No. ________ Date: ________

18.5 Exercises

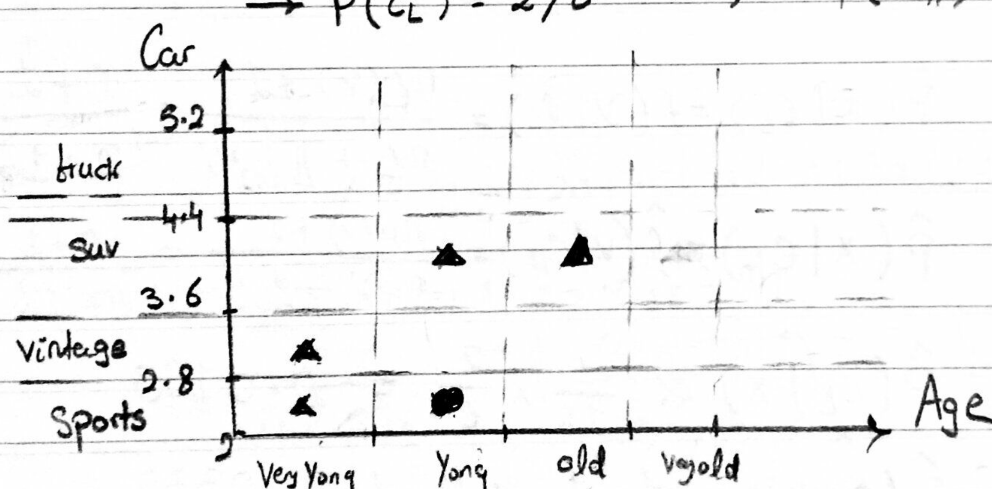
dom (Car) = { sports, vintage, suv, truck }

Q1 : • Data:

X_i	Age	Car	Class
X_1	25	sports	L
X_2	20	vintage	H
X_3	25	sport	L
X_4	45	suv	H
X_5	20	sports	H
X_6	25	suv	H

• Classify X (Age = 23 , Car = truck)

Full Bayes method :
→ $D_L = \{ X_1, X_3 \}$, $n_L = 2$
→ $D_H = \{ X_2, X_4, X_5, X_6 \}$, $n_H = 4$
→ $\hat{P}(C_L) = 2/6$, $\hat{P}(C_H) = 4/6$



Bins	Domain
[18 < x < 20]	Very Young
[20 < x < 25]	Young
[25 < x < 45]	Old
[> 45]	Very old

Bins	Domain
[2.0, 2.8]	sports
[2.8, 3.6]	vintage
[3.6, 4.4]	suv
[4.4, 5.2]	truck

Class : L		X_2: Sports	Vintage	SUV	truck	$\hat{f}_{X_1}$
X_1	Very Young	0	0	0	0	0
	Young	1	0	0	0	1
	Old	0	0	0	0	0
	Very Old	0	0	0	0	0
	$\hat{f}_{X_2}$	1	0	0	0	

Class : H		X_2: Sports	vintage	suv	truck	$\hat{f}_{X_1}$
X_1	Very Young	1/4	1/4	0	0	1/2
	Young	0	0	1/4	0	1/4
	old	0	0	1/4	0	1/4
	Very old	0	0	0	0	0
	$\hat{f}_{X_2}$	1/4	1/4	1/2	0	

Full Bayes:

$$\hat{P}(x \mid C_L) = \hat{f}(v \mid C_L) = \frac{n_L(v) + 1}{n_L + \prod_{j=1}^{d} m_j} = \frac{0+1}{2+16} = \frac{1}{18}$$

$$\hat{P}(x \mid C_H) = \hat{f}(v \mid C_H) = \frac{n_H(v) + 1}{n_H + \prod_{j=1}^{d} m_j} = \frac{0+1}{4+16} = \frac{1}{20}$$

$$\hat{P}(C_L \mid x) \propto \frac{1}{18} \times \frac{2}{6} = 0.0185$$

$$\hat{P}(C_H \mid x) \propto \frac{1}{20} \times \frac{4}{6} = 0.0333$$

$$\underline{\underline{\hat{y} = C_H}}$$

Naive Bayes:

$$\hat{P}(x \mid C_L) = \prod_{j=1}^{d} P(x_j \mid C_i) = \hat{f}_{X_1 = \text{Young}} \cdot \hat{f}_{X_2 = \text{truck}}$$

$$= 1 \cdot \left(\frac{0+1}{2+4}\right)$$

$$\hat{P}(x \mid C_H) = \prod_{j=1}^{d} P(x_j \mid C_i) = \hat{f}_{X_1 = \text{Young}} \cdot \hat{f}_{X_2 = \text{truck}} = \frac{1}{4} \cdot \frac{0+1}{4+4} = \frac{1}{32}$$

$$\hat{P}(C_L \mid x) \propto \frac{1}{6} \times \frac{2}{6} = 0.055$$

$$\hat{P}(C_H \mid x) \propto \frac{1}{32} \times \frac{4}{6} = 0.020 \qquad \hat{y} = C_L$$

Q2:

x_i	a_1	a_2	a_3	Class
x_1	T	T	5.0	Y
x_2	T	T	7.0	Y
x_3	T	F	8.0	N
x_4	F	F	3.0	Y
x_5	F	T	7.0	N
x_6	F	T	4.0	N
x_7	F	F	5.0	N
x_8	T	F	6.0	Y
x_9	F	T	1.0	N

• Classify $x = (T, F, 1.0)$

• $Dom(a1) = Dom(a2) = \{T, F\}$

$\rightarrow D_Y = [x_1, x_2, x_4, x_8\}$, $\rightarrow D_N = [x_3, x_5, x_6, x_7, x_9\}$

$\rightarrow n_Y = 4$ $\rightarrow n_N = 5$

$\rightarrow \hat{P}(C_Y) = 4/9$ $\rightarrow \hat{P}(C_N) = 5/9$

In Naive Bayes, $\hat{P}(x \mid C_i) = \prod_{j=1}^{d} P(x_j \mid C_i)$

$$= P(a_1 \mid C_i) \cdot P(a_2 \mid C_i) \cdot P(a_3 \mid C_i)$$

$$P(a_3 \mid C_Y) \propto f(a_3 \mid \mu_{Y3}, \sigma^2_{Y3})$$

$$= \frac{1}{\sqrt{2\pi}\,\sigma_{Y3}} e^{-\left(\frac{(a_3 - \mu_{Y3})^2}{2\sigma^2_{Y3}}\right)} = \underline{4.344 \times 10^{-3}}$$

$\mu_{Y3} = 5.25$, $\sigma^2_{Y3} = \frac{1}{n_Y} Z_{Y3}^T Z_{Y3}$

$$= \frac{1}{4}(-0.25, 1.75, -2.25, 0.75)^T (\,)$$

$$= 2.1875$$

$$\underline{\sigma_{Y3} = 1.479}$$

No: Date:

$P(a_1|C_y) = 3/4$, $P(a_2|C_y) = 2/4$

$\Rightarrow \hat{P}(x|C_y) = 4.344 \times 10^{-3} \times 3/4 \times 2/4 = 1.629 \times 10^{-3}$

$\Rightarrow P(C_y|x) \propto \hat{P}(x|C_y) \times \hat{P}(C_y) = \checkmark \times 4/9 = \underline{7.24 \times 10^{-4}}$

For class C_N:

$$P(a_3|C_N) \propto f(a_3 | \mu_{N3}, \sigma^2_{N3})$$

$$= \frac{1}{\sqrt{2\pi}\,\sigma_{N3}} e^{\left(\frac{-(a_3 - \mu_{N3})^2}{2\sigma^2_{N3}}\right)} = 0.0429$$

$\mu_{N3} = 5.0$, $\sigma^2_{N3} = \frac{1}{n_N} Z^T_{N3} Z_{N3}$

$$= \frac{1}{5}(3, 2, -1, 0, -4)^T \begin{pmatrix} 3 \\ 2 \\ -1 \\ 0 \\ -4 \end{pmatrix}$$

$$= \frac{1}{5}(30) = 6$$

$$\sigma_{N3} = 2.45$$

$P(a_1|C_N) = 1/5$, $P(a_2|C_N) = 2/5$

$\hat{P}(x|C_N) = 0.0429 \times 1/5 \times 2/5 = 3.432 \times 10^{-3}$

$P(C_N|x) \propto \hat{P}(x|C_N) \times \hat{P}(C_N) = 3.432 \times 10^{-3} \times 5/9$

$$= \underline{1.91 \times 10^{-3}}$$

$$\boxed{\hat{y} = C_y}$$

No: Date:

Q3: Given $\mu_1 = (1, 3)$ $\mu_2 = (5, 5)$

$\Sigma_1 = \begin{pmatrix} 5 & 3 \\ 3 & 2 \end{pmatrix}$ $\Sigma_2 = \begin{pmatrix} 2 & 0 \\ 0 & 1 \end{pmatrix}$

Req. Classify $x = (3, 4)^T$ using Full Bayes.

$P(c_1) = P(c_2) = 0.5$

$10 - 3$ $\frac{1}{1}\begin{pmatrix} 5 & -3 \\ -3 & 2 \end{pmatrix}$

- $\hat{y} \leftarrow \max_{c_i} \{ f(x \mid \hat{\mu}_i, \hat{\Sigma}_i) \, P(c_i) \}$

- $f_1(x) = \frac{1}{(\sqrt{2\pi})^2 \sqrt{1}} e^{-\left(\frac{(2,1)^T \Sigma_1^{-1} (2,1)}{2} \right)}$

$= \frac{1}{2\pi} e^{-\left(\frac{(7, -4)^T (2, 1)}{2} \right)}$

$= \frac{1}{2\pi} e^{-\left(\frac{\cancel{10}\, 5}{\cancel{2}} \right)}$

$= 1.07 \times 10^{-3}$

$P(c_1 \mid x) \propto 1.07 \times 10^{-3} \times 0.5 = 5.36 \times 10^{-4}$

- $f_2(x) = \frac{1}{2\pi\sqrt{2}} e^{-\left(\frac{(-2,-1)^T \begin{pmatrix} 1/2 & 0 \\ 0 & 1 \end{pmatrix} (-2,-1)}{2} \right)}$

$= \frac{1}{2\pi\sqrt{2}} e^{-\left(\cancel{\frac{(-1, -1)^T (-2, -1)}{2}} \right)^{3/2}}$

$= 0.025$

$P(c_2 \mid x) \propto 0.025 \times 0.5 = 0.01255$

$\boxed{\hat{y} = c_2}$

Sheet 5_Probabilistic_Classifiers

March 27, 2019

1 Sheet 5: Probabilistic Classification

1.1 Question on Naive Bayes Implementation

a. Use the Face-data set we used in Assignment 1.

```
In [215]: #imports cell
          from os import listdir
          from PIL import Image as PImage
          import matplotlib.pyplot as plt
          import numpy as np
          import math
          from sklearn.metrics import accuracy_score
          from sklearn.metrics import confusion_matrix
```

```
In [2]: #Utility Method to loadImages into list from a given path parameter.
        def loadImages(path):
            # return array of images
            foldersList = listdir(path)
            loadedImages = []
            for folder in foldersList :
                imagesList = listdir(path+folder)
                for image in imagesList:
                    img = PImage.open(path +folder+'/'+ image)
                    loadedImages.append(img)
            return loadedImages
```

```
In [5]: #Loading our Faces dataset.
        path = "../Projects/Face-Recoginition/orl_faces/"
        imgs = loadImages(path)
```

```
In [8]: #converting the images list into data matrix.
        dataMatrix = np.arange(4121600).reshape(400,10304)
        j=0
        label = []
        for i in range(0,400) :
            if(i%10 == 0):
                j = j+1
            dataMatrix[i] = np.array(imgs[i]).flatten()      #flatten() method is used to unrol
            label.append(j)
```

```
In [11]: # (50 - 50) split is used here.
         #Even instances for test set.
         #Odd instances for train set.
         trainSet=np.arange(200*10304).reshape(200,10304)
         testSet=np.arange(200*10304).reshape(200,10304)
         trainLabel=[]
         testLabel=[]
         j,k=0,0
         for i in range(0,400):
             if(i%2==0):
                 testSet[j]=dataMatrix[i]
                 testLabel.append(label[i])
                 j+=1
             else:
                 trainSet[k]=dataMatrix[i]
                 trainLabel.append(label[i])
                 k+=1
```

b. Implement your Naive Bayes Classifier. You have now a long feature vector.

```
In [263]: def NaiveBayes(data_set,label_vec):
              n = data_set.shape[0]
              d = data_set.shape[1]
              data_classes = {}
              #separate data into classes. Each class is stored into a matrix.
              for i in range(n):
                  if(label_vec[i] not in data_classes):
                      data_classes[label_vec[i]] = np.array([data_set[i]])
                  else:
                      data_classes[label_vec[i]] = np.concatenate([data_classes[label_vec[i]],

              #compute for each class: cardinality or size,
              #   it's prior probability,
              #   it's mean, it's centered data matrix,
              #
              n_classes = len(data_classes)
              size_classes = np.zeros(n_classes)
              prior_prob = np.zeros(n_classes)
              mean_classes = np.zeros((n_classes,d))
              var_classes = np.zeros((n_classes,d))
              for i in range(1,n_classes+1):
                  size_classes[i-1] = data_classes[i].shape[0]
                  prior_prob[i-1] = size_classes[i-1]/n
                  mean_classes[i-1] = np.mean(data_classes[i], axis=0)
                  var_classes[i-1] = np.var(data_classes[i], axis=0)
                 # for j in range(d):
                  #    var_classes[i-1,j] = (1/(size_classes[i-1])) * (np.dot(data_classes[i][
```

```
        return prior_prob,mean_classes,var_classes

    def calculateProbability(x, mean, var):
        if(var == 0): return 1.0
        exponent = math.exp(-(math.pow(x-mean,2)/(2.0*var)))
        return (1 / (math.sqrt(2.0*math.pi) * math.sqrt(var))) * exponent

    def Predict(test_point,prior_prob,mean_classes,var_classes):
        n_classes = mean_classes.shape[0]
        d = mean_classes.shape[1]
        posterior_prob = np.zeros(n_classes)
        epsilon = 0.0001
        likelihood = 0
        for i in range(n_classes):
            likelihood = 0
            for j in range(d):
                probabiltiy = calculateProbability(test_point[j],mean_classes[i,j],var_cl
                likelihood += np.log(probabiltiy)
            posterior_prob[i] = likelihood + np.log(prior_prob[i])

        return np.argmax(posterior_prob)
```

```
In [145]: prior_prob,mean_classes,var_classes = NaiveBayes(trainSet,trainLabel)
```

```
In [213]: label_predict = np.zeros(testSet.shape[0])
          for i in range(testSet.shape[0]):
              label_predict[i] = Predict(testSet[i],prior_prob,mean_classes,var_classes)+1
```

c. Report Classification Accuracy.

```
In [218]: accuracy_score(testLabel,label_predict)

Out[218]: 0.95
```

d. Show the confusion matrix and the error cases. Discuss.

```
In [226]: np.set_printoptions(threshold=np.inf)
          print(confusion_matrix(testLabel,label_predict))
          np.set_printoptions()
```

```
[[4 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0
  0 0 0 0]
 [0 5 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  0 0 0 0]
 [0 0 5 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  0 0 0 0]
 [0 0 0 5 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  0 0 0 0]
 [0 0 0 0 5 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

```
 0 0 0 0]
[0 0 0 0 0 5 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0]
[0 0 0 0 0 0 4 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0]
[0 0 0 0 0 0 0 5 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0]
[0 0 0 0 0 0 0 0 5 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0]
[0 0 0 0 0 0 0 0 0 5 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0]
[0 0 0 0 0 0 0 0 0 0 5 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0]
[0 0 0 0 0 0 0 0 0 0 0 5 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0]
[0 0 0 0 0 0 0 0 0 0 0 0 5 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0]
[0 0 0 0 0 0 0 0 0 0 0 0 0 5 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 5 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 5 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 5 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
 0 0 0 0]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 5 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
 0 0 0 0]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 5 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 5 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4 0 0 0 0 0 1 0 0 0 0 0 0 0
 0 0 0 0]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 4 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 5 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 5 0 0 0 0 0 0 0 0 0 0
 0 0 0 0]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 5 0 0 0 0 0 0 0 0 0
 0 0 0 0]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 5 0 0 0 0 0 0 0 0
 0 0 0 0]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 5 0 0 0 0 0 0 0
```

```
 0 0 0 0]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 5 0 0 0 0 0 0
 0 0 0 0]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 5 0 0 0 0 0
 0 0 0 0]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 5 0 0 0 0
 0 0 0 0]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 5 0 0 0
 0 0 0 0]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 5 0 0
 0 0 0 0]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 5 0
 0 0 0 0]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 5
 0 0 0 0]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 5 0 0 0]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0
 0 4 0 0]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 5 0]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 3 0
 0 0 0 2]]
```

e. Appy Naive Bayes on the first 40 Components using PCA.

```
In [223]: def PCA(data_set,r):
              #calculate mean face of the training set.
              mean = np.mean(data_set,axis=0)
              #subtract the mean from the training set.
              center_data = data_set-mean
              #compute the covariance matrix from obtained centered data matrix.
              cov_matrix=np.cov(center_data, rowvar=False, bias=True)
              #compute eigen vectors and values from obtained covariance matrix.
              eigVal,eigVectMatrix=np.linalg.eigh(cov_matrix)
              #flip both eigen values and vectors in order to be sorted descendingly.
              eigVal=np.flip(eigVal,axis=0)
              eigVectMatrix=np.flip(eigVectMatrix,axis=1)
              return eigVal[0:r],eigVectMatrix[:,0:r]
```

```
In [224]: eigvalues, eigvectors = PCA(trainSet,40)
```

```
In [257]: #Projecting the data instances on the new basis.
          reduced_trainset = (eigvectors.T @ trainSet.T)
          reduced_testset = (eigvectors.T @ testSet.T)
```

```
In [258]: prior_prob,mean_classes,var_classes = NaiveBayes(reduced_trainset.T,trainLabel)
```

```
In [264]: label_predict = np.zeros(testSet.shape[0])
          for i in range(testSet.shape[0]):
              label_predict[i] = Predict(reduced_testset.T[i],prior_prob,mean_classes,var_class
```

f. Compare results with what you obtained in c,d.

```
In [265]: accuracy_score(testLabel,label_predict)
```

```
Out[265]: 0.865
```

```
In [266]: np.set_printoptions(threshold=np.inf)
          print(confusion_matrix(testLabel,label_predict))
          np.set_printoptions()
```

```
[[3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0
  0 0 0 1]
 [0 4 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0
  0 0 0 0]
 [0 0 5 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  0 0 0 0]
 [0 0 0 5 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  0 0 0 0]
 [0 0 0 0 5 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  0 0 0 0]
 [0 0 0 0 0 5 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  0 0 0 0]
 [0 0 0 0 0 0 4 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0
  0 0 0 0]
 [0 0 0 0 0 0 0 5 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  0 0 0 0]
 [0 0 0 0 0 0 0 0 5 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  0 0 0 0]
 [0 0 0 0 0 0 0 0 0 5 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 5 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 5 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 4 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0
  0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 5 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 3 0
  0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 5 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  0 0 0 0]
 [0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 4 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 5 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

```
 0 0 0 0]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 5 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0]
[0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 2 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0
 0 0 0 0]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 5 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 5 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 2 0
 0 0 0 0]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 3 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 5 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 5 0 0 0 0 0 0 0 0 0 0
 0 0 0 0]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 5 0 0 0 0 0 0 0 0 0
 0 0 0 0]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 5 0 0 0 0 0 0 0 0
 0 0 0 0]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 5 0 0 0 0 0 0 0
 0 0 0 0]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4 0 0 0 0 1 0
 0 0 0 0]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 5 0 0 0 0 0
 0 0 0 0]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 5 0 0 0 0
 0 0 0 0]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 5 0 0 0
 0 0 0 0]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 4 0 0
 0 0 0 0]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 5 0
 0 0 0 0]
[0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4
 0 0 0 0]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 5 0 0 0]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 0
 0 3 0 0]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 5 0]
[0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 2 0
 0 0 0 1]]
```

1.2 Question 3: Midterm 2 Question (Laptops were allowed)

```
In [268]: data_matrix = np.array([[2,6],[3,3],[3,5],[4,3],[4,4],[4,5],
                                  [5,3],[5,5],[6,2],[6,4],[6,6],[7,2],[7,3],[7,4],[7,5],
                                  [8,4],[9,2],[9,3],[10,1],[10,3],[10,5],
                                  [11,3],[11,4],[12,2],[13,5]])
          data_label = np.array([1,2,1,2,1,2,1,2,1,1,1,3,2,2,2,1,1,1,3,3,3,3,3,3,3])
```

```
In [271]: prior_prob,mean_classes,var_classes = NaiveBayes(data_matrix,data_label)
```

a. Compute the prior for the three classes.

```
In [273]: print("Prior probability of C1 = {0}\nPrior probability of C2 = {1}\nPrior probabili
```

```
Prior probability of C1 = 0.4
Prior probability of C2 = 0.28
Prior probability of C3 = 0.32
```

b. Compute the mean and covariance matrix for C1, C2 and C3

```
In [274]: print("Mean of C1 = {0}\nMean of C2 = {1}\nMean of C3 = {2}".format(mean_classes[0],m
```

```
Mean of C1 = [5.8 3.9]
Mean of C2 = [5.28571429 4.        ]
Mean of C3 = [10.5    3.125]
```

```
In [276]: print("Variance of C1 = {0}\nVariance of C2 = {1}\nVariance of C3 = {2}".format(mean_
```

```
Variance of C1 = [5.8 3.9]
Variance of C2 = [5.28571429 4.        ]
Variance of C3 = [10.5    3.125]
```

d. Use the Naive Bayes classifier to classify the samples p1=(6,5), p2=(9,4), p3=(8,5)

```
In [280]: print("Point (6,5) is Class",Predict([6,5],prior_prob,mean_classes,var_classes)+1)
          print("Point (9,4) is Class",Predict([9,4],prior_prob,mean_classes,var_classes)+1)
          print("Point (8,5) is Class",Predict([8,5],prior_prob,mean_classes,var_classes)+1)
```

```
Point (6,5) is Class 2
Point (9,4) is Class 3
Point (8,5) is Class 1
```