SVM MODEL TRAINING (STEP 1)

```
clear; clc; close all
```

load TrainING data

DATA FORMAT:

- exp_dat_raw : numeric (m x n)
- class_label : label (1 xn)
- gene_names : variable name (m x 1)

```
no_dat_set = 1;

switch no_dat_set
    case 1 % (prostate PDE samples)
        filename=strcat('Explant_ResponseGroup_Allprot_DEGs.xlsx');
        tbl_PDE_dat=readtable(filename,...
            'Sheet','DEG expression');
        tbl_PDE_label=readtable(filename,...
            'Sheet','Target labels');
        tbl_PDE_dat.GeneName(ismember(tbl_PDE_dat.GeneName,{'SEPTIN8'}))= {'SEPT8'};

        exp_dat_raw=table2array(tbl_PDE_dat(:,2:end));
        class_label = table2array(tbl_PDE_label);
        gene_names = tbl_PDE_dat.GeneName;
        model_name = 'deg157';

    case 2 % (maker gene only)

        filename=strcat('Explant_ResponseGroup_Allprot_DEGs.xlsx');
        tbl_PDE_dat=readtable(filename,...
            'Sheet','DEG expression');
        tbl_PDE_label=readtable(filename,...
            'Sheet','Target labels');
        tbl_PDE_dat.GeneName(ismember(tbl_PDE_dat.GeneName,{'SEPTIN8'}))= {'SEPT8'};


        filename=strcat('training_outcome_dat.xlsx');
        tbl_markers=readtable(filename,...
            'Sheet','High Ranked IS');
        tbl_PDE_dat(~ismember(tbl_PDE_dat.GeneName,tbl_markers.tmp_xlabs),:)=[];

        exp_dat_raw=table2array(tbl_PDE_dat(:,2:end));
        class_label = table2array(tbl_PDE_label);
        gene_names = tbl_PDE_dat.GeneName;
        model_name = '';

    case 8 % (specific marker genes)

        marker_genes = {'TACC1','CDK2','AQP1','FDFT1','RBM17','TRIM47','VPS25'};
```

```
        filename=strcat('Explant_ResponseGroup_Allprot_DEGs.xlsx');
        tbl_PDE_dat=readtable(filename,...
            'Sheet','DEG expression');
        tbl_PDE_label=readtable(filename,...
            'Sheet','Target labels');
        tbl_PDE_dat.GeneName(ismember(tbl_PDE_dat.GeneName,{'SEPTIN8'}))= {'SEPT8'};
        tbl_PDE_dat(~ismember(tbl_PDE_dat.GeneName,marker_genes),:)= [];

        exp_dat_raw=table2array(tbl_PDE_dat(:,2:end));
        class_label = table2array(tbl_PDE_label);
        gene_names = tbl_PDE_dat.GeneName;
        model_name = '';


        %  set-up input data for plotting
        col_labels = class_label;
        % Class label of samples
        row_labels = gene_names;
        % gene names
        dat_exp = log10((exp_dat_raw));
        % expression data (row: genes, col: samples)


        hmo = clustergram(dat_exp,...
            'Cluster','column',...
            'Standardize',2,...
            'RowLabels',row_labels,...
            'ColumnLabels',col_labels,...
            'ColumnLabelsRotate',45);


end
```

Training data normalization between 0 and 1

```
tmp_min = repmat(min(exp_dat_raw')',1,size(exp_dat_raw,2));
tmp_max = repmat(max(exp_dat_raw')',1,size(exp_dat_raw,2));
predictor = (exp_dat_raw -  tmp_min)./(tmp_max - tmp_min);


% figure('position',[296 786 1305 192])
% boxplot(predictor','Notch','on')
% box off
% set(gca,'xticklabel',{})
% xlabel(strcat('Differentially Expressed Genes(N=',num2str(size(predictor,1)),')'))
% ylabel('Normalized')
% xticklabels(gene_names);
% xtickangle(45)

% % save the data for prism
% dat_NR = predictor(:,ismember(table2array(tbl_target_class),{'NR'}));
% prism_NR_dat  = [mean(dat_NR,2) std(dat_NR,0,2) size(dat_NR,2)*ones(size(dat_NR,1),1)];
%
```

```
% dat_RD = predictor(:,ismember(table2array(tbl_target_class),{'RD'}));
% prism_RD_dat  = [mean(dat_RD,2) std(dat_RD,0,2) size(dat_RD,2)*ones(size(dat_RD,1),1)];
%
% dat_PR = predictor(:,ismember(table2array(tbl_target_class),{'PR'}));
% prism_PR_dat  = [mean(dat_PR,2) std(dat_PR,0,2) size(dat_PR,2)*ones(size(dat_PR,1),1)];
%
% dat_gene_sym = tbl_exp_dat.GeneName(marker_idx);
```

RUN THE SIMULATION

- 50 cross-validation
- predictor : data used for the model training (train + text)
- predictor_train : training data
- predictor_test : test data
- class_label_train : label for traning data
- class_label_true : lavel for test data

```
no_cross_val = 50;
no_samples = size(predictor,2);
```

Rrandom sampling for train (80%) and test (20%)

```
for ii = 1:no_cross_val
    rng(ii); % random number seed for reproducibility
    [train_sample_idx(ii,:),~,test_sample_idx(ii,:)] = dividerand(no_samples,0.8,0.0,0.2);
end
```

training the models

```
parfor  idx1 = 1:no_cross_val

    %disp(idx1)

    % copy variables for parfor
    predictor_par = predictor;
    class_label_par = class_label;
    train_sample_idx_par = train_sample_idx;
    test_sample_idx_par = test_sample_idx;

    % random sampling for training an test
    train_idx = train_sample_idx_par(idx1,:);
    test_idx = test_sample_idx_par(idx1,:);

    % Train & Test data
    predictor_train = predictor_par(:,train_idx); % training input
    predictor_test = predictor_par(:,test_idx); % test input
```

```
    class_label_train = class_label_par(train_idx);
    class_label_true_compiled(:,idx1) = class_label_par(test_idx);

    rng(idx1);

    % SVM options
    svmtemp=templateSVM('Standardize',1,...
        'KernelFunction','linear');
    Mdl = fitcecoc(predictor_train',class_label_train',...
        'Learners',svmtemp,...
        'ClassNames',{'NR', 'RD','PR'});
    class_label_pred_compiled(:,idx1) = predict(Mdl,predictor_test');

    % to plot the ROC curve
    Mdl2 = fitcecoc(predictor_train',class_label_train',...
        'Learners',svmtemp,...
        'ClassNames',{'NR', 'RD','PR'},...
        'FitPosterior',true);
    [class_label_pred_compiled2(:,idx1),~,~,cnum2(:,:,idx1)]=predict(Mdl2,predictor_test');

    % save the Models for validation
    trained_Mdl{idx1} = Mdl;
end

fpath = strcat(pwd,'\Trained_models\Mdl_',model_name,'.mat');
save(fpath,'trained_Mdl');
```

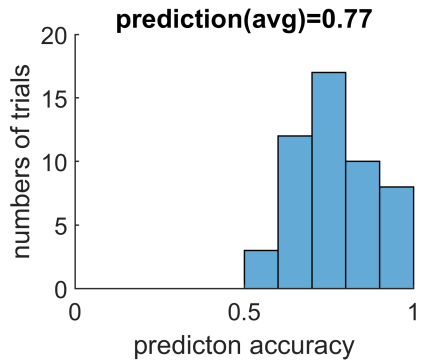
Post-processing

```
for idx1 = 1:no_cross_val

    [cmat,~] = confusionmat(class_label_true_compiled(:,idx1),class_label_pred_compiled(:,idx1)
    score_mat(idx1)= sum(diag(cmat)/length(class_label_true_compiled(:,idx1)));

end

% averaged prediction accuracy
avg_prediction_acc = mean(score_mat);
disp(strcat('Prediction = ',num2str(avg_prediction_acc)))
```

```
Prediction =0.77
```

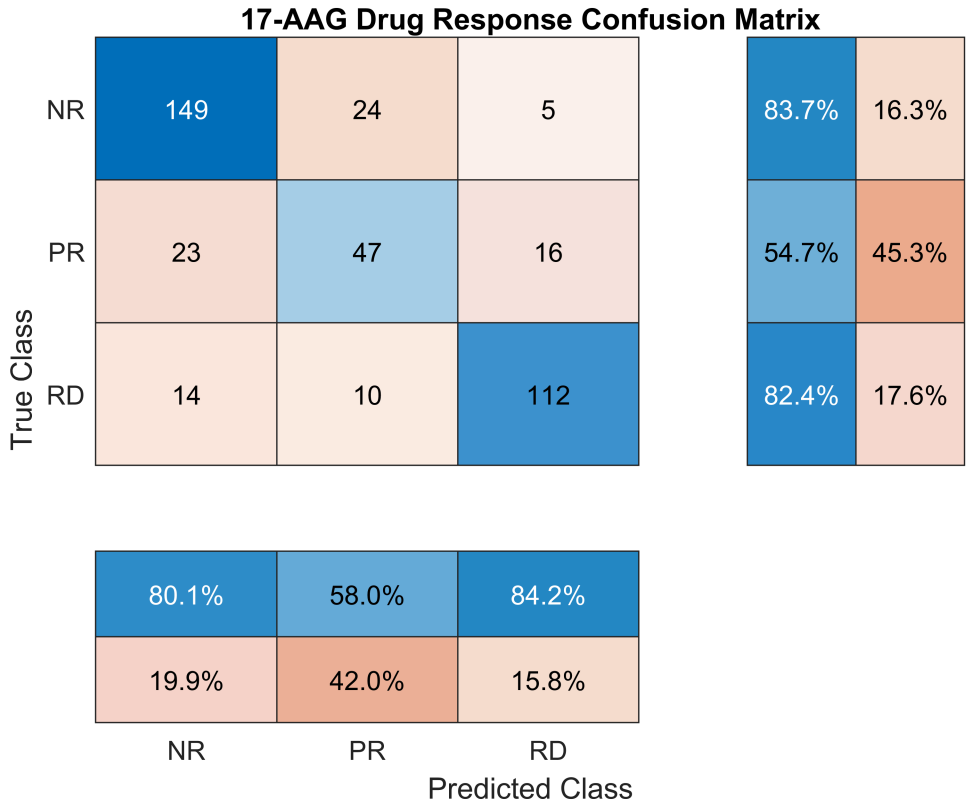
Plot a distribution of curve for prediction accuracy

```
figure('Position',[680   796   308   182])
edges = [0:0.1:1];
histogram(score_mat,edges)
title(strcat('prediction(avg)=',num2str(mean(score_mat))))
xlabel('predicton accuracy'),ylabel('numbers of trials')
pbaspect([4 3 1]/4)
box off
```



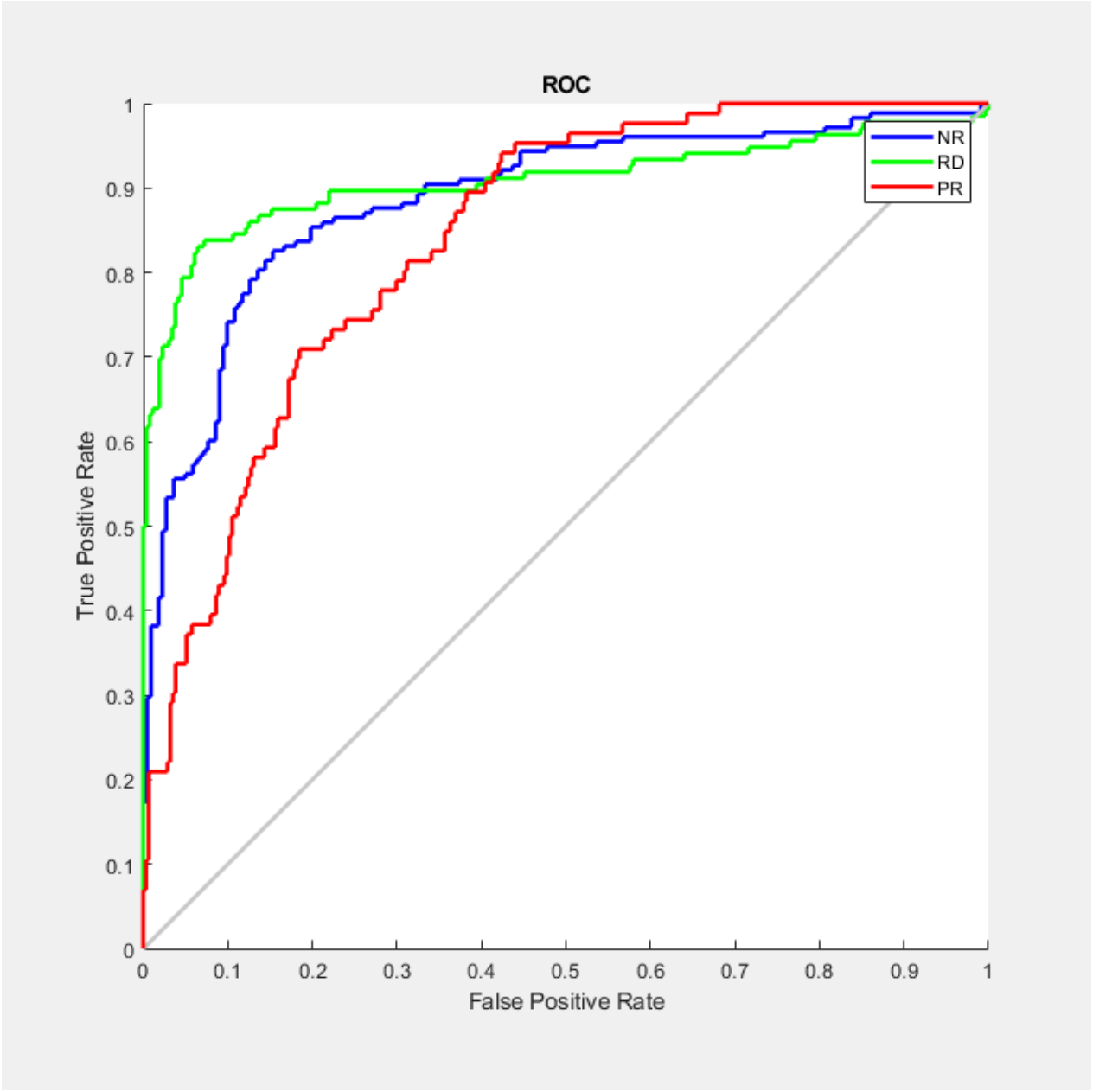
confusion matrix

```
[cfm,order] = confusionmat(class_label_true_compiled(:),class_label_pred_compiled(:));
figure
cm = confusionchart(class_label_true_compiled(:),class_label_pred_compiled(:), ...
    'ColumnSummary','column-normalized', ...
    'RowSummary','row-normalized');
cm.Title = '17-AAG Drug Response Confusion Matrix';
```



ROC curves

```
class_label_true_binary = zeros(3,size(class_label_true_compiled(:),1));
class_label_true_binary(1,ismember(class_label_true_compiled(:),{'NR'})) = 1;
class_label_true_binary(2,ismember(class_label_true_compiled(:),{'RD'})) = 1;
class_label_true_binary(3,ismember(class_label_true_compiled(:),{'PR'})) = 1;

class_label_pred_probability = [];

for ii = 1:size(cnum2,3)
    class_label_pred_probability = [class_label_pred_probability;cnum2(:,:,ii)];
end

[tpr,fpr,thresholds] = roc(class_label_true_binary,class_label_pred_probability');
figure('position',[610   613   337   277])
plotroc(class_label_true_binary,class_label_pred_probability')
axesUserData = get(gca, 'userdata');
legend(axesUserData.lines, 'NR', 'RD','PR');
```


ROC
NR
RD
PR
True Positive Rate
False Positive Rate
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1