Sergio Munguia

MatLab Assignment 2 Report

K Fold – Cross validation is an honest assessment of the true accuracy.
In cross validation, we divide our data into a large training set and smaller validation set, then train on the training set and use the validation set to measure our accuracy.

Questions for additional reflection:

1. What happens to some key values (and decisions made upon those values), if you change line 13 in the code (settings for the random number generator)?

The misclassification error changes & It is better to randomly select validation examples, rather than go on a set of examples specifically for validation, because you want the validation set to bediverse.

2. *What are the main differences between "feature selection by filtering" and "sequential filter selection"? Why was the former used as a preprocessing step for the latter?*

For random feature variations , the sequential filtering was selected because it sets up between the boundaries for the features.

3. *Why did we not use manual / interactive feature selection (using the Classification Learner App) in this case (as we did for the Fisher Iris problem, for instance)?*

You can use Classification Learner to train models to classify data. Using this app, you can explore supervised machine learning using various classifiers. You can explore your data, select features, specify validation schemes, train models

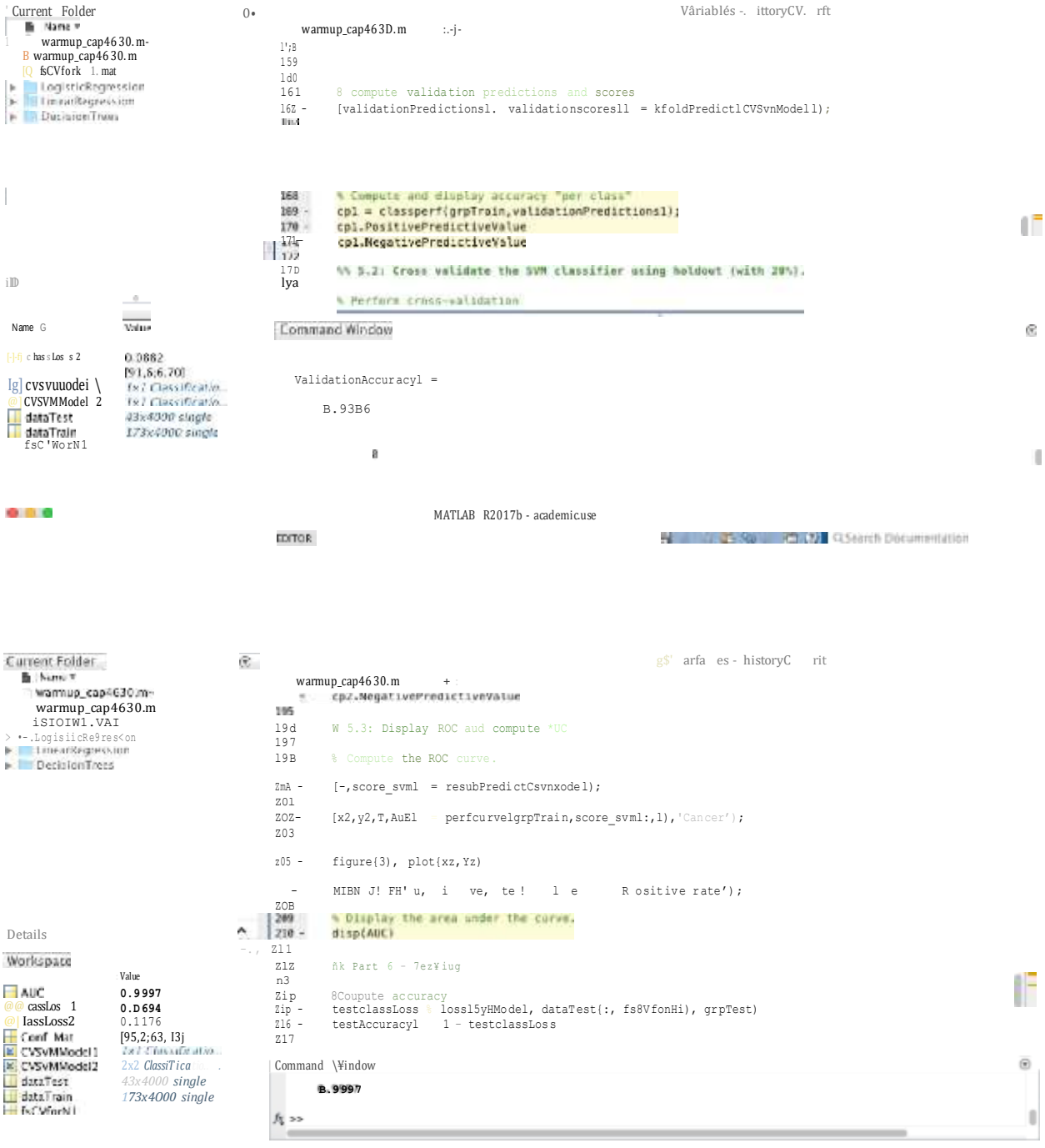
4. *Why is the accuracy "per class" (computed in part 5.1) relevant?*

The mean of data accuracy per model can be different. It shows the difference between the models of K- Fold Cross validation and SVM is fairly.

5. *The test results for SVM were basically perfect. Can you trust them completely? Why (not)?*

No because of the boundaries that we set on the decision line, it's to make it generalized, and it always is still an approximate and there is always a chance for a classifier to be untrustworthy.

In practice, the reason that SVMs tend to be resistant to over-fitting, even in cases where the number of attributes is greater than the number of observations, is that it uses regularization. The SVM is an approximate implementation of a bound on the generalization error, that depends on the margin (essentially the distance from the decision boundary to the nearest pattern from each class).


Current Folder
warmup_cap4630.m
warmup_cap4630.m
fsCVfork 1.mat
LogisticRegression
LinearRegression
DecisionTrees
warmup_cap463D.m
% Compute validation predictions and scores
[validationPredictions1, validationScores11 = kfoldPredict1CVSvnModel1);
% Compute and display accuracy "per class"
cp1 = classperf(grpTrain,validationPredictions1);
cp1.PositivePredictiveValue
cp1.NegativePredictiveValue
%% 5.2: Cross validate the SVM classifier using holdout (with 20%).
% Perform cross-validation
Command Window
ValidationAccuracy1 =
B.93B6
Name G
Value
0.0882
[91,6,6,70]
CVSVMModel 2
dataTest
dataTrain
43x4000 single
173x4000 single
MATLAB R2017b - academicuse
EDITOR
Search Documentation
Current Folder
warmup_cap4630.m
warmup_cap4630.m
iSIOIW1.VAI
DecisionTrees
warmup_cap4630.m
cpz.NegativePredictiveValue
W 5.3: Display ROC aud compute *UC
% Compute the ROC curve.
[-,score_svm1 = resubPredictCsvnxodel);
[x2,y2,T,AuEl perfcurvelgrpTrain,score_svm1:,1),'Cancer');
figure(3), plot(xz,Yz)
MIBN J! FH' u, i ve, te ! l e R ositive rate');
% Display the area under the curve.
disp(AUC)
ñk Part 6 - 7ez¥iug
8Coupute accuracy
testclassLoss % lossl5yHModel, dataTest(:, fs8VfonHi), grpTest)
testAccuracy1 1 - testclassLoss
Details
Workspace
Value
AUC
0.9997
cassLos 1
0.D694
IassLoss2
0.1176
Conf Mat
[95,2;63, I3j
CVSvMModel1
CVSvMModel2
2x2 ClassTica
dataTest
43x4000 single
dataTrain
173x4000 single
Command \¥indow
B.9997
>>

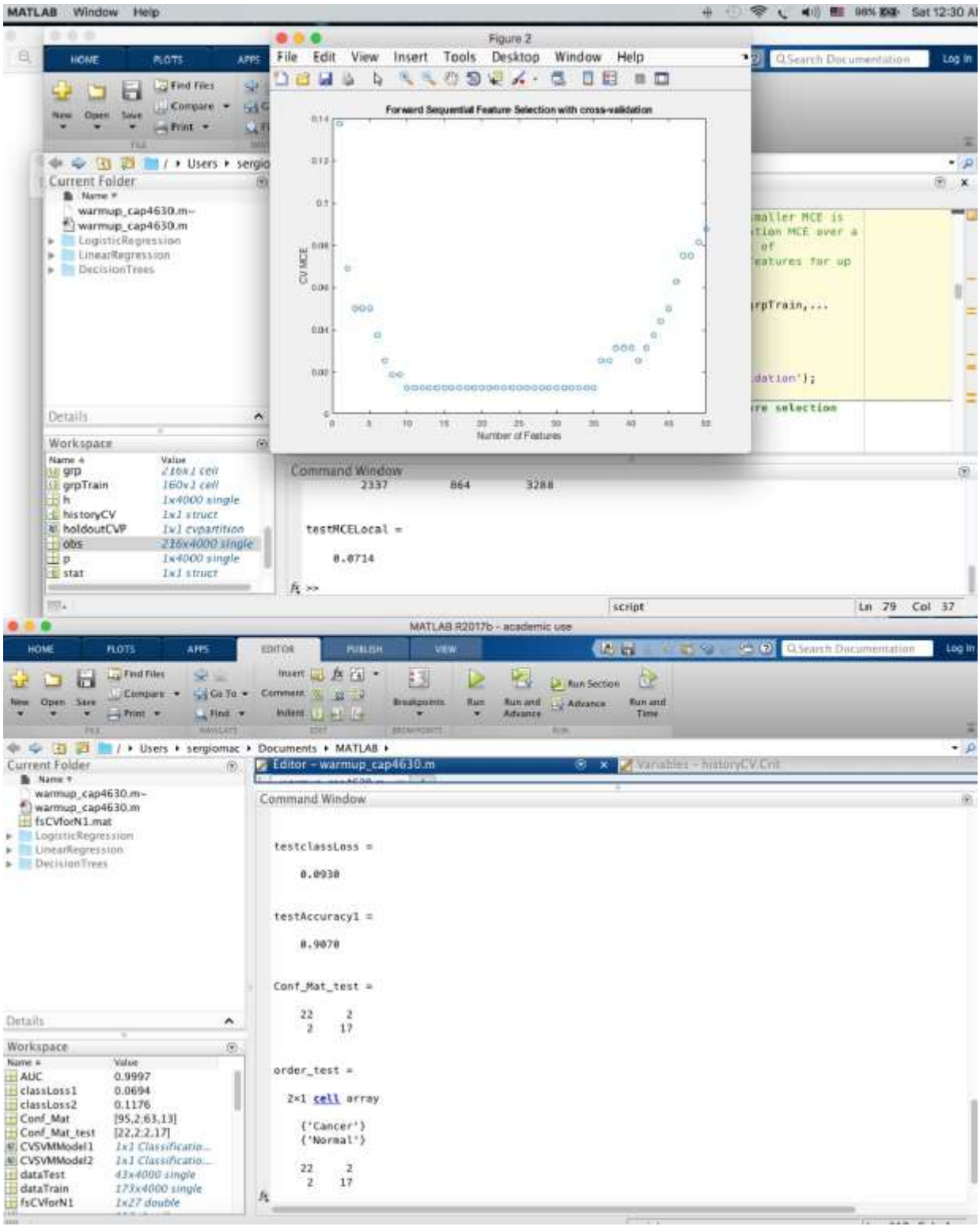


Figure 2
File Edit View Insert Tools Desktop Window Help
Forward Sequential Feature Selection with cross-validation
CV MCE
Number of Features
Command Window
testMCELocal =
0.0714
Command Window
testclassLoss =
0.0930
testAccuracy1 =
0.9070
Conf_Mat_test =
22 2
2 17
order_test =
2x1 cell array
{'Cancer'}
{'Normal'}
22 2
2 17

```
%% CAP 4630 - Intro to AI - FAU - Dr. Marques - Fall 2016
%% Final Project - Warmup exercise (2-class classifier)
%% Part 1 - Loading the data

load ovariancancer;

%% Part 2 - Feature selection
%% 2.1: Prepare the data

% Set the random number generator to a known state.
% Otherwise, your results may differ.
rng(2000,'twister');

% Partitioning the dataset (for feature selection): 160 points for the
% training set and the remaining 56 for testing.
holdoutCVP = cvpartition(grp,'holdout',56);
dataTrain = obs(holdoutCVP.training,:);
grpTrain = grp(holdoutCVP.training);

%% 2.2: Feature selection (step 1): using a simple filter approach

% Filters are usually used as a pre-processing step in feature selection,
% due to their simplicity and speed.
% A widely-used filter method for bioinformatics data is to apply
% a statistical test separately on each feature, assuming that there is
% no interaction between features.

% For example, we might apply the _t_-test on each feature and compare
% _p_-value (or the absolute values of _t_-statistics) for each
% feature as a measure of how effective it is at separating groups.
dataTrainG1 = dataTrain(grp2idx(grpTrain)==1,:);
dataTrainG2 = dataTrain(grp2idx(grpTrain)==2,:);
[h,p,ci,stat] = ttest2(dataTrainG1,dataTrainG2,'Vartype','unequal');

% In order to get a general idea of how well-separated the two groups are
% by each feature, we plot the empirical cumulative distribution function
% (CDF) of the _p_-values:
figure(1), ecdf(p);
xlabel('P value');
ylabel('CDF value')

%% 2.3: Feature selection (step 2): using sequential feature selection

% Use the filter results from the previous section as a
% pre-processing step to select features: sort the features according
% to their p values and select the top 150 features.
[~,featureIdxSortbyP] = sort(p,2);
fs1 = featureIdxSortbyP(1:150);

% Generate a stratified 10-fold partition for the training set:
tenfoldCVP = cvpartition(grpTrain,'kfold',10);

% Apply forward sequential feature selection on these 150 features.
% The function |sequentialfs| provides a simple way (the default option) to
% decide how many features are needed. It stops when the first local
% minimum of the cross-validation MCE (misclassification error) is found.
fun = @(xtrain,ytrain,xtest,ytest) ...
              sum(~strcmp(ytest,classify(xtest,xtrain,ytrain,'quadratic')));
fsLocal = sequentialfs(fun,dataTrain(:,fs1),grpTrain,'cv',tenfoldCVP);

% The selected features are the following:
```

```
fs1(fsLocal)

% To evaluate the performance of the selected model with these four features,
% we compute the MCE on the 56 test samples.
testMCELocal = crossval(fun,obs(:,fs1(fsLocal)),grp,'partition',...
    holdoutCVP)/holdoutCVP.TestSize

%% 2.4: Feature selection (step 3): improving sequential feature selection

% The algorithm may have stopped prematurely. Sometimes a smaller MCE is
% achievable by looking for the minimum of the cross-validation MCE over a
% reasonable range of number of features. Let's draw a plot of
% the cross-validation MCE as a function of the number of features for up
% to 50 features.

[fsCVfor50,historyCV] = sequentialfs(fun,dataTrain(:,fs1),grpTrain,...
    'cv',tenfoldCVP,'Nf',50);
figure(2), plot(historyCV.Crit,'o');
xlabel('Number of Features');
ylabel('CV MCE');
title('Forward Sequential Feature Selection with cross-validation');

%% 2.5: Feature selection (step 4): performing actual feature selection

%%%% ENTER THE VALUE OF N1 HERE!!! %%%%
%
N1 = 27
%
  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

 fsCVforN1  =  fs1(historyCV.In(N1,:));

% Save selected features for later use
save('fsCVforN1.mat', 'fsCVforN1');

% To show these N1 features in the order in which they are selected in the
% sequential forward procedure, we find the row in which they first become
% true in the |historyCV| output:
       [orderlist,ignore] = find( [historyCV.In(1,:); diff(historyCV.In(1:N1,:) )]' );

       fs1(orderlist);

       % To evaluate these N1 features, we compute their MCE for QDA on the test
       % set. We get the smallest MCE value so far:
       testMCECVforN1 = crossval(fun,obs(:,fsCVforN1),grp,'partition',...
           holdoutCVP)/holdoutCVP.TestSize

       %% Part 3 - Starting fresh (with only the selected features)

       close all; clear all; clc

       load ovariancancer;
       load fsCVforN1;

       %% 3.1: Partition dataset into 3 groups

       % 80% for training and cross validation
       % 20% for testing

       %%%% ENTER YOUR CODE HERE!!! %%%%
       holdoutCVP = cvpartition(grp,'holdout',43);
```

```
122.        dataTrain = obs(holdoutCVP.training,:);
123.        grpTrain = grp(holdoutCVP.training,:);
124.        dataTest = obs(holdoutCVP.test,:);
125.        grpTest = grp(holdoutCVP.test,:);
126.        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
127.
128.        %% Part 4 - Building a model (SVM)
129.
130.        X = dataTrain(:,fsCVforN1);
131.        Y = grpTrain;
132.
133.        % Train an SVM classifier using the radial basis kernel.  Let the software
134.        % find a scale value for the kernel function.  It is good
135.        % practice to standardize the predictors.
136.
137.        SVMModel = fitcsvm(X,Y,'Standardize',true,'KernelFunction','RBF',...
138.            'KernelScale','auto', 'ClassNames', {'Cancer','Normal'});
139.
140.        %% Part 5 - Evaluating the model
141.
142.        %% 5.1: Cross validate the SVM classifier using 10-fold cross validation.
143.
144.        % Perform cross-validation
145.        %%%%% ENTER YOUR CODE HERE!!! %%%%
146.        CVSVMModel1 = crossval(SVMModel,'kfold',10);
147.        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
148.
149.        % Estimate the out-of-sample misclassification rate.
150.        %%%%% ENTER YOUR CODE HERE!!! %%%%
151.        classLoss1 = kfoldLoss(CVSVMModel1)
152.
153.        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
154.
155.        % Compute validation accuracy
156.        %%%%% ENTER YOUR CODE HERE!!! %%%%
157.        validationAccuracy1 = 1 - classLoss1
158.
159.        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
160.
161.        % Compute validation predictions and scores
162.        [validationPredictions1, validationScores1] = kfoldPredict(CVSVMModel1);
163.
164.        % Display confusion matrix
165.        [Conf_Mat,order] = confusionmat(grpTrain,validationPredictions1);
166.        disp(Conf_Mat)
167.
168.        % Compute and display accuracy "per class"
169.        cp1 = classperf(grpTrain,validationPredictions1);
170.        cp1.PositivePredictiveValue
171.        cp1.NegativePredictiveValue
172.
173.        %% 5.2: Cross validate the SVM classifier using holdout (with 20%).
174.
175.        % Perform cross-validation
176.        CVSVMModel2 = crossval(SVMModel,'holdout',0.2);
177.
178.        % Estimate the out-of-sample misclassification rate.
179.        classLoss2 = kfoldLoss(CVSVMModel2)
180.
181.        % Compute validation accuracy
182.        validationAccuracy2 = 1 - classLoss2
```

```
183.
184.        % Compute validation predictions and scores
185.        [validationPredictions2, validationScores2] = kfoldPredict(CVSVMModel2);
186.
187.        % Display confusion matrix
188.        [Conf_Mat,order] = confusionmat(grpTrain,validationPredictions2);
189.        disp(Conf_Mat)
190.
191.        % Compute and display accuracy "per class"
192.        cp2 = classperf(grpTrain,validationPredictions1);
193.        cp2.PositivePredictiveValue
194.        cp2.NegativePredictiveValue
195.
196.        %% 5.3: Display ROC and compute AUC
197.
198.        % Compute the ROC curve.
199.        SVMModel = fitPosterior(SVMModel);
200.        [~,score_svm] = resubPredict(SVMModel);
201.
202.        [X2,Y2,T,AUC] = perfcurve(grpTrain,score_svm(:,1),'Cancer');
203.
204.        % Plot the ROC curve
205.        figure(3), plot(X2,Y2)
206.        xlabel('False positive rate'); ylabel('True positive rate');
207.        title('ROC Curves for SVM, Training dataset')
208.
209.        % Display the area under the curve.
210.        disp(AUC)
211.
212.        %% Part 6 - Testing
213.
214.        %Compute accuracy
215.        testclassLoss = loss(SVMModel, dataTest(:,fsCVforN1), grpTest)
216.        testAccuracy1 = 1 - testclassLoss
217.
218.        % Label the test sample observations.
219.        % Display the results for the observations in the test sample.
220.        [label_test,score_test] = predict(SVMModel,dataTest(:,fsCVforN1));
221.
222.        % 3.2: Display confusion matrix
223.        % PASTE YOUR CODE HERE!
224.        [Conf_Mat_test,order_test] = confusionmat(grpTest,label_test)
225.        disp(Conf_Mat_test)
226.
227.        % Compute and display accuracy "per class"
228.        cp3 = classperf(grpTest,label_test);
229.        cp3.PositivePredictiveValue
230.        cp3.NegativePredictiveValue
231.
232.        % Display ROC and compute AUC
233.        % Compute the ROC curve.
234.        [X_test,Y_test,T_test,AUC_test] = perfcurve(grpTest,score_test(:,1),'Cancer');
235.
236.        % Plot the ROC curve
237.        figure(4), plot(X_test,Y_test)
238.        xlabel('False positive rate')
239.        ylabel('True positive rate')
240.        title('ROC for Classification by SVM, Test Data Set')
241.
242.        disp(AUC_test)
```