```
% SOLUTION 1
load("usps_all.mat");

% Merge the resulting 2000 entries into a single matrix of size 256 × 2000 that will be used as your train set.
usps_train_class1 = double(data(:,1:1000,3)')./255;
usps_train_class2 = double(data(:,1:1000,8)')./255;

% Merge the train set images into a single matrix
%Extract the remaining 200 images for each class
usps_test_class1 = double(data(:,1001:1100,3)')./255;
usps_test_class2 = double(data(:,1001:1100,8)')./255;


% Handpick ten pixels that are least relevant for classifying the data
% Here we choose the first ten pixels, assuming they are on the edge of the image and thus contain less information for classification
% You should adjust this based on your own analysis
least_relevant_pixels = 1:10;

% Remove the least relevant pixels from the training and testing data
% Remove the least relevant pixels from the training and testing data
usps_train_class1(:, least_relevant_pixels) = [];
usps_train_class2(:, least_relevant_pixels) = [];
usps_test_class1(:, least_relevant_pixels) = [];
usps_test_class2(:, least_relevant_pixels) = [];


% Concatenate
usps_train = cat(1, usps_train_class1, usps_train_class2)';
usps_test = cat(1, usps_test_class1, usps_test_class2)';

% Set the number of samples
samples_count = 1100;
class1_labels = zeros(samples_count, 1);
class2_labels = ones(samples_count, 1);

% Concatenate training and test labels
train_labels = cat(1, class1_labels(1:1000), class2_labels(1:1000));
test_labels = cat(1, class1_labels(1001:end), class2_labels(1001:end));

% Fit a Soft SVM model to the data
% 'Standardize' is set to true to standardize the predictors
% 'KernelFunction' is set to 'linear' to use a linear kernel function
% 'KernelScale' is set to 'auto' to automatically scale the predictors
% 'BoxConstraint' is set to 1 for a soft-margin SVM
svm_model = fitcsvm(usps_train', train_labels, 'Standardize',true,'KernelFunction','linear','KernelScale','auto','BoxConstraint', 1);
svm_model = fitPosterior(svm_model, usps_train',train_labels);
[labels, posterior] = predict(svm_model, usps_test');
% Count the number of mislabeled points
mislabeled = sum(labels ~= test_labels);
disp(['Mislabeled Points: ', num2str(mislabeled)]);
% Calculate accuracy
accuracy = sum(labels == test_labels) / length(test_labels) * 100;
disp(['Accuracy: ', num2str(accuracy), '%']);
```

```
Mislabeled Points: 5
Accuracy: 97.5%
```

Load the USPS dataset

```
load("usps_all.mat");

% Normalize the data by dividing by 255 and split it into training and testing sets for two classes
% The first 1000 entries of each class are used for training
usps_train_class1 = double(data(:,1:1000,3)')./255;
usps_train_class2 = double(data(:,1:1000,8)')./255;

% The next 100 entries of each class are used for testing
usps_test_class1 = double(data(:,1001:1100,3)')./255;
usps_test_class2 = double(data(:,1001:1100,8)')./255;

% Concatenate the training data from both classes
usps_train = cat(1, usps_train_class1, usps_train_class2)';
```

```
% Concatenate the testing data from both classes
usps_test = cat(1, usps_test_class1, usps_test_class2)';

% Handpick fifty pixels that are most relevant for classifying the data
% Here we choose the 50 pixels in the center of the image, assuming they contain the most information for classification
% You should adjust this based on your own analysis
most_relevant_pixels = 104:153;

% Keep only the most relevant pixels in the training and testing data
usps_train = usps_train(most_relevant_pixels, :);
usps_test = usps_test(most_relevant_pixels, :);

% Set the number of samples
samples_count = 1100;
class1_labels = zeros(samples_count, 1);
class2_labels = ones(samples_count, 1);

% Concatenate training and test labels
train_labels = cat(1, class1_labels(1:1000), class2_labels(1:1000));
test_labels = cat(1, class1_labels(1001:end), class2_labels(1001:end));

% Fit a Soft SVM model to the data
% 'Standardize' is set to true to standardize the predictors
% 'KernelFunction' is set to 'linear' to use a linear kernel function
% 'KernelScale' is set to 'auto' to automatically scale the predictors
% 'BoxConstraint' is set to 1 for a soft-margin SVM
svm_model = fitcsvm(usps_train', train_labels, 'Standardize',true,'KernelFunction','linear','KernelScale','auto','BoxConstraint', 1);
svm_model = fitPosterior(svm_model, usps_train',train_labels);
[labels, posterior] = predict(svm_model, usps_test');
% Count the number of mislabeled points
mislabeled = sum(labels ~= test_labels);
disp(['Mislabeled Points: ', num2str(mislabeled)]);
% Calculate accuracy
accuracy = sum(labels == test_labels) / length(test_labels) * 100;
disp(['Accuracy: ', num2str(accuracy), '%']);
```

```
Mislabeled Points: 24
Accuracy: 88%
```

Published with MATLAB® R2023b