
POLITECNICO
DI TORINO

**Master of science in ICT for smart societies**

# Report on
# ICT for Health Laboratory N°1

Regression on Parkinson data

*Author*:
Bruno VALENTE

a.a. 2018-2019

# Introduction

Parkinson's disease is a chronic progressive neurological disorder caused by the reduction of dopamine level, due to the death of dopaminergic neurons in the substantia nigra. It is characterised by several motor and non-motor symptoms, impacting differently on the patient's functioning. The former include tremor at rest, rigidity, postural instability, bradykinesia and akinesia. Bradykinesia indicates slowness of movements, while akinesia is the difficulty in starting an action. They can cause impairments in the performance of sequential and simultaneous tasks, fine motor tasks, gesturing, swallowing and talking. On the other hand, non-motor symptoms consist of mood swings, cognitive impairment and progressive decline, sleep disorders, depression, apathy, pain and autonomic dysfunctions.
The **Unified Parkinson's Disease Rating Scale (UPDRS)** is one of the most valid and reliable scales currently available to assess the severity of both motor and non-motor symptoms of Parkinson's disease. It is a 42-item rating scale organized in 4 subscales, scored with numeric options (0-4) or yes/no responses (in Part IV).
A substantial percentage of patients with Parkinson's disease shows speech impairments, such as reduced loudness, weakness of the voice, hoarseness, short rushes of speech, monotonous voice and imprecise articulation.
In this project, we aim to investigate the potential linear correlation between those voice features and the total UPDRS scores, since it is an expensive measurement in terms of price and time and it would be useful to predict its value in an easier way. In order to find the best regression algorithm to express this correlation, we are going to analyze a sample of 42 patients affected by Parkinson's disease examined several times by health care professionals for a total of 5875 measurements, 5 or 6 measures per day for each patient, during a 6-months time lapse. Useless information, such as the subject identifier, age, sex and the test time, have been removed from the dataset. Therefore, the resulting table consist of 18 columns: motor UPDRS, total UPDRS and 16 voice measurements (e.g. shimmer, jitter, etc.).
The first half of the data has been chosen as **training set**, while the second one has been divided into two groups: one as **validation set** and the other as **testing set**. The records were ordered by patient. Hence, they were previously shuffled in order to guarantee the heterogeneity of the patients' features in each of the three sets. Finally, data has been standardized.

# Regression Algorithms

The regression techniques implemented in Python, and described in the following sections, are:

1. The linear least square estimation
2. The conjugate gradient algorithm
3. The gradient algorithm
4. The stochastic gradient algorithm
5. The steepest descent algorithm
6. The ridge regression

In the descriptions of these methods the notations rules will be the following:
bold lowercase letters for column vectors, bold capital letters for matrices; $\mathbf{y}$ will represent the column vector of the **regressands** (in our case the total UPDRS), $\mathbf{X}$ will be the matrix of the

**regressors** (in our case a matrix having the motor UPDRS and the voice features in each row) and $\mathbf{w}$ will be the weight vector which expresses the correlation between regressors and regressands.
Furthermore, variables ending in **_train** will be the ones used to train the algorithms, variables ending in **_test** the ones related to the testing set, variables ending in **_val** will represent the validation set. The "**hat**" variables will be the predicted ones.
As mentioned in the Introduction, data have been standardized. It means that they are all centred around zero and this allow us to simplify the model because there is no need to take the "offset" into consideration. Indeed, if data are not centred, it is necessary to modify the model. This might be done by adding a column of 1's to the matrix of the regressors ($\mathbf{X}$), so that the linear regression has an intercept term. Standardization could also reduce numerical problems, because each feature has been "scaled", dividing it by its standard deviation.

## 1. The Linear least square estimation

The purpose of the linear least square estimation (LLS) is to find the values of the vector $\mathbf{w}$ in order to minimize the function $f(\mathbf{w}) = \|\mathbf{y} - \mathbf{X}\mathbf{w}\|^2$. To achieve this aim, we evaluate the gradient of $f(\mathbf{w})$ and set it to 0, getting the following relationship: $\widehat{\mathbf{w}} = (\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{y}$, where $(\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathrm{T}}$ is called pseudo-inverse of the matrix $\mathbf{X}$ and $\widehat{\mathbf{w}}$ is the weight vector that minimizes $f(\mathbf{w})$.

*Results on Parkinson data*
The weight vector obtained with the LLS is showed in *Figure 1.1*. As we expected, it assumes large values for some of the features, especially feature 8 and 11 (which are both related to different measurements of the shimmer), while it is almost zero for others.

$$\mathbf{w} = \begin{bmatrix} 0.94996142 \\ -0.15227387 \\ 0.10935644 \\ -2.6279677 \\ 0.08202066 \\ 2.66799633 \\ -0.16586647 \\ 0.03671549 \\ -7.16056731 \\ 0.11319543 \\ -0.03343807 \\ 7.19162309 \\ -0.06890604 \\ -0.03177305 \\ 0.03602064 \\ -0.0314352 \\ -0.03392362 \end{bmatrix}$$

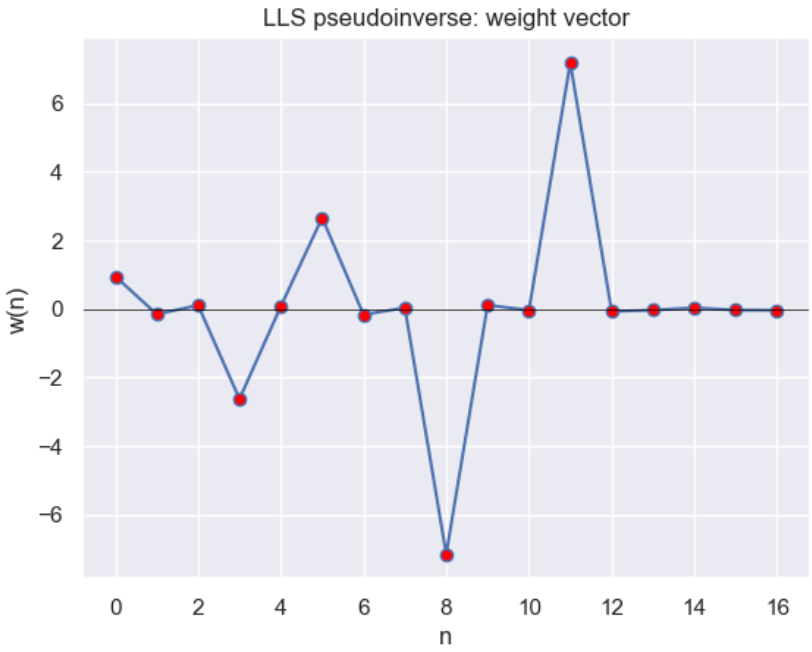


Figure 1.1

In *Figure 1.2* it is possible to see the plot of **yhat_train** versus **y_train**, while *Figure 1.3* shows the plot of **yhat_test** versus **y_yest.** The marginal charts, at the top and at the right, show the distribution of the 2 variables using histograms and the fitted kernel density estimations (KDE).

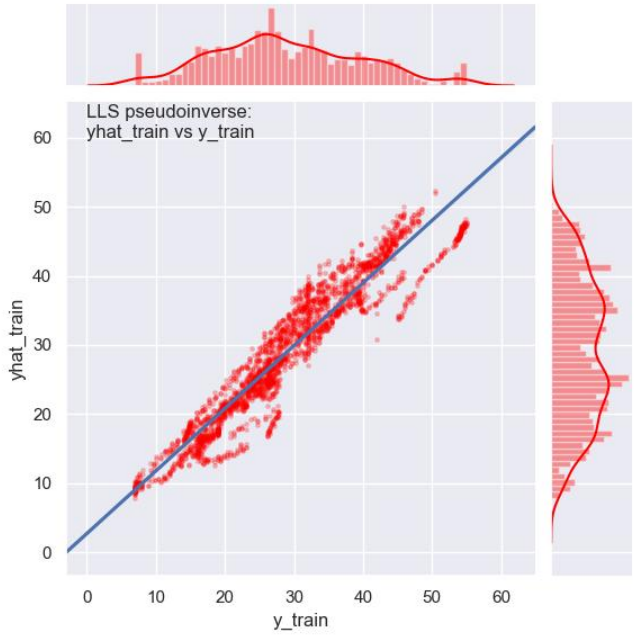


Figure 1.2

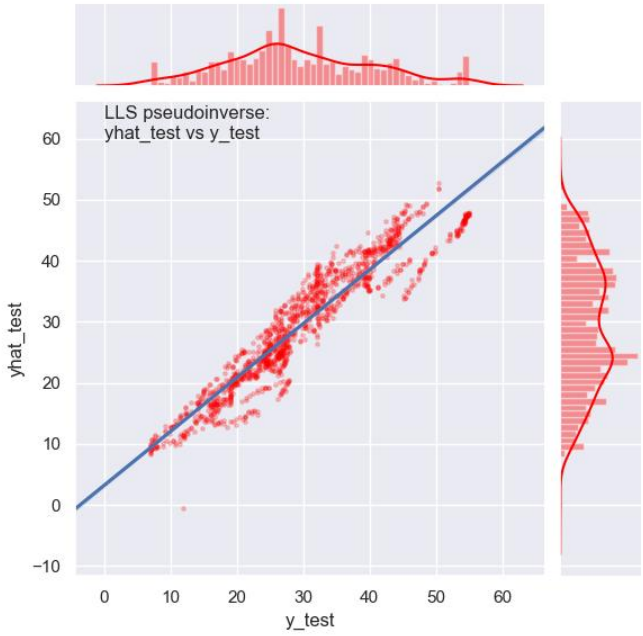


Figure 1.3

*Figure 1.4* and *Figure 1.5* provide the plots for the histograms of the errors **y_train** - **yhat_train** and **y_test** - **yhat_test**. It is possible to notice that errors approximately follow a Gaussian distribution with mean zero. In addition we can see that the maximum error for **y_train** is a little bit much higher than 10, while for **y_test** it is around 12.5 for some of the instances.

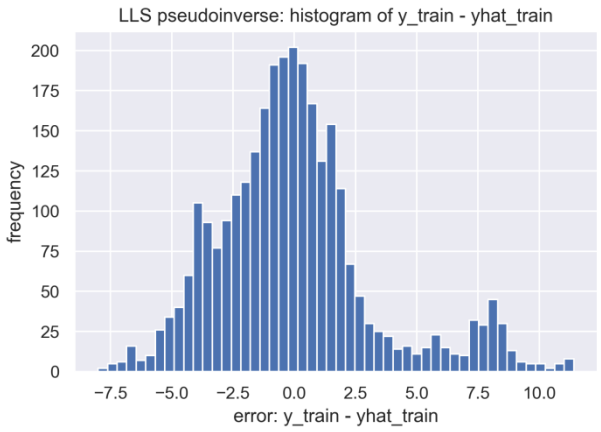


Figure 1.4

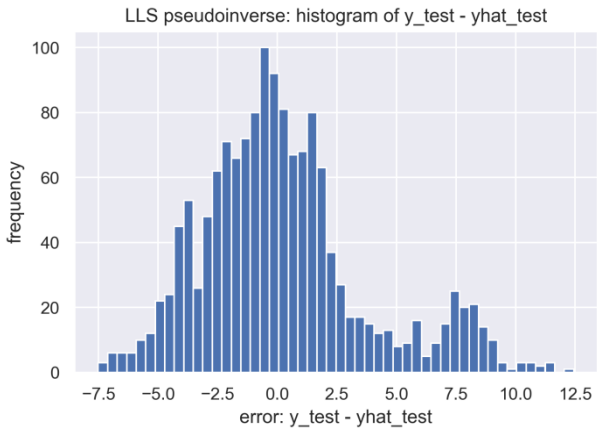


Figure 1.5

The measured mean square errors using the LLS algorithm are:

- Mean square error for the training set: 10.5186
- Mean square error for the testing set: 12.27817
- Mean square error for the validation set: 11.4999

## 2. The conjugate gradient algorithm

The problem posed by the conjugate gradient algorithm consists in solving, for $\mathbf{w}$, the set of equations $\nabla f(\mathbf{w}) = \mathbf{Q}\mathbf{w} - \mathbf{b} = 0$, where $\mathbf{Q} = \mathbf{X}^{\mathrm{T}}\mathbf{X}$ ($\mathbf{Q} \in \mathbb{R}^{N \times N}$ is a symmetric and positive-definite matrix) and $\mathbf{b} = \mathbf{X}^{\mathrm{T}}\mathbf{y}$. Using the fact that search directions are required to be $\mathbf{Q}$-orthogonal to each other, conjugate gradient converges in at most $N$ steps (with $N$ the number of columns of $\mathbf{X}$).

*Results on Parkinson data*

The weight vector obtained with the conjugate gradient algorithm is showed in *Figure 2.1*. As can be seen, it gives much weight to the feature 0 (motor UPDRS), while the others are pretty damped.

$$\mathbf{w} = \begin{bmatrix} 0.95004942 \\ -0.15235869 \\ 0.10951198 \\ 0.01936095 \\ 0.08217866 \\ 0.02046072 \\ -0.1657037 \\ 0.03663851 \\ 0.01550396 \\ 0.113075 \\ -0.03345257 \\ 0.01566702 \\ -0.06900158 \\ -0.03188839 \\ 0.03587647 \\ -0.0313596 \\ -0.03399738 \end{bmatrix}$$

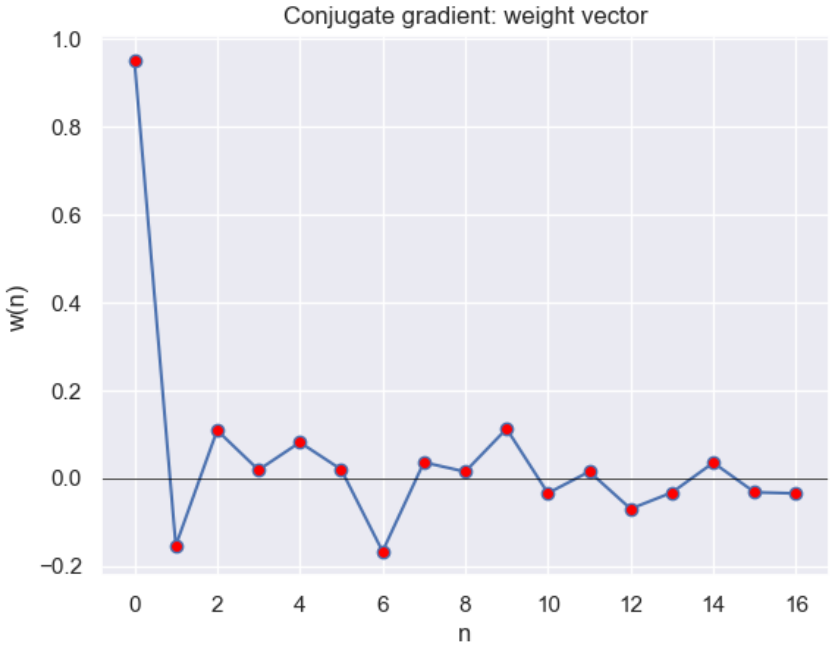


Figure 2.1

In *Figure 2.2* it is possible to notice the plot of **yhat_train** versus **y_train**, while *Figure 2.3* shows the plot of **yhat_test** versus **y_yest.** The marginal charts, at the top and at the right, show the distribution of the 2 variables using histograms and the fitted kernel density estimations (KDE).

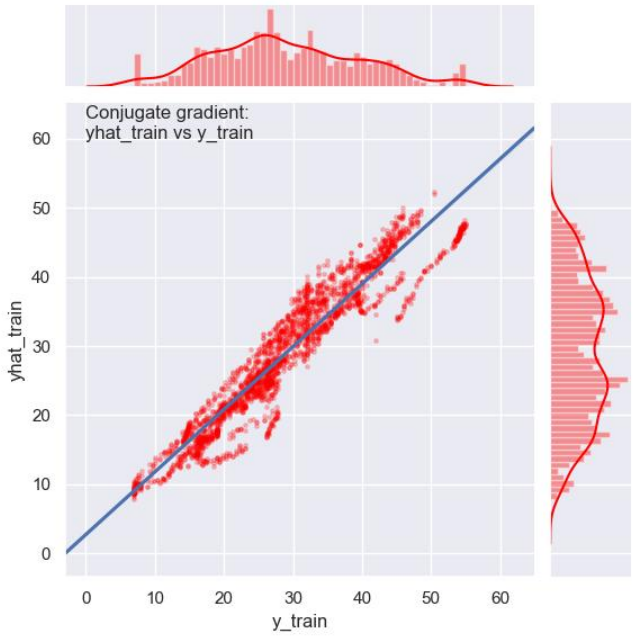


Figure 2.2

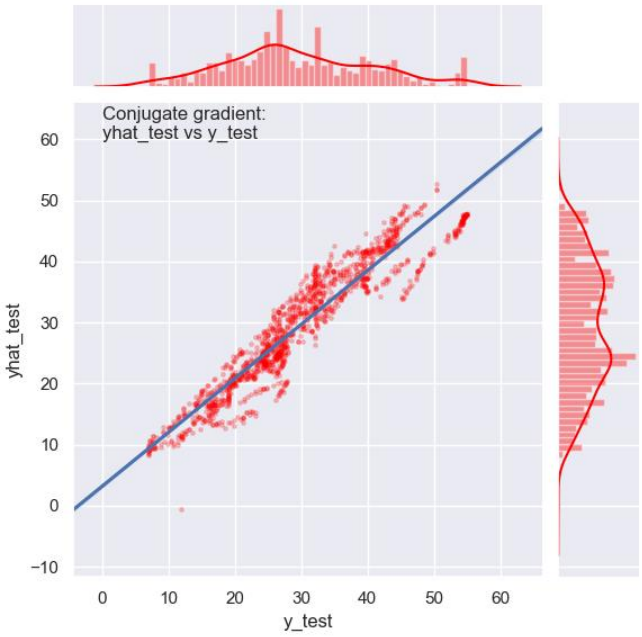


Figure 2.3

*Figure 2.4* and *Figure 2.5* provide the plots for the histograms of the errors **y_train** - **yhat_train** and **y_test** - **yhat_test**. It is possible to notice that errors approximately follow a Gaussian

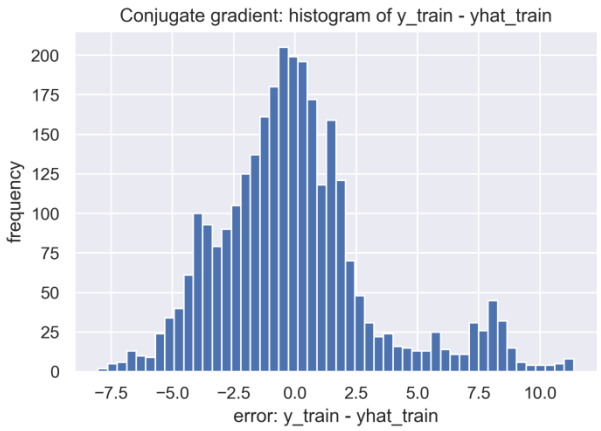


Figure 2.4

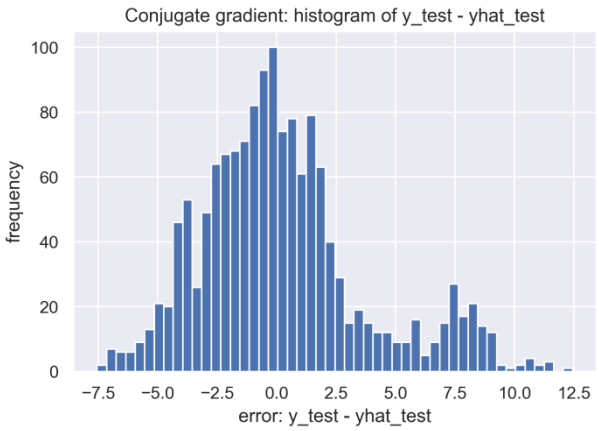


Figure 2.5

distribution with mean zero. In addition we can see that the maximum error for **y_train** is a little bit much higher than 10, while for **y_test** it is around 12.5 for some instances.

The measured mean square errors using the conjugate gradient algorithm are:

- Mean square error for the training set: 10.51946
- Mean square error for the testing set: 12.27376
- Mean square error for the validation set: 11.50238

## 3. The gradient algorithm

The gradient algorithm, as the name suggests, uses the gradient to find the minimum of a function. It starts from a random initial $\mathbf{w}$ and reaches the minimum, moving towards the opposite direction compared to the one of the gradient. Mathematically speaking: $\mathbf{w}_{i+1} = \mathbf{w}_i - \gamma \nabla f(\mathbf{w}_i)$, where $i$ represents the $i$-th step and $\gamma$ is called "learning coefficient". It is important to reasonably choose the value of $\gamma$ and a correct stopping condition (e.g. a specified number of iterations), in order to avoid unwanted behaviours.

*Results on Parkinson data*

For the Parkinson dataset analyzed, the value of $\gamma$ has been set to $10^{-5}$ and the algorithm stops after 10 thousands iterations. The weight vector obtained is showed in *Figure 3.1*. It gives much weight to the feature 0 (motor UPDRS), even though other features play an important role in the regression.

$$\mathbf{w} = \begin{bmatrix} 0.95003694 \\ -0.15314584 \\ 0.10936653 \\ 0.27538296 \\ 0.0810067 \\ -0.23393206 \\ -0.14929886 \\ 0.0285475 \\ -0.34998788 \\ 0.11157705 \\ -0.03460048 \\ 0.37577015 \\ -0.06938125 \\ -0.03199163 \\ 0.03573163 \\ -0.03151784 \\ -0.03360355 \end{bmatrix}$$

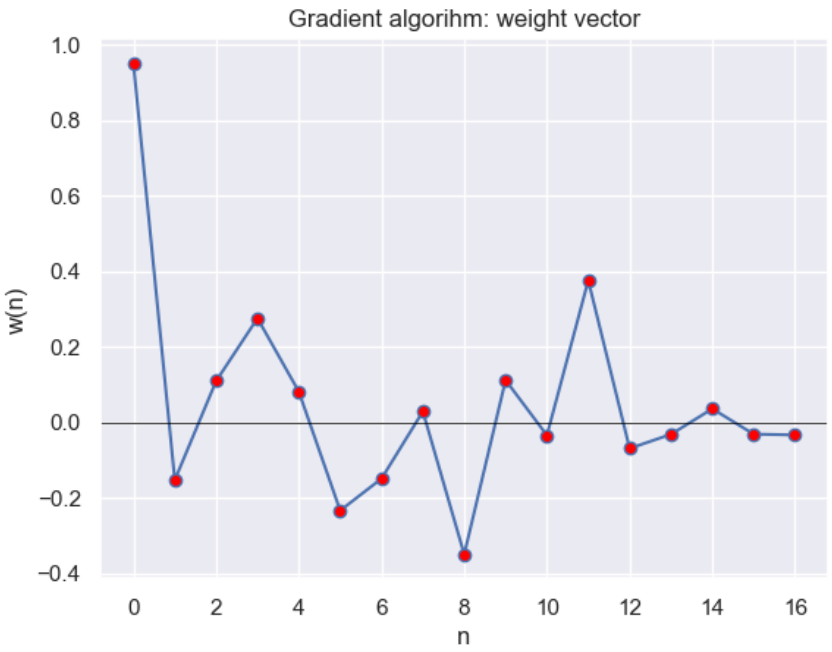


Figure 3.1

In *Figure 3.2* it is possible to notice the plot of **yhat_train** versus **y_train**, while *Figure 3.3* shows the plot of **yhat_test** versus **y_yest.** The marginal charts, at the top and at the right, show the distribution of the 2 variables using histograms and the fitted kernel density estimations (KDE).

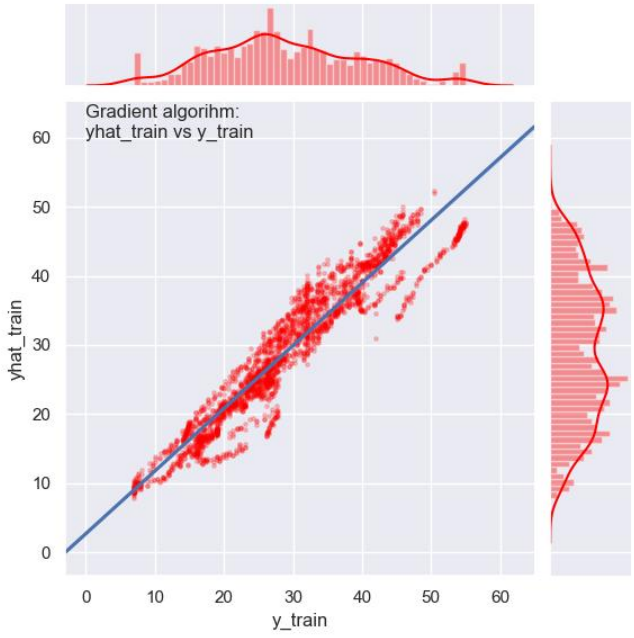


Figure 3.2

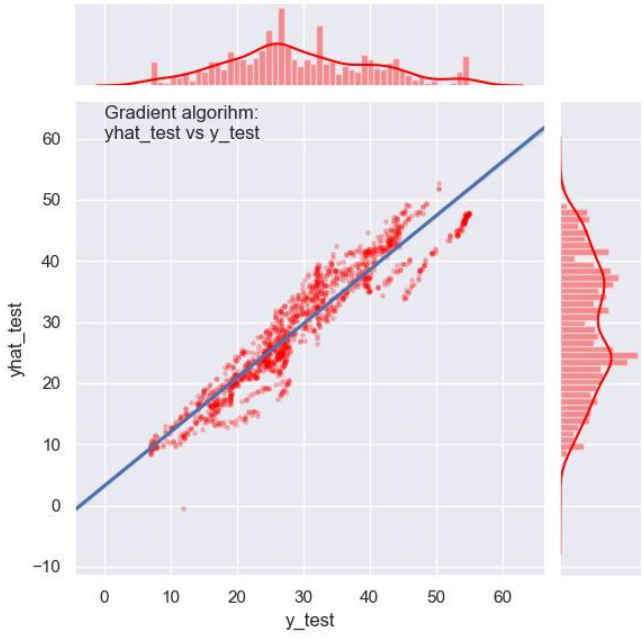


Figure 3.3

*Figure 3.4* and *Figure 3.5* provide the plots for the histograms of the errors **y_train** - **yhat_train** and **y_test** - **yhat_test**. It is possible to notice that errors approximately follow a Gaussian distribution with mean zero. In addition we can see that the maximum error for **y_train** is a little bit much higher than 10, while for **y_test** it is around 12.5 for some instances.

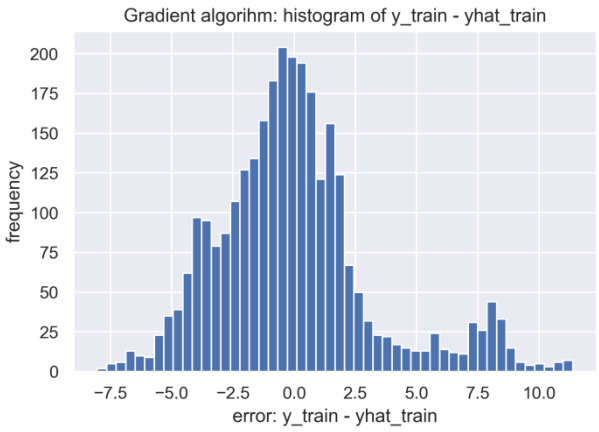


Figure 3.4

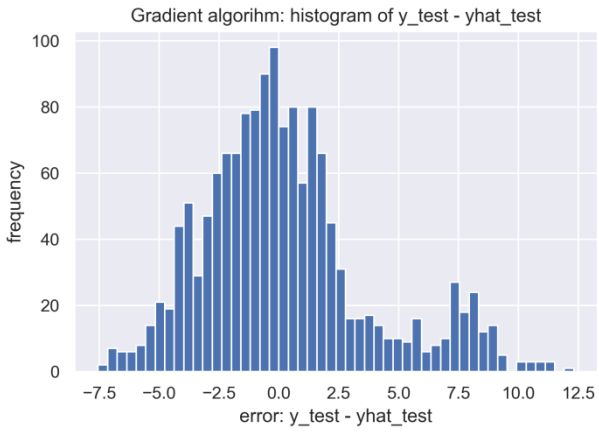


Figure 3.5

The measured mean square errors using the gradient algorithm are:

- Mean square error for the training set: 10.51972
- Mean square error for the testing set: 12.2771
- Mean square error for the validation set: 11.50314

## 4. The stochastic gradient algorithm

With the stochastic gradient algorithm, we want to minimize the function $f(\mathbf{w}) = \sum_{n=0}^{N} f_n(w)$, where each function $f_n(w)$ represent the cost function of the n-th row.
Therefore, the mathematical operations done at the i-th step are: $\mathbf{w}_{i+1} = \mathbf{w}_i - \gamma \nabla f_i(\mathbf{w}_i)$, with $\gamma$ the learning coefficient. As with the gradient algorithm, we have to find correct values for $\gamma$ and for the number of iterations.

*Results on Parkinson data*
The chosen value of $\gamma$ for the regression on Parkinson data is $10^{-3}$.
The number of iterations has been set to 100 thousand, in order to achieve mean square errors similar to the ones obtained with the other algorithms.
The weight vector obtained with the stochastic gradient algorithm is showed in *Figure 4.1*. We can see that its values are oscillating. As usual, motor UPDRS (feature 0) has most of the weight in the regression.

$$\mathbf{w} = \begin{bmatrix} 0.87064282 \\ -0.42148654 \\ -0.05613752 \\ 0.39107156 \\ 0.02909757 \\ -0.11964118 \\ 0.15593513 \\ 0.1138485 \\ -0.33459836 \\ 0.18704368 \\ -0.3952414 \\ 0.39032905 \\ 0.07163028 \\ 0.06240149 \\ -0.26650618 \\ -0.02411977 \\ 0.22865216 \end{bmatrix}$$

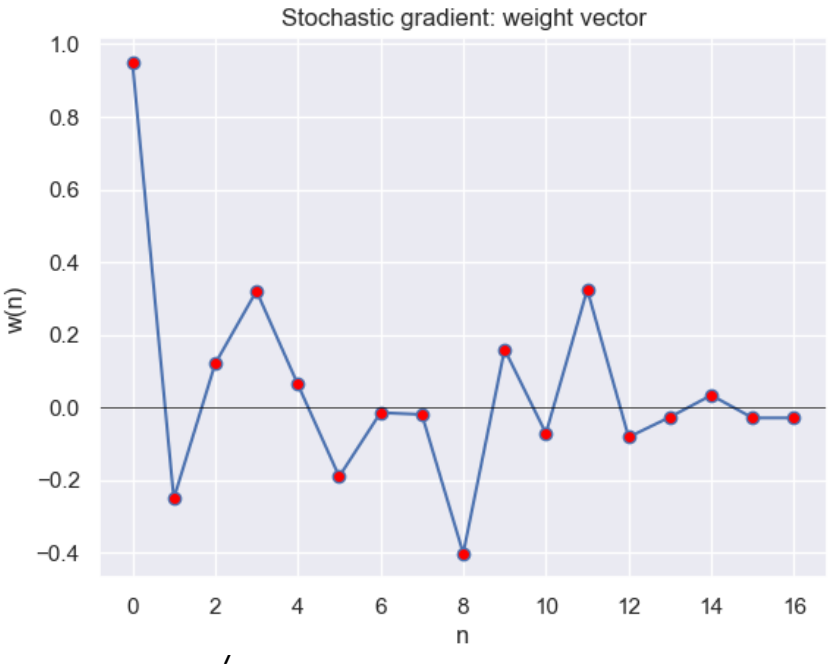


Figure 4.1

In *Figure 4.2* it is possible to notice the plot of **yhat_train** versus **y_train**, while *Figure 4.3* shows the plot of **yhat_test** versus **y_yest.** The marginal charts, at the top and at the right, show the distribution of the 2 variables using histograms and the fitted kernel density estimations (KDE).

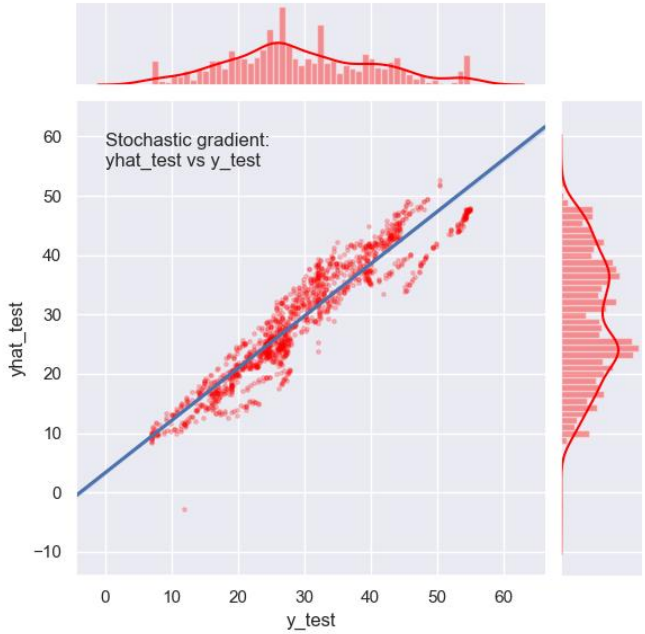


Figure 4.2

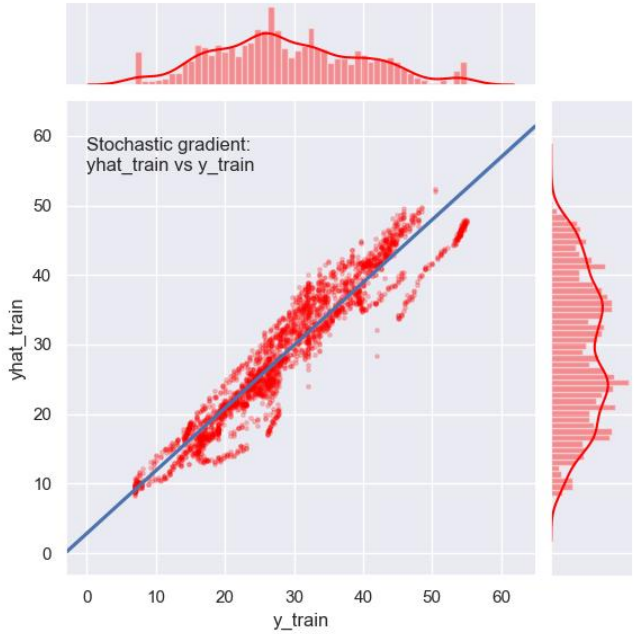


Figure 4.3

*Figure 4.4* and *Figure 4.5* provide the plots for the histograms of the errors **y_train** - **yhat_train** and **y_test** - **yhat_test**. It is possible to notice that errors approximately follow a Gaussian distribution with mean zero. In addition we can see that the maximum error for **y_train** is a little bit much higher than 10, while for **y_test** it is around 15 for some instances.

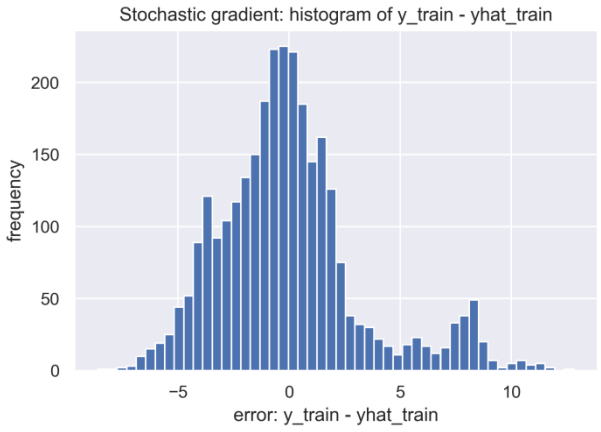


Figure 4.4

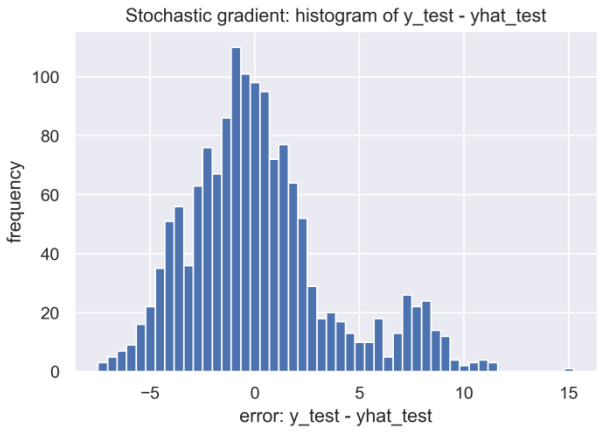


Figure 4.5

The measured mean square errors using the stochastic gradient algorithm are:

- Mean square error for the training set: 10.63319
- Mean square error for the testing set: 12.31955
- Mean square error for the validation set: 11.58485

## 5. The steepest descent algorithm

Using the steepest descent algorithm, the learning coefficient γ is optimized in such a way as that the gradient will always be orthogonal to the one calculated at the previous step:

$$\mathbf{w}_{i+1} = \mathbf{w}_i - \frac{\|\nabla f(\mathbf{w}_i)\|^2}{\nabla f(\mathbf{w}_i)^T \mathbf{H}(w_i) \nabla f(\mathbf{w}_i)} \nabla f(\mathbf{w}_i)$$

where $\mathbf{H}$ is the Hessian matrix.

*Results on Parkinson data*

The only parameter to be set with the steepest descent is the stopping condition, being γ fixed. The number of iterations is 1000 in our case.

The weight vector obtained is showed in *Figure 5.1*. The value of the first feature (the motor UPDRS) is predominant.

$$\mathbf{w} = \begin{bmatrix} 0.95012453 \\ -0.20006979 \\ 0.11235094 \\ 0.29548270 \\ 0.08199399 \\ -0.21471747 \\ -0.06104373 \\ -0.00056113 \\ -0.37426468 \\ 0.11873677 \\ -0.04787136 \\ 0.35136975 \\ -0.07445924 \\ -0.03176169 \\ 0.03559177 \\ -0.03246310 \\ -0.02922539 \end{bmatrix}$$

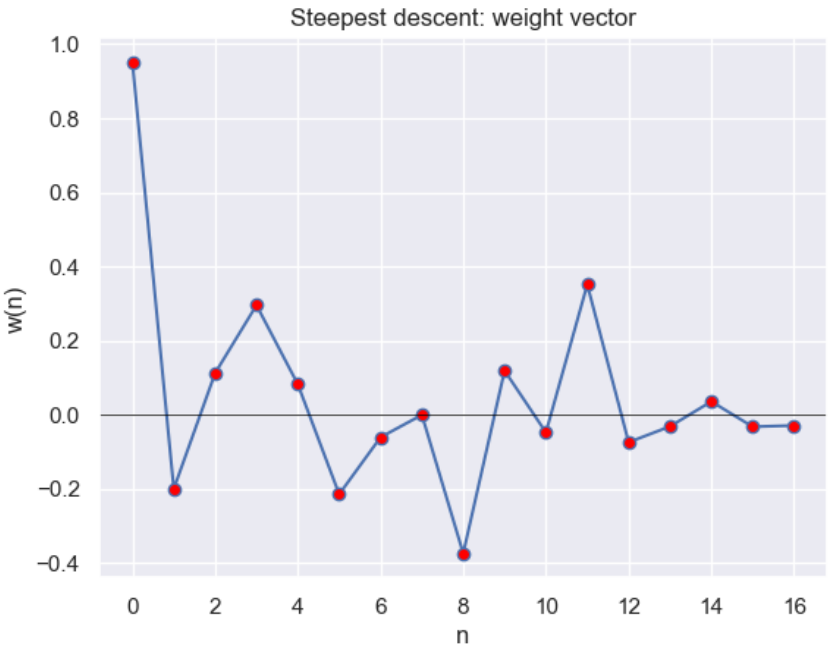


Figure 5.1

In *Figure 5.2* it is possible to notice the plot of **yhat_train** versus **y_train**, while *Figure 5.3* shows the plot of **yhat_test** versus **y_yest.** The marginal charts, at the top and at the right, show the distribution of the 2 variables using histograms and the fitted kernel density estimations (KDE).

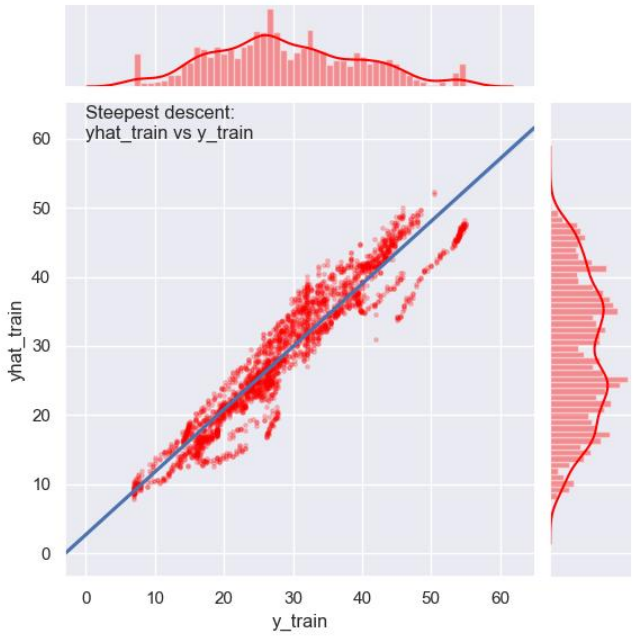


Figure 5.2

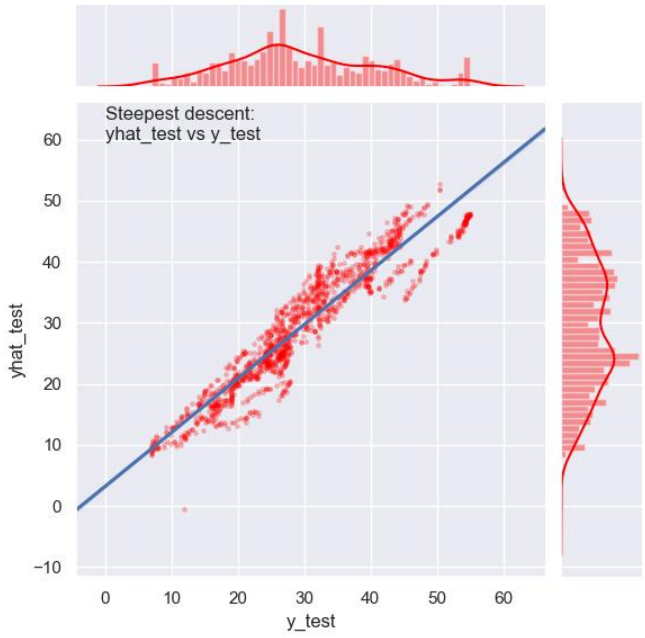


Figure 5.3

*Figure 5.4* and *Figure 5.5* provide the plots for the histograms of the errors **y_train** - **yhat_train** and **y_test** - **yhat_test**. It is possible to notice that errors approximately follow a Gaussian distribution with mean zero. In addition we can see that the maximum error for **y_train** is a little bit much higher than 10, while for **y_test** it is around 12.5 for some instances.

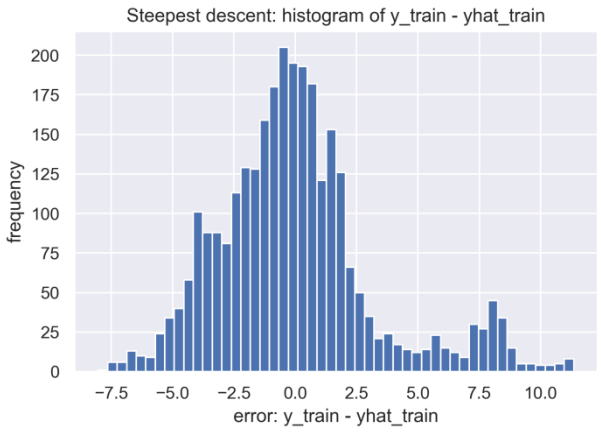


Figure 5.4

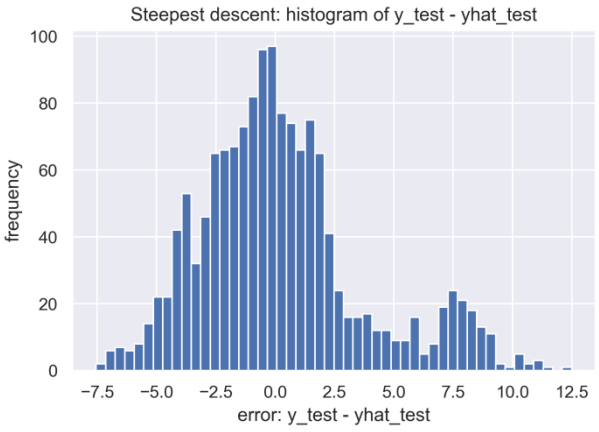


Figure 5.5

The measured mean square errors using the steepest descent algorithm are:

- Mean square error for the training set: 10.52025
- Mean square error for the testing set: 12.26328
- Mean square error for the validation set: 11.50543

## 6. The ridge regression

Ridge regression aims to find the values of the vector $\mathbf{w}$ in order to minimize the constrained function $f(\mathbf{w}) = \|\mathbf{y} - \mathbf{X}\mathbf{w}\|^2 + \lambda\|\mathbf{w}\|^2$, where the coefficient $\lambda$ has to be set properly.

Evaluating the gradient of $f(\mathbf{w})$ and setting it to $0$, we get the following minimum weight vector: $\widehat{\mathbf{w}} = (\mathbf{X}^{\mathrm{T}}\mathbf{X} + \lambda\,\mathbf{I})^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{y}$, where $\mathbf{I}$ is the identity matrix.

*Results on Parkinson data*

The value of $\lambda$ has been tuned considering the plot of the mean square error of the validation set (*Figure 6.0*). As can be seen, $\lambda = 4$ is a reasonable choice.

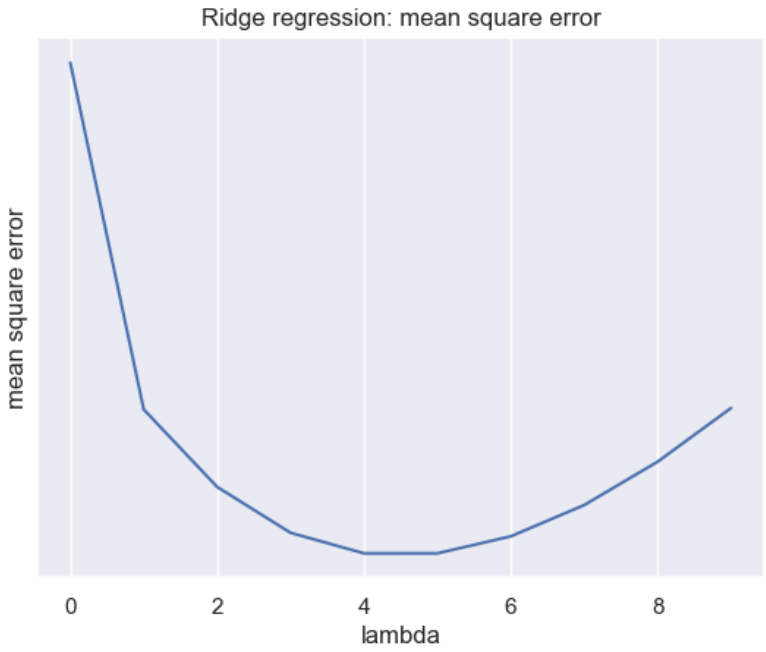


Figure 6.0

The weight vector obtained is showed in *Figure 6.1*. As we expected, its values are greatly reduced compared to the ones obtained with the LLS.

$$\mathbf{w} = \begin{bmatrix} 0.94838671 \\ -0.13124158 \\ 0.10542722 \\ 0.01463047 \\ 0.07201228 \\ 0.01614633 \\ -0.12896534 \\ 0.01685197 \\ 0.01234279 \\ 0.1024854 \\ -0.0337295 \\ 0.01257398 \\ -0.0680294 \\ -0.03256007 \\ 0.03534705 \\ -0.03154125 \\ -0.03359112 \end{bmatrix}$$


Figure 6.1

In *Figure 6.2* it is possible to notice the plot of **yhat_train** versus **y_train**, while *Figure 6.3* shows the plot of **yhat_test** versus **y_yest.** The marginal charts, at the top and at the right, show the distribution of the 2 variables using histograms and the fitted kernel density estimations (KDE).

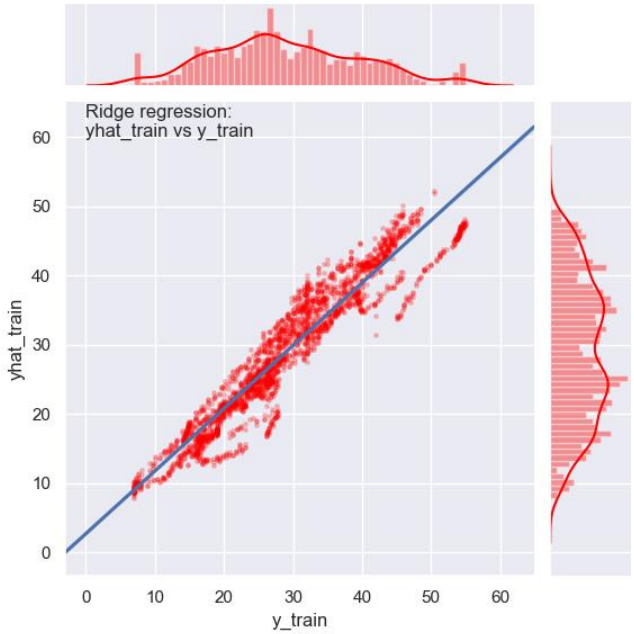


Figure 6.2

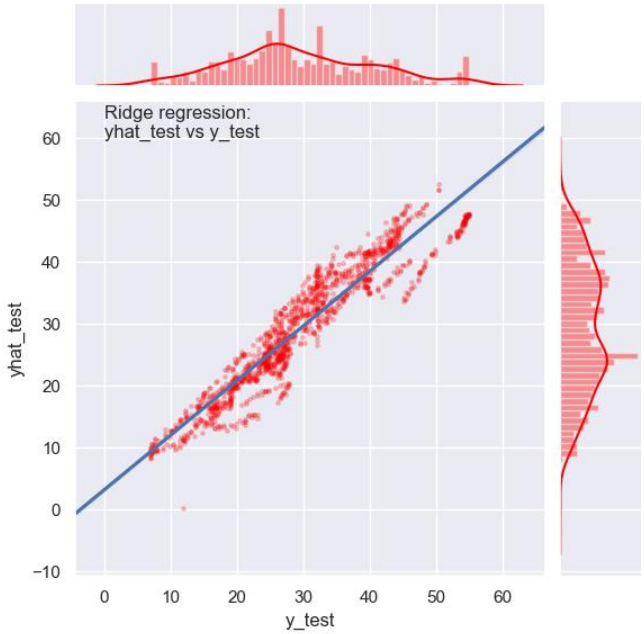


Figure 6.3

*Figure 6.4* and *Figure 6.5* provide the plots for the histograms of the errors **y_train** - **yhat_train** and **y_test** - **yhat_test**. It is possible to notice that errors approximately follow a Gaussian distribution with mean zero. In addition, we can see that the maximum errors for both **y_train** and **y_test** is a bit higher than 10.

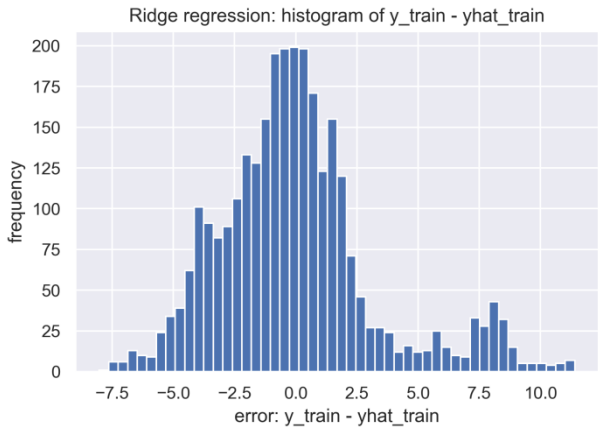


Figure 6.4

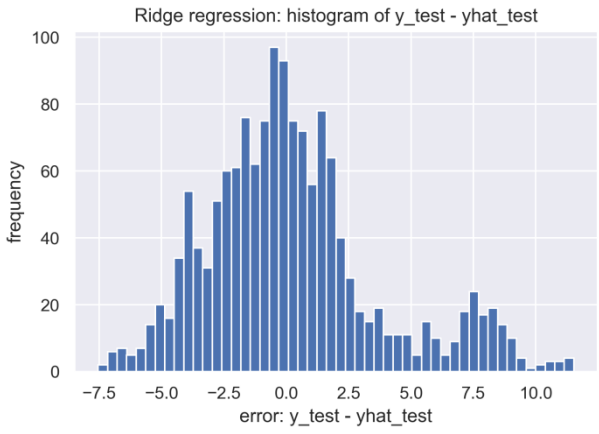


Figure 6.5

The measured mean square errors using the ridge regression algorithm are:

- Mean square error for the training set: 10.5213
- Mean square error for the testing set: 12.26884
- Mean square error for the validation set: 11.50722

# Conclusions and comparisons

The findings show very different weight vectors resulting from the application of each method, because the minimum of the cost function could be very “flat” in a 17th dimensional space like the one analyzed. Anyway, they all show a good level of validity basing on the purpose of the regression.
All of the techniques have both pros and cons.
LLS is non-iterative, but it could be expensive from the computational point of view because of the evaluation of the pseudo-inverse matrix. Moreover, overfitting may occur. So, generally speaking, it could be too precise on the training set, less on the testing and the validation ones.
Conjugate gradient’s best quality is the fact that it converges at most in N steps (being N the number of features). Indeed, it only takes 17 steps to obtain errors comparable with the ones reached by the gradient algorithm in 10000 steps or by the steepest descent in 5000 steps. It does not require to compute any matrix inverse, but only matrices and vectors multiplications. Although it makes more computations than steepest descent at each iteration, far fewer steps are required.
Gradient algorithm usually requires a very large number of iterations to get a suitable solution. So the fact that it does not need matrix inversions might not be enough.
The stochastic gradient method appears to be the worst solution in our analysis: compared to the other iterative methods it requires much more iterations to get similar errors.
Using the steepest descent, we have to consider the computation resources required for the evaluation of the optimum learning coefficient. Anyway, it is exactly this optimization that make it faster than the gradient algorithm to converge.
At the end, ridge regression. It shares pros and cons with LLS, but the shrinkage of the pseudo-inverse matrix could prevent overfitting problems.