EARIN

Laboratory report

EXERCISE 4: Regression and classification

Paweł Borsukiewicz & Bartłomiej Mastej

Table of contents

[1. Introduction 1](#_Toc101523440)

[2. Implementation of Random Forest 1](#_Toc101523441)

[3. Implementation of SVM 4](#_Toc101523442)

[4. Comparison of results 4](#_Toc101523443)

# Introduction

For the purpose of the laboratory two regression methods – Random Forest and SVM – were used to predict the house prices. Versions with and without hyperparameter optimization were considered, analysed and assessed.

# Implementation of Random Forest

In order to prepare the Random Forest algorithm, data had to be pre-processed prior to training process, as presented below.

Obraz zawierający tekst

Opis wygenerowany automatycznie

**Figure 1. Dataset pre-processing**

Firstly, data was read from the csv file and then separated into two arrays provided by numpy library. Column named “price”, storing values to predict, was cut from the original dataset and stored in the second array. Finally, data in column “data” had to be converted to a format that could be interpreted by the model.

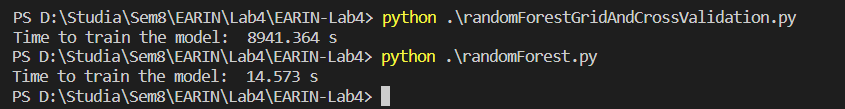
Then data was randomly split into training and test set with the same seed used throughout all experiment to provide repeatability. 75% of the data was used as a training set. Division and implementation of the algorithm was done using scikit-learn library.

Obraz zawierający tekst

Opis wygenerowany automatycznie

**Figure 2. Division, training and predicting**

As it can be observed on figure 2, timer was also added to calculate time needed to perform computations. After using fit() function to train the model, predict() function was used to predict values of “prize” metric in test set. Second version of algorithm, presented on figure 4, was implemented with addition of grid search and 5-fold cross validation. Due to the time-related issues, only three parameters were affected by grid search, which for each of them had two options to choose from. While program without hyperparameter optimization run for around 15s, optimization extended it to over 2 hours.



**Figure 3. Training times**

Obraz zawierający tekst

Opis wygenerowany automatycznie

**Figure 4. Algorithm with grid search and cross validation**

Finally, results were saved to csv file. Additionally, mean absolute error and mean square errors were calculated.

Obraz zawierający tekst

Opis wygenerowany automatycznie

**Figure 5. Error calculation and storing result**

What is more, exemplary plots of price vs house size were created with the use of matplotlib library to help visualize predictions. First one was used to show general trend, while the second one rather focuses on individual predictions.

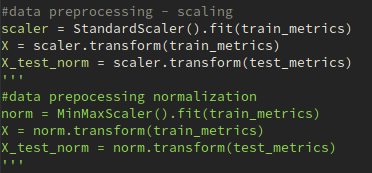
Obraz zawierający tekst

Opis wygenerowany automatycznie

**Figure 6. Plots**

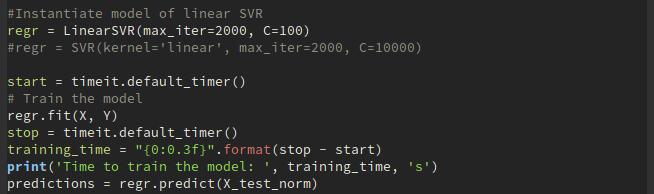
# Implementation of SVM

The main difference in implementation is in the data pre-processing. The SVM algorithms are very sensitive to the data variety as it is using distances between points to determine similarity. As the data in the dataset variates significantly, thus there was a need for scaling to reduce the effect of huge data differences. Consequently, there were used two methods of scaling: standardization and normalization, however, each of them had to be used exclusively at one time.

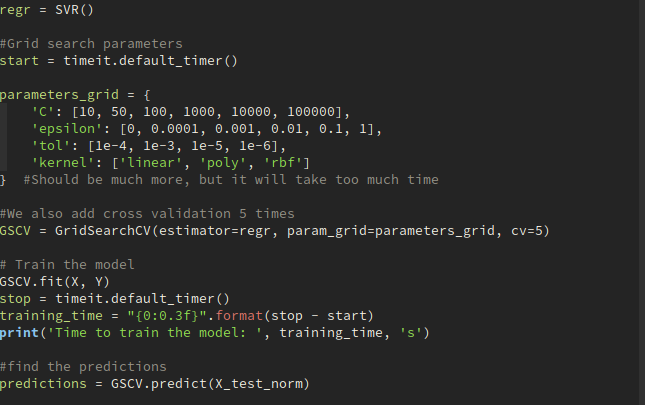


**Figure 7. Standardization and normalization**

Subsequently, there was performed the same scheme as in random forest – the model was created, and there was performed it’s training which can be seen on the figure 8. What turned out to be crucial was the regularization parameter - if it was smaller than 100 it did not converge and as a result, the model was completely inaccurate. In the end the results were saved to a file and the plots were created in the same manner as it was showed previously on the figure 5 and figure 6.

**Figure 8. Division, training and predicting - SVM**

Further, there was performed hyperparameter optimization using grid search and 5-fold cross validation that can be seen on the figure 9. For the scope of more extensive testing in stead of using LinearSVR as it was done earlier, there was used SVR class as it enables to use not only linear type of kernel, but also the RBF (radial basis function) and polynomial kernel types. LinearSVR is slightly more optimized than the SVR with linear kerenl set. When it comes to timing, it takes only 0.008s LinearSVR in comparison to the RESULT OF GRID SEARCH.

**Figure 9. Algorithm with grid search and cross validation for SVM**

# Comparison of results

To begin with, when the mean absolute errors and mean square errors are considered, following results were obtained.

|  |  |  |
| --- | --- | --- |
|  | Mean absolute error | Mean square error |
| Random Forest without  hyperparameter optimization | 71744.74 | 1.95E+10 |
| Random Forest with  hyperparameter optimization | 71630.02 | 1.93E+10 |
| SVM without  hyperparameter optimization | 129380.57 | 6.59E+10 |
| SVM with  hyperparameter optimization |  |  |

**Table 1. Errors comparison**

It can be observed that hyperparameter optimization decreases errors. Nonetheless, change is not significant in case of Random Forest. It can be observed that RF is about three times more accurate than the SVM which is a rather significant value. COŚ O SVM. Porównanie z RF. COŚ o tym, że average value dla mojego datasetu to 546081.7221, czyli w sumie jest zauważalny średni błąd dla RF, na oko tak z 14%. Ale tragedii nie ma.

State presented by means can be easily observed when individual results are concerned as in table 2. Accuracy of prediction varies, however, optimized prediction seems to be at least slightly closer in its prediction in high majority of cases.

Obraz zawierający stół

Opis wygenerowany automatycznie

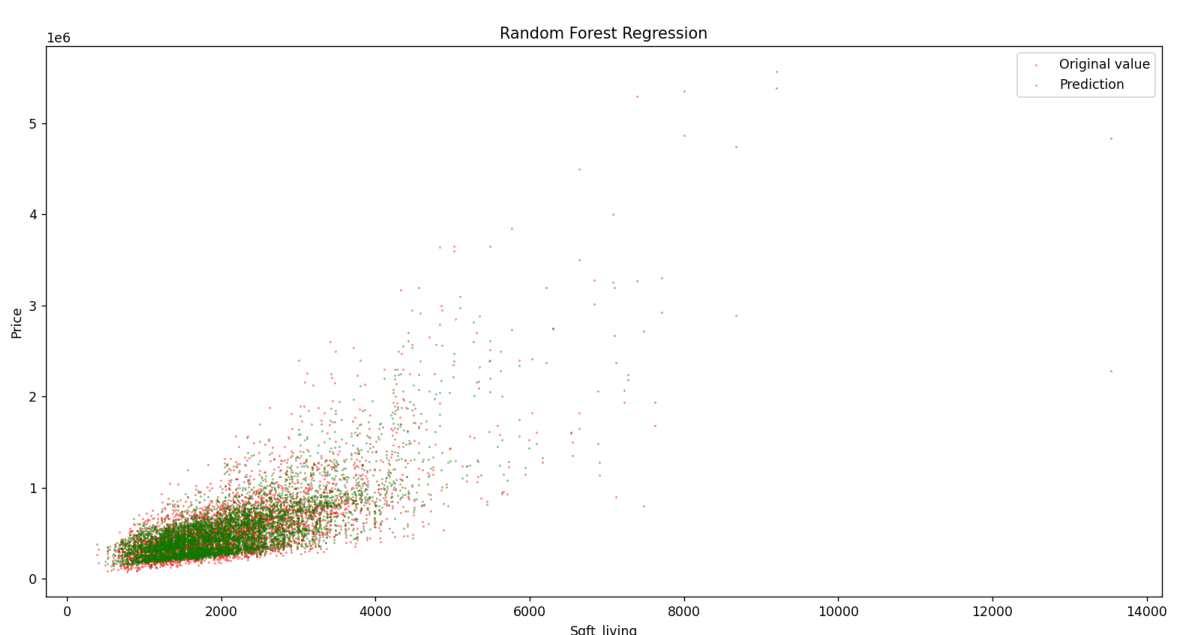
**Table 2. Random Forest - comparison of exemplary results**

CHYBA ZNOWU COŚ O SVM

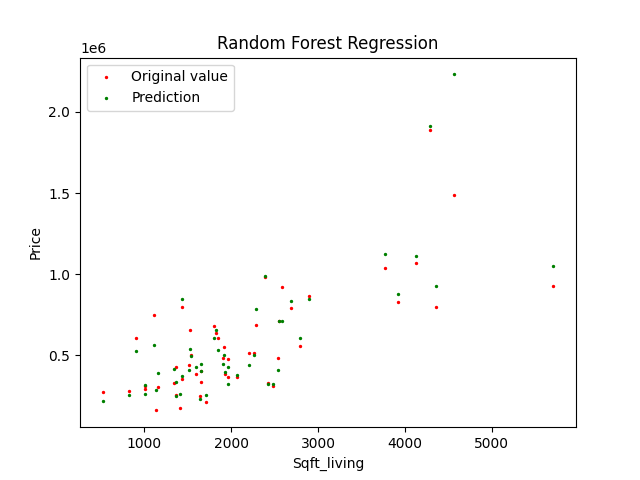
Further, we could also analysed results presented on the graphs.



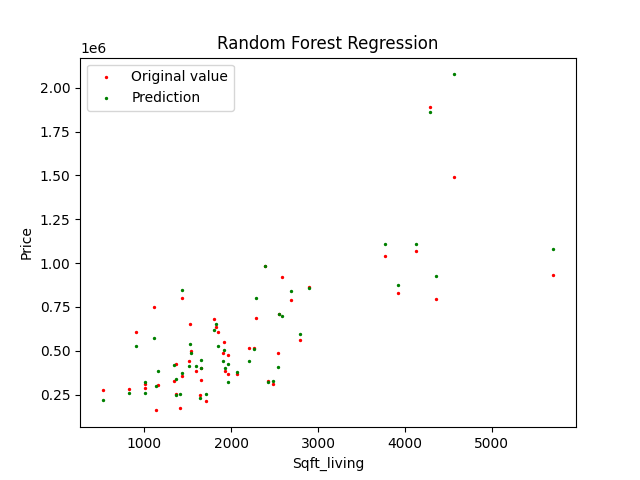
**Figure ?. Price vs house size, RF with no hyperparameter optimization, general overview**



**Figure ?. Price vs house size, RF with hyperparameter optimization, general overview**

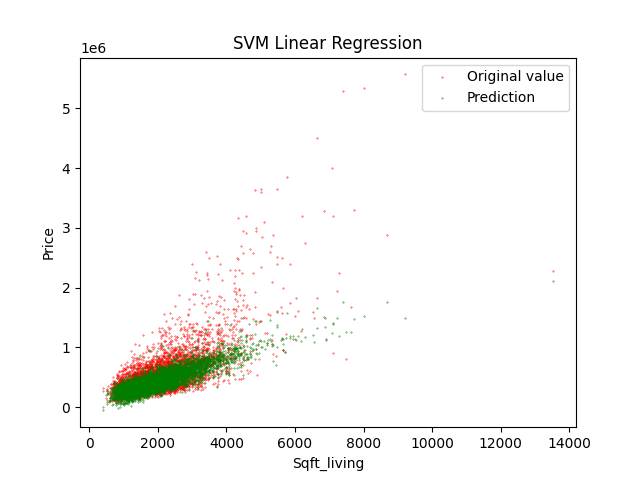


**Figure ?. Price vs house size, RF with no hyperparameter optimization, 50 random samples**

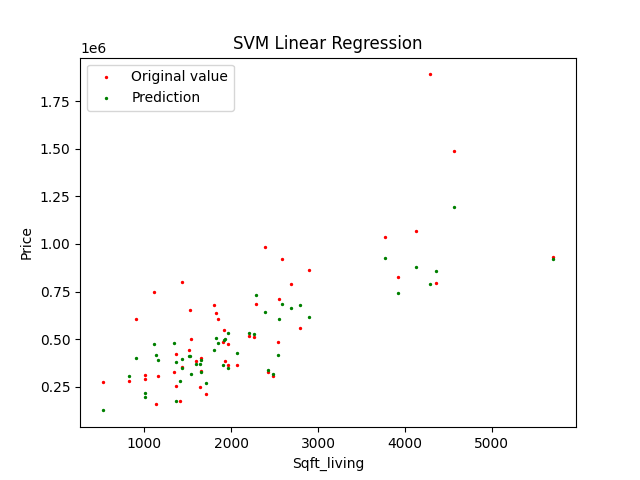


**Figure ?. Price vs house size, RF with hyperparameter optimization, 50 random samples**

It can be observed that in case of random forest general trend seems to be followed and without numeric values it is almost impossible to spot any difference between the version with and without hyperparameter optimization. When only 50 samples are presented it is clearly visible that the predictions are not uniform, some of them are very good and overlap with actual values, however, there are also instances of results differing significantly.



**Figure ?. Price vs house size, LinearSVR with no hyperparameter optimization, general overview**



**Figure ?. Price vs house size, LinearSVR with no hyperparameter optimization, 50 random samples**

As it can be clearly seen, the random forest algorithm coverage is much greater than the SVR algorithm, especially in comparison to the LinearSVR, and it is also reflected by the mean absolute error and the mean square error, as it was earlier indicated the LinearSVR is about 3 times less accurate. Nevertheless, LinearSVR is much faster than the RF – the runtime of the pure algorithm without optimization is 1360 times faster when it comes to LinearSVR.

ZNOWU COŚ O SVM. I JAKIEŚ PORÓWANIE.

Dopisać o hyperparameter optimization dla SVM