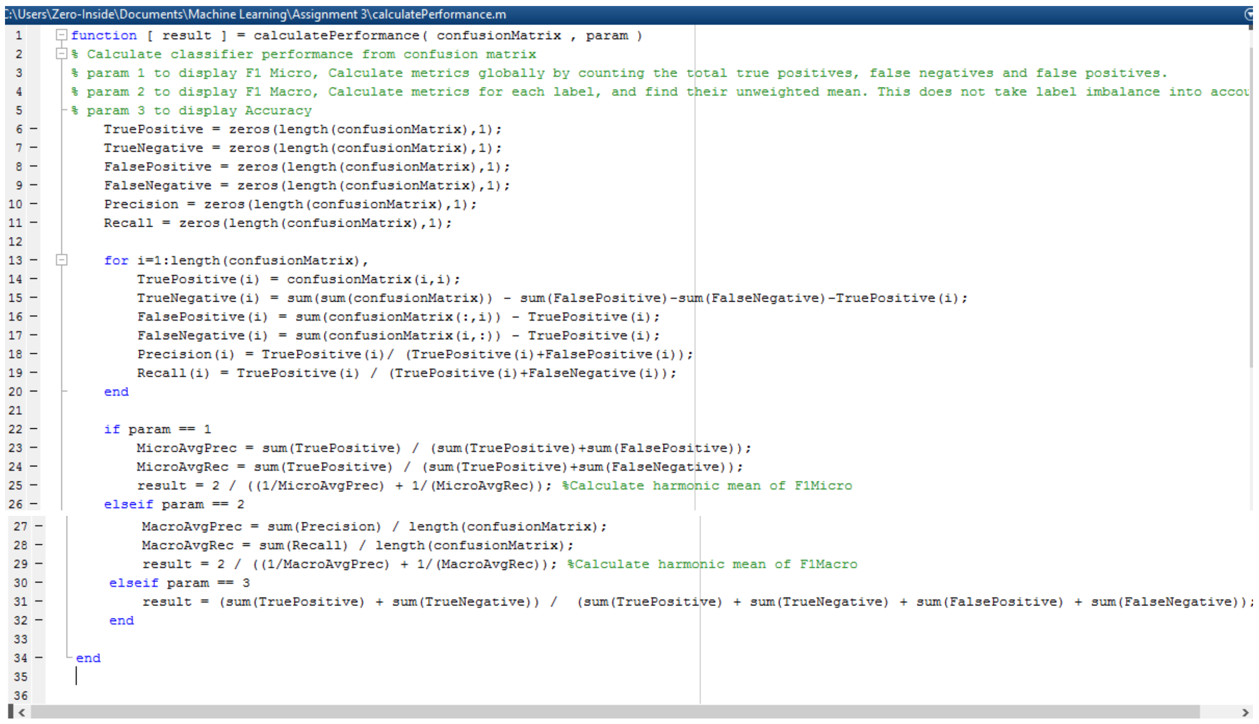
Satrya Budi Pratama
1301154428

1. Calculate Performance



C:\Users\Zero-Inside\Documents\Machine Learning\Assignment 3\calculatePerformance.m

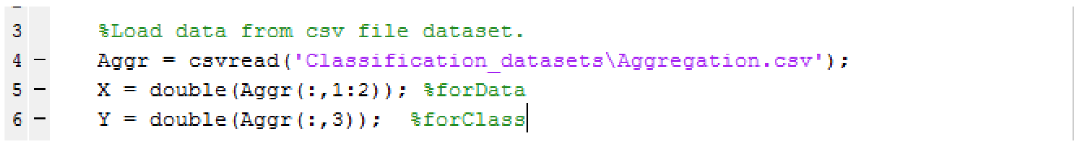
```
function [ result ] = calculatePerformance( confusionMatrix , param )
% Calculate classifier performance from confusion matrix
% param 1 to display F1 Micro, Calculate metrics globally by counting the total true positives, false negatives and false positives.
% param 2 to display F1 Macro, Calculate metrics for each label, and find their unweighted mean. This does not take label imbalance into accou
% param 3 to display Accuracy
    TruePositive = zeros(length(confusionMatrix),1);
    TrueNegative = zeros(length(confusionMatrix),1);
    FalsePositive = zeros(length(confusionMatrix),1);
    FalseNegative = zeros(length(confusionMatrix),1);
    Precision = zeros(length(confusionMatrix),1);
    Recall = zeros(length(confusionMatrix),1);

    for i=1:length(confusionMatrix),
        TruePositive(i) = confusionMatrix(i,i);
        TrueNegative(i) = sum(sum(confusionMatrix)) - sum(FalsePositive)-sum(FalseNegative)-TruePositive(i);
        FalsePositive(i) = sum(confusionMatrix(:,i)) - TruePositive(i);
        FalseNegative(i) = sum(confusionMatrix(i,:)) - TruePositive(i);
        Precision(i) = TruePositive(i)/ (TruePositive(i)+FalsePositive(i));
        Recall(i) = TruePositive(i) / (TruePositive(i)+FalseNegative(i));
    end

    if param == 1
        MicroAvgPrec = sum(TruePositive) / (sum(TruePositive)+sum(FalsePositive));
        MicroAvgRec = sum(TruePositive) / (sum(TruePositive)+sum(FalseNegative));
        result = 2 / ((1/MicroAvgPrec) + 1/(MicroAvgRec)); %Calculate harmonic mean of F1Micro
    elseif param == 2
        MacroAvgPrec = sum(Precision) / length(confusionMatrix);
        MacroAvgRec = sum(Recall) / length(confusionMatrix);
        result = 2 / ((1/MacroAvgPrec) + 1/(MacroAvgRec)); %Calculate harmonic mean of F1Macro
    elseif param == 3
        result = (sum(TruePositive) + sum(TrueNegative)) /  (sum(TruePositive) + sum(TrueNegative) + sum(FalsePositive) + sum(FalseNegative));
    end

end
```

Analisis Program :

Fungsi calculatePerformance untuk menghitung performansi dari classifier dengan inputan berupa confusion matrix dan param yaitu jenis kalkulasi yang ingin ditampilkan, jika param adalah 1 maka menghitung F1 Micro, jika param adalah 2 maka menghitung F1 Macro dan jika param adalah 3 maka menghitung accuracy.

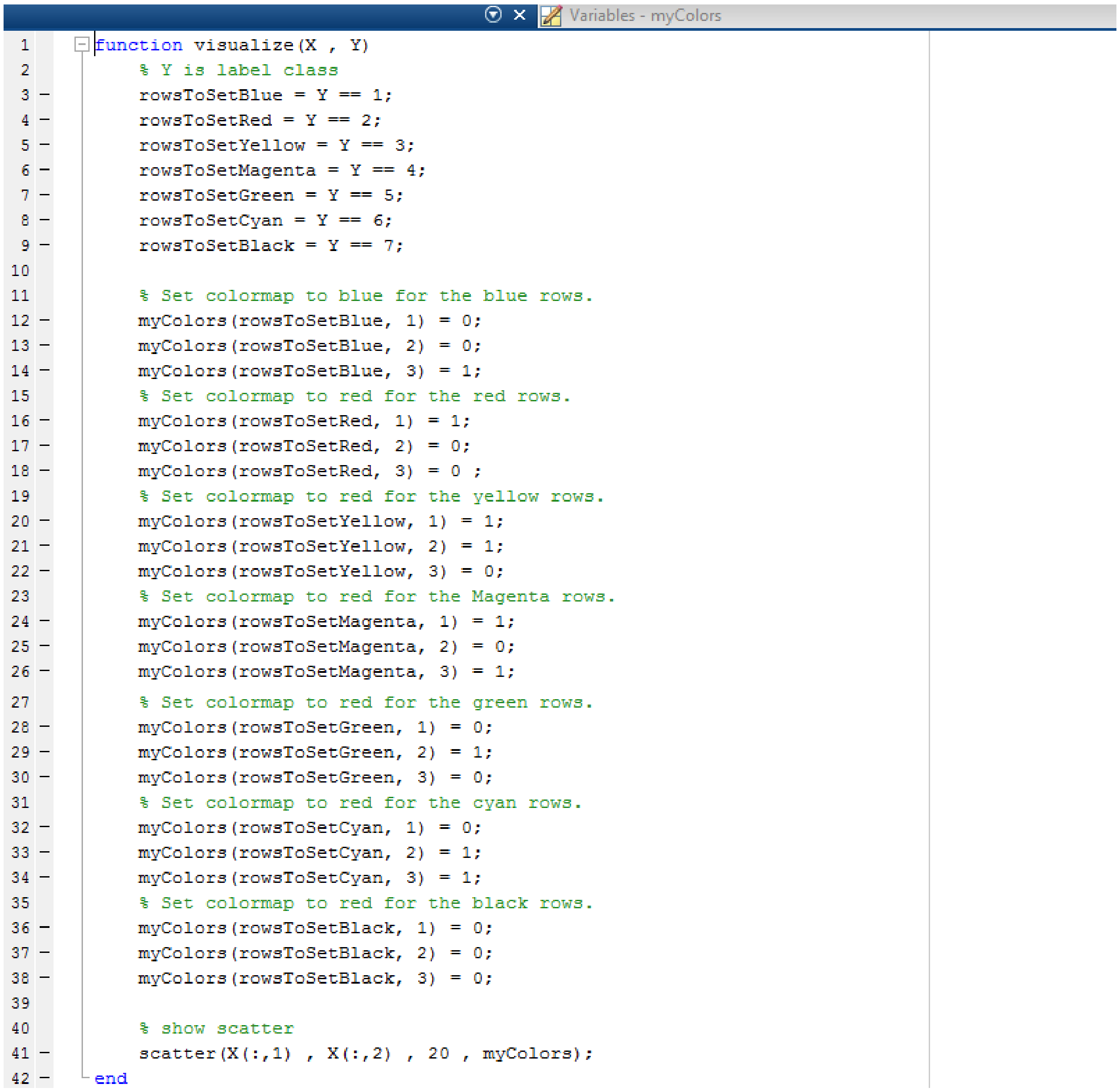
2. Naïve Bayes

a. Load & Visualize Data



```
%Load data from csv file dataset.
Aggr = csvread('Classification_datasets\Aggregation.csv');
X = double(Aggr(:,1:2)); %forData
Y = double(Aggr(:,3));  %forClass
```

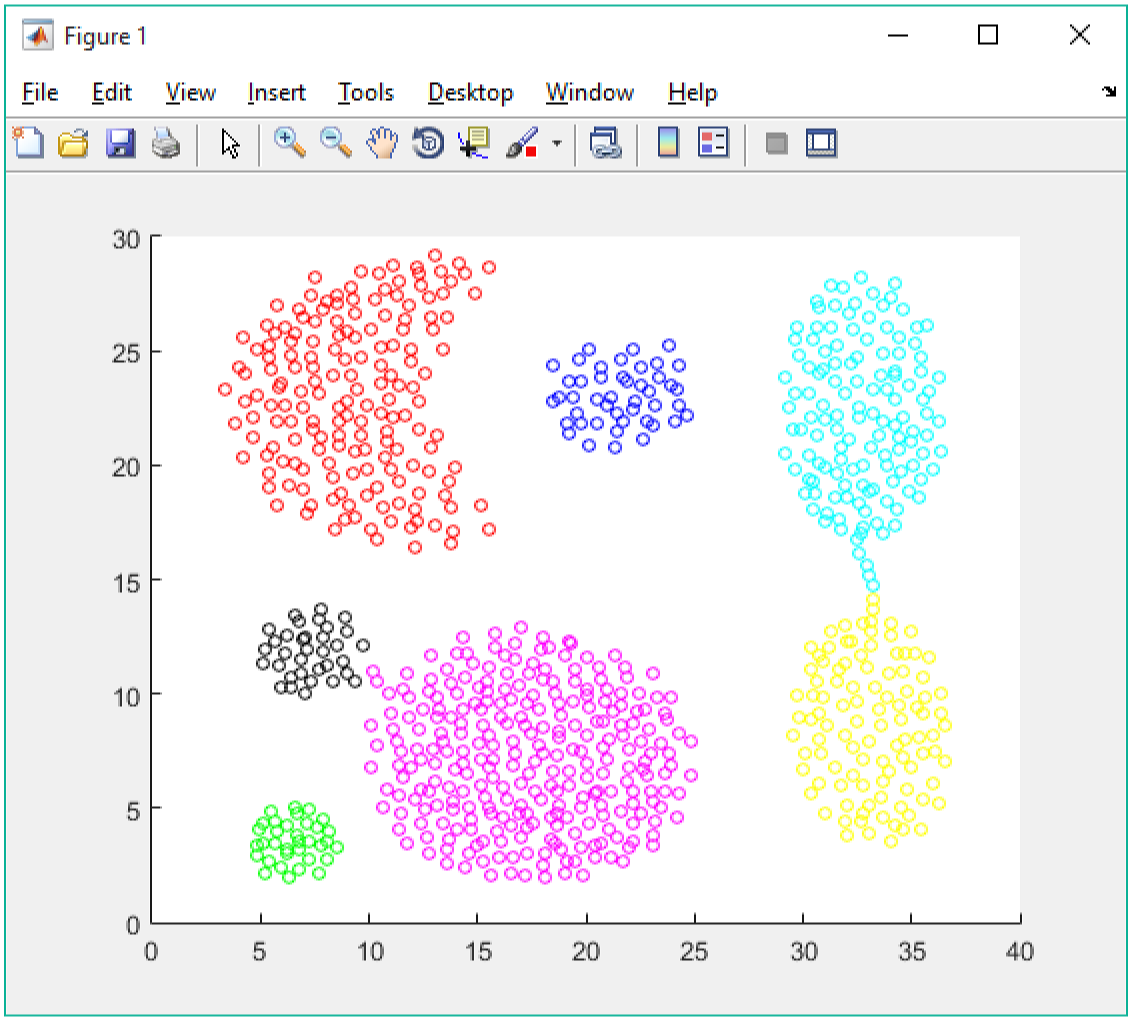
Satrya Budi Pratama
1301154428

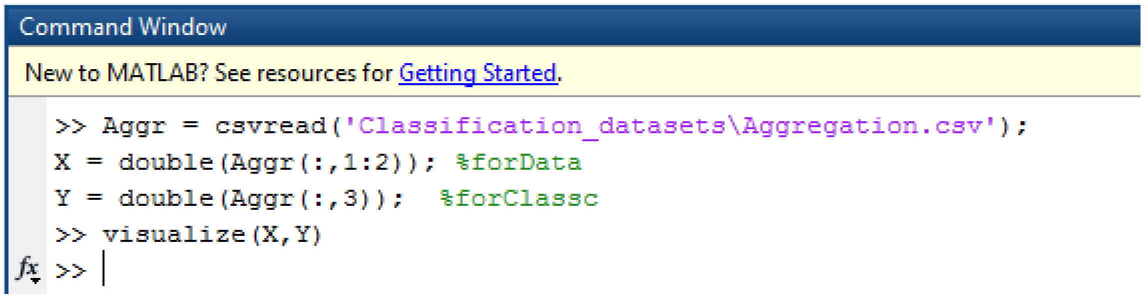


```
function visualize(X , Y)
    % Y is label class
    rowsToSetBlue = Y == 1;
    rowsToSetRed = Y == 2;
    rowsToSetYellow = Y == 3;
    rowsToSetMagenta = Y == 4;
    rowsToSetGreen = Y == 5;
    rowsToSetCyan = Y == 6;
    rowsToSetBlack = Y == 7;

    % Set colormap to blue for the blue rows.
    myColors(rowsToSetBlue, 1) = 0;
    myColors(rowsToSetBlue, 2) = 0;
    myColors(rowsToSetBlue, 3) = 1;
    % Set colormap to red for the red rows.
    myColors(rowsToSetRed, 1) = 1;
    myColors(rowsToSetRed, 2) = 0;
    myColors(rowsToSetRed, 3) = 0 ;
    % Set colormap to red for the yellow rows.
    myColors(rowsToSetYellow, 1) = 1;
    myColors(rowsToSetYellow, 2) = 1;
    myColors(rowsToSetYellow, 3) = 0;
    % Set colormap to red for the Magenta rows.
    myColors(rowsToSetMagenta, 1) = 1;
    myColors(rowsToSetMagenta, 2) = 0;
    myColors(rowsToSetMagenta, 3) = 1;
    % Set colormap to red for the green rows.
    myColors(rowsToSetGreen, 1) = 0;
    myColors(rowsToSetGreen, 2) = 1;
    myColors(rowsToSetGreen, 3) = 0;
    % Set colormap to red for the cyan rows.
    myColors(rowsToSetCyan, 1) = 0;
    myColors(rowsToSetCyan, 2) = 1;
    myColors(rowsToSetCyan, 3) = 1;
    % Set colormap to red for the black rows.
    myColors(rowsToSetBlack, 1) = 0;
    myColors(rowsToSetBlack, 2) = 0;
    myColors(rowsToSetBlack, 3) = 0;

    % show scatter
    scatter(X(:,1) , X(:,2) , 20 , myColors);
end
```

Output :

Satrya Budi Pratama
1301154428





```
>> Aggr = csvread('Classification_datasets\Aggregation.csv');
X = double(Aggr(:,1:2)); %forData
Y = double(Aggr(:,3));  %forClassc
>> visualize(X,Y)
>> 
```

Analisis Program :

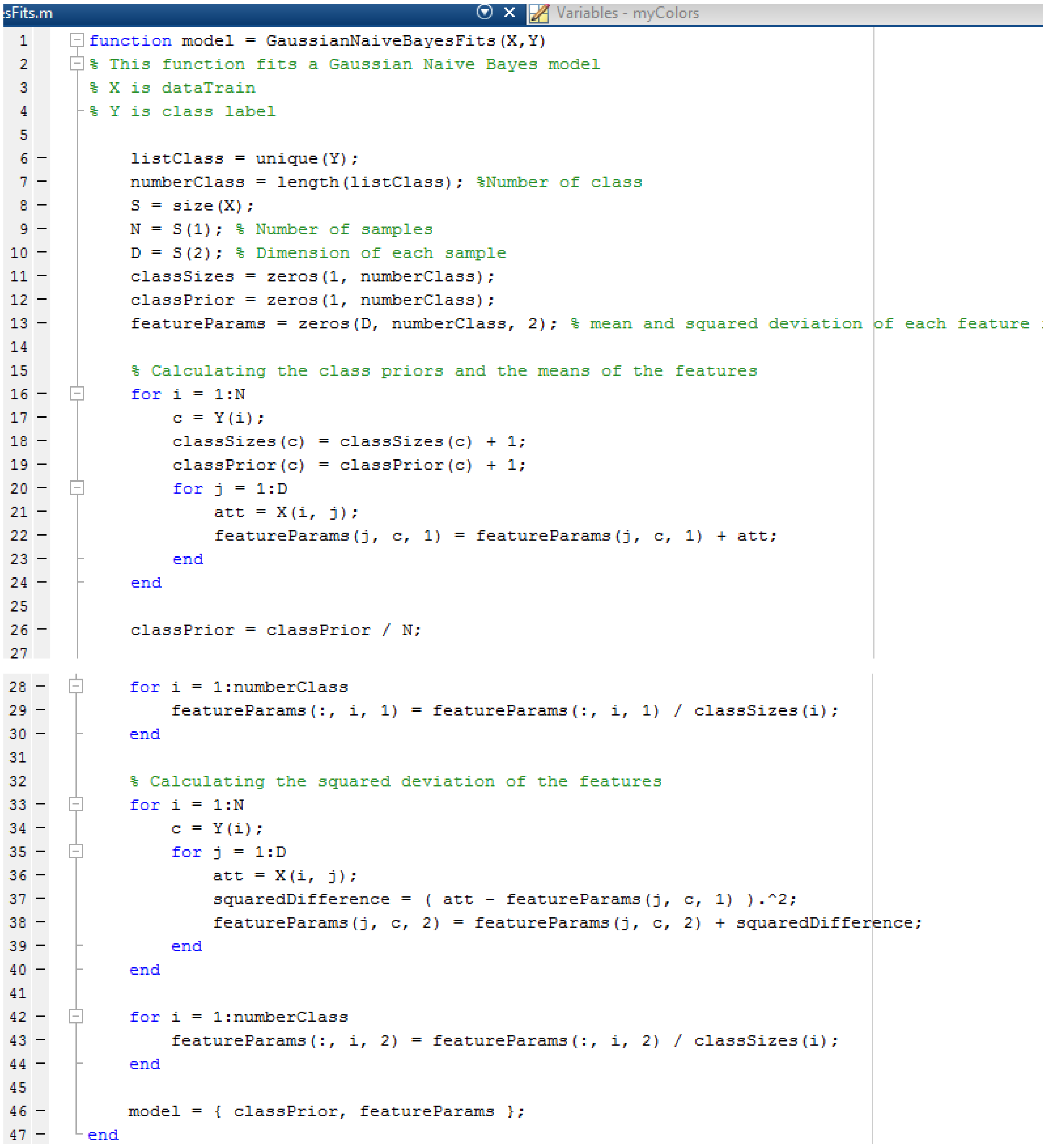
Dengan menggunakan scatter, memvisualisasikan dataset lebih mudah. Terlebih dahulu meload dataset dan membagi data train dan data class label. Kemudian memanggil funsi visualize dengan parameter data dan label. Pada fungsi visualize(x,y) masing-masing data label class disesuaikan dengan data warna RGB. Kemudian memanggil scatter(param1,param2,param3,param4) dengan param1 dan param2 adalah

Satrya Budi Pratama
1301154428

kolom pertama dari data train dan kolom kedua dari data train, kemudian param3 adalah size dari ukuran plot, dan param4 adalah data color.

b. Apply Naïve Bayes

i. Function for Learning



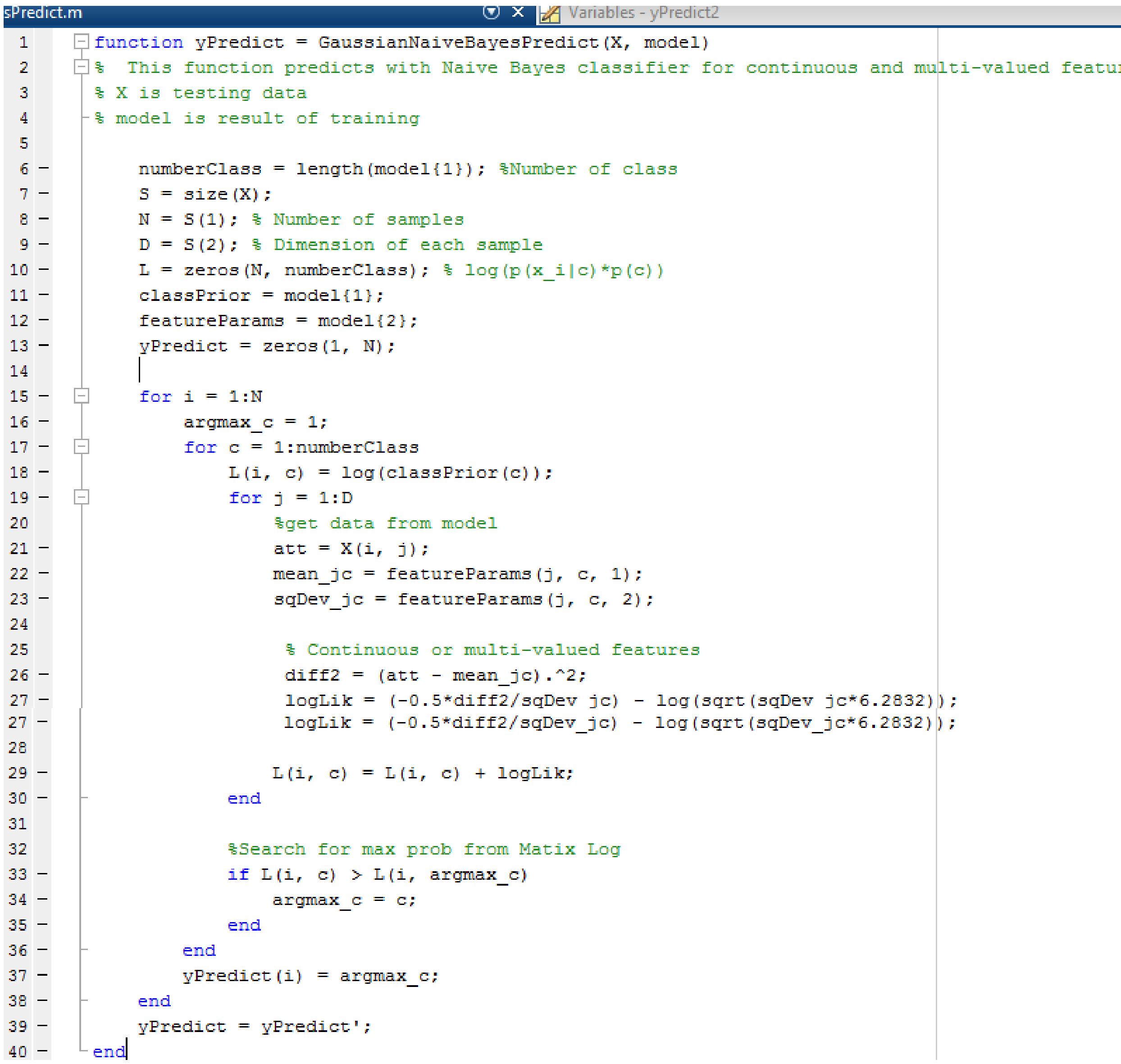
```matlab
function model = GaussianNaiveBayesFits(X,Y)
% This function fits a Gaussian Naive Bayes model
% X is dataTrain
% Y is class label

    listClass = unique(Y);
    numberClass = length(listClass); %Number of class
    S = size(X);
    N = S(1); % Number of samples
    D = S(2); % Dimension of each sample
    classSizes = zeros(1, numberClass);
    classPrior = zeros(1, numberClass);
    featureParams = zeros(D, numberClass, 2); % mean and squared deviation of each feature

    % Calculating the class priors and the means of the features
    for i = 1:N
        c = Y(i);
        classSizes(c) = classSizes(c) + 1;
        classPrior(c) = classPrior(c) + 1;
        for j = 1:D
            att = X(i, j);
            featureParams(j, c, 1) = featureParams(j, c, 1) + att;
        end
    end

    classPrior = classPrior / N;

    for i = 1:numberClass
        featureParams(:, i, 1) = featureParams(:, i, 1) / classSizes(i);
    end

    % Calculating the squared deviation of the features
    for i = 1:N
        c = Y(i);
        for j = 1:D
            att = X(i, j);
            squaredDifference = ( att - featureParams(j, c, 1) ).^2;
            featureParams(j, c, 2) = featureParams(j, c, 2) + squaredDifference;
        end
    end

    for i = 1:numberClass
        featureParams(:, i, 2) = featureParams(:, i, 2) / classSizes(i);
    end

    model = { classPrior, featureParams };
end
```

Analisis Program :

Satrya Budi Pratama
1301154428

Fungsi ini membuat sebuah variable objek dari hasil training. Objek tersebut terdiri dari 2 kolom, yang pertama array class prior yang didapat dari menghitung semua class yang sama pada label dan membaginya dengan total dari y yaitu 788, maka akan didapat prior masing-masing class yaitu 0.0571065989847716 , 0.215736040609137 , 0.129441624365482 , 0.346446700507614, 0.0431472081218274 , 0.164974619289340, 0.0431472081218274. yang kedua adalah likelihood yang terdiri dari array mean dan standart deviasi pada setiap kelas.

ii. Function for Predict



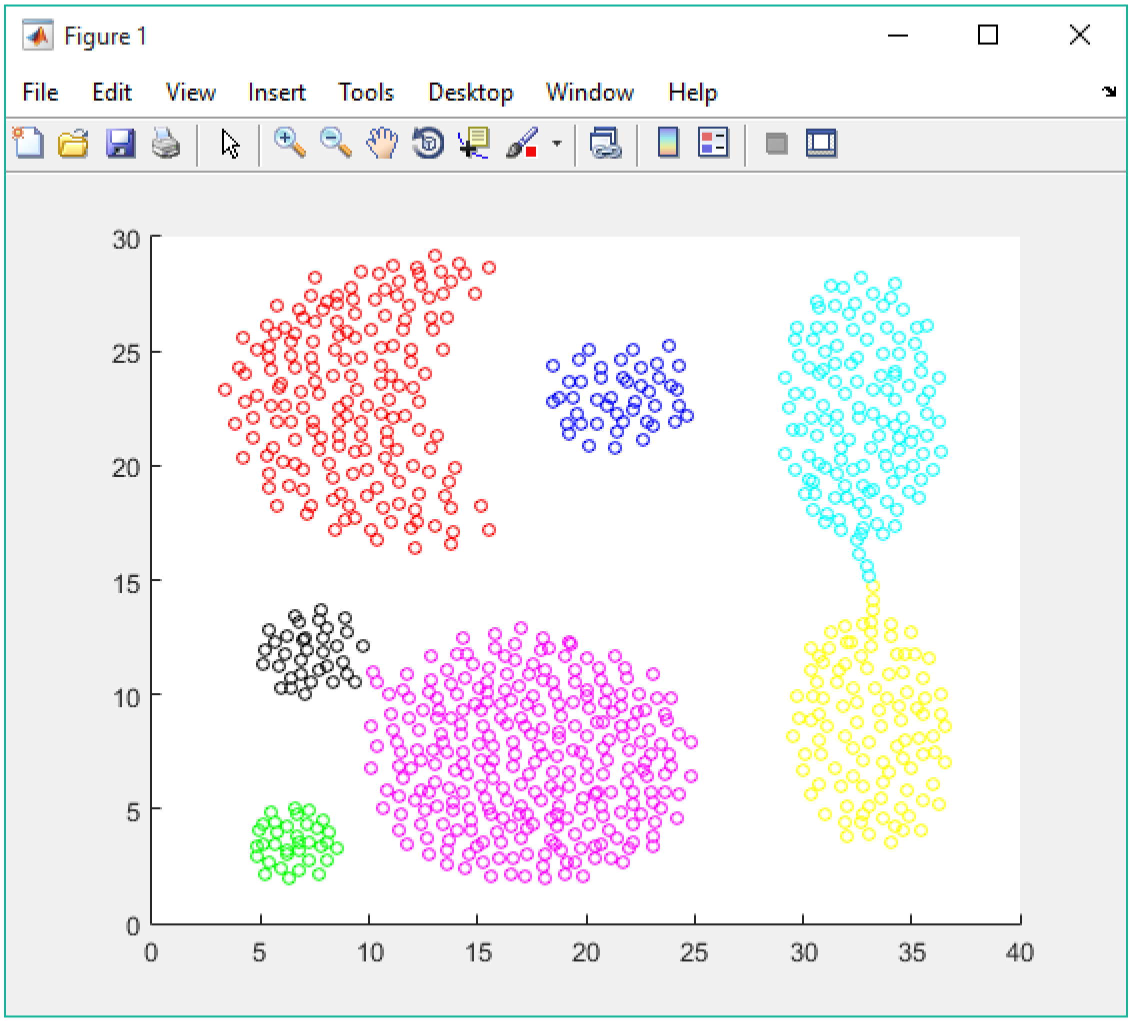
```
function yPredict = GaussianNaiveBayesPredict(X, model)
%  This function predicts with Naive Bayes classifier for continuous and multi-valued featu
% X is testing data
% model is result of training

    numberClass = length(model{1}); %Number of class
    S = size(X);
    N = S(1); % Number of samples
    D = S(2); % Dimension of each sample
    L = zeros(N, numberClass); % log(p(x_i|c)*p(c))
    classPrior = model{1};
    featureParams = model{2};
    yPredict = zeros(1, N);

    for i = 1:N
        argmax_c = 1;
        for c = 1:numberClass
            L(i, c) = log(classPrior(c));
            for j = 1:D
                %get data from model
                att = X(i, j);
                mean_jc = featureParams(j, c, 1);
                sqDev_jc = featureParams(j, c, 2);

                 % Continuous or multi-valued features
                 diff2 = (att - mean_jc).^2;
                 logLik = (-0.5*diff2/sqDev jc) - log(sqrt(sqDev jc*6.2832));
                 logLik = (-0.5*diff2/sqDev_jc) - log(sqrt(sqDev_jc*6.2832));

                L(i, c) = L(i, c) + logLik;
            end

            %Search for max prob from Matix Log
            if L(i, c) > L(i, argmax_c)
                argmax_c = c;
            end
        end
        yPredict(i) = argmax_c;
    end
    yPredict = yPredict';
end
```

Satrya Budi Pratama
1301154428

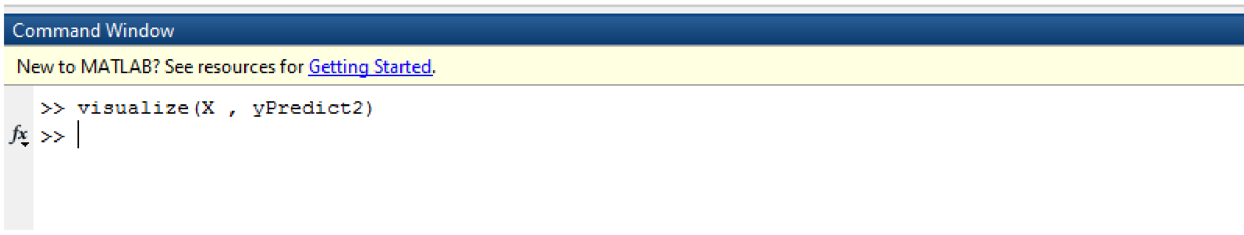
Analisis Program :

Fungsi ini mengeluarkan sebuah vector dari hasil prediksi berisi label class. Label class didapat dari perhitungan data testing dengan data model yang sudah dihasilkan melalui fungsi GaussianNaiveBayesFits. Proses perhitungan posterior yaitu dengan menambahkan hasil log pada classPrior dengan log (-0.5*diff2/sqDev_jc) - log(sqrt(sqDev_jc*6.2832)). Dimana sqDev_jc adalah standart deviasi dari model, dan diff2 diperoleh dengan mengurangkan nilai kolom dengan mean kolom yang ada pada model. Kemudian untuk menentukan class dilakukan perulangan pada setiap data, kemudian membandingkan nilai log setiap kelas mana yang lebih besar maka itu kelasnya.

iii. Plot result of Predict



Satrya Budi Pratama
1301154428



Command Window

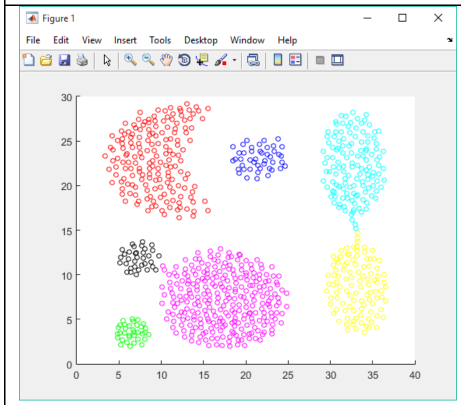
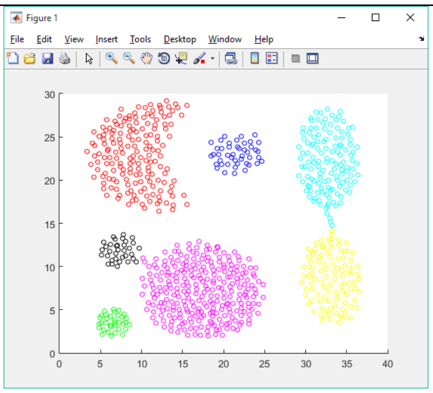
New to MATLAB? See resources for Getting Started.

```
>> visualize(X , yPredict2)
>> 
```

Analisis Program :

Dengan memanggil visualize(x,yPredict2), dimana x adalah train data dan yPredict adalah class yang dihasilkan melalui testing maka ditampilkan scatter yang memplot posisi dari data train.

iv. Compare Visualize

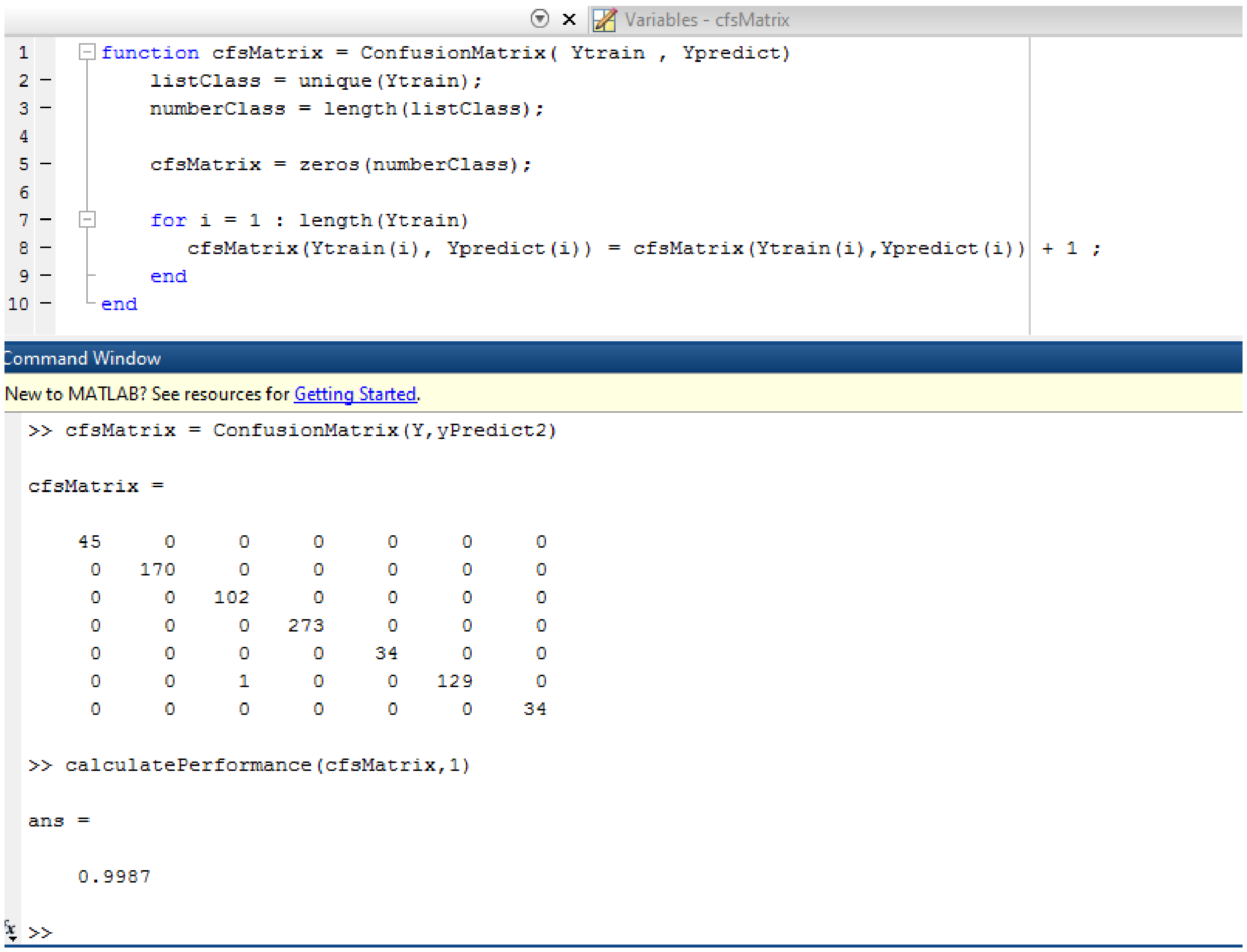
Scatter Data Asli	Scatter Data Prediksi
	

Analisis :

Terlihat ada satu warna yang berubah pada class cyan (6) terhadap class yellow(3). Tetapi secara keseluruhan naïve bayes dapat mengkelompokkan data tersebut sesuai kelasnya dalam visual ini adalah warna.

c. Calculate Performance

Satrya Budi Pratama
1301154428



```
function cfsMatrix = ConfusionMatrix( Ytrain , Ypredict)
    listClass = unique(Ytrain);
    numberClass = length(listClass);

    cfsMatrix = zeros(numberClass);

    for i = 1 : length(Ytrain)
       cfsMatrix(Ytrain(i), Ypredict(i)) = cfsMatrix(Ytrain(i),Ypredict(i)) + 1 ;
    end
end
```

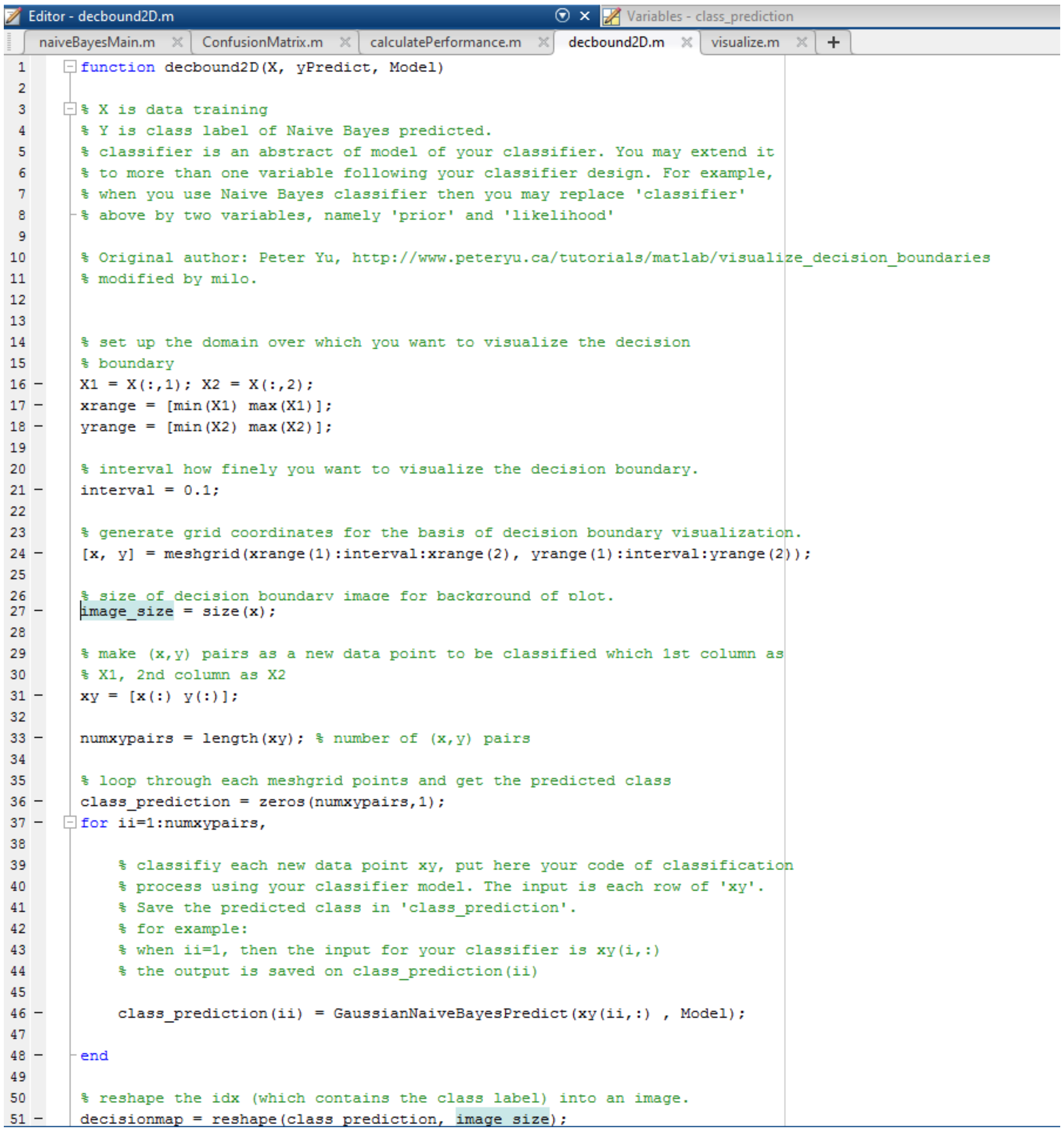
```
>> cfsMatrix = ConfusionMatrix(Y,yPredict2)

cfsMatrix =

    45     0     0     0     0     0     0
     0   170     0     0     0     0     0
     0     0   102     0     0     0     0
     0     0     0   273     0     0     0
     0     0     0     0    34     0     0
     0     0     1     0     0   129     0
     0     0     0     0     0     0    34

>> calculatePerformance(cfsMatrix,1)

ans =

    0.9987

>>
```

Analisis Program :

Untuk menghitung performance terlebih dahulu membuat confusion matrix berdasarkan label data train dan label data testing. Kemudian untuk menghitung performasi memanggil fungsi calculatePerformance(param1, param2). Dengan param1 adalah confusion matrix dan param2 adalah mode atau jenis performasi, 1 untuk F1 Micro 2 untuk F2 Macro dan 3 untuk Accuracy. Berdasarkan hasil perhitungan didapat hasil klasifikasi naïve bayes adalah 0.9987. atau dalam persen adalah 99.87%.

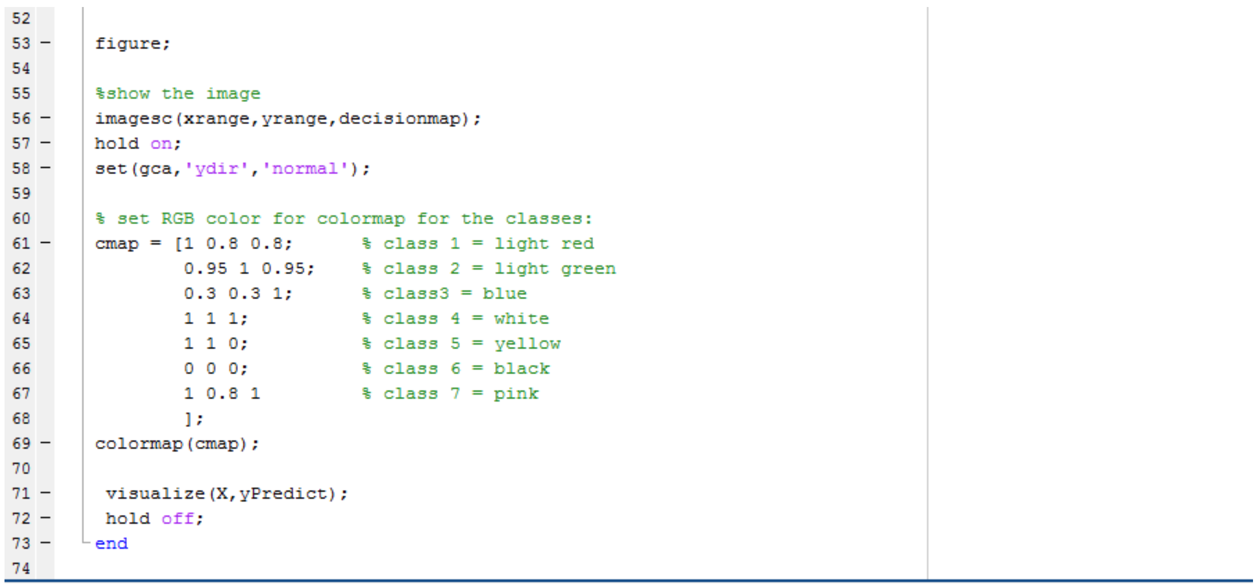
d. Plot Decision Boundary

Satrya Budi Pratama
1301154428



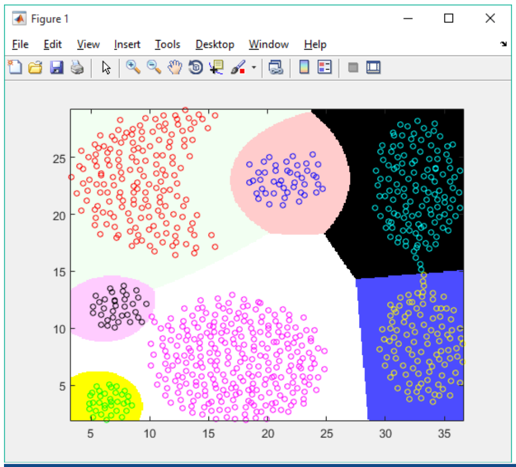
```
function decbound2D(X, yPredict, Model)

% X is data training
% Y is class label of Naive Bayes predicted.
% classifier is an abstract of model of your classifier. You may extend it
% to more than one variable following your classifier design. For example,
% when you use Naive Bayes classifier then you may replace 'classifier'
% above by two variables, namely 'prior' and 'likelihood'

% Original author: Peter Yu, http://www.peteryu.ca/tutorials/matlab/visualize_decision_boundaries
% modified by milo.


% set up the domain over which you want to visualize the decision
% boundary
X1 = X(:,1); X2 = X(:,2);
xrange = [min(X1) max(X1)];
yrange = [min(X2) max(X2)];

% interval how finely you want to visualize the decision boundary.
interval = 0.1;

% generate grid coordinates for the basis of decision boundary visualization.
[x, y] = meshgrid(xrange(1):interval:xrange(2), yrange(1):interval:yrange(2));

% size of decision boundary image for background of plot.
image_size = size(x);

% make (x,y) pairs as a new data point to be classified which 1st column as
% X1, 2nd column as X2
xy = [x(:) y(:)];

numxypairs = length(xy); % number of (x,y) pairs

% loop through each meshgrid points and get the predicted class
class_prediction = zeros(numxypairs,1);
for ii=1:numxypairs,

    % classifiy each new data point xy, put here your code of classification
    % process using your classifier model. The input is each row of 'xy'.
    % Save the predicted class in 'class_prediction'.
    % for example:
    % when ii=1, then the input for your classifier is xy(i,:)
    % the output is saved on class_prediction(ii)

    class_prediction(ii) = GaussianNaiveBayesPredict(xy(ii,:) , Model);

end

% reshape the idx (which contains the class label) into an image.
decisionmap = reshape(class_prediction, image_size);
```

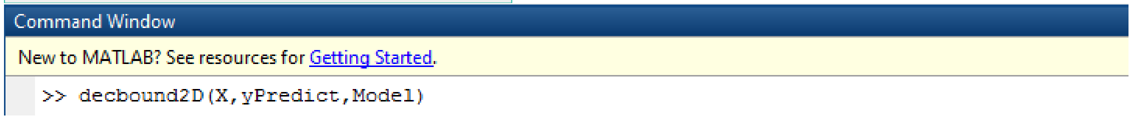
Satrya Budi Pratama
1301154428



```
figure;

%show the image
imagesc(xrange,yrange,decisionmap);
hold on;
set(gca,'ydir','normal');

% set RGB color for colormap for the classes:
cmap = [1 0.8 0.8;       % class 1 = light red
        0.95 1 0.95;     % class 2 = light green
        0.3 0.3 1;       % class3 = blue
        1 1 1;           % class 4 = white
        1 1 0;           % class 5 = yellow
        0 0 0;           % class 6 = black
        1 0.8 1          % class 7 = pink
        ];
colormap(cmap);

 visualize(X,yPredict);
 hold off;
end
```

Output :





Command Window

New to MATLAB? See resources for Getting Started.

```
>> decbound2D(X,yPredict,Model)
```

Analisis Program :

Fungsi decbound2d(param1,param2,param3) akan menampilkan decision boundaries dengan param1 adalah data testing , param2 adalah class label hasil dari naïve bayes, dan param3 adalah model yang akan digunakan untuk memberi label kelas pada new data point yang dipetakan pada imagesec.