STATISTICAL METHODS IN AI
ASSIGNMENT4:
NAÏVE BAYES (NB) CLASSIFIER

Name – Megha Agarwal
Roll No – **201506511**
Course – M.Tech CSIS(PG1)
Dated : 10 March, 2016

AIM:

This assignment requires to implement the Naïve Bayes (NB) classifier and test it on two different datasets from the UCI Machine learning repository

DATA SETS DESCRIPTION :

Categorical Data:

Data Set : BREAST CANCER - WISCONSIN DATA SET

- ➔ Title: Bank Marketing (with social/economic context)
- ➔ Number of Instances: 41188 for bank-additional-full.csv
- ➔ Number of Attributes: 20 + output attribute.
- ➔ Attribute information:
  1 - age (numeric)
  2 - job : type of job
  3 - marital : marital status
  4 - education
  5 - default: has credit in default? (categorical: "no","yes","unknown")
  6 - housing: has housing loan? (categorical: "no","yes","unknown")
  7 - loan: has personal loan? (categorical: "no","yes","unknown")
  8 - contact: contact communication type
  9 - month: last contact month of year
  10 - day_of_week: last contact day of the week
  11 - duration: last contact duration, in seconds (numeric).
  12 - campaign: number of contacts performed during this campaign
  13 - pdays: number of days that passed by after the client
  14 - previous: number of contacts performed before this campaign
  15 - poutcome: outcome of the previous marketing campaign
  16 - emp.var.rate: employment variation rate

17 - cons.price.idx: consumer price index
18 - cons.conf.idx: consumer confidence index
19 - euribor3m: euribor 3 month rate - daily indicator (numeric)
20 - nr.employed: number of employees - quarterly indicator (numeric)
Output variable (desired target):
21 - y - has the client subscribed a term deposit? (binary: "yes","no")

- ➔ Missing Attribute Values: There are several missing values in some categorical attributes, all coded with the "unknown" label. These missing values are treated as a possible class label.
- ➔ For dataset to make continous, attributes with numbers 0, 10, 11, 12, 13, 15, 16, 17, 18, 19 are not taken as they are numeric attribute and not suitable for categorical data

Continous Data:

Data Set : BREAST CANCER - WISCONSIN DATA SET

➔ Title of Database: Wisconsin Breast Cancer Database (January 8, 1991)
➔ Number of Instances: (Number of Instances: 699 (as of 15 July 1992))
- ◆ Benign: 458 (65.5%)
- ◆ Malignant: 241 (34.5%)

➔ Number of Attributes: Number of Attributes: 10 plus the class attribute
➔ Attribute Information:
- ◆ Sample code number: id number
- ◆ Clump Thickness
- ◆ Uniformity of Cell Size
- ◆ Uniformity of Cell Shape
- ◆ Marginal Adhesion
- ◆ Single Epithelial Cell Size
- ◆ Bare Nuclei
- ◆ Bland Chromatin
- ◆ Normal Nucleoli
- ◆ Mitoses
- ◆ Class: (2 for benign, 4 for malignant)

➔ Missing Attribute Values: 16 (The ‘?’ in the missing attributes are replaced by the average value)

CODE (FOR ALL QUESTIONS)

```
import random
from matplotlib.pyplot import *
import csv
from math import *
import operator
from tabulate import tabulate

# Function to load the dataset and create training sample and test sample
# Divide the dataset into ratio 1:1

def read_file_load_dataset_categorical(filename):
	f = open(filename, 'rb')
	dataset = f.readlines()
	dataset = dataset[1:]
	length = len(dataset)
	random.shuffle(dataset)
	test, train, training_sample, test_sample =[], [], [], []
	list_not_to_take = [0, 10, 11, 12, 13, 15, 16, 17, 18, 19]

	#Spliting the dataset into training sample : list of lists
	for i in range(0, length/2):
		train = dataset[i].split(';')
		temp = list(train[len(train)-1])
		temp = temp[:-2]
		temp = "".join(temp)
		train[len(train)-1]=temp
		train_list=[]
		for j in range(len(train)):
			if j not in list_not_to_take:
				train_list.append(train[j])
		training_sample.append(train_list)
		train=[]
		train_list=[]

	#Spliting the dataset into test sample - with classname(for simplicity): list of lists
	for i in range((length/2), length):
		test = dataset[i].split(';')
		temp = list(test[len(test)-1])
		temp = temp[:-2]
		temp = "".join(temp)
		test[len(test)-1]=temp
		test_list=[]
		for j in range(len(test)):
			if j not in list_not_to_take:
				test_list.append(test[j])
		test_sample.append(test_list)
		test=[]
		test_list=[]
```

```
	return training_sample, test_sample

def read_file_load_dataset_continous(filename):
	f = open(filename, 'rb')
	dataset = f.readlines()
	length = len(dataset)
	random.shuffle(dataset)
	test, train, training_sample, test_sample =[], [], [], []

	#Spliting the dataset into training sample : list of lists
	for i in range(0, length/2):
		train = dataset[i].split(',')
		train = train[1:]
		for i in range(0, len(train)):
			if train[i]=='?':
				train[i]=5
			train[i]= float(train[i])
		training_sample.append(train)
		train=[]

	#Spliting the dataset into training sample : list of lists
	for i in range((length/2), length):
		test = dataset[i].split(',')
		test = test[1:]
		for i in range(0, len(test)):
			if test[i]=='?':
				test[i]=5
			test[i]= float(test[i])
		test_sample.append(test)
		test=[]

	return training_sample, test_sample


def forming_dictionary(training_sample):
	dict_training_sample=[]

	dict={}
	for i in range(len(training_sample[0])-1):
		dict_training_sample.append(dict)


	for j in range(len(dict_training_sample)):
		d_temp={}
		d_temp[1], d_temp[0]  = {}, {}
		d_temp_yes,d_temp_no ={}, {}

		for i in range(len(training_sample)):
			if   training_sample[i][j]   in   d_temp_yes   and   training_sample[i][-
```

```
1]=="'yes'":
                    d_temp_yes[training_sample[i][j]]+=1
                elif training_sample[i][j] in d_temp_no and training_sample[i]
[10]=="'no'":
                    d_temp_no[training_sample[i][j]]+=1
                else:
                    d_temp_yes[training_sample[i][j]]=1
                    d_temp_no[training_sample[i][j]]=1

            d_temp[1] = d_temp_yes
            d_temp[0] = d_temp_no
            dict_training_sample[j]=d_temp


    return dict_training_sample


def forming_dictionary_mean_std(training_sample):
    dict_training_sample = {}
    training_sample_2, training_sample_4 =[], []

    for i in range(len(training_sample)):
        if training_sample[i][len(training_sample[i])-1]==2.0:
            training_sample_2.append(training_sample[i])
        else:
            training_sample_4.append(training_sample[i])

    training_sample_4, training_sample_2 = np.array(training_sample_4),
np.array(training_sample_2)
    dict_training_sample={}

    l1, l2 =[], []
    for i in range(len(training_sample[0])-1):
        x = training_sample_2[:, i]
        y = training_sample_4[:, i]
        mean1, mean2 = np.mean(x), np.mean(y)
        std1, std2 = np.std(x), np.std(y)
        tup1, tup2 = (mean1, std1),(mean2, std2)
        l1.append(tup1)
        l2.append(tup2)

    dict_training_sample[2], dict_training_sample[4]=l1, l2
    return dict_training_sample


def calculating_class_label(dict_training_sample, test_sample, no_count, yes_count):
    y=1
    n=1
    temp1=1
    temp2=1
```

```
    prob_yes = float(yes_count)/float(yes_count+no_count)
    prob_no = float(no_count)/float(no_count+yes_count)

    for i in range(4, len(dict_training_sample)):
        if not dict_training_sample[i][1][test_sample[i]]:
            temp1=0
        else:
            temp1 = dict_training_sample[i][1][test_sample[i]]
            temp1 = temp1 / float(yes_count)
        if not dict_training_sample[i][0][test_sample[i]]:
            temp2=0
        else:
            temp2 = dict_training_sample[i][0][test_sample[i]]
            temp2 = temp2/ float(no_count)
        y = y*temp1
        n = n*temp2

    final_prob_yes= float(prob_yes)*float(y)
    final_prob_no = float(prob_no)*float(n)
    prob_no_final_list.append(final_prob_no)
    prob_yes_final_list.append(final_prob_yes)
    if final_prob_yes>final_prob_no:
        return '''yes'''
    else:
        return '''no'''

def calculating_class_label_continous(dict_training_sample, test_sample):
    list2 = dict_training_sample[2]
    list4 = dict_training_sample[4]
    p2 = 1.0
    p4 = 1.0
    for i in range(len(test_sample)):
        p2 = p2 * gaussian_function(list2[i][0], list2[i][1] ,test_sample[i])
        p4 = p4 * gaussian_function(list4[i][0], list4[i][1] ,test_sample[i])

    prob_no_final_list.append(p2)
    prob_yes_final_list.append(p4)
    if p2>p4:
        return 2.0
    else:
        return 4.0



def calculate_accuracy(predictions, actual_classes):
    correct=0
    for i in range(len(actual_classes)):
        if actual_classes[i] == predictions[i]:
            correct = correct+1
    accuracy_percentage = (correct/(float(len(actual_classes)))) * 100
```

```
        return accuracy_percentage

def calculating_class_probability(training_sample):
        D={}
        for i in range(len(training_sample)):
                if training_sample[i][len(training_sample[i])-1] in D:
                        D[training_sample[i][len(training_sample[i])-1]]+=1
                else:
                        D[training_sample[i][len(training_sample[i])-1]]=1
        yes_count = D["'yes'"]
        no_count =  D["'no'"]
        return yes_count, no_count

def confusion_matrix(predictions, actual_classes):
        #Fetching the name of the classes to dictionary and then to the list
        classes={}
        for i in range(len(actual_classes)):
                if actual_classes[i] in classes:
                        classes[actual_classes[i]]= 1
                else:
                        classes[actual_classes[i]]= 1
        c =[]
        for i in classes.keys():
                c.append(i)
        length = len(c)


        #Creating confusion matrix as list -> empty list and hence comparing and increasing 
the count
        confusion_matrix=[]
        for i in range(length):
                for j in range(length):
                        confusion_matrix.append(0)

        count = 0
        for i in range(len(actual_classes)):
                for j in range(length):
                        for k in range(length):
                                if actual_classes[i] == c[j] and predictions[i] == c[k]:
                                        count = count +1
                                        confusion_matrix[j*length+k]                          = 
confusion_matrix[j*length+k]+1

        #Printing confusion matrix
        if filename == 'wisconsin.data':
                for i in range(length):
                        if c[i] == '2':
                                c[i] = 'Benign'
                        if c[i]=='4':
                                c[i] ='Malignant'
```

```
	print "\t\t"+'PREDICTED'
	table = []

	#Append Classes name
	L=[]
	L.append('\t')
	L.append('\t')
	for i in range(length):
		L.append(c[i])
	table.append(L)

	#Create Empty Table
	L=[]
	for i in range(length):
		for j in range(length+2):
			if i==length/2:
				if j==0:
					L.append('ACTUAL')
				elif j==1:
					L.append(c[i])
				else:
					L.append('\t')
			else:
				if j==1:
					L.append(c[i])
				else:
					L.append('\t')
		table.append(L)
		L=[]

	#Populate value to the confusion matrix/empty table
	value_index=0
	for i in range(1, length+1):
		for j in range(2, length+2):
			table[i][j] = confusion_matrix[value_index]
			value_index+=1

	print tabulate(table, tablefmt="grid")

def gaussian_function(mean,stddev,x):
	temp=float((x-mean))/stddev
	temp=temp*temp*0.5
	b=np.exp(-temp)
	a=float(1)/(stddev*sqrt(2*(float(22)/7)))
	return a*b

def maximum_accuracy_find(accuracy_percentage_list):
	maximum = accuracy_percentage_list[0]
	for i in range(1, len(accuracy_percentage_list)):
		if accuracy_percentage_list[i]>maximum:
```

```
                maximum = accuracy_percentage_list[i]
                index = i

    return maximum, index

if __name__ == '__main__':
    print "Enter 1...........categorical data"
    print "Enter 2...........continous data"
    choice = input("Enter Choice\n")
    #choice=2
    if choice == 1:
        filename='bank-full.data'
    elif choice==2:
        filename ='wisconsin.data'

    accuracy_percentage_list=[]

    for x in range(10):
        prob_yes_final_list=[]
        prob_no_final_list=[]
        print "\n"
        print 'ITERATION NO : ' + repr(x+1)
        if choice==1:
            training_sample, test_sample= read_file_load_dataset_categorical(filename)
            dict_training_sample = forming_dictionary(training_sample)
            prob_yes, prob_no = calculating_class_probability(training_sample)
            print "Probability P(yes) = ", float(prob_yes)/float(prob_no+prob_yes)
            print "Probability P(no) = ", float(prob_no)/float(prob_no+prob_yes)

        elif choice==2:
            training_sample, test_sample= read_file_load_dataset_continous(filename)
            dict_training_sample = forming_dictionary_mean_std(training_sample)

        #print dict_training_sample
        actual_classes, prediction_list=[],[]
        for i in range(len(test_sample)):
            actual_classes.append(test_sample[i][-1])

        for i in range(len(test_sample)):
            test_sample[i] = test_sample[i][:-1]
            if choice==1:
                prediction = calculating_class_label(dict_training_sample, test_sample[i], prob_no, prob_yes)
            if choice==2:
                prediction = calculating_class_label_continous(dict_training_sample, test_sample[i])
            prediction_list.append(prediction)
```

```
            accuracy_percentage = calculate_accuracy(prediction_list, actual_classes)
            accuracy_percentage_list.append(accuracy_percentage)
            prob_no_final_list = np.array(prob_no_final_list)
            prob_yes_final_list = np.array(prob_yes_final_list)

            if choice==1:
                print "Average P(xi/yes) * P(yes)= ", np.average(prob_yes_final_list)
                print "Average P(xi/no) * P(no)= ", np.average(prob_no_final_list)
            if choice==2:
                print "Average P(xi/2) * P(2)= ", np.average(prob_no_final_list)
                print "Average P(xi/4) * P(4)= ", np.average(prob_yes_final_list)

            print 'Accuracy: ' + repr(accuracy_percentage)
            print
            confusion_matrix(prediction_list, actual_classes)
            print "*********************************************"
            print


        print
        print
'================================================'
        maximum_accuracy, index = maximum_accuracy_find(accuracy_percentage_list)
        print "Maximum Accuracy ", maximum_accuracy
        print "Iteration No of Maximum Accuracy ", index+1
        accuracy_percentage_list = np.array(accuracy_percentage_list)
        print "Mean", np.mean(accuracy_percentage_list)
        print "Standard Deviation", np.std(accuracy_percentage_list)
        print
'================================================'
```

SAMPLE OUTPUT:

Enter 1...........categorical data
Enter 2...........continous data
Enter Choice

When Choice entered is 1 i.e Categorical Data:

```
ITERATION NO : 1
Probability P(yes) =  0.11372244343
Probability P(no) =  0.88627755657
Average P(xi/yes) * P(yes)=  0.000387131104108
Average P(xi/no) * P(no)=  0.0048544182917
Accuracy: 89.61347965426823

            PREDICTED
+---------+-------+-------+-------+
|         |       | "no"  | "yes" |
+---------+-------+-------+-------+
|         | "no"  | 17898 | 398   |
+---------+-------+-------+-------+
| ACTUAL  | "yes" | 1741  | 557   |
+---------+-------+-------+-------+
********************************************



ITERATION NO : 2
Probability P(yes) =  0.11474215791
Probability P(no) =  0.88525784209
Average P(xi/yes) * P(yes)=  0.000386743015886
Average P(xi/no) * P(no)=  0.00484399520283
Accuracy: 89.47751772360881

            PREDICTED
+---------+-------+-------+-------+
|         |       | "no"  | "yes" |
+---------+-------+-------+-------+
|         | "no"  | 17853 | 464   |
+---------+-------+-------+-------+
| ACTUAL  | "yes" | 1703  | 574   |
+---------+-------+-------+-------+
********************************************



ITERATION NO : 3
Probability P(yes) =  0.112799844615
Probability P(no) =  0.887200155385
Average P(xi/yes) * P(yes)=  0.000378898545776
Average P(xi/no) * P(no)=  0.0048961705273
Accuracy: 89.6231912207439

            PREDICTED
+---------+-------+-------+-------+
|         |       | "no"  | "yes" |
+---------+-------+-------+-------+
|         | "no"  | 17884 | 393   |
+---------+-------+-------+-------+
| ACTUAL  | "yes" | 1744  | 573   |
```

```
+--------+-------+-------+-------+
******************************************



ITERATION NO : 4
Probability P(yes) =  0.113042633777
Probability P(no) =  0.886957366223
Average P(xi/yes) * P(yes)=  0.000379364539028
Average P(xi/no) * P(no)=  0.00487207498547
Accuracy: 89.42410410799262

            PREDICTED
+--------+-------+-------+-------+
|        |       | "no"  | "yes" |
+--------+-------+-------+-------+
|        | "no"  | 17864 | 418   |
+--------+-------+-------+-------+
| ACTUAL | "yes" | 1760  | 552   |
+--------+-------+-------+-------+
******************************************



ITERATION NO : 5
Probability P(yes) =  0.112459939788
Probability P(no) =  0.887540060212
Average P(xi/yes) * P(yes)=  0.000376347300161
Average P(xi/no) * P(no)=  0.00488167609172
Accuracy: 89.29299796057104

            PREDICTED
+--------+-------+-------+-------+
|        |       | "no"  | "yes" |
+--------+-------+-------+-------+
|        | "no"  | 17823 | 447   |
+--------+-------+-------+-------+
| ACTUAL | "yes" | 1758  | 566   |
+--------+-------+-------+-------+
******************************************



ITERATION NO : 6
Probability P(yes) =  0.11391667476
Probability P(no) =  0.88608332524
Average P(xi/yes) * P(yes)=  0.000374692890709
Average P(xi/no) * P(no)=  0.0048946423169
Accuracy: 89.31727687676022

            PREDICTED
+--------+-------+-------+-------+
|        |       | "no"  | "yes" |
+--------+-------+-------+-------+
|        | "no"  | 17832 | 468   |
+--------+-------+-------+-------+
| ACTUAL | "yes" | 1732  | 562   |
+--------+-------+-------+-------+
******************************************



ITERATION NO : 7
Probability P(yes) =  0.110614742158
```

```
Probability P(no) =  0.889385257842
Average P(xi/yes) * P(yes)=  0.000364774549623
Average P(xi/no) * P(no)=  0.00488255377314
Accuracy: 89.43381567446829

            PREDICTED
+--------+-------+-------+-------+
|        |       | "no"  | "yes" |
+--------+-------+-------+-------+
|        | "no"  | 17850 | 382   |
+--------+-------+-------+-------+
| ACTUAL | "yes" | 1794  | 568   |
+--------+-------+-------+-------+
********************************************



ITERATION NO : 8
Probability P(yes) =  0.111197436146
Probability P(no) =  0.888802563854
Average P(xi/yes) * P(yes)=  0.00037242247346
Average P(xi/no) * P(no)=  0.00486226039138
Accuracy: 89.24929591143052

            PREDICTED
+--------+-------+-------+-------+
|        |       | "no"  | "yes" |
+--------+-------+-------+-------+
|        | "no"  | 17820 | 424   |
+--------+-------+-------+-------+
| ACTUAL | "yes" | 1790  | 560   |
+--------+-------+-------+-------+
********************************************



ITERATION NO : 9
Probability P(yes) =  0.115082062737
Probability P(no) =  0.884917937263
Average P(xi/yes) * P(yes)=  0.000382767910241
Average P(xi/no) * P(no)=  0.00488315995467
Accuracy: 89.59405652131689

            PREDICTED
+--------+-------+-------+-------+
|        |       | "no"  | "yes" |
+--------+-------+-------+-------+
|        | "no"  | 17876 | 448   |
+--------+-------+-------+-------+
| ACTUAL | "yes" | 1695  | 575   |
+--------+-------+-------+-------+
********************************************



ITERATION NO : 10
Probability P(yes) =  0.112120034962
Probability P(no) =  0.887879965038
Average P(xi/yes) * P(yes)=  0.000387517567152
Average P(xi/no) * P(no)=  0.00484910252862
Accuracy: 89.47751772360881

            PREDICTED
+--------+-------+-------+-------+
```

```
|        |       | "no"  | "yes" |
+--------+-------+-------+-------+
|        | "no"  | 17878 | 385   |
+--------+-------+-------+-------+
| ACTUAL | "yes" | 1782  | 549   |
+--------+-------+-------+-------+
*******************************************


==================================================
Maximum Accuracy  89.6231912207
Iteration No of Maximum Accuracy  3
Mean 89.4503253375
Standard Deviation 0.127462228926
==================================================
```

When Choice entered is 2 i.e Continous Data:

```
ITERATION NO : 1
Average P(xi/2) * P(2)=  2.40336216836e-05
Average P(xi/4) * P(4)=  5.43472111691e-10
Accuracy: 96.85714285714285

             PREDICTED
+--------+-----+-----+-----+
|        |     |   2 |   4 |
+--------+-----+-----+-----+
|        | 2.0 | 221 |  10 |
+--------+-----+-----+-----+
| ACTUAL | 4.0 |   1 | 118 |
+--------+-----+-----+-----+
*********************************************



ITERATION NO : 2
Average P(xi/2) * P(2)=  3.5550862007e-05
Average P(xi/4) * P(4)=  4.42847885398e-10
Accuracy: 95.71428571428572

             PREDICTED
+--------+-----+-----+-----+
|        |     |   2 |   4 |
+--------+-----+-----+-----+
|        | 2.0 | 223 |  10 |
+--------+-----+-----+-----+
| ACTUAL | 4.0 |   5 | 112 |
+--------+-----+-----+-----+
*********************************************



ITERATION NO : 3
Average P(xi/2) * P(2)=  5.76021004301e-05
Average P(xi/4) * P(4)=  4.23395091724e-10
Accuracy: 95.42857142857143

             PREDICTED
+--------+-----+-----+-----+
|        |     |   2 |   4 |
+--------+-----+-----+-----+
|        | 2.0 | 223 |  11 |
+--------+-----+-----+-----+
| ACTUAL | 4.0 |   5 | 111 |
+--------+-----+-----+-----+
*********************************************



ITERATION NO : 4
Average P(xi/2) * P(2)=  7.42993519377e-05
Average P(xi/4) * P(4)=  5.03599546993e-10
Accuracy: 94.28571428571428

             PREDICTED
+--------+-----+-----+-----+
|        |     |   2 |   4 |
+--------+-----+-----+-----+
```

```
|        | 2.0 | 220 |  17 |
+--------+-----+-----+-----+
| ACTUAL | 4.0 |   3 | 110 |
+--------+-----+-----+-----+
********************************************



ITERATION NO : 5
Average P(xi/2) * P(2)=  2.08865718288e-05
Average P(xi/4) * P(4)=  5.45373376959e-10
Accuracy: 96.57142857142857

            PREDICTED
+--------+-----+-----+-----+
|        |     |   2 |   4 |
+--------+-----+-----+-----+
|        | 2.0 | 216 |   9 |
+--------+-----+-----+-----+
| ACTUAL | 4.0 |   3 | 122 |
+--------+-----+-----+-----+
********************************************



ITERATION NO : 6
Average P(xi/2) * P(2)=  1.77027019602e-05
Average P(xi/4) * P(4)=  4.27329577479e-10
Accuracy: 97.14285714285714

            PREDICTED
+--------+-----+-----+-----+
|        |     |   2 |   4 |
+--------+-----+-----+-----+
|        | 2.0 | 224 |   6 |
+--------+-----+-----+-----+
| ACTUAL | 4.0 |   4 | 116 |
+--------+-----+-----+-----+
********************************************

ITERATION NO : 7
Average P(xi/2) * P(2)=  3.13502666554e-05
Average P(xi/4) * P(4)=  5.25053639247e-10
Accuracy: 96.28571428571429

            PREDICTED
+--------+-----+-----+-----+
|        |     |   2 |   4 |
+--------+-----+-----+-----+
|        | 2.0 | 219 |   8 |
+--------+-----+-----+-----+
| ACTUAL | 4.0 |   5 | 118 |
+--------+-----+-----+-----+
********************************************



ITERATION NO : 8
Average P(xi/2) * P(2)=  6.1162889043e-05
Average P(xi/4) * P(4)=  4.71416397451e-10
Accuracy: 95.71428571428572

            PREDICTED
+--------+-----+-----+-----+
```

```
|        |     |   2 |   4 |
+--------+-----+-----+-----+
|        | 2.0 | 211 |  13 |
+--------+-----+-----+-----+
| ACTUAL | 4.0 |   2 | 124 |
+--------+-----+-----+-----+
********************************************


ITERATION NO : 9
Average P(xi/2) * P(2)=  4.51571575833e-05
Average P(xi/4) * P(4)=  4.79099104475e-10
Accuracy: 95.71428571428572

            PREDICTED
+--------+-----+-----+-----+
|        |     |   2 |   4 |
+--------+-----+-----+-----+
|        | 2.0 | 214 |  15 |
+--------+-----+-----+-----+
| ACTUAL | 4.0 |   0 | 121 |
+--------+-----+-----+-----+
********************************************


ITERATION NO : 10
Average P(xi/2) * P(2)=  3.65806535524e-05
Average P(xi/4) * P(4)=  4.35591215393e-10
Accuracy: 96.0

            PREDICTED
+--------+-----+-----+-----+
|        |     |   2 |   4 |
+--------+-----+-----+-----+
|        | 2.0 | 219 |  10 |
+--------+-----+-----+-----+
| ACTUAL | 4.0 |   4 | 117 |
+--------+-----+-----+-----+
********************************************
```

==

Maximum Accuracy 97.1428571429

Iteration No of Maximum Accuracy 6

Mean 95.9714285714

Standard Deviation 0.771428571429

==

QUESTIONS TO BE ANSWERED:

Q-1 Take three example records that are misclassified in each dataset and explain why these were misclassified.

1. BANK-DATA SET (Categorical Data)

***************TEST SAMPLE NO 1 *********************

["'self-employed'", "'married'", "'university.degree'", "'no'", "'no'", "'no'", "'cellular'", "'apr'", "'tue'", "'nonexistent'", **"'yes'"**]

1. P(x/yes) = 0.0307492420961 |&| P(x/no) = 0.036587366694
YES = 0.0307492420961 NO = 0.036587366694 - **CUMULATIVE ENTRIES**

2. P(x/yes) = 0.55911650065 |&| P(x/no) = 0.614820891441
YES = 0.0171924086384 NO = 0.0224946774063

3. P(x/yes) = 0.354265915981 |&| P(x/no) = 0.288269073011
YES = 0.00609068439421 NO = 0.00648451980359

4. P(x/yes) = 0.902988306626 |&| P(x/no) = 0.778288214383
YES = 0.00549981678732 NO = 0.00504682533907

5. P(x/yes) = 0.439151147683 |&| P(x/no) = 0.456220946131
YES = 0.0024152508542 NO = 0.00230246743115

6. P(x/yes) = 0.83239497618 |&| P(x/no) = 0.823844681433
YES = 0.00201044267725 NO = 0.00189687554732

7. P(x/yes) = 0.828497184929 |&| P(x/no) = 0.607054963084
YES = 0.00166564609856 NO = 0.00115150771536

8. P(x/yes) = 0.11130359463 |&| P(x/no) = 0.0577522559475
YES = 0.000185392398151 NO = 6.65021683027e-05

9. P(x/yes) = 0.195755738415 |&| P(x/no) = 0.195187312004
YES = 3.62916257965e-05 NO = 1.29803794735e-05

10. P(x/yes) = 0.675617150282 |&| P(x/no) = 0.886409625376
YES = 2.45192447997e-05 NO = 1.15059333063e-05

PREDICTION = "no"

Reason of Misclassification :

The prob of yes of second attribute is much less than no, here the weight of no become large.
The probability of 10^{th} attribute is also less for yes, where weight of no become much more than yes in the total product.
Hence, the prediction is “no” instead of “yes”

*************TEST SAMPLE NO 2 ***********************

["'admin.'", "'single'", "'university.degree'", "'no'", "'no'", "'no'", "'cellular'", "'mar'", "'thu'", "'nonexistent'", **"'no'"**]

1. P(x/yes) = 0.297098310957 |&| P(x/no) = 0.247251845775
YES = 0.297098310957 NO = 0.247251845775

2. P(x/yes) = 0.342572542226 |&| P(x/no) = 0.270494941209
YES = 0.101777723676 NO = 0.0668803734867

3. P(x/yes) = 0.354265915981 |&| P(x/no) = 0.288269073011
YES = 0.0360563785044 NO = 0.0192795432676

4. P(x/yes) = 0.902988306626 |&| P(x/no) = 0.778288214383
YES = 0.0325584881688 NO = 0.0150050413039

5. P(x/yes) = 0.439151147683 |&| P(x/no) = 0.456220946131
YES = 0.0142980974461 NO = 0.00684561414039

6. P(x/yes) = 0.83239497618 |&| P(x/no) = 0.823844681433
YES = 0.0119016644831 NO = 0.0056397228007

7. P(x/yes) = 0.828497184929 |&| P(x/no) = 0.607054963084
YES = 0.00986049552021 NO = 0.00342362171659

8. P(x/yes) = 0.056734517107 |&| P(x/no) = 0.00771123872026
YES = 0.000559430451775 NO = 2.64003643445e-05

9. P(x/yes) = 0.229969683846 |&| P(x/no) = 0.208422203992
YES = 0.000128652044128 NO = 5.50242212287e-06

10. P(x/yes) = 0.675617150282 |&| P(x/no) = 0.886409625376
YES = 8.69195274319e-05 NO = 4.8773999326e-06

PREDICTION = "yes"

Reason of Misclassification :

In this sample, probability of attribute given yes is dominating for each attribute. Hence yes dominates over no.
Hence, prediction is missclassified.

***************TEST SAMPLE NO 3 **********************

["'retired'", "'married'", "'professional.course'", "'no'", "'yes'", "'yes'", "'cellular'", "'jul'", "'thu'", "'success'", "'**no**'"]

1. P(x/yes) = 0.0887830229537 |&| P(x/no) = 0.0351107465135
YES = 0.0887830229537 NO = 0.0351107465135

2. P(x/yes) = 0.55911650065 |&| P(x/no) = 0.614820891441
YES = 0.0496400531109 NO = 0.0215868204706

3. P(x/yes) = 0.136422693807 |&| P(x/no) = 0.124200164069
YES = 0.00677202976611 NO = 0.00268108664418

4. P(x/yes) = 0.902988306626 |&| P(x/no) = 0.778288214383
YES = 0.00611506369092 NO = 0.0020866581369

5. P(x/yes) = 0.53702901689 |&| P(x/no) = 0.520153130982
YES = 0.00328396664216 NO = 0.0010853817632

6. P(x/yes) = 0.143785188393 |&| P(x/no) = 0.15252939568
YES = 0.00047218576232 NO = 0.000165552624422

7. P(x/yes) = 0.828497184929 |&| P(x/no) = 0.607054963084
YES = 0.000391204574845 NO = 0.000100499542307

8. P(x/yes) = 0.132957990472 |&| P(x/no) = 0.175444353295
YES = 5.20137741349e-05 NO = 1.76320772065e-05

9. P(x/yes) = 0.229969683846 |&| P(x/no) = 0.208422203992
YES = 1.19615911934e-05 NO = 3.67491639235e-06

10. P(x/yes) = 0.197055002165 |&| P(x/no) = 0.0140005468964
YES = 2.35709137852e-06 NO = 5.14508392913e-08

PREDICTION = "yes"

Reason of Misclassification :

In this sample, probability of attribute given yes is dominating for each attribute. Hence yes dominates over no.
Hence, prediction is missclassified.

2. BREAST CANCER -DATA SET (Categorical Data)

It is observed from the above confusion matrix as well, that “2” is the class that is missclassified majority of times. “4” is also missclassified but less number of times

*************TEST SAMPLE NO 1 ******************

[10.0, 2.0, 2.0, 1.0, 2.0, 6.0, 1.0, 1.0, 2.0, 4.0]
1 Mean(2) = 2.93303571429 Stddev(2) = 1.59238637141
Mean(4) = 7.296 Stddev(4) = 2.31179237822
P(x/2) = 1.32413494064e-05 |&| P(x/4) = 0.0870554904198
Cumltive. 2 = 1.32413494064e-05 Cumltive 4 = 0.0870554904198

2 Mean(2) = 1.29017857143 Stddev(2) = 0.80216891495
Mean(4) = 6.512 Stddev(4) = 2.6999733332
P(x/2) = 0.336146572349 |&| P(x/4) = 0.036563112809
Cumltive. 2 = 4.45103421624e-06 Cumltive 4 = 0.00318301971686

3 Mean(2) = 1.39285714286 Stddev(2) = 0.869699690135
Mean(4) = 6.464 Stddev(4) = 2.59705679568
P(x/2) = 0.359439769911 |&| P(x/4) = 0.035057223762
Cumltive. 2 = 1.59987871455e-06 Cumltive 4 = 0.000111587834453

4 Mean(2) = 1.29017857143 Stddev(2) = 0.732346988093
Mean(4) = 5.688 Stddev(4) = 3.30675913849
P(x/2) = 0.503516704359 |&| P(x/4) = 0.0441550387647
Cumltive. 2 = 8.05565657725e-07 Cumltive 4 = 4.92716515594e-06

5 Mean(2) = 2.10714285714 Stddev(2) = 0.879906071054
Mean(4) = 5.208 Stddev(4) = 2.48610860583
P(x/2) = 0.449952554218 |&| P(x/4) = 0.0697815375043
Cumltive. 2 = 3.62466325284e-07 Cumltive 4 = 3.43825160119e-07

6 Mean(2) = 1.47767857143 Stddev(2) = 1.43907273105
Mean(4) = 7.824 Stddev(4) = 3.10371132678
P(x/2) = 0.00198751716053 |&| P(x/4) = 0.108129650924
Cumltive. 2 = 7.20408041618e-10 Cumltive 4 = 3.71776945425e-08

7 Mean(2) = 2.13392857143 Stddev(2) = 1.08968815751
Mean(4) = 5.848 Stddev(4) = 2.06322466057
P(x/2) = 0.21300181331 |&| P(x/4) = 0.0122283087524
Cumltive. 2 = 1.53448219187e-10 Cumltive 4 = 4.54620327567e-10

8 Mean(2) = 1.26339285714 Stddev(2) = 0.989813793429
Mean(4) = 5.32 Stddev(4) = 3.33610551392
P(x/2) = 0.388949088312 |&| P(x/4) = 0.0516969374448
Cumltive. 2 = 5.96835449561e-11 Cumltive 4 = 2.35024786354e-11

9 Mean(2) = 1.11607142857 Stddev(2) = 0.703886350839

Mean(4) = 2.608 Stddev(4) = 2.60121817616
P(x/2) = 0.257561449547 |&| P(x/4) = 0.149204711128
Cumltive. 2 = 1.5372180353e-11 Cumltive 4 = 3.50668053557e-12

PREDICTION = 2.0

Reason of Misclassification :

In our training samples, the mean and standard deviation of the value is more close to the class 2.0. Since (x-u)/sigma will make the value of the attribute more close to the mean, 2.0 is predicted.
Second reason can be, the values in the sample are in defined range, there is very less difference between them, thus it is able to distinguish that properly with the two classes and the classes are misclassified.

**************TEST SAMPLE NO 2 ************************
[4.0, 3.0, 1.0, 1.0, 2.0, 1.0, 4.0, 8.0, 1.0, **2.0**]
1 Mean(2) = 2.93303571429 Stddev(2) = 1.59238637141
Mean(4) = 7.296 Stddev(4) = 2.31179237822
P(x/2) = 0.200117412707 |&| P(x/4) = 0.062441760204
Cumltive. 2 = 0.200117412707 Cumltive 4 = 0.062441760204

2 Mean(2) = 1.29017857143 Stddev(2) = 0.80216891495
Mean(4) = 6.512 Stddev(4) = 2.6999733332
P(x/2) = 0.0512855879175 |&| P(x/4) = 0.0633955514232
Cumltive. 2 = 0.0102631391632 Cumltive 4 = 0.00395852981997

3 Mean(2) = 1.39285714286 Stddev(2) = 0.869699690135
Mean(4) = 6.464 Stddev(4) = 2.59705679568
P(x/2) = 0.414137994551 |&| P(x/4) = 0.016793582768
Cumltive. 2 = 0.00425035587086 Cumltive 4 = 6.64778981713e-05

4 Mean(2) = 1.29017857143 Stddev(2) = 0.732346988093
Mean(4) = 5.688 Stddev(4) = 3.30675913849
P(x/2) = 0.503516704359 |&| P(x/4) = 0.0441550387647
Cumltive. 2 = 0.00214012518045 Cumltive 4 = 2.93533417075e-06

5 Mean(2) = 2.10714285714 Stddev(2) = 0.879906071054
Mean(4) = 5.208 Stddev(4) = 2.48610860583
P(x/2) = 0.449952554218 |&| P(x/4) = 0.0697815375043
Cumltive. 2 = 0.000962954791289 Cumltive 4 = 2.04832131524e-07

6 Mean(2) = 1.47767857143 Stddev(2) = 1.43907273105
Mean(4) = 7.824 Stddev(4) = 3.10371132678
P(x/2) = 0.26230977515 |&| P(x/4) = 0.0114612095729
Cumltive. 2 = 0.000252592454782 Cumltive 4 = 2.34762398667e-09

7 Mean(2) = 2.13392857143 Stddev(2) = 1.08968815751

Mean(4) = 5.848 Stddev(4) = 2.06322466057
P(x/2) = 0.0844725441305 |&| P(x/4) = 0.129440242884
Cumltive. 2 = 2.13371272836e-05 Cumltive 4 = 3.03877019035e-10

8 Mean(2) = 1.26339285714 Stddev(2) = 0.989813793429
Mean(4) = 5.32 Stddev(4) = 3.33610551392
P(x/2) = 3.52248723935e-11 |&| P(x/4) = 0.08658619124
Cumltive. 2 = 7.51597585809e-16 Cumltive 4 = 2.63115536836e-11

9 Mean(2) = 1.11607142857 Stddev(2) = 0.703886350839
Mean(4) = 2.608 Stddev(4) = 2.60121817616
P(x/2) = 0.559004636557 |&| P(x/4) = 0.126667737602
Cumltive. 2 = 4.20146535292e-16 Cumltive 4 = 3.33282497789e-12

PREDICTION = 4.0

Reason of Misclassification :

In our training samples, the mean and standard deviation of the value is more close to the class 4.0. Since (x-u)/sigma will make the value of the attribute more close to the mean, 4.0 is predicted.
Second reason can be, the values in the sample are in defined range, there is very less difference between them, thus it is able to distinguish that properly with the two classes and the classes are misclassified.

***************TEST SAMPLE NO 3 *************************

[6.0, 3.0, 2.0, 1.0, 3.0, 4.0, 4.0, 1.0, 1.0, **4.0**]
1 Mean(2) = 2.93303571429 Stddev(2) = 1.59238637141
Mean(4) = 7.296 Stddev(4) = 2.31179237822
P(x/2) = 0.0391973739755 |&| P(x/4) = 0.147444816796
Cumltive. 2 = 0.0391973739755 Cumltive 4 = 0.147444816796

2 Mean(2) = 1.29017857143 Stddev(2) = 0.80216891495
Mean(4) = 6.512 Stddev(4) = 2.6999733332
P(x/2) = 0.0512855879175 |&| P(x/4) = 0.0633955514232
Cumltive. 2 = 0.00201026036916 Cumltive 4 = 0.0093473454653

3 Mean(2) = 1.39285714286 Stddev(2) = 0.869699690135
Mean(4) = 6.464 Stddev(4) = 2.59705679568
P(x/2) = 0.359439769911 |&| P(x/4) = 0.035057223762
Cumltive. 2 = 0.000722567524552 Cumltive 4 = 0.000327691981557

4 Mean(2) = 1.29017857143 Stddev(2) = 0.732346988093
Mean(4) = 5.688 Stddev(4) = 3.30675913849
P(x/2) = 0.503516704359 |&| P(x/4) = 0.0441550387647
Cumltive. 2 = 0.000363824818639 Cumltive 4 = 1.44692521486e-05

5 Mean(2) = 2.10714285714 Stddev(2) = 0.879906071054
Mean(4) = 5.208 Stddev(4) = 2.48610860583
P(x/2) = 0.270894279731 |&| P(x/4) = 0.108148468578

Cumltive. 2 = 9.85580621936e-05 Cumltive 4 = 1.56482746133e-06

6 Mean(2) = 1.47767857143 Stddev(2) = 1.43907273105
Mean(4) = 7.824 Stddev(4) = 3.10371132678
P(x/2) = 0.0596543150752 |&| P(x/4) = 0.060160414751
Cumltive. 2 = 5.8794136953e-06 Cumltive 4 = 9.41406690875e-08

7 Mean(2) = 2.13392857143 Stddev(2) = 1.08968815751
Mean(4) = 5.848 Stddev(4) = 2.06322466057
P(x/2) = 0.0844725441305 |&| P(x/4) = 0.129440242884
Cumltive. 2 = 4.96649032837e-07 Cumltive 4 = 1.2185591072e-08

8 Mean(2) = 1.26339285714 Stddev(2) = 0.989813793429
Mean(4) = 5.32 Stddev(4) = 3.33610551392
P(x/2) = 0.388949088312 |&| P(x/4) = 0.0516969374448
Cumltive. 2 = 1.93171188533e-07 Cumltive 4 = 6.29957739376e-10

9 Mean(2) = 1.11607142857 Stddev(2) = 0.703886350839
Mean(4) = 2.608 Stddev(4) = 2.60121817616
P(x/2) = 0.559004636557 |&| P(x/4) = 0.126667737602
Cumltive. 2 = 1.07983590039e-07 Cumltive 4 = 7.97953216315e-11

PREDICTION = 2.0

Reason of Misclassification :

In our training samples, the mean and standard deviation of the value is more close to the class 2.0. Since (x-u)/sigma will make the value of the attribute more close to the mean, 2.0 is predicted.
Second reason can be, the values in the sample are in defined range, there is very less difference between them, thus it is able to distinguish that properly with the two classes and the classes are misclassified.

Q-2 What is the role of the following Laplacian smoothing used in the pseudocode for estimating posterior probabilities?

$$P(w_k|v_j) = \frac{n_k+1}{n+|vocabulary|}$$

Ans – It is a technique used to smooth categorical data. This estimation of probabilities could be problematic when we get probability 0 for documents with unknown words. A common way of solving this problem is to use Laplace smoothing.

Let *c* refer to a class (such as Positive or Negative), and let *w* refer to a tolken or word.
The maximum likelihood estimator for *P*(*w*|*c*) is

***count*(*w*,*c*)/*count*(*c*)=counts w in class c/counts of words in class c.**

Here the P(w/c) can be 0 as well. Hence the value for whole attribute will become zero in this case.

So we replace it by,

P(w/c) = count(w,c)+1 / count(c) + |V| + 1
V refers to the vocabulary (the words in the training set)
c refer to a class
w refer to a tolken or word.

Q-3 Briefly explain what modifications you would suggest in order to build an NB classifier dealing with mixed data (consisting of both continuous and discrete features) in the first dataset (Adult Dataset).

Ans.
For Naive Baise Classification we have considered the assumption that the likelihood is calculated by product of probabilities of all the features as they are considered to be independent of each other.

So, if we have the dataset that consists of mixed data (i.e both continous data and discrete features).

In this case :

- If the feature is discrete, we can calculate it using the frequencies.
- If the feature is continous, it can be calculated by gauss distribution using parameters(mean, variance) of the particulare feature/attribute over entire dataset.

The final probability will be the product of the estimated probabilities if the feature was continous and if the feature was discrete. Prediciton will be on the basis of final probability.

Q-4 What procedure would you suggest for considering missing values (and not discarding them!)?

For Discrete numeric/continuous data:

a) Mean/Mode substitution:
Replace missing value with sample mean or mode.

b) Dummy variable control:
Replaces missing values with predicted score from a regression equation.

c) Nearest neighbour substitution:
Replace the missing values with whichever instance is most resembling to the current instance can be given the same values.

d) Missing values can be replaced by the average of all the observed values.

For Categorical attributes:

- Missing values can be replaced by the most frequent value among all the observed values.
- Missing values can be taken as a separate feature only. e.g. if it is unknown, it can be taken as feature only.