Linear classifier – Perceptron

Two classes (Iris-Setosa and Iris-Versicolor) linearly separable are extracted from Iris dataset. We do some adjustments, including adopting different number of training samples and subsets of features, to conduct simulations.

Take 90%, 50%, and 10% of dataset as training samples in Experiment 1-3 where the blue line is the decision boundary. By adopting the trained classifier, testing data can be classified and the accuracy of the classification result is then calculated as shown in Figures 1-3.

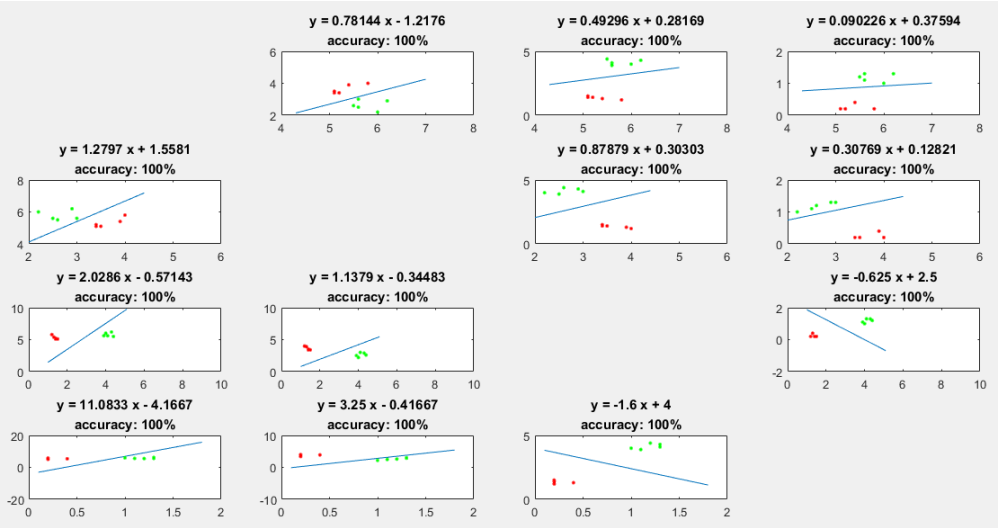

Figure 1. Experiment 1 (90% of dataset for testing).

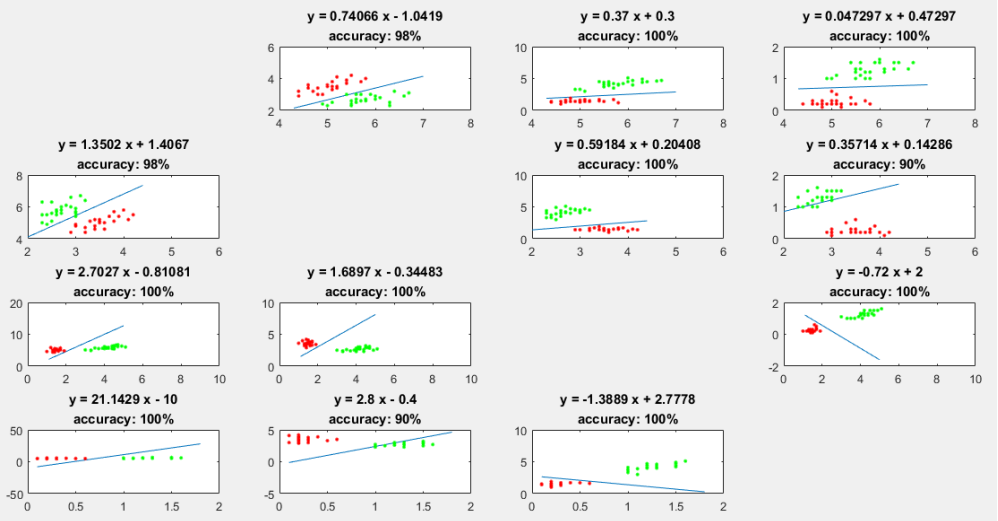

Figure 2. Experiment 2 (50% of dataset for testing).

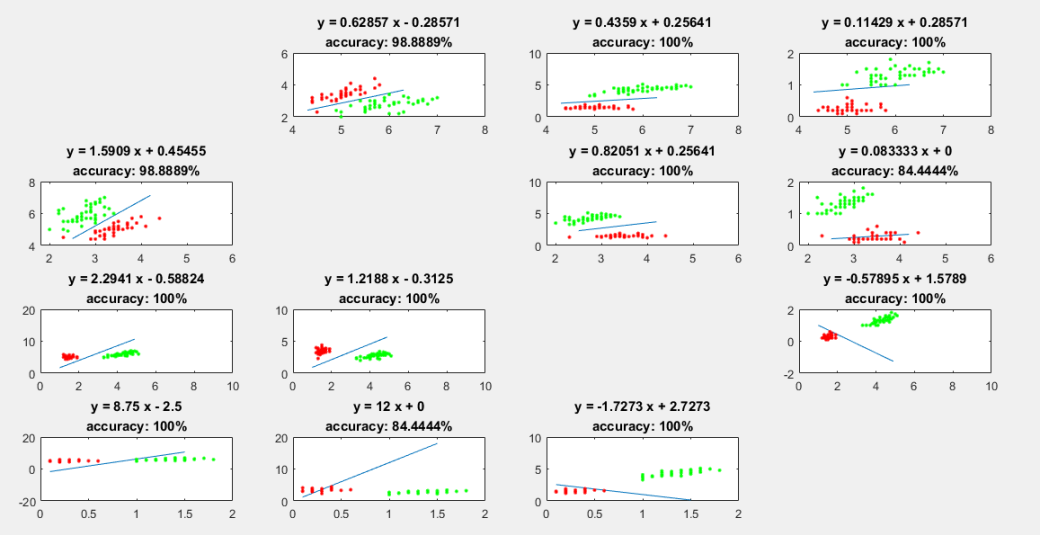

Figure 3. Experiment 3 (10% of dataset for testing).

Observed from Figures 1-3, the trained classifier performs inaccurate classification with a decreased number of training data. In the second experiment, the trained classifier using half of dataset for training gives the accuracy over 90% in average. However, the lower bound accuracy goes to around 84% in Experiment 3. It is expected that the classification accuracy drops fast with less training samples.

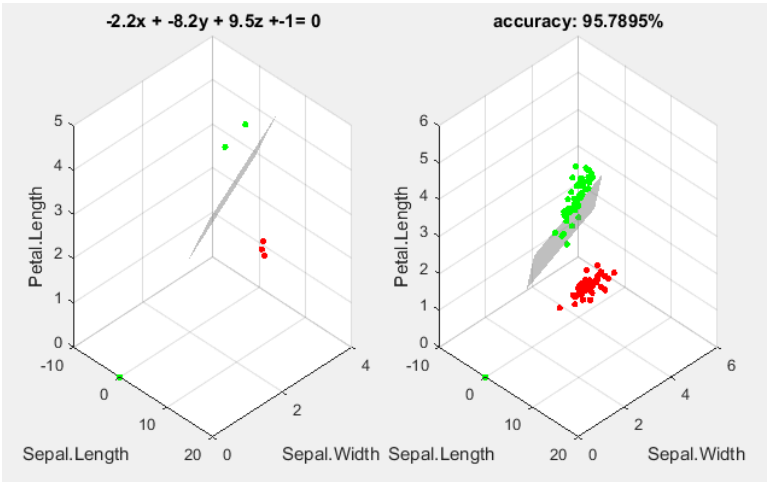

Figure 4. Experiment 4 (5% of dataset for training and three feature involved).

In respect to different subsets of features involved in training process, it shall affect the classification accuracy. The more features for training, the more stable (i.e., gives higher accuracy with less training samples) is the trained classifier. There are three features (attributes) involved and 5% of dataset served as training data in Experiment 4; the classification accuracy is around 96% as shown in Figure 4, where the decision boundary is depicted as gray plane.

In my version of the perceptron classifier, it is under the condition that the training accuracy is 100%. As the result, this classifier performs a higher classification accuracy. The confusion matrix and ROC curve of Experiment 1 are given in Figure 5. Since the classification accuracy is 100%, the ROC curve is the line of the left border and upper border in the ROC space and the area under the ROC curve (i.e., AUC) is equal to one.

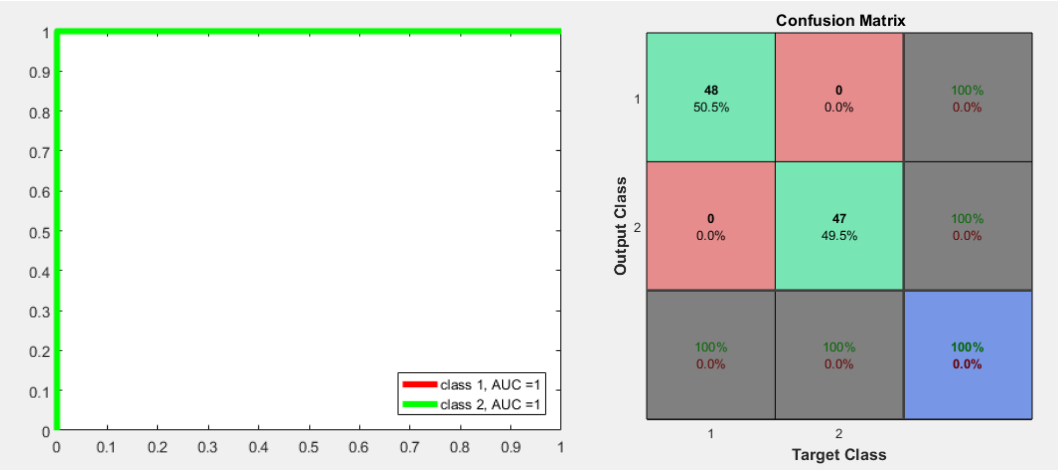

Figure 5. The ROC curve and confusion matrix in most cases.

Naïve Bayes classifier

We take two datasets (Wireless Indoor Localizatio dataset and banknote authentication dataset) to conduct classification simulations. The former dataset is two-class dataset whereas the latter one is mutli-class dataset. The feature space of the first dataset is addressed in Figure 6.

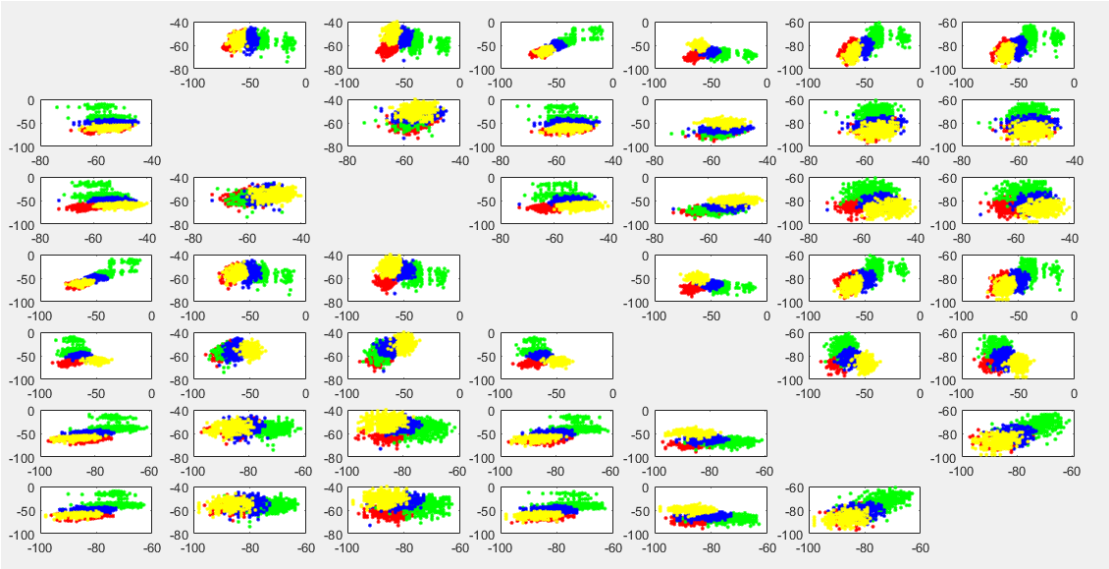

Figure 6. The feature space of Wireless Indoor Localizatio dataset.

We take 90% of the dataset as training samples, and it gives the accuracy round 50%. Involving more features in training process performs an inferior classification if each class is highly overlapped to each other in one dataset. Figure 8 shows the classification result where only the first feature is involved in training process; it gives superior accuracy compared to the simulation as shown in Figure 7.

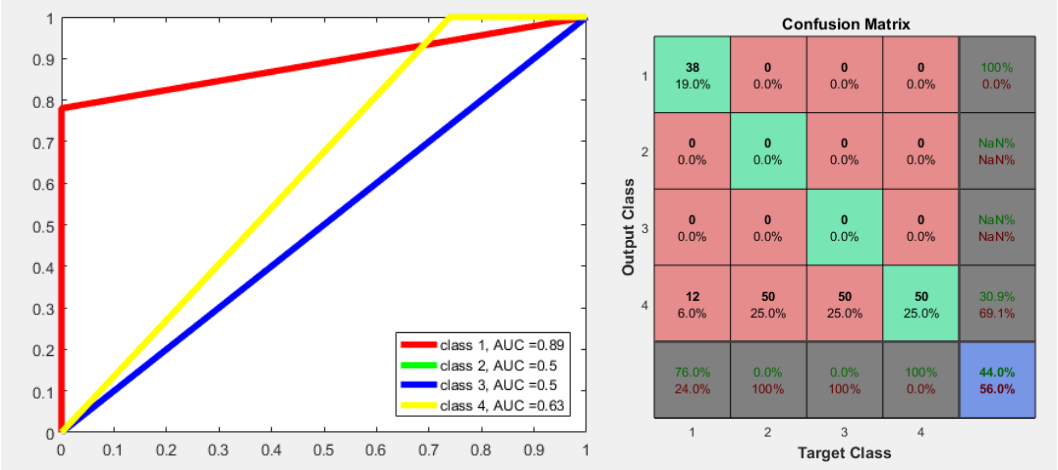

Figure 7. Experiment 5 (90% of dataset for training and all features involved).

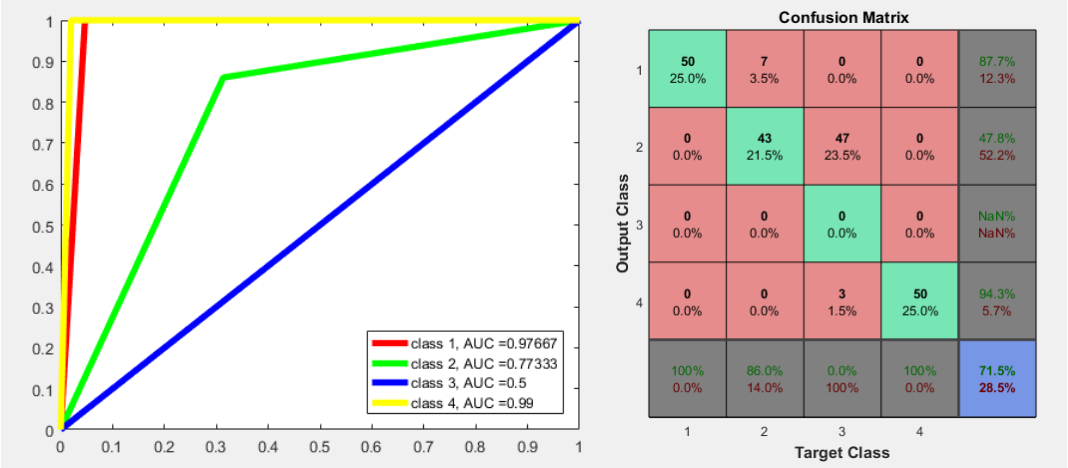

Figure 8. Experiment 6 (90% of dataset for training and the first feature involved).

In respect of the dispersity between each class, the classes in banknote authentication dataset are less overlapped compared to that of Wireless Indoor Localizatio dataset. Take all features into training process, and the classification result is addressed in Figure 10. We can select the pivotal feature(s) and discard others for training to improve the accuracy. Observed from Figure 9, the least overlapped regions between these two classes is located at the first row. Therefore, the first feature can be treated as dominant feature. The simulation is conducted in Experiment 8. Compared with Experiment 9, the accuracy increases over 40%.

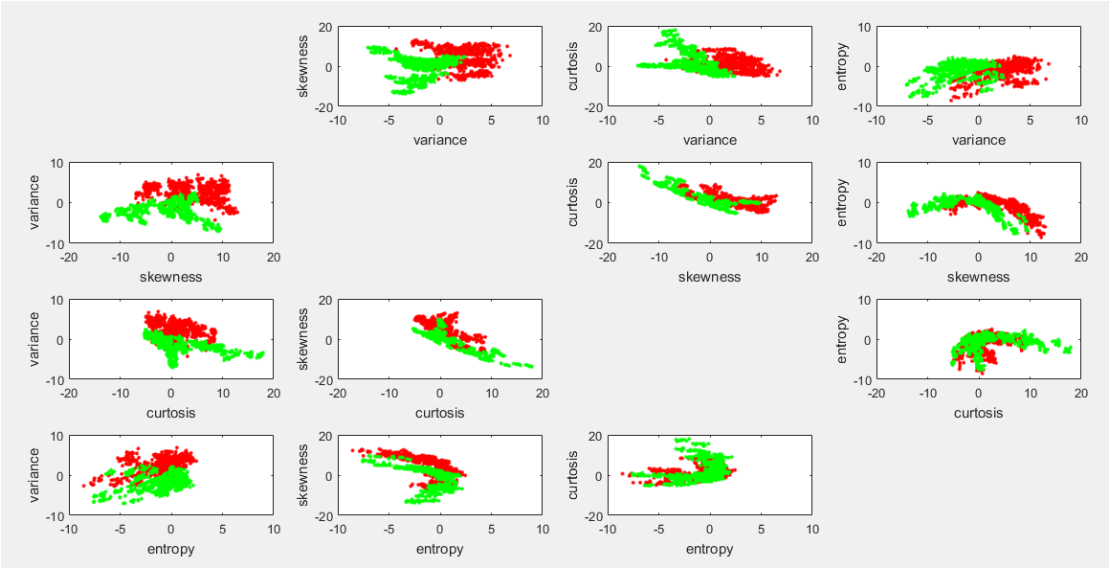

Figure 9. The feature space of banknote authentication dataset.

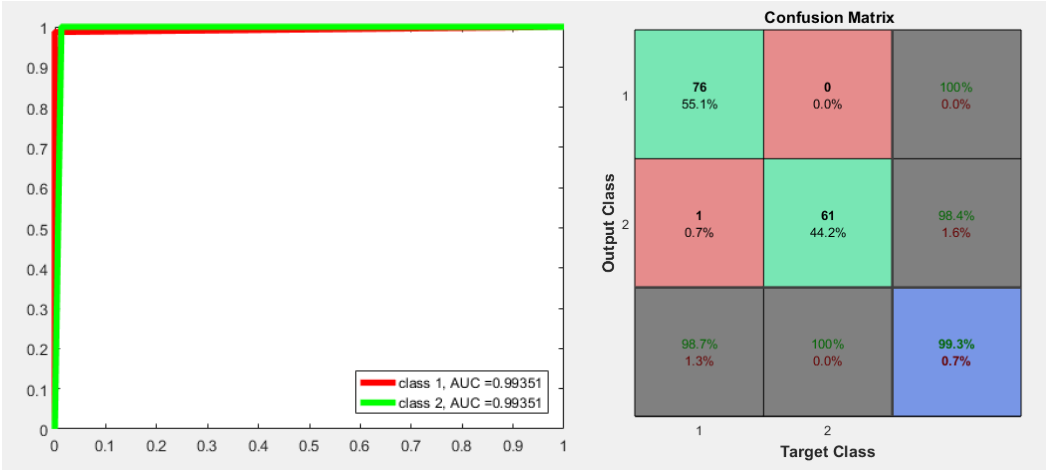

Figure 10. Experiment 7 (90% of dataset for training and all features involved).

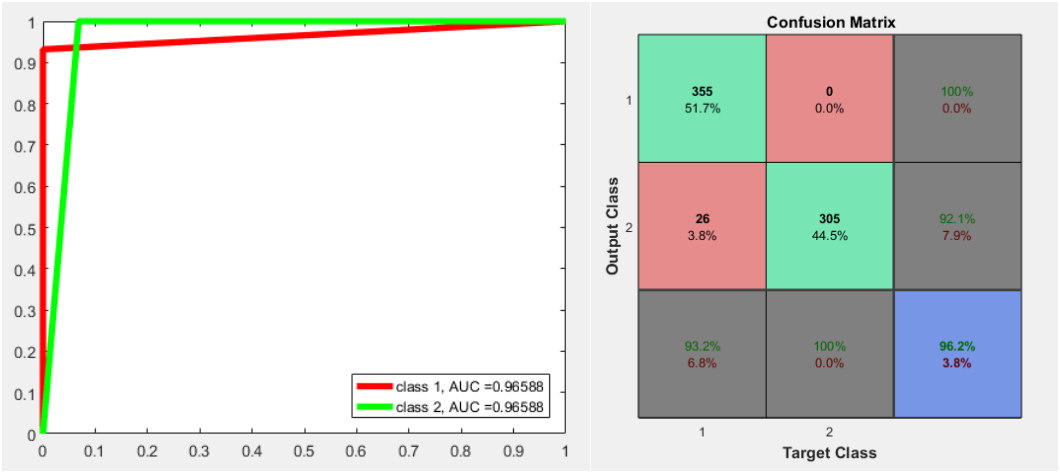

Figure 11. Experiment 8 (50% of dataset for training and the first feature involved).

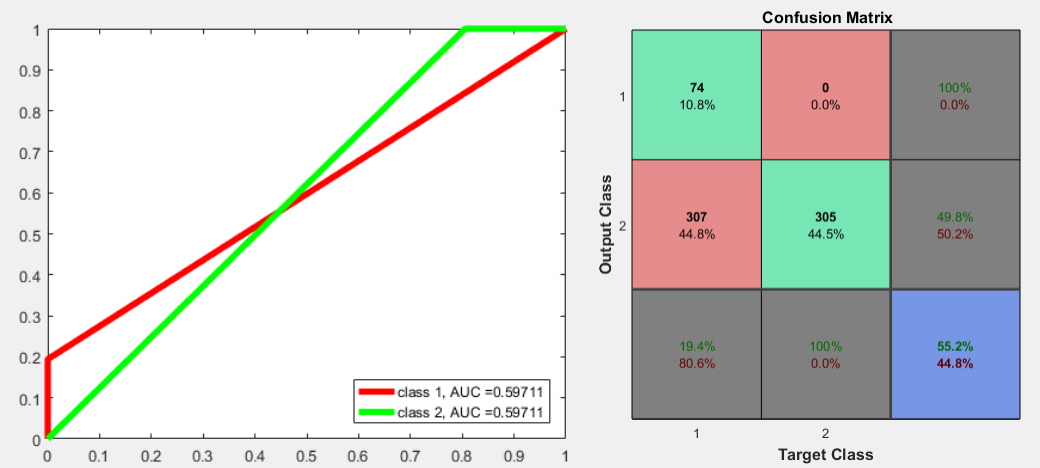

Figure 12. Experiment 9 (50% of dataset for training and all feature involved).

Bayesian classifier

Compared with Naïve Bayes classifier, all features are not treated as independent. That is, all features of each data are treated simultaneously in training process. As the result, it learns not only features but also correlations. We first take the same datasets performed in the previous classifier. There are seven features in Wireless Indoor Localizatio dataset as shown in Figure 6. In this test dataset, Bayesian classifier (45%) gives a little higher accuracy than Naïve Bayes one does (44%) as shown in Figure 13. It can be said that learning the correlated features enhances the classification accuracy. However, it may be over inflating these features and degrade the performance.

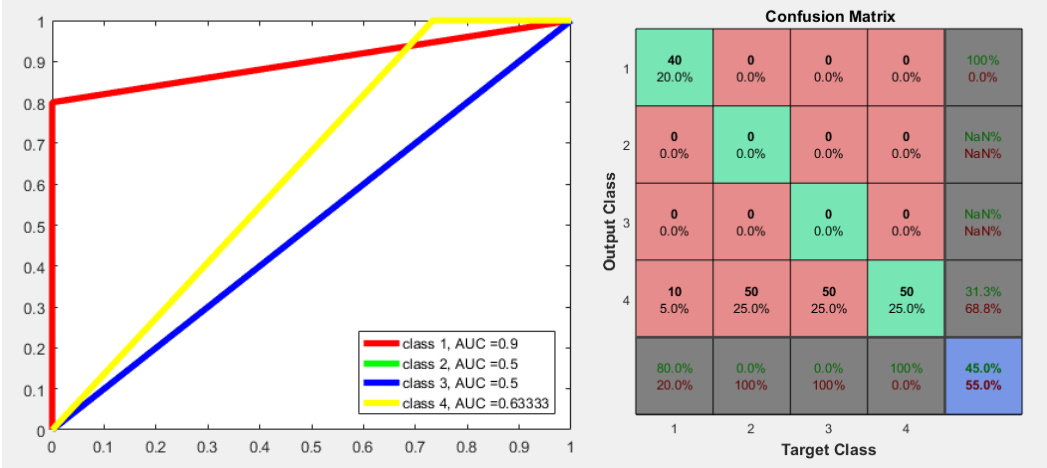

Figure 13. Experiment 10 (90% of dataset for training and all features involved).

The advantages Bayesian classifier holds is that it performs a better accuracy if the dataset has less correlated features. In other normal cases, Bayesian classifier gives the similar result as Naïve Bayes classifier does.

Moreover, we take Haberman's Survival dataset for comparing the performance of the Bayesian classifier and Naïve Bayes classifier. In this dataset, there is only two features and classified into two classes as shown in Figure 14. As mentioned above, Bayesian classifier learns more in training process than Naïve Bayes classifier. This dataset has only two features, and therefore each feature may provide higher misleading information. The classification results performed by Naïve Bayes classifier and Bayesian classifier are presented in Experiment 11 and Experiment 12 respectively. As observed, Naïve Bayes classifier gives the superior performance in this dataset. Notably, the ROC curve in Experiment 12 is under the diagonal regions (AUC is less than 0.5). It means that Bayesian classifier gives an inferior prediction for testing data (even worse than randomly guess).

Figure 14. The feature space of Haberman's Survival dataset.

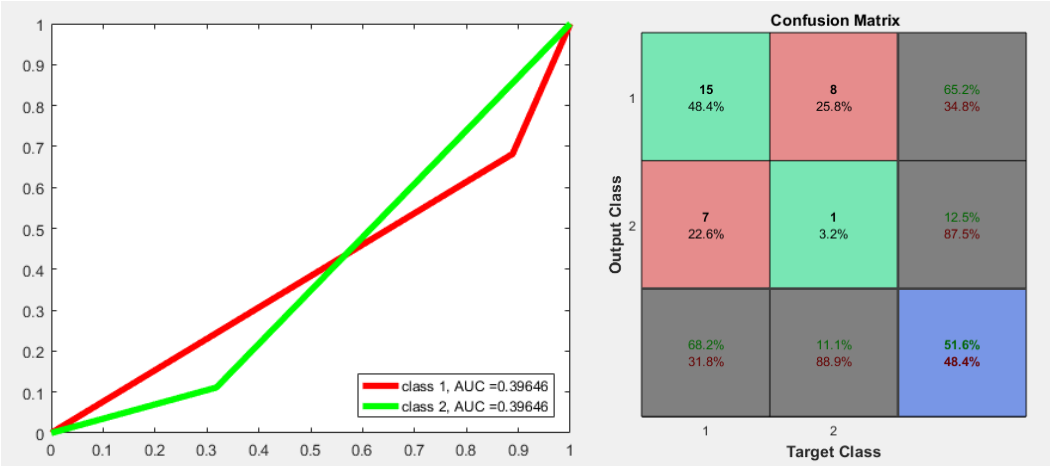

Figure 15. Experiment 11 (90% of dataset for training and all features involved).

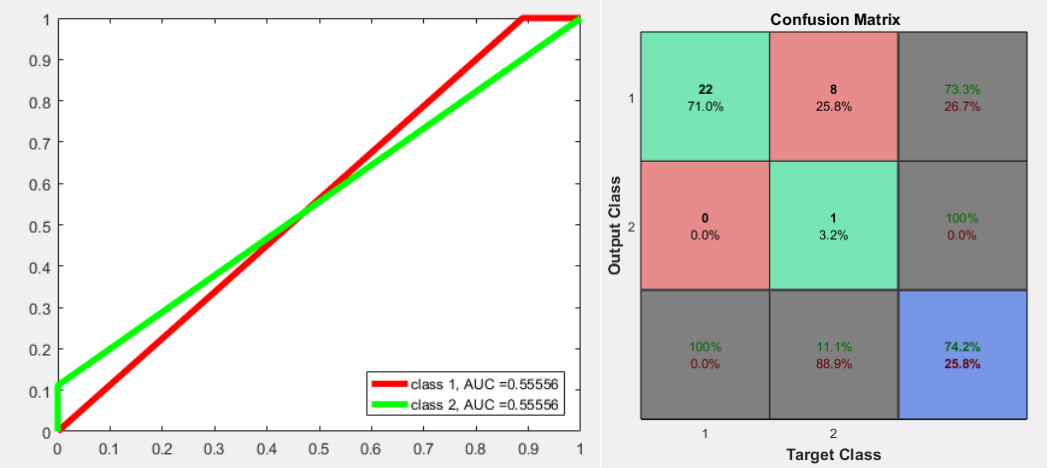

Figure 16. Experiment 12 (90% of dataset for training and all features involved).

Appendix

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%                          load dataset                          %%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[slen,swid,plen,pwid,name]=textread('Iris_2.txt','%f%f%f%f%s','delimiter',',');
C={slen,swid,plen,pwid};
%convert cell array to double array
C=cell2mat(C);
```

The same for three classifiers.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%     corss validation (split training data & testing data)   %%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%KFold -> for corss vailation (fixed ratio: 0.9)
%Holdout -> split training and testing data with different ratio
r=10; %number of cross vaildation
ratio=(r-1)*1.0/r; %the ratio of training samples to testing samples
[dataCount,atrb]=size(C);%number of features except for class name
cv = cvpartition(name,'KFold',r); %(KFold, r) ; (Holdout, ratio)
dataTrain=zeros(floor(length(name)*ratio),atrb);
dataTrainName=cell(floor(length(name)*ratio),1);
dataTest=zeros(length(name)-length(dataTrain),atrb);
dataTestName=cell(length(name)-length(dataTrainName),1);
idx=test(cv,1); %the split result (1/0 represent training/
%show the ratio of each class in dataTrain and dataTest
%tabulate(categorical(dataTrainName)); %tabulate(categorical(dataTestName));
cnt=1;cnt2=1;
for k=1:length(name)
    if(idx(k)==1)
        dataTrain(cnt,:)=C(k,:); dataTrainName(cnt,1)=name(k,1); cnt=cnt+1;
    else
        dataTest(cnt2,:)=C(k,:); dataTestName(cnt2,1)=name(k,1); cnt2=cnt2+1;
    end
```

The same for three classifiers.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%                        feature space                           %%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
figure(1);
Title={'Sepal.Length','Sepal.Width','Petal.Length','Petal.Width'};
for i=1:atrb
  for j=1:atrb
    if(i==j)
        continue;
    end
    subplot(atrb,atrb,(i-1)*atrb+j);
    gscatter(C(:,i),C(:,j),name,'rgb','.',8);
    xlabel(Title(i)); ylabel(Title(j));
    legend('off'); %turn off the legend
  end
end
```

The same for three classifiers.

Perceptron classifier (two features involved)

```
%//////////////////////////////////////////////////////////////////////////%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%      training model ("two" features are involoved)          %%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%initial weight vector array
%there are (atrb*atrb-atrb) weight vector [w1i,w2i,w3i] in w
%where 1<=i<=(atrb*atrb-atrb)
w=zeros(atrb*atrb-atrb,3);

%weight vector index, where windex is the same as "i" defined above
widx=1;

for i=1:atrb
  for j=1:atrb
    if(i==j); continue; end
    correct=0; % the number of correctly classified data
    %to find the value of the weight vector [wi1, wi2, wi3]
    %note that all the training data should correctly classified
    while(correct~=length(dataTrainName))
        for k=1:length(dataTrainName)
            %do adjustment when meeting a misclassified data
            if(w(widx,1)+dataTrain(k,i)*w(widx,2)+dataTrain(k,j)*w(widx,3)>0)
                if(strcmp(dataTrainName(k),'Iris-setosa'))
                    w(widx,:)=w(widx,:)-[1,dataTrain(k,i),dataTrain(k,j)];
                    correct=0;
                    break;
                else
                    correct=correct+1;
                end
            end

            %do adjustment when meeting a misclassified data
            if(w(widx,1)+dataTrain(k,i)*w(widx,2)+dataTrain(k,j)*w(widx,3)<=0)
               if(strcmp(dataTrainName(k),'Iris-versicolor'))
                    w(widx,:)=w(widx,:)+[1,dataTrain(k,i),dataTrain(k,j)];
                    correct=0;
                    break;
               else
                   correct=correct+1;
               end
            end
        end
    end

    %plot the discrimination function by weight vector
    %x*w2+y*w3=-1*w1
    figure(2);
    x=min(dataTrain(:,i)):0.01:max(dataTrain(:,i));
    y=-w(widx,2)./w(widx,3).*x-w(widx,1)./w(widx,3);
    subplot(atrb,atrb,(i-1)*atrb+j);
    plot(x,y); hold on
```

```
101         %show the all training samples
102 -       gscatter(dataTrain(:,i),dataTrain(:,j),dataTrainName,'rg','.',8);
103
104         %add title to each subplot
105 -       if(-w(widx,1)/w(widx,3)>=0)
106 -           title(['y = ',num2str(-w(widx,2)./w(widx,3)),' x + ',...
107                 num2str(-w(widx,1)./w(widx,3))]);
108 -       else
109 -           title(['y = ',num2str(-w(widx,2)./w(widx,3)),' x - ',....
110                 num2str(w(widx,1)./w(widx,3))]);
111 -       end
112 -       legend('off'); %turn off the legend
113 -       widx=widx+1;
114 -     end
115 -   end
117     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
118     %%%%%%            calculate the classification accuracy            %%%%%%
119     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
120 -   widx=1;
121 -   for i=1:atrb
122 -     for j=1:atrb
123 -       wrong=0; %the number of the misclassified data
124 -       if(i==j); continue; end
125 -       for k=1:length(dataTestName)
126 -           if(w(widx,1)+dataTest(k,i)*w(widx,2)+dataTest(k,j)*w(widx,3)>0)
127 -               if(strcmp(dataTestName(k),'Iris-setosa'))
128 -                   wrong=wrong+1;
129 -               end
130 -           end
131 -           if(w(widx,1)+dataTest(k,i)*w(widx,2)+dataTest(k,j)*w(widx,3)<=0)
132 -               if(strcmp(dataTestName(k),'Iris-versicolor'))
133 -                   wrong=wrong+1;
134 -               end
135 -           end
136 -       end
137
138 -       accuracy=(length(dataTestName)-wrong)/length(dataTestName);
140         %plot the discrimination function
141 -       figure(3)
142 -       x=min(dataTrain(:,i)):0.01:max(dataTrain(:,i));
143 -       y=-w(widx,2)./w(widx,3).*x-w(widx,1)./w(widx,3);
144 -       subplot(atrb,atrb,(i-1)*atrb+j);
145 -       plot(x,y); hold on
```

```
147         %show the all testing samples
148 -       gscatter(dataTest(:,i),dataTest(:,j),dataTestName,'rg','.',8);
149
150         %add title to each subplot
151 -       if(-w(widx,1)/w(widx,3)>=0)
152 -           title({['y = ',num2str(-w(widx,2)./w(widx,3)),' x + ',...
153                 num2str(-w(widx,1)./w(widx,3))];...
154                 ['accuracy: ',num2str(accuracy*100),'%']});
155 -       else
156 -           title({['y = ',num2str(-w(widx,2)./w(widx,3)),' x - ',...
157                 num2str(w(widx,1)./w(widx,3))];...
158                 ['accuracy: ',num2str(accuracy*100),'%']});
159 -       end
160 -       legend('off'); %turn off the legend
161 -       widx=widx+1;
162 -     end
163 - end
```

Perceptron classifier (three features involved)

```
211     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
212     %%%%%%      training model ("three" features are involoved)        %%%%%%
213     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
214 -   correct=0; % the number of correctly classified data
215 -   w=[0 0 0 0]; %initial feature weight vector
216
217     %to find the value of the weight vector
218     %all the training data are correctly classified
219 -   while(correct~=length(dataTrainName))
220 -       for k=1:length(dataTrainName)
221             %do adjustment when meeting a misclassified data
222 -           if(w(1)+dataTrain(k,1)*w(2)+dataTrain(k,2)*w(3)+dataTrain(k,3)*w(4)>0)
223 -               if(strcmp(dataTrainName(k),'Iris-setosa'))
224 -                   w=w-[1,dataTrain(k,1),dataTrain(k,2),dataTrain(k,3)];
225 -                   correct=0;
226 -                   break;
227 -               else
228 -                   correct=correct+1;
229 -               end
230 -           end

231             %do adjustment when meeting a misclassified data
232 -           if(w(1)+dataTrain(k,1)*w(2)+dataTrain(k,2)*w(3)+dataTrain(k,3)*w(4)<=0)
233 -              if(strcmp(dataTrainName(k),'Iris-versicolor'))
234 -                   w=w+[1,dataTrain(k,1),dataTrain(k,2),dataTrain(k,3)];
235 -                   correct=0;
236 -                   break;
237 -              else
238 -                  correct=correct+1;
239 -              end
240 -           end
241 -       end
242 -   end
```

```
244     %plot the discrimination function by weight vector
245     %x*w2+y*w3+z*w4=1*w1
246 -   Y=min(dataTrain(:,2)):0.01:max(dataTrain(:,2));
247 -   Z=min(dataTrain(:,3)):0.01:max(dataTrain(:,3));
248 -   [y,z]=meshgrid(Y,Z);
249 -   x=-w(3)./w(2).*y+w(1)./w(2)-w(4)./w(2).*z;
250 -   surf(x,y,z,'FaceColor',[0.5 0.5 0.5],'EdgeColor','none','FaceAlpha' ,0.5);
251     %colorbar
252 -   title([num2str(w(2)),'x + ',num2str(w(3)), 'y + ',...
253         num2str(w(4)),'z +',num2str(w(1)),'= 0']);
```

```
%//////////////////////////////////////////////////////////////////////////%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%      classify dataTrain into dataTrain1 & dataTrain2        %%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%because i dont know how to design the color mapping function in scatter3
dataTrain1=zeros(length(name),4);
dataTrain2=zeros(length(name),4);

cnt=1;cnt2=1;
for i=1:length(dataTrainName)
    if(strcmp(dataTrainName(i),'Iris-setosa'))
        dataTrain1(cnt,:)=dataTrain(i,:);
        cnt=cnt+1;
    end
    if(strcmp(dataTrainName(i),'Iris-versicolor'))
        dataTrain2(cnt2,:)=dataTrain(i,:);
        cnt2=cnt2+1;
    end
end

figure(6);
%Title={'Sepal.Length','Sepal.Width','Petal.Length','Petal.Width'};
subplot(1,2,1);
scatter3(dataTrain1(:,1),dataTrain1(:,2),dataTrain1(:,3),16,[1 0 0],'filled'); hold on
scatter3(dataTrain2(:,1),dataTrain2(:,2),dataTrain2(:,3),16,[0 1 0],'filled');
xlabel(Title(1)); ylabel(Title(2)); zlabel(Title(3)); view(45,30);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%        classify dataTest into dataTest1 & dataTest2         %%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%because i dont know how to design the color mapping function in scatter3
dataTest1=zeros(length(name),4);
dataTest2=zeros(length(name),4);

cnt=1;cnt2=1;
for i=1:length(dataTestName)
    if(strcmp(dataTestName(i),'Iris-setosa'))
        dataTest1(cnt,:)=dataTest(i,:);
        cnt=cnt+1;
    end
    if(strcmp(dataTestName(i),'Iris-versicolor'))
        dataTest2(cnt2,:)=dataTest(i,:);
        cnt2=cnt2+1;
    end
end
%show the all testing samples
subplot(1,2,2);
scatter3(dataTest1(:,1),dataTest1(:,2),dataTest1(:,3),16,[1 0 0],'filled'); hold on
scatter3(dataTest2(:,1),dataTest2(:,2),dataTest2(:,3),16,[0 1 0],'filled');
xlabel(Title(1)); ylabel(Title(2)); zlabel(Title(3)); view(45,30);
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%            calculate the classification accuracy            %%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
wrong=0; % the number of the misclassified data
for k=1:length(dataTestName)
    if(w(1)+dataTest(k,1)*w(2)+dataTest(k,2)*w(3)+dataTest(k,3)*w(4)>0)
        if(strcmp(dataTestName(k),'Iris-setosa'))
            wrong=wrong+1;
        end
    end
    if(w(1)+dataTest(k,1)*w(2)+dataTest(k,2)*w(3)+dataTest(k,3)*w(4)<=0)
        if(strcmp(dataTestName(k),'Iris-versicolor'))
            wrong=wrong+1;
        end
    end
end
accuracy=(length(dataTestName)-wrong)/length(dataTestName);
surf(x,y,z,'FaceColor',[0.5 0.5 0.5],'EdgeColor','none','FaceAlpha' ,0.5);
%colorbar
title(['accuracy: ',num2str(accuracy*100),'%']);
```

Preprocessing of the following two classifiers

```
52      %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
53      %%%%%%          classify training data into dataTrain1/2/3/4          %%%%%%
54      %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
55 -    cnt=1;cnt2=1;cnt3=1;cnt4=1;
56 -    nbcls=4;
57      %find the size of each class in training data
58 -    for i=1:length(dataTrainName)
59 -        if(dataTrainName(i)==1); cnt=cnt+1;
60 -        elseif(dataTrainName(i)==2); cnt2=cnt2+1;
61 -        elseif(dataTrainName(i)==3); cnt3=cnt3+1;
62 -        elseif(dataTrainName(i)==4); cnt4=cnt4+1;
63 -        end
64 -    end
65 -    dataTrain1=zeros(cnt-1,atrb); dataTrain2=zeros(cnt2-1,atrb);
66 -    dataTrain3=zeros(cnt3-1,atrb); dataTrain4=zeros(cnt4-1,atrb);
68      %classify
69 -    cnt=1;cnt2=1;cnt3=1; cnt4=1;
70 -    for i=1:length(dataTrainName)
71 -        if(dataTrainName(i)==1)
72 -            dataTrain1(cnt,:)=dataTrain(i,:); cnt=cnt+1;
73 -        elseif(dataTrainName(i)==2)
74 -            dataTrain2(cnt2,:)=dataTrain(i,:); cnt2=cnt2+1;
75 -        elseif(dataTrainName(i)==3)
76 -            dataTrain3(cnt3,:)=dataTrain(i,:); cnt3=cnt3+1;
77 -        elseif(dataTrainName(i)==4)
78 -            dataTrain4(cnt4,:)=dataTrain(i,:); cnt4=cnt4+1;
79 -        end
80 -    end
82      %prior probability of each class
83 -    p1=cnt/length(dataTrainName); p2=cnt2/length(dataTrainName);
84 -    p3=cnt3/length(dataTrainName); p4=cnt4/length(dataTrainName);
85
86      %calculate the variance of each class
87 -    dataTrainVar1=var(dataTrain1); dataTrainVar2=var(dataTrain2);
88 -    dataTrainVar3=var(dataTrain3); dataTrainVar4=var(dataTrain4);
89
90      %calculate the mean of each class
91 -    dataTrainMean1=mean(dataTrain1); dataTrainMean2=mean(dataTrain2);
92 -    dataTrainMean3=mean(dataTrain3); dataTrainMean4=mean(dataTrain4);
93
94      %calculate the variance of each class
95 -    dataTrainCov1=cov(dataTrain1); dataTrainCov2=cov(dataTrain2);
96 -    dataTrainCov3=cov(dataTrain3); dataTrainCov4=cov(dataTrain4);
```

Naïve Bayes classifier

```
98      %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
99      %%%%%%                    Naive Bayes classifier                    %%%%%%
100     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
101 -   wrong=0; prediction=0;
102 -   pdarray=zeros(length(dataTestName),1);
103 -   for i=1:length(dataTestName)
104         %baye's rule: P(A | B) * P(B) = P(B | A) * P(A)
105         %known information: P(C), P(fn | C)
106         %wanted: P(C | f1,f2,...,fn)
107         %P(C | f1,f2,...,fn) * P(f1,f2,...,fn)
108         %= P(C) * [P(f1 | C) * P(f2 | C) * ... * P(fn | C)]
109         %P(C | f1,f2,...,fn) is the probability of class C given feature1~featuren
110         %P(C) is the ratio of the class C in one dataset
111         %P(f1|C) is the probability of feature1 given class C
112 -       for j=1
113 -           p1=p1*normpdf(dataTest(i,j),dataTrainMean1(1,j),sqrt(dataTrainVar1(1,j)));
114 -           p2=p2*normpdf(dataTest(i,j),dataTrainMean2(1,j),sqrt(dataTrainVar2(1,j)));
115 -           p3=p3*normpdf(dataTest(i,j),dataTrainMean3(1,j),sqrt(dataTrainVar3(1,j)));
116 -           p4=p4*normpdf(dataTest(i,j),dataTrainMean4(1,j),sqrt(dataTrainVar4(1,j)));
117 -       end

118         %highest probability -> the most protable class
119 -       if(p1>p2 && p1>p3 && p1>p4); prediction=1;
120 -       elseif(p2>p1 && p2>p3 && p2>p4); prediction=2;
121 -       elseif(p3>p1 && p3>p2 && p3>p4); prediction=3;
122 -       else; prediction=4;
123 -       end
124
```

Bayesian classifier

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%                       Bayesian classifier                   %%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
wrong=0; prediction=0;
pdarray=zeros(length(dataTestName),1);
for i=1:length(dataTestName)
    %multivariate normal pdf
    %view all features at once and put into mvnpdf
    p1=p1*mvnpdf(dataTest(i,:),dataTrainMean1(:)',dataTrainCov1);
    p2=p2*mvnpdf(dataTest(i,:),dataTrainMean2(:)',dataTrainCov2);
    p3=p3*mvnpdf(dataTest(i,:),dataTrainMean3(:)',dataTrainCov3);
    p4=p4*mvnpdf(dataTest(i,:),dataTrainMean4(:)',dataTrainCov4);

    %maximum liklihood -> the most protable class
    if(p1>p2 && p1>p3 && p1>p4); prediction=1;
    elseif(p2>p1 && p2>p3 && p2>p4); prediction=2;
    elseif(p3>p1 && p3>p2 && p3>p4); prediction=3;
    else; prediction=4;
    end
    pdarray(i)=prediction;
    if(prediction~=dataTestName(i))
        wrong=wrong+1;
    end
end
```

```
129     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
130     %%%%%%                confusion matrix & ROC curve & AUC                %%%%%%
131     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
132 -   target=zeros(nbcls,length(dataTestName));
133 -   output=zeros(nbcls,length(dataTestName));
134 -   for i=1:length(dataTestName)
135 -       if(dataTestName(i)==1); target(1,i)=1;
136 -       elseif(dataTestName(i)==2); target(2,i)=1;
137 -       elseif(dataTestName(i)==3); target(3,i)=1;
138 -       elseif(dataTestName(i)==4); target(4,i)=1;
139 -       end
140 -       if(pdarray(i)==1);  output(1,i)=1;
141 -       elseif(pdarray(i)==2);  output(2,i)=1;
142 -       elseif(pdarray(i)==3);  output(3,i)=1;
143 -       elseif(pdarray(i)==4);  output(4,i)=1;
144 -       end
145 -   end
```

The same for three classifiers.

```
147     %calculate AUC
148 -   [x1,y1,~,auc1]=perfcurve(target(1,:),output(1,:),1);
149 -   [x2,y2,~,auc2]=perfcurve(target(2,:),output(2,:),1);
150 -   [x3,y3,~,auc3]=perfcurve(target(3,:),output(3,:),1);
151 -   [x4,y4,~,auc4]=perfcurve(target(4,:),output(4,:),1);
152 -   figure(2); h=plot(x1,y1,'-r',x2,y2,'-g',x3,y3,'-b',x4,y4,'-y'); %plot ROC curve
153 -   set(h, 'linewidth', 4);
154 -   legend(strcat('class 1, AUC = ',num2str(auc1)),...
155         strcat('class 2, AUC = ',num2str(auc2)),...
156         strcat('class 3, AUC = ',num2str(auc3)),...
157         strcat('class 4, AUC = ',num2str(auc4)),...
158         'Location' , 'southeast');
159 -   figure(3); plotconfusion(target,output); %confusion matrix
```

The same for three classifiers.