Machine Learning Assignment 2 report

Linear regression

Roberto Albanese 4234506
Andrea Tiranti 4856315
Università degli Studi di Genova
Robotics Enginireeng

***Abstract*—In Statistics, Linear regression refers to a model that can show relationship between two variables and how one can impact the other, this process aims at finding a linear relationship between a target and one or more predictors based on measured data. In this laboratory the objective is to implement different example of linear regression, in both one and multi-dimensional cases and to asses their performances by graphic comparison.**

I. INTRODUCTION

A. Assignment Description

In this assignment we are asked to develop a linear regression model with two different datasets and then to test the algorithm to check its robustness. The two datasets are given in advance and they are structured in the following way:

- turkish-se-SP500vsMSCI
- mtcarsdata-4features

where turkish-se-SP500vsMSCI represents the evolution of two Turkish stock market indexes, S&P500 and MSCI (536 observations), and mtcarsdata-4features represents the collection of cars with 4 relative features, namely: miles per gallon (*mpg*), displacement (*disp*), horsepower (*hp*) and *weight* (32 observations).

The first task is to upload the datasets and make them readable in MATLAB. Then we are asked to implement a linear regression model with four different instances:

1. 1-dimensional problem without intercept on the Turkish stock exchange data;
2. Compare graphically the solution obtained on different random subsets (10%) of the whole data set;
3. 1-dimensional problem with intercept on the Motor Trends car data, using columns *mpg* and *weight*;
4. Multi-dimensional problem on the complete MTcars data, using all four columns (predict *mpg* with the other three columns).

Once we get a proper algorithm, we are asked to re-run multiple times instance 1, 3 and 4 for different randomly-splitted training-test and to show graphically the results of this operation.

B. Linear Regression

Linear regression is a linear approach to modelling the relationship between a scalar response (or dependent variable) and one or more explanatory variables (or independent variables). The case of one explanatory variable is called simple linear regression.
If the explanatory variable is made of only one feature, we will talk about 1-dimensional linear regression. If the features are multiple, we will handle a multidimensional linear regression.

The generic goal is to minimize the mean value of the loss over the whole dataset, evaluating an objective function made in the following way:

$$J = \sum_{l=0}^{N} \lambda(t_l, y_l) \tag{1}$$

where:

- J is the objective function or cost function;
- λ is the loss function;
- N is the number of observations;
- t_l is the measured target value;
- y_l is the inferred target value.

For this experiment the square error loss function was used, yielding to a formula for the mean square error such as:

$$J_{MSE} = \sum_{l=0}^{N} \lambda(t_l, y_l)^2 \tag{2}$$

In all the below cases, the goal is to minimize the J_{MSE} with respect to the parameters with fixed data, which means finding a value of ω such that:

$$\frac{\mathrm{d}}{\mathrm{d}w} J_{MSE} = 0 \tag{3}$$

where:

- ω is the parameter which approximates the model predicting the target t given the observation x (i.e $y = wx$).

C. 1-dimensional linear regression

In a 1-dimensional linear regression, we have that the observation vector is *N*x*1* dimensioned. In this case the linear combination between input and output becomes the equation of a straight line:

$$y(x,w) = w_0 + w_1 x_1 \tag{4}$$

where:

- w_1 is called *slope*;
- w_0 is called *intercept*.

1) Case with no intercept: In the case of no intercept the strait line becomes $y(x,\omega) = \omega_1 x_1$. For this particular case the mean square error is minimized if:

$$\omega_1 = \frac{\sum_{l=1}^{N} x_1 t_1}{\sum_{l=1}^{N} x_1^2} \tag{5}$$

2) Case with intercept: in this case we can minimize the objective function if:

$$\omega_1 = \frac{\sum_{l=1}^{N}(x_1 - \bar{x})(t_1 - \bar{t})}{\sum_{l=1}^{N}(x_1 - \bar{x})^2} \tag{6}$$

$$\omega_0 = \bar{t} - \omega_1 \bar{x} \tag{7}$$

where:

- $\bar{x} = \sum_{l=1}^{N} x_l$;
- $\bar{t} = \sum_{l=1}^{N} t_l$.

D. Multi-dimensional linear regression

In a multi-dimensional regression problem, we have that D>1 (where D is the number of features of the observations vector).
This means that ω will be a vector of dimension D. In this case the mean square error objective is minimized when:

$$\omega = (X^T X)^{-1} X^T t = X^{\dagger} t \tag{8}$$

where:

- X is the matrix of observations of dimension NxD;
- $X^{\dagger}$ is the so called Moore-Penrose pseudoinverse if X.

II. IMPLEMENTATION

A. Data Pre-Processing

In order to make the code faster and lighter, the first step of our implementation is the data loading and pre-processing. We tried to implement an universal way to load datasets to make the code adaptable to every input. This was done with the Matlab function *loadSet()* built up by us. It receives as inputs the desired type of format and the name of the set, and it returns back the dataset inside a matrix. By using this function, in the very first phase of the project, we were able to load the 2 datasets "turkish-se-SP500vsMSCI.csv" and "mtcarsdata-4features.csv" and convert them from '.csv' format to double. The generated matrices were then stored in a data struct for a better encapsulation of the code.

B. Linear Regression

The main scope of the project is the implementation of an algorithm for linear regression. Since it is necessary to do this type of approximation with two different sets of different dimensions, we split the linear regression into two functions, one for the 1-dimensional case (*"linearRegression1D()"*) and one for the multidimensional case (*"linearRegression()"*) .

For the 1-dimensional linear regression the function adapts itself to give as output the weights ω_0, ω_1 when the offset is present or ω_1 otherwise.
By receiving as input a set, this function first extracts the vectors ***t*** and ***x*** and then computes the weights using (6) and (7) or (5).

For what concerns the multidimensional case, the function extracts the matrix of observation ***X*** and the target vector ***t*** and then computes the weights using the closed form (8). Finally, the function outputs the weights and stores them inside a vector.

C. Test the linear regression model

We tested our regression model by executing the algorithm with a randomly chosen set made by a 5% of the input dataset; then, the same operation was applied also on the remaining 95%. The goal is to compare the algorithm when the size of the dataset is either too small or reasonably big, highlighting the necessity of having a sufficiently big dataset for a correct behavior of the model.

For the randomization we have used *"randSet()"*, which receives as inputs a set and the percentage of the set that we want to extract, and it gives as outputs the requested set and the remaining part of the original set.

We proceeded by computing the "Square Error Loss" (2) with the function *"evalJMSE()"* that receives 2 vectors as inputs, the target vector and the predictions. We computed the objective function for instances 1, 3 and 4. Finally, we have collected the errors in 3 different vectors.
By iterating the process ten times we were able to graphically compare the errors using a bar graph (***Figure 7***).

III. CONCLUSION

The results are shown following the order of the exercises requested in the assignment. The figures are shown at the bottom of the document for a better visualization.

A. 1-dimensional problem without intercept on the Turkish stock exchange data (*Figure 1*)

The results in this case are satisfying. First of all, the straight line passes through the origin of the axis as expected, since the intercept is not considered; then, our output properly approximates the points whose coordinates are represented by the attribute's instance value on the x-axis, and by the target's corresponding instance value on the y-axis.

*B. Compare graphically the solution obtained on different random subsets (10%) of the whole data set (**Figure 2-3-4**)*

Also in this case the results' behaviour is quite good, since we expect that the regression applied to the subset must try to follow the straight line resulting from exercise 1. For clarity, we reported different case studies to capture all the possible random generations of the subsets and their variations. In the three figures the observations of the *Turkish Stock Exchange* are depicted by empty red circles, while the observations of the subset are represented by full red circles. In all of them we can clearly see that the slope of the red line changes w.r.t. the subset randomly chosen, and that all the lines still intercept the origin.

*C. 1-dimensional problem with intercept on the Motor Trends car data, using columns mpg and weight (**Figure 5**)*

Moving to the second dataset, the first experiment concerns a linear regression using only 2 attributes of the dataset (*mpg*, *weights*). We expect a similar result to the graphs of the first and second exercises, but in this case we must use the intercept (offset) for approximating the data. Thus, the result, as shown, must have a straight line that does not pass through the origin but follows the equation (4). The line also clearly approximates the distribution of the data, so we can be satisfied about this result.

*D. Multi-dimensional problem on the complete MTcars data, using all four columns (predict mpg with the other three columns) (**Figure 6**)*

This time we are applying a multidimensional linear regression since we are considering four features (*disp*, *hp*, *weight*, *mpg*). The result is depicted using a 3D representation, where the points correspond to the value of the first three variables and the colour represents the 4th variable (*mpg*) that we chose as target. On the right side, there is the real distribution of data, so we can see, for example, that the most heavy and powerful cars have the highest fuel consumption rate. Comparing the two graphs we can see that the results of the approximation are quite similar to the real distribution, hence the algorithm behaviour is consistent with the expected one.

*E. Results of the testing of the regression model (**Figure 7-8**)*

For what concerns the first dataset (*Turkish Stock Exchange*) the "Square Error Loss" of the 95% set is similar for all the iterations as we expected, since we have an appropriate amount of data; for the other subset, corresponding to the 5%, the error fluctuates more because the data belong to different period of time and they may not be correlated to one another (data collected in similar periods may be more similar than data collected from the beginning and the end of the whole period).

Considering the "*mtcarsdata-4features*" dataset, we can notice an unusual behaviour for both the 1-dimensional and the multi-dimensional cases. Focusing on the value of the MSE, we can see that for the 95% of the set the error is very high w.r.t. the "*Turkish Stock Exchange*" due to the limited number of observation in the car dataset. For the 5% subset the situation seems to be the opposite as the error is nearly zero, but this is caused by an overfitting problem since we have more features than observations. By selecting a bigger subset, for example 25%, as proposed in the ***Figure 8***, we can have a more clear result as the error has decreased in value. Nevertheless, the error remains too big and the overall set is too small for being accepted.

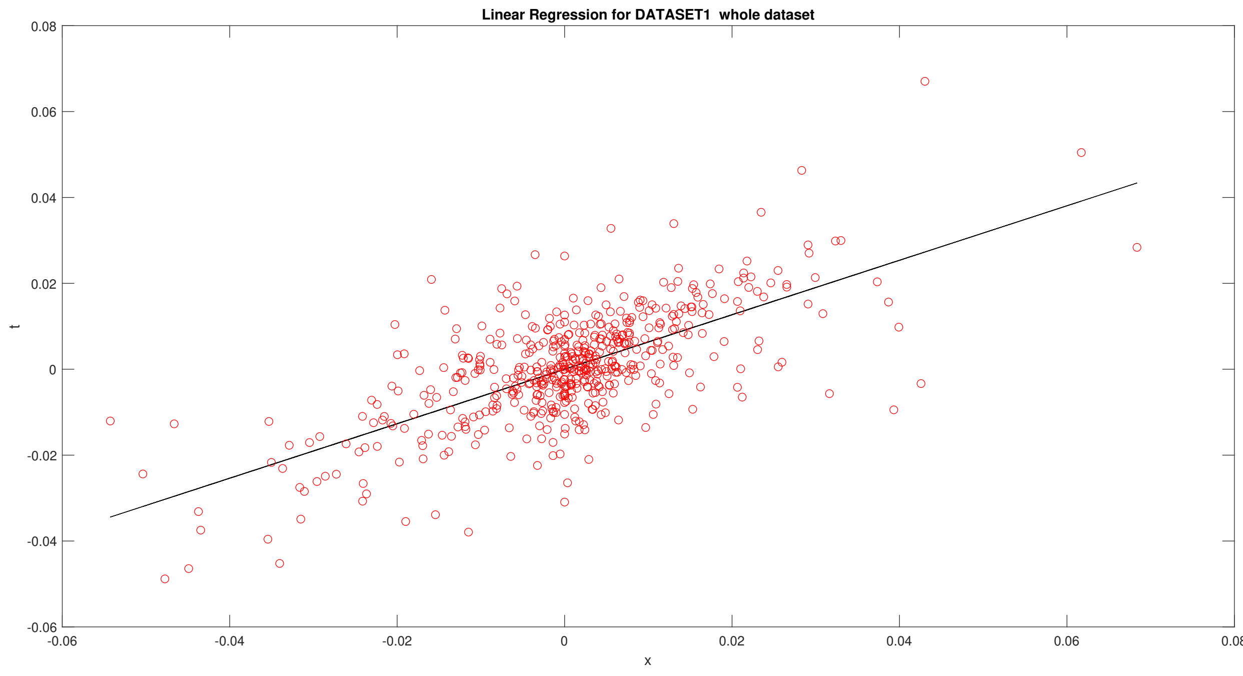

Fig. 1: Task 1: 1-dimensional linear regression (turkish-se-SP500vsMSCI)

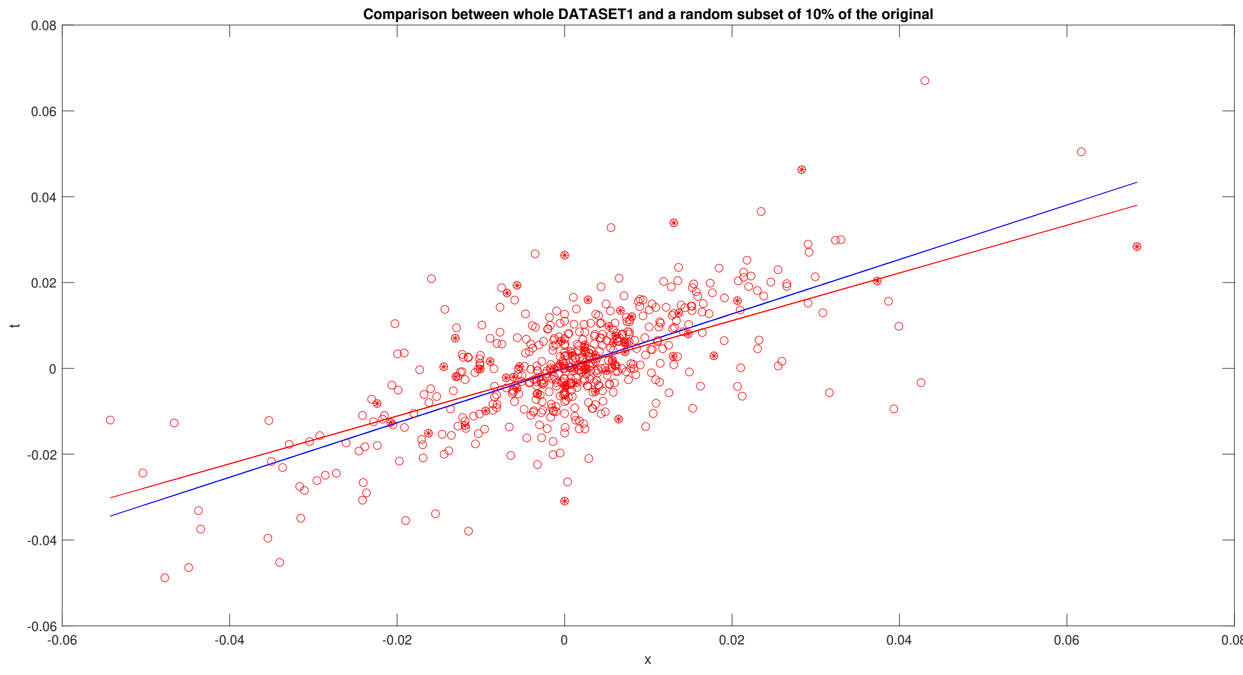

Fig. 2: First comparison: turkish-se-SP500vsMSCI with a 10% reduced set

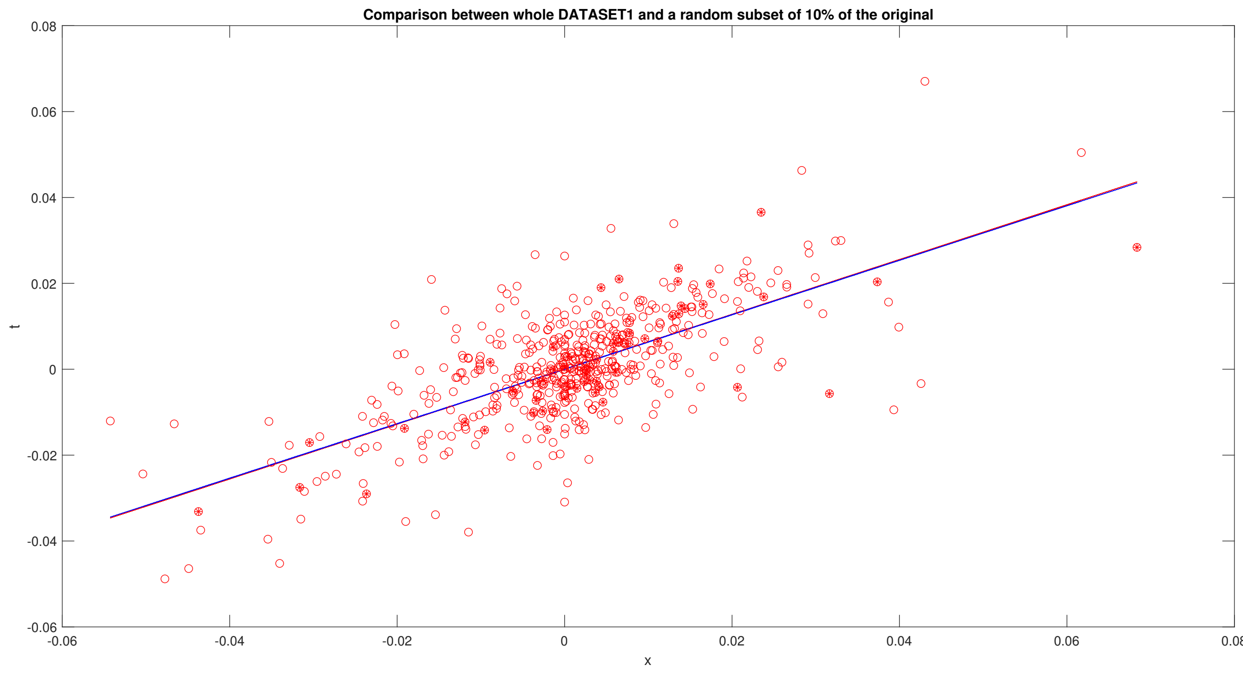

Fig. 3: Second comparison: turkish-se-SP500vsMSCI with a 10% reduced set

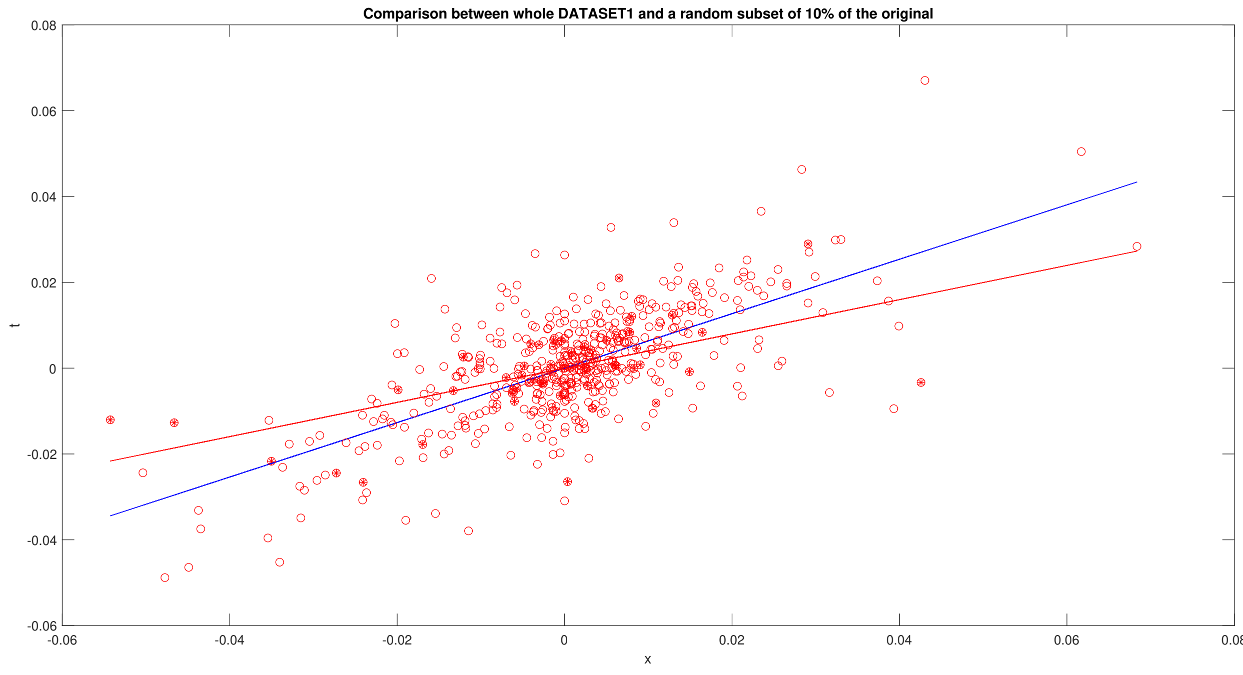

Fig. 4: Third comparison: turkish-se-SP500vsMSCI with a 10% reduced set

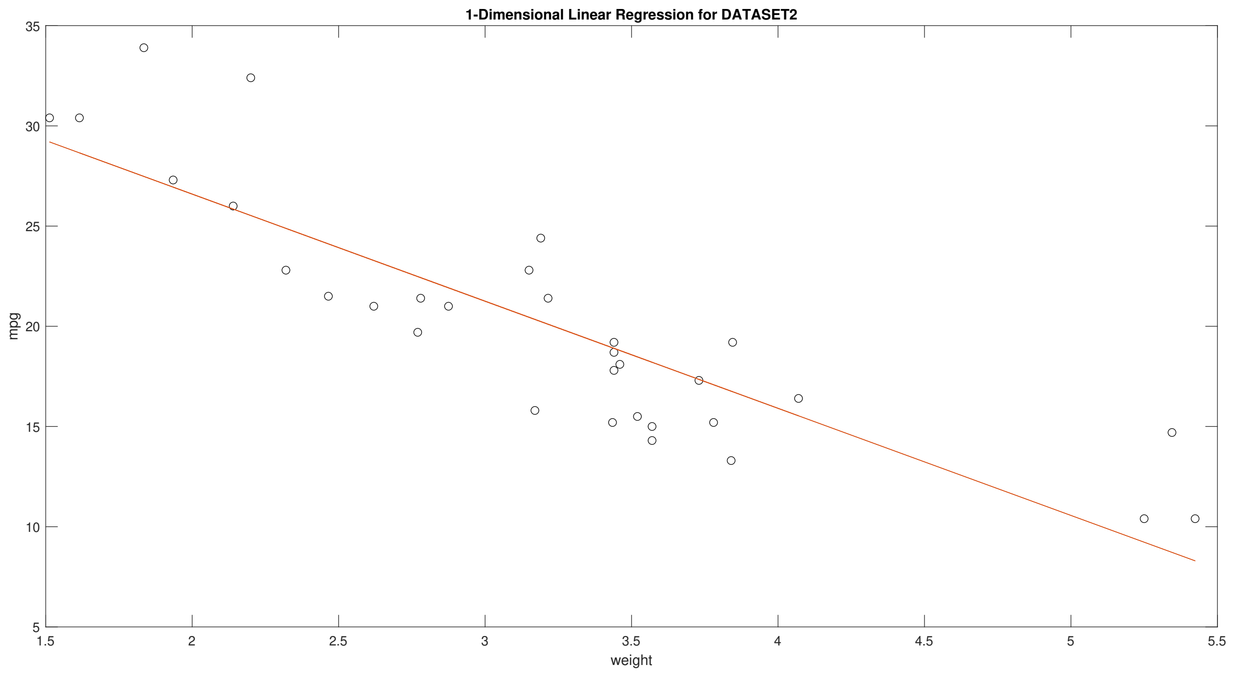

Fig. 5: Task 3: 1-dimensional linear regression (mtcarsdata-4features)

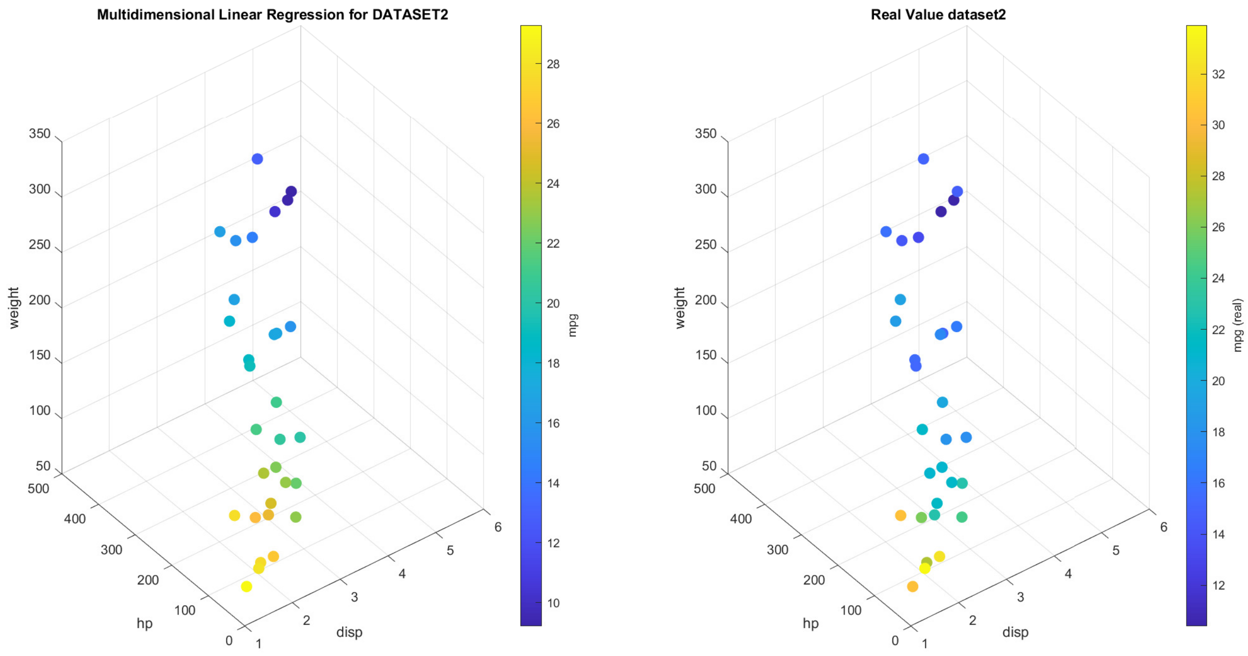

Fig. 6: Multi-dimensional linear regression (mtcarsdata-4features)

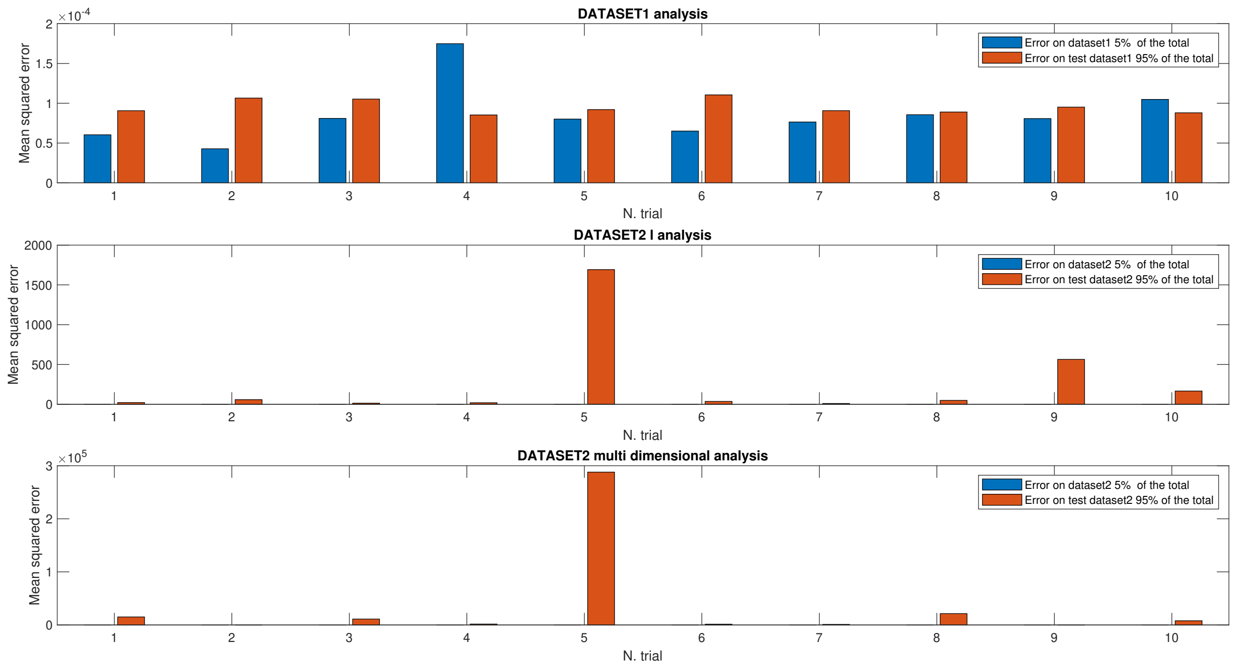

Fig. 7: Error comparison (5% of the set in blue, 95% of the set in orange)

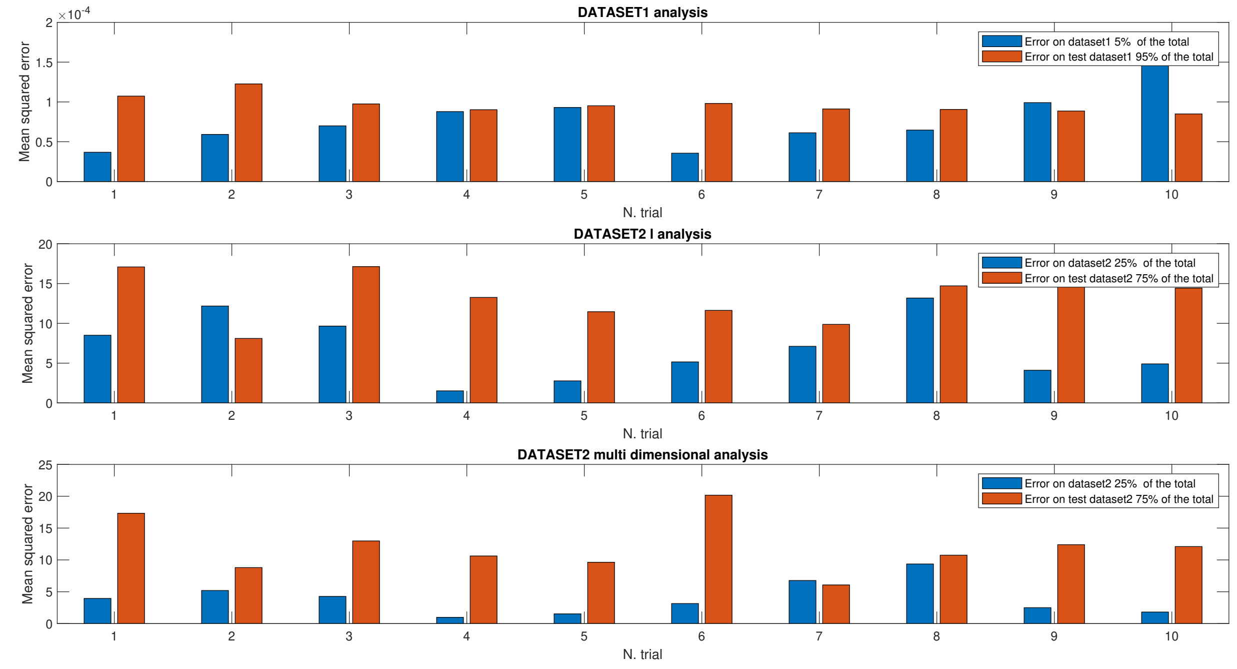

Fig. 8: Error comparison (25% of the set in blue, 75% of the set in orange)