

COURSE DESCRIPTION

Machine Learning is concerned with computer programs that automatically improve their performance through experience. This course covers the theory and practical algorithms for machine learning from a variety of perspectives. We cover topics such as **FIND-S**, **Candidate Elimination Algorithm**, **Decision tree (ID3 Algorithm)**, **Backpropagation Algorithm**, **Naïve Bayesian classifier**, **Bayesian Network**, **k-Means Algorithm**, **k-Nearest Neighbour Algorithm**, **Locally Weighted Regression Algorithm**.

LAB EXPERIMENTS

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

```
import csv
a = []

with open('prg1.csv', 'r') as csvfile:
    for row in csv.reader(csvfile):
        a.append(row)
    print(a)

print("\n The total number of training instances are : ",len(a))

num_attribute = len(a[0])-1

print("\n The initial hypothesis is : ")
hypothesis = ['0']*num_attribute

print(hypothesis)

for i in range(0, len(a)):
    if a[i][num_attribute] == 'yes':
        for j in range(0, num_attribute):
            if hypothesis[j] == '0' or hypothesis[j] == a[i][j]:
                hypothesis[j] = a[i][j]
```

```
else:
    hypothesis[j] = '?'
    print("\n The hypothesis for the training instance {} is : \n" .format(i+1),hypothesis)
```

```
print("\n The Maximally specific hypothesis for the training instance is ")
print(hypothesis)
```

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
```

2. 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.read_csv('prg1.csv')
concepts = np.array(data.iloc[:,0:-1])
print(concepts)
target = np.array(data.iloc[:,-1])
print(target)

    if h[x]!= specific_h[x]:
        specific_h[x] ='?'
        general_h[x][x] ='?'

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 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)
```

```
print("Final General_h:", g_final)
```

```
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
```

3. 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 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
```

4. Build an Artificial Neural Network by implementing the Backpropagation algorithm and test the same using appropriate data sets.

```
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 Propagation
    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)
```

5. Write a program to implement the naïve 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 csv
import pandas as pd
import numpy as np
from sklearn.naive_bayes import GaussianNB

data = pd.read_csv('prima_indian_diabetes.csv')
```

```
x = np.array(data.iloc[:,0:-1])
y = np.array(data.iloc[:, -1])
```

```
print(data.head())
```

```
#Create a Gaussian Classifier
model = GaussianNB()
```

```
# Train the model using the training sets
model.fit(x,y)
```

```
#Predict Output
predicted= model.predict([[6,149,78,35,0,34,0.625,54]])
print("Predicted Value:", predicted)
```

```
prima_indian_diabetes.csv
```

```
pregnancies,Glucose,BloodPressure,SkinThickness,Insuline,BMI,DiabeticPed,Age,Outc
ome
```

```
6,148,72,35,0,33.6,0.627,50,1
1,85,66,29,0,26.6,0.351,31,0
8,183,64,0,0,23.3,0.672,32,1
1,89,66,23,94,28.1,0.167,21,0
0,137,40,35,168,43.1,2.288,33,1
5,116,74,0,0,25.6,0.201,30,0
3,78,50,32,88,31,0.248,26,1
10,115,0,0,0,35.3,0.134,29,0
2,197,70,45,543,30.5,0.158,53,1
8,125,96,0,0,0,0.232,54,1
4,110,92,0,0,37.6,0.191,30,0
```


10,168,74,0,0,38,0.537,34,1
10,139,80,0,0,27.1,1.441,57,0
1,189,60,23,846,30.1,0.398,59,1
5,166,72,19,175,25.8,0.587,51,1
7,100,0,0,0,30,0.484,32,1
0,118,84,47,230,45.8,0.551,31,1
7,107,74,0,0,29.6,0.254,31,1
1,103,30,38,83,43.3,0.183,33,0
1,115,70,30,96,34.6,0.529,32,1
3,126,88,41,235,39.3,0.704,27,0
8,99,84,0,0,35.4,0.388,50,0
7,196,90,0,0,39.8,0.451,41,1
9,119,80,35,0,29,0.263,29,1
11,143,94,33,146,36.6,0.254,51,1
10,125,70,26,115,31.1,0.205,41,1
7,147,76,0,0,39.4,0.257,43,1
1,97,66,15,140,23.2,0.487,22,0
13,145,82,19,110,22.2,0.245,57,0
5,117,92,0,0,34.1,0.337,38,0
5,109,75,26,0,36,0.546,60,0
3,158,76,36,245,31.6,0.851,28,1
3,88,58,11,54,24.8,0.267,22,0
6,92,92,0,0,19.9,0.188,28,0
10,122,78,31,0,27.6,0.512,45,0
4,103,60,33,192,24,0.966,33,0
11,138,76,0,0,33.2,0.42,35,0
9,102,76,37,0,32.9,0.665,46,1
2,90,68,42,0,38.2,0.503,27,1
4,111,72,47,207,37.1,1.39,56,1
3,180,64,25,70,34,0.271,26,0
7,133,84,0,0,40.2,0.696,37,0
7,106,92,18,0,22.7,0.235,48,0
9,171,110,24,240,45.4,0.721,54,1
7,159,64,0,0,27.4,0.294,40,0
0,180,66,39,0,42,1.893,25,1
1,146,56,0,0,29.7,0.564,29,0
2,71,70,27,0,28,0.586,22,0
7,103,66,32,0,39.1,0.344,31,1
7,105,0,0,0,0,0.305,24,0
1,103,80,11,82,19.4,0.491,22,0
1,101,50,15,36,24.2,0.526,26,0
5,88,66,21,23,24.4,0.342,30,0
8,176,90,34,300,33.7,0.467,58,1
7,150,66,42,342,34.7,0.718,42,0

1,73,50,10,0,23,0.248,21,0
7,187,68,39,304,37.7,0.254,41,1
0,100,88,60,110,46.8,0.962,31,0
0,146,82,0,0,40.5,1.781,44,0
0,105,64,41,142,41.5,0.173,22,0
2,84,0,0,0,0,0.304,21,0
8,133,72,0,0,32.9,0.27,39,1
5,44,62,0,0,25,0.587,36,0
2,141,58,34,128,25.4,0.699,24,0
7,114,66,0,0,32.8,0.258,42,1
5,99,74,27,0,29,0.203,32,0
0,109,88,30,0,32.5,0.855,38,1
2,109,92,0,0,42.7,0.845,54,0
1,95,66,13,38,19.6,0.334,25,0
4,146,85,27,100,28.9,0.189,27,0
2,100,66,20,90,32.9,0.867,28,1
5,139,64,35,140,28.6,0.411,26,0
13,126,90,0,0,43.4,0.583,42,1
4,129,86,20,270,35.1,0.231,23,0
1,79,75,30,0,32,0.396,22,0
1,0,48,20,0,24.7,0.14,22,0
7,62,78,0,0,32.6,0.391,41,0
5,95,72,33,0,37.7,0.37,27,0
0,131,0,0,0,43.2,0.27,26,1
2,112,66,22,0,25,0.307,24,0
3,113,44,13,0,22.4,0.14,22,0
2,74,0,0,0,0,0.102,22,0
7,83,78,26,71,29.3,0.767,36,0
0,101,65,28,0,24.6,0.237,22,0
5,137,108,0,0,48.8,0.227,37,1
2,110,74,29,125,32.4,0.698,27,0
13,106,72,54,0,36.6,0.178,45,0
2,100,68,25,71,38.5,0.324,26,0
15,136,70,32,110,37.1,0.153,43,1
1,107,68,19,0,26.5,0.165,24,0
1,80,55,0,0,19.1,0.258,21,0
4,123,80,15,176,32,0.443,34,0
7,81,78,40,48,46.7,0.261,42,0
4,134,72,0,0,23.8,0.277,60,1
2,142,82,18,64,24.7,0.761,21,0
6,144,72,27,228,33.9,0.255,40,0
2,92,62,28,0,31.6,0.13,24,0
1,71,48,18,76,20.4,0.323,22,0
6,93,50,30,64,28.7,0.356,23,0

1,122,90,51,220,49.7,0.325,31,1
1,163,72,0,0,39,1.222,33,1
1,151,60,0,0,26.1,0.179,22,0
0,125,96,0,0,22.5,0.262,21,0
1,81,72,18,40,26.6,0.283,24,0
2,85,65,0,0,39.6,0.93,27,0
1,126,56,29,152,28.7,0.801,21,0
1,96,122,0,0,22.4,0.207,27,0
4,144,58,28,140,29.5,0.287,37,0
3,83,58,31,18,34.3,0.336,25,0
0,95,85,25,36,37.4,0.247,24,1
3,171,72,33,135,33.3,0.199,24,1
8,155,62,26,495,34,0.543,46,1
1,89,76,34,37,31.2,0.192,23,0
4,76,62,0,0,34,0.391,25,0
7,160,54,32,175,30.5,0.588,39,1
4,146,92,0,0,31.2,0.539,61,1
5,124,74,0,0,34,0.22,38,1
5,78,48,0,0,33.7,0.654,25,0
4,97,60,23,0,28.2,0.443,22,0
4,99,76,15,51,23.2,0.223,21,0
0,162,76,56,100,53.2,0.759,25,1
6,111,64,39,0,34.2,0.26,24,0
2,107,74,30,100,33.6,0.404,23,0
5,132,80,0,0,26.8,0.186,69,0
0,113,76,0,0,33.3,0.278,23,1
1,88,30,42,99,55,0.496,26,1
3,120,70,30,135,42.9,0.452,30,0
1,118,58,36,94,33.3,0.261,23,0
1,117,88,24,145,34.5,0.403,40,1
0,105,84,0,0,27.9,0.741,62,1
4,173,70,14,168,29.7,0.361,33,1
9,122,56,0,0,33.3,1.114,33,1
3,170,64,37,225,34.5,0.356,30,1
8,84,74,31,0,38.3,0.457,39,0
2,96,68,13,49,21.1,0.647,26,0
2,125,60,20,140,33.8,0.088,31,0
0,100,70,26,50,30.8,0.597,21,0
0,93,60,25,92,28.7,0.532,22,0
0,129,80,0,0,31.2,0.703,29,0
5,105,72,29,325,36.9,0.159,28,0
3,128,78,0,0,21.1,0.268,55,0
5,106,82,30,0,39.5,0.286,38,0
2,108,52,26,63,32.5,0.318,22,0

10,108,66,0,0,32.4,0.272,42,1
4,154,62,31,284,32.8,0.237,23,0
0,102,75,23,0,0,0.572,21,0
9,57,80,37,0,32.8,0.096,41,0
2,106,64,35,119,30.5,1.4,34,0
5,147,78,0,0,33.7,0.218,65,0
2,90,70,17,0,27.3,0.085,22,0
1,136,74,50,204,37.4,0.399,24,0
4,114,65,0,0,21.9,0.432,37,0
9,156,86,28,155,34.3,1.189,42,1
1,153,82,42,485,40.6,0.687,23,0
8,188,78,0,0,47.9,0.137,43,1
7,152,88,44,0,50,0.337,36,1
2,99,52,15,94,24.6,0.637,21,0
1,109,56,21,135,25.2,0.833,23,0
2,88,74,19,53,29,0.229,22,0
17,163,72,41,114,40.9,0.817,47,1
4,151,90,38,0,29.7,0.294,36,0
7,102,74,40,105,37.2,0.204,45,0
0,114,80,34,285,44.2,0.167,27,0
2,100,64,23,0,29.7,0.368,21,0
0,131,88,0,0,31.6,0.743,32,1
6,104,74,18,156,29.9,0.722,41,1
3,148,66,25,0,32.5,0.256,22,0
4,120,68,0,0,29.6,0.709,34,0
4,110,66,0,0,31.9,0.471,29,0
3,111,90,12,78,28.4,0.495,29,0
6,102,82,0,0,30.8,0.18,36,1
6,134,70,23,130,35.4,0.542,29,1
2,87,0,23,0,28.9,0.773,25,0
1,79,60,42,48,43.5,0.678,23,0
2,75,64,24,55,29.7,0.37,33,0
8,179,72,42,130,32.7,0.719,36,1
6,85,78,0,0,31.2,0.382,42,0
0,129,110,46,130,67.1,0.319,26,1
5,143,78,0,0,45,0.19,47,0
5,130,82,0,0,39.1,0.956,37,1
6,87,80,0,0,23.2,0.084,32,0
0,119,64,18,92,34.9,0.725,23,0
1,0,74,20,23,27.7,0.299,21,0
5,73,60,0,0,26.8,0.268,27,0
4,141,74,0,0,27.6,0.244,40,0
7,194,68,28,0,35.9,0.745,41,1
8,181,68,36,495,30.1,0.615,60,1

1,128,98,41,58,32,1.321,33,1
8,109,76,39,114,27.9,0.64,31,1
5,139,80,35,160,31.6,0.361,25,1
3,111,62,0,0,22.6,0.142,21,0
9,123,70,44,94,33.1,0.374,40,0
7,159,66,0,0,30.4,0.383,36,1
11,135,0,0,0,52.3,0.578,40,1
8,85,55,20,0,24.4,0.136,42,0
5,158,84,41,210,39.4,0.395,29,1
1,105,58,0,0,24.3,0.187,21,0
3,107,62,13,48,22.9,0.678,23,1
4,109,64,44,99,34.8,0.905,26,1
4,148,60,27,318,30.9,0.15,29,1
0,113,80,16,0,31,0.874,21,0
1,138,82,0,0,40.1,0.236,28,0
0,108,68,20,0,27.3,0.787,32,0
2,99,70,16,44,20.4,0.235,27,0
6,103,72,32,190,37.7,0.324,55,0
5,111,72,28,0,23.9,0.407,27,0
8,196,76,29,280,37.5,0.605,57,1
5,162,104,0,0,37.7,0.151,52,1
1,96,64,27,87,33.2,0.289,21,0
7,184,84,33,0,35.5,0.355,41,1
2,81,60,22,0,27.7,0.29,25,0
0,147,85,54,0,42.8,0.375,24,0
7,179,95,31,0,34.2,0.164,60,0
0,140,65,26,130,42.6,0.431,24,1
9,112,82,32,175,34.2,0.26,36,1
12,151,70,40,271,41.8,0.742,38,1
5,109,62,41,129,35.8,0.514,25,1
6,125,68,30,120,30,0.464,32,0
5,85,74,22,0,29,1.224,32,1
5,112,66,0,0,37.8,0.261,41,1
0,177,60,29,478,34.6,1.072,21,1
2,158,90,0,0,31.6,0.805,66,1
7,119,0,0,0,25.2,0.209,37,0
7,142,60,33,190,28.8,0.687,61,0
1,100,66,15,56,23.6,0.666,26,0
1,87,78,27,32,34.6,0.101,22,0
0,101,76,0,0,35.7,0.198,26,0
3,162,52,38,0,37.2,0.652,24,1
4,197,70,39,744,36.7,2.329,31,0
0,117,80,31,53,45.2,0.089,24,0
4,142,86,0,0,44,0.645,22,1

6,134,80,37,370,46.2,0.238,46,1
1,79,80,25,37,25.4,0.583,22,0
4,122,68,0,0,35,0.394,29,0
3,74,68,28,45,29.7,0.293,23,0
4,171,72,0,0,43.6,0.479,26,1
7,181,84,21,192,35.9,0.586,51,1
0,179,90,27,0,44.1,0.686,23,1
9,164,84,21,0,30.8,0.831,32,1
0,104,76,0,0,18.4,0.582,27,0
1,91,64,24,0,29.2,0.192,21,0
4,91,70,32,88,33.1,0.446,22,0
3,139,54,0,0,25.6,0.402,22,1
6,119,50,22,176,27.1,1.318,33,1
2,146,76,35,194,38.2,0.329,29,0
9,184,85,15,0,30,1.213,49,1
10,122,68,0,0,31.2,0.258,41,0
0,165,90,33,680,52.3,0.427,23,0
9,124,70,33,402,35.4,0.282,34,0
1,111,86,19,0,30.1,0.143,23,0
9,106,52,0,0,31.2,0.38,42,0
2,129,84,0,0,28,0.284,27,0
2,90,80,14,55,24.4,0.249,24,0
0,86,68,32,0,35.8,0.238,25,0
12,92,62,7,258,27.6,0.926,44,1
1,113,64,35,0,33.6,0.543,21,1
3,111,56,39,0,30.1,0.557,30,0
2,114,68,22,0,28.7,0.092,25,0
1,193,50,16,375,25.9,0.655,24,0
11,155,76,28,150,33.3,1.353,51,1
3,191,68,15,130,30.9,0.299,34,0
3,141,0,0,0,30,0.761,27,1
4,95,70,32,0,32.1,0.612,24,0
3,142,80,15,0,32.4,0.2,63,0
4,123,62,0,0,32,0.226,35,1
5,96,74,18,67,33.6,0.997,43,0
0,138,0,0,0,36.3,0.933,25,1
2,128,64,42,0,40,1.101,24,0
0,102,52,0,0,25.1,0.078,21,0
2,146,0,0,0,27.5,0.24,28,1 just
10,101,86,37,0,45.6,1.136,38,1
2,108,62,32,56,25.2,0.128,21,0
3,122,78,0,0,23,0.254,40,0
1,71,78,50,45,33.2,0.422,21,0
13,106,70,0,0,34.2,0.251,52,0

2,100,70,52,57,40.5,0.677,25,0
7,106,60,24,0,26.5,0.296,29,1
0,104,64,23,116,27.8,0.454,23,0
5,114,74,0,0,24.9,0.744,57,0
2,108,62,10,278,25.3,0.881,22,0
0,146,70,0,0,37.9,0.334,28,1
10,129,76,28,122,35.9,0.28,39,0
7,133,88,15,155,32.4,0.262,37,0
7,161,86,0,0,30.4,0.165,47,1
2,108,80,0,0,27,0.259,52,1
7,136,74,26,135,26,0.647,51,0
5,155,84,44,545,38.7,0.619,34,0
1,119,86,39,220,45.6,0.808,29,1
4,96,56,17,49,20.8,0.34,26,0
5,108,72,43,75,36.1,0.263,33,0
0,78,88,29,40,36.9,0.434,21,0
0,107,62,30,74,36.6,0.757,25,1
2,128,78,37,182,43.3,1.224,31,1
1,128,48,45,194,40.5,0.613,24,1
0,161,50,0,0,21.9,0.254,65,0
6,151,62,31,120,35.5,0.692,28,0
2,146,70,38,360,28,0.337,29,1
0,126,84,29,215,30.7,0.52,24,0
14,100,78,25,184,36.6,0.412,46,1
8,112,72,0,0,23.6,0.84,58,0
0,167,0,0,0,32.3,0.839,30,1
2,144,58,33,135,31.6,0.422,25,1
5,77,82,41,42,35.8,0.156,35,0
5,115,98,0,0,52.9,0.209,28,1
3,150,76,0,0,21,0.207,37,0
2,120,76,37,105,39.7,0.215,29,0
10,161,68,23,132,25.5,0.326,47,1
0,137,68,14,148,24.8,0.143,21,0
0,128,68,19,180,30.5,1.391,25,1
2,124,68,28,205,32.9,0.875,30,1
6,80,66,30,0,26.2,0.313,41,0
0,106,70,37,148,39.4,0.605,22,0
2,155,74,17,96,26.6,0.433,27,1
3,113,50,10,85,29.5,0.626,25,0
7,109,80,31,0,35.9,1.127,43,1
2,112,68,22,94,34.1,0.315,26,0
3,99,80,11,64,19.3,0.284,30,0
3,182,74,0,0,30.5,0.345,29,1
3,115,66,39,140,38.1,0.15,28,0

6,194,78,0,0,23.5,0.129,59,1
4,129,60,12,231,27.5,0.527,31,0
3,112,74,30,0,31.6,0.197,25,1
0,124,70,20,0,27.4,0.254,36,1
13,152,90,33,29,26.8,0.731,43,1
2,112,75,32,0,35.7,0.148,21,0
1,157,72,21,168,25.6,0.123,24,0
1,122,64,32,156,35.1,0.692,30,1
10,179,70,0,0,35.1,0.2,37,0
2,102,86,36,120,45.5,0.127,23,1
6,105,70,32,68,30.8,0.122,37,0
8,118,72,19,0,23.1,1.476,46,0
2,87,58,16,52,32.7,0.166,25,0
1,180,0,0,0,43.3,0.282,41,1
12,106,80,0,0,23.6,0.137,44,0
1,95,60,18,58,23.9,0.26,22,0
0,165,76,43,255,47.9,0.259,26,0
0,117,0,0,0,33.8,0.932,44,0
5,115,76,0,0,31.2,0.343,44,1
9,152,78,34,171,34.2,0.893,33,1
7,178,84,0,0,39.9,0.331,41,1
1,130,70,13,105,25.9,0.472,22,0
1,95,74,21,73,25.9,0.673,36,0
1,0,68,35,0,32,0.389,22,0
5,122,86,0,0,34.7,0.29,33,0
8,95,72,0,0,36.8,0.485,57,0
8,126,88,36,108,38.5,0.349,49,0
1,139,46,19,83,28.7,0.654,22,0
3,116,0,0,0,23.5,0.187,23,0
3,99,62,19,74,21.8,0.279,26,0
5,0,80,32,0,41,0.346,37,1
4,92,80,0,0,42.2,0.237,29,0
4,137,84,0,0,31.2,0.252,30,0
3,61,82,28,0,34.4,0.243,46,0
1,90,62,12,43,27.2,0.58,24,0
3,90,78,0,0,42.7,0.559,21,0
9,165,88,0,0,30.4,0.302,49,1
1,125,50,40,167,33.3,0.962,28,1
13,129,0,30,0,39.9,0.569,44,1
12,88,74,40,54,35.3,0.378,48,0
1,196,76,36,249,36.5,0.875,29,1
5,189,64,33,325,31.2,0.583,29,1
5,158,70,0,0,29.8,0.207,63,0
5,103,108,37,0,39.2,0.305,65,0

4,146,78,0,0,38.5,0.52,67,1
4,147,74,25,293,34.9,0.385,30,0
5,99,54,28,83,34,0.499,30,0
6,124,72,0,0,27.6,0.368,29,1
0,101,64,17,0,21,0.252,21,0
3,81,86,16,66,27.5,0.306,22,0
1,133,102,28,140,32.8,0.234,45,1
3,173,82,48,465,38.4,2.137,25,1
0,118,64,23,89,0,1.731,21,0
0,84,64,22,66,35.8,0.545,21,0
2,105,58,40,94,34.9,0.225,25,0
2,122,52,43,158,36.2,0.816,28,0
12,140,82,43,325,39.2,0.528,58,1
0,98,82,15,84,25.2,0.299,22,0
1,87,60,37,75,37.2,0.509,22,0
4,156,75,0,0,48.3,0.238,32,1
0,93,100,39,72,43.4,1.021,35,0
1,107,72,30,82,30.8,0.821,24,0
0,105,68,22,0,20,0.236,22,0
1,109,60,8,182,25.4,0.947,21,0
1,90,62,18,59,25.1,1.268,25,0
1,125,70,24,110,24.3,0.221,25,0
1,119,54,13,50,22.3,0.205,24,0
5,116,74,29,0,32.3,0.66,35,1
8,105,100,36,0,43.3,0.239,45,1
5,144,82,26,285,32,0.452,58,1
3,100,68,23,81,31.6,0.949,28,0
1,100,66,29,196,32,0.444,42,0
5,166,76,0,0,45.7,0.34,27,1
1,131,64,14,415,23.7,0.389,21,0
4,116,72,12,87,22.1,0.463,37,0
4,158,78,0,0,32.9,0.803,31,1
2,127,58,24,275,27.7,1.6,25,0
3,96,56,34,115,24.7,0.944,39,0
0,131,66,40,0,34.3,0.196,22,1
3,82,70,0,0,21.1,0.389,25,0
3,193,70,31,0,34.9,0.241,25,1
4,95,64,0,0,32,0.161,31,1
6,137,61,0,0,24.2,0.151,55,0
5,136,84,41,88,35,0.286,35,1
9,72,78,25,0,31.6,0.28,38,0
5,168,64,0,0,32.9,0.135,41,1
2,123,48,32,165,42.1,0.52,26,0
4,115,72,0,0,28.9,0.376,46,1

0,101,62,0,0,21.9,0.336,25,0
8,197,74,0,0,25.9,1.191,39,1
1,172,68,49,579,42.4,0.702,28,1
6,102,90,39,0,35.7,0.674,28,0
1,112,72,30,176,34.4,0.528,25,0
1,143,84,23,310,42.4,1.076,22,0
1,143,74,22,61,26.2,0.256,21,0
0,138,60,35,167,34.6,0.534,21,1
3,173,84,33,474,35.7,0.258,22,1
1,97,68,21,0,27.2,1.095,22,0
4,144,82,32,0,38.5,0.554,37,1
1,83,68,0,0,18.2,0.624,27,0
3,129,64,29,115,26.4,0.219,28,1
1,119,88,41,170,45.3,0.507,26,0
2,94,68,18,76,26,0.561,21,0
0,102,64,46,78,40.6,0.496,21,0
2,115,64,22,0,30.8,0.421,21,0
8,151,78,32,210,42.9,0.516,36,1
4,184,78,39,277,37,0.264,31,1
0,94,0,0,0,0,0.256,25,0
1,181,64,30,180,34.1,0.328,38,1
0,135,94,46,145,40.6,0.284,26,0
1,95,82,25,180,35,0.233,43,1
2,99,0,0,0,22.2,0.108,23,0
3,89,74,16,85,30.4,0.551,38,0
1,80,74,11,60,30,0.527,22,0
2,139,75,0,0,25.6,0.167,29,0
1,90,68,8,0,24.5,1.138,36,0
0,141,0,0,0,42.4,0.205,29,1
12,140,85,33,0,37.4,0.244,41,0
5,147,75,0,0,29.9,0.434,28,0
1,97,70,15,0,18.2,0.147,21,0
6,107,88,0,0,36.8,0.727,31,0
0,189,104,25,0,34.3,0.435,41,1
2,83,66,23,50,32.2,0.497,22,0
4,117,64,27,120,33.2,0.23,24,0
8,108,70,0,0,30.5,0.955,33,1
4,117,62,12,0,29.7,0.38,30,1
0,180,78,63,14,59.4,2.42,25,1
1,100,72,12,70,25.3,0.658,28,0
0,95,80,45,92,36.5,0.33,26,0
0,104,64,37,64,33.6,0.51,22,1
0,120,74,18,63,30.5,0.285,26,0
1,82,64,13,95,21.2,0.415,23,0

2,134,70,0,0,28.9,0.542,23,1
0,91,68,32,210,39.9,0.381,25,0
2,119,0,0,0,19.6,0.832,72,0
2,100,54,28,105,37.8,0.498,24,0
14,175,62,30,0,33.6,0.212,38,1
1,135,54,0,0,26.7,0.687,62,0
5,86,68,28,71,30.2,0.364,24,0
10,148,84,48,237,37.6,1.001,51,1
9,134,74,33,60,25.9,0.46,81,0
9,120,72,22,56,20.8,0.733,48,0
1,71,62,0,0,21.8,0.416,26,0
8,74,70,40,49,35.3,0.705,39,0
5,88,78,30,0,27.6,0.258,37,0
10,115,98,0,0,24,1.022,34,0
0,124,56,13,105,21.8,0.452,21,0
0,74,52,10,36,27.8,0.269,22,0
0,97,64,36,100,36.8,0.6,25,0
8,120,0,0,0,30,0.183,38,1
6,154,78,41,140,46.1,0.571,27,0
1,144,82,40,0,41.3,0.607,28,0
0,137,70,38,0,33.2,0.17,22,0
0,119,66,27,0,38.8,0.259,22,0
7,136,90,0,0,29.9,0.21,50,0
4,114,64,0,0,28.9,0.126,24,0
0,137,84,27,0,27.3,0.231,59,0
2,105,80,45,191,33.7,0.711,29,1
7,114,76,17,110,23.8,0.466,31,0
8,126,74,38,75,25.9,0.162,39,0
4,132,86,31,0,28,0.419,63,0
3,158,70,30,328,35.5,0.344,35,1
0,123,88,37,0,35.2,0.197,29,0
4,85,58,22,49,27.8,0.306,28,0
0,84,82,31,125,38.2,0.233,23,0
0,145,0,0,0,44.2,0.63,31,1
0,135,68,42,250,42.3,0.365,24,1
1,139,62,41,480,40.7,0.536,21,0
0,173,78,32,265,46.5,1.159,58,0
4,99,72,17,0,25.6,0.294,28,0
8,194,80,0,0,26.1,0.551,67,0
2,83,65,28,66,36.8,0.629,24,0
2,89,90,30,0,33.5,0.292,42,0
4,99,68,38,0,32.8,0.145,33,0
4,125,70,18,122,28.9,1.144,45,1
3,80,0,0,0,0,0.174,22,0

6,166,74,0,0,26.6,0.304,66,0
5,110,68,0,0,26,0.292,30,0
2,81,72,15,76,30.1,0.547,25,0
7,195,70,33,145,25.1,0.163,55,1
6,154,74,32,193,29.3,0.839,39,0
2,117,90,19,71,25.2,0.313,21,0
3,84,72,32,0,37.2,0.267,28,0
6,0,68,41,0,39,0.727,41,1
7,94,64,25,79,33.3,0.738,41,0
3,96,78,39,0,37.3,0.238,40,0
10,75,82,0,0,33.3,0.263,38,0
0,180,90,26,90,36.5,0.314,35,1
1,130,60,23,170,28.6,0.692,21,0
2,84,50,23,76,30.4,0.968,21,0
8,120,78,0,0,25,0.409,64,0
12,84,72,31,0,29.7,0.297,46,1
0,139,62,17,210,22.1,0.207,21,0
9,91,68,0,0,24.2,0.2,58,0
2,91,62,0,0,27.3,0.525,22,0
3,99,54,19,86,25.6,0.154,24,0
3,163,70,18,105,31.6,0.268,28,1
9,145,88,34,165,30.3,0.771,53,1
7,125,86,0,0,37.6,0.304,51,0
13,76,60,0,0,32.8,0.18,41,0
6,129,90,7,326,19.6,0.582,60,0
2,68,70,32,66,25,0.187,25,0
3,124,80,33,130,33.2,0.305,26,0
6,114,0,0,0,0,0.189,26,0
9,130,70,0,0,34.2,0.652,45,1
3,125,58,0,0,31.6,0.151,24,0
3,87,60,18,0,21.8,0.444,21,0
1,97,64,19,82,18.2,0.299,21,0
3,116,74,15,105,26.3,0.107,24,0
0,117,66,31,188,30.8,0.493,22,0
0,111,65,0,0,24.6,0.66,31,0
2,122,60,18,106,29.8,0.717,22,0
0,107,76,0,0,45.3,0.686,24,0
1,86,66,52,65,41.3,0.917,29,0
6,91,0,0,0,29.8,0.501,31,0
1,77,56,30,56,33.3,1.251,24,0
4,132,0,0,0,32.9,0.302,23,1
0,105,90,0,0,29.6,0.197,46,0
0,57,60,0,0,21.7,0.735,67,0
0,127,80,37,210,36.3,0.804,23,0

3,129,92,49,155,36.4,0.968,32,1
8,100,74,40,215,39.4,0.661,43,1
3,128,72,25,190,32.4,0.549,27,1
10,90,85,32,0,34.9,0.825,56,1
4,84,90,23,56,39.5,0.159,25,0
1,88,78,29,76,32,0.365,29,0
8,186,90,35,225,34.5,0.423,37,1
5,187,76,27,207,43.6,1.034,53,1
4,131,68,21,166,33.1,0.16,28,0
1,164,82,43,67,32.8,0.341,50,0
4,189,110,31,0,28.5,0.68,37,0
1,116,70,28,0,27.4,0.204,21,0
3,84,68,30,106,31.9,0.591,25,0
6,114,88,0,0,27.8,0.247,66,0
1,88,62,24,44,29.9,0.422,23,0
1,84,64,23,115,36.9,0.471,28,0
7,124,70,33,215,25.5,0.161,37,0
1,97,70,40,0,38.1,0.218,30,0
8,110,76,0,0,27.8,0.237,58,0
11,103,68,40,0,46.2,0.126,42,0
11,85,74,0,0,30.1,0.3,35,0
6,125,76,0,0,33.8,0.121,54,1
0,198,66,32,274,41.3,0.502,28,1
1,87,68,34,77,37.6,0.401,24,0
6,99,60,19,54,26.9,0.497,32,0
0,91,80,0,0,32.4,0.601,27,0
2,95,54,14,88,26.1,0.748,22,0
1,99,72,30,18,38.6,0.412,21,0
6,92,62,32,126,32,0.085,46,0
4,154,72,29,126,31.3,0.338,37,0
0,121,66,30,165,34.3,0.203,33,1
3,78,70,0,0,32.5,0.27,39,0
2,130,96,0,0,22.6,0.268,21,0
3,111,58,31,44,29.5,0.43,22,0
2,98,60,17,120,34.7,0.198,22,0
1,143,86,30,330,30.1,0.892,23,0
1,119,44,47,63,35.5,0.28,25,0
6,108,44,20,130,24,0.813,35,0
2,118,80,0,0,42.9,0.693,21,1
10,133,68,0,0,27,0.245,36,0
2,197,70,99,0,34.7,0.575,62,1
0,151,90,46,0,42.1,0.371,21,1
6,109,60,27,0,25,0.206,27,0
12,121,78,17,0,26.5,0.259,62,0

8,100,76,0,0,38.7,0.19,42,0
8,124,76,24,600,28.7,0.687,52,1
1,93,56,11,0,22.5,0.417,22,0
8,143,66,0,0,34.9,0.129,41,1
6,103,66,0,0,24.3,0.249,29,0
3,176,86,27,156,33.3,1.154,52,1
0,73,0,0,0,21.1,0.342,25,0
11,111,84,40,0,46.8,0.925,45,1
2,112,78,50,140,39.4,0.175,24,0
3,132,80,0,0,34.4,0.402,44,1
2,82,52,22,115,28.5,1.699,25,0
6,123,72,45,230,33.6,0.733,34,0
0,188,82,14,185,32.0,682,22,1
0,67,76,0,0,45.3,0.194,46,0
1,89,24,19,25,27.8,0.559,21,0
1,173,74,0,0,36.8,0.088,38,1
1,109,38,18,120,23.1,0.407,26,0
1,108,88,19,0,27.1,0.4,24,0
6,96,0,0,0,23.7,0.19,28,0
1,124,74,36,0,27.8,0.1,30,0
7,150,78,29,126,35.2,0.692,54,1
4,183,0,0,0,28.4,0.212,36,1
1,124,60,32,0,35.8,0.514,21,0
1,181,78,42,293,40,1.258,22,1
1,92,62,25,41,19.5,0.482,25,0
0,152,82,39,272,41.5,0.27,27,0
1,111,62,13,182,24,0.138,23,0
3,106,54,21,158,30.9,0.292,24,0
3,174,58,22,194,32.9,0.593,36,1
7,168,88,42,321,38.2,0.787,40,1
6,105,80,28,0,32.5,0.878,26,0
11,138,74,26,144,36.1,0.557,50,1
3,106,72,0,0,25.8,0.207,27,0
6,117,96,0,0,28.7,0.157,30,0
2,68,62,13,15,20.1,0.257,23,0
9,112,82,24,0,28.2,1.282,50,1
0,119,0,0,0,32.4,0.141,24,1
2,112,86,42,160,38.4,0.246,28,0
2,92,76,20,0,24.2,1.698,28,0
6,183,94,0,0,40.8,1.461,45,0
0,94,70,27,115,43.5,0.347,21,0
2,108,64,0,0,30.8,0.158,21,0
4,90,88,47,54,37.7,0.362,29,0
0,125,68,0,0,24.7,0.206,21,0

0,132,78,0,0,32.4,0.393,21,0
5,128,80,0,0,34.6,0.144,45,0
4,94,65,22,0,24.7,0.148,21,0
7,114,64,0,0,27.4,0.732,34,1
0,102,78,40,90,34.5,0.238,24,0
2,111,60,0,0,26.2,0.343,23,0
1,128,82,17,183,27.5,0.115,22,0
10,92,62,0,0,25.9,0.167,31,0
13,104,72,0,0,31.2,0.465,38,1
5,104,74,0,0,28.8,0.153,48,0
2,94,76,18,66,31.6,0.649,23,0
7,97,76,32,91,40.9,0.871,32,1
1,100,74,12,46,19.5,0.149,28,0
0,102,86,17,105,29.3,0.695,27,0
4,128,70,0,0,34.3,0.303,24,0
6,147,80,0,0,29.5,0.178,50,1
4,90,0,0,0,28,0.61,31,0
3,103,72,30,152,27.6,0.73,27,0
2,157,74,35,440,39.4,0.134,30,0
1,167,74,17,144,23.4,0.447,33,1
0,179,50,36,159,37.8,0.455,22,1
11,136,84,35,130,28.3,0.26,42,1
0,107,60,25,0,26.4,0.133,23,0
1,91,54,25,100,25.2,0.234,23,0
1,117,60,23,106,33.8,0.466,27,0
5,123,74,40,77,34.1,0.269,28,0
2,120,54,0,0,26.8,0.455,27,0
1,106,70,28,135,34.2,0.142,22,0
2,155,52,27,540,38.7,0.24,25,1
2,101,58,35,90,21.8,0.155,22,0
1,120,80,48,200,38.9,1.162,41,0
11,127,106,0,0,39,0.19,51,0
3,80,82,31,70,34.2,1.292,27,1
10,162,84,0,0,27.7,0.182,54,0
1,199,76,43,0,42.9,1.394,22,1
8,167,106,46,231,37.6,0.165,43,1
9,145,80,46,130,37.9,0.637,40,1
6,115,60,39,0,33.7,0.245,40,1
1,112,80,45,132,34.8,0.217,24,0
4,145,82,18,0,32.5,0.235,70,1
10,111,70,27,0,27.5,0.141,40,1
6,98,58,33,190,34,0.43,43,0
9,154,78,30,100,30.9,0.164,45,0
6,165,68,26,168,33.6,0.631,49,0

1,99,58,10,0,25.4,0.551,21,0
10,68,106,23,49,35.5,0.285,47,0
3,123,100,35,240,57.3,0.88,22,0
8,91,82,0,0,35.6,0.587,68,0
6,195,70,0,0,30.9,0.328,31,1
9,156,86,0,0,24.8,0.23,53,1
0,93,60,0,0,35.3,0.263,25,0
3,121,52,0,0,36,0.127,25,1
2,101,58,17,265,24.2,0.614,23,0
2,56,56,28,45,24.2,0.332,22,0
0,162,76,36,0,49.6,0.364,26,1
0,95,64,39,105,44.6,0.366,22,0
4,125,80,0,0,32.3,0.536,27,1
5,136,82,0,0,0,0.64,69,0
2,129,74,26,205,33.2,0.591,25,0
3,130,64,0,0,23.1,0.314,22,0
1,107,50,19,0,28.3,0.181,29,0
1,140,74,26,180,24.1,0.828,23,0
1,144,82,46,180,46.1,0.335,46,1
8,107,80,0,0,24.6,0.856,34,0
13,158,114,0,0,42.3,0.257,44,1
2,121,70,32,95,39.1,0.886,23,0
7,129,68,49,125,38.5,0.439,43,1
2,90,60,0,0,23.5,0.191,25,0
7,142,90,24,480,30.4,0.128,43,1
3,169,74,19,125,29.9,0.268,31,1
0,99,0,0,0,25,0.253,22,0
4,127,88,11,155,34.5,0.598,28,0
4,118,70,0,0,44.5,0.904,26,0
2,122,76,27,200,35.9,0.483,26,0
6,125,78,31,0,27.6,0.565,49,1
1,168,88,29,0,35,0.905,52,1
2,129,0,0,0,38.5,0.304,41,0
4,110,76,20,100,28.4,0.118,27,0
6,80,80,36,0,39.8,0.177,28,0
10,115,0,0,0,0,0.261,30,1
2,127,46,21,335,34.4,0.176,22,0
9,164,78,0,0,32.8,0.148,45,1
2,93,64,32,160,38,0.674,23,1
3,158,64,13,387,31.2,0.295,24,0
5,126,78,27,22,29.6,0.439,40,0
10,129,62,36,0,41.2,0.441,38,1
0,134,58,20,291,26.4,0.352,21,0
3,102,74,0,0,29.5,0.121,32,0

7,187,50,33,392,33.9,0.826,34,1
3,173,78,39,185,33.8,0.97,31,1
10,94,72,18,0,23.1,0.595,56,0
1,108,60,46,178,35.5,0.415,24,0
5,97,76,27,0,35.6,0.378,52,1
4,83,86,19,0,29.3,0.317,34,0
1,114,66,36,200,38.1,0.289,21,0
1,149,68,29,127,29.3,0.349,42,1
5,117,86,30,105,39.1,0.251,42,0
1,111,94,0,0,32.8,0.265,45,0
4,112,78,40,0,39.4,0.236,38,0
1,116,78,29,180,36.1,0.496,25,0
0,141,84,26,0,32.4,0.433,22,0
2,175,88,0,0,22.9,0.326,22,0
2,92,52,0,0,30.1,0.141,22,0
3,130,78,23,79,28.4,0.323,34,1
8,120,86,0,0,28.4,0.259,22,1
2,174,88,37,120,44.5,0.646,24,1
2,106,56,27,165,29,0.426,22,0
2,105,75,0,0,23.3,0.56,53,0
4,95,60,32,0,35.4,0.284,28,0
0,126,86,27,120,27.4,0.515,21,0
8,65,72,23,0,32,0.6,42,0
2,99,60,17,160,36.6,0.453,21,0
1,102,74,0,0,39.5,0.293,42,1
11,120,80,37,150,42.3,0.785,48,1
3,102,44,20,94,30.8,0.4,26,0
1,109,58,18,116,28.5,0.219,22,0
9,140,94,0,0,32.7,0.734,45,1
13,153,88,37,140,40.6,1.174,39,0
12,100,84,33,105,30,0.488,46,0
1,147,94,41,0,49.3,0.358,27,1
1,81,74,41,57,46.3,1.096,32,0
3,187,70,22,200,36.4,0.408,36,1
6,162,62,0,0,24.3,0.178,50,1
4,136,70,0,0,31.2,1.182,22,1
1,121,78,39,74,39,0.261,28,0
3,108,62,24,0,26,0.223,25,0
0,181,88,44,510,43.3,0.222,26,1
8,154,78,32,0,32.4,0.443,45,1
1,128,88,39,110,36.5,1.057,37,1
7,137,90,41,0,32,0.391,39,0
0,123,72,0,0,36.3,0.258,52,1
1,106,76,0,0,37.5,0.197,26,0

6,190,92,0,0,35.5,0.278,66,1
2,88,58,26,16,28.4,0.766,22,0
9,170,74,31,0,44,0.403,43,1
9,89,62,0,0,22.5,0.142,33,0
10,101,76,48,180,32.9,0.171,63,0
2,122,70,27,0,36.8,0.34,27,0
5,121,72,23,112,26.2,0.245,30,0
1,126,60,0,0,30.1,0.349,47,1
1,93,70,31,0,30.4,0.315,23,0

6. Assuming a set of documents that need to be classified, use the naïve Bayesian Classifier model to perform this task. Built-in Java classes/API can be used to write the program. Calculate the accuracy, precision, and recall for your data set.

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn import metrics

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

#splitting the dataset into train and test data
xtrain,xtest,ytrain,ytest=train_test_split(X,y)
print('\n the total number of Training Data :',ytrain.shape)
print ('\n the total number of Test Data :',ytest.shape)

#output the words or Tokens in the text documents
cv = CountVectorizer()
xtrain_dtm = cv.fit_transform(xtrain)
xtest_dtm=cv.transform(xtest)
print('\n The words or Tokens in the text documents \n')
print(cv.get_feature_names())
df=pd.DataFrame(xtrain_dtm.toarray(),columns=cv.get_feature_names())

# Training Naive Bayes (NB) classifier on training data.
clf = MultinomialNB().fit(xtrain_dtm,ytrain)
predicted = clf.predict(xtest_dtm)

#printing accuracy, Confusion matrix, Precision and Recall
print('\n Accuracy of the classifier is',metrics.accuracy_score(ytest,predicted))
print('\n Confusion matrix')
```

```
print(metrics.confusion_matrix(ytest,predicted))
print('\n The value of Precision', metrics.precision_score(ytest,predicted))
print('\n The value of Recall', metrics.recall_score(ytest,predicted))
```

naive.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
```

7. Write a program to construct a Bayesian network considering medical data. Use this model to demonstrate the diagnosis of heart patients using standard Heart Disease Data Set. You can use Java/Python ML library classes/API.

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

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

print('Few examples from the dataset are given below')
print(heartDisease.head())

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

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

print('\nInferencing with Bayesian Network:')
HeartDisease_infer = VariableElimination(model)

print('\n 1. Probability of HeartDisease given Age=28')
q=HeartDisease_infer.query(variables=['heartdisease'], evidence={'age':28})
print(q['heartdisease'])

print('\n 2. Probability of HeartDisease given cholesterol=100')
q=HeartDisease_infer.query(variables=['heartdisease'], evidence={'chol':100})
print(q['heartdisease'])

heart.csv
```

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

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
48,1,2,130,245,0,2,180,0,0.2,2,0,3,0
58,1,4,150,270,0,2,111,1,0.8,1,0,7,3
45,1,4,104,208,0,2,148,1,3,2,0,3,0
53,0,4,130,264,0,2,143,0,0.4,2,0,3,0
39,1,3,140,321,0,2,182,0,0,1,0,3,0
68,1,3,180,274,1,2,150,1,1.6,2,0,7,3
52,1,2,120,325,0,0,172,0,0.2,1,0,3,0
44,1,3,140,235,0,2,180,0,0,1,0,3,0
47,1,3,138,257,0,2,156,0,0,1,0,3,0

53,0,3,128,216,0,2,115,0,0,1,0,?,0
53,0,4,138,234,0,2,160,0,0,1,0,3,0
51,0,3,130,256,0,2,149,0,0,5,1,0,3,0
66,1,4,120,302,0,2,151,0,0,4,2,0,3,0
62,0,4,160,164,0,2,145,0,6,2,3,3,7,3
62,1,3,130,231,0,0,146,0,1,8,2,3,7,0
44,0,3,108,141,0,0,175,0,0,6,2,0,3,0
63,0,3,135,252,0,2,172,0,0,1,0,3,0
52,1,4,128,255,0,0,161,1,0,1,1,7,1
59,1,4,110,239,0,2,142,1,1,2,2,1,7,2
60,0,4,150,258,0,2,157,0,2,6,2,2,7,3
52,1,2,134,201,0,0,158,0,0,8,1,1,3,0
48,1,4,122,222,0,2,186,0,0,1,0,3,0
45,1,4,115,260,0,2,185,0,0,1,0,3,0
34,1,1,118,182,0,2,174,0,0,1,0,3,0
57,0,4,128,303,0,2,159,0,0,1,1,3,0
71,0,3,110,265,1,2,130,0,0,1,1,3,0
49,1,3,120,188,0,0,139,0,2,2,3,7,3
54,1,2,108,309,0,0,156,0,0,1,0,7,0
59,1,4,140,177,0,0,162,1,0,1,1,7,2
57,1,3,128,229,0,2,150,0,0,4,2,1,7,1
61,1,4,120,260,0,0,140,1,3,6,2,1,7,2
39,1,4,118,219,0,0,140,0,1,2,2,0,7,3
61,0,4,145,307,0,2,146,1,1,2,0,7,1
56,1,4,125,249,1,2,144,1,1,2,2,1,3,1
52,1,1,118,186,0,2,190,0,0,2,0,6,0
43,0,4,132,341,1,2,136,1,3,2,0,7,2
62,0,3,130,263,0,0,97,0,1,2,2,1,7,2
41,1,2,135,203,0,0,132,0,0,2,0,6,0
58,1,3,140,211,1,2,165,0,0,1,0,3,0
35,0,4,138,183,0,0,182,0,1,4,1,0,3,0
63,1,4,130,330,1,2,132,1,1,8,1,3,7,3
65,1,4,135,254,0,2,127,0,2,8,2,1,7,2
48,1,4,130,256,1,2,150,1,0,1,2,7,3
63,0,4,150,407,0,2,154,0,4,2,3,7,4
51,1,3,100,222,0,0,143,1,1,2,2,0,3,0
55,1,4,140,217,0,0,111,1,5,6,3,0,7,3
65,1,1,138,282,1,2,174,0,1,4,2,1,3,1
45,0,2,130,234,0,2,175,0,0,6,2,0,3,0
56,0,4,200,288,1,2,133,1,4,3,2,7,3
54,1,4,110,239,0,0,126,1,2,8,2,1,7,3
44,1,2,120,220,0,0,170,0,0,1,0,3,0
62,0,4,124,209,0,0,163,0,0,1,0,3,0
54,1,3,120,258,0,2,147,0,0,4,2,0,7,0

51,1,3,94,227,0,0,154,1,0,1,1,7,0
29,1,2,130,204,0,2,202,0,0,1,0,3,0
51,1,4,140,261,0,2,186,1,0,1,0,3,0
43,0,3,122,213,0,0,165,0,0,2,2,0,3,0
55,0,2,135,250,0,2,161,0,1,4,2,0,3,0
70,1,4,145,174,0,0,125,1,2,6,3,0,7,4
62,1,2,120,281,0,2,103,0,1,4,2,1,7,3
35,1,4,120,198,0,0,130,1,1,6,2,0,7,1
51,1,3,125,245,1,2,166,0,2,4,2,0,3,0
59,1,2,140,221,0,0,164,1,0,1,0,3,0
59,1,1,170,288,0,2,159,0,0,2,2,0,7,1
52,1,2,128,205,1,0,184,0,0,1,0,3,0
64,1,3,125,309,0,0,131,1,1,8,2,0,7,1
58,1,3,105,240,0,2,154,1,0,6,2,0,7,0
47,1,3,108,243,0,0,152,0,0,1,0,3,1
57,1,4,165,289,1,2,124,0,1,2,3,7,4
41,1,3,112,250,0,0,179,0,0,1,0,3,0
45,1,2,128,308,0,2,170,0,0,1,0,3,0
60,0,3,102,318,0,0,160,0,0,1,1,3,0
52,1,1,152,298,1,0,178,0,1,2,2,0,7,0
42,0,4,102,265,0,2,122,0,0,6,2,0,3,0
67,0,3,115,564,0,2,160,0,1,6,2,0,7,0
55,1,4,160,289,0,2,145,1,0,8,2,1,7,4
64,1,4,120,246,0,2,96,1,2,2,3,1,3,3
70,1,4,130,322,0,2,109,0,2,4,2,3,3,1
51,1,4,140,299,0,0,173,1,1,6,1,0,7,1
58,1,4,125,300,0,2,171,0,0,1,2,7,1
60,1,4,140,293,0,2,170,0,1,2,2,2,7,2
68,1,3,118,277,0,0,151,0,1,1,1,7,0
46,1,2,101,197,1,0,156,0,0,1,0,7,0
77,1,4,125,304,0,2,162,1,0,1,3,3,4
54,0,3,110,214,0,0,158,0,1,6,2,0,3,0
58,0,4,100,248,0,2,122,0,1,2,0,3,0
48,1,3,124,255,1,0,175,0,0,1,2,3,0
57,1,4,132,207,0,0,168,1,0,1,0,7,0
52,1,3,138,223,0,0,169,0,0,1,?,3,0
54,0,2,132,288,1,2,159,1,0,1,1,3,0
35,1,4,126,282,0,2,156,1,0,1,0,7,1
45,0,2,112,160,0,0,138,0,0,2,0,3,0
70,1,3,160,269,0,0,112,1,2,9,2,1,7,3
53,1,4,142,226,0,2,111,1,0,1,0,7,0
59,0,4,174,249,0,0,143,1,0,2,0,3,1
62,0,4,140,394,0,2,157,0,1,2,2,0,3,0
64,1,4,145,212,0,2,132,0,2,2,2,6,4

57,1,4,152,274,0,0,88,1,1.2,2,1,7,1
52,1,4,108,233,1,0,147,0,0.1,1,3,7,0
56,1,4,132,184,0,2,105,1,2.1,2,1,6,1
43,1,3,130,315,0,0,162,0,1.9,1,1,3,0
53,1,3,130,246,1,2,173,0,0,1,3,3,0
48,1,4,124,274,0,2,166,0,0.5,2,0,7,3
56,0,4,134,409,0,2,150,1,1.9,2,2,7,2
42,1,1,148,244,0,2,178,0,0.8,1,2,3,0
59,1,1,178,270,0,2,145,0,4.2,3,0,7,0
60,0,4,158,305,0,2,161,0,0,1,0,3,1
63,0,2,140,195,0,0,179,0,0,1,2,3,0
42,1,3,120,240,1,0,194,0,0.8,3,0,7,0
66,1,2,160,246,0,0,120,1,0,2,3,6,2
54,1,2,192,283,0,2,195,0,0,1,1,7,1
69,1,3,140,254,0,2,146,0,2,2,3,7,2
50,1,3,129,196,0,0,163,0,0,1,0,3,0
51,1,4,140,298,0,0,122,1,4.2,2,3,7,3
43,1,4,132,247,1,2,143,1,0.1,2,?,7,1
62,0,4,138,294,1,0,106,0,1.9,2,3,3,2
68,0,3,120,211,0,2,115,0,1.5,2,0,3,0
67,1,4,100,299,0,2,125,1,0.9,2,2,3,3
69,1,1,160,234,1,2,131,0,0.1,2,1,3,0
45,0,4,138,236,0,2,152,1,0.2,2,0,3,0
50,0,2,120,244,0,0,162,0,1.1,1,0,3,0
59,1,1,160,273,0,2,125,0,0,1,0,3,1
50,0,4,110,254,0,2,159,0,0,1,0,3,0
64,0,4,180,325,0,0,154,1,0,1,0,3,0
57,1,3,150,126,1,0,173,0,0.2,1,1,7,0
64,0,3,140,313,0,0,133,0,0.2,1,0,7,0
43,1,4,110,211,0,0,161,0,0,1,0,7,0
45,1,4,142,309,0,2,147,1,0,2,3,7,3
58,1,4,128,259,0,2,130,1,3,2,2,7,3
50,1,4,144,200,0,2,126,1,0.9,2,0,7,3
55,1,2,130,262,0,0,155,0,0,1,0,3,0
62,0,4,150,244,0,0,154,1,1.4,2,0,3,1
37,0,3,120,215,0,0,170,0,0,1,0,3,0
38,1,1,120,231,0,0,182,1,3.8,2,0,7,4
41,1,3,130,214,0,2,168,0,2,2,0,3,0
66,0,4,178,228,1,0,165,1,1,2,2,7,3
52,1,4,112,230,0,0,160,0,0,1,1,3,1
56,1,1,120,193,0,2,162,0,1.9,2,0,7,0
46,0,2,105,204,0,0,172,0,0,1,0,3,0
46,0,4,138,243,0,2,152,1,0,2,0,3,0
64,0,4,130,303,0,0,122,0,2,2,2,3,0

59,1,4,138,271,0,2,182,0,0,1,0,3,0
41,0,3,112,268,0,2,172,1,0,1,0,3,0
54,0,3,108,267,0,2,167,0,0,1,0,3,0
39,0,3,94,199,0,0,179,0,0,1,0,3,0
53,1,4,123,282,0,0,95,1,2,2,2,7,3
63,0,4,108,269,0,0,169,1,1.8,2,2,3,1
34,0,2,118,210,0,0,192,0,0.7,1,0,3,0
47,1,4,112,204,0,0,143,0,0.1,1,0,3,0
67,0,3,152,277,0,0,172,0,0,1,1,3,0
54,1,4,110,206,0,2,108,1,0,2,1,3,3
66,1,4,112,212,0,2,132,1,0.1,1,1,3,2
52,0,3,136,196,0,2,169,0,0.1,2,0,3,0
55,0,4,180,327,0,1,117,1,3.4,2,0,3,2
49,1,3,118,149,0,2,126,0,0.8,1,3,3,1
74,0,2,120,269,0,2,121,1,0.2,1,1,3,0
54,0,3,160,201,0,0,163,0,0,1,1,3,0
54,1,4,122,286,0,2,116,1,3.2,2,2,3,3
56,1,4,130,283,1,2,103,1,1.6,3,0,7,2
46,1,4,120,249,0,2,144,0,0.8,1,0,7,1
49,0,2,134,271,0,0,162,0,0,2,0,3,0
42,1,2,120,295,0,0,162,0,0,1,0,3,0
41,1,2,110,235,0,0,153,0,0,1,0,3,0
41,0,2,126,306,0,0,163,0,0,1,0,3,0
49,0,4,130,269,0,0,163,0,0,1,0,3,0
61,1,1,134,234,0,0,145,0,2.6,2,2,3,2
60,0,3,120,178,1,0,96,0,0,1,0,3,0
67,1,4,120,237,0,0,71,0,1,2,0,3,2
58,1,4,100,234,0,0,156,0,0.1,1,1,7,2
47,1,4,110,275,0,2,118,1,1,2,1,3,1
52,1,4,125,212,0,0,168,0,1,1,2,7,3
62,1,2,128,208,1,2,140,0,0,1,0,3,0
57,1,4,110,201,0,0,126,1,1.5,2,0,6,0
58,1,4,146,218,0,0,105,0,2,2,1,7,1
64,1,4,128,263,0,0,105,1,0.2,2,1,7,0
51,0,3,120,295,0,2,157,0,0.6,1,0,3,0
43,1,4,115,303,0,0,181,0,1.2,2,0,3,0
42,0,3,120,209,0,0,173,0,0,2,0,3,0
67,0,4,106,223,0,0,142,0,0.3,1,2,3,0
76,0,3,140,197,0,1,116,0,1.1,2,0,3,0
70,1,2,156,245,0,2,143,0,0,1,0,3,0
57,1,2,124,261,0,0,141,0,0.3,1,0,7,1
44,0,3,118,242,0,0,149,0,0.3,2,1,3,0
58,0,2,136,319,1,2,152,0,0,1,2,3,3
60,0,1,150,240,0,0,171,0,0.9,1,0,3,0

44,1,3,120,226,0,0,169,0,0,1,0,3,0
61,1,4,138,166,0,2,125,1,3.6,2,1,3,4
42,1,4,136,315,0,0,125,1,1.8,2,0,6,2
52,1,4,128,204,1,0,156,1,1,2,0,?,2
59,1,3,126,218,1,0,134,0,2.2,2,1,6,2
40,1,4,152,223,0,0,181,0,0,1,0,7,1
42,1,3,130,180,0,0,150,0,0,1,0,3,0
61,1,4,140,207,0,2,138,1,1.9,1,1,7,1
66,1,4,160,228,0,2,138,0,2.3,1,0,6,0
46,1,4,140,311,0,0,120,1,1.8,2,2,7,2
71,0,4,112,149,0,0,125,0,1.6,2,0,3,0
59,1,1,134,204,0,0,162,0,0.8,1,2,3,1
64,1,1,170,227,0,2,155,0,0.6,2,0,7,0
66,0,3,146,278,0,2,152,0,0,2,1,3,0
39,0,3,138,220,0,0,152,0,0,2,0,3,0
57,1,2,154,232,0,2,164,0,0,1,1,3,1
58,0,4,130,197,0,0,131,0,0.6,2,0,3,0
57,1,4,110,335,0,0,143,1,3,2,1,7,2
47,1,3,130,253,0,0,179,0,0,1,0,3,0
55,0,4,128,205,0,1,130,1,2,2,1,7,3
35,1,2,122,192,0,0,174,0,0,1,0,3,0
61,1,4,148,203,0,0,161,0,0,1,1,7,2
58,1,4,114,318,0,1,140,0,4.4,3,3,6,4
58,0,4,170,225,1,2,146,1,2.8,2,2,6,2
58,1,2,125,220,0,0,144,0,0.4,2,?,7,0
56,1,2,130,221,0,2,163,0,0,1,0,7,0
56,1,2,120,240,0,0,169,0,0,3,0,3,0
67,1,3,152,212,0,2,150,0,0.8,2,0,7,1
55,0,2,132,342,0,0,166,0,1.2,1,0,3,0
44,1,4,120,169,0,0,144,1,2.8,3,0,6,2
63,1,4,140,187,0,2,144,1,4,1,2,7,2
63,0,4,124,197,0,0,136,1,0,2,0,3,1
41,1,2,120,157,0,0,182,0,0,1,0,3,0
59,1,4,164,176,1,2,90,0,1,2,2,6,3
57,0,4,140,241,0,0,123,1,0.2,2,0,7,1
45,1,1,110,264,0,0,132,0,1.2,2,0,7,1
68,1,4,144,193,1,0,141,0,3.4,2,2,7,2
57,1,4,130,131,0,0,115,1,1.2,2,1,7,3
57,0,2,130,236,0,2,174,0,0,2,1,3,1

- Apply EM algorithm to cluster a set of data stored in a .CSV file. Use the same data set for clustering using k-Means algorithm. Compare the results of these two algorithms and comment on the quality of clustering. You can add Java/Python ML library classes/API in the program.

```
import matplotlib.pyplot as plt
from sklearn import datasets
from sklearn.cluster import KMeans
import sklearn.metrics as sm
import pandas as pd
import numpy as np
#import matplotlib inline

iris = datasets.load_iris()

X = pd.DataFrame(iris.data)
X.columns = ['Sepal_Length', 'Sepal_Width', 'Petal_Length', 'Petal_Width']

y = pd.DataFrame(iris.target)
y.columns = ['Targets']

#colormap = np.array(['red', 'lime', 'black'])

# K Means Cluster
model = KMeans(n_clusters=3)
model.fit(X)
# This is what KMeans thought
model.labels_

# View the results

# Set the size of the plot
plt.figure(figsize=(14,7))

# Create a colormap
colormap = np.array(['red', 'lime', 'black'])

# Plot the Original Classifications
```

```

plt.subplot(1, 2, 1)
plt.scatter(X.Petal_Length, X.Petal_Width, c=colormap[y.Targets], s=40)
plt.title('Real Classification')

# Plot the Models Classifications
plt.subplot(1, 2, 2)
plt.scatter(X.Petal_Length, X.Petal_Width, c=colormap[model.labels_], s=40)
plt.title('K Mean Classification')

# View the results
# Set the size of the plot
plt.figure(figsize=(14,7))
# Create a colormap
#print('The accuracy score : ',sm.accuracy_score(y, model.labels_))
#sm.confusion_matrix(y, model.labels_)

predY = np.choose(model.labels_, [0, 1, 2]).astype(np.int64)
print (predY)

#colormap = np.array(['red', 'lime', 'black'])
# Plot Orginal
plt.subplot(1, 2, 1)
plt.scatter(X.Petal_Length, X.Petal_Width, c=colormap[y.Targets], s=40)
plt.title('Real Classification')
# Plot Predicted with corrected values
plt.subplot(1, 2, 2)
plt.scatter(X.Petal_Length,X.Petal_Width, c=colormap[predY], s=40)
plt.title('K Mean Classification')

print('The accuracy score of K-Mean: ',sm.accuracy_score(y, model.labels_))
print('The Confusion matrixof K-Mean: ',sm.confusion_matrix(y, model.labels_))

from sklearn import preprocessing
scaler = preprocessing.StandardScaler()
scaler.fit(X)
xsa = scaler.transform(X)
xs = pd.DataFrame(xsa, columns = X.columns)
#xs.sample(5)

from sklearn.mixture import GaussianMixture
gmm = GaussianMixture(n_components=3)
gmm.fit(xs)

y_cluster_gmm = gmm.predict(xs)

```

```
#y_cluster_gmm

plt.subplot(2, 2, 3)
plt.scatter(X.Petal_Length, X.Petal_Width, c=colormap[y_cluster_gmm], s=40)
plt.title('GMM Classification')

print('The accuracy score of EM: ',sm.accuracy_score(y, y_cluster_gmm))
print('The Confusion matrix of EM: ',sm.confusion_matrix(y, y_cluster_gmm))
```

9. Write a program to implement k-Nearest Neighbour algorithm to classify the iris data set. Print both correct and wrong predictions. Java/Python ML library classes can be used for this problem.

```
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));
```


10. Implement the non-parametric Locally Weighted Regression algorithm in order to fit data points. Select appropriate data set for your experiment and draw graphs

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# kernel smoothing function
def kernel(point, xmat, k):
    m,n = np.shape(xmat)
    weights = np.mat(np.eye((m)))

    for j in range(m):
        diff = point - X[j]
        weights[j, j] = np.exp(diff * diff.T / (-2.0 * k**2))

    return weights

# function to return local weight of each training example
def localWeight(point, xmat, ymat, k):
    wt = kernel(point, xmat, k)
    W = (X.T * (wt*X)).I * (X.T * wt * ymat.T)
    return W

# root function that drives the algorithm
def localWeightRegression(xmat, ymat, k):
    m,n = np.shape(xmat)
    ypred = np.zeros(m)

    for i in range(m):
        ypred[i] = xmat[i] * localWeight(xmat[i], xmat, ymat, k)

    return ypred

#import data
data = pd.read_csv('tips.csv')

# place them in suitable data types
```

```

colA = np.array(data.total_bill)
colB = np.array(data.tip)

mcolA = np.mat(colA)
mcolB = np.mat(colB)

m = np.shape(mcolB)[1]
one = np.ones((1, m), dtype = int)

# horizontal stacking
X = np.hstack((one.T, mcolA.T))
print(X.shape)

# predicting values using LWLR
ypred = localWeightRegression(X, mcolB, 0.8)

# plotting the predicted graph
xsort = X.copy()
xsort.sort(axis=0)
plt.scatter(colA, colB, color='blue')
plt.plot(xsort[:, 1], ypred[X[:, 1].argsort(0)], color='yellow', linewidth=5)
plt.xlabel('Total Bill')
plt.ylabel('Tip')
plt.show()

```

tips.csv

```

total_bill,tip,sex,smoker,day,time,size
16.99,1.01,Female,No,Sun,Dinner,2
10.34,1.66,Male,No,Sun,Dinner,3
21.01,3.5,Male,No,Sun,Dinner,3
23.68,3.31,Male,No,Sun,Dinner,2
24.59,3.61,Female,No,Sun,Dinner,4
25.29,4.71,Male,No,Sun,Dinner,4
8.77,2.0,Male,No,Sun,Dinner,2
26.88,3.12,Male,No,Sun,Dinner,4
15.04,1.96,Male,No,Sun,Dinner,2
14.78,3.23,Male,No,Sun,Dinner,2
10.27,1.71,Male,No,Sun,Dinner,2
35.26,5.0,Female,No,Sun,Dinner,4
15.42,1.57,Male,No,Sun,Dinner,2
18.43,3.0,Male,No,Sun,Dinner,4

```

14.83,3.02,Female,No,Sun,Dinner,2
21.58,3.92,Male,No,Sun,Dinner,2
10.33,1.67,Female,No,Sun,Dinner,3
16.29,3.71,Male,No,Sun,Dinner,3
16.97,3.5,Female,No,Sun,Dinner,3
20.65,3.35,Male,No,Sat,Dinner,3
17.92,4.08,Male,No,Sat,Dinner,2
20.29,2.75,Female,No,Sat,Dinner,2
15.77,2.23,Female,No,Sat,Dinner,2
39.42,7.58,Male,No,Sat,Dinner,4
19.82,3.18,Male,No,Sat,Dinner,2
17.81,2.34,Male,No,Sat,Dinner,4
13.37,2.0,Male,No,Sat,Dinner,2
12.69,2.0,Male,No,Sat,Dinner,2
21.7,4.3,Male,No,Sat,Dinner,2
19.65,3.0,Female,No,Sat,Dinner,2
9.55,1.45,Male,No,Sat,Dinner,2
18.35,2.5,Male,No,Sat,Dinner,4
15.06,3.0,Female,No,Sat,Dinner,2
20.69,2.45,Female,No,Sat,Dinner,4
17.78,3.27,Male,No,Sat,Dinner,2
24.06,3.6,Male,No,Sat,Dinner,3
16.31,2.0,Male,No,Sat,Dinner,3
16.93,3.07,Female,No,Sat,Dinner,3
18.69,2.31,Male,No,Sat,Dinner,3
31.27,5.0,Male,No,Sat,Dinner,3
16.04,2.24,Male,No,Sat,Dinner,3
17.46,2.54,Male,No,Sun,Dinner,2
13.94,3.06,Male,No,Sun,Dinner,2
9.68,1.32,Male,No,Sun,Dinner,2
30.4,5.6,Male,No,Sun,Dinner,4
18.29,3.0,Male,No,Sun,Dinner,2
22.23,5.0,Male,No,Sun,Dinner,2
32.4,6.0,Male,No,Sun,Dinner,4
28.55,2.05,Male,No,Sun,Dinner,3
18.04,3.0,Male,No,Sun,Dinner,2
12.54,2.5,Male,No,Sun,Dinner,2
10.29,2.6,Female,No,Sun,Dinner,2
34.81,5.2,Female,No,Sun,Dinner,4
9.94,1.56,Male,No,Sun,Dinner,2
25.56,4.34,Male,No,Sun,Dinner,4
19.49,3.51,Male,No,Sun,Dinner,2
38.01,3.0,Male,Yes,Sat,Dinner,4
26.41,1.5,Female,No,Sat,Dinner,2

11.24,1.76,Male,Yes,Sat,Dinner,2
48.27,6.73,Male,No,Sat,Dinner,4
20.29,3.21,Male,Yes,Sat,Dinner,2
13.81,2.0,Male,Yes,Sat,Dinner,2
11.02,1.98,Male,Yes,Sat,Dinner,2
18.29,3.76,Male,Yes,Sat,Dinner,4
17.59,2.64,Male,No,Sat,Dinner,3
20.08,3.15,Male,No,Sat,Dinner,3
16.45,2.47,Female,No,Sat,Dinner,2
3.07,1.0,Female,Yes,Sat,Dinner,1
20.23,2.01,Male,No,Sat,Dinner,2
15.01,2.09,Male,Yes,Sat,Dinner,2
12.02,1.97,Male,No,Sat,Dinner,2
17.07,3.0,Female,No,Sat,Dinner,3
26.86,3.14,Female,Yes,Sat,Dinner,2
25.28,5.0,Female,Yes,Sat,Dinner,2
14.73,2.2,Female,No,Sat,Dinner,2
10.51,1.25,Male,No,Sat,Dinner,2
17.92,3.08,Male,Yes,Sat,Dinner,2
27.2,4.0,Male,No,Thur,Lunch,4
22.76,3.0,Male,No,Thur,Lunch,2
17.29,2.71,Male,No,Thur,Lunch,2
19.44,3.0,Male,Yes,Thur,Lunch,2
16.66,3.4,Male,No,Thur,Lunch,2
10.07,1.83,Female,No,Thur,Lunch,1
32.68,5.0,Male,Yes,Thur,Lunch,2
15.98,2.03,Male,No,Thur,Lunch,2
34.83,5.17,Female,No,Thur,Lunch,4
13.03,2.0,Male,No,Thur,Lunch,2
18.28,4.0,Male,No,Thur,Lunch,2
24.71,5.85,Male,No,Thur,Lunch,2
21.16,3.0,Male,No,Thur,Lunch,2
28.97,3.0,Male,Yes,Fri,Dinner,2
22.49,3.5,Male,No,Fri,Dinner,2
5.75,1.0,Female,Yes,Fri,Dinner,2
16.32,4.3,Female,Yes,Fri,Dinner,2
22.75,3.25,Female,No,Fri,Dinner,2
40.17,4.73,Male,Yes,Fri,Dinner,4
27.28,4.0,Male,Yes,Fri,Dinner,2
12.03,1.5,Male,Yes,Fri,Dinner,2
21.01,3.0,Male,Yes,Fri,Dinner,2
12.46,1.5,Male,No,Fri,Dinner,2
11.35,2.5,Female,Yes,Fri,Dinner,2
15.38,3.0,Female,Yes,Fri,Dinner,2

44.3,2.5,Female,Yes,Sat,Dinner,3
22.42,3.48,Female,Yes,Sat,Dinner,2
20.92,4.08,Female,No,Sat,Dinner,2
15.36,1.64,Male,Yes,Sat,Dinner,2
20.49,4.06,Male,Yes,Sat,Dinner,2
25.21,4.29,Male,Yes,Sat,Dinner,2
18.24,3.76,Male,No,Sat,Dinner,2
14.31,4.0,Female,Yes,Sat,Dinner,2
14.0,3.0,Male,No,Sat,Dinner,2
7.25,1.0,Female,No,Sat,Dinner,1
38.07,4.0,Male,No,Sun,Dinner,3
23.95,2.55,Male,No,Sun,Dinner,2
25.71,4.0,Female,No,Sun,Dinner,3
17.31,3.5,Female,No,Sun,Dinner,2
29.93,5.07,Male,No,Sun,Dinner,4
10.65,1.5,Female,No,Thur,Lunch,2
12.43,1.8,Female,No,Thur,Lunch,2
24.08,2.92,Female,No,Thur,Lunch,4
11.69,2.31,Male,No,Thur,Lunch,2
13.42,1.68,Female,No,Thur,Lunch,2
14.26,2.5,Male,No,Thur,Lunch,2
15.95,2.0,Male,No,Thur,Lunch,2
12.48,2.52,Female,No,Thur,Lunch,2
29.8,4.2,Female,No,Thur,Lunch,6
8.52,1.48,Male,No,Thur,Lunch,2
14.52,2.0,Female,No,Thur,Lunch,2
11.38,2.0,Female,No,Thur,Lunch,2
22.82,2.18,Male,No,Thur,Lunch,3
19.08,1.5,Male,No,Thur,Lunch,2
20.27,2.83,Female,No,Thur,Lunch,2
11.17,1.5,Female,No,Thur,Lunch,2
12.26,2.0,Female,No,Thur,Lunch,2
18.26,3.25,Female,No,Thur,Lunch,2
8.51,1.25,Female,No,Thur,Lunch,2
10.33,2.0,Female,No,Thur,Lunch,2
14.15,2.0,Female,No,Thur,Lunch,2
16.0,2.0,Male,Yes,Thur,Lunch,2
13.16,2.75,Female,No,Thur,Lunch,2
17.47,3.5,Female,No,Thur,Lunch,2
34.3,6.7,Male,No,Thur,Lunch,6
41.19,5.0,Male,No,Thur,Lunch,5
27.05,5.0,Female,No,Thur,Lunch,6
16.43,2.3,Female,No,Thur,Lunch,2
8.35,1.5,Female,No,Thur,Lunch,2

18.64,1.36,Female,No,Thur,Lunch,3
11.87,1.63,Female,No,Thur,Lunch,2
9.78,1.73,Male,No,Thur,Lunch,2
7.51,2.0,Male,No,Thur,Lunch,2
14.07,2.5,Male,No,Sun,Dinner,2
13.13,2.0,Male,No,Sun,Dinner,2
17.26,2.74,Male,No,Sun,Dinner,3
24.55,2.0,Male,No,Sun,Dinner,4
19.77,2.0,Male,No,Sun,Dinner,4
29.85,5.14,Female,No,Sun,Dinner,5
48.17,5.0,Male,No,Sun,Dinner,6
25.0,3.75,Female,No,Sun,Dinner,4
13.39,2.61,Female,No,Sun,Dinner,2
16.49,2.0,Male,No,Sun,Dinner,4
21.5,3.5,Male,No,Sun,Dinner,4
12.66,2.5,Male,No,Sun,Dinner,2
16.21,2.0,Female,No,Sun,Dinner,3
13.81,2.0,Male,No,Sun,Dinner,2
17.51,3.0,Female,Yes,Sun,Dinner,2
24.52,3.48,Male,No,Sun,Dinner,3
20.76,2.24,Male,No,Sun,Dinner,2
31.71,4.5,Male,No,Sun,Dinner,4
10.59,1.61,Female,Yes,Sat,Dinner,2
10.63,2.0,Female,Yes,Sat,Dinner,2
50.81,10.0,Male,Yes,Sat,Dinner,3
15.81,3.16,Male,Yes,Sat,Dinner,2
7.25,5.15,Male,Yes,Sun,Dinner,2
31.85,3.18,Male,Yes,Sun,Dinner,2
16.82,4.0,Male,Yes,Sun,Dinner,2
32.9,3.11,Male,Yes,Sun,Dinner,2
17.89,2.0,Male,Yes,Sun,Dinner,2
14.48,2.0,Male,Yes,Sun,Dinner,2
9.6,4.0,Female,Yes,Sun,Dinner,2
34.63,3.55,Male,Yes,Sun,Dinner,2
34.65,3.68,Male,Yes,Sun,Dinner,4
23.33,5.65,Male,Yes,Sun,Dinner,2
45.35,3.5,Male,Yes,Sun,Dinner,3
23.17,6.5,Male,Yes,Sun,Dinner,4
40.55,3.0,Male,Yes,Sun,Dinner,2
20.69,5.0,Male,No,Sun,Dinner,5
20.9,3.5,Female,Yes,Sun,Dinner,3
30.46,2.0,Male,Yes,Sun,Dinner,5
18.15,3.5,Female,Yes,Sun,Dinner,3
23.1,4.0,Male,Yes,Sun,Dinner,3

15.69,1.5, Male, Yes, Sun, Dinner, 2
19.81,4.19, Female, Yes, Thur, Lunch, 2
28.44,2.56, Male, Yes, Thur, Lunch, 2
15.48,2.02, Male, Yes, Thur, Lunch, 2
16.58,4.0, Male, Yes, Thur, Lunch, 2
7.56,1.44, Male, No, Thur, Lunch, 2
10.34,2.0, Male, Yes, Thur, Lunch, 2
43.11,5.0, Female, Yes, Thur, Lunch, 4
13.0,2.0, Female, Yes, Thur, Lunch, 2
13.51,2.0, Male, Yes, Thur, Lunch, 2
18.71,4.0, Male, Yes, Thur, Lunch, 3
12.74,2.01, Female, Yes, Thur, Lunch, 2
13.0,2.0, Female, Yes, Thur, Lunch, 2
16.4,2.5, Female, Yes, Thur, Lunch, 2
20.53,4.0, Male, Yes, Thur, Lunch, 4
16.47,3.23, Female, Yes, Thur, Lunch, 3
26.59,3.41, Male, Yes, Sat, Dinner, 3
38.73,3.0, Male, Yes, Sat, Dinner, 4
24.27,2.03, Male, Yes, Sat, Dinner, 2
12.76,2.23, Female, Yes, Sat, Dinner, 2
30.06,2.0, Male, Yes, Sat, Dinner, 3
25.89,5.16, Male, Yes, Sat, Dinner, 4
48.33,9.0, Male, No, Sat, Dinner, 4
13.27,2.5, Female, Yes, Sat, Dinner, 2
28.17,6.5, Female, Yes, Sat, Dinner, 3
12.9,1.1, Female, Yes, Sat, Dinner, 2
28.15,3.0, Male, Yes, Sat, Dinner, 5
11.59,1.5, Male, Yes, Sat, Dinner, 2
7.74,1.44, Male, Yes, Sat, Dinner, 2
30.14,3.09, Female, Yes, Sat, Dinner, 4
12.16,2.2, Male, Yes, Fri, Lunch, 2
13.42,3.48, Female, Yes, Fri, Lunch, 2
8.58,1.92, Male, Yes, Fri, Lunch, 1
15.98,3.0, Female, No, Fri, Lunch, 3
13.42,1.58, Male, Yes, Fri, Lunch, 2
16.27,2.5, Female, Yes, Fri, Lunch, 2
10.09,2.0, Female, Yes, Fri, Lunch, 2
20.45,3.0, Male, No, Sat, Dinner, 4
13.28,2.72, Male, No, Sat, Dinner, 2
22.12,2.88, Female, Yes, Sat, Dinner, 2
24.01,2.0, Male, Yes, Sat, Dinner, 4
15.69,3.0, Male, Yes, Sat, Dinner, 3
11.61,3.39, Male, No, Sat, Dinner, 2
10.77,1.47, Male, No, Sat, Dinner, 2

15.53,3.0,Male,Yes,Sat,Dinner,2
10.07,1.25,Male,No,Sat,Dinner,2
12.6,1.0,Male,Yes,Sat,Dinner,2
32.83,1.17,Male,Yes,Sat,Dinner,2
35.83,4.67,Female,No,Sat,Dinner,3
29.03,5.92,Male,No,Sat,Dinner,3
27.18,2.0,Female,Yes,Sat,Dinner,2
22.67,2.0,Male,Yes,Sat,Dinner,2
17.82,1.75,Male,No,Sat,Dinner,2
18.78,3.0,Female,No,Thur,Dinner,2