```
function [GPRmodel, validationRMSE] = trainRegressionModel(trainingData)
% [trainedModel, validationRMSE] = trainRegressionModel(trainingData)
% Returns a trained regression model and its RMSE. This code recreates the
% model trained in Regression Learner app. Use the generated code to
% automate training the same model with new data, or to learn how to
% programmatically train models.
%
%  Input:
%      trainingData: A table containing the same predictor and response
%       columns as those imported into the app.
%
%
%  Output:
%      trainedModel: A struct containing the trained regression model. The
%       struct contains various fields with information about the trained
%       model.
%
%      trainedModel.predictFcn: A function to make predictions on new data.
%
%      validationRMSE: A double representing the validation RMSE. In the
%       app, the Models pane displays the validation RMSE for each model.
%
% Use the code to train the model with new data. To retrain your model,
% call the function from the command line with your original data or new
% data as the input argument trainingData.
%
% For example, to retrain a regression model trained with the original data
% set T, enter:
%   [trainedModel, validationRMSE] = trainRegressionModel(T)
%
% To make predictions with the returned 'trainedModel' on new data T2, use
%   yfit = trainedModel.predictFcn(T2)
%
```

```
% T2 must be a table containing at least the same predictor columns as used
% during training. For details, enter:
%   trainedModel.HowToPredict

% Auto-generated by MATLAB on 05-Dec-2024 11:19:12


% Extract predictors and response
% This code processes the data into the right shape for training the
% model.
inputTable = trainingData;
predictorNames = {'CAC', 'load', 'Fd', 'time'};
predictors = inputTable(:, predictorNames);
response = inputTable.release_obs;
isCategoricalPredictor = [false, false, false, false];

% Train a regression model
% This code specifies all the model options and trains the model.
regressionGP = fitrgp(...
    predictors, ...
    response, ...
    'BasisFunction', 'none', ...
    'KernelFunction', 'matern52', ...
    'Standardize', true);

% Create the result struct with predict function
predictorExtractionFcn = @(t) t(:, predictorNames);
gpPredictFcn = @(x) predict(regressionGP, x);
GPRmodel.predictFcn = @(x) gpPredictFcn(predictorExtractionFcn(x));

% Add additional fields to the result struct
GPRmodel.RequiredVariables = {'CAC', 'load', 'Fd', 'time'};
GPRmodel.RegressionGP = regressionGP;
```

```
GPRmodel.About = 'This struct is a trained model exported from Regression Learner R2024a.';
GPRmodel.HowToPredict = sprintf('To make predictions on a new table, T, use: \n  yfit = c.predictFcn(T)
\nreplacing "c" with the name of the variable that is this struct, e.g. "trainedModel". \n \nThe table, T, must
contain the variables returned by: \n  c.RequiredVariables \nVariable formats (e.g. matrix/vector,
datatype) must match the original training data. \nAdditional variables are ignored. \n \nFor more
information, see <a href="matlab:helpview(fullfile(docroot, "stats", "stats.map"),
"appregression_exportmodeltoworkspace")">How to predict using an exported model</a>.');

% Extract predictors and response
% This code processes the data into the right shape for training the
% model.
inputTable = trainingData;
predictorNames = {'CAC', 'load', 'Fd', 'time'};
predictors = inputTable(:, predictorNames);
response = inputTable.release_obs;
isCategoricalPredictor = [false, false, false, false];

% Perform cross-validation
partitionedModel = crossval(GPRmodel.RegressionGP, 'KFold', 10);

% Compute validation predictions
validationPredictions = kfoldPredict(partitionedModel);

% Compute validation RMSE
validationRMSE = sqrt(kfoldLoss(partitionedModel, 'LossFun', 'mse'));

save('model_data.mat', 'predictors', 'response', 'validationPredictions');
```