```
%%                                    Note:
% Note 1:   Dependent varaible = ResponseVar = SalePrice
%-------------------------------------------------------------------------
% Note 2:   lrmPred  = Linear Regression Model, Prediction of Y (SalePrice)
%           lrmTest  = Linear Regression Model, Test Data.
%           lrmTrain = Linear Regression Model, Train Data
% ------------------------------------------------------------------------
% Note 3:   dtmPred  = Decision Tree Model, Prediction of Y (SalePrice)
%           dtmTest  = Linear Regression Model, Test Data.
%           dtmTrain = Linear Regression Model, Train Data.

%%                            Load the data set
houses = readtable('houses.csv');

%%         Partition the data set into Train and Test                 %%

% For reproducibility, to reproduce always the same results.
rng('default')

% Partition the data set, reserve 30% for testing
c = cvpartition(height(houses), "holdout", 0.3);

% Split the data
trainData = houses(training(c), :);
testData = houses(test(c), :);

% Lets ignore the "Id" column:
Train = removevars(trainData, 'Id');
Test = removevars(testData, 'Id');

%%                       LINEAR REGRESSION MODEL  (lrm)                   %%

% Train the model
lrm = fitlm(Train, "PredictorVars",["totalBath", "BedroomAbvGr", ...
"KitchenAbvGr", "Fireplaces", "GarageCars", "totalSpcBlt_standardized"...
"LotArea_standardized", "GrLivArea_normalized", "Neighborhood_BrkSide"...
"Neighborhood_ClearCr", "Neighborhood_CollgCr", "Neighborhood_Crawfor"...
"Neighborhood_Edwards", "Neighborhood_Gilbert", "Neighborhood_IDOTRR"...
"Neighborhood_MeadowV", "Neighborhood_Mitchel", "Neighborhood_NAmes"...
"Neighborhood_NPkVill", "Neighborhood_NWAmes", "Neighborhood_NoRidge"...
"Neighborhood_NridgHt", "Neighborhood_OldTown", "Neighborhood_SWISU"...
"Neighborhood_Sawyer", "Neighborhood_SawyerW", "Neighborhood_Somerst"...
"Neighborhood_StoneBr", "Neighborhood_Timber", "Neighborhood_Veenker"...
"BsmtFinType1_BLQ", "BsmtFinType1_GLQ", "BsmtFinType1_LwQ"...
"BsmtFinType1_Rec", "BsmtFinType1_Unf"], "ResponseVar", "SalePrice");

% Grahp for the lrm
plotAdded(lrm)

%% Evaluate the Lineal Regression Training Model, metrics:
yTrain = Train.SalePrice;

lrmPredTrain = predict(lrm, Train);

lrmTrainMAE = mean(abs(yTrain - lrmPredTrain));

lrmTrainRMSE = sqrt(mean((yTrain - lrmPredTrain).^2));

lrmTrainRSquared = lrm.Rsquared.Ordinary;
```

```matlab
% Plot residuals
residuals = yTrain - lrmPredTrain;

figure;
scatter(lrmPredTrain, residuals);
xlabel('Predicted Train Values');
ylabel('Residuals');
title('(lrm)Residuals vs Predicted Train Values');

% Adding horizontal line at zero:
hold on;
plot([min(lrmPredTrain), max(lrmPredTrain)], [0, 0], 'k--');
hold off;

%% Evaluate the Linear Regression Test Model, metrics:

yTest = Test.SalePrice;

lrmPredTest = predict(lrm, Test);

lrmTestMAE = mean(abs(yTest - lrmPredTest));

lrmTestRMSE = sqrt(mean((yTest - lrmPredTest).^2));

lrmTest_SSres = sum((yTest - lrmPredTest).^2);
lrmTest_SStot = sum((yTest - mean(yTest)).^2);
lrmTestRSquared = 1 - (lrmTest_SSres / lrmTest_SStot);

% Plot residuals
residuals = yTest - lrmPredTest;

figure;
scatter(lrmPredTest, residuals);
xlabel('Predicted Test Values');
ylabel('Residuals');
title('(lrm)Residuals vs Predicted Test Values');

% Adding horizontal line at zero:
hold on;
plot([min(lrmPredTest), max(lrmPredTest)], [0, 0], 'k--');
hold off;

%% Linear Regression Print Results:
fprintf("===============================================\n");
fprintf("Linear Regression Model, Train Vs Test Results:\n");
fprintf("-----------------------------------------------\n");
fprintf('(Train) Mean Absolute Error: %.3f\n', lrmTrainMAE);
fprintf('(Test) Mean Absolute Error: %.3f\n', lrmTestMAE);
fprintf("-----------------------------------------------\n");
fprintf('(Train) Root Mean Squared Error: %.3f\n', lrmTrainRMSE);
fprintf('(Test) Root Mean Squared Error: %.3f\n', lrmTestRMSE);
fprintf("-----------------------------------------------\n");
fprintf('(Train) R-squared: %.3f\n', lrmTrainRSquared);
fprintf('(Test) R-squared: %.3f\n', lrmTestRSquared);

%%                      DECISION TREE MODEL (dtm)

% Train the model, we define the splits and the leafs.
```

```
dtm = fitrtree(Train,"SalePrice", 'MaxNumSplits', 99, 'MinLeafSize',8);

% Graph for the dtm
view(dtm, 'Mode', 'graph');
%% Evaluate the Decision Tree Regreesion Training Model, metrics:

dtmPredTrain = predict(dtm, Train);

dtmTrainMAE = mean(abs(yTrain - dtmPredTrain));

dtmTrainRMSE = sqrt(mean((yTrain - dtmPredTrain).^2));

dtmTrain_SSres = sum((yTrain - dtmPredTrain).^2);
dtmTrain_SStot = sum((yTrain - mean(yTrain)).^2);
dtmTrainRSquared = 1 - (dtmTrain_SSres / dtmTrain_SStot);

% Plot residuals
residuals = yTrain - dtmPredTrain;

figure;
scatter(dtmPredTrain, residuals);
xlabel('Predicted Train Values');
ylabel('Residuals');
title('(dtm)Residuals vs Predicted Train Values');

% Adding horizontal line at zero:
hold on;
plot([min(dtmPredTrain), max(dtmPredTrain)], [0, 0], 'k--');
hold off;

%% Evaluate the Decision Tree Test model, metrics.

dtmPredTest = predict(dtm, Test);

dtmTestMAE = mean(abs(yTest - dtmPredTest));

dtmTestRMSE = sqrt(mean((yTest - dtmPredTest).^2));

dtmTest_SSres = sum((yTest - dtmPredTest).^2);
dtmTest_SStot = sum((yTest - mean(yTest)).^2);
dtmTestRSquared = 1 - (dtmTest_SSres / dtmTest_SStot);

% Plot residuals
residuals = yTest - dtmPredTest;

figure;
scatter(dtmPredTest, residuals);
xlabel('Predicted Test Values');
ylabel('Residuals');
title('(dtm)Residuals vs Predicted Test Values');

% Adding horizontal line at zero:
hold on;
plot([min(dtmPredTest), max(dtmPredTest)], [0, 0], 'k--');
hold off;

%% Decision Tree Print Results:
fprintf("===============================================\n");
fprintf("===============================================\n");
```

```matlab
fprintf("Decision Tree Model, Train Vs Test Results:\n");
fprintf("-----------------------------------------------\n");
fprintf('(Train) Mean Absolute Error: %.3f\n', dtmTrainMAE);
fprintf('(Test) Mean Absolute Error: %.3f\n', dtmTestMAE);
fprintf("-----------------------------------------------\n");
fprintf('(Train) Root Mean Squared Error: %.3f\n', dtmTrainRMSE);
fprintf('(Test) Root Mean Squared Error: %.3f\n', dtmTestRMSE);
fprintf("-----------------------------------------------\n");
fprintf('(Train) R-squared: %.3f\n', dtmTrainRSquared);
fprintf('(Test) R-squared: %.3f\n', dtmTestRSquared);
```

```matlab
% Load the data set
houses = readtable('houses.csv');

%%                           1st BUILD THE MODELS                        %%

% For reproducibility, which means it store a random seed to get the same
% results in futures runnings.
rng('default')

% Partition 30/70
c = cvpartition(height(houses),"holdout",0.3);

% Split the data
train = houses(training(c), :);
test = houses(test(c), :);

% Dependent Variable
y = train.SalePrice;

% Independent variables, Excluding 'SalePrice' and 'Id'
predictorNames = setdiff(train.Properties.VariableNames, {'SalePrice', 'Id'});
X = train{:, predictorNames};
%%
%  X is features matrix and y is the target variable
% Standardize the features before applying Lasso
[X, mu, sigma] = zscore(X);
%%
% Apply Lasso regression with 7-fold cross-validation
[B, FitInfo] = lasso(X, y, 'CV', 7);

% Looking the index with the best Lambda value
minMSEIndex = FitInfo.IndexMinMSE;
optimalLambda = FitInfo.Lambda(minMSEIndex);

% Coefficients for the optimal Lambda value
optimalCoefficients = B(:, minMSEIndex);

% Display the optimal Lambda and coefficients
disp(['Optimal Lambda: ', num2str(optimalLambda)]);
disp('Optimal Coefficients:');
disp(optimalCoefficients);
```