Sergio Munguia

MatLab Assignment 2 Report

We take in a data->prepare data and then train the model , select a model - > train the model

- Using known data and known data responses (cancer vs normal)

We then measure the accuracy of the model selected
Then use for prediction the model and new data for predicted responses.
By doing classification we can predict the best group a new observation belongs to.

Supervised learning, There is a tradeoff between the expressiveness of a hypothesis space and the complexity of finding a good hypothesis within that space.

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.

1. Maximize both of the sets. – To get best observation

SVM - A Support Vector Machine (SVM) is a discriminative classifier formally defined by a separating hyperplane. In other words, given labeled training data (supervised learning), the algorithm outputs an optimal hyperplane which categorizes new examples.

The operation of the SVM algorithm is based on finding the hyperplane that gives the largest minimum distance to the training examples. Twice, this distance receives the important name of margin within SVM's theory. Therefore, the optimal separating hyperplane maximizes the margin of the training data.

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 be diverse.

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?*

The goal is to make the size of the dimensions of the data smaller.
The sequential filter selection gives the smaller MCE % on the test set.
When we graph it shows number N features in a line of data plots, excluding the redundant.
Feature selection algorithm does not consider interaction between features; besides, features selected from the list based on their individual ranking may also contain redundant information since the typical goal of classification is to minimize the MCE, the feature selection procedure performs a sequential search using the MCE of the learning algorithm

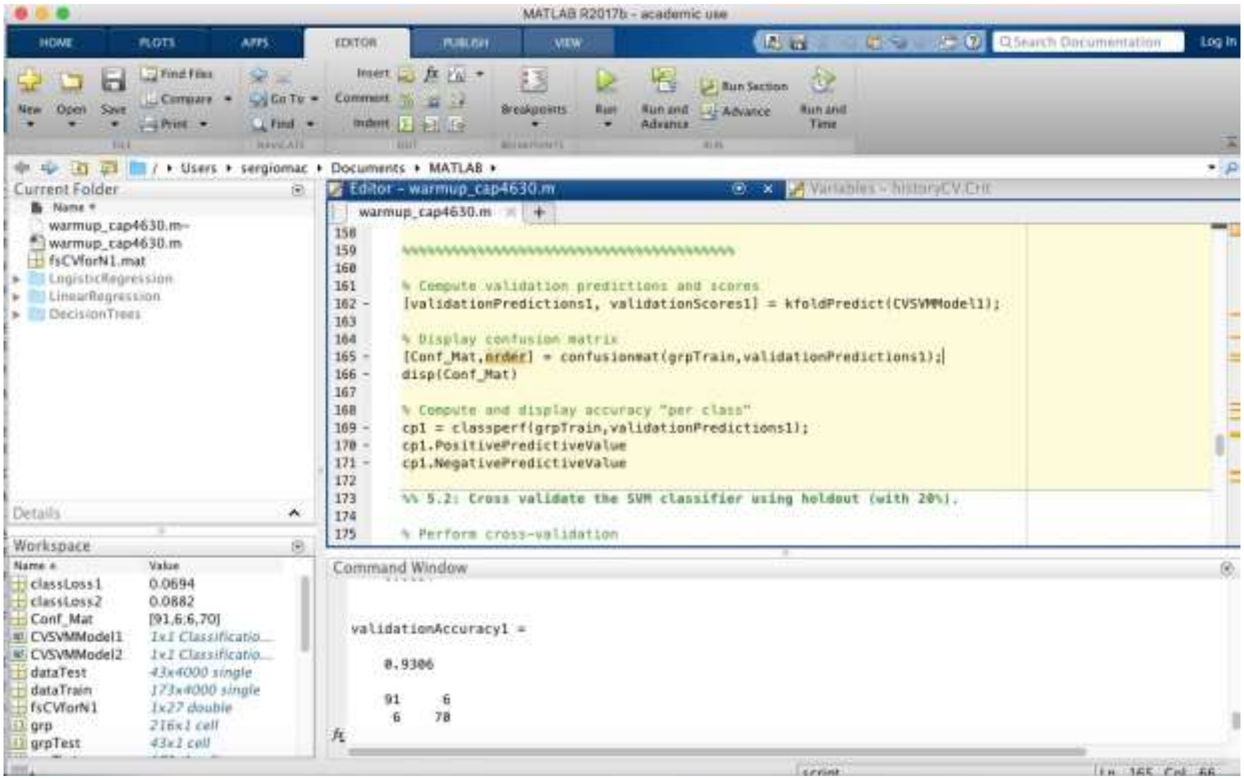
3. *Why did we not use manual / interactive feature selection (using the Classification Learner App) in this case (as we did for the Fisher Iris problem, for instance)?*

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.

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).



```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Compute validation predictions and scores
[validationPredictions1, validationScores1] = kfoldPredict(CVSVMModel1);

% Display confusion matrix
[Conf_Mat,order] = confusionmat(grpTrain,validationPredictions1);
disp(Conf_Mat)

% 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
```

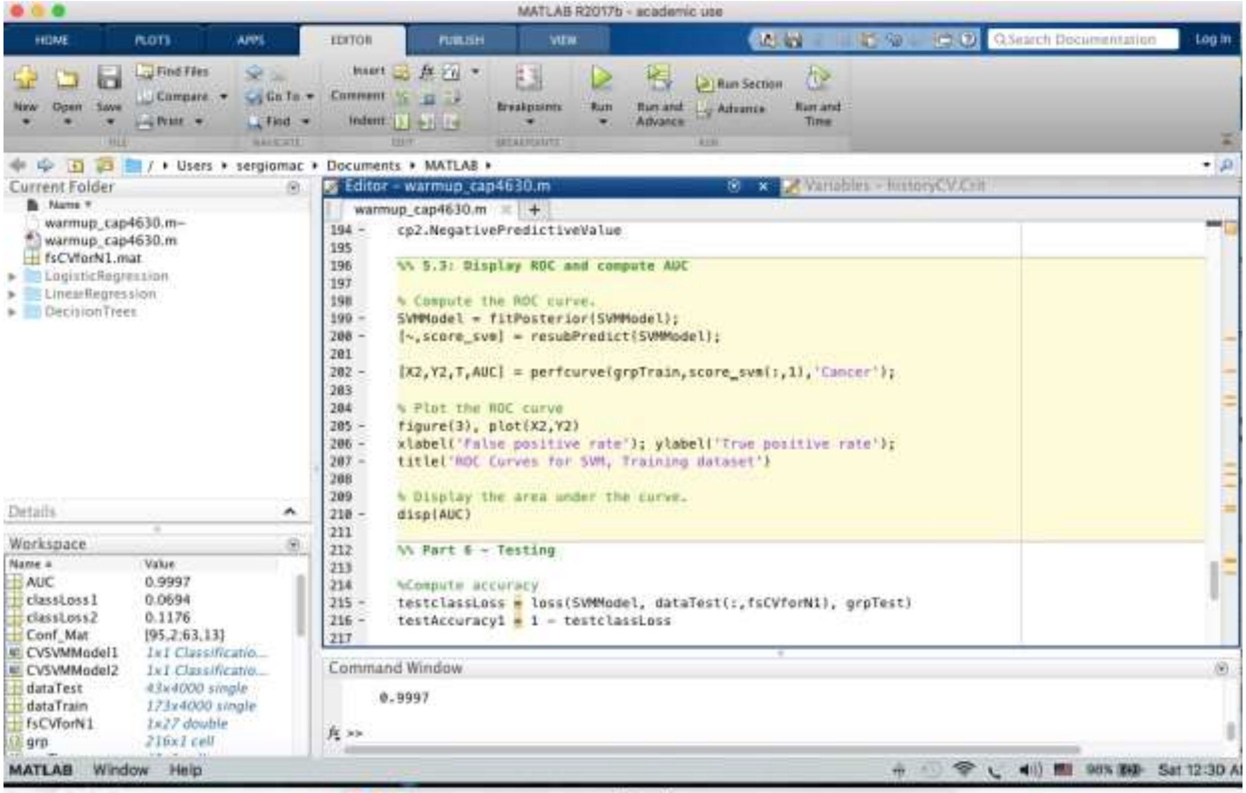
```
validationAccuracy1 =

    0.9306

    91     6
     6    78
```

Workspace

Name	Value
classLoss1	0.0694
classLoss2	0.0882
Conf_Mat	[91,6;6,70]
CVSVMModel1	1x1 Classificatio...
CVSVMModel2	1x1 Classificatio...
dataTest	43x4000 single
dataTrain	173x4000 single
fsCVforN1	1x27 double
grp	216x1 cell
grpTest	43x1 cell



Current Folder:

```
warmup_cap4630.m~
warmup_cap4630.m
fsCVforN1.mat
LogisticRegression
LinearRegression
DecisionTrees
```

```matlab
cp2.NegativePredictiveValue

%% 5.3: Display ROC and compute AUC

% Compute the ROC curve.
SVMModel = fitPosterior(SVMModel);
[~,score_svm] = resubPredict(SVMModel);

[X2,Y2,T,AUC] = perfcurve(grpTrain,score_svm(:,1),'Cancer');

% Plot the ROC curve
figure(3), plot(X2,Y2)
xlabel('False positive rate'); ylabel('True positive rate');
title('ROC Curves for SVM, Training dataset')

% Display the area under the curve.
disp(AUC)

%% Part 6 - Testing

%Compute accuracy
testclassLoss = loss(SVMModel, dataTest(:,fsCVforN1), grpTest)
testAccuracy1 = 1 - testclassLoss

```

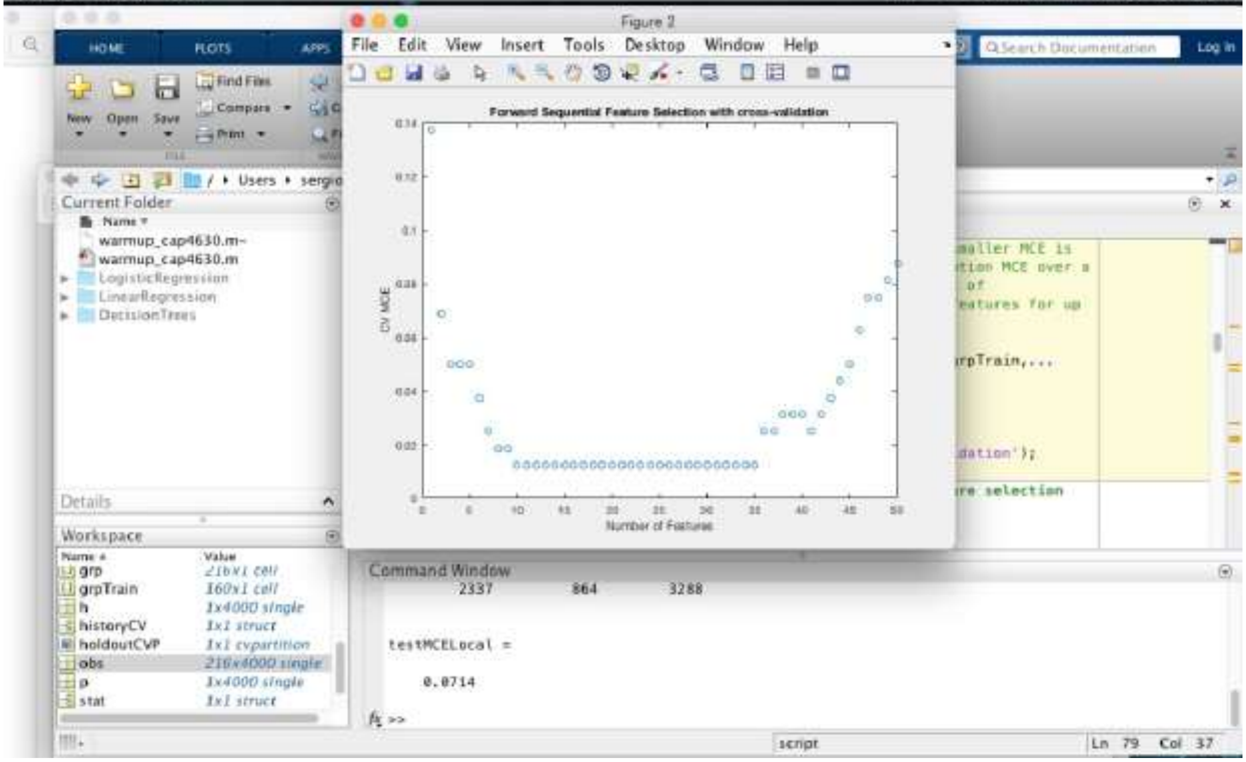
Workspace:

Name ▲	Value
AUC	0.9997
classLoss1	0.0694
classLoss2	0.1176
Conf_Mat	[95,2;63,13]
CVSVMModel1	1x1 Classificatio...
CVSVMModel2	1x1 Classificatio...
dataTest	43x4000 single
dataTrain	173x4000 single
fsCVforN1	1x27 double
grp	216x1 cell

Command Window:

```
    0.9997

fx >>
```



Current Folder:

```
warmup_cap4630.m~
warmup_cap4630.m
LogisticRegression
LinearRegression
DecisionTrees
```

```matlab
maller MCE is
tion MCE over a
 of
eatures for up

rpTrain,...

dation');

re selection
```

Workspace:

Name ▲	Value
grp	216x1 cell
grpTrain	160x1 cell
h	1x4000 single
historyCV	1x1 struct
holdoutCVP	1x1 cvpartition
obs	216x4000 single
p	1x4000 single
stat	1x1 struct

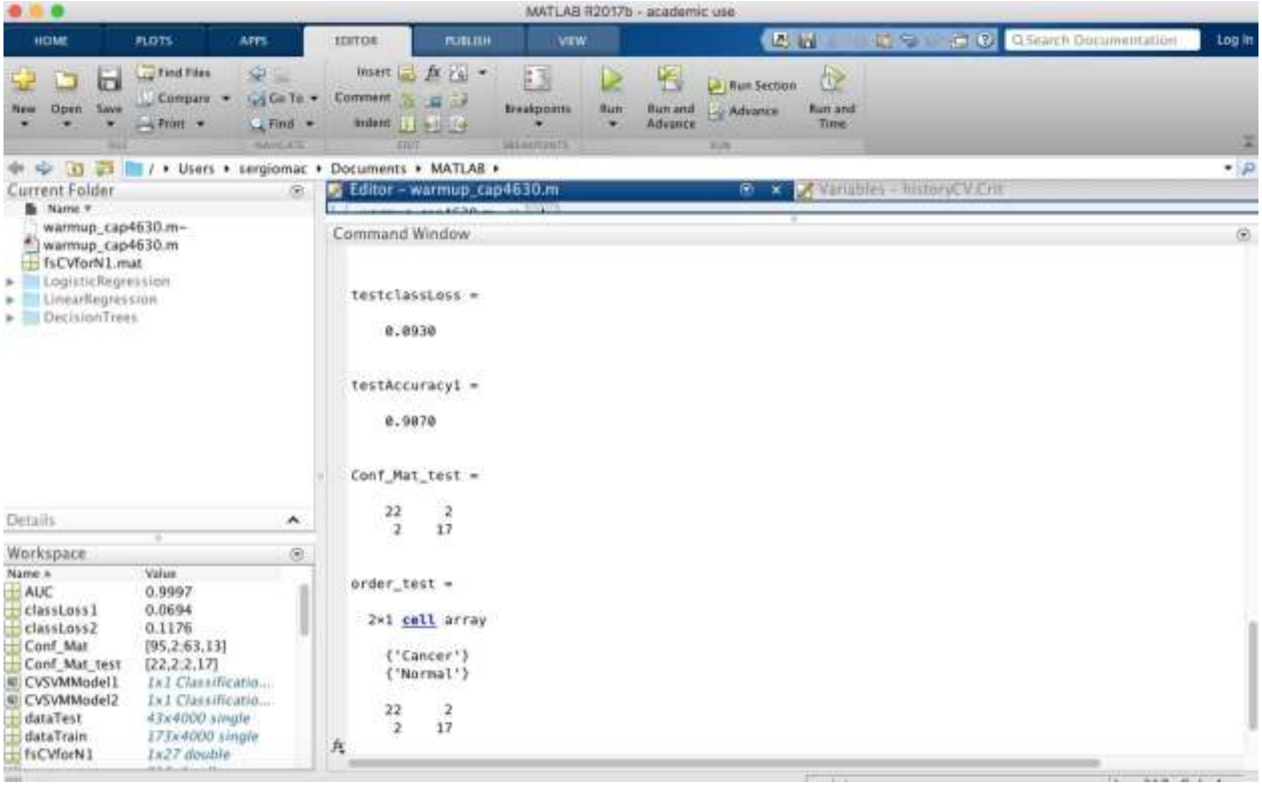
Command Window:

```
        2337        864        3288


testMCELocal =

    0.0714

fx >>
```



```
testclassLoss =

    0.0930


testAccuracy1 =

    0.9070


Conf_Mat_test =

    22     2
     2    17


order_test =

  2×1 cell array

    {'Cancer'}
    {'Normal'}

    22     2
     2    17
```

Name ▴	Value
AUC	0.9997
classLoss1	0.0694
classLoss2	0.1176
Conf_Mat	[95,2;63,13]
Conf_Mat_test	[22,2;2,17]
CVSVMModel1	1x1 Classificatio...
CVSVMModel2	1x1 Classificatio...
dataTest	43x4000 single
dataTrain	173x4000 single
fsCVforN1	1x27 double

```
%% 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);
        dataTrain = obs(holdoutCVP.training,:);
        grpTrain = grp(holdoutCVP.training,:);
        dataTest = obs(holdoutCVP.test,:);
        grpTest = grp(holdoutCVP.test,:);
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

        %% Part 4 - Building a model (SVM)

        X = dataTrain(:,fsCVforN1);
        Y = grpTrain;

        % Train an SVM classifier using the radial basis kernel.  Let the software
        % find a scale value for the kernel function.  It is good
        % practice to standardize the predictors.

        SVMModel = fitcsvm(X,Y,'Standardize',true,'KernelFunction','RBF',...
            'KernelScale','auto', 'ClassNames', {'Cancer','Normal'});

        %% Part 5 - Evaluating the model

        %% 5.1: Cross validate the SVM classifier using 10-fold cross validation.

        % Perform cross-validation
        %%%%% ENTER YOUR CODE HERE!!! %%%%
        CVSVMModel1 = crossval(SVMModel,'kfold',10);
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

        % Estimate the out-of-sample misclassification rate.
        %%%%% ENTER YOUR CODE HERE!!! %%%%
        classLoss1 = kfoldLoss(CVSVMModel1)

        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

        % Compute validation accuracy
        %%%%% ENTER YOUR CODE HERE!!! %%%%
        validationAccuracy1 = 1 - classLoss1

        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

        % Compute validation predictions and scores
        [validationPredictions1, validationScores1] = kfoldPredict(CVSVMModel1);

        % Display confusion matrix
        [Conf_Mat,order] = confusionmat(grpTrain,validationPredictions1);
        disp(Conf_Mat)

        % 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
        CVSVMModel2 = crossval(SVMModel,'holdout',0.2);

        % Estimate the out-of-sample misclassification rate.
        classLoss2 = kfoldLoss(CVSVMModel2)

        % Compute validation accuracy
        validationAccuracy2 = 1 - classLoss2

        % Compute validation predictions and scores
        [validationPredictions2, validationScores2] = kfoldPredict(CVSVMModel2);

        % Display confusion matrix
        [Conf_Mat,order] = confusionmat(grpTrain,validationPredictions2);
        disp(Conf_Mat)

        % Compute and display accuracy "per class"
        cp2 = classperf(grpTrain,validationPredictions1);
        cp2.PositivePredictiveValue
        cp2.NegativePredictiveValue

        %% 5.3: Display ROC and compute AUC

        % Compute the ROC curve.
        SVMModel = fitPosterior(SVMModel);
        [~,score_svm] = resubPredict(SVMModel);

        [X2,Y2,T,AUC] = perfcurve(grpTrain,score_svm(:,1),'Cancer');

        % Plot the ROC curve
        figure(3), plot(X2,Y2)
        xlabel('False positive rate'); ylabel('True positive rate');
        title('ROC Curves for SVM, Training dataset')

        % Display the area under the curve.
        disp(AUC)
```

```
%% Part 6 - Testing

%Compute accuracy
testclassLoss = loss(SVMModel, dataTest(:,fsCVforN1), grpTest)
testAccuracy1 = 1 - testclassLoss

% Label the test sample observations.
% Display the results for the observations in the test sample.
[label_test,score_test] = predict(SVMModel,dataTest(:,fsCVforN1));

% 3.2: Display confusion matrix
% PASTE YOUR CODE HERE!
[Conf_Mat_test,order_test] = confusionmat(grpTest,label_test)
disp(Conf_Mat_test)

% Compute and display accuracy "per class"
cp3 = classperf(grpTest,label_test);
cp3.PositivePredictiveValue
cp3.NegativePredictiveValue

% Display ROC and compute AUC
% Compute the ROC curve.
[X_test,Y_test,T_test,AUC_test] = perfcurve(grpTest,score_test(:,1),'Cancer');

% Plot the ROC curve
figure(4), plot(X_test,Y_test)
xlabel('False positive rate')
ylabel('True positive rate')
title('ROC for Classification by SVM, Test Data Set')

disp(AUC_test)
```