Boosted GARCH for VaR forecasts

Student Number: 31912

January 20, 2024

Contents

1 Introduction

Recently, Artificial Intelligence has been thrown from the basements of academia and technology companies to the hands of the public. With the rise of chatgpt, the public is stating to become well familiarized with the technology. Although a lot of publicity has been offered to the area of deep-learning, its more approachable supervised-learning segment has often been forgotten. The goal of this paper is to assess the validity of a well-known and easily applicable Supervised Learning algorithms to risk management measures. To that end, we restrict our analysis to a boosting algorithm named XGboost and compare it with a standard GARCH model. We compare both their volatility forecasting capabilities as well as Value-at-Risk forecasts. We leave out Expected-Shortfall from the analysis, since, given an assumption of normality, VaR and ES provide the same information. The paper is divided into three different sections. First, we explain the theory behind boosting algorithms in general as well as review the literature attempting an endeavour similar to ours. Secondly, we explain specifically our methodology. More precisely, we review the estimation methods that are used as well as the tools to compare our two models. The third section evaluates the accuracy of our estimations. We use our volatility estimates to calculate VaR forecasts and evaluate them for the desirable properties one should find, in effect, independence and proper frequency of the violations. Our thesis, is the following: boosting a simple GARCH(1,1) algorithm using XgBoost will reduce estimation bias of volatility and thus provide a better forecasting power of the Value-at-Risk.

2 Theoretical Overview

2.1 Introduction to boosting

We first provide an introduction to the vocabulary and mathematical grounding of boosting methods so that the reader understands the following literature review.

The new method we use in this investigation is classified under the machine learning umbrella of *Ensemble Methods*. These methods use multiple models, called weak learners and aggregates them to create strong learners. Weak learners are models which perform on average only slightly better than a random guess. This idea of "collective intelligence" dates back to Schapire's paper *The Strenght of Weak Learnability* where he describes a method "for converting weak learning algorithm into one that achieves arbitrarily high accurancy." Schapire (1990)

Most of the information for the XGBoost algorithm was extracted from Chen and Guestrin (2016)'s XGBoost documentation for R.

Three pillars form the basis of the XGBoost algorithm:

Equation 1: Model

$$\hat{y}_i = \sum_{k=1}^{K} \alpha_k f_k(x_i), f_k \in F \tag{1}$$

Equation 2: Loss Function

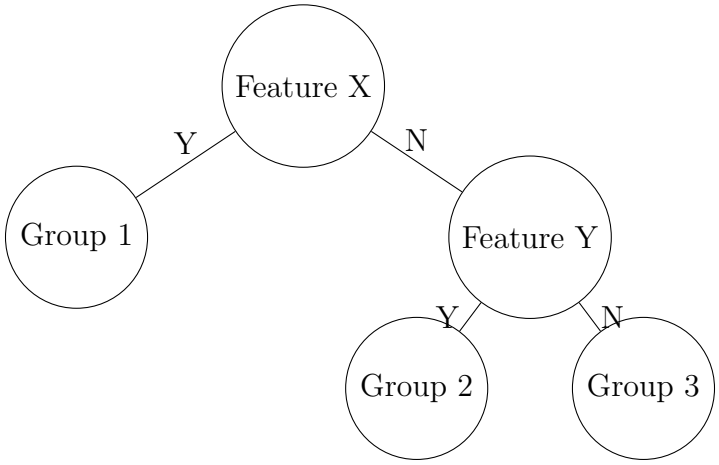
$$L(\theta) = \sum_i (y_i - \hat{y}_i)^2 \tag{2}$$

Equation 3: Regularization Function

$$\Omega(\theta) = \sum_{k=1}^{K} \omega(f_k) \tag{3}$$

Equation (1) represents the linear combination of weak learners f_k, each a part of the set of possible classification or regression trees, otherwise called CART. CARTs can be represented in the following way, and although the provided example is specifically tailored to classification purposes, a regression tree would follow the same structure, with regression

criteria instead of categorical ones.



The above pictured tree is the combination of two weak learners, features X and Y. Mathematically, the above tree is f_1 according to our notation. At each iteration, the algorithm creates a new tree. It is the aggregation of all the trees which make the final classification decision. Equation (2) is the loss function, which we try to minimize. The loss function can take a different form than the usual mean square depicted above. In the given example, say y_1 is missclassified as a part of group 1, when really it should be part of group 2, then $y_1 - \hat{y_1}$ takes a non-zero value. Conversely, the difference is equal to zero whenever y_i is correctly classified. Equation (3) is called a regularization function and is an important step in reducing the over-fitting which plagued earlier versions of ensemble methods such as AdaBoost, Meir and Rätsch (2003). Theoretically, one could have a combination of very complex trees which perform with a high degree of accuracy in sample, but poorly out of sample. Regularization then introduces the term Ω which penalizes the loss function with each added tree. Together with equation (2), they form the objective which we minimize. Within equation (3), the value of ω rises with the degree of complexity of f_k, which worsens our loss. The algorithms thus tries to find a balance between accuracy, captured by (2) and over-fitting, captured by (3).

The classic GARCH model is not necessarily a weak learner, and does not suffer from

complexity or over-fitting issues. Why do we believe that our approach will provide a better forecasting power than the simple GARCH? The answer lies in the following step. The interesting step in the boosting algorithm is that at each iteration, the weights α_k are chosen such that the derivative of the loss function with respect to the failed data-point estimation is higher than in the previous iteration. In other words, the loss-function puts more weight on the data-points which were incorrectly estimated in the previous iteration, Meir and Rätsch (2003).

Our thesis partially rests on this step: the most difficult volatility forecasts are those in periods of high stress. That is a common problem in volatility estimation with more simple models like the ones in the GARCH family: that they often fail in periods of crisis. For example, returns exceeding the Value-at-Risk estimation tend to cluster around periods of intense volatility like the Covid-19 pandemic or the 2008 Great Financial Crisis, violating independence. The algorithm should in theory perform a better forecast than the simple GARCH model during these time-periods, thanks to the re-weighting step. The second reason we believe our approach dominates the GARCH model is bias reduction. Meir and Rätsch (2003) mention in their *Introduciton to boosting and leveraging* that boosting helps reduce bias in simple models while bagging, another subset of *Ensemble methods*, helps reduce over fitting. Since we wish to reduce the bias of a "not-complex-enough" model, we use the boosting approach.

2.2 Literature Review

Researchers from the university of Santiago evaluate the performance of boosting methods with standard GARCH as base hypothesis as well as neural networks for the prediction of heteroskedastic time series Matías et al. (2010). In the boosting segment of their investigation, the authors find that the boosted model with a loss likelihood function outperforms simple ARMA-GARCH models estimated using Maximum Likelihood. We follow in part their methodology, by using simulated data as inputs. However, we favor the mean squared

error to evaluate our boosted model. While the authors restrict their analysis to conditional mean and variance estimation, we do not attempt to model mean returns and focus our analysis on conditional variance and Value-at-Risk, which they do not evaluate. On all generated data series, at least one model outperformed the simple GARCH, but none universally outperformed.

A second paper worth noticing is by Blom et al. (2023). They compare boosted Quantile Regression-Implied Moments with neural networks as well as a set of standalone Random forest and XgBoost tree methods when estimating Value-at-Risk on the Euro-USD Currency Cross using implied volatility from ATM options contract. Comparing and constrating with our methodology, they combine use a boosted quantile regression, which is closely related to our boosted GARCH method. The authors use 10 years of data which they split into training and testing sub samples at the 80% and 20% marks for all methods. At the 5% VaR, the authors find that the two XgBoost tree methods have an expected violation ratios of 5.47 and 7.01 and outperformed Neural Network models. The boosted quantile regression methods perform similarly. Results are statistically similar across all thresholds.

3 Investigation

The path we choose to investigate the accuracy of our refined models is to use simulated data. A data series for which we know true volatility, and thus VaR is first generated. Both models are then fitted and compared across different performance indicators on the different models.

3.1 Simulated Data Analysis

The goal of this investigation is to simulate a return series that is complex enough so that our conventional models and boosted models cannot fit perfectly the simulated data. Thus, the DGP we use for this analysis is a Markov-Switching GARCH model, implemented using

the MSGARCH package in R by Ardia et al. (2019). This method by and large follow's Boucher et al. (2014)'s method, used for the same purpose.

Without diving too deep into the details of the model, the Markov-Switching model, or MS(2)-GARCH(1,1) is a simple GARCH(1,1) model where the parameters change from one state to another with probabilities following a first order markov-chain with a 2x2 transition probability like equation (4). Equations (5) and (6) represent the standard GARCH model.

$$P = \begin{bmatrix} p_{1,1} & p_{1,2} \\ p_{2,1} & p_{2,2} \end{bmatrix} \tag{4}$$

$$r_t = \mu_{s,t} + \sigma_{s,t} \cdot z_t \tag{5}$$

$$\sigma^2_{s,t} = \omega_{s,t} + \alpha_{s,t}\varepsilon^2_{t-1} + \beta_{s,t}\sigma^2_{s,t-1} \tag{6}$$

$p_{i,j}$ is the probability of moving to state "j" given current state "i", z_t is i.i.d normaly distributed and the set of possible states is of length two. Furthermore, we assume μ_{s_t} to be zero and we thus demean all the relevant time-series.

Using S&P500 adjusted returns between 2002 and 2023, we fit the time series to the MS(2)-GARCH(1,1) model and then simulate 5000 trading days, or around 40 years of data. The reasoning behind the choice of DGP is the following. Since we believe that the boosted method should overcome the shortcomings of the standard GARCH during high periods of volatility, we choose a DGP where one of the concatenated GARCH models exhibits a much higher dependence on recent events than the other. We hope that the boosted GARCH is flexible enough to estimate properly those two paradigms. This dependence is translated mathematically through a higher value for $\alpha_{s,t}$. The following table depicts the non-zero parameters of the MS(2)-GARCH(1,1) model, equation (7) represents the model probabilities while the graph depicts the simulated returns.

	α_1	α_2	β_1	β_2
Parameter Value	0.0858	0.216	0.884	0.778
t-Value	4.358	1.090	116.39	122.30

Table 1: MS(2)-GARCH(1,1) Parameters

$$P = \begin{bmatrix} 0.790 & 0.210 \\ 0.803 & 0.197 \end{bmatrix} \tag{7}$$

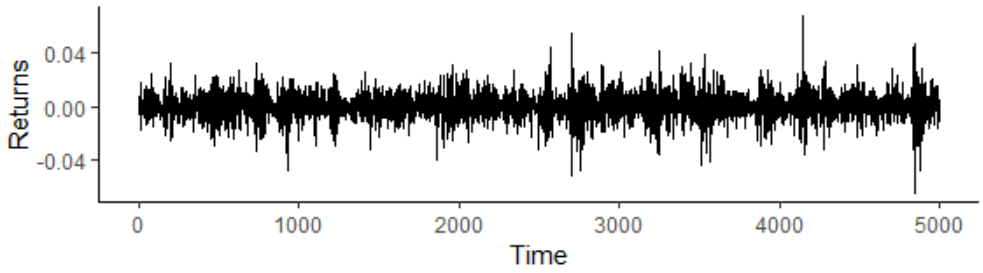


Figure 1: MS(2)-GARCH(1,1) Simulated Returns

The parameters of the simulated model indicate that the second GARCH model puts more emphasis on the recent news than the first model. As previously mention, this is indicated by the value of α_2 being more than twice that of α_1.

3.2 Models

The GARCH model follows the one presented in the MS(2)-GARCH(1,1) simulation model, without the model subscript, thus we refrain from presenting it mathematically. For the boosted model, we follow by and large the methodology of Matías et al. (2010). Keeping the same notation as in section 3.1:

Equation 1: Model

$$\hat{\sigma}_{t,j} = \sum_{k=1}^{j} \alpha_k f_k(\hat{\sigma}_{t-1,j}, \hat{\epsilon}^2_{t-1,j}), f_k \in F \tag{8}$$

Equation 2: Loss Function

$$L(\theta) = \sum_i (y_i - \hat{y}_i)^2 \tag{9}$$

The loss function we use in the boosted GARCH model is the same as the default function of the rugarch package, which is the mean squared error. Moreover, the decision is made to not include a regularization function as no substantial improvement to forecasting were made when including it, leading to the conclusion that the model does not show signs of over-fitting. Moreover, other over-fitting countermeasures were adopted, as seen later.

3.3 Methodology

As mentioned earlier, the model is tested in two ways, first by modelling volatility throughout the full sample, then, by estimating Value-at-Risk using a rolling estimation window. The methodology for the boosted volatility estimation component is the following. First, we split our sample in two equal sub-parts of 2500 data points, one for training and one for testing. We then fit the boosted model to the training set and verify its prediction power on the testing set, out-of-sample. The maximum depth of the model is set at 4. This parameter controls over fitting by setting the maximum layers any tree can have. The example shown in Section 2 has a depth of 2. In addition, since the computation of VaR is computationally expensive, we restrict the number of iterations to 50. In any case, the model showed quick convergence, already after 50 iterations.

For the standard normal GARCH(1,1), we fit the the model to the full 5000 time periods simulated sample and enforce stationary while using a hybrid solver. Without it, the rugarch package cannot fit the desired model to our data.

To compare the volatility estimates of our two models, we conduct a Welch's t-test with respect to the difference between the estimated volatility and the actual volatility. The

Welch's t-test is used to compare the means of two different populations with different variances. Therefore, we wish to test whether the mean difference in estimation between the GARCH and boosted GARCH are the same. Mathematically, we take the average over $\{t\}_{t=1}^{N}$ of equation (10) presented further below.

We then move to Value-at-Risk estimation and analysis on a unit one portfolio of our simulated series. First, we backtest our GARCH model using the *ugarchroll* function from the *rugarch* package on the full simulated sample, with an estimation window of 1000 time-periods, making sure we refit the model at each iteration. We perform the same estimation with the XgBoost model, using the same estimation window and refit frequency. The only VaR we extract from those estimations is the 5% VaR threshold as it is the most widely used. Moreover, as seen in the literature review, the results did not vary significantly with the VaR threshold. In other words, if model 1 outperforms model 2 at the VaR 5%, it won't under perform at the VaR 1% with statistical significance, Blom et al. (2023).

4 Results and Analysis

4.1 Volatility Estimation

Table 2 presentsthe fitted GARCH parameters.

Param.	Estimate	p-value
ω	0.000	0.468
α	0.116	0.000
β	0.847	0.000

Table 2: Fitted GARCH(1,1) Parameters

Comparing those parameters to the actual MS(2)-GARCH(1,1) parameters, we conclude

that the model correctly estimates the value of ω, while fitting a value of α and β between those of the MS(2)-GARCH(1,1) model. Thus, the standard GARCH model should both over-estimate and under-estimate volatility at different periods of time. Figure 2 below depicts the actual vs estimated volatility in the top panel. The bottom panel depicts the difference between GARCH and actual.

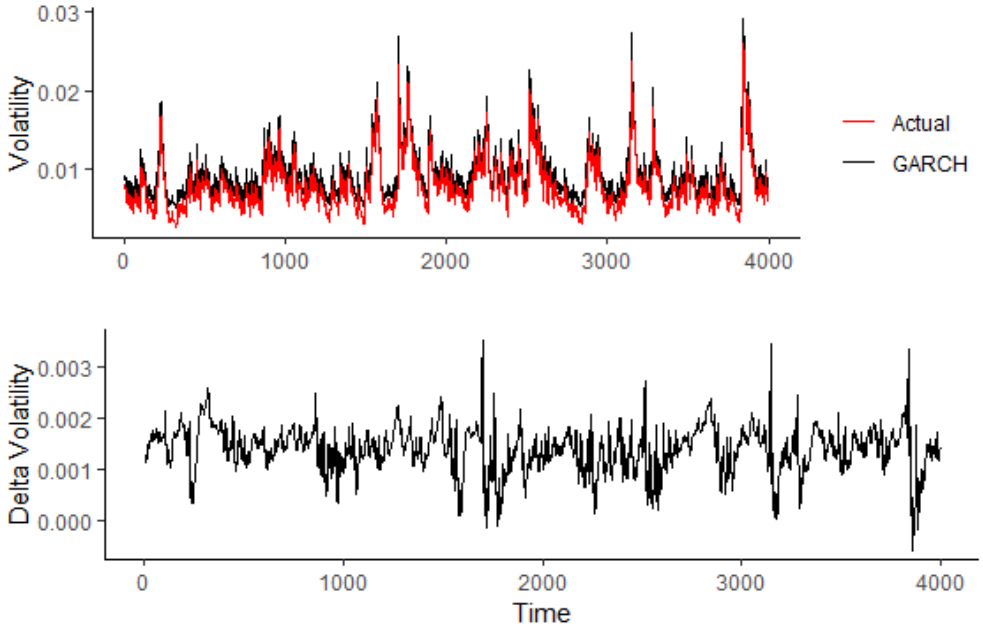


Figure 2: GARCH(1,1) Volatility Estimates

Three lessons can be learned from the table above. First, the fitted GARCH model consistently overestimates actual volatility. This bias is a sign of model miss-specification, in effect, the model is not complex enough to capture all the important relationships within the data. Secondly, the highest differences between estimated volatility and actual volatility happen, as predicted, around the peaks. Finally, the lowest differences coincide with drops in volatility from high peaks. Why is that? Because the half-life of our fitted model, represented by the sum of α and β, is lower than that of the true model. In other words, the fitted model is less persistent than the true model. This means that the volatility estimate will drop more sharply than the true value.

As for the Boosted GARCH model, one can retrieve the importance of each input to the model. The lagged residual squared, represented by α in the standard model, accounts for 22.6% of the volatility estimate while the lagged sigma squared, β, accounts for 77.4% of the volatility. Figure 3 depicts the same features as Figure 2, but for the boosted model.

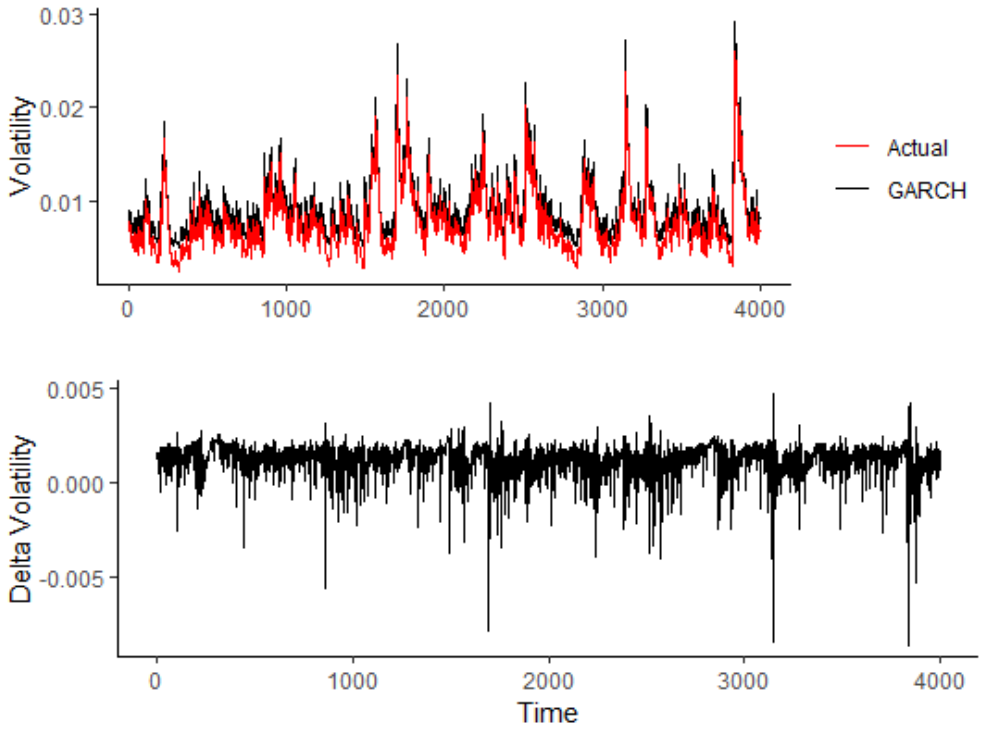


Figure 3: Boosted GARCH(1,1) Volatility Estimates

Comparing the bottom panel from the two models, one difference strikes out: none of the peaks in Delta Volatility were lower than zero in the standard GARCH model, while most of them are for the boosted GARCH. It appears that the boosted model underestimates volatility at the peaks, while displaying positive bias throughout the training sample. Figure 4 displays throughout time the ratio of delta volatility (10).

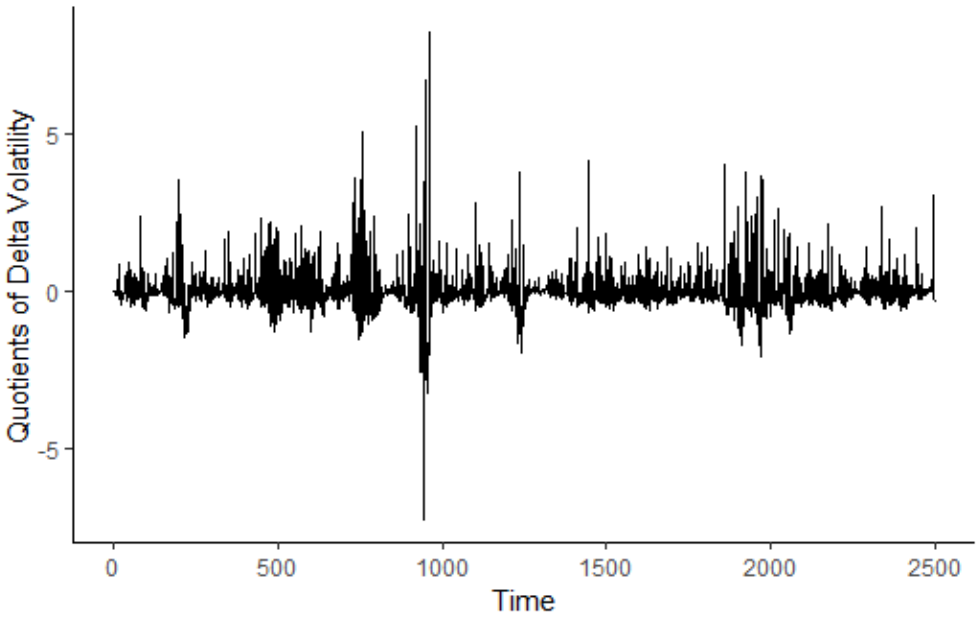


Figure 4: Ratio of Delta Volatility

$$RV_t = \frac{\hat{\sigma}_{t,B} - \sigma}{\hat{\sigma}_{t,G} - \sigma} - 1 \tag{10}$$

Equation (10) represents the ratio of delta volatility, where $\hat{\sigma}_{t,B}$ and $\hat{\sigma}_{t,G}$ are the respective estimated components of the boosted and standard models. Values above 0 are values where the bias, conditional on t, is higher for the boosted model and vice-versa. The mean value of the quotient is -0.0224. Interpretation is easier when adding one, leading to a mean of 0.9775, meaning that the boosted model displays on average a 2.5% reduction in bias. However, we test whether this difference is statistically significant. The formula for Welch's t-test is given by:

$$t = \frac{\bar{x}_1 - \bar{x}_2}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}}$$

where:

$\bar{x}_1$ and $\bar{x}_2$ are the sample means of the two groups,

s_1^2 and s_2^2 are the sample variances of the two groups,

n_1 and n_2 are the sample sizes of the two groups.

The results of the test are not conclusive, we cannot reject the base hypothesis that the means are equal. Although the boosted GARCH reduces the bias in calmer periods, its inability to correctly forecast volatility peaks is too much of a drawback.

4.2 Value-at-Risk estimation

Since the the Value-at-Risk forecast is computed directly using the volatility estimate, we do not expect meaningful differences between the VaR forecasts. However, in the previous section, the GARCH model was fitted using the full sample size while the boosted GARCH used only half of it for fitting, and the other half for testing. Since we calculate VaR using a rolling estimation window for both, results might differ. Figure 5 confirms this. It depicts on the top panel the returns as well as the GARCH Value-at-Risk estimation while the bottom panel depicts instances where one model saw a violation but not the other, with 27 in total.

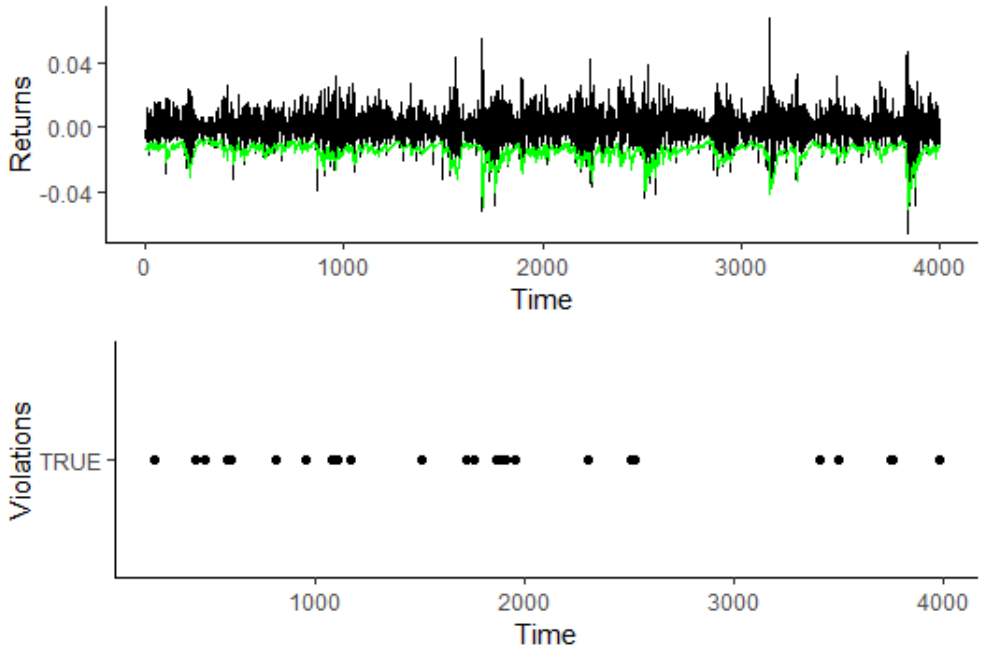


Figure 5: VaR Violations

We do not depict the VaR estimates of the boosted GARCH method as they are visually too similar to those of the standard GARCH method.

4.3 Coverage Test and Independence Test

The first test we use is the Bernoulli coverage test. Assuming independence, at each time-period, the probability of exceeding the VaR follows a Bernoulli distribution. Let the sequence $\{v\}_{t=1}^{N}$ be our violation sequence. The hypotheses are:

H_0 : The number of VaR exceedances follows a Bernoulli distribution

H_1 : The number of VaR exceedances does not follow a Bernoulli distribution

Given $\hat{\rho}$, the number of violations divided by the sample size, and v_1 the number of

violations, the likelihood function for the empirical distribution is the following:

$$L_U(\hat{\rho}) = (1-\hat{\rho})^{v_0}(\hat{\rho})^{v_1} \tag{11}$$

While the true likelihood function, called $L_R(\rho)$ is equal to the one above, with $\hat{p} = 0.05$, we test the following Likelihood Ratio which follows asymptotically a chi-square distribution:

$$LR = 2log\frac{L_U(\hat{\rho})}{L_R(\rho)} \tag{12}$$

If equation (12) is close to 0, then the empirical and theoretical likelihood functions are close to each other, if not, we reject the Null hypothesis given a threshold. The results are displayed in Table 3.

The second test we perform only tests for independence between two immediate violations. We perform Christoffersen (1998) conditional test. This test does not reject all types of independence, only the specified one. Why? Say we have a series where no violations follow each other within a 1 period range. For example: $1, 0, 1, 0, 1, 0, ...$ Christoffersen's test would reject independence since no two violations directly follow each other. However, there is clear dependence using a lag of 2 time periods.

The results can be found in the following table. Both the boosted GARCH and standard GARCH, with 185 and 181 respective violations, pass the Coverage Test. In effect, although the number of violations are not perfect, we cannot reject that they are different from 200 at the 5% confidence level. As for the Independence Test, surprisingly, the Boosted GARCH model does not pass the Independence Test while the standard GARCH does.

	Coverage Test	Independence Test
XgBoost	1.21	7.29
GARCH(1,1)	1.95	1.72
Threshold (5%)	3.84	5.99

Table 3: Violation Tests Results

5 Conclusion

When discussing the results of the comparison between the boosted GARCH and the standard GARCH, we must keep in mind that the boosting methodology is an optimizer. It's forecasting power is still bounded by the model parameters of the GARCH model. Moreover, no hyperparameter optimization was discussed since the goal of the presented work was to implement a simple solution to a well known risk management problem. Unlike Matías et al. (2010) and Blom et al. (2023) who both optimized their models with the plethora of parameters at the disposition of boosting package users, we have not done so. Such a follow-up could enhance the performance of our boosted model.

To summarize, given our simulated data, the boosted GARCH model does not deliver any benefits over the standard GARCH model. The expected benefits in terms of overall bias reduction are not statistically significant. With respect to forecasting in periods of higher stress, our upgraded model underestimates volatility. In terms of Value-at-Risk optimization, the boosted model performed poorly compared to its simpler cousin, as it was rejected on grounds of dependence of violations. Thus, we cannot validate our thesis.

Changes to the research methodology could provide better results. First, ensemble methods' benefits are seen when a lot of data is used. The boosted model would certainly benefit from this, although, real world applications are usually constrained in terms of data. Thus, even if the results were positive, they might not be applicable to real price series.

References

Ardia, D., Bluteau, K., Boudt, K., Catania, L., and Trottier, D.-A. (2019). Markov-switching GARCH models in R: The MSGARCH package. *Journal of Statistical Software*, 91(4):1–38.

Blom, H. M., de Lange, P. E., and Risstad, M. (2023). Estimating value-at-risk in the eurusd currency cross from implied volatilities using machine learning methods and quantile regression. *Journal of Risk and Financial Management*, 16(7):312.

Boucher, C. M., Daníelsson, J., Kouontchou, P. S., and Maillet, B. B. (2014). Risk models-at-risk. *Journal of Banking & Finance*, 44:72–92.

Chen, T. and Guestrin, C. (2016). XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 785–794, New York, NY, USA. ACM.

Christoffersen, P. F. (1998). Evaluating interval forecasts. *International economic review*, pages 841–862.

Matías, J. M., Febrero-Bande, M., González-Manteiga, W., and Reboredo, J. C. (2010). Boosting garch and neural networks for the prediction of heteroskedastic time series. *Mathematical and Computer Modelling*, 51(3-4):256–271.

Meir, R. and Rätsch, G. (2003). An introduction to boosting and leveraging. In *Advanced Lectures on Machine Learning: Machine Learning Summer School 2002 Canberra, Australia, February 11–22, 2002 Revised Lectures*, pages 118–183. Springer.

Schapire, R. E. (1990). The strength of weak learnability. *Machine learning*, 5:197–227.