```
clear; close all; clc;

%% Question 1%%

% Feature pairs (1,2)

% Load dataset X and ys
load fisheriris
% Choose setosa and versicolor
inds = ~strcmp(species,'virginica');
X = meas(inds,1:2);
y = species(inds);

% Feature Pairs (1,2), (1,3), (1,4), (2,3), (2,4), (3,4)
feature_pairs = [1, 2; 1, 3; 1, 4; 2, 3; 2, 4; 3, 4];

% For each feature pairs
for i = 1:size(feature_pairs, 1)
    % Choose one pair
    X = meas(inds, feature_pairs(i, :));

    % Train SVM model
    SVMModel = fitcsvm(X, y);

    % Get SVs
    sv       = SVMModel.SupportVectors;
    beta     = SVMModel.Beta; % Linear predictor coefficients
    b        = SVMModel.Bias; % Bias term

    % Plot
    figure
    gscatter(X(:, 1), X(:, 2), y)
    hold on
    plot(sv(:, 1), sv(:, 2), 'ko', 'MarkerSize', 10)
    X1 = linspace(min(X(:,1)),max(X(:,1)),100);
    X2 = -(beta(1)/beta(2)*X1)-b/beta(2);
    plot(X1,X2,'-')
    m = 1/sqrt(beta(1)^2 + beta(2)^2);  % Margin half-width
    X1margin_low = X1+beta(1)*m^2;
    X2margin_low = X2+beta(2)*m^2;
    X1margin_high = X1-beta(1)*m^2;
    X2margin_high = X2-beta(2)*m^2;
    plot(X1margin_high,X2margin_high,'b--')
    plot(X1margin_low,X2margin_low,'r--')
    legend('versicolor', 'setosa', 'Support Vector')
    title("Features " + int2str(feature_pairs(i, 1)) + " & " + int2str
(feature_pairs(i, 2)) + ...
          " - Number of SVs = " + int2str(length(sv)))
    hold off
end
clear; close all; clc;
%% Question 2 - Cross-Validate SVM %%
```

```
% Load dataset
load fisheriris
% Choose setosa and versicolor
inds = ~strcmp(species,'virginica');
% Use all 4 features
X = meas(inds,:);
y = species(inds);

Y= categorical(species(inds)); % Convert species to categorical for coloring


SVMModel = fitcsvm(X,y,'Standardize',true,'KernelFunction','RBF',...
    'KernelScale','auto');

% 10-Fold Cross-Validation
CVSVMModel1 = crossval(SVMModel,'Kfold',10,'Leaveout','off');
classLoss = kfoldLoss(CVSVMModel1);

% Leave-One-Out Cross-Validation (LOOCV)
CVSVMModel2          = crossval(SVMModel,'Leaveout','on');
leave_one_out_cv     = kfoldLoss(CVSVMModel2);

% Initialize arrays to store cross-validation results
results = [];

% Loop through each feature individually
for i = 1:4
    % Select a single feature
    X_single = X(:, i);

    % Train SVM model with a single feature
    SVMModel = fitcsvm(X_single, y, 'Standardize', true, 'KernelFunction', 'RBF',↙
'KernelScale', 'auto');

    % 10-Fold Cross-Validation
    CVSVMModel1 = crossval(SVMModel, 'Kfold', 10);
    classLoss_10fold = kfoldLoss(CVSVMModel1);

    % Leave-One-Out Cross-Validation (LOOCV)
    CVSVMModel2 = crossval(SVMModel, 'Leaveout', 'on');
    classLoss_LOOCV = kfoldLoss(CVSVMModel2);

    % Store the results
    results = [results; {['Feature ', num2str(i)], classLoss_10fold,↙
classLoss_LOOCV}];
end

% Display results
resultsTable = cell2table(results, 'VariableNames', {'Feature', '10FoldError',↙
'LOOCVError'});
disp(resultsTable);
y_categorical = categorical(y);
```

```
% Loop through each feature individually
for i = 1:4
    % Select a single feature
    X_single = X(:, i);

    % Train SVM model with a single feature
    SVMModel = fitcsvm(X_single, y, 'Standardize', true, 'KernelFunction', 'RBF',↙
'KernelScale', 'auto');

    % Create a range of values for plotting decision boundaries
    x_min = min(X_single) - 1;
    x_max = max(X_single) + 1;
    x_range = linspace(x_min, x_max, 100)';

    % Predict scores over the range to find the decision boundary
    [~, scores] = predict(SVMModel, x_range);

    % Plot the data and the decision boundary
    figure
    gscatter(X_single, zeros(size(X_single)), y_categorical, 'rb', 'xo');
    hold on
    % plot(x_range, scores(:, 2), 'k-', 'LineWidth', 2); % Decision boundary
    plot(x_range, zeros(size(x_range)), 'k--'); % Decision line at score 0
    title(['Decision Boundary for Feature ', num2str(i)])
    xlabel(['Feature ', num2str(i)])
    ylabel('Decision Score')
    legend('Setosa', 'Versicolor', 'Decision Boundary', 'Location', 'Best')
    hold off
end
clear; clc; close all;

%% Question 3 Multiclass SVM %%

% Load data
load fisheriris
% Choose features 1 and 2
X = meas(:,1:2);
Y = species;

SVMModels   = cell(3,1);
classes     = unique(Y);
rng(1); % For reproducibility

for j = 1:numel(classes)
    indx            = strcmp(Y,classes(j)); % Create binary classes for each↙
classifier
    SVMModels{j}    = fitcsvm(X,indx,'ClassNames',[false true],'Standardize',↙
false,...
        'KernelFunction','linear','BoxConstraint',1);
end

% Examine scatter plot of the data
figure
```

```
gscatter(X(:,1),X(:,2),Y);
hold on
h = gca;
lims = [h.XLim h.YLim]; % Extract the x and y axis limits
title('{\bf Scatter Diagram of Iris Measurements}');
xlabel('Sepal Length (cm)');
ylabel('Sepal Width (cm)');
legend('Location','Northwest');
hold off

d = 0.02;
[x1Grid,x2Grid] = meshgrid(min(X(:,1)):d:max(X(:,1)),...
    min(X(:,2)):d:max(X(:,2)));
xGrid = [x1Grid(:),x2Grid(:)];
N = size(xGrid,1);
Scores = zeros(N,numel(classes));

for j = 1:numel(classes)
    [~,score] = predict(SVMModels{j},xGrid);
    Scores(:,j) = score(:,2); % Second column contains positive-class scores
end

[~,maxScore] = max(Scores,[],2);

figure
h(1:3) = gscatter(xGrid(:,1),xGrid(:,2),maxScore,...
    [0.1 0.5 0.5; 0.5 0.1 0.5; 0.5 0.5 0.1]);
hold on
h(4:6) = gscatter(X(:,1),X(:,2),Y);

for j = 1:numel(classes)
    sv = SVMModels{j}.SupportVectors;
    h(6+j) = plot(sv(:,1), sv(:,2), 'ko', 'MarkerSize', 8, 'LineWidth', 1.5,↙
'DisplayName', 'Support Vectors');
end
title('{\bf Iris Classification Regions}');
xlabel('Sepal Length (cm)');
ylabel('Sepal Width (cm)');
legend(h,{'setosa region','versicolor region','virginica region',...
    'observed setosa','observed versicolor','observed virginica','Support↙
Vectors'},...
    'Location','Northwest');
axis tight
hold off
clear; clc; close all;

%% Question 4 %% Optimize an SVM classifier

% Generate data
rng('default') % For reproducibility
grnpop = mvnrnd([1,0],eye(2),10);
redpop = mvnrnd([0,1],eye(2),10);
```

```
% View the base points
figure;
plot(grnpop(:,1),grnpop(:,2),'go')
hold on
plot(redpop(:,1),redpop(:,2),'ro')
hold off

% Generate 100 data points
redpts = zeros(100,2);
grnpts = redpts;
for i = 1:100
    grnpts(i,:) = mvnrnd(grnpop(randi(10),:),eye(2)*0.02);
    redpts(i,:) = mvnrnd(redpop(randi(10),:),eye(2)*0.02);
end

% View the data points
figure
plot(grnpts(:,1),grnpts(:,2),'go')
hold on
plot(redpts(:,1),redpts(:,2),'ro')
hold off

% Prepare data for classification

cdata = [grnpts;redpts];
grp = ones(200,1);
grp(101:200) = -1;

% Prepare cross validation
c = cvpartition(200,'KFold',10);

% Optimize fit
opts = struct('CVPartition',c,'AcquisitionFunctionName', ...
    'expected-improvement-plus');
Mdl = fitcsvm(cdata,grp,'KernelFunction','rbf', ...
    'OptimizeHyperparameters','auto','HyperparameterOptimizationOptions',opts);
```