Male,No,Sat,Dinner,2,0.14958863126402394
12.69,2.0,Male,No,Sat,Dinner,2,0.15760441292356187
21.7,4.3,Male,No,Sat,Dinner,2,0.19815668202764977
19.65,3.0,Female,No,Sat,Dinner,2,0.15267175572519084
9.55,1.45,Male,No,Sat,Dinner,2,0.1518324607329843
18.35,2.5,Male,No,Sat,Dinner,4,0.13623978201634876
15.06,3.0,Female,No,Sat,Dinner,2,0.199203187250996
20.69,2.45,Female,No,Sat,Dinner,4,0.11841469308844853
17.78,3.27,Male,No,Sat,Dinner,2,0.1839145106861642
24.06,3.6,Male,No,Sat,Dinner,3,0.14962593516209477
16.31,2.0,Male,No,Sat,Dinner,3,0.12262415695892091
16.93,3.07,Female,No,Sat,Dinner,3,0.18133490844665446
18.69,2.31,Male,No,Sat,Dinner,3,0.12359550561797752
31.27,5.0,Male,No,Sat,Dinner,3,0.1598976654940838
16.04,2.24,Male,No,Sat,Dinner,3,0.13965087281795513
17.46,2.54,Male,No,Sun,Dinner,2,0.14547537227949597
13.94,3.06,Male,No,Sun,Dinner,2,0.21951219512195122
9.68,1.32,Male,No,Sun,Dinner,2,0.13636363636363638
30.4,5.6,Male,No,Sun,Dinner,4,0.18421052631578946
18.29,3.0,Male,No,Sun,Dinner,2,0.16402405686167304
22.23,5.0,Male,No,Sun,Dinner,2,0.22492127755285649
32.4,6.0,Male,No,Sun,Dinner,4,0.1851851851851852
28.55,2.05,Male,No,Sun,Dinner,3,0.07180385288966724
18.04,3.0,Male,No,Sun,Dinner,2,0.16629711751662973
12.54,2.5,Male,No,Sun,Dinner,2,0.19936204146730463
10.29,2.6,Female,No,Sun,Dinner,2,0.2526724975704568
34.81,5.2,Female,No,Sun,Dinner,4,0.14938236139040506
```

```
9.94,1.56,Male,No,Sun,Dinner,2,0.15694164989939638

25.56,4.34,Male,No,Sun,Dinner,4,0.1697965571205008

19.49,3.51,Male,No,Sun,Dinner,2,0.18009235505387378

38.01,3.0,Male,Yes,Sat,Dinner,4,0.07892659826361484

26.41,1.5,Female,No,Sat,Dinner,2,0.05679666792881484

11.24,1.76,Male,Yes,Sat,Dinner,2,0.15658362989323843

48.27,6.73,Male,No,Sat,Dinner,4,0.13942407292314066

20.29,3.21,Male,Yes,Sat,Dinner,2,0.1582060128141942

13.81,2.0,Male,Yes,Sat,Dinner,2,0.1448225923244026

11.02,1.98,Male,Yes,Sat,Dinner,2,0.17967332123411978

18.29,3.76,Male,Yes,Sat,Dinner,4,0.20557681793329688

17.59,2.64,Male,No,Sat,Dinner,3,0.15008527572484368

20.08,3.15,Male,No,Sat,Dinner,3,0.15687250996015936

16.45,2.47,Female,No,Sat,Dinner,2,0.1501519756838906

3.07,1.0,Female,Yes,Sat,Dinner,1,0.32573289902280134

20.23,2.01,Male,No,Sat,Dinner,2,0.09935739001482945

15.01,2.09,Male,Yes,Sat,Dinner,2,0.13924050632911392

12.02,1.97,Male,No,Sat,Dinner,2,0.1638935108153078

17.07,3.0,Female,No,Sat,Dinner,3,0.1757469244288225

26.86,3.14,Female,Yes,Sat,Dinner,2,0.11690245718540582

25.28,5.0,Female,Yes,Sat,Dinner,2,0.19778481012658228

14.73,2.2,Female,No,Sat,Dinner,2,0.14935505770536323

10.51,1.25,Male,No,Sat,Dinner,2,0.11893434823977164

17.92,3.08,Male,Yes,Sat,Dinner,2,0.171875

27.2,4.0,Male,No,Thur,Lunch,4,0.14705882352941177

22.76,3.0,Male,No,Thur,Lunch,2,0.13181019332161686

17.29,2.71,Male,No,Thur,Lunch,2,0.156737998843262

19.44,3.0,Male,Yes,Thur,Lunch,2,0.15432098765432098
```

```
16.66,3.4,Male,No,Thur,Lunch,2,0.20408163265306123
10.07,1.83,Female,No,Thur,Lunch,1,0.1817279046673287
32.68,5.0,Male,Yes,Thur,Lunch,2,0.15299877600979192
15.98,2.03,Male,No,Thur,Lunch,2,0.12703379224030037
34.83,5.17,Female,No,Thur,Lunch,4,0.14843525696238874
13.03,2.0,Male,No,Thur,Lunch,2,0.15349194167306218
18.28,4.0,Male,No,Thur,Lunch,2,0.2188183807439825
24.71,5.85,Male,No,Thur,Lunch,2,0.23674625657628487
21.16,3.0,Male,No,Thur,Lunch,2,0.14177693761814744
28.97,3.0,Male,Yes,Fri,Dinner,2,0.10355540214014498
22.49,3.5,Male,No,Fri,Dinner,2,0.15562472209871056
5.75,1.0,Female,Yes,Fri,Dinner,2,0.17391304347826086
16.32,4.3,Female,Yes,Fri,Dinner,2,0.26348039215686275
22.75,3.25,Female,No,Fri,Dinner,2,0.14285714285714285
40.17,4.73,Male,Yes,Fri,Dinner,4,0.1177495643515061
27.28,4.0,Male,Yes,Fri,Dinner,2,0.14662756598240467
12.03,1.5,Male,Yes,Fri,Dinner,2,0.12468827930174564
21.01,3.0,Male,Yes,Fri,Dinner,2,0.1427891480247501
12.46,1.5,Male,No,Fri,Dinner,2,0.1203852327447833
11.35,2.5,Female,Yes,Fri,Dinner,2,0.22026431718061676
15.38,3.0,Female,Yes,Fri,Dinner,2,0.19505851755526657
44.3,2.5,Female,Yes,Sat,Dinner,3,0.05643340857787811
22.42,3.48,Female,Yes,Sat,Dinner,2,0.15521855486173058
20.92,4.08,Female,No,Sat,Dinner,2,0.1950286806883365
15.36,1.64,Male,Yes,Sat,Dinner,2,0.10677083333333333
20.49,4.06,Male,Yes,Sat,Dinner,2,0.19814543679843827
25.21,4.29,Male,Yes,Sat,Dinner,2,0.1701705672352241
18.24,3.76,Male,No,Sat,Dinner,2,0.20614035087719298
```

```
14.31,4.0,Female,Yes,Sat,Dinner,2,0.2795248078266946
14.0,3.0,Male,No,Sat,Dinner,2,0.21428571428571427
7.25,1.0,Female,No,Sat,Dinner,1,0.13793103448275862
38.07,4.0,Male,No,Sun,Dinner,3,0.10506960861570791
23.95,2.55,Male,No,Sun,Dinner,2,0.10647181628392484
25.71,4.0,Female,No,Sun,Dinner,3,0.15558148580318942
17.31,3.5,Female,No,Sun,Dinner,2,0.20219526285384173
29.93,5.07,Male,No,Sun,Dinner,4,0.1693952555963916
10.65,1.5,Female,No,Thur,Lunch,2,0.14084507042253522
12.43,1.8,Female,No,Thur,Lunch,2,0.14481094127111827
24.08,2.92,Female,No,Thur,Lunch,4,0.1212624584717608
11.69,2.31,Male,No,Thur,Lunch,2,0.19760479041916168
13.42,1.68,Female,No,Thur,Lunch,2,0.12518628912071533
14.26,2.5,Male,No,Thur,Lunch,2,0.1753155680224404
15.95,2.0,Male,No,Thur,Lunch,2,0.12539184952978058
12.48,2.52,Female,No,Thur,Lunch,2,0.20192307692307693
29.8,4.2,Female,No,Thur,Lunch,6,0.14093959731543623
8.52,1.48,Male,No,Thur,Lunch,2,0.17370892018779344
14.52,2.0,Female,No,Thur,Lunch,2,0.13774104683195593
11.38,2.0,Female,No,Thur,Lunch,2,0.17574692442882248
22.82,2.18,Male,No,Thur,Lunch,3,0.09553023663453111
19.08,1.5,Male,No,Thur,Lunch,2,0.07861635220125787
20.27,2.83,Female,No,Thur,Lunch,2,0.13961519486926494
11.17,1.5,Female,No,Thur,Lunch,2,0.13428827215756492
12.26,2.0,Female,No,Thur,Lunch,2,0.1631321370309951
18.26,3.25,Female,No,Thur,Lunch,2,0.17798466593647316
8.51,1.25,Female,No,Thur,Lunch,2,0.14688601645123384
10.33,2.0,Female,No,Thur,Lunch,2,0.1936108422071636
```

```
14.15,2.0,Female,No,Thur,Lunch,2,0.1413427561837456
16.0,2.0,Male,Yes,Thur,Lunch,2,0.125
13.16,2.75,Female,No,Thur,Lunch,2,0.20896656534954408
17.47,3.5,Female,No,Thur,Lunch,2,0.20034344590726963
34.3,6.7,Male,No,Thur,Lunch,6,0.19533527696793004
41.19,5.0,Male,No,Thur,Lunch,5,0.12138868657441128
27.05,5.0,Female,No,Thur,Lunch,6,0.18484288354898337
16.43,2.3,Female,No,Thur,Lunch,2,0.1399878271454656
8.35,1.5,Female,No,Thur,Lunch,2,0.17964071856287425
18.64,1.36,Female,No,Thur,Lunch,3,0.07296137339055794
11.87,1.63,Female,No,Thur,Lunch,2,0.13732097725358045
9.78,1.73,Male,No,Thur,Lunch,2,0.1768916155419223
7.51,2.0,Male,No,Thur,Lunch,2,0.2663115845539281
14.07,2.5,Male,No,Sun,Dinner,2,0.17768301350390903
13.13,2.0,Male,No,Sun,Dinner,2,0.15232292460015232
17.26,2.74,Male,No,Sun,Dinner,3,0.15874855156431053
24.55,2.0,Male,No,Sun,Dinner,4,0.0814663951120163
19.77,2.0,Male,No,Sun,Dinner,4,0.10116337885685382
29.85,5.14,Female,No,Sun,Dinner,5,0.1721943048576214
48.17,5.0,Male,No,Sun,Dinner,6,0.10379904504878555
25.0,3.75,Female,No,Sun,Dinner,4,0.15
13.39,2.61,Female,No,Sun,Dinner,2,0.19492158327109782
16.49,2.0,Male,No,Sun,Dinner,4,0.12128562765312312
21.5,3.5,Male,No,Sun,Dinner,4,0.16279069767441862
12.66,2.5,Male,No,Sun,Dinner,2,0.19747235387045814
16.21,2.0,Female,No,Sun,Dinner,3,0.12338062924120913
13.81,2.0,Male,No,Sun,Dinner,2,0.1448225923244026
17.51,3.0,Female,Yes,Sun,Dinner,2,0.17133066818960593
```

```
24.52,3.48,Male,No,Sun,Dinner,3,0.14192495921696574
20.76,2.24,Male,No,Sun,Dinner,2,0.10789980732177264
31.71,4.5,Male,No,Sun,Dinner,4,0.14191106906338694
10.59,1.61,Female,Yes,Sat,Dinner,2,0.15203021718602455
10.63,2.0,Female,Yes,Sat,Dinner,2,0.18814675446848542
50.81,10.0,Male,Yes,Sat,Dinner,3,0.19681165124975397
15.81,3.16,Male,Yes,Sat,Dinner,2,0.19987349778621127
7.25,5.15,Male,Yes,Sun,Dinner,2,0.710344827586207
31.85,3.18,Male,Yes,Sun,Dinner,2,0.09984301412872841
16.82,4.0,Male,Yes,Sun,Dinner,2,0.23781212841854935
32.9,3.11,Male,Yes,Sun,Dinner,2,0.0945288753799392
17.89,2.0,Male,Yes,Sun,Dinner,2,0.11179429849077696
14.48,2.0,Male,Yes,Sun,Dinner,2,0.13812154696132597
9.6,4.0,Female,Yes,Sun,Dinner,2,0.4166666666666667
34.63,3.55,Male,Yes,Sun,Dinner,2,0.102512272596015
34.65,3.68,Male,Yes,Sun,Dinner,4,0.10620490620490622
23.33,5.65,Male,Yes,Sun,Dinner,2,0.2421774539219889
45.35,3.5,Male,Yes,Sun,Dinner,3,0.07717750826901874
23.17,6.5,Male,Yes,Sun,Dinner,4,0.2805351747949935
40.55,3.0,Male,Yes,Sun,Dinner,2,0.07398273736128237
20.69,5.0,Male,No,Sun,Dinner,5,0.2416626389560174
20.9,3.5,Female,Yes,Sun,Dinner,3,0.1674641148325359
30.46,2.0,Male,Yes,Sun,Dinner,5,0.06565988181221273
18.15,3.5,Female,Yes,Sun,Dinner,3,0.1928374655647383
23.1,4.0,Male,Yes,Sun,Dinner,3,0.17316017316017315
15.69,1.5,Male,Yes,Sun,Dinner,2,0.09560229445506692
19.81,4.19,Female,Yes,Thur,Lunch,2,0.21150933871781932
28.44,2.56,Male,Yes,Thur,Lunch,2,0.090014064697609
```

```
15.48,2.02,Male,Yes,Thur,Lunch,2,0.13049095607235142

16.58,4.0,Male,Yes,Thur,Lunch,2,0.24125452352231608

7.56,1.44,Male,No,Thur,Lunch,2,0.19047619047619047

10.34,2.0,Male,Yes,Thur,Lunch,2,0.19342359767891684

43.11,5.0,Female,Yes,Thur,Lunch,4,0.1159823706796567

13.0,2.0,Female,Yes,Thur,Lunch,2,0.15384615384615385

13.51,2.0,Male,Yes,Thur,Lunch,2,0.14803849000740193

18.71,4.0,Male,Yes,Thur,Lunch,3,0.2137894174238375

12.74,2.01,Female,Yes,Thur,Lunch,2,0.15777080062794346

13.0,2.0,Female,Yes,Thur,Lunch,2,0.15384615384615385

16.4,2.5,Female,Yes,Thur,Lunch,2,0.15243902439024393

20.53,4.0,Male,Yes,Thur,Lunch,4,0.1948368241597662

16.47,3.23,Female,Yes,Thur,Lunch,3,0.19611414693381907

26.59,3.41,Male,Yes,Sat,Dinner,3,0.1282437006393381

38.73,3.0,Male,Yes,Sat,Dinner,4,0.0774593338497289

24.27,2.03,Male,Yes,Sat,Dinner,2,0.08364235681911825

12.76,2.23,Female,Yes,Sat,Dinner,2,0.17476489028213166

30.06,2.0,Male,Yes,Sat,Dinner,3,0.06653359946773121

25.89,5.16,Male,Yes,Sat,Dinner,4,0.1993047508690614

48.33,9.0,Male,No,Sat,Dinner,4,0.186219739292365

13.27,2.5,Female,Yes,Sat,Dinner,2,0.18839487565938207

28.17,6.5,Female,Yes,Sat,Dinner,3,0.23074192403265883

12.9,1.1,Female,Yes,Sat,Dinner,2,0.08527131782945736

28.15,3.0,Male,Yes,Sat,Dinner,5,0.10657193605683837

11.59,1.5,Male,Yes,Sat,Dinner,2,0.12942191544434858

7.74,1.44,Male,Yes,Sat,Dinner,2,0.18604651162790697

30.14,3.09,Female,Yes,Sat,Dinner,4,0.10252156602521566

12.16,2.2,Male,Yes,Fri,Lunch,2,0.18092105263157895
```

```
13.42,3.48,Female,Yes,Fri,Lunch,2,0.2593144560357675
8.58,1.92,Male,Yes,Fri,Lunch,1,0.22377622377622378
15.98,3.0,Female,No,Fri,Lunch,3,0.18773466833541927
13.42,1.58,Male,Yes,Fri,Lunch,2,0.11773472429210134
16.27,2.5,Female,Yes,Fri,Lunch,2,0.15365703749231716
10.09,2.0,Female,Yes,Fri,Lunch,2,0.19821605550049554
20.45,3.0,Male,No,Sat,Dinner,4,0.1466992665036675
13.28,2.72,Male,No,Sat,Dinner,2,0.20481927710843376
22.12,2.88,Female,Yes,Sat,Dinner,2,0.13019891500904157
24.01,2.0,Male,Yes,Sat,Dinner,4,0.08329862557267805
15.69,3.0,Male,Yes,Sat,Dinner,3,0.19120458891013384
11.61,3.39,Male,No,Sat,Dinner,2,0.29198966408268734
10.77,1.47,Male,No,Sat,Dinner,2,0.13649025069637882
15.53,3.0,Male,Yes,Sat,Dinner,2,0.19317450096587252
10.07,1.25,Male,No,Sat,Dinner,2,0.12413108242303872
12.6,1.0,Male,Yes,Sat,Dinner,2,0.07936507936507936
32.83,1.17,Male,Yes,Sat,Dinner,2,0.03563813585135547
35.83,4.67,Female,No,Sat,Dinner,3,0.13033770583310075
29.03,5.92,Male,No,Sat,Dinner,3,0.2039269720978298
27.18,2.0,Female,Yes,Sat,Dinner,2,0.07358351729212656
22.67,2.0,Male,Yes,Sat,Dinner,2,0.08822232024702249
17.82,1.75,Male,No,Sat,Dinner,2,0.09820426487093153
18.78,3.0,Female,No,Thur,Dinner,2,0.1597444089456869
```

[Note:Dataset will be provided in external lab exam]