Norwegian Business School

GRA65182 - Data Science for Finance
Assignment - Linear Regression vs XGBoost

April 22, 2024

Data Generation

We first generated our data from the following model :

$$y = \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_4 x_4 + \epsilon$$

where $\beta = [2, 1.5, 1, 0.5]$ and the $x_i \sim \mathcal{N}(0, 1)$, i.i.d and are such that $corr(x_i, x_j) \neq 0$ as you can see from the correlation Matrix:

Table 1. Correlation Matrix

	x_1	x_2	x_3	x_4
x_1	1.	-0.014	-0.005	-0.002
x_2	-0.014	1.	-0.009	-0.021
x_3	-0.005	-0.009	1.	-0.009
x_4	-0.002	-0.021	-0.009	1.

Finally we set ϵ such that $V(\epsilon) = 1$

Comparing OLS and XGBoost on this Linear Data set.

After fixing the learning rate to 0.1 and the max depth to 4 for XGBoost. We fit the model on our training set of 10 000 observations and we cross-validated on the validation set of 5000 observations to find the optimal numbers of trees.

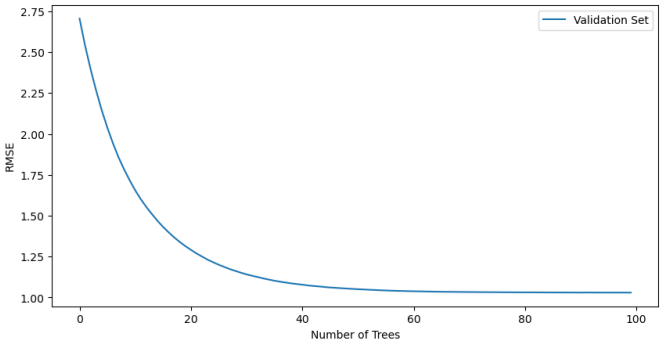

Figure 1. Optimal number of trees according to RMSE : **100**

Even on a Linear data set XGBoost perform almost as well as OLS as you can see from the out of sample (OOS) R^2 and RMSE that are really close in **Table 2**.

Table 2. Performance Metrics

	OLS	XGBoost
R^2 OOS	0.882	0.874
RMSE OOS	1.00	1.03

The XBGoost model explain 87% of the variance in the dependent variable which shows really good predictive power especially for a model that is non-linear. The slightly difference in RMSE can be interpreted by the fact that XGBoost optimize for different aspects of the data, handling non-linearity and complex interactions, that here are not useful to fit linear data.

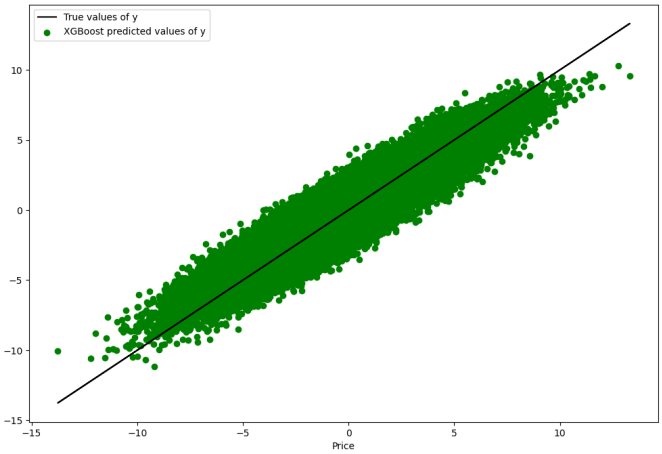

Figure 2. Actual vs Predicted Values of y on Test set for XGBoost Model

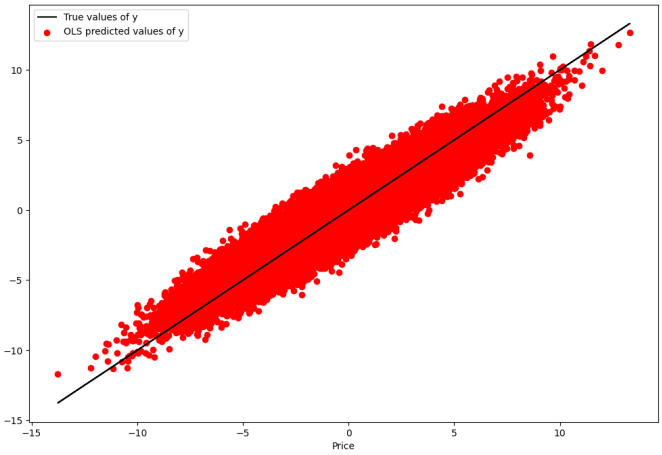

Figure 3. Actual vs Predicted Values of y on Test set for OLS Model

Potential Optimization of XGBoost.

We have already cross-validated the optimal number of tree but we could also cross-validate the maximum depth of the tree in order to determine the optimal level of complexity to fit the data as you can see from **Figure 4** the optimal level of complexity is 4, since after this level the RMSE start to increase again. Therefore we can not increase the performance of XGBoost based on this parameters.

Table 3. RMSE of maximum depth of the trees

	RMSE
$max\_depth = 1$	1.243
$max\_depth = 2$	1.056
$max\_depth = 3$	1.032
$max\_depth = 4$	1.029
$max\_depth = 5$	1.037

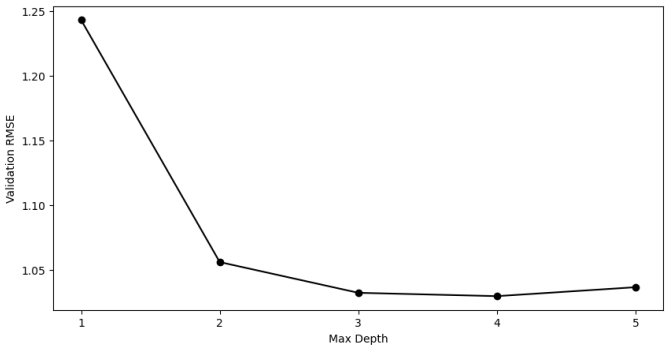

Figure 4. Optimal Maximum Depth of the trees

Log transformation of the data and comparation of XGBoost and OLS.

We now apply a log transformation to our data generation process, so that our new model become:

$$log(y) = \beta_1 log(x_1) + \beta_2 log(x_2) + \beta_3 log(x_3) + \beta_4 log(x_4) + \epsilon$$

We again fit XGBoost and OLS to this new model and we observe the following results:

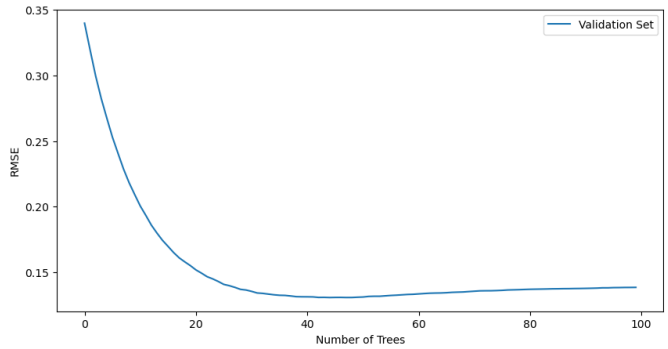

Figure 5. Optimal number of trees according to RMSE : **45**

Here again we observe very similar results between OLS and XGBoost with still a slightly better performance from OLS as you can see from the **Table 4**. Note that with the log transformation the scale of the RMSE as significantly change and that is why we observe much lower RMSE compared to the first model.

Table 4. Performance Metrics on the test set

	OLS	XGBoost
R^2 OOS	0.857	0.853
RMSE OOS	0.079	0.080

We can notice that even though both models still fit the data pretty well, the R^2 slightly decreased compared to Model 1. This is primarily due to the fact that logarithmic transformation is very useful when linearizing the relationship between variables, especially when dealing with exponentials. However, in this case, we applied a logarithmic transformation to a dataset that was already perfectly linear. As a result, applying the logarithmic transformation actually deteriorated this linear relationship.

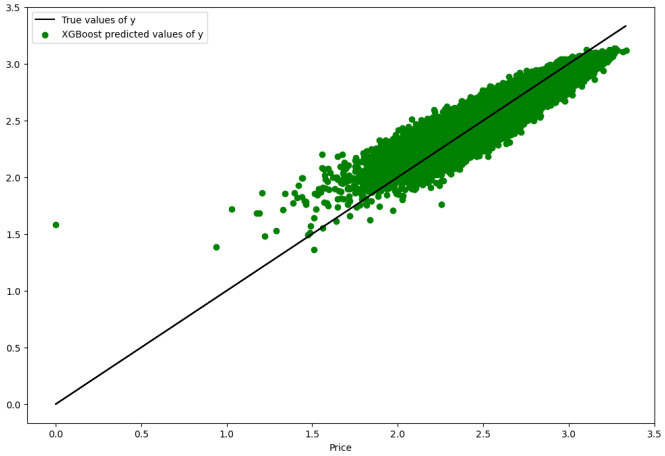

Figure 6. Actual vs Predicted Values of y on Test set for XGBoost Model

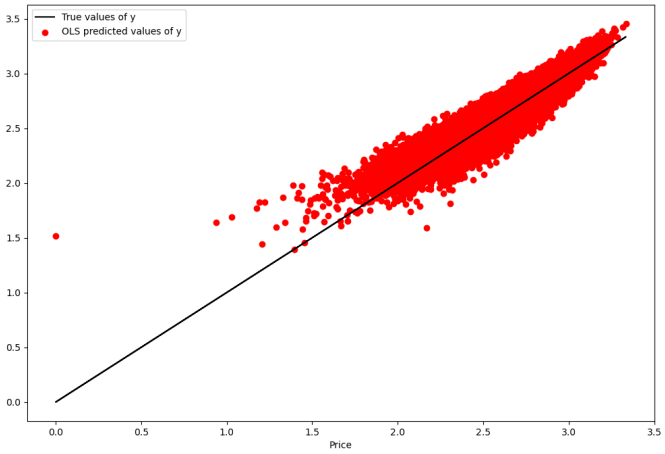

Figure 7. Actual vs Predicted Values of y on Test set for OLS Model

Potential Optimization of XGBoost

We can here, also try to find a more optimal level of complexity to fit the data by again cross-validating the maximum depth of the tree. Here we find that the optimal depth would be 2. If we now run again the XGBoost model with a maximum depth of 2 we actually do slightly better than OLS with a R^2 of 0.860 and a RMSE of 0.078. This result show that as linearity deteriorates OLS deteriotes too and XGBoost become better since it can handle non-linear relation ship between variable.

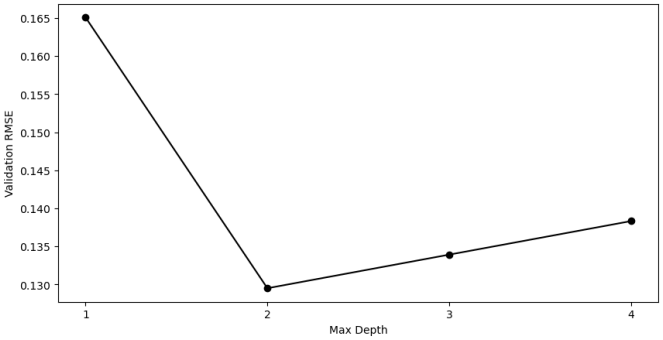

Figure 8. Optimal Maximum Depth of the trees

Table 5. RMSE of maximum depth of the trees on validation set

	RMSE
$max\_depth = 1$	0.165
$max\_depth = 2$	0.130
$max\_depth = 3$	0.134
$max\_depth = 4$	0.138

Introduction of Leverage points in the train data

In this section we use the same model as before but this time we set the first 10 values of x_1 to be equal to -100. By doing so we introduce extreme values in the training set and we want to see how OLS and XGBoost are going to handle such extreme values. First we fit XGBoost to cross-validated the number of optimal trees:

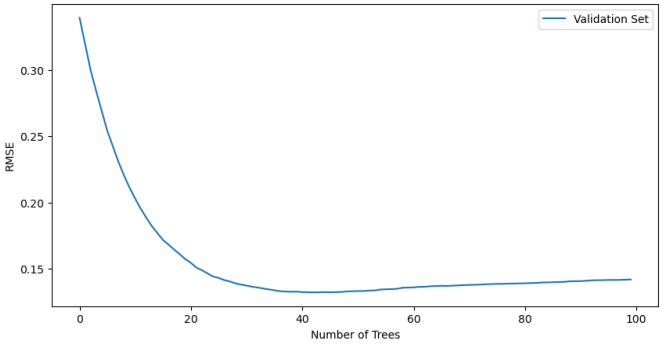

Figure 9. Optimal number of trees according to RMSE : **43**

Table 6. Performance Metrics on the test set

	OLS	XGBoost
R^2 OOS	0.341	0.849
RMSE OOS	0.170	0.080

We can clearly see that the introduction of leverage points completely misled the OLS model. The R^2 decreased a lot whereas the R^2 of XGBoost stayed almost the same. This give us very important insight on OLS and XGBoost. First OLS, which is a paramteric model that attempt to minimize the sum of squared residuals is really affected by leverage points since it amplifies the impact of any error associated with these points on the overall model fit. Those points pull the regression line toward themselves, its distort the estimated relationship between variables across the entire dataset. As you can see on **Figure 10** where we computed the standarized importance of each feature we should normally observe a higher importance to the features associated to a higher beta. However what we observe with the introduction of leverage points is that x_1 has the lowest variable importance while it should have the highest. It reflects well how those extremes values in x_1 impact the global fit of the model, this is futher seen in **Figure 11**.

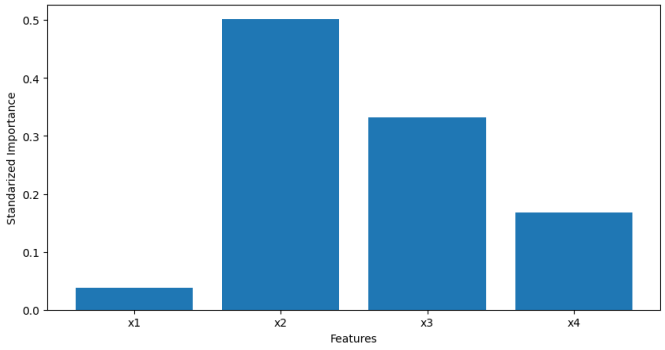

Figure 10. standarized feature Importance in the OLS model

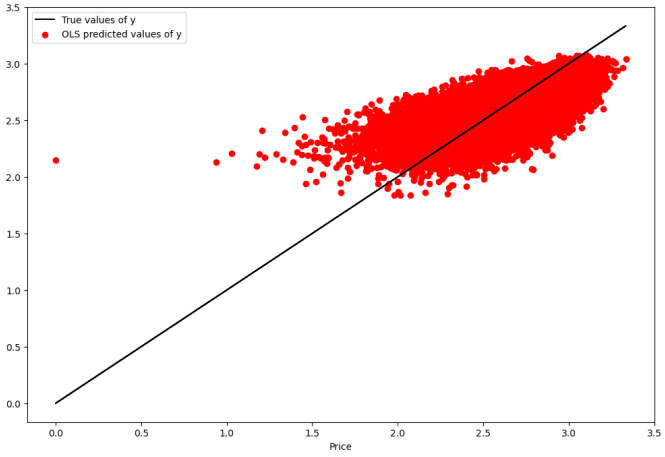

Figure 11. Actual vs Predicted Values of y on Test set for OLS Model

Secondly for XGBoost, since it is a gradient boosting model that is using decision trees as base learners, this model does not have any assumption on how the data should be distributed contrary to OLS. It create the model by learning splits based on the distribution and the relationship between features to the target variable in the data. This approach make it very much less sensitive to leverage points in predictor variables. That explains the fact that it R^2 and RMSE are almost the same.

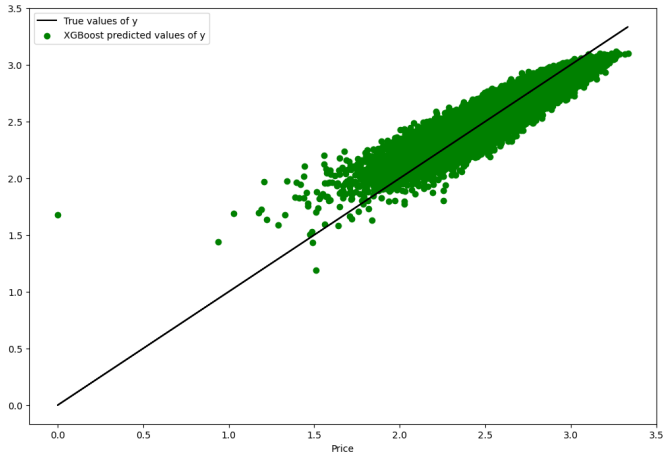

Figure 12. Actual vs Predicted Values of y on Test set for XGBoost Model

Potential Optimization of XGBoost

We can again try to optimize the fit of XGBoost by cross-validating the maximum depth paramters. In this set up we end up with an optimal depth of 2 like in the previous model, which demonstrates that the presence of leverage points does not require more complex data handling.

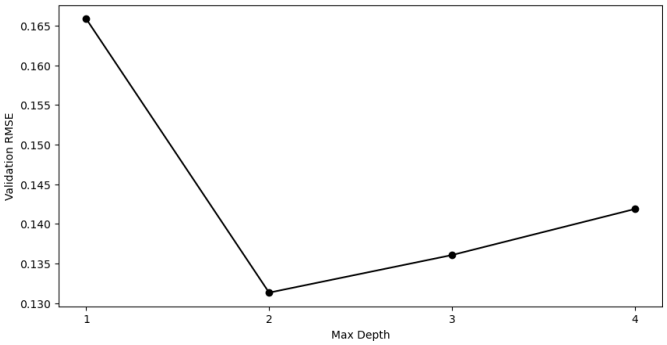

Figure 13. Optimal Maximum Depth of the trees

Table 7. RMSE of maximum depth of the trees on validation set

	RMSE
$max\_depth = 1$	0.166
$max\_depth = 2$	0.131
$max\_depth = 3$	0.136
$max\_depth = 4$	0.142

By running again XGBoost with the optimal maximum depth we can, as in the previous model, improve slightly the performance of the model with a R^2 of 0.860 and a RMSE of 0.078.

Introduction of Multicolinearity in the data set.

In this Section we go back to our log transfrom data set and we are going to introduce a fifth variable defined as $x_5 = x_2 - x_3 + u$ where $u \sim \mathcal{N}(0, 0.001)$. We want to observe how OLS and XGBoost handle features when there are multicolinearity issues.