```
clear all
rng("default")

%loadind data
TestData = csvread('.\Test1.csv', 1);
TrainData = csvread('.\Train1.csv', 1);


% feature selection using Chi2
[idx, score] = fscchi2(TrainData(:,1:9),TrainData(:,10 ));
TrainData = TrainData(:,[1,5,2,8,10]);
TestData = TestData(:,[1,5,2,8,10]);


% Hyperparametes list
dis_list = {'Normal','kernel'};
prior = {'empirical','uniform'};


% K-fold Cross validation
cv = cvpartition(size(TrainData, 1),"KFold",10);


% Gridseach and K-fold CV
hp_perf = [];
cv_results =[];

for i = 1:length(dis_list)
    for j = 1:length(prior)
        for k = 1:10
       % spliting train and validation set in each iteration
        idx_Train = training(cv,k);
        TrainData_Kfold =TrainData(idx_Train,:);
        idx_val = test(cv,k);
        ValData_kfold =TrainData(idx_val,:);
        X = TrainData_Kfold(:,1:4);
        y = TrainData_Kfold(:,5);

        NBmodel = fitcnb(X,y,'DistributionNames',char(dis_list(i)),'Prior', ...
            char(prior(j)));
        predictions = predict(NBmodel,ValData_kfold(:,1:4));
        %predictions = str2num(cell2mat(predictions));
        iscorrect = predictions == ValData_kfold(:,5);
        correctrate = sum(iscorrect)/numel(predictions);
        cv_results(k) = correctrate;
        avg_cvresults = mean(cv_results); %average Accrurcy of K number of Models
        hp_perf(i,j) = avg_cvresults;


        end
```

```
    end

end


% choosing the best set of Hyper parameters

[MaxAccuracy,I] = max(hp_perf(:));
[I_row, I_col] = ind2sub(size(hp_perf),I); %I_row is the row index and I_col is the
column index
best_distribution = dis_list(I_row);
best_prior = prior(I_col);



% traning the model with the best set of Hyperparmeters on the entire size
% of trainning set (without splitting the validation set)

tic
NBmodel_final = fitcnb(TrainData(:,1:4),TrainData(:,5), ...
    "DistributionNames",char(best_distribution), ...
    'Prior',char(best_prior));

final_model_traintime =toc;


%evaluating the model on test set
tic
predictions = predict(NBmodel_final,TestData(:,1:4));
%predictions = str2num(cell2mat(predictions));
predict_time = toc
% Print the elapsed time
fprintf('Elapsed time: %f seconds\n', predict_time);

%Accrurcy
iscorrect = predictions == TestData(:,5);
Test_accuracy = sum(iscorrect)/numel(predictions);



% Generate the confusion matrix
cm = confusionmat(TestData(:,5),predictions);

% Extract the values from the confusion matrix
TP = cm(1,1);
TN = cm(2,2);
FP = cm(2,1);
FN = cm(1,2);

% Calculate precision and recall
```

```
precision = TP / (TP + FP);
recall = TP / (TP + FN);
f1 = 2 * precision * recall / (precision + recall);
%plot confusion matrix
confusionchart(TestData(:,end),predictions,"Normalization","absolute")
confusionchart(TestData(:,end),predictions,"Normalization","row-normalized")
confusionchart(TestData(:,end),predictions,"Normalization","column-normalized")

% Get the predicted probabilities for the test set
[predictions, scores] = predict(NBmodel_final,TestData(:,1:end-1));
% Convert the predicted labels to a binary vector
%predictions = str2num(cell2mat(predictions));
% Compute the ROC curve
[fpr, tpr, thr] = perfcurve(TestData(:,end), scores(:,2), 1);
% Compute the AUC value
auc = trapz(fpr,tpr);
% Plot the ROC curve
figure;
plot(fpr,tpr);
xlabel('False Positive Rate');
ylabel('True Positive Rate');
title(sprintf('ROC curve (AUC = %0.2f)', auc));
```