Table of Contents

Introduction

- Problem definition – predict school performance score (SPGScore)
- decision tree and svmwith cv and grid search
- Ensemble Learners: TreeBaggerand Boosting
- Predictor Importance and feature reduction
- Feed forward neural network to predict SPGScorewith best predictors
- Compare various neural network architectures

Loading the data

```
clear; clc; close all;
% load the data
load ml_data

% Remove the SPGrade variable :
ml_data(:,'SPGGrade') = [];

% Store the target variable
ml_data_output = ml_data.SPGScore;

% Remove the class variable : SPGScore;
ml_data(:,'SPGScore') = [];

% Scale the data; Normalize it; use zscore
ml_data{:,:} = zscore(ml_data{:,:});
```

```
% Split the data into training and test sets
% Create the cvpartition variable
pt = cvpartition(ml_data_output, 'HoldOut', 0.25);
```

```
Warning: The training set does not contain points from all groups.
```

```
% Create the training and test tables
nc_train_input = ml_data(training(pt), :);
nc_train_output = ml_data_output(training(pt), :);

nc_test_input = ml_data(test(pt), :);
nc_test_output = ml_data_output(test(pt), :);

% set random seed.
rng(1);
```

Decision Tree Regression (fitrtree)

Fitrtree

Predict

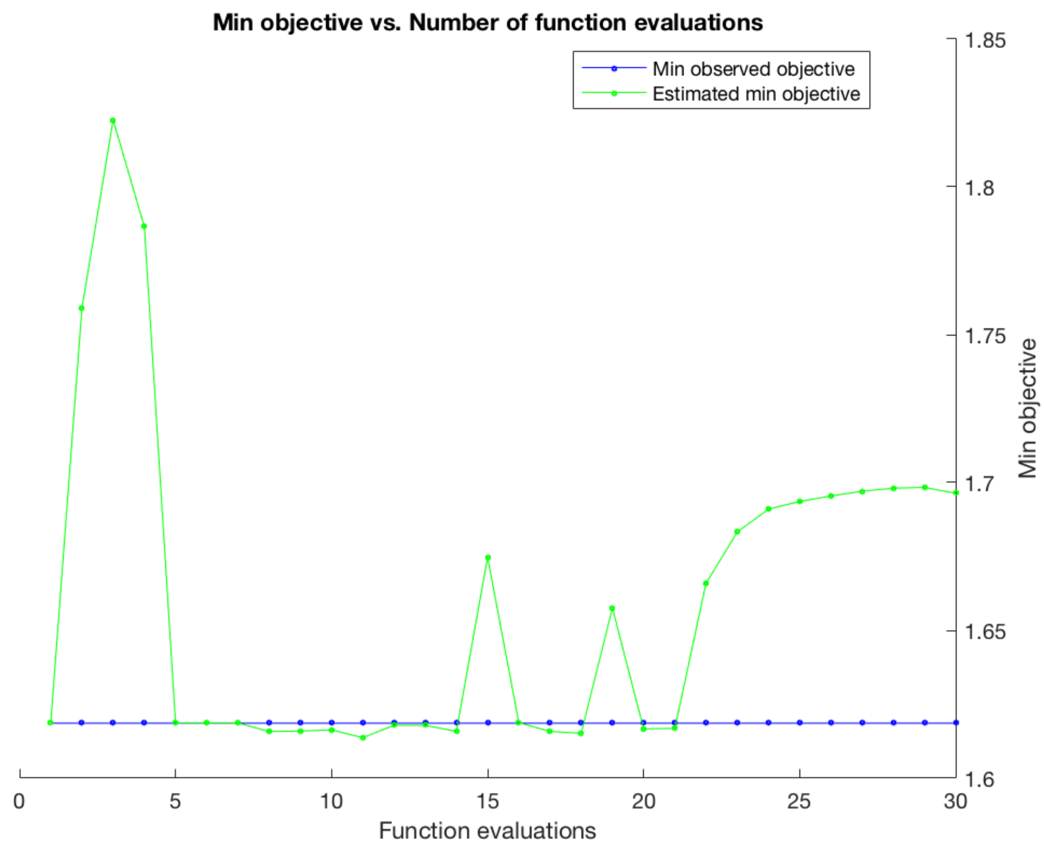
Calculate error

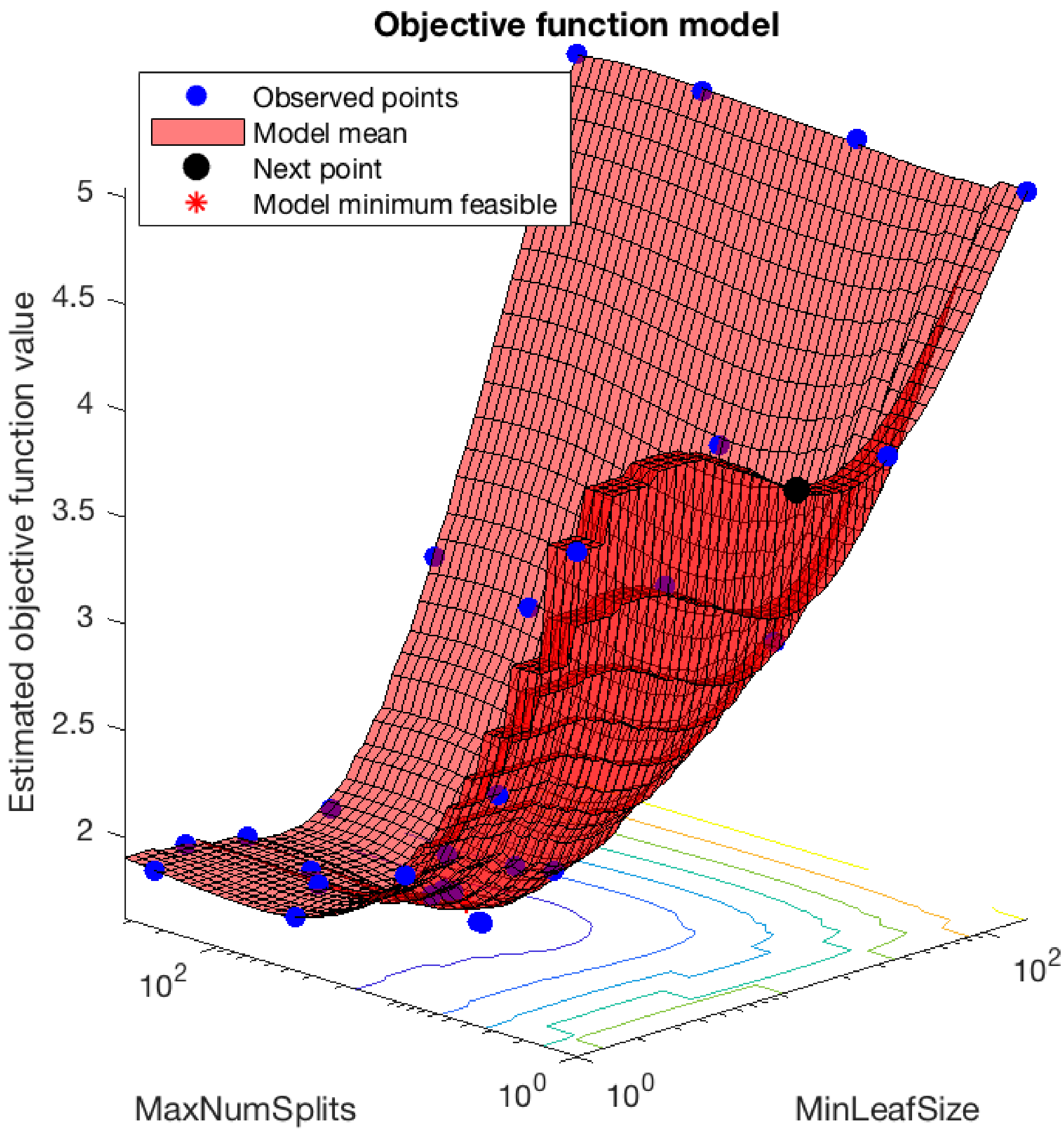
Calculate R-square

Plot Error Histogram

Plot Scatter Plot

```
Mdl_rtree = fitrtree(nc_train_input,nc_train_output,'OptimizeHyperparameters',{'MinLea
    'HyperparameterOptimizationOptions',struct('AcquisitionFunctionName',...
    'expected-improvement-plus'))
```


Min objective vs. Number of function evaluations
Min observed objective
Estimated min objective
1.85
1.8
1.75
1.7
1.65
1.6
Min objective
0
5
10
15
20
25
30
Function evaluations



Iter	Eval result	Objective	Objective runtime	BestSoFar (observed)	BestSoFar (estim.)	MinLeafSize	MaxNumSplits
1	Best	1.6186	0.6223	1.6186	1.6186	7	32
2	Accept	3.6373	0.30916	1.6186	1.7589	64	18
3	Accept	3.546	0.2202	1.6186	1.8226	5	2
4	Accept	1.8832	0.53536	1.6186	1.7866	1	223
5	Accept	1.6186	0.30534	1.6186	1.6187	7	30
6	Accept	3.032	0.17695	1.6186	1.6187	25	3
7	Accept	1.9323	0.20635	1.6186	1.6187	8	14
8	Accept	1.8422	0.20773	1.6186	1.6158	10	319
9	Accept	1.8286	0.30885	1.6186	1.6159	3	105
10	Accept	1.8705	0.32351	1.6186	1.6163	1	37
11	Accept	1.8171	0.28577	1.6186	1.6137	4	31
12	Accept	1.8216	0.23028	1.6186	1.6179	9	64
13	Accept	1.8288	0.28662	1.6186	1.6179	4	329
14	Accept	1.7617	0.26122	1.6186	1.6158	6	42
15	Accept	1.8422	0.23408	1.6186	1.6747	10	30
16	Accept	1.6186	0.23705	1.6186	1.6188	7	31

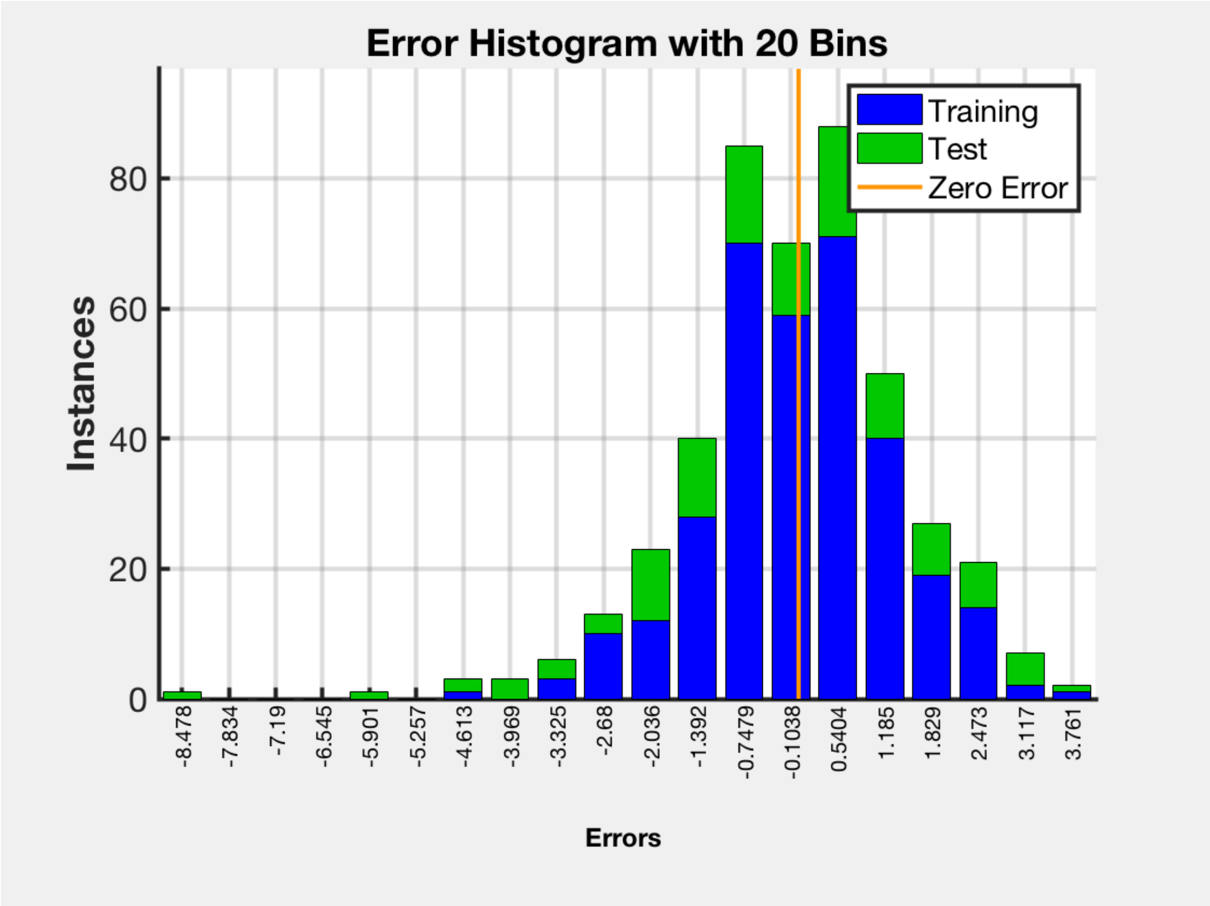
```
|   17 | Accept |      1.8774 |     0.30105 |      1.6186 |      1.6158 |            2 |          327 |
|   18 | Accept |      1.8774 |     0.28406 |      1.6186 |      1.6152 |            2 |           60 |
|   19 | Accept |      2.8695 |     0.20781 |      1.6186 |      1.6576 |           33 |          327 |
|   20 | Accept |      1.6186 |     0.27633 |      1.6186 |      1.6167 |            7 |           30 |
|====================================================================================================|
| Iter | Eval   | Objective   | Objective   | BestSoFar   | BestSoFar   |  MinLeafSize | MaxNumSplits |
|      | result |             | runtime     | (observed)  | (estim.)    |              |              |
|====================================================================================================|
|   21 | Accept |       2.227 |     0.24828 |      1.6186 |      1.6169 |            1 |            9 |
|   22 | Accept |      5.0283 |     0.16493 |      1.6186 |      1.6659 |          165 |            1 |
|   23 | Accept |      1.7617 |     0.22152 |      1.6186 |      1.6832 |            6 |           37 |
|   24 | Accept |      5.0283 |     0.16548 |      1.6186 |       1.691 |          165 |          323 |
|   25 | Accept |       3.986 |     0.20959 |      1.6186 |      1.6936 |            1 |            1 |
|   26 | Accept |      5.0283 |     0.17883 |      1.6186 |      1.6954 |          165 |            9 |
|   27 | Accept |       3.986 |     0.18557 |      1.6186 |      1.6971 |           34 |            1 |
|   28 | Accept |      5.0283 |     0.16023 |      1.6186 |      1.6981 |          165 |           65 |
|   29 | Accept |      2.5613 |      0.2505 |      1.6186 |      1.6984 |            2 |            6 |
|   30 | Accept |      2.7939 |     0.21546 |      1.6186 |      1.6964 |           30 |           87 |

__________________________________________________________
Optimization completed.
MaxObjectiveEvaluations of 30 reached.
Total function evaluations: 30
Total elapsed time: 97.0025 seconds.
Total objective function evaluation time: 7.8204

Best observed feasible point:
    MinLeafSize    MaxNumSplits
    ___________    ____________

         7              32

Observed objective function value = 1.6186
Estimated objective function value = 1.6964
Function evaluation time = 0.6223

Best estimated feasible point (according to models):
    MinLeafSize    MaxNumSplits
    ___________    ____________

         7              32

Estimated objective function value = 1.6964
Estimated function evaluation time = 0.2666
```

```
Mdl_rtree = 
  RegressionTree
                         ResponseName: 'Y'
                CategoricalPredictors: []
                    ResponseTransform: 'none'
                      NumObservations: 330
    HyperparameterOptimizationResults: [1×1 BayesianOptimization]


  Properties, Methods
```

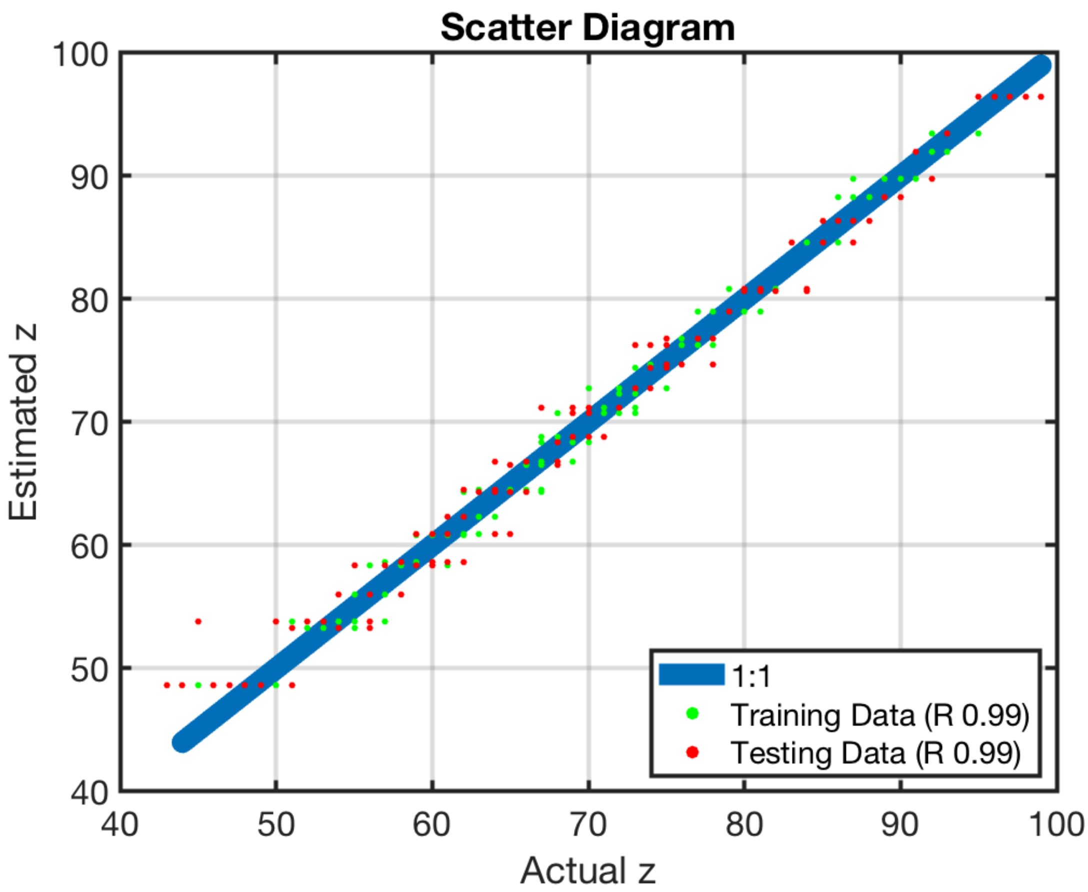
```
% predict
outputs_rtree_train=predict(Mdl_rtree,nc_train_input);
outputs_rtree_test=predict(Mdl_rtree,nc_test_input);
```

```
%------------------------------------------------------------------------------
% calculate the mean square error (MSE) of the test points
mse_train=sum((outputs_rtree_train - nc_train_output).^2)/length(nc_train_output);
mse_test=sum((outputs_rtree_test - nc_test_output).^2)/length(nc_test_output);

%------------------------------------------------------------------------------
% calculate the correlation coefficients for the training and test data
% sets with the associated linear fits hint: check out the function corrcoef
R_train = corrcoef(outputs_rtree_train,nc_train_output);
R_test = corrcoef(outputs_rtree_test,nc_test_output);
r_train=R_train(1,2);
r_test=R_test(1,2);
```

```
% plot error histogram
plotErrorHistogram(nc_train_output, outputs_rtree_train, nc_test_output, outputs_rtree_
```




```
% plot scatter plot
plotScatterDiagram(nc_train_output, outputs_rtree_train, nc_test_output, outputs_rtree_
```

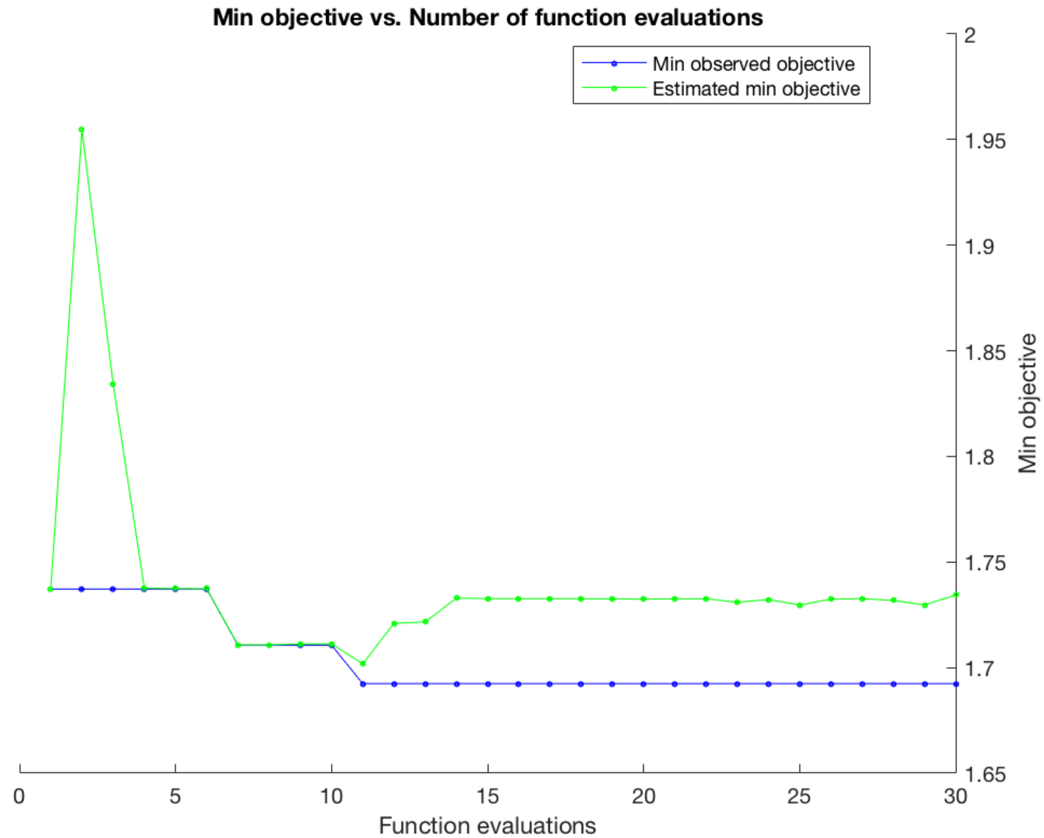


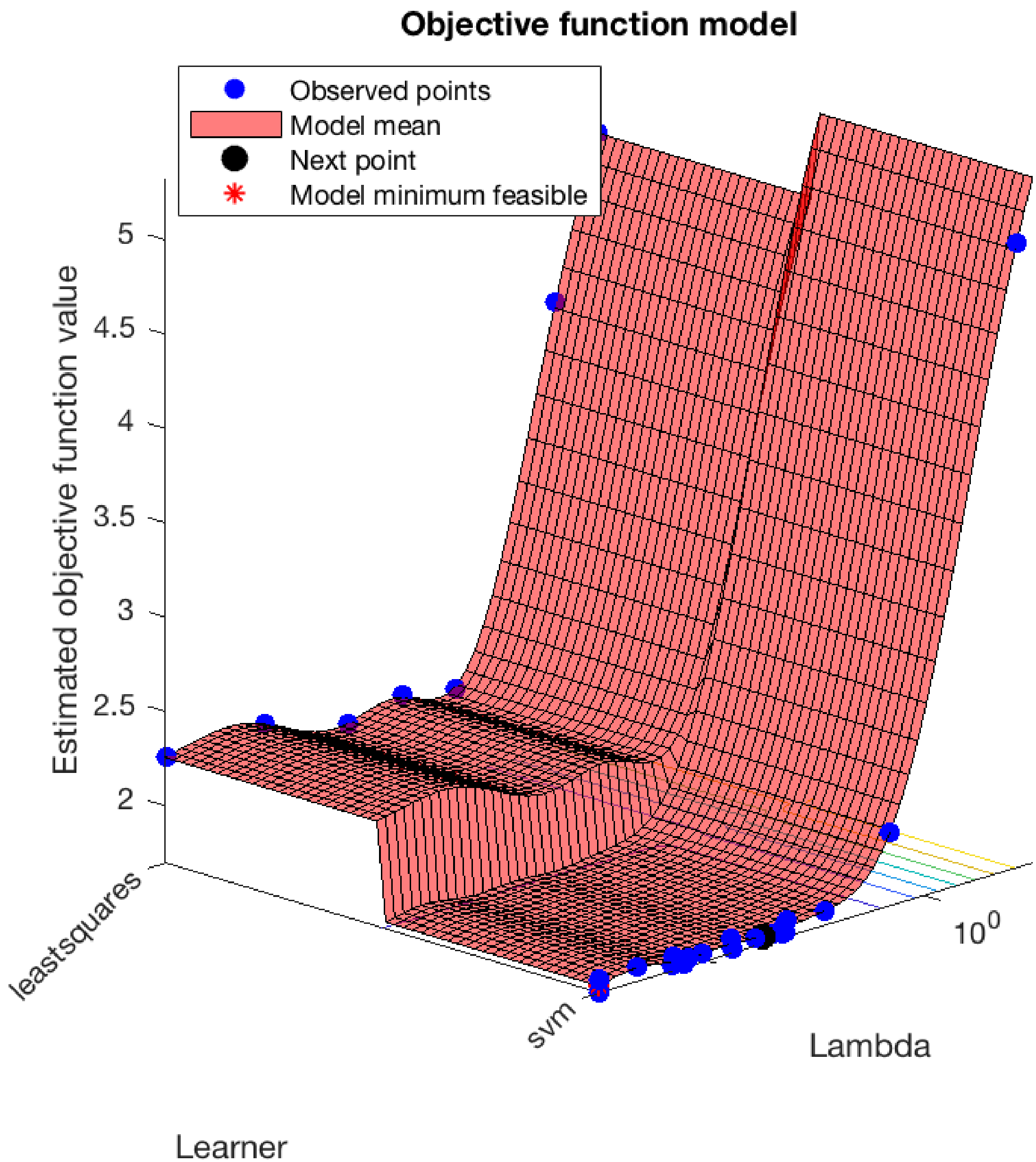
SVM (fitrlinear)

`fitrlinear` efficiently trains linear regression models with high-dimensional, full or sparse predictor data. Available linear regression models include regularized support vector machines (SVM) and least-squares regression methods. `fitrlinear` minimizes the objective function using techniques that reduce computing time (e.g., stochastic gradient descent).

A high-dimensional data set includes many predictor variables. Although such a data set can consume a significant fraction of memory, it must fit in the MATLAB® Workspace. For low- through medium-dimensional predictor data sets, see Alternatives for Lower-Dimensional Data.

```
hyperopts = struct('AcquisitionFunctionName','expected-improvement-plus');
nc_train_input_matrix = nc_train_input{:,:};
[Mdl_svm,FitInfo,HyperparameterOptimizationResults] = fitrlinear(nc_train_input_matrix,
    'OptimizeHyperparameters','auto',...
    'HyperparameterOptimizationOptions',hyperopts)
```


Min objective vs. Number of function evaluations
Min observed objective
Estimated min objective
2
1.95
1.9
1.85
1.8
1.75
1.7
1.65
Min objective
0
5
10
15
20
25
30
Function evaluations




```
|======================================================================================================|
| Iter | Eval   | Objective   | Objective   | BestSoFar   | BestSoFar   |       Lambda |      Learner |
|      | result |             | runtime     | (observed)  | (estim.)    |              |              |
|======================================================================================================|
|    1 | Best   |      1.7372 |      2.2362 |      1.7372 |      1.7372 |   0.00012256 |          svm |
|    2 | Accept |      5.0085 |      0.3269 |      1.7372 |      1.9546 |       127.51 |          svm |
|    3 | Accept |      2.2564 |     0.38839 |      1.7372 |      1.8344 |   3.4291e-08 | leastsquares |
|    4 | Accept |      2.1398 |     0.29407 |      1.7372 |      1.7376 |   0.00051864 | leastsquares |
|    5 | Accept |      1.7407 |     0.30593 |      1.7372 |      1.7374 |   7.2114e-06 |          svm |
|    6 | Accept |      4.0463 |     0.25114 |      1.7372 |      1.7374 |       30.583 | leastsquares |
|    7 | Best   |      1.7106 |     0.18785 |      1.7106 |      1.7108 |   3.0445e-08 |          svm |
|    8 | Accept |      2.2764 |      0.2289 |      1.7106 |      1.7108 |   6.0729e-06 | leastsquares |
|    9 | Accept |      1.7706 |      0.2894 |      1.7106 |      1.7112 |   2.2586e-07 |          svm |
|   10 | Accept |      1.7171 |     0.31524 |      1.7106 |      1.7113 |   3.5621e-05 |          svm |
|   11 | Best   |      1.6924 |     0.22252 |      1.6924 |      1.7018 |   3.0528e-08 |          svm |
|   12 | Accept |       1.757 |     0.17284 |      1.6924 |       1.721 |   3.0436e-08 |          svm |
|   13 | Accept |      1.7778 |      0.2147 |      1.6924 |      1.7216 |   3.3815e-05 |          svm |
|   14 | Accept |      1.7669 |     0.26125 |      1.6924 |      1.7329 |   3.0323e-08 |          svm |
|   15 | Accept |      2.0822 |     0.22893 |      1.6924 |      1.7327 |      0.15785 |          svm |
|   16 | Accept |      1.7727 |     0.15846 |      1.6924 |      1.7327 |    0.0047469 |          svm |
```

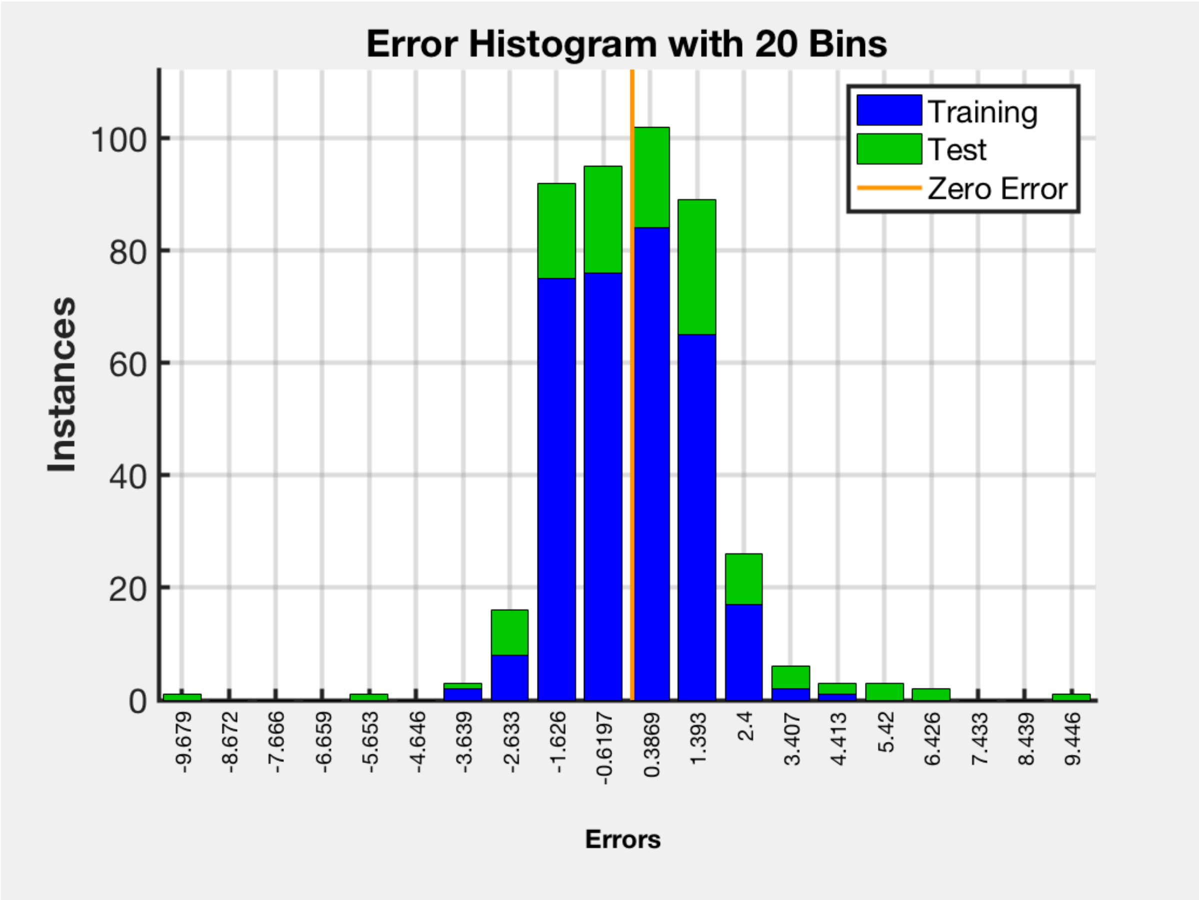
```
|   17 | Accept |      1.7162 |     0.32045 |      1.6924 |      1.7327 |   0.00053764 |          svm |
|   18 | Accept |      1.7457 |     0.27009 |      1.6924 |      1.7326 |   0.00050672 |          svm |
|   19 | Accept |      1.7314 |     0.19942 |      1.6924 |      1.7325 |    0.0003232 |          svm |
|   20 | Accept |      2.1584 |     0.22895 |      1.6924 |      1.7324 |      0.15068 | leastsquares |
|===================================================================================================|
| Iter | Eval   | Objective   | Objective   | BestSoFar   | BestSoFar   |       Lambda |      Learner |
|      | result |             | runtime     | (observed)  | (estim.)    |              |              |
|===================================================================================================|
|   21 | Accept |      1.7293 |     0.22496 |      1.6924 |      1.7326 |   2.2854e-06 |          svm |
|   22 | Accept |      1.7155 |     0.27619 |      1.6924 |      1.7326 |    2.867e-06 |          svm |
|   23 | Accept |      1.7354 |      0.2562 |      1.6924 |      1.7309 |   2.7046e-06 |          svm |
|   24 | Accept |      1.7379 |     0.22745 |      1.6924 |      1.7322 |   3.4764e-06 |          svm |
|   25 | Accept |       1.721 |    0.096524 |      1.6924 |      1.7296 |   0.00061386 |          svm |
|   26 | Accept |      1.7791 |      0.1942 |      1.6924 |      1.7324 |   0.00068042 |          svm |
|   27 | Accept |      4.8745 |     0.17788 |      1.6924 |      1.7327 |       303.03 | leastsquares |
|   28 | Accept |      2.2046 |     0.26928 |      1.6924 |      1.7318 |    0.0093482 | leastsquares |
|   29 | Accept |      1.7246 |     0.16985 |      1.6924 |      1.7297 |   1.5295e-06 |          svm |
|   30 | Accept |      1.7698 |     0.16949 |      1.6924 |      1.7343 |   1.5605e-06 |          svm |

__________________________________________________________
Optimization completed.
MaxObjectiveEvaluations of 30 reached.
Total function evaluations: 30
Total elapsed time: 90.1843 seconds.
Total objective function evaluation time: 9.1637

Best observed feasible point:
      Lambda      Learner
    __________    _______

    3.0528e-08      svm

Observed objective function value = 1.6924
Estimated objective function value = 1.7343
Function evaluation time = 0.22252

Best estimated feasible point (according to models):
     Lambda      Learner
    _________    _______

    2.867e-06      svm

Estimated objective function value = 1.7343
Estimated function evaluation time = 0.24855
```

```
Mdl_svm =
  RegressionLinear
         ResponseName: 'Y'
    ResponseTransform: 'none'
                 Beta: [141×1 double]
                 Bias: 70.4889
               Lambda: 2.8670e-06
              Learner: 'svm'


  Properties, Methods
FitInfo = struct with fields:
                    Lambda: 2.8670e-06
                 Objective: 0.1975
                 PassLimit: 10
                 NumPasses: 10
                BatchLimit: []
             NumIterations: 3300
              GradientNorm: NaN
         GradientTolerance: 0
```

```
      RelativeChangeInBeta: 0.0085
             BetaTolerance: 1.0000e-04
             DeltaGradient: 5.3138
    DeltaGradientTolerance: 0.1000
           TerminationCode: 0
         TerminationStatus: {'Iteration limit exceeded.'}
                     Alpha: [330×1 double]
                   History: []
                   FitTime: 0.0030
                    Solver: {'dual'}
HyperparameterOptimizationResults = 
  BayesianOptimization with properties:

                      ObjectiveFcn: @createObjFcn/theObjFcn
              VariableDescriptions: [3×1 optimizableVariable]
                           Options: [1×1 struct]
                      MinObjective: 1.6924
                   XAtMinObjective: [1×2 table]
             MinEstimatedObjective: 1.7343
          XAtMinEstimatedObjective: [1×2 table]
           NumObjectiveEvaluations: 30
                  TotalElapsedTime: 90.1843
                         NextPoint: [1×2 table]
                            XTrace: [30×2 table]
                    ObjectiveTrace: [30×1 double]
                  ConstraintsTrace: []
                     UserDataTrace: {30×1 cell}
      ObjectiveEvaluationTimeTrace: [30×1 double]
                IterationTimeTrace: [30×1 double]
                        ErrorTrace: [30×1 double]
                  FeasibilityTrace: [30×1 logical]
       FeasibilityProbabilityTrace: [30×1 double]
               IndexOfMinimumTrace: [30×1 double]
             ObjectiveMinimumTrace: [30×1 double]
    EstimatedObjectiveMinimumTrace: [30×1 double]
```

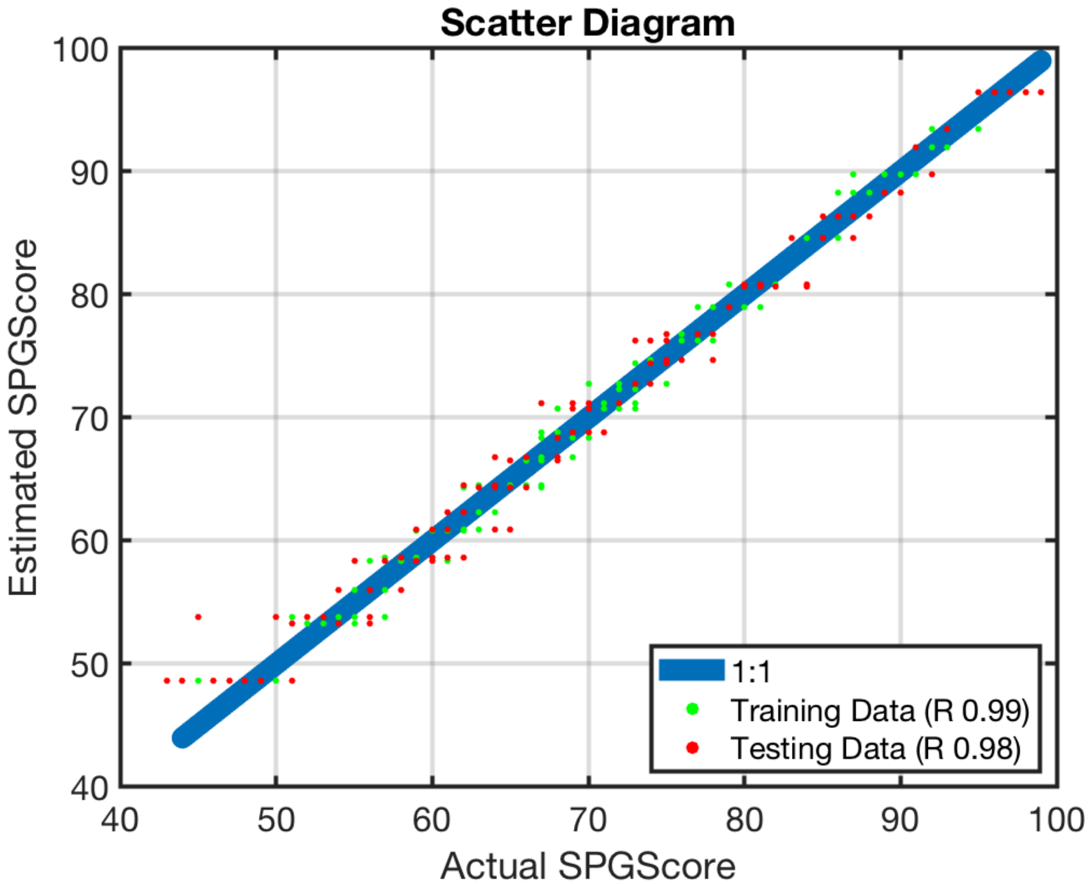
```
% predict
outputs_svm_train=predict(Mdl_svm,nc_train_input_matrix);
nc_test_input_matrix = nc_test_input{:,:};
outputs_svm_test=predict(Mdl_svm,nc_test_input_matrix);
```

```
% plot error histogram
plotErrorHistogram(nc_train_output, outputs_svm_train, nc_test_output, outputs_svm_tes
```




```
%--------------------------------------------------------------------------
% calculate the mean square error (MSE) of the test points
mse_train=sum((outputs_svm_train - nc_train_output).^2)/length(nc_train_output);
mse_test=sum((outputs_svm_test - nc_test_output).^2)/length(nc_test_output);

%--------------------------------------------------------------------------
% calculate the correlation coefficients for the training and test data
% sets with the associated linear fits hint: check out the function corrcoef
R_train = corrcoef(outputs_svm_train,nc_train_output);
R_test = corrcoef(outputs_svm_test,nc_test_output);
r_train=R_train(1,2);
r_test=R_test(1,2);
```

```
% plot scatter plot
plotScatterDiagram(nc_train_output, outputs_rtree_train, nc_test_output, outputs_rtree_
```



TreeBagger Regression

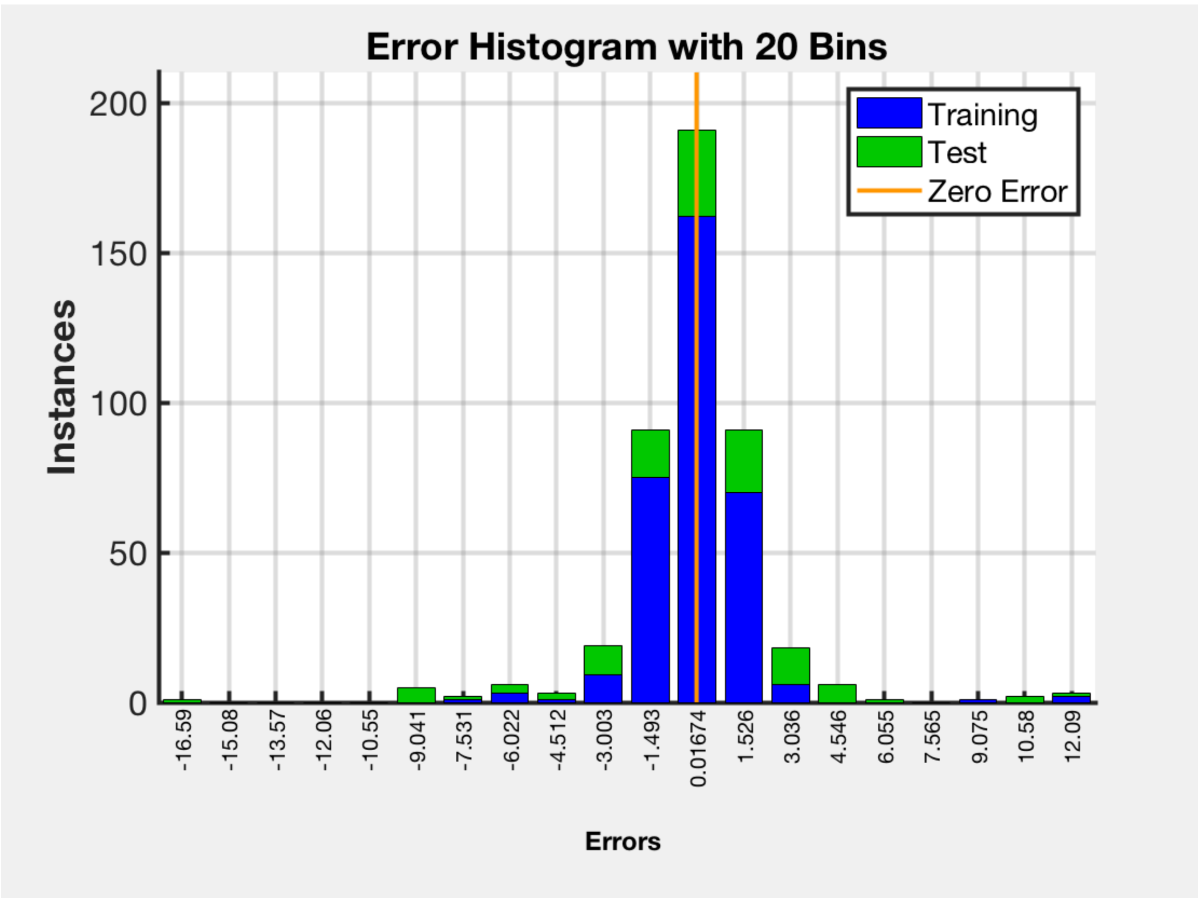
```
Mdl_TB = TreeBagger(...
    100,nc_train_input,nc_train_output,...
    'Method','Regression',...
    'Surrogate','on',...
    'PredictorSelection','curvature',...
    'OOBPredictorImportance','on'...
    )
```

```
Mdl_TB = 
  TreeBagger
Ensemble with 100 bagged decision trees:
                    Training X:             [330x141]
                    Training Y:               [330x1]
                        Method:            regression
                 NumPredictors:                   141
         NumPredictorsToSample:                    47
                   MinLeafSize:                     5
                 InBagFraction:                     1
         SampleWithReplacement:                     1
          ComputeOOBPrediction:                     1
 ComputeOOBPredictorImportance:                     1
                     Proximity:                    []
```

```
Properties, Methods
```

```
% predict
outputs_tb_train=predict(Mdl_TB, nc_train_input);
outputs_tb_test=predict(Mdl_TB,nc_test_input);
```

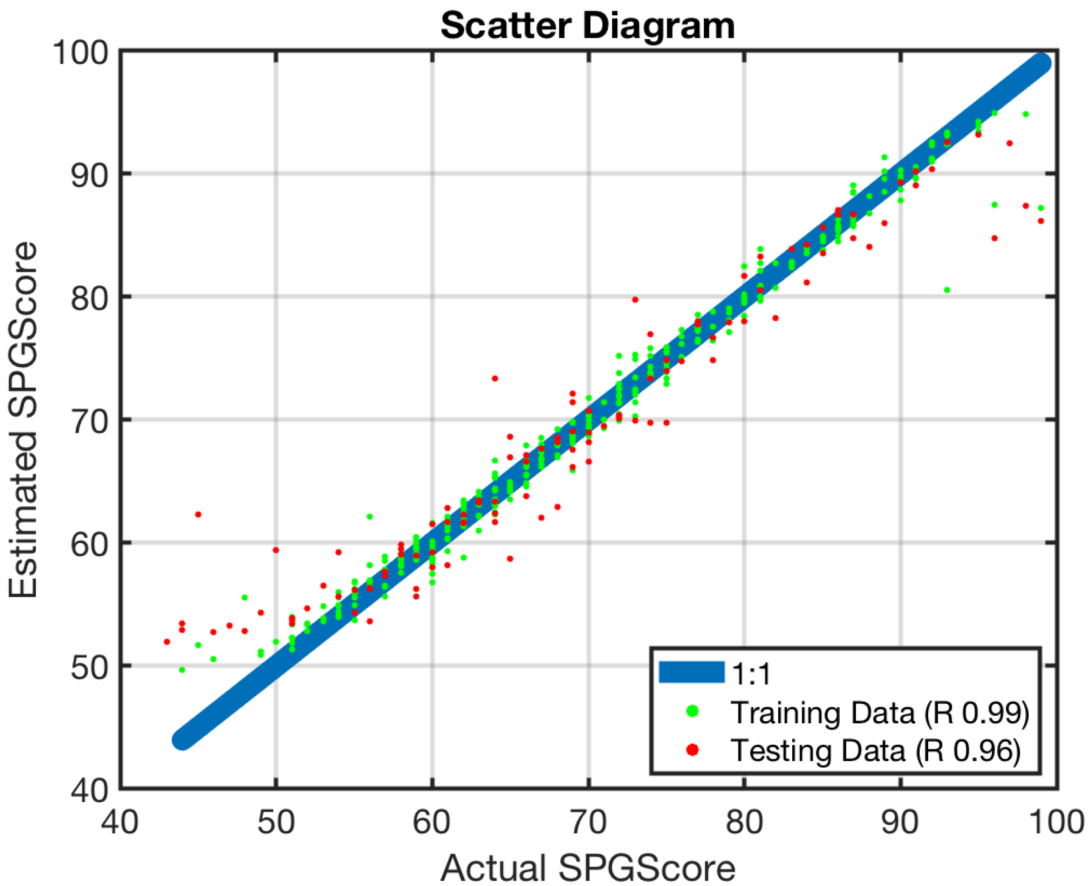
```
% plot error histogram
plotErrorHistogram(nc_train_output, outputs_tb_train, nc_test_output, outputs_tb_test)
```




```
%-------------------------------------------------------------------------
% calculate the mean square error (MSE) of the test points
mse_train=sum((outputs_tb_train - nc_train_output).^2)/length(nc_train_output);
mse_test=sum((outputs_tb_test - nc_test_output).^2)/length(nc_test_output);

%-------------------------------------------------------------------------
% calculate the correlation coefficients for the training and test data
% sets with the associated linear fits hint: check out the function corrcoef
R_train = corrcoef(outputs_tb_train,nc_train_output);
R_test = corrcoef(outputs_tb_test,nc_test_output);
r_train=R_train(1,2);
```

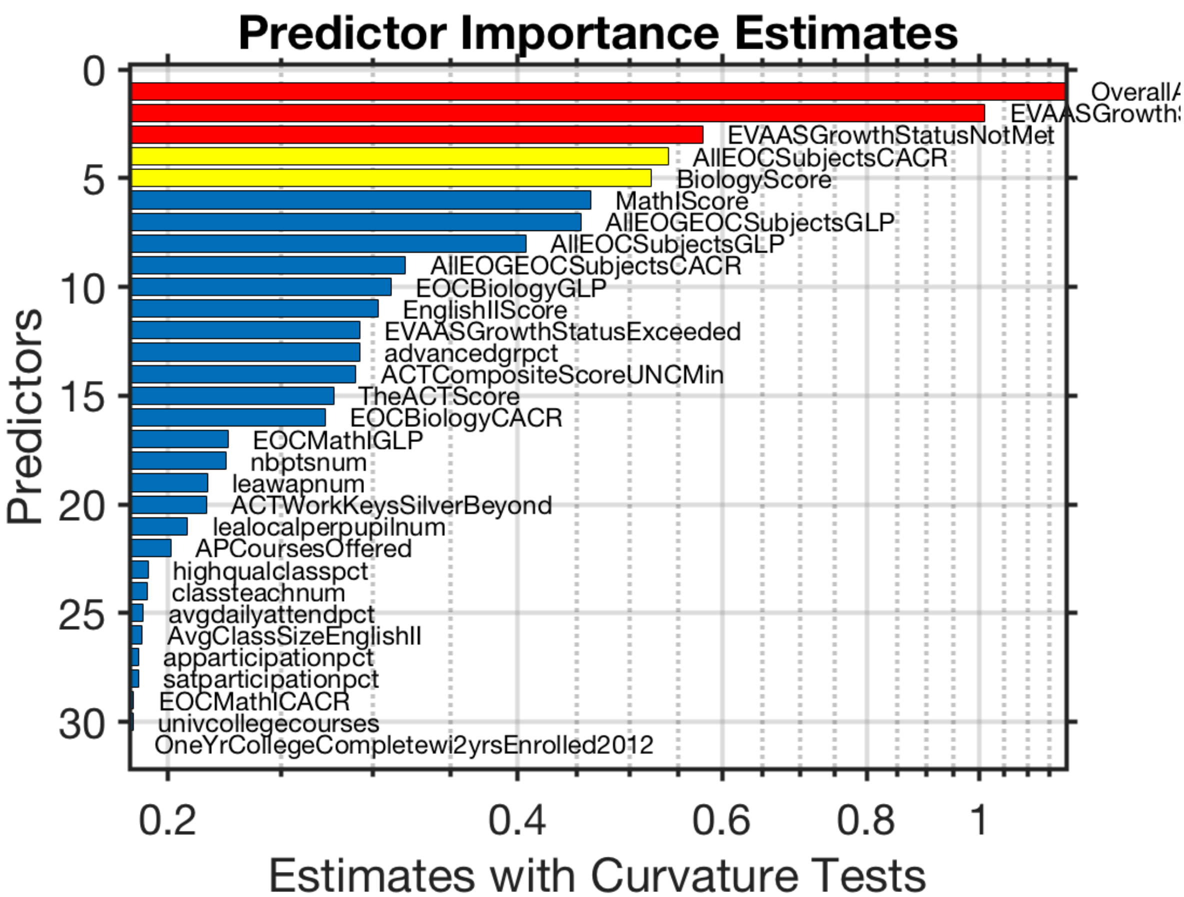
```
r_test=R_test(1,2);
```

```
% plot scatter plot
plotScatterDiagram(nc_train_output, outputs_tb_train, nc_test_output, outputs_tb_test,
```




```
%----------------------------------------------------------------------------
% Estimate the predictor importance
imp=Mdl_TB.OOBPermutedPredictorDeltaError;
[sorted_imp,isorted_imp] = sort(imp,'descend');

n = sum(imp>0);
if n  > 31
    n = 31;
end

%----------------------------------------------------------------------------
% Draw a horizontal bar chart showing the variables in descending order of
% importance. Hint: look up the function barh.
% Label each variable with its name.
% Hints: (1) Look up the function text. (2) Variable names are held in
% Mdl.PredictorNames
figure;barh(imp(isorted_imp(1:n)));hold on;grid on;
barh(imp(isorted_imp(1:5)),'y');barh(imp(isorted_imp(1:3)),'r');
```

```
title('Predictor Importance Estimates');
xlabel('Estimates with Curvature Tests');ylabel('Predictors');
set(gca,'FontSize',20); set(gca,'TickDir','out'); set(gca,'LineWidth',2);
ax = gca;ax.YDir='reverse';ax.XScale = 'log';

sorted_predictor_names = Mdl_TB.PredictorNames(isorted_imp(1:n));

% label the bars
for i=1:length(sorted_predictor_names)
   text(...
        1.05*imp(isorted_imp(i)),i,...
        strrep(sorted_predictor_names{i},'_',''),...
        'FontSize',12 ...
    )
end
print('-dpng','NC-full-input-importance.png');% save to an png file
```




```
for col=sorted_predictor_names
disp(col)
end
```

```
'OverallAchievementScore'

'EVAASGrowthScore'
```

```
'EVAASGrowthStatus_NotMet'

'All_EOC_Subjects_CACR'

'BiologyScore'

'MathIScore'

'All_EOG_EOC_Subjects_GLP'

'All_EOC_Subjects_GLP'

'All_EOG_EOC_Subjects_CACR'

'EOC_Biology_GLP'

'EnglishIIScore'

'EVAASGrowthStatus_Exceeded'

'advance_dgr_pct'

'ACT_Composite_Score_UNC_Min'

'TheACTScore'

'EOC_Biology_CACR'

'EOC_Math_I_GLP'

'nbpts_num'

'lea_wap_num'

'ACT_WorkKeys_Silver_Beyond'

'lea_local_perpupil_num'

'AP_Courses_Offered'

'highqual_class_pct'

'class_teach_num'

'avg_daily_attend_pct'

'Avg_Class_Size_EnglishII'

'ap_participation_pct'

'sat_participation_pct'

'EOC_Math_I_CACR'

'univ_college_courses'

'One_Yr_College_Complete_wi_2_yrs_Enrolled_2012'
```

TreeBagger Occum's Razor (Reduced Model)

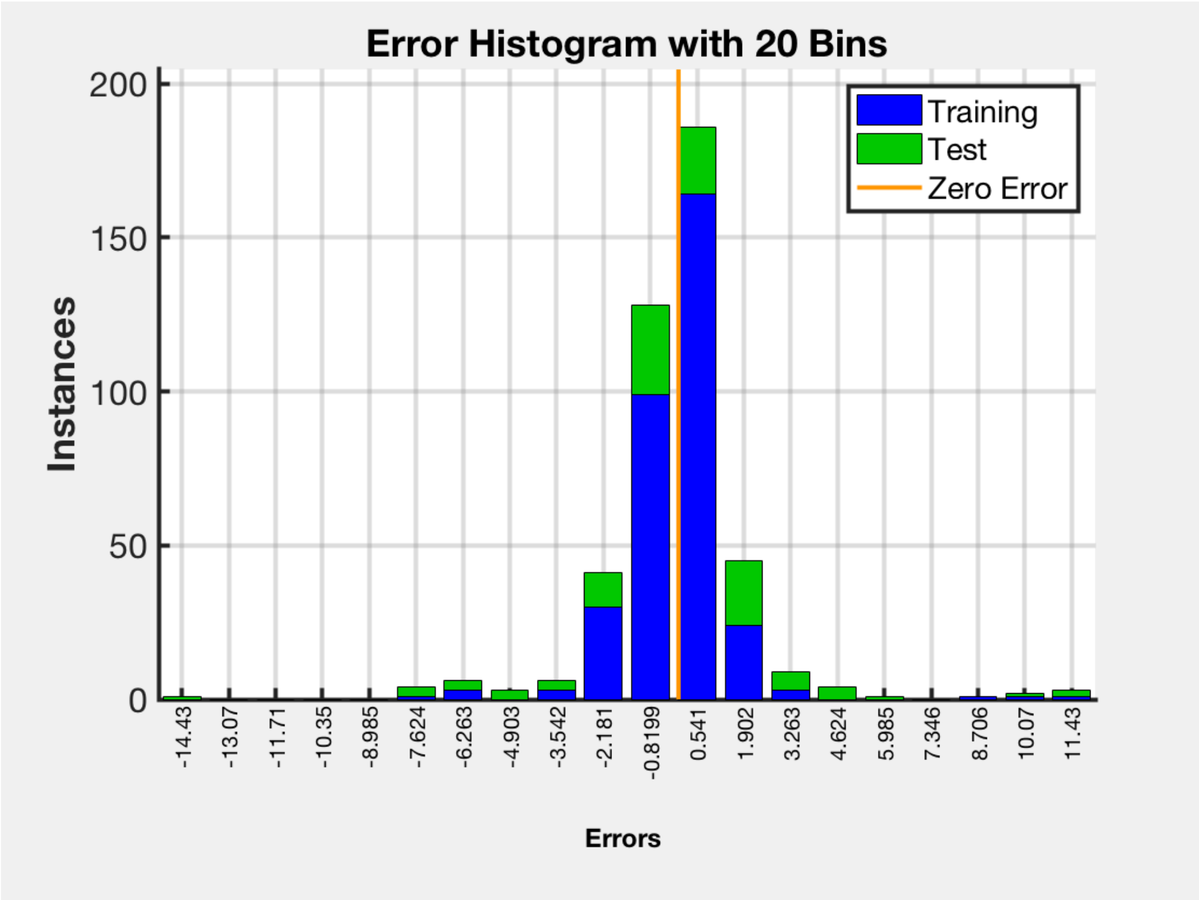
```
% Model using the best predictors
```

```
nc_train_input_simpler = nc_train_input(:, isorted_imp(1:n));
nc_test_input_simpler = nc_test_input(:, isorted_imp(1:n));

% set random seed.
rng(1);
```

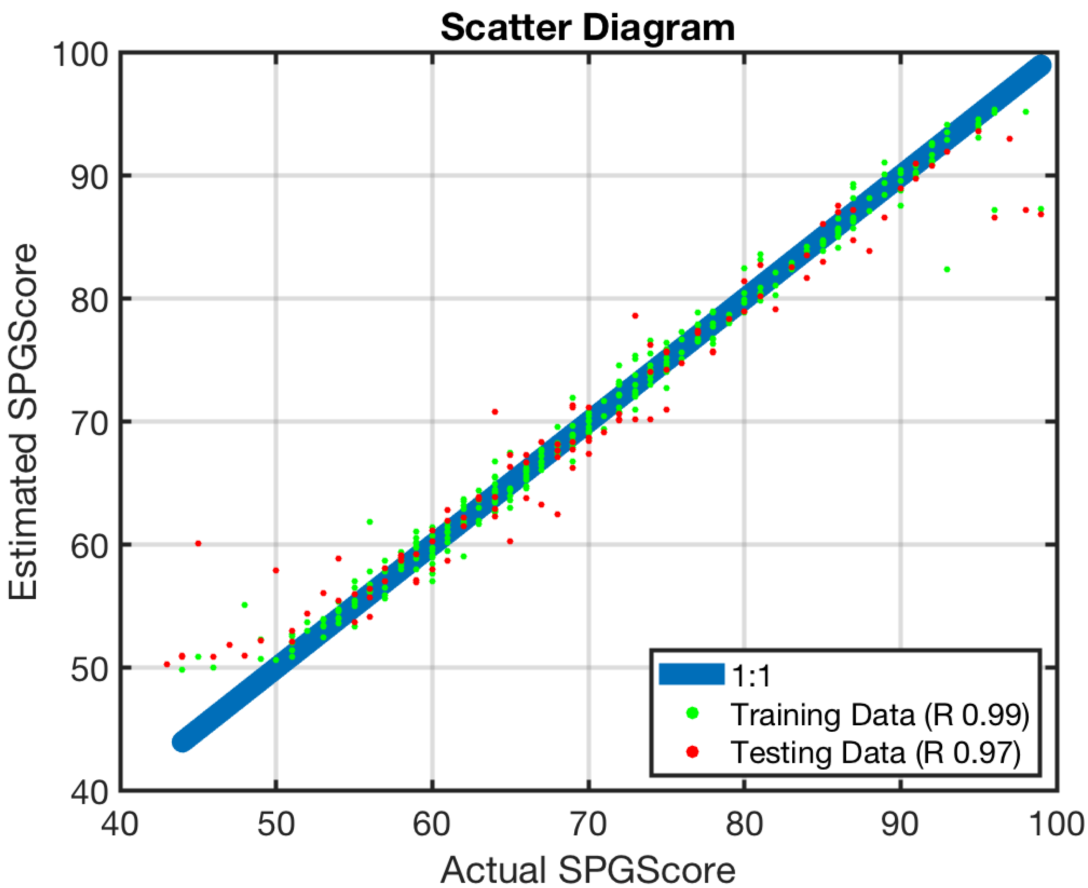
```
Mdl_TB_simpler = TreeBagger(...
    100,nc_train_input_simpler,nc_train_output,...
    'Method','Regression',...
    'Surrogate','on',...
    'PredictorSelection','curvature',...
    'OOBPredictorImportance','on'...
    )
```

```
Mdl_TB_simpler = 
  TreeBagger
Ensemble with 100 bagged decision trees:
                    Training X:             [330x31]
                    Training Y:              [330x1]
                        Method:           regression
                 NumPredictors:                   31
         NumPredictorsToSample:                   11
                   MinLeafSize:                    5
                 InBagFraction:                    1
         SampleWithReplacement:                    1
          ComputeOOBPrediction:                    1
 ComputeOOBPredictorImportance:                    1
                     Proximity:                   []

  Properties, Methods
```

```
% predict
outputs_tb_train_simpler=predict(Mdl_TB_simpler, nc_train_input_simpler);
outputs_tb_test_simpler=predict(Mdl_TB_simpler,nc_test_input_simpler);
% plot error histogram
plotErrorHistogram(nc_train_output, outputs_tb_train_simpler, nc_test_output, outputs_t
```



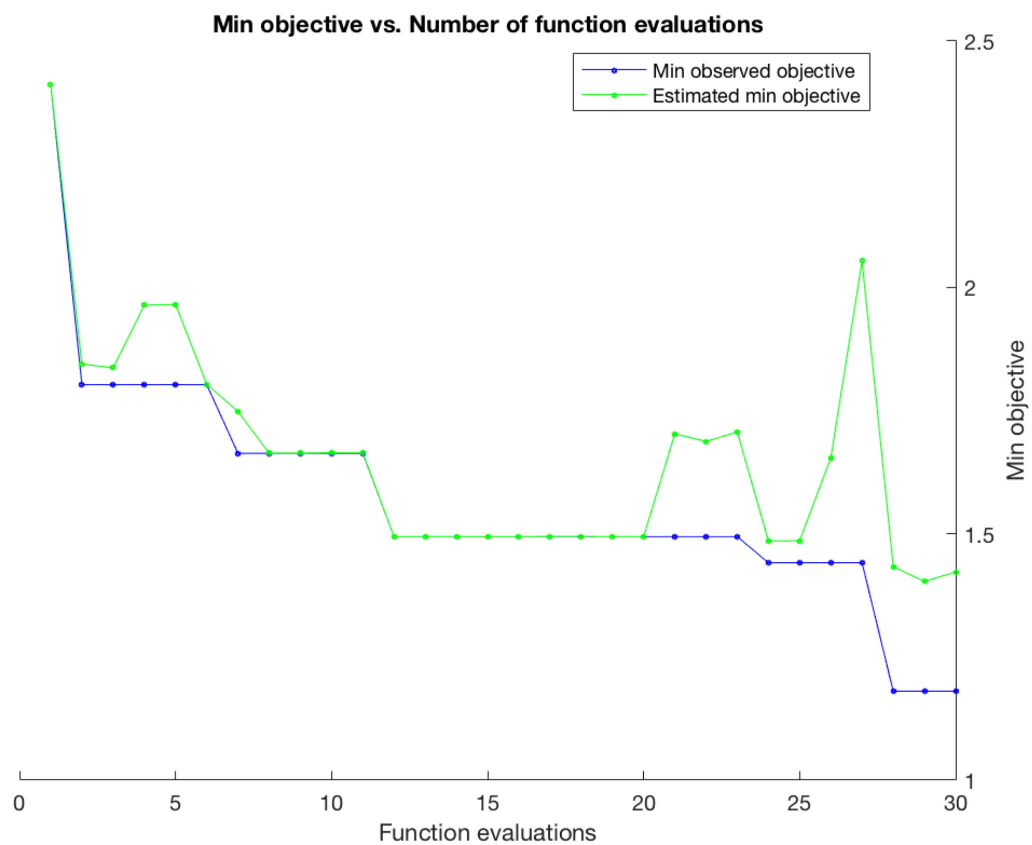

```
%-----------------------------------------------------------------------------
% calculate the mean square error (MSE) of the test points
mse_train=sum((outputs_tb_train_simpler - nc_train_output).^2)/length(nc_train_output)
mse_test=sum((outputs_tb_test_simpler - nc_test_output).^2)/length(nc_test_output);

%-----------------------------------------------------------------------------
% calculate the correlation coefficients for the training and test data
% sets with the associated linear fits hint: check out the function corrcoef
R_train = corrcoef(outputs_tb_train_simpler,nc_train_output);
R_test = corrcoef(outputs_tb_test_simpler,nc_test_output);
r_train=R_train(1,2);
r_test=R_test(1,2);
% plot scatter plot
plotScatterDiagram(nc_train_output, outputs_tb_train_simpler, nc_test_output, outputs_t
```



Ensemble Regression Models (Boosting)

```
%-----------------------------------------------------------------------
% Create ensemble model with Hyperparameter Optimization
Mdl_en = fitrensemble(...
    nc_train_input,nc_train_output,...
    'OptimizeHyperparameters','all', ...
    'HyperparameterOptimizationOptions',struct('UseParallel',true,'ShowPlots',true) ..
    )
```

```
Starting parallel pool (parpool) using the 'local' profile ...
connected to 2 workers.
Copying objective function to workers...
Done copying objective function to workers.
```



Iter	Active workers	Eval result	Objective	Objective runtime	BestSoFar (observed)	BestSoFar (estim.)	Method	NumLea ycles
1	2	Best	2.4115	22.223	2.4115	2.4115	Bag	
2	2	Best	1.8026	8.0278	1.8026	1.8446	LSBoost	
3	2	Accept	1.9823	13.469	1.8026	1.8366	Bag	
4	2	Accept	3.0084	3.4441	1.8026	1.964	LSBoost	
5	2	Accept	2.7754	3.1254	1.8026	1.9653	LSBoost	
6	2	Accept	2.6873	98.941	1.8026	1.8035	LSBoost	
7	2	Best	1.6633	8.2984	1.6633	1.7482	LSBoost	
8	2	Accept	1.9742	7.0786	1.6633	1.6636	LSBoost	
9	2	Accept	2.7138	2.9397	1.6633	1.6637	LSBoost	
10	2	Accept	6.216	4.0317	1.6633	1.6645	LSBoost	
11	2	Accept	3.3933	5.6969	1.6633	1.6641	LSBoost	
12	2	Best	1.4942	5.685	1.4942	1.4942	LSBoost	
13	2	Accept	2.5537	1.6756	1.4942	1.4943	LSBoost	
14	2	Accept	1.8272	6.0751	1.4942	1.4943	LSBoost	
15	2	Accept	4.3671	1.218	1.4942	1.4943	LSBoost	
16	2	Accept	3.0933	1.442	1.4942	1.4943	LSBoost	
17	2	Accept	8.5246	1.7099	1.4942	1.4946	LSBoost	
18	2	Accept	8.4351	19.44	1.4942	1.4948	LSBoost	
19	2	Accept	5.0192	18.636	1.4942	1.4941	LSBoost	
20	2	Accept	4.6614	3.7141	1.4942	1.4944	LSBoost	

Iter	Active workers	Eval result	Objective	Objective runtime	BestSoFar (observed)	BestSoFar (estim.)	Method	NumLea ycles
21	2	Accept	2.2539	2.1228	1.4942	1.7029	LSBoost	
22	2	Accept	2.3998	2.3081	1.4942	1.6873	LSBoost	
23	2	Accept	2.6492	67.642	1.4942	1.7068	Bag	
24	2	Best	1.4412	1.9298	1.4412	1.4854	LSBoost	
25	2	Accept	3.1553	1.8052	1.4412	1.4859	Bag	

```
|   26 |       2 | Accept |      3.8783 |      2.9453 |      1.4412 |      1.6543 |      LSBoost |
|   27 |       2 | Accept |      5.0213 |      1.2751 |      1.4412 |      2.0535 |          Bag |
|   28 |       2 | Best   |      1.1807 |      9.1691 |      1.1807 |      1.4332 |      LSBoost |
|   29 |       2 | Accept |      2.8595 |       1.558 |      1.1807 |      1.4034 |      LSBoost |
|   30 |       2 | Accept |       2.455 |      1.7831 |      1.1807 |      1.4225 |      LSBoost |

__________________________________________________________
Optimization completed.
MaxObjectiveEvaluations of 30 reached.
Total function evaluations: 30
Total elapsed time: 473.4118 seconds.
Total objective function evaluation time: 329.4098

Best observed feasible point:
```

Method	NumLearningCycles	LearnRate	MinLeafSize	MaxNumSplits	NumVariablesToSample
LSBoost	33	0.44255	1	10	136

```
Observed objective function value = 1.1807
Estimated objective function value = 1.4225
Function evaluation time = 9.1691

Best estimated feasible point (according to models):
```

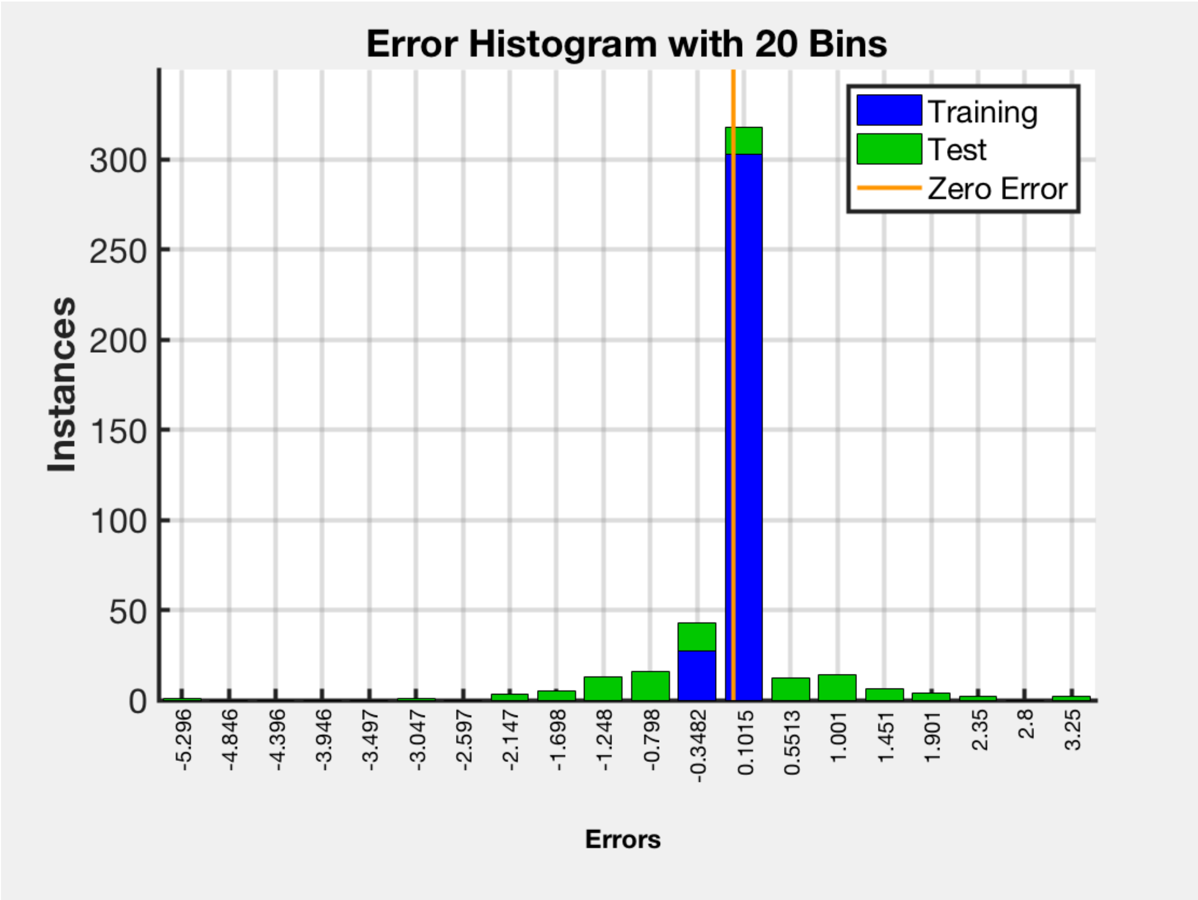
Method	NumLearningCycles	LearnRate	MinLeafSize	MaxNumSplits	NumVariablesToSample
LSBoost	33	0.44255	1	10	136

```
Estimated objective function value = 1.4225
Estimated function evaluation time = 9.0838
```

```
Mdl_en = 
  classreg.learning.regr.RegressionEnsemble
                       ResponseName: 'Y'
              CategoricalPredictors: []
                  ResponseTransform: 'none'
                    NumObservations: 330
    HyperparameterOptimizationResults: [1×1 BayesianOptimization]
                         NumTrained: 33
                             Method: 'LSBoost'
                       LearnerNames: {'Tree'}
               ReasonForTermination: 'Terminated normally after completing the requested number of trai
                            FitInfo: [33×1 double]
                 FitInfoDescription: {2×1 cell}
                     Regularization: []


  Properties, Methods
```

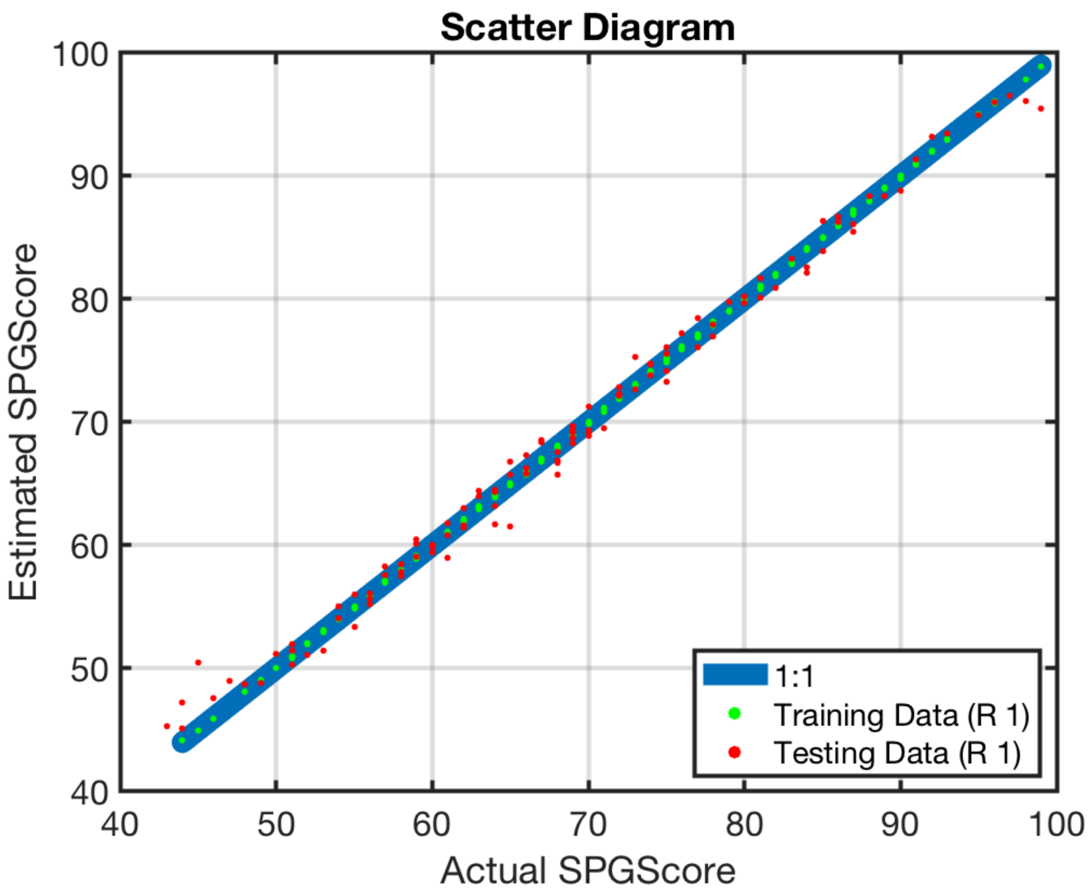
```
% predict
outputs_en_train=predict(Mdl_en, nc_train_input);
outputs_en_test=predict(Mdl_en,nc_test_input);
```

```
% plot error histogram
plotErrorHistogram(nc_train_output, outputs_en_train, nc_test_output, outputs_en_test)
```




```
%----------------------------------------------------------------------------
% calculate the mean square error (MSE) of the test points
mse_train=sum((outputs_en_train - nc_train_output).^2)/length(nc_train_output);
mse_test=sum((outputs_en_test - nc_test_output).^2)/length(nc_test_output);

%----------------------------------------------------------------------------
% calculate the correlation coefficients for the training and test data
% sets with the associated linear fits hint: check out the function corrcoef
R_train = corrcoef(outputs_en_train,nc_train_output);
R_test = corrcoef(outputs_en_test,nc_test_output);
r_train=R_train(1,2);
r_test=R_test(1,2);
```

```
% plot scatter plot
plotScatterDiagram(nc_train_output, outputs_en_train, nc_test_output, outputs_en_test,
```

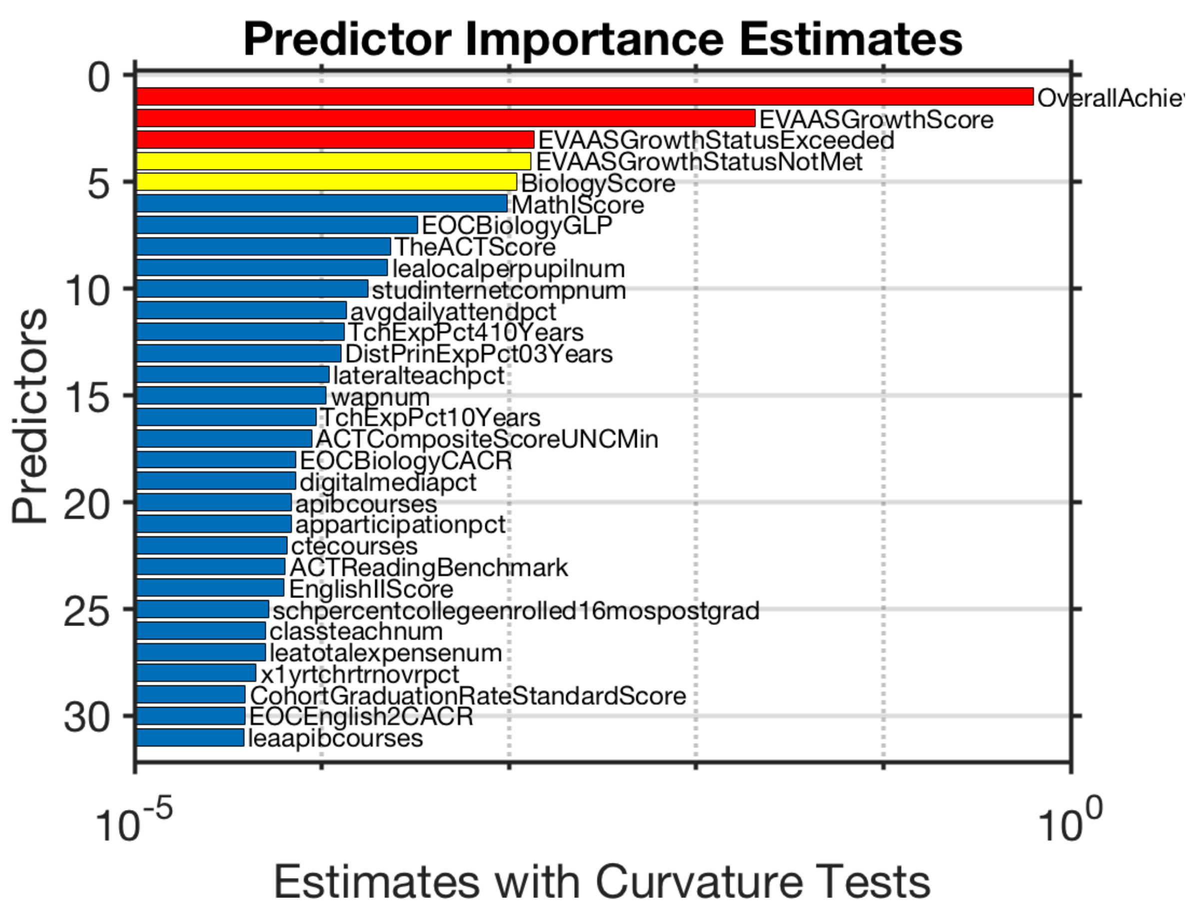



```
%------------------------------------------------------------------------
% Estimate the predictor importance
imp=predictorImportance(Mdl_en)
```

```
imp = 1×141
    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000         0 ···
```

```
[sorted_imp,isorted_imp] = sort(imp,'descend');

n = sum(imp>0);
if n  > 31
    n = 31;
end

%------------------------------------------------------------------------
% Draw a horizontal bar chart showing the variables in descending order of
% importance. Hint: look up the function barh.
% Label each variable with its name.
% Hints: (1) Look up the function text. (2) Variable names are held in
% Mdl.PredictorNames
figure;barh(imp(isorted_imp(1:n)));hold on;grid on;
barh(imp(isorted_imp(1:5)),'y');barh(imp(isorted_imp(1:3)),'r');
title('Predictor Importance Estimates');
xlabel('Estimates with Curvature Tests');ylabel('Predictors');
```

```
set(gca,'FontSize',20); set(gca,'TickDir','out'); set(gca,'LineWidth',2);
ax = gca;ax.YDir='reverse';ax.XScale = 'log';

sorted_predictor_names = Mdl_en.PredictorNames(isorted_imp(1:n));

% label the bars
for i=1:length(sorted_predictor_names)
   text(...
        1.05*imp(isorted_imp(i)),i,...
        strrep(sorted_predictor_names{i},'_',''),...
        'FontSize',12 ...
     )
end
print('-dpng','NC-full-input-importance.png');% save to an png file
```




```
for col=sorted_predictor_names
disp(col)
end
```

```
    'OverallAchievementScore'

    'EVAASGrowthScore'

    'EVAASGrowthStatus_Exceeded'
```

```
'EVAASGrowthStatus_NotMet'
'BiologyScore'
'MathIScore'
'EOC_Biology_GLP'
'TheACTScore'
'lea_local_perpupil_num'
'stud_internet_comp_num'
'avg_daily_attend_pct'
'Tch_Exp_Pct_4_10_Years'
'Dist_Prin_Exp_Pct_0_3_Years'
'lateral_teach_pct'
'wap_num'
'Tch_Exp_Pct_10__Years'
'ACT_Composite_Score_UNC_Min'
'EOC_Biology_CACR'
'digital_media_pct'
'ap_ib_courses'
'ap_participation_pct'
'cte_courses'
'ACT_Reading_Benchmark'
'EnglishIIScore'
'sch_percent_college_enrolled_16_mos_post_grad'
'class_teach_num'
'lea_total_expense_num'
'x_1yr_tchr_trnovr_pct'
'CohortGraduationRateStandardScore'
'EOC_English_2_CACR'
'lea_ap_ib_courses'
```

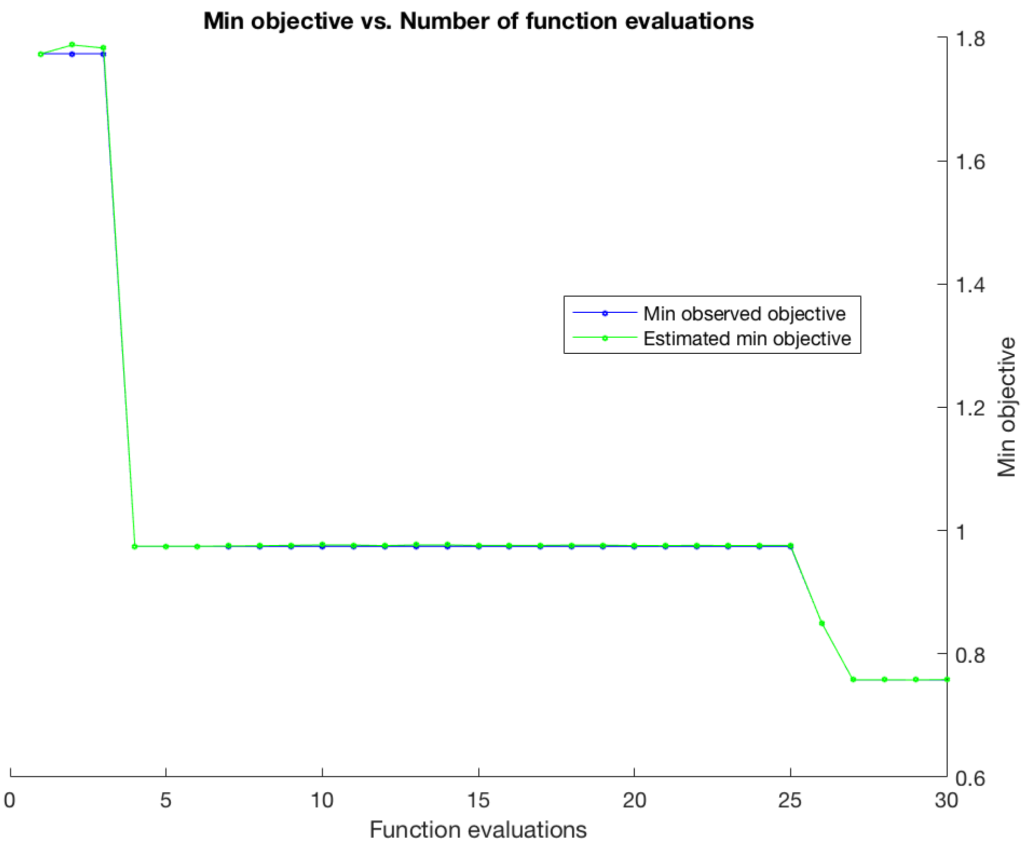
Ensemble Model Regression with simpler model

```
% Model using the best predictors
nc_train_input_simpler = nc_train_input(:, isorted_imp(1:n));
nc_test_input_simpler = nc_test_input(:, isorted_imp(1:n));
```

```
% set random seed.
rng(1);
```

```
%-------------------------------------------------------------------------------
% Create ensemble model with Hyperparameter Optimization
Mdl_en_simpler = fitrensemble(...
    nc_train_input_simpler,nc_train_output,...
    'OptimizeHyperparameters','all', ...
    'HyperparameterOptimizationOptions',struct('UseParallel',true,'ShowPlots',true) ..
    )
```

```
Starting parallel pool (parpool) using the 'local' profile ...
connected to 2 workers.
Copying objective function to workers...
Done copying objective function to workers.
```




```
|=====================================================================================================
| Iter | Active  | Eval   | Objective   | Objective   | BestSoFar   | BestSoFar   |       Method | NumLea
|      | workers | result |             | runtime     | (observed)  | (estim.)    |              | ycles
|=====================================================================================================
|    1 |       2 | Best   |      1.7737 |      29.783 |      1.7737 |      1.7737 |          Bag |
|    2 |       2 | Accept |      1.9479 |      41.618 |      1.7737 |      1.7879 |      LSBoost |
|    3 |       2 | Accept |      1.8419 |      13.014 |      1.7737 |      1.7828 |      LSBoost |
|    4 |       2 | Best   |     0.97454 |      13.203 |     0.97454 |     0.97462 |      LSBoost |
|    5 |       2 | Accept |       1.453 |      4.5573 |     0.97454 |     0.97463 |      LSBoost |
|    6 |       2 | Accept |      1.5865 |      24.173 |     0.97454 |     0.97461 |          Bag |
|    7 |       2 | Accept |      6.8301 |      1.4458 |     0.97454 |     0.97495 |      LSBoost |
|    8 |       2 | Accept |      7.2642 |       2.673 |     0.97454 |     0.97545 |      LSBoost |
```

```
|    9 |       2 | Accept |      8.4332 |      1.3793 |     0.97454 |     0.97614 |      LSBoost |
|   10 |       2 | Accept |      1.3718 |       9.482 |     0.97454 |     0.97674 |      LSBoost |
|   11 |       2 | Accept |      1.8762 |      1.9733 |     0.97454 |     0.97651 |          Bag |
|   12 |       2 | Accept |      3.1512 |      1.7664 |     0.97454 |     0.97568 |      LSBoost |
|   13 |       2 | Accept |      1.5507 |      9.7235 |     0.97454 |     0.97668 |      LSBoost |
|   14 |       2 | Accept |      1.5351 |      36.595 |     0.97454 |     0.97672 |      LSBoost |
|   15 |       2 | Accept |      1.5091 |      4.0467 |     0.97454 |     0.97592 |          Bag |
|   16 |       2 | Accept |      2.6128 |      3.5955 |     0.97454 |     0.97593 |      LSBoost |
|   17 |       2 | Accept |      2.7596 |      4.8166 |     0.97454 |     0.97602 |      LSBoost |
|   18 |       2 | Accept |      1.0087 |      11.093 |     0.97454 |     0.97649 |      LSBoost |
|   19 |       2 | Accept |      5.0162 |      1.5039 |     0.97454 |     0.97642 |          Bag |
|   20 |       2 | Accept |      3.4222 |      1.8638 |     0.97454 |     0.97573 |      LSBoost |
|=====================================================================================================================
| Iter | Active  | Eval   | Objective   | Objective   | BestSoFar   | BestSoFar   |       Method | NumLea
|      | workers | result |             | runtime     | (observed)  | (estim.)    |              | ycles
|=====================================================================================================================
|   21 |       2 | Accept |      2.2698 |      3.4061 |     0.97454 |     0.97564 |      LSBoost |
|   22 |       2 | Accept |      2.3784 |      18.928 |     0.97454 |     0.97611 |      LSBoost |
|   23 |       2 | Accept |      3.6818 |      5.3271 |     0.97454 |     0.97567 |          Bag |
|   24 |       2 | Accept |      2.2552 |      3.6487 |     0.97454 |     0.97587 |          Bag |
|   25 |       2 | Accept |      1.1478 |      4.5434 |     0.97454 |     0.97599 |          Bag |
|   26 |       2 | Best   |     0.85042 |      18.429 |     0.85042 |     0.85022 |      LSBoost |
|   27 |       2 | Best   |      0.7577 |      8.6086 |      0.7577 |     0.75785 |      LSBoost |
|   28 |       2 | Accept |     0.83167 |      17.924 |      0.7577 |     0.75786 |      LSBoost |
|   29 |       2 | Accept |     0.91315 |      8.1308 |      0.7577 |     0.75774 |      LSBoost |
|   30 |       2 | Accept |      2.2709 |      4.8539 |      0.7577 |     0.75831 |          Bag |

__________________________________________________________
Optimization completed.
MaxObjectiveEvaluations of 30 reached.
Total function evaluations: 30
Total elapsed time: 476.1317 seconds.
Total objective function evaluation time: 312.1063

Best observed feasible point:
```

Method	NumLearningCycles	LearnRate	MinLeafSize	MaxNumSplits	NumVariablesToSample
LSBoost	54	0.2333	2	15	31

```
Observed objective function value = 0.7577
Estimated objective function value = 0.75831
Function evaluation time = 8.6086

Best estimated feasible point (according to models):
```

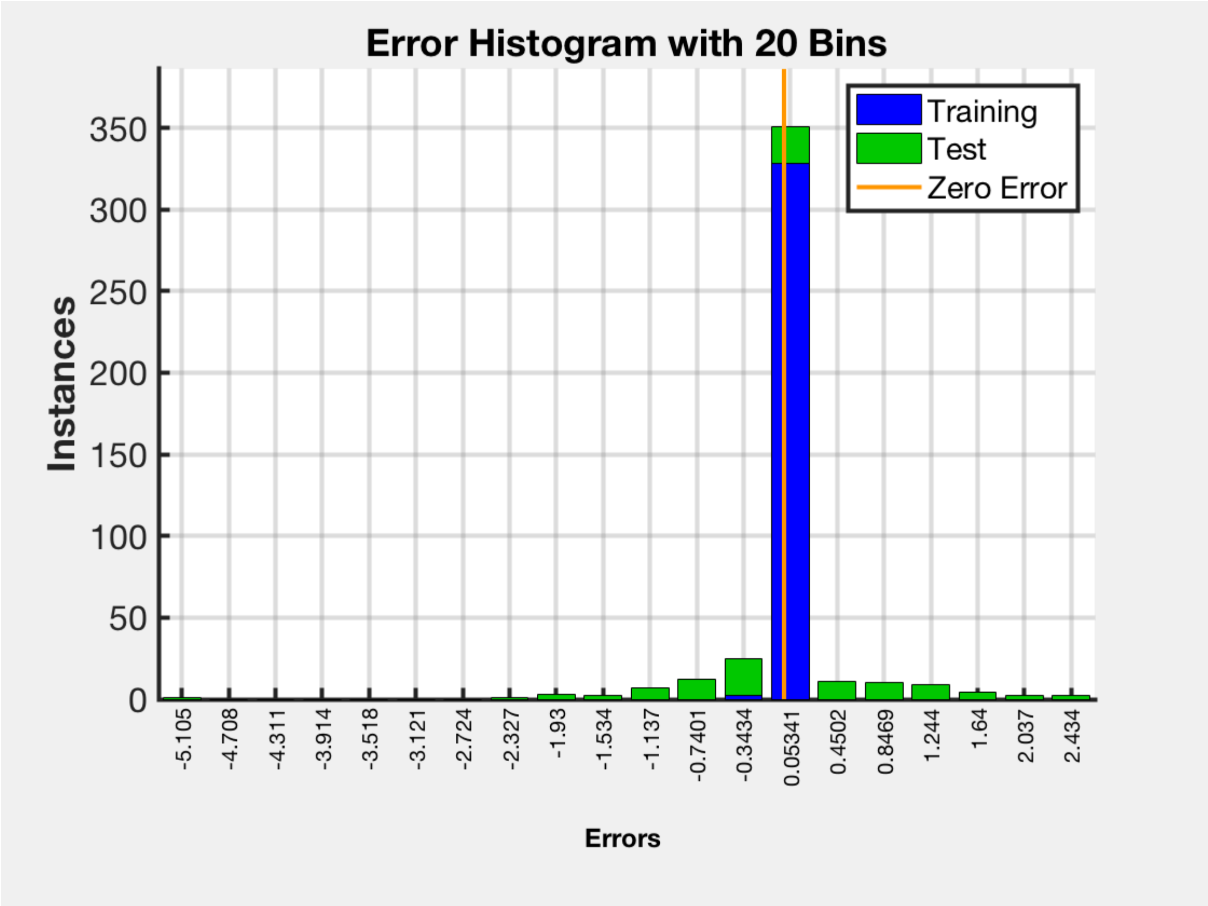
Method	NumLearningCycles	LearnRate	MinLeafSize	MaxNumSplits	NumVariablesToSample
LSBoost	54	0.2333	2	15	31

```
Estimated objective function value = 0.75831
Estimated function evaluation time = 8.5592

Mdl_en_simpler = 
  classreg.learning.regr.RegressionEnsemble
                       PredictorNames: {1×31 cell}
                         ResponseName: 'Y'
                CategoricalPredictors: []
                    ResponseTransform: 'none'
                      NumObservations: 330
    HyperparameterOptimizationResults: [1×1 BayesianOptimization]
                           NumTrained: 54
                               Method: 'LSBoost'
                         LearnerNames: {'Tree'}
                 ReasonForTermination: 'Terminated normally after completing the requested number of trai
```

```
                FitInfo: [54×1 double]
     FitInfoDescription: {2×1 cell}
         Regularization: []

  Properties, Methods
```

```
% predict
outputs_en_train_simpler=predict(Mdl_en_simpler, nc_train_input_simpler);
outputs_en_test_simpler=predict(Mdl_en_simpler,nc_test_input_simpler);
```

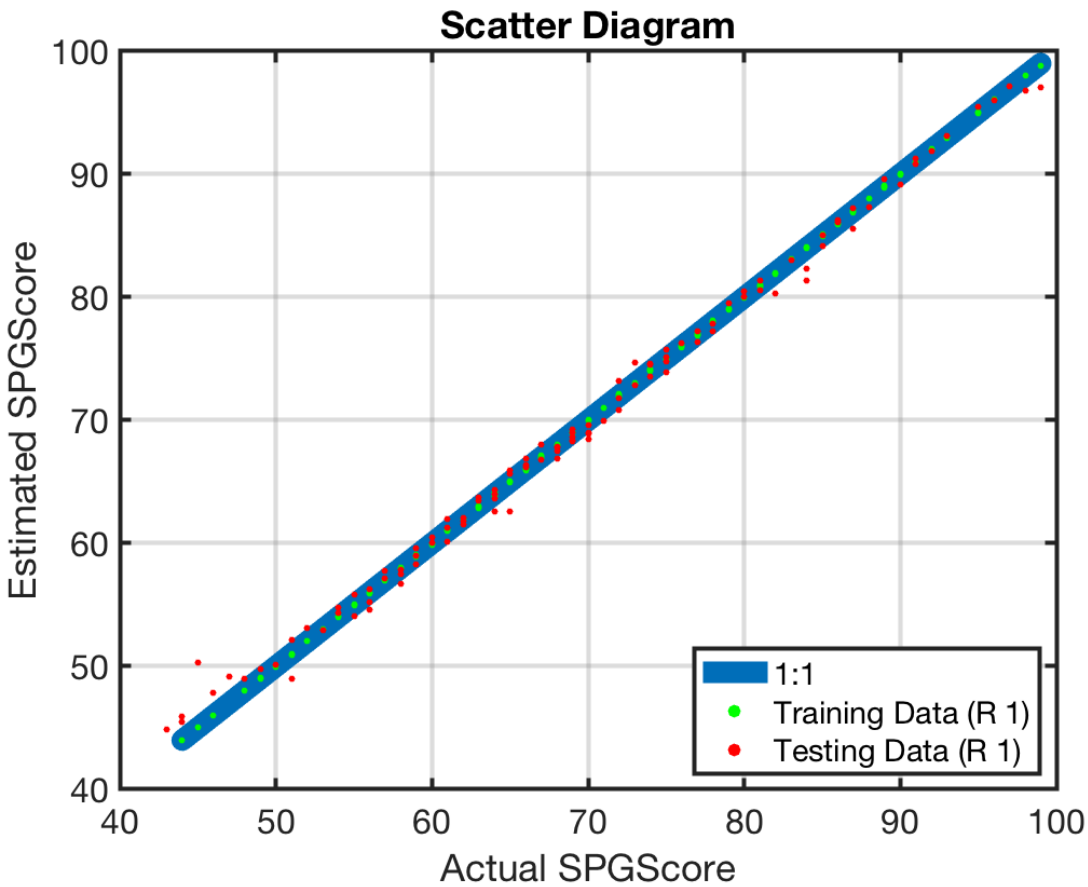
```
% plot error histogram
plotErrorHistogram(nc_train_output, outputs_en_train_simpler, nc_test_output, outputs_e
```




```
%----------------------------------------------------------------------------
% calculate the mean square error (MSE) of the test points
mse_train=sum((outputs_en_train_simpler - nc_train_output).^2)/length(nc_train_output)
mse_test=sum((outputs_en_test_simpler - nc_test_output).^2)/length(nc_test_output);

%----------------------------------------------------------------------------
% calculate the correlation coefficients for the training and test data
% sets with the associated linear fits hint: check out the function corrcoef
```

```
R_train = corrcoef(outputs_en_train_simpler,nc_train_output);
R_test = corrcoef(outputs_en_test_simpler,nc_test_output);
r_train=R_train(1,2);
r_test=R_test(1,2);
```

```
% plot scatter plot
plotScatterDiagram(nc_train_output, outputs_en_train_simpler, nc_test_output, outputs_e
```



Neural Network Regression using Ensemble Boosting algorithm's best predictors.

```
% save the variables
% save AllRegressionLearners
% ml_data_simple_en = ml_data(:, isorted_imp(1:n));
% save Ensemble_Regression_Simple.mat ml_data_simple_en
```

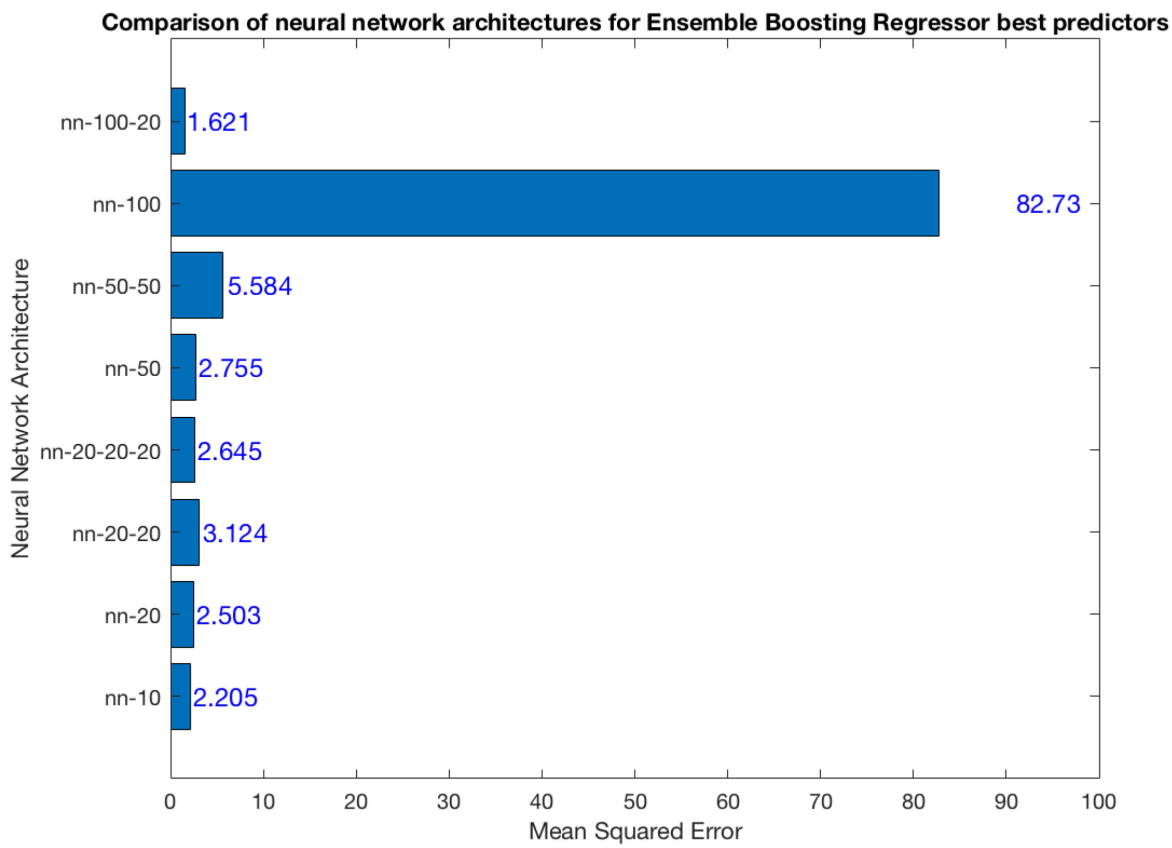
Prepare data for sending it into a neural network

```
% Commands to create Ensemble Neural Network data with simple model
clear;  close all; clc;
```

```
load Ensemble_Regression_Simple.mat
target = ml_data_output;
X_EN = ml_data_simple_en{:,:};
X_EN = X_EN';
Y_EN = target;
Y_EN = Y_EN';
save nn_reg_en.mat X_EN Y_EN;
```

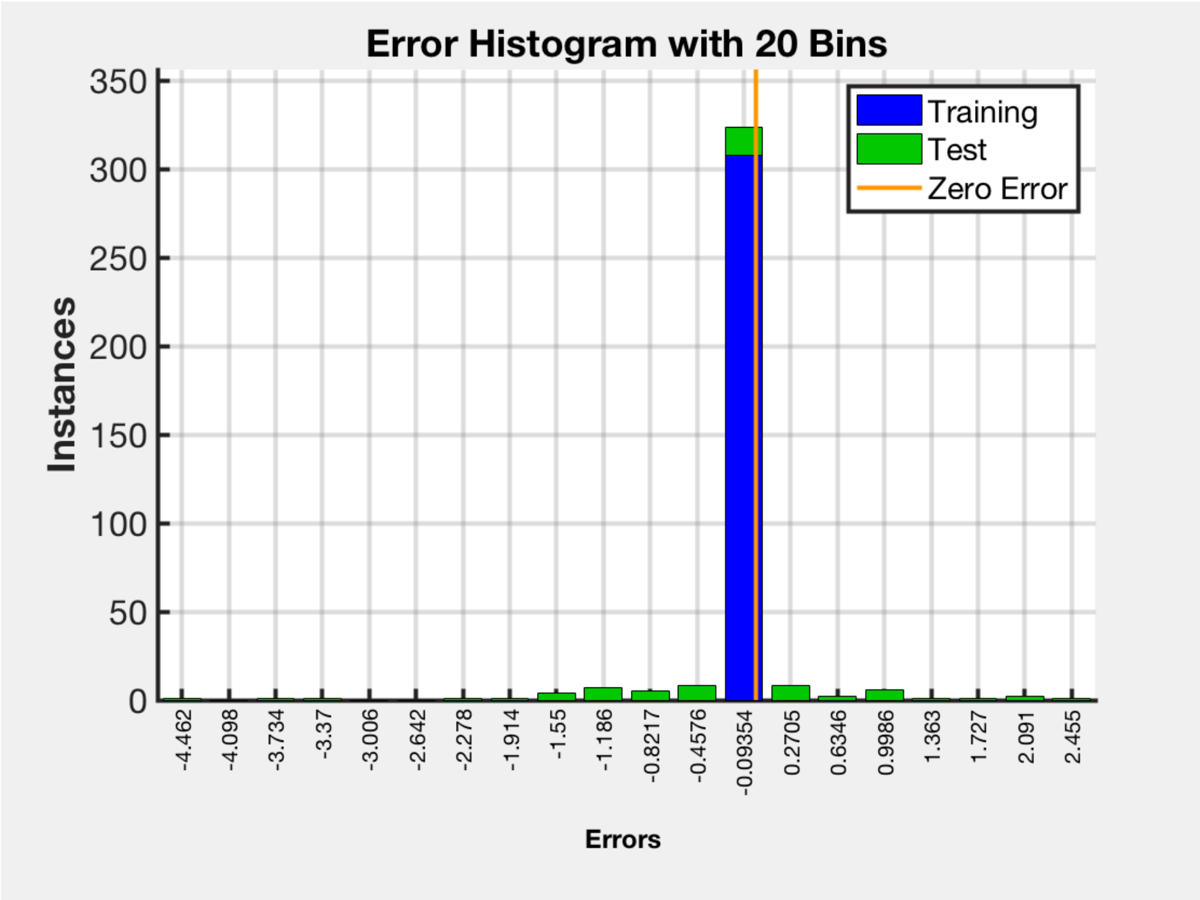
```
run('en_reg_nn.m')
```

```
Mean Squared Error for nn_10 is: 2.205103
Mean Squared Error for nn_20 is: 2.503250
Mean Squared Error for nn_20_20 is: 3.123663
Mean Squared Error for nn_20_20_20 is: 2.644605
Mean Squared Error for nn_50 is: 2.754724
Mean Squared Error for nn_50_50 is: 5.583735
Mean Squared Error for nn_100 is: 82.732977
Mean Squared Error for nn_100_20 is: 1.621472
```

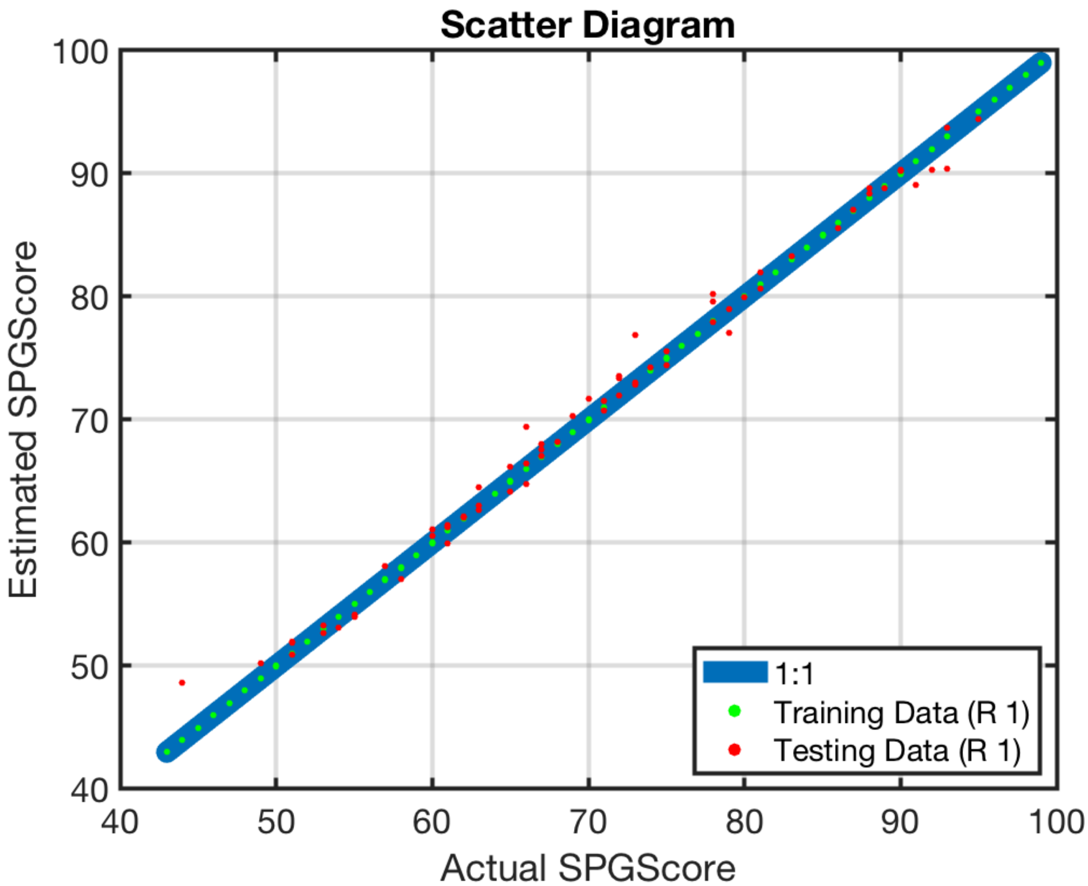


Performance of th best Neural Network Architecture using Ensemble Boosting Regressor's best predictors

```
run('en_reg_nn_best.m')
```




```
R_train = 2×2
    1.0000    1.0000
    1.0000    1.0000
R_test = 2×2
    1.0000    0.9954
    0.9954    1.0000
r_train = 1.0000
r_test = 0.9954
```




```
meanSqErr = 1.6215
Mean Squared Error for nn_100_20 is: 1.621472
```

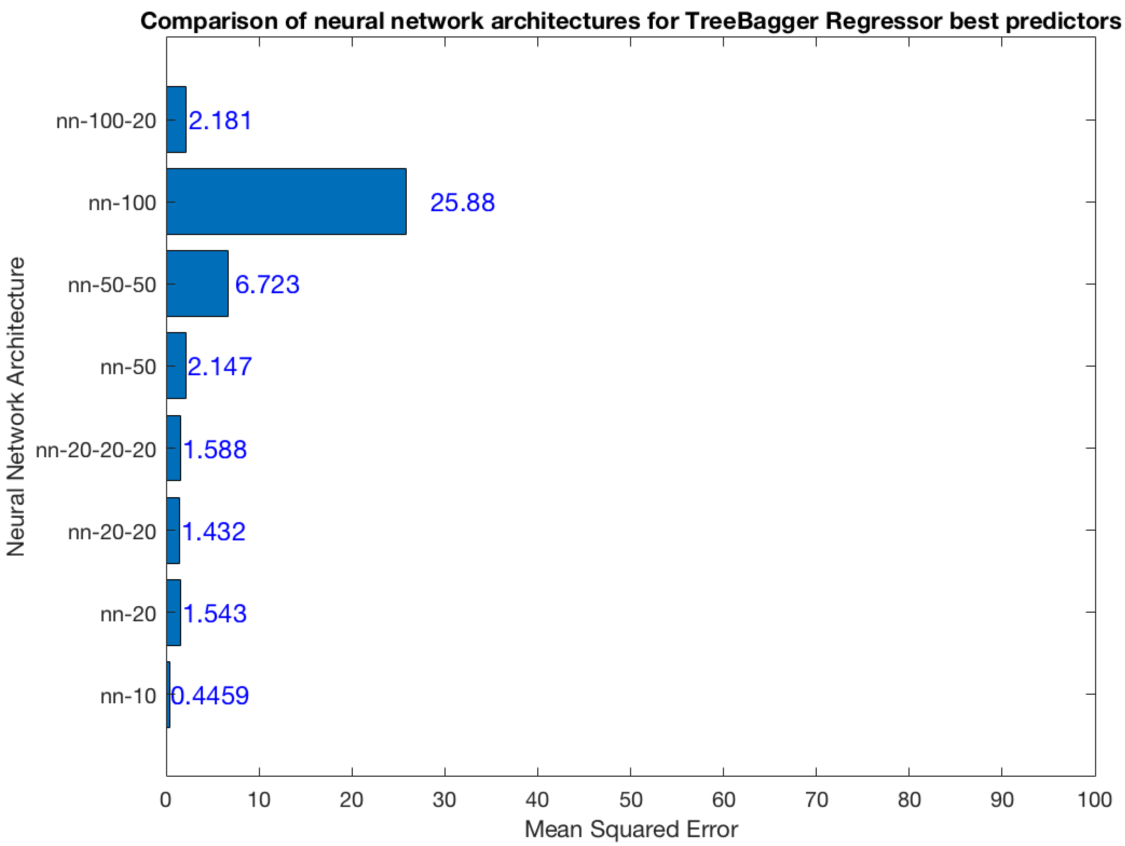
Neural Network Regression using TreeBagger algorithm's best predictors.

```
%ml_data_simple_tb = ml_data(:, isorted_imp(1:n));
%save TreeBagger_Regression_Simple.mat ml_data_simple_tb
```

```
% Commands to create Ensemble Neural Network data with simple model
clear; close all; clc;
load TreeBagger_Regression_Simple.mat;
target = ml_data_output;
X_TB = ml_data_simple_tb{:,:};
X_TB = X_TB';
Y_TB = target;
Y_TB = Y_TB';
save nn_reg_tb.mat X_TB Y_TB;
```

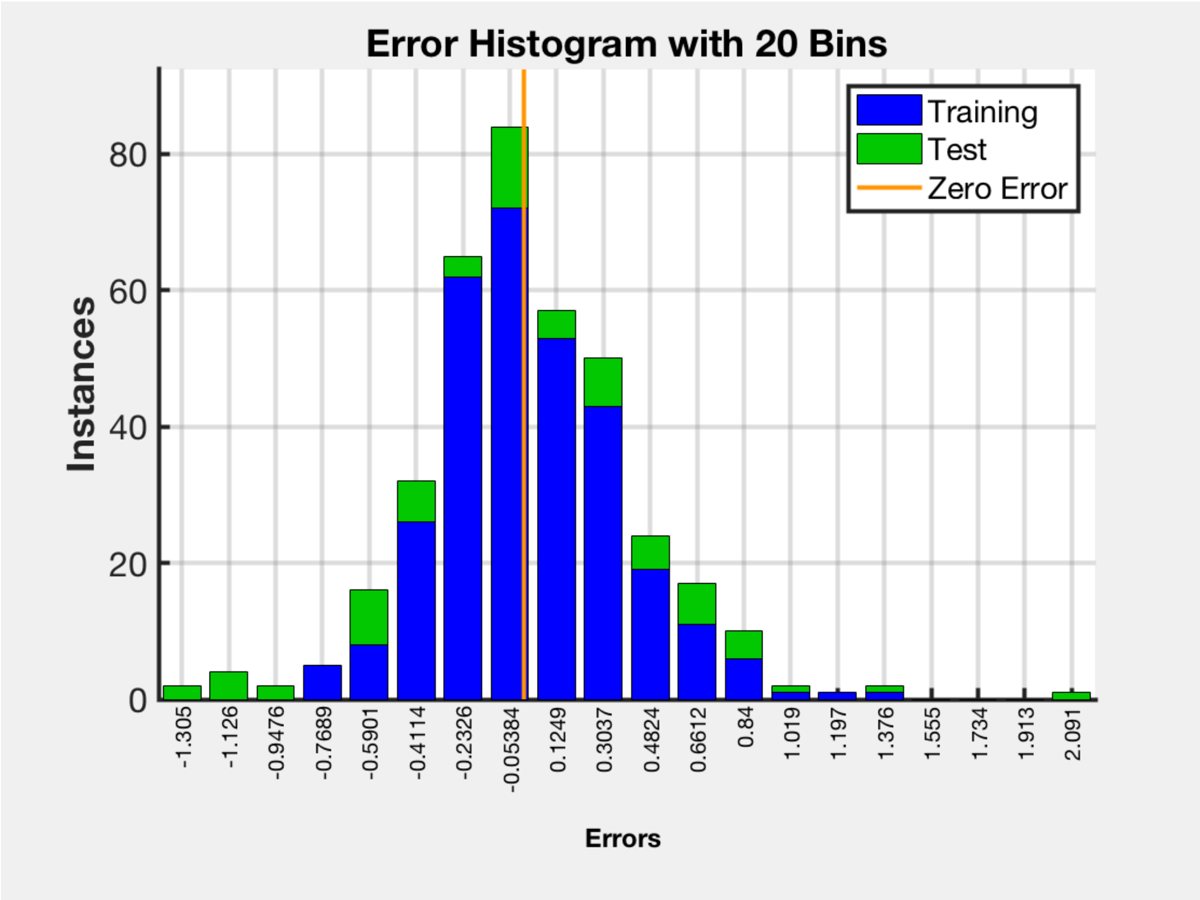
```
run('tb_reg_nn.m')
```

```
Mean Squared Error for nn_10 is: 0.445942
Mean Squared Error for nn_20 is: 1.543293
Mean Squared Error for nn_20_20 is: 1.431745
Mean Squared Error for nn_20_20_20 is: 1.587794
Mean Squared Error for nn_50 is: 2.146929
Mean Squared Error for nn_50_50 is: 6.722760
Mean Squared Error for nn_100 is: 25.879183
Mean Squared Error for nn_100_20 is: 2.180521
```

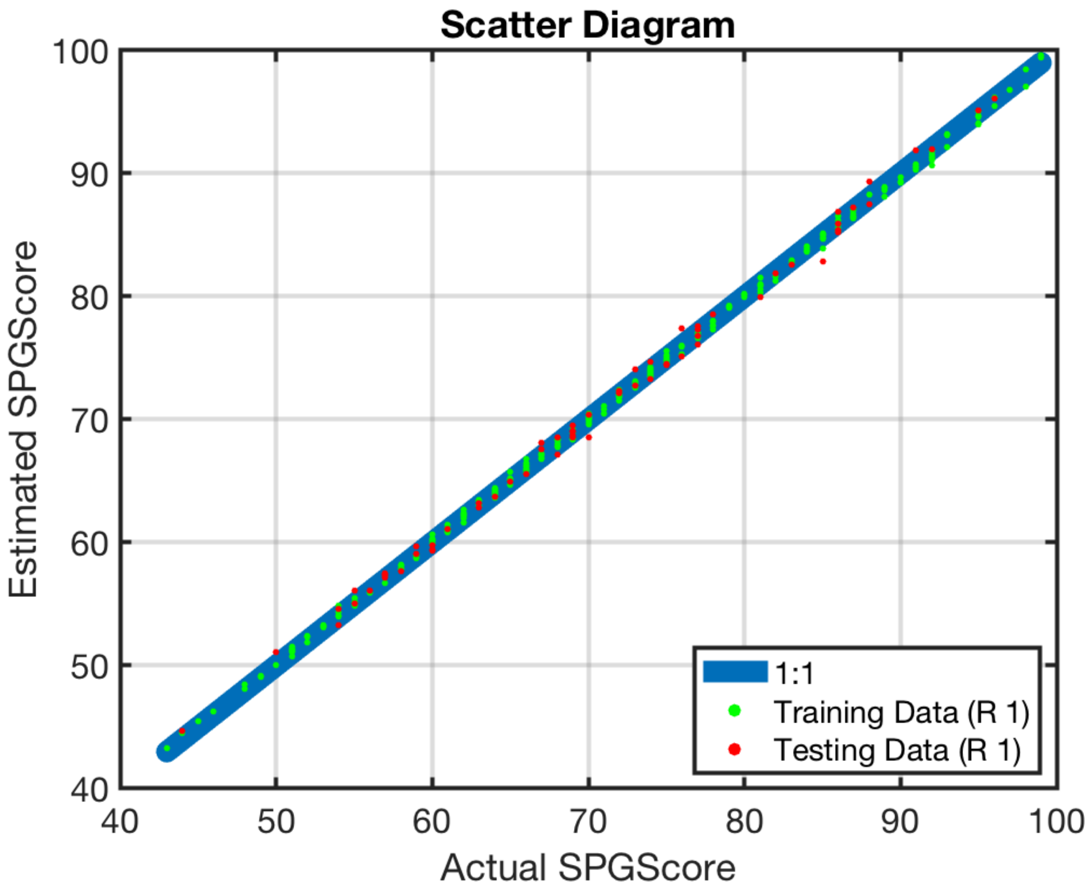


Performance of th best Neural Network Architecture using TreeBagger Regressor's best predictors

```
run('tb_reg_nn_best.m')
```




```
R_train = 2×2
    1.0000    0.9997
    0.9997    1.0000
R_test = 2×2
    1.0000    0.9984
    0.9984    1.0000
r_train = 0.9997
r_test = 0.9984
```




```
meanSqErr = 0.4459
Mean Squared Error for nn_10 is: 0.445942
```

Conclusion

•Educators, State/School officials can examine these features and recommend areas of improvement to schools.

•Important to study the feature importance over the years, a key question to answer in the coming years: do the same features appear each year or do new features play an important role in differentiating these schools ?