

Linear Regression using Machine Learning

Yi Qiang Ji Zhang
Biel Galot Pérez
Gerard Villalta Quintana
Oriol Miras Robles

Professor: Alex Ferrer Ferre

Aerospace Engineering
Numerical Tools in Machine Learning for Aeronautical Engineering
Polytechnic University of Catalonia — BarcelonaTech



UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

Escola Superior d'Enginyeries Industrial,
Aeroespacial i Audiovisual de Terrassa

Date: 11 May 2021

I. INTRODUCTION

Linear regression attempts to model the relationship between two variables by fitting a linear equation to observed data. One variable is considered to be an explanatory variable (data), and the other is considered to be a dependent variable (outcome). It is used to predict values within a continuous range rather than trying to classify them into categories [1] [2].

There are two main types:

- **Simple regression:** Simple linear regression uses traditional slope-intercept form, where m and n are the variables the algorithm will try to “learn” to produce the most accurate predictions. x represents the input data and y represents the algorithm’s prediction.

$$y = mx + n \quad (1)$$

- **Multi-variable regression:** This is a more complex linear regression. Where w_i represents the coefficients, or weights, that the algorithm model will try to “learn”.

$$f(x_i) = \sum_i^N w_i x_j \quad \forall i, j \in \mathbb{N} \quad (2)$$

and x_j represents the different variables.

II. LINEAR REGRESSION

In order to implement a linear regression predictor, as a first approach, a simple linear regression without an independent parameter n will be implemented and so the model will simply output

$$y = mx \quad (3)$$

Where m is a vector of parameters (weights) and x is the vector of input data. In order to do so, a set of data is randomly generated using the function `generateData`.

This function is set to generate a quadratic set of data with some error that is artificially added.

Secondly, 25% of the data points are reserved in order to be used to perform the test of our model, while the rest is set to feed the training of the linear regression model. This is done with `splitDataTrainAndTest`.

Once the data has been separated, the model itself is finally trained, by minimizing the MSE of the data destined to train. Attending to [3], this is done following the following procedure

$$\nabla_w \text{MSE}_{\text{train}} = 0 \quad (4)$$

$$\Rightarrow \nabla_w \frac{1}{m} \left\| \hat{\mathbf{y}}^{(\text{train})} - \mathbf{y}^{(\text{train})} \right\|_2^2 = 0 \quad (5)$$

$$\Rightarrow \frac{1}{m} \nabla_w \left\| \mathbf{X}^{(\text{train})} \mathbf{w} - \mathbf{y}^{(\text{train})} \right\|_2^2 = 0 \quad (6)$$

$$\Rightarrow \nabla_w \left(\mathbf{X}^{(\text{train})} \mathbf{w} - \mathbf{y}^{(\text{train})} \right)^\top \left(\mathbf{X}^{(\text{train})} \mathbf{w} - \mathbf{y}^{(\text{train})} \right) = 0 \quad (7)$$

$$\Rightarrow \nabla_w \left(\mathbf{w}^\top \mathbf{X}^{(\text{train})\top} \mathbf{X}^{(\text{train})} \mathbf{w} - 2\mathbf{w}^\top \mathbf{X}^{(\text{train})\top} \mathbf{y}^{(\text{train})} + \mathbf{y}^{(\text{train})\top} \mathbf{y}^{(\text{train})} \right) = 0 \quad (8)$$

$$\Rightarrow 2\mathbf{X}^{(\text{train})\top} \mathbf{X}^{(\text{train})} \mathbf{w} - 2\mathbf{X}^{(\text{train})\top} \mathbf{y}^{(\text{train})} = 0 \quad (9)$$

$$\Rightarrow \mathbf{w} = \left(\mathbf{X}^{(\text{train})\top} \mathbf{X}^{(\text{train})} \right)^{-1} \mathbf{X}^{(\text{train})\top} \mathbf{y}^{(\text{train})} \quad (10)$$

Which finally leads to the expression used in the `computeW` function

$$\Rightarrow \mathbf{w} = \left(\mathbf{X}^{(\text{train})\top} \mathbf{X}^{(\text{train})} \right)^{-1} \mathbf{X}^{(\text{train})\top} \mathbf{y}^{(\text{train})} \quad (11)$$

With the vector of \mathbf{w} , the predictor function can be obtained simply by means of equation (3).

Finally the MSE of the predicted vs actual data-points for the $\mathbf{y}^{(\text{train})}$ is computed with the function `computeMSE`, which in turn uses MATLAB’s `immse` function.

III. K-FOLD CROSS-VALIDATION

Cross-validation techniques are used to train algorithms when the information given by the datasets is not large enough. Thus, splitting this datasets in different permutations of training and test sets allows to reuse the original data and improve the performance of the algorithm.

One of these techniques is K-Fold Cross-Validation. This method splits the dataset in k number of folds, which will create k mutually exclusive subsets, whose union results on the original dataset.

Algorithm 1 KFold Algorithm

```

1: Define KFoldXV( $\mathbb{D}, A, L, k$ )
2: Require:  $\mathbb{D}$ , the given dataset, with elements  $\mathbf{z}^{(i)}$ 
3: Require:  $A$ , the learning algorithm, seen as a function that takes a dataset as input and outputs a learned function
4: Require:  $L$ , the loss function, seen as a function from a learned function  $f$  and an example  $\mathbf{z}^{(i)} \in \mathbb{D}$  to a scalar  $\in \mathbb{R}$ 
5: Require:  $k$ , the number of folds
6: for  $i$  from 1 to  $k$  do
7:    $f_i = A(\mathbb{D} \setminus \mathbb{D}_i)$ 
8:   for  $\mathbf{z}^{(j)}$  in  $\mathbb{D}_i$  do
9:      $e_j = L(f_i, \mathbf{z}^{(j)})$ 
10:  end for
11: end for

```

Each of this combinations of test subsets, with their own training subsets (the remaining values for each one) will return an error. The total error of the algorithm is calculated with the mean value of the k number of errors.

$$E = \frac{1}{K} \sum_{i=1}^K E_i \quad (12)$$

IV. RESULTS

To begin with, the following report tries random points of a parabolic curve

$$y = x^2 \quad (13)$$

for values of $x \in (-1, 1)$

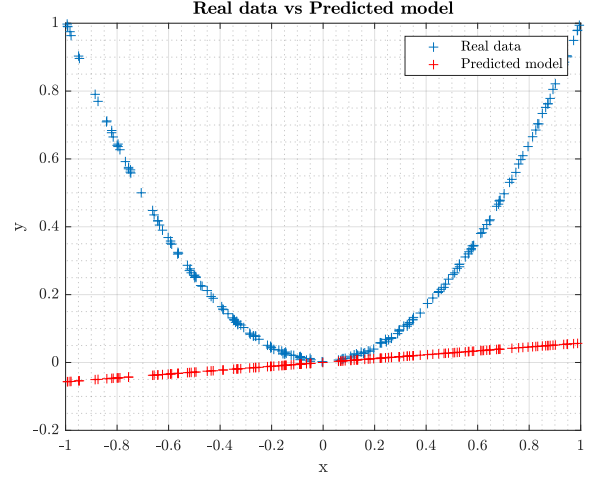


Fig. 1. Real data vs Predicted model

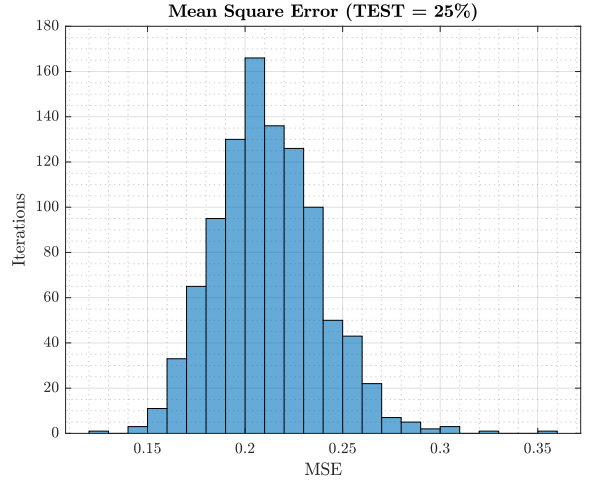


Fig. 2. Histogram for a Test ratio of (25%)

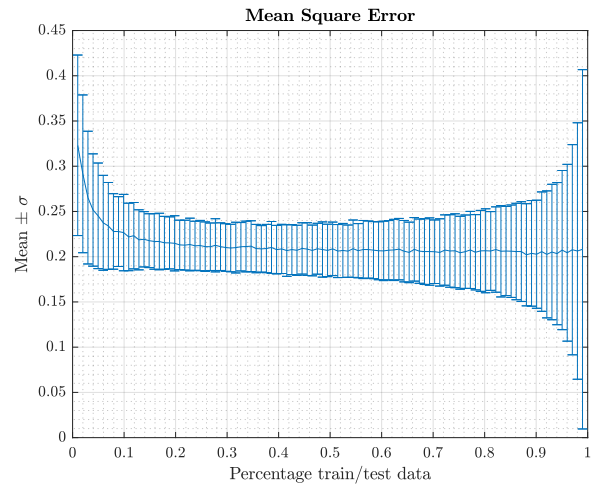


Fig. 3. MSE as a function of the Training/test ratio

A. Figure 1

MSE plot (mean and standard deviation) based on training set / test set ratio.

B. Figure 2

MSE plot (mean and standard deviation) based on nData for a specific training set / test set ratio value.

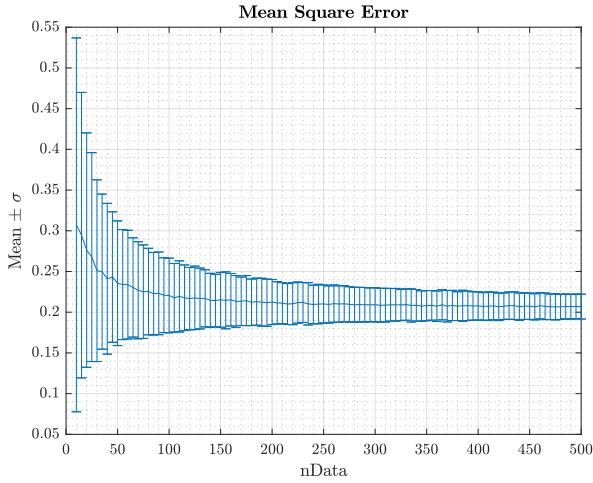


Fig. 4. MSE for a Test ratio of (25%) for different values of $nData$.

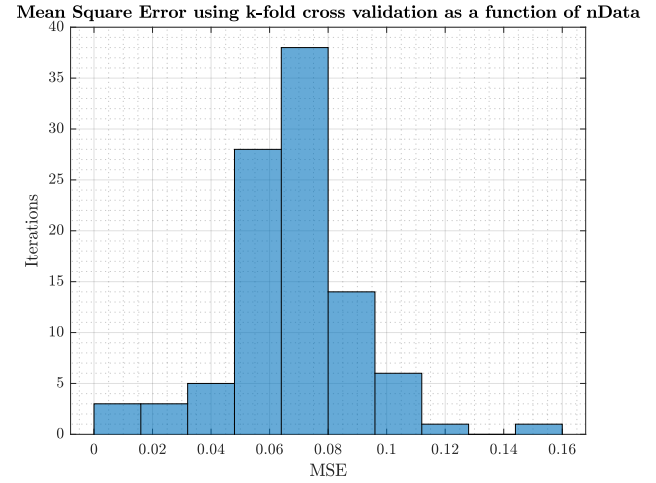


Fig. 7. Histogram as a function of $nData$ for a specific value of folds.

C. Figure 3

MSE plot (mean and standard deviation) based on the number of folds using the k -fold algorithm.

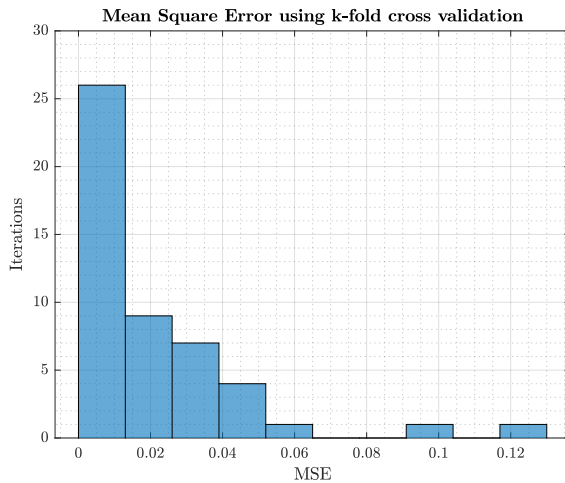


Fig. 5. Histogram as a function of folds using k -fold algorithm.

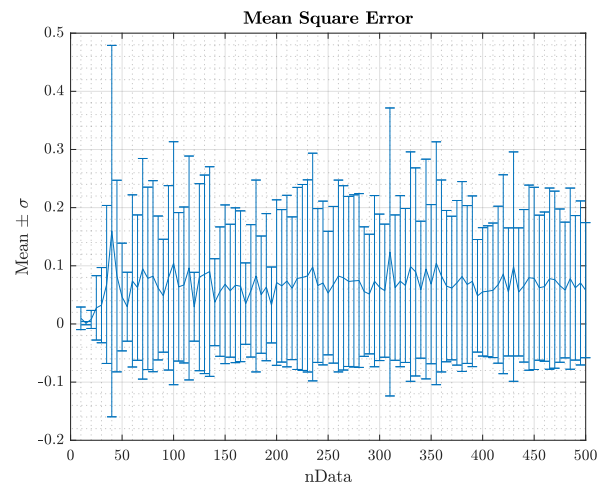


Fig. 8. MSE as a function of $nData$ for a specific value of folds.

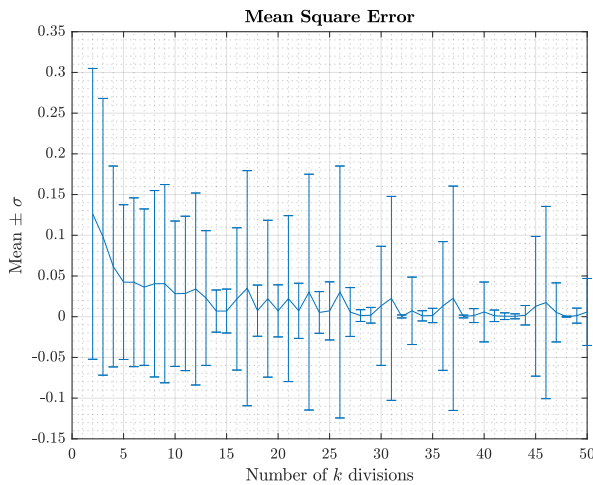


Fig. 6. MSE as a function of folds using k -fold algorithm.

D. Figure 4

MSE plot (mean and standard deviation) based on $nData$ for a specific value of folds.

E. Extra

Same problem with the term constant (i.e. add an unknown bias term b within the parameters to be found). This model is known as affine.

Comparison of the two models (linear and affine)

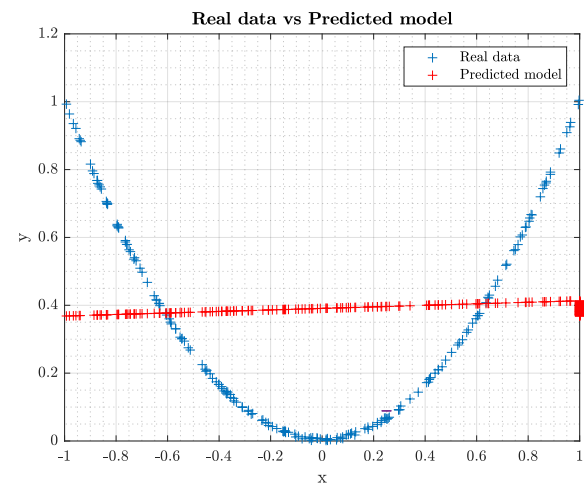


Fig. 9. Real data vs Predicted affine model.

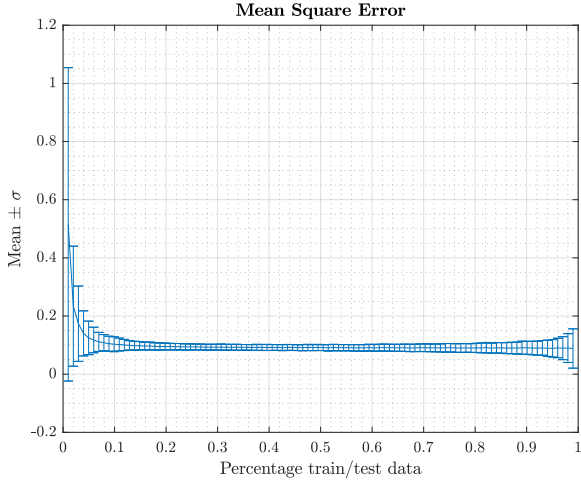


Fig. 10. MSE for affine model as a function of the Training/test ratio.

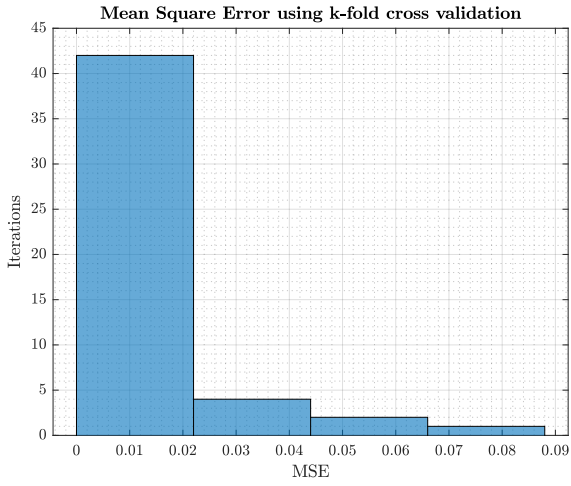


Fig. 11. Histogram for affine model using k -fold algorithm.

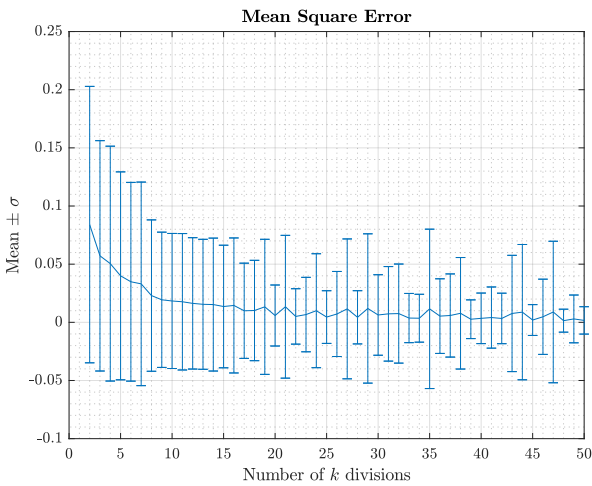


Fig. 12. MSE as a function of folds using k -fold algorithm.

V. CONCLUSIONS

This report shows different methods for predicting data using the linear regression method, along with the errors associated to it.

Comparing affine model and the linear one, it is clear how the error diminishes more rapidly in the latter. This is evidenced by comparing Figure 3 and Figure 10. The MSE when using the affine model is not only appreciably lower, but also much more defined, in the sense that the σ is very close to 0 between ratios of train/test data 0.3 to 0.6.

In both the affine and linear models, comparing Figures 5 and 6 from the linear model and their analogous Figures for the affine model 11 and 12, it can be seen that the K-Fold algorithm is somewhat effective in reducing the MSE, as it is diminished with an increasing number of folds. This, however, does not apply to its standard deviation, which does not have an apparent correlation with the number of folds in either of the models.

In summary, with the different conclusions extracted from the aforementioned figures, along with the general result presented in Figures 1 and 9, it can be stated that the affine model better predicts the data generated by a quadratic curve with some randomized error. Furthermore, both models benefited from the use of the K-Fold Cross-Validation in its training.

REFERENCES

- [1] Yale University. *Linear Regression*. 2021. URL: <http://www.stat.yale.edu/Courses/1997-98/101/linreg.htm>.
- [2] Machine Learning Glossary. *Linear Regression*. 2021. URL: https://ml-cheatsheet.readthedocs.io/en/latest/linear_regression.html#:~:text=Linear%20Regression%20is%20a%20supervised,Simple%20regression.
- [3] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. The MIT Press, 2017.

VI. CODE

A. Figure 1

```

1  %-----%
2  % Assignment 2: Linear Regression using Machine Learning
3  %-----%
4
5  % Date: 11/05/2021
6  % Author/s:
7  % Yi Qiang Ji
8  % Biel Galiot
9  % Gerard Villalta
10 % Oriol Miras
11
12 % Subject: Robotic Exploration of the Solar System
13 % Professor: Manel Soria & Arnau Miro
14
15 clc;
16 close all;
17 clear all;
18
19 % Set interpreter to latex
20 set(groot,'defaultAxesTickLabelInterpreter','latex');
21 set(groot,'defaulttextinterpreter','latex');
22 set(groot,'defaultLegendInterpreter','latex');
23
24
25 % Loop for every iteration
26 iter = 1000;
27 MSE = zeros(1,iter);
28
29 for i=1:1:iter
30     % Number of points
31     nData = 200;
32
33     % Generate data
34     [xdata,ydata] = generateData(nData);
35
36     % Split train and test data
37     [Xtrain,Ytrain,Xtest,Ytest] = splitDataTrainAndTest(xdata,ydata,nData);
38
39     % Compute coefficients
40     w = computeW(Xtrain,Ytrain);
41
42     % Predicted Y
43     YtestPredicted = predictor(w,Xtest);
44
45     % Minimum Squared Error
46     MSE(i) = computeMSE(YtestPredicted,Ytest);
47
48     % % Plots
49     % figure(1)
50     % plot(xdata,ydata,'+');
51     % hold on
52     % plot(Xtest,YtestPredicted,'r+')
53
54 end
55
56 % plot_pdf = figure(1);
57 % grid on;
58 % grid minor;
59 % box on;
60 % xlabel('x');
61 % ylabel('y');
62 % title('\textbf{Real data vs Predicted model}');
63 % legend('Real data', 'Predicted model');
64
65 % % Save pdf
66 % set(plot_pdf, 'Units', 'Centimeters');
67 % pos = get(plot_pdf, 'Position');
68 % set(plot_pdf, 'PaperPositionMode', 'Auto', 'PaperUnits', 'Centimeters', ...
69 % 'PaperSize',[pos(3), pos(4)]);
70 % print(plot_pdf, 'data_vs_predicted.pdf', '-dpdf', '-r0')
71
72 % Save png
73 % print(gcf,'data_vs_predicted.png','-dpng','-r600');
74
75 plot_pdf2 = figure(2);
76 histogram(MSE);
77 grid on;
78 grid minor;

```

```

79 box on;
80 xlabel('MSE');
81 ylabel('Iterations');
82 title('\textbf{Mean Square Error (TEST = 25\%)}');
83
84 % % Save pdf
85 % set(plot_pdf2, 'Units', 'Centimeters');
86 % pos = get(plot_pdf2, 'Position');
87 % set(plot_pdf2, 'PaperPositionMode', 'Auto', 'PaperUnits', 'Centimeters', ...
88 % 'PaperSize',[pos(3), pos(4)]);
89 % print(plot_pdf2, 'MSE_25_test.pdf', '-dpdf', '-r0')
90 %
91 % % Save png
92 % print(gcf, 'MSE_25_test.png', '-dpng', '-r600');
93
94 % Functions
95 function [Xtrain,Ytrain,Xtest,Ytest] = splitDataTrainAndTest(xdata,ydata,nData)
96
97     % Split data
98     % Holdout validation percentage
99     h_validation = 0.25; % Percentage
100
101     train_data = round(h_validation*nData);
102     % test_data = nData - train_data;
103
104     Xtrain = xdata(1:train_data);
105     Ytrain = ydata(1:train_data);
106     Xtest = xdata((train_data+1):end);
107     Ytest = ydata((train_data+1):end);
108 end
109
110 function [xdata,ydata] = generateData(nData)
111
112     % Uniform distributed values between 'init' and 'final'
113     % r = a + (b-a).*rand(N,1)
114
115     % Initial and final range
116     init = -1;
117     final = 1;
118
119     % Generate data
120     xdata = -init + (init-(final))*rand(nData,1);
121     ydata = xdata.^2 + 0.01*rand(nData,1); % Add some error
122 end
123
124 function MSE = computeMSE(YtestPredicted,Ytest)
125     MSE = immse(Ytest,YtestPredicted);
126 end
127
128 function w = computeW(Xtrain,Ytrain)
129     w = (Xtrain.'*Xtrain)\(Xtrain.'*Ytrain);
130 end
131
132 function y = predictor(w,x)
133     y = w*x;
134 end
135
136 function y = f(x)
137     y = -x.^2;
138 end

```

B. Figure 1.1

```

1 %-----%
2 % Assignment 2: Linear Regression using Machine Learning
3 %-----%
4
5 % Date: 11/05/2021
6 % Author/s:
7 % Yi Qiang Ji
8 % Biel Galiot
9 % Gerar Villalta
10 % Oriol Miras
11
12 % Subject: Robotic Exploration of the Solar System
13 % Professor: Manel Soria & Arnau Miro
14
15 clc;
16 close all;
17 clear all;

```

```

18
19 % Set interpreter to latex
20 set(groot,'defaultAxesTickLabelInterpreter','latex');
21 set(groot,'defaultTextInterpreter','latex');
22 set(groot,'defaultLegendInterpreter','latex');
23
24
25 % Loop for 1000 times
26 iter = 1000;
27
28 % Loop for each percentage
29 p_number = 100; % Δof increment
30 percentage = linspace(0.01,0.99,p_number); % From [0,1]
31 MSE = zeros(p_number,iter);
32
33 % Loop for every percentage
34 for j = 1:length(percentage)
35
36     for i=1:iter
37
38         % Number of points
39         nData = 200;
40
41         % Generate data
42         [xdata,ydata] = generateData(nData);
43
44         % Split train and test data
45         [Xtrain,Ytrain,Xtest,Ytest] = splitDataTrainAndTest(xdata,ydata,nData,percentage(j));
46
47         % Compute coefficients
48         w = computeW(Xtrain,Ytrain);
49
50         % Predicted Y
51         YtestPredicted = predictor(w,Xtest);
52
53         % Minimum Squared Error
54         MSE(j,i) = computeMSE(YtestPredicted,Ytest);
55
56         % Plots
57         % plot(xdata,ydata,'+');
58         % hold on
59         % plot(Xtest,YtestPredicted,'r+')
60         end
61         mean_MSE(1,j) = mean(MSE(j,:));
62         std_MSE(1,j) = std(MSE(j,:));
63     end
64
65     plot_pdf = figure(1);
66     errorbar(percentage,mean_MSE(1,:),std_MSE(1,:));
67     grid on;
68     grid minor;
69     box on;
70     xlabel('Percentage train/test data');
71     ylabel('Mean  $\pm$   $\sigma$ ');
72     title('\textbf{Mean Square Error}');
73
74     % % Save pdf
75     % set(plot_pdf, 'Units', 'Centimeters');
76     % pos = get(plot_pdf, 'Position');
77     % set(plot_pdf, 'PaperPositionMode', 'Auto', 'PaperUnits', 'Centimeters', ...
78     % 'PaperSize',[pos(3), pos(4)]);
79     % print(plot_pdf, 'MSE_error.pdf', '-dpdf', '-r0')
80     %
81     % % Save png
82     % print(gcf, 'MSE_error.png', '-dpng', '-r600');
83
84     % Functions
85     function [Xtrain,Ytrain,Xtest,Ytest] = splitDataTrainAndTest(xdata,ydata,nData,percentage)
86
87         % Split data
88         % Holdout validation percentage
89         h_validation = percentage; % Percentage
90
91         train_data = round(h_validation*nData);
92         % test_data = nData - train_data;
93
94         Xtrain = xdata(1:train_data);
95         Ytrain = ydata(1:train_data);
96         Xtest = xdata((train_data+1):end);
97         Ytest = ydata((train_data+1):end);
98     end
99
100 function [xdata,ydata] = generateData(nData)
101

```

```

102 % Uniform distributed values between 'init' and 'final'
103 % r = a + (b-a).*rand(N,1)
104
105 % Initial and final range
106 init = -1;
107 final = 1;
108
109 % Generate data
110 xdata = -init + (init-(final))*rand(nData,1);
111 ydata = xdata.^2 + 0.01*rand(nData,1); % Add some error
112 end
113
114 function MSE = computeMSE(YtestPredicted,Ytest)
115     MSE = immse(Ytest,YtestPredicted);
116 end
117
118 function w = computeW(Xtrain,Ytrain)
119     w = (Xtrain.'*Xtrain)\(Xtrain.'*Ytrain);
120 end
121
122 function y = predictor(w,x)
123     y = w*x;
124 end
125
126
127 function y = f(x)
128     y = -x.^2;
129 end

```

C. Figure 2

```

1
2 clc;
3 close all;
4 clear all;
5
6 % Loop for 1000 times
7 iter = 1000;
8
9 % Loop for each nData
10 data = 10:5:500; % From [10,500]
11 MSE = zeros(length(data),iter);
12
13 % Loop for every nData
14 for j = 1:1:length(data)
15
16     for i=1:1:iter
17
18         % Number of points
19         nData = data(j);
20
21         % Generate data
22         [xdata,ydata] = generateData(nData);
23
24         % Split train and test data
25         [Xtrain,Ytrain,Xtest,Ytest] = splitDataTrainAndTest(xdata,ydata,nData);
26
27         % Compute coefficients
28         w = computeW(Xtrain,Ytrain);
29
30         % Predicted Y
31         YtestPredicted = predictor(w,Xtest);
32
33         % Minimum Squared Error
34         MSE(j,i) = computeMSE(YtestPredicted,Ytest);
35
36     end
37     mean_MSE(1,j) = mean(MSE(j,:));
38     std_MSE(1,j) = std(MSE(j,:));
39 end
40
41 plot_pdf = figure(1);
42 errorbar(data,mean_MSE(1,:),std_MSE(1,:));
43 grid on;
44 grid minor;
45 box on;
46 xlabel('nData');
47 ylabel('Mean  $\pm$   $\sigma$ ');
48 title('\textbf{Mean Square Error}');
49

```



```

50 % % Save pdf
51 % set(plot_pdf, 'Units', 'Centimeters');
52 % pos = get(plot_pdf, 'Position');
53 % set(plot_pdf, 'PaperPositionMode', 'Auto', 'PaperUnits', 'Centimeters', ...
54 % 'PaperSize',[pos(3), pos(4)]);
55 % print(plot_pdf, 'MSE_error_ndata_25.pdf', '-dpdf', '-r0')
56 %
57 % % Save png
58 % print(gcf, 'MSE_error_ndata_25.png', '-dpng', '-r600');
59
60 % Functions
61 function [Xtrain,Ytrain,Xtest,Ytest] = splitDataTrainAndTest(xdata,ydata,nData)
62
63 % Split data
64 % Holdout validation percentage
65 h_validation = 0.25; % Percentage
66
67 train_data = round(h_validation*nData);
68 % test_data = nData - train_data;
69
70 Xtrain = xdata(1:train_data);
71 Ytrain = ydata(1:train_data);
72 Xtest = xdata((train_data+1):end);
73 Ytest = ydata((train_data+1):end);
74 end
75
76 function [xdata,ydata] = generateData(nData)
77
78 % Initial and final range
79 init = -1;
80 final = 1;
81
82 % Generate data
83 xdata = -init + (init-final)*rand(nData,1);
84 ydata = xdata.^2 + 0.01*rand(nData,1); % Add some error
85 end
86
87 function MSE = computeMSE(YtestPredicted,Ytest)
88 MSE = immse(Ytest,YtestPredicted);
89 end
90
91 function w = computeW(Xtrain,Ytrain)
92 w = (Xtrain.'*Xtrain)\(Xtrain.'*Ytrain);
93 end
94
95 function y = predictor(w,x)
96 y = w*x;
97 end
98
99
100 function y = f(x)
101 y = -x.^2;
102 end

```

D. Figure 3

```

1 %-----%
2 % Assignment 2: Linear Regression using Machine Learning
3 %-----%
4
5 % Date: 11/05/2021
6 % Author/s:
7 % Yi Qiang Ji
8 % Biel Galiot
9 % Gerar Villalta
10 % Oriol Miras
11
12 % Subject: Robotic Exploration of the Solar System
13 % Professor: Manel Soria & Arnau Miro
14
15 clc;
16 close all;
17 clear all;
18
19 % Set interpreter to latex
20 set(groot, 'defaultAxesTickLabelInterpreter', 'latex');
21 set(groot, 'defaultTextInterpreter', 'latex');
22 set(groot, 'defaultLegendInterpreter', 'latex');
23
24

```

```

25 % Divisions
26 k_div = 2:1:50;
27
28 % MSE vector
29 MSE = zeros(1,length(k_div));
30
31 % Number of points
32 nData = 200;
33
34 % Generate data
35 [xdata,ydata] = generateData(nData);
36
37
38 for j=1:1:length(k_div)
39     % Loop for every division
40     for i=1:1:k_div(j)
41
42         MSE_div = zeros(k_div(j),1);
43         % Split train and test data
44         [Xtrain,Ytrain,Xtest,Ytest] = splitDataTrainAndTest(xdata,ydata,nData, i, k_div(j));
45
46         % Compute coefficients
47         w = computeW(Xtrain,Ytrain);
48
49         % Predicted Y
50         YtestPredicted = predictor(w,Xtest);
51
52         % Minimum Squared Error
53         MSE_div(i,:) = computeMSE(YtestPredicted,Ytest);
54     % % Plots
55     % figure(1)
56     % plot(xdata,ydata,'+');
57     % hold on
58     % plot(Xtest,YtestPredicted,'r+')
59 end
60     MSE(1,j) = mean(MSE_div);
61     mean_MSE = MSE;
62     std_MSE(1,j) = std(MSE_div);
63     hold on;
64 end
65
66 plot_pdf = figure(1);
67 histogram(MSE,10);
68 grid on;
69 grid minor;
70 box on;
71 xlabel('MSE');
72 ylabel('Iterations');
73 title('\textbf{Mean Square Error using k-fold cross validation}');
74
75 % % Save pdf
76 % set(plot_pdf, 'Units', 'Centimeters');
77 % pos = get(plot_pdf, 'Position');
78 % set(plot_pdf, 'PaperPositionMode', 'Auto', 'PaperUnits', 'Centimeters', ...
79 % 'PaperSize',[pos(3), pos(4)]);
80 % print(plot_pdf, 'MSE_kfold.pdf', '-dpdf', '-r0')
81 %
82 % % Save png
83 % print(gcf,'MSE_kfold.png','-dpng','-r600');
84
85 plot_pdf2 = figure(2);
86 errorbar(k_div,mean_MSE(1,:),std_MSE(1,:));
87 grid on;
88 grid minor;
89 box on;
90 xlabel('Number of $k$ divisions');
91 ylabel('Mean $\pm$ $\sigma$');
92 title('\textbf{Mean Square Error}');
93
94 % % Save pdf
95 % set(plot_pdf2, 'Units', 'Centimeters');
96 % pos = get(plot_pdf2, 'Position');
97 % set(plot_pdf2, 'PaperPositionMode', 'Auto', 'PaperUnits', 'Centimeters', ...
98 % 'PaperSize',[pos(3), pos(4)]);
99 % print(plot_pdf2, 'MSE_kfold_error.pdf', '-dpdf', '-r0')
100 %
101 % % Save png
102 % print(gcf,'MSE_kfold_error.png','-dpng','-r600');
103
104 % Functions
105 function [Xtrain,Ytrain,Xtest,Ytest] = splitDataTrainAndTest(xdata,ydata,nData, index, k_div)
106
107     % Split data
108     % Holdout validation percentage

```

```

109     % h_validation = 0.25; % Percentage
110
111     % train_data = round(h_validation*nData);
112     % test_data = nData - train_data;
113
114     div = floor(nData/k_div);
115     Xtest = xdata(((index-1)*div + 1):index*div);
116     Ytest = ydata(((index-1)*div + 1):index*div);
117     Xtrain = setdiff(xdata,Xtest);
118     Ytrain = setdiff(ydata,Ytest);
119 end
120
121 function [xdata,ydata] = generateData(nData)
122
123     % Uniform distributed values between 'init' and 'final'
124     % r = a + (b-a).*rand(N,1)
125
126     % Initial and final range
127     init = -1;
128     final = 1;
129
130     % Generate data
131     xdata = -init + (init-(final))*rand(nData,1);
132     ydata = xdata.^2 + 0.01*rand(nData,1); % Add some error
133 end
134
135 function MSE = computeMSE(YtestPredicted,Ytest)
136     MSE = immse(Ytest,YtestPredicted);
137 end
138
139 function w = computeW(Xtrain,Ytrain)
140     w = (Xtrain.'*Xtrain)\(Xtrain.'*Ytrain);
141 end
142
143 function y = predictor(w,x)
144     y = w*x;
145 end
146
147
148 function y = f(x)
149     y = -x.^2;
150 end

```

E. Figure 4

```

1  %-----%
2  % Assignment 2: Linear Regression using Machine Learning
3  %-----%
4
5  % Date: 11/05/2021
6  % Author/s:
7  % Yi Qiang Ji
8  % Biel Galiot
9  % Gerard Villalta
10 % Oriol Miras
11
12 % Subject: Robotic Exploration of the Solar System
13 % Professor: Manel Soria & Arnau Miro
14
15 clc;
16 close all;
17 clear all;
18
19 % Set interpreter to latex
20 set(groot,'defaultAxesTickLabelInterpreter','latex');
21 set(groot,'defaulttextinterpreter','latex');
22 set(groot,'defaultLegendInterpreter','latex');
23
24
25 % Divisions
26 k_div = 4;
27
28 % MSE_vector
29 MSE = zeros(1,length(k_div));
30
31 % Number of points
32 nData = 10:5:500; % From [10,500]
33
34
35 for j=1:length(nData)

```

```

36
37 % Generate data
38 [xdata,ydata] = generateData(nData(j));
39
40 % Loop for every division
41 for i=1:1:k_div
42
43     MSE_div = zeros(k_div,1);
44
45     % Split train and test data
46     [Xtrain,Ytrain,Xtest,Ytest] = splitDataTrainAndTest(xdata,ydata,nData(j), i, k_div);
47
48     % Compute coefficients
49     w = computeW(Xtrain,Ytrain);
50
51     % Predicted Y
52     YtestPredicted = predictor(w,Xtest);
53
54     % Minimum Squared Error
55     MSE_div(i,:) = computeMSE(YtestPredicted,Ytest);
56
57     % % Plots
58     % figure(1)
59     % plot(xdata,ydata,'+');
60     % hold on
61     % plot(Xtest,YtestPredicted,'r+')
62 end
63     MSE(1,j) = mean(MSE_div);
64     mean_MSE = MSE;
65     std_MSE(1,j) = std(MSE_div);
66     hold on;
67 end
68 plot_pdf = figure(1);
69 histogram(MSE,10);
70 grid on;
71 grid minor;
72 box on;
73 xlabel('MSE');
74 ylabel('Iterations');
75 title('\textbf{Mean Square Error using k-fold cross validation as a function of nData}');
76
77 % % Save pdf
78 % set(plot_pdf, 'Units', 'Centimeters');
79 % pos = get(plot_pdf, 'Position');
80 % set(plot_pdf, 'PaperPositionMode', 'Auto', 'PaperUnits', 'Centimeters', ...
81 % 'PaperSize',[pos(3), pos( 4)]);
82 % print(plot_pdf, 'MSE_kfold_nData.pdf', '-dpdf', '-r0')
83 %
84 % % Save png
85 % print(gcf, 'MSE_kfold_nData.png', '-dpng', '-r600');
86
87 plot_pdf2 = figure(2);
88 errorbar(nData,mean_MSE(1,:),std_MSE(1,:));
89 grid on;
90 grid minor;
91 box on;
92 xlabel('nData');
93 ylabel('Mean  $\pm$   $\sigma$ ');
94 title('\textbf{Mean Square Error}');
95
96 % % Save pdf
97 % set(plot_pdf2, 'Units', 'Centimeters');
98 % pos = get(plot_pdf2, 'Position');
99 % set(plot_pdf2, 'PaperPositionMode', 'Auto', 'PaperUnits', 'Centimeters', ...
100 % 'PaperSize',[pos(3), pos(4)]);
101 % print(plot_pdf2, 'MSE_kfold_nData_error.pdf', '-dpdf', '-r0')
102 %
103 % % Save png
104 % print(gcf, 'MSE_kfold_nData_error.png', '-dpng', '-r600');
105
106 % Functions
107 function [Xtrain,Ytrain,Xtest,Ytest] = splitDataTrainAndTest(xdata,ydata,nData, index, k_div)
108
109     % Split data
110     % Holdout validation percentage
111     % h_validation = 0.25; % Percentage
112
113     % train_data = round(h_validation*nData);
114     % test_data = nData - train_data;
115
116     div = floor(nData/k_div);
117     Xtest = xdata(((index-1)*div + 1):index*div);
118     Ytest = ydata(((index-1)*div + 1):index*div);
119     Xtrain = setdiff(xdata,Xtest);

```

```

120     Ytrain = setdiff(ydata,Ytest);
121 end
122
123 function [xdata,ydata] = generateData(nData)
124
125     % Uniform distributed values between 'init' and 'final'
126     % r = a + (b-a).*rand(N,1)
127
128     % Initial and final range
129     init = -1;
130     final = 1;
131
132     % Generate data
133     xdata = -init + (init-final).*rand(nData,1);
134     ydata = xdata.^2 + 0.01*rand(nData,1); % Add some error
135 end
136
137 function MSE = computeMSE(YtestPredicted,Ytest)
138     MSE = immse(Ytest,YtestPredicted);
139 end
140
141 function w = computeW(Xtrain,Ytrain)
142     w = (Xtrain.*Xtrain)\(Xtrain.*Ytrain);
143 end
144
145 function y = predictor(w,x)
146     y = w*x;
147 end
148
149
150 function y = f(x)
151     y = -x.^2;
152 end

```

F. Extra 1

```

1  %-----%
2  % Assignment 2: Linear Regression using Machine Learning
3  %-----%
4
5  % Date: 11/05/2021
6  % Author/s:
7  % Yi Qiang Ji
8  % Biel Galiot
9  % Gerard Villalta
10 % Oriol Miras
11
12 % Subject: Robotic Exploration of the Solar System
13 % Professor: Manel Soria & Arnau Miro
14
15 clc;
16 close all;
17 clear all;
18
19 % Set interpreter to latex
20 set(groot,'defaultAxesTickLabelInterpreter','latex');
21 set(groot,'defaultTextInterpreter','latex');
22 set(groot,'defaultLegendInterpreter','latex');
23
24
25 % Loop for 1000 times
26 iter = 1000;
27
28 % Loop for each percentage
29 p_number = 100; % Δof increment
30 percentage = linspace(0.01,0.99,p_number); % From [0,1]
31 MSE = zeros(p_number,iter);
32
33 % Loop for every percentage
34 for j = 1:length(percentage)
35
36     for i=1:iter
37
38         % Number of points
39         nData = 200;
40
41         % Generate data
42         [xdata,ydata] = generateData(nData);
43
44         % Split train and test data

```

```

45     [Xtrain,Ytrain,Xtest,Ytest] = splitDataTrainAndTest(xdata,ydata,nData,percentage(j));
46
47     % Compute coefficients
48     w = computeW(Xtrain,Ytrain);
49
50     % Predicted Y
51     YtestPredicted = predictor(w,Xtest);
52
53     % Minimum Squared Error
54     MSE(j,i) = computeMSE(YtestPredicted,Ytest);
55
56     % Plots
57     % plot(xdata,ydata,'+');
58     % hold on
59     % plot(Xtest,YtestPredicted,'r+')
60     end
61     mean_MSE(1,j) = mean(MSE(j,:));
62     std_MSE(1,j) = std(MSE(j,:));
63 end
64
65 plot_pdf = figure(1);
66 errorbar(percentage,mean_MSE(1,:),std_MSE(1,:));
67 grid on;
68 grid minor;
69 box on;
70 xlabel('Percentage train/test data');
71 ylabel('Mean  $\pm$   $\sigma$ ');
72 title('\textbf{Mean Square Error}');
73
74 % % Save pdf
75 % set(plot_pdf, 'Units', 'Centimeters');
76 % pos = get(plot_pdf, 'Position');
77 % set(plot_pdf, 'PaperPositionMode', 'Auto', 'PaperUnits', 'Centimeters', ...
78 % 'PaperSize',[pos(3), pos(4)]);
79 % print(plot_pdf, 'MSE_afi.pdf', '-dpdf', '-r0')
80 %
81 % % Save png
82 % print(gcf, 'MSE_afi.png', '-dpng', '-r600');
83
84 % Functions
85 function [Xtrain,Ytrain,Xtest,Ytest] = splitDataTrainAndTest(xdata,ydata,nData,percentage)
86
87     % Split data
88     % Holdout validation percentage
89     h_validation = percentage; % Percentage
90
91     train_data = round(h_validation*nData);
92     % test_data = nData - train_data;
93
94     Xtrain = xdata(1:train_data);
95     Xtrain = [xdata(1:train_data) ones(size(Xtrain,1),1)];
96     Ytrain = ydata(1:train_data);
97
98     Xtest = xdata((train_data+1):end);
99     Xtest = [xdata((train_data+1):end) ones(size(Xtest,1),1)];
100    Ytest = ydata((train_data+1):end);
101 end
102
103 function [xdata,ydata] = generateData(nData)
104
105     % Uniform distributed values between 'init' and 'final'
106     % r = a + (b-a).*rand(N,1)
107
108     % Initial and final range
109     init = -1;
110     final = 1;
111
112     % Generate data
113     xdata = -init + (init-(final))*rand(nData,1);
114     ydata = xdata.^2 + 0.01*rand(nData,1); % Add some error
115 end
116
117 function MSE = computeMSE(YtestPredicted,Ytest)
118     MSE = immse(Ytest,YtestPredicted);
119 end
120
121 function w = computeW(Xtrain,Ytrain)
122     w = (Xtrain.*Xtrain)\(Xtrain.*Ytrain);
123 end
124
125 function y = predictor(w,x)
126     y = x*w;
127 end
128

```

```

129
130 function y = f(x)
131 y = -x.^2;
132 end

```

G. Extra 2

```

1  %-----%
2  % Assignment 2: Linear Regression using Machine Learning
3  %-----%
4
5  % Date: 11/05/2021
6  % Author/s:
7  % Yi Qiang Ji
8  % Biel Galiot
9  % Gerard Villalta
10 % Oriol Miras
11
12 % Subject: Robotic Exploration of the Solar System
13 % Professor: Manel Soria & Arnau Miro
14
15 clc;
16 close all;
17 clear all;
18
19 % Set interpreter to latex
20 set(groot, 'defaultAxesTickLabelInterpreter', 'latex');
21 set(groot, 'defaulttextinterpreter', 'latex');
22 set(groot, 'defaultLegendInterpreter', 'latex');
23
24
25 % Divisions
26 k_div = 2:1:50;
27
28 % MSE vector
29 MSE = zeros(1, length(k_div));
30
31 % Number of points
32 nData = 200;
33
34 % Generate data
35 [xdata, ydata] = generateData(nData);
36
37
38 for j=1:length(k_div)
39     % Loop for every division
40     for i=1:k_div(j)
41
42         MSE_div = zeros(k_div(j), 1);
43         % Split train and test data
44         [Xtrain, Ytrain, Xtest, Ytest] = splitDataTrainAndTest(xdata, ydata, nData, i, k_div(j));
45
46         % Compute coefficients
47         w = computeW(Xtrain, Ytrain);
48
49         % Predicted Y
50         YtestPredicted = predictor(w, Xtest);
51
52         % Minimum Squared Error
53         MSE_div(i, :) = computeMSE(YtestPredicted, Ytest);
54
55         % % Plots
56         % figure(1)
57         % plot(xdata, ydata, '+');
58         % hold on
59         % plot(Xtest, YtestPredicted, 'r+')
60     end
61     MSE(1, j) = mean(MSE_div);
62     mean_MSE = MSE;
63     std_MSE(1, j) = std(MSE_div);
64     hold on;
65 end
66
67 plot_pdf = figure(1);
68 histogram(MSE);
69 grid on;
70 grid minor;
71 box on;
72 xlabel('MSE');
73 ylabel('Iterations');
74 title('\textbf{Mean Square Error using k-fold cross validation}');

```

```

74
75 % % Save pdf
76 % set(plot_pdf, 'Units', 'Centimeters');
77 % pos = get(plot_pdf, 'Position');
78 % set(plot_pdf, 'PaperPositionMode', 'Auto', 'PaperUnits', 'Centimeters', ...
79 % 'PaperSize',[pos(3), pos(4)]);
80 % print(plot_pdf, 'MSE_afi_kfold.pdf', '-dpdf', '-r0')
81 %
82 % % Save png
83 % print(gcf, 'MSE_afi_kfold.png', '-dpng', '-r600');
84
85 plot_pdf2 = figure(2);
86 errorbar(k_div, mean_MSE(1,:), std_MSE(1,:));
87 grid on;
88 grid minor;
89 box on;
90 xlabel('Number of  $k$  divisions');
91 ylabel('Mean  $\pm \sigma$ ');
92 title('\textbf{Mean Square Error}');
93
94 % % Save pdf
95 % set(plot_pdf2, 'Units', 'Centimeters');
96 % pos = get(plot_pdf2, 'Position');
97 % set(plot_pdf2, 'PaperPositionMode', 'Auto', 'PaperUnits', 'Centimeters', ...
98 % 'PaperSize',[pos(3), pos(4)]);
99 % print(plot_pdf2, 'MSE_afi_kfold_error.pdf', '-dpdf', '-r0')
100 %
101 % % Save png
102 % print(gcf, 'MSE_afi_kfold_error.png', '-dpng', '-r600');
103
104 % Functions
105 function [Xtrain,Ytrain,Xtest,Ytest] = splitDataTrainAndTest(xdata,ydata,nData, index, k_div)
106
107     % Split data
108     % Holdout validation percentage
109     % h_validation = 0.25; % Percentage
110
111     % train_data = round(h_validation*nData);
112     % test_data = nData - train_data;
113
114     div = floor(nData/k_div);
115     Xtest = xdata(((index-1)*div + 1):index*div);
116     Xtest = [xdata(((index-1)*div + 1):index*div) ones(size(Xtest,1),1)];
117     Ytest = ydata(((index-1)*div + 1):index*div);
118
119     Xtrain = setdiff(xdata,Xtest);
120     Xtrain = [setdiff(xdata,Xtest) ones(size(Xtrain,1),1)];
121     Ytrain = setdiff(ydata,Ytest);
122 end
123
124 function [xdata,ydata] = generateData(nData)
125
126     % Uniform distributed values between 'init' and 'final'
127     % r = a + (b-a).*rand(N,1)
128
129     % Initial and final range
130     init = -1;
131     final = 1;
132
133     % Generate data
134     xdata = -init + (init-final)*rand(nData,1);
135     ydata = xdata.^2 + 0.01*rand(nData,1); % Add some error
136 end
137
138 function MSE = computeMSE(YtestPredicted,Ytest)
139     MSE = immse(Ytest,YtestPredicted);
140 end
141
142 function w = computeW(Xtrain,Ytrain)
143     w = (Xtrain.'*Xtrain)\(Xtrain.'*Ytrain);
144 end
145
146 function y = predictor(w,x)
147     y = x*w;
148 end
149
150
151 function y = f(x)
152     y = -x.^2;
153 end

```