Machine Learning for Multi-step Ahead Forecasting of Volatility Proxies

Jacopo De Stefani, Olivier Caelen[1], Dalila Hattab[2], and Gianluca Bontempi

Machine Learning Group, Departement d'Informatique, Université Libre de Bruxelles, Boulevard du Triomphe CP212, 1050 Brussels, Belgium
Worldline SA/NV R&D, Bruxelles, Belgium[1]
Equens Worldline R&D, Lille (Seclin), France[2]
{jacopo.de.stefani,gianluca.bontempi}@ulb.ac.be
olivier.caelen@worldline.com
dalila.hattab@equensworldline.com

Abstract. In finance, volatility is defined as a measure of variation of a trading price series over time. As volatility is a latent variable, several measures, named proxies, have been proposed in the literature to represent such quantity. The purpose of our work is twofold. On one hand, we aim to perform a statistical assessment of the relationships among the most used proxies in the volatility literature. On the other hand, while the majority of the reviewed studies in the literature focuses on a univariate time series model (NAR), using a single proxy, we propose here a NARX model, combining two proxies to predict one of them, showing that it is possible to improve the prediction of the future value of some proxies by using the information provided by the others. Our results, employing artificial neural networks (ANN), k-Nearest Neighbours (kNN) and support vector regression (SVR), show that the supplementary information carried by the additional proxy could be used to reduce the forecasting error of the aforementioned methods. We conclude by explaining how we wish to further investigate such relationship.

Keywords: financial time series, volatility forecasting, multi-step ahead forecast, machine learning

1 Introduction and problem statement

In time series forecasting, the largest body of research focuses on the prediction of the future values of a time series, with either a single or a multiple steps ahead forecasting horizon, given historical knowledge about the series itself. In statistical terms, such problem is equivalent to the forecast of the expected value of the time series in the future, conditioned on the past available information. In the context of stock market, the solution to the aforementioned problem could allow to determine the future valuation of a company, thus giving an information to the traders about how to act upon such change in the valuation. However, from the traders' standpoint, price is not the only variable of interest.The knowledge of the intensity of the fluctuations affecting this price (i.e. the stock volatility)

allow them to assess the risk associated to their investment. Since volatility is not directly observable given the time series, according to the granularity and the type of the available data, one could compute different measures, named volatility proxies [21]. Although volatility proxies based on intraday trading data exist [17], due to the restrictions on the access to such fine grained data, the rest of our analysis will be focused on proxies for daily data. A standard approach to volatility forecasting, once a given proxy has been selected, is to apply either a statistical Generalized AutoRegressive Conditional Heteroskedasticity (GARCH)-like model [2], or to apply a machine learning model. In addition, several hybrid approaches are emerging [16, 10, 19], including a non-linear computational component into the standard GARCH equations. In all the aforementioned cases, we deal with a univariate problem, where a single time series is used to predict the future values of the series itself. An exception is represented by the work of [30] where a volatility proxy is combined with external information (namely the volume of the queries to a web search engine for a given keyword). This paper proposes a method for multiple step ahead forecast of a volatility proxy incorporating the information from a second proxy in order to improve the prediction quality. The purpose of our work is twofold. First, we aim to perform a statistical assessment of the relationships among the most used proxies in the volatility literature. Second, we explore a NARX (Nonlinear Autoregressive with eXogenous input) approach to estimate multiple steps of the output, where the output and the input are two different proxies. In particular, our preliminary results show that the statistical dependencies between proxies can be used to improve the forecasting accuracy. The rest of the paper will be structured as follows: Section 2 will introduce the notation and provide a unified view on the different volatility proxies. Section 3 will introduce the formulation of the volatility forecasting problem as a machine learning task and will described the different tested models. Section 4 concludes the paper with a discussion of the results and the future research directions.

2 Volatility proxies: definition and notation

In this paper we consider univariate time series whose value at time t is denoted by the scalar value y_t. Let us consider the following quantities of interest, each of them on a daily time scale: $P_t^{(o)}, P_t^{(c)}, P_t^{(h)}, P_t^{(l)}$, respectively the stock prices at the *opening*, *closing* of the trading day and the *maximum* and *minimum* value for each trading day; v_t being the volume [1]. We will assume the availability of a training set of T past observations of each univariate series.

In the absence of detailed information concerning the price movements within a given trading day, stock volatility becomes directly unobservable [27]. To cope with such problem, several different measures (also called *proxies*) have been proposed in the econometrics literature [21, 12, 20, 13] to capture this information. However, there is no consensus in the scientific literature upon which volatility

[1] Number of traded stocks in a given day.

proxy should be employed for a given purpose. We will proceed by reviewing the different types of proxies available in the literature: $\sigma^{SD,n}$,σ^{i} and σ^{G}.

Volatility as variance The first proxy corresponds to the natural definition of volatility [21], that is a rolling standard deviation of a given stock's continuously compounded returns over a past time window of size n:

$$\sigma_t^{SD,n} = \sqrt{\frac{1}{n-1}\sum_{i=0}^{n-1}(r_{t-i} - \bar{r}_n)^2} \tag{1}$$

where

$$r_t = \ln\left(\frac{P_t^{(c)}}{P_{t-1}^{(c)}}\right) \tag{2}$$

represents the daily continuously compounded return for day t computed from the closing prices $P_t^{(c)}$ and $\bar{r}_n$ represents the returns' average over the period $\{t, \cdots, t-n\}$. In this formulation, n represents the degree of smoothing that is applied to the original time series.

Volatility as a proxy of the coarse grained intraday information The σ_t^i family of proxies is analytically derived in [12] by incorporating supplementary information (i.e. opening, maximum and minimum price for a given trading day) and trying to optimize the quality of the estimation.

The first estimator σ_t^0, which the authors propose as benchmark value, simply consists of the squared value of the returns (i.e. the ratio of the logarithms of the closing price time series):

$$\sigma_t^0 = \left[\ln\left(\frac{P_{t+1}^{(c)}}{P_t^{(c)}}\right)\right]^2 = r_t^2 \tag{3}$$

The second proposition σ_t^1 is able to reduce the variance of the estimator, by including the opening price, and computing a weighted average between two components, representing respectively the nightly and daily volatility:

$$\sigma_t^1 = \underbrace{\frac{1}{2f}\cdot\left[\ln\left(\frac{P_{t+1}^{(o)}}{P_t^{(c)}}\right)\right]^2}_{\text{Nightly volatility}} + \underbrace{\frac{1}{2(1-f)}\cdot\left[\ln\left(\frac{P_t^{(c)}}{P_t^{(o)}}\right)\right]^2}_{\text{Intraday volatility}} \tag{4}$$

The value of $f \in [0,1]$ represents the fraction of the trading day in which the market is closed. In the case of CAC40, we have that $f > 1-f$, since trading is only performed during roughly one third of the day. In this case, the weighting scheme proposed in (4) will give higher weight to the intraday volatility, with respect to the nightly one.

The third estimator, derived in [20] through the modeling of the price evolution as a stochastic diffusion process with unknown variance, is a function of the variation range (i.e. the difference between maximum and minimum value for the current trading day):

$$\sigma_t^2 = \frac{1}{2\ln 4} \cdot \left[\ln\left(\frac{P_t^{(h)}}{P_t^{(l)}} \right) \right]^2 \tag{5}$$

where the value $\frac{1}{2\ln 4}$ corresponds to the variance of the distribution of the high-low displacement difference, under the assumption of a stochastic Wiener process (i.e. with normally distributed increments).

Garman et al. [12] further improves the efficiency of the estimator in Equation (5) by including the information concerning nightly volatility.

$$\sigma_t^3 = \underbrace{\frac{a}{f} \cdot \left[\ln\left(\frac{P_{t+1}^{(o)}}{P_t^{(c)}} \right) \right]^2}_{\text{Nightly volatility}} + \underbrace{\frac{1-a}{1-f} \cdot \hat{\sigma}_2(t)}_{\text{Intraday volatility}} \tag{6}$$

Here, a is a weighting parameter, whose optimal value, according to the authors is shown to be 0.17, regardless of the value of f.

Furthermore, the same study introduces a family of estimators based on the normalization of the maximum, minimum and closing values by the opening price of the considered day. We can then define:

$$u = \ln\left(\frac{P_t^{(h)}}{P_t^{(o)}} \right) \qquad d = \ln\left(\frac{P_t^{(l)}}{P_t^{(o)}} \right) \qquad c = \ln\left(\frac{P_t^{(c)}}{P_t^{(o)}} \right) \tag{7}$$

where u is the normalized high price, d is the normalized low price and c is the normalized closing price.

We can derive Equation (8), by starting from a general, analytic form for the estimator, and then deriving the optimal values of the coefficient by minimizing the estimation variance.

$$\sigma_t^4 = 0.511(u-d)^2 - 0.019[c(u+d) - 2ud] - 0.383c^2 \tag{8}$$

The values of the coefficients are set assuming that the price dynamics follows a Brownian motion and enforcing scale invariance properties and price and time symmetry conditions. For all the details concerning the proof, we refer the interested reader to [12].

Equation (9) is derived from Equation (8) by eliminating the cross product terms.

$$\sigma_t^5 = 0.511(u-d)^2 - (2\ln 2 - 1)c^2 \tag{9}$$

Last but not least, the best estimator in terms of estimation variance efficiency is obtained by combining the overnight volatility measure with the optimal estimator described in Equation (8).

$$\sigma_t^6 = \underbrace{\frac{a}{f} \cdot \log\left(\frac{P_{t+1}^{(o)}}{P_t^{(c)}}\right)^2}_{\text{Nightly volatility}} + \underbrace{\frac{1-a}{1-f} \cdot \hat{\sigma}_4(t)}_{\text{Intraday volatility}} \tag{10}$$

GARCH-based volatility Even though the GARCH(p,q) [14] (Generalized AutoRegressive Conditional Heteroskedasticity) family of models is generally employed for volatility forecasting, we decided to consider it here as a filter, that, given the original time series, returns its estimation of the series volatility. All GARCH models assume that the return time series can be expressed as the sum of two components: a deterministic trend μ and a stochastic time-varying component ε_t. The stochastic component can be further decomposed and expressed as the product between a sequence of independent and identically distributed random variables Z_t with null mean and unit variance and a time varying scaling factor σ_t^G.

$$r_t = \mu + \varepsilon_t \tag{11}$$

$$\varepsilon_t = \sigma_t^G z_t z_t \sim \mathbb{N}(0,1) \tag{12}$$

The core of the model is the variance equation, describing how the residuals ε_t and the σ_t^G past volatility affects the future volatility.

$$\sigma_t^G = \sqrt{\omega + \sum_{j=1}^{p} \beta_j (\sigma_{t-j}^G)^2 + \sum_{i=1}^{q} \alpha_i \varepsilon_{t-i}^2} \tag{13}$$

The coefficients $\omega, \alpha_i, \beta_j$ are fitted according to the maximum likelihood estimated procedure proposed in [4]. In the case of our proxies, we consider the estimation of the volatility made by a GARCH ($p = 1$,$q = 1$) model as suggested in [13].

3 Multiple step ahead volatility forecasting

The Nonlinear Auto Regressive (NAR) formulation of a univariate time series as an input-output mapping allows the use of supervised machine learning techniques for time series one-step-ahead forecasting [6].

$$\mathbf{y} = f(\mathbf{x}) + \omega \tag{14}$$

$$\mathbf{y} = [y_{t+1}] \tag{15}$$

$$\mathbf{x} = [y_{t-d}, \cdots, y_{t-d-m+1}] \tag{16}$$

To be more precise, this model assumes an autoregressive dependence of the future value of the time series on the past m (lag or embedding order) values, with a given delay[2] d and an additional null-mean noise term ω.

[2] In the following of the paper we will assume $d = 0$ for the sake of simplicity.

With this structure, the forecasting task can be reduced to a two-step process. First the mapping f between the inputs x and the outputs y is learned through a supervised learning task, and then such mapping is used to produce the one-step-ahead forecast of the future values.

Extensions of this technique allows to perform multiple-step ahead forecast (i.e. $\mathbf{y} = [y_{t+H}, \cdots, y_{t+1}]$). Such extensions can be summarised into two main classes: single output (*Direct* and *Recursive* strategies) and multiple output (*MIMO*) strategies. The former learns a multi-input single output dependency while the latter learns a multi-input multiple output dependency. We invite the interested reader to see [24], [6], [25] for more details.

In what follows, we focus on two multi-step ahead single output learning task, employing the *Direct* strategy [5], [23], [8]. In the first one (NAR), we will focus on the multiple step ahead forecast of a primary volatility proxy σ_t^P using only its past values as input information, while in the second one (NARX), also the past values of an additional volatility proxy σ_t^X will be incorporated in the model described in (14):

$$\mathbf{y}_{NAR} = \mathbf{y}_{NARX} = [\sigma_{t+H}^P, \cdots, \sigma_{t+1}^P] \tag{17}$$

$$\mathbf{x}_{NAR} = [\sigma_{t-d}^P, \cdots, \sigma_{t-d-m+1}^P] \tag{18}$$

$$\mathbf{x}_{NARX} = [\sigma_{t-d}^P, \cdots, \sigma_{t-d-m+1}^P, \sigma_{t-d}^X, \cdots, \sigma_{t-d-m+1}^X] \tag{19}$$

We compare the two approaches for embedding orders $m \in \{2, 5\}$, several forecasting horizons $h \in \{2, 5, 8, 10, 12\}$ and for different estimators of the dependency f. More precisely, as estimators of the dependency, we employ a naive model, a GARCH(1,1), and three machine learning approaches: a feedforward Artificial Neural Networks, a k-Nearest Neighbors approach and Support Vector Machine based regression.

3.1 Naive

The *Naive* method is employed mainly as a benchmark for comparison for the other models, simply consisting in taking the last available historical value:

$$\hat{\sigma}_{t+h}^P = \sigma_{t-1}^P \tag{20}$$

3.2 GARCH(1,1)

The GARCH model corresponds to the one described in subsection 2, in equations (13) and (11), with $p = 1$ and $q = 1$.

3.3 Artificial Neural Networks

In machine learning and cognitive science, an artificial neural network (ANN) is a network of interconnected processing elements, called neurons, which are used to estimate or approximate functions that can depend on a large number of inputs

that are generally unknown. For our task, we will focus on a specific family of artificial neural networks, the multi-layer perceptron (MLP), with a single hidden layer. Equations (21) and (22) describe the structure of the model for a single forecasting horizon $t+h$ in the context of the *Direct* strategy, respectively for a NAR and a NARX model.

It should be noted that, as shown in Equation (21), the model can be decomposed into a linear autoregressive component of order m and a nonlinear component whose structure depends on the number of hidden nodes H (selected through k-fold cross-validation). When an external regressor is added, its influence will affect both the linear and nonlinear component, as shown in (22). In both cases the activity functions $f_o^h(\cdot)$ and $f_h(\cdot)$ are both logistic functions. Finally, our implementation of the MLP models is based on the `nnet` package for R [28].

$$\hat{\sigma}^P_{t+h} = f_o^h\left(\underbrace{b_o + \sum_{i=1}^{m} w_{io}\sigma^P_{t-i}}_{\textbf{Linear AR(m)}} + \underbrace{\sum_{j=1}^{H} w_{jo}\cdot f_h\left(\sum_{i=1}^{m} w_{ij}\sigma^P_{t-i} + b_j\right)}_{\textbf{Non-linear component}}\right) \tag{21}$$

$$\hat{\sigma}^P_{t+h} = f_o^h\left(b_o \underbrace{+ \sum_{i=1}^{m} w_{io}\sigma^P_{t-i} + w_{(i+m)o}\sigma^X_{t-i}}_{\textbf{Linear ARX(m)}} + \underbrace{\sum_{j=1}^{H} w_{jo}\cdot f_h\left(\sum_{i=1}^{m} w_{ij}\sigma^P_{t-i} + w_{(i+m)j}\sigma^X_{t-i} + b_j\right)}_{\textbf{Non-linear component}}\right) \tag{22}$$

3.4 K Nearest Neighbors

The k-Nearest neighbors (kNN) model is a local nonlinear model used for classification and regression. In the case of regression, the prediction for a given input vector $\mathbf{x}^*$ is obtained through local learning [3], a method that produces predictions by fitting a simple local model in the neighborhood of the point to be predicted. The neighborhood of a point is defined by taking the the k values having the minimal values for a chosen distance metric defined on the space of the input vector [1].

In this case, every data point is represented in the form $(\mathbf{x}, \mathbf{y})$ where $\mathbf{x}$ represents the vector of input values and $\mathbf{y}$ the corresponding output vector, as described in Figure 1. Then the prediction for an unknown input vector $\mathbf{x}^*$ is computed as follows:

$$\hat{\mathbf{y}}(\mathbf{x}^*) = \frac{1}{k}\sum_{i \in k\text{NN}} \mathbf{y}(\mathbf{x_i}) \tag{23}$$

where $\mathbf{y}(\mathbf{x}_i)$ is the output vector of the i^{th} nearest neighbor of the input vector $\mathbf{x}$ in the dataset. The choice of the optimal number of neighbors k will

be performed through automatic leave-one-out selection as described in [5]. Our implementation of the kNN models is based on the R package `gbcode` [7].

3.5 Support Vector Regression

Support Vector Regression is a regression methodology, based on the Support Vector Machine theoretical framework [9]. The key idea behind SVR is that the regression model can be expressed using a subset of the input training examples, called the support vectors. In more formal terms, the model (Equation (24)) is a linear combination over all the n support vector of a bivariate kernel function $k(\cdot,\cdot)$ taking as inputs the data point $\mathbf{x}$ whose forecast is required and the i^{th} support vector $\mathbf{x}_i$. The coefficients α_i, α_i^* are determined through the minimization of an empirical risk function (cf. [22]), solved as a continuous optimization problem.

$$\mathbf{y} = \sum_{i=1}^{n} (\alpha_i - \alpha_i^*) k(\mathbf{x}, \mathbf{x}_i) \tag{24}$$

$$k(\mathbf{x}, \mathbf{x}_i) = e^{\frac{\|\mathbf{x}-\mathbf{x}_i\|^2}{2\gamma^2}} \tag{25}$$

Among the different available kernel functions we employ the radial basis one (Equation(25)), for which the optimal value of the γ parameter is determined through grid search. Here, the SVM implementation of the R package `e1071` [18] is used for the experiments. As for ANN and kNN, we will be testing two different dataset structures (cf. Figure 1), representing respectively the exclusion (on the left) and the inclusion (on the right) of an external regressor.

Direct NAR

- A single model f^h for each horizon h.
- Forecast at h step is made using h^{th} model.

x			**y**
σ_3^P	σ_2^P	σ_1^P	σ_5^P
σ_4^P	σ_3^P	σ_2^P	σ_6^P
...	...	...	...
σ_{T-5}^P	σ_{T-6}^P	σ_{T-7}^P	σ_{T-2}^J

Direct NARX

- A single model f^h for each horizon h.
- Forecast at h step is made using h^{th} model.

x						**y**
σ_3^P	σ_2^P	σ_1^P	σ_3^X	σ_2^X	σ_1^X	σ_5^P
σ_4^P	σ_3^P	σ_2^P	σ_4^X	σ_3^X	σ_2^X	σ_6^P
...	...	...	...	...	...	...
σ_{T-5}^P	σ_{T-6}^P	σ_{T-7}^P	σ_{T-5}^X	σ_{T-6}^X	σ_{T-7}^X	σ_{T-2}^P

Fig. 1: Comparison of the dataset structure and model identification procedure for NAR and NARX forecasting strategies. The primary proxy is denoted with σ_t^P, while the secondary one is σ_t^X. The example datasets are shown for a model order $m = 3$ and a forecasting horizon $h = 3$.

4 Experimental Results

4.1 Dataset description

The proxies have been computed on the 40 time series of the french stock market index CAC40 from 05-01-2009 to 22-10-2014 (approximately 6 years) for a total 1489 OHLC (Opening, High, Low, Closing) samples for each time series. In addition to the proxies, we include also the continuously compounded return and the volume variable (representing the number of trades in given trading day).

4.2 Statistical analysis

Figure 2 shows the aggregated correlation (over all the 40 time series) between all the proxies under study, obtained by meta-analysis [11]. Black rectangles indicate the results of an hierarchical clustering using [29] with k=3. As expected, we can observe a correlation clustering phenomenon between proxies belonging to the same family, i.e. σ_t^i and $\sigma_t^{SD,n}$. The presence of σ_t^0 in the $\sigma_t^{SD,n}$ cluster can be explained by the fact that the former represents a degenerate case of the latter when $n = 1$. Moreover, we find a correlation between the volume and the σ_t^i family.

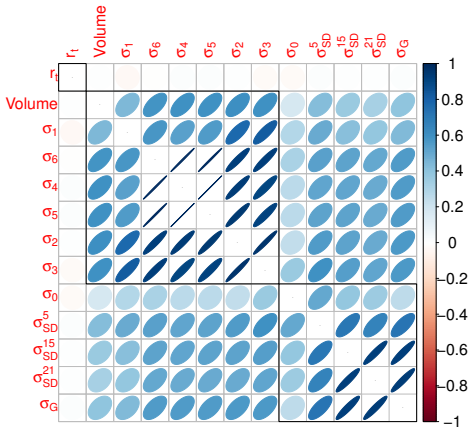

Fig. 2: Summary of the correlations among the different volatility proxies for the 40 time series composing the CAC40 indexes. All the correlations shown in the table (except those of r_t) are statistically significant (pv=0.05).

4.3 Forecasts

Table 1 shows the performances of the machine learning techniques under study both with (NARX) and without (NAR) external regressor on proxy σ^4, averaged across the 40 times series of CAC40. For each combination of time series, method,

				Horizon - H				
	σ^X	Family	m	2	5	8	10	12
Average	$\emptyset$	-	-	2.1409	1.5064	1.3456	1.2928	1.2716
GARCH(1,1)	$\emptyset$	-	-	4.7169	3.2481	2.8989	2.7878	2.7560
ANN-Dir	$\emptyset$	-	2	0.7914	0.8401	0.8488	0.8469	0.8527
	$\emptyset$	-	5	0.6140	0.7180	0.7707	0.7702	0.7767
	σ_6	=	2	0.5786	**0.5772**	0.5877^{+}	0.5924^{+}	0.6063^{+}
	$Volume$	=	2	0.5492	0.5738	0.5945^{+}	0.6022^{+}	0.6126^{+}
	$\sigma^{SD,5}$	$\neq$	2	0.5856	0.5835	0.5958^{+}	0.6016^{+}	0.6168^{+}
	$\sigma^{SD,15}$	$\neq$	2	0.5856	0.5835	0.5958^{+}	0.6016^{+}	0.6168^{+}
	$\sigma^{SD,21}$	$\neq$	2	0.5890	0.5715	0.5710	0.5649	0.5756
kNN-Dir	$\emptyset$	-	2	0.6375	0.7684	0.7924	0.7905	0.7933
	$\emptyset$	-	5	0.6147	0.7132	0.7425	0.7381	0.7436
	σ_6	=	2	0.5030	0.5665^{+}	0.5822^{+}	0.5825^{+}	0.5877^{+}
	$Volume$	=	2	0.5901	0.5618^{+}	0.5773^{+}	0.5794^{+}	0.5842^{+}
	$\sigma^{SD,5}$	$\neq$	2	0.5596	0.5753^{+}	0.5835^{+}	0.5849^{+}	0.5892^{+}
	$\sigma^{SD,15}$	$\neq$	2	0.6306	0.5760^{+}	0.5817^{+}	0.5770^{+}	0.5780^{+}
	$\sigma^{SD,21}$	$\neq$	2	0.5396	0.5790^{+}	0.5744^{+}	0.5670^{+}	0.5678^{+}
SVR-Dir	$\emptyset$	-	2	0.4261	0.5965	0.6442	0.6463	0.6503
	$\emptyset$	-	5	**0.4070**	0.5718	0.6158	0.6145	0.6208
	σ_6	=	2	0.7265	0.5577^{+}	$\mathbf{0.5442}^{+}$	0.5339^{+}	0.5329^{+}
	$Volume$	=	2	0.5901	0.5618^{+}	0.5773^{+}	0.5794^{+}	0.5842^{+}
	$\sigma^{SD,5}$	$\neq$	2	0.6605	0.5691^{+}	0.5489^{+}	0.5408^{+}	0.5407^{+}
	$\sigma^{SD,15}$	$\neq$	2	0.8313	0.5764^{+}	0.5489^{+}	$\mathbf{0.5336}^{+}$	$\mathbf{0.5307}^{+}$
	$\sigma^{SD,21}$	$\neq$	2	0.5890	0.5715^{+}	0.5710^{+}	0.5649^{+}	0.5756^{+}

Table 1: σ^4 - Naive Normalized MASE - 40 TS. All the ML methods perform significantly better (pv=0.05) than the GARCH(1,1) method across all the horizons. The superscript $^{+}$ denotes a significant improvement (pv=0.05) with respect to the corresponding method with no additional regressor. The bold notation is used to identify the best method for each horizon.

forecasting horizon and model order we performed a number of training and test tasks by following a rolling origin strategy [26]. The size of the training set is $\frac{2N}{3}$ and the procedure is repeated for 50 testing sets of length H. The regressor combinations have been selected in order to test whether the belonging (=) or not to the same proxy family ($\neq$) impacts the forecasting performance. The employed error measure is the Mean Absolute Scaled Error [15], normalized at each forecasting horizon by the the MASE of the *Naive* method.

$$\text{MASE} = \frac{\sum_{t=1}^{T} \left|\sigma_t^P - \hat{\sigma}_t^P\right|}{\frac{T}{T-1}\sum_{t=2}^{T} \left|\sigma_t^P - \sigma_{t-1}^P\right|} \tag{26}$$

We include in our analysis a GARCH(1,1) method [13] as a baseline reference method. While employing an additional regressor, model orders higher than 2 have not been tested due to the excessive computational time required by the

corresponding technique for the given task or due to numerical convergence problems. A first observation from the table is that all the ML methods, both in the single input and the multiple input configuration, are able to outperform the reference GARCH method. Moreover, both the increase of the model order m and the introduction of an additional regressor are able to improve the methods' performances. However, only the addition of an external regressor, for horizons greater than 8 steps ahead is shown to bring a statistically significant improvement (paired t-test, pv=0.05). Even though no model appear to clearly outperform all the others on every horizons, we can observe that the SVR model family is generally able to produce smaller forecast errors than those based on ANN and k-NN.

5 Conclusion and Future work

After having shown the benefits of including an additional proxies in our models, our main aim is to investigate how the forecasting quality of volatility could be improved, mainly by tuning three parameters in our methods: the choice of the additional proxy, the employed machine learning technique and the size of the training window. In order to further advance our research, we also plan to study how the current approach could be generalized, in order to include an arbitrary number of volatility proxies.

Acknowledgments. Jacopo De Stefani acknowledges the support of the ULB-WORLDLINE agreement. Gianluca Bontempi acknowledges the funding of the Brufence project (Scalable machine learning for automating defense system) supported by INNOVIRIS (Brussels Institute for the encouragement of scientific research and innovation).

References

1. Altman, N.S.: An introduction to kernel and nearest-neighbor nonparametric regression. The American Statistician 46(3), 175–185 (1992)
2. Andersen, T.G., Bollerslev, T.: Arch and garch models. Encyclopedia of Statistical Sciences (1998)
3. Atkeson, C.G., Moore, A.W., Schaal, S.: Locally weighted learning for control. In: Lazy learning, pp. 75–113. Springer (1997)
4. Bollerslev, T.: Generalized autoregressive conditional heteroskedasticity. Journal of econometrics 31(3), 307–327 (1986)
5. Bontempi, G., Taieb, S.B.: Conditionally dependent strategies for multiple-step-ahead prediction in local learning. International journal of forecasting 27(3), 689–699 (2011)
6. Bontempi, G., Taieb, S.B., Le Borgne, Y.A.: Machine learning strategies for time series forecasting. In: Business Intelligence, pp. 62–77. Springer (2013)
7. Bontempi, Gianluca: Code from the handbook "statistical foundations of machine learning", `https://github.com/gbonte/gbcode`

8. Cheng, H., Tan, P.N., Gao, J., Scripps, J.: Multistep-ahead time series prediction. In: Pacific-Asia Conference on Knowledge Discovery and Data Mining. pp. 765–774. Springer (2006)
9. Cortes, C., Vapnik, V.: Support vector machine. Machine learning 20(3), 273–297 (1995)
10. Dash, R., Dash, P.: An evolutionary hybrid fuzzy computationally efficient egarch model for volatility prediction. Applied Soft Computing 45, 40–60 (2016)
11. Field, A.P.: Meta-analysis of correlation coefficients: a monte carlo comparison of fixed-and random-effects methods. Psychological methods 6(2), 161 (2001)
12. Garman, M.B., Klass, M.J.: On the estimation of security price volatilities from historical data. Journal of business pp. 67–78 (1980)
13. Hansen, P.R., Lunde, A.: A forecast comparison of volatility models: does anything beat a garch (1, 1)? Journal of applied econometrics 20(7), 873–889 (2005)
14. Hentschel, L.: All in the family nesting symmetric and asymmetric garch models. Journal of Financial Economics 39(1), 71–104 (1995)
15. Hyndman, R.J., Koehler, A.B.: Another look at measures of forecast accuracy. International journal of forecasting 22(4), 679–688 (2006)
16. Kristjanpoller, W., Fadic, A., Minutolo, M.C.: Volatility forecast using hybrid neural network models. Expert Systems with Applications 41(5), 2437–2442 (2014)
17. Martens, M.: Measuring and forecasting s&p 500 index-futures volatility using high-frequency data. Journal of Futures Markets 22(6), 497–518 (2002)
18. Meyer, D., Dimitriadou, E., Hornik, K., Weingessel, A., Leisch, F., Chang, C.C., Lin, C.C.: e1071: Misc functions of the department of statistics, probability theory group (formerly: E1071), tu wien, `https://cran.r-project.org/web/packages/e1071/`
19. Monfared, S.A., Enke, D.: Volatility forecasting using a hybrid gjr-garch neural network model. Procedia Computer Science 36, 246–253 (2014)
20. Parkinson, M.: The extreme value method for estimating the variance of the rate of return. Journal of Business pp. 61–65 (1980)
21. Poon, S.H., Granger, C.W.: Forecasting volatility in financial markets: A review. Journal of economic literature 41(2), 478–539 (2003)
22. Sapankevych, N.I., Sankar, R.: Time series prediction using support vector machines: a survey. IEEE Computational Intelligence Magazine 4(2) (2009)
23. Sorjamaa, A., Hao, J., Reyhani, N., Ji, Y., Lendasse, A.: Methodology for long-term prediction of time series. Neurocomputing 70(16), 2861–2869 (2007)
24. Taieb, S.B., Bontempi, G., Atiya, A.F., Sorjamaa, A.: A review and comparison of strategies for multi-step ahead time series forecasting based on the nn5 forecasting competition. Expert systems with applications 39(8), 7067–7083 (2012)
25. Taieb, S.B., Sorjamaa, A., Bontempi, G.: Multiple-output modeling for multi-step-ahead time series forecasting. Neurocomputing 73(10), 1950–1957 (2010)
26. Tashman, L.J.: Out-of-sample tests of forecasting accuracy: an analysis and review. International journal of forecasting 16(4), 437–450 (2000)
27. Tsay, R.S.: Analysis of financial time series, vol. 543. John Wiley & Sons (2005)
28. Venables, W.N., Ripley, B.D.: Modern Applied Statistics with S. Springer, New York, fourth edn. (2002), `http://www.stats.ox.ac.uk/pub/MASS4`, iSBN 0-387-95457-0
29. Ward Jr, J.H.: Hierarchical grouping to optimize an objective function. Journal of the American statistical association 58(301), 236–244 (1963)
30. Xiong, R., Nichols, E.P., Shen, Y.: Deep learning stock volatility with google domestic trends. arXiv preprint arXiv:1512.04916 (2015)