A Multivariate and Multi-step Ahead Machine Learning Approach to Traditional and Cryptocurrencies Volatility Forecasting

Jacopo De Stefani[1(✉)], Olivier Caelen[2], Dalila Hattab[3], Yann-Aël Le Borgne[1], and Gianluca Bontempi[1]

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

Abstract. Multivariate time series forecasting involves the learning of historical multivariate information in order to predict the future values of several quantities of interests, accounting for interdependencies among them. In finance, several of this quantities of interests (stock valuations, return, volatility) have been shown to be mutually influencing each other, making the prediction of such quantities a difficult task, especially while dealing with an high number of variables and multiple horizons in the future. Here we propose a machine learning based framework, the DFML, based on the Dynamic Factor Model, to first perform a dimensionality reduction and then perform a multiple step ahead forecasting of a reduced number of components. Finally, the components are transformed again into an high dimensional space, providing the desired forecast. Our results, comparing the DFML with several state of the art techniques from different domanins (PLS, RNN, LSTM, DFM), on both traditional stock markets and cryptocurrencies market and for different families of volatility proxies show that the DFML outperforms the concurrent methods, especially for longer horizons. We conclude by explaining how we wish to further improve the performances of the framework, both in terms of accuracy and computational efficiency.

G. Bontempi acknowledges the support of the INNOVIRIS SecurIT project *BruFence: Scalable machine learning for automating defense system*. J. De Stefani acknowledges the support of the ULB-Worldline Collaboration Agreement. Computational resources have been provided by the Consortium des Équipements de Calcul Intensif (CÉCI), funded by the Fonds de la Recherche Scientifique de Belgique (F.R.S.-FNRS) under Grant No. 2.5020.11.

C. Alzate et al. (Eds.): MIDAS 2018/PAP 2018, LNAI 11054, pp. 7–22, 2019.
https://doi.org/10.1007/978-3-030-13463-1_1

Keywords: Multivariate time series forecasting · Volatility forecasting · Multi-step ahead forecast · Dynamic factor models

1 Introduction

The problem of time series forecasting, in its simplest form, deals with the prediction of a given quantity of interest in the future, given its historical values. Moreover, one could be interested in forecasting the immediate next value in the future (one-step-ahead forecasting) as well as being concerned with the estimation of a sequence of future values (multi-step-ahead forecasting). In a similar fashion, the problem might involve a single quantity (univariate forecasting), or several quantities at once (multivariate forecasting), in order to exploit potential interrelationships among them. In the context of finance, specific quantities of interest are: the stock price of a given company over time, its returns or the intensity of the fluctuations affecting the price (i.e. its volatility), among others. Specifically, in the case of stock markets, the underlying trend of the market influences all the stocks that are currently traded. As shown in [18], stock prices of firms acting on the same market often show similar patterns in the sequel of news that are important for the entire market. Moreover, analyzing global volatility transmission, Engle et al. [12] found evidence supporting volatility interdependence among the world's major trading areas. For these reasons, while modeling these time dependent quantities of interest, a multivariate model appears to be a natural choice to incorporate interdependencies into the forecasting process.

Among all the quantities of interest, in the following, we will focus on the problem of multivariate volatility forecasting. In this specific case, the quantity of interest is a latent variable, which cannot be directly observed given the time series, but only estimated, according to the granularity and the type of the available data, through different measures, named volatility proxies [27]. According to the choice of the proxy, several approaches have been proposed to tackle this multivariate problem. The largest body of the volatility forecasting literature focus on multivariate extensions of the well known GARCH [2] on traditional stock market data, for instance, citing some recently published work: [13] and [3]. For a thorough review of the different univariate and multivariate methods, we refer the interested reader to the latter. Due to the steadily growth of the cryptocurrencies market capitalization [11], coupled with the currencies' volatility, GARCH-like models [7], [32] have also been applied for non-traditional markets. The main problem of these approaches is that traditional multivariate models often suffers from the "curse of dimensionality": as the number of dimensions increase, the number of parameters grows superlinearly in the number of dimensions, making the model estimation heavily computationally intensive, especially in the case of multiple step ahead forecasts.

In order to profit from the richness of a multivariate model, while maintaining a reasonable computational complexity, we propose to employ the DFML [4], a multivariate, multistep-ahead machine learning forecasting framework involving a dimensionality compression process, based on the dynamic factor model

(DFM) principle [14]. The choice of this generic time series forecasting framework requires the usage of model-independent volatility proxies which will be discussed in Sect. 3, requiring us to dismiss GARCH as a proxy of volatility, due to his tight coupling between the proxy and the corresponding forecasting model to use, as discussed in [8].

At the time of writing, we had been able to find either multivariate techniques dealing with the forecasting of either cryptocurrencies prices [1,6] or univariate techniques dealing with the forecasting of volatility either with a one-step ahead [7,32] or multistep-ahead [10]. Nevertheless, we are not aware of any other work tackling both the problems of multivariate and multi-step ahead cryptocurrencies volatility forecasting, specifically in the case of large dimensionality and a reduced number of datapoints. Our technique will then be tested on two different benchmarks: one concerning cryptocurrencies and a second one, concerning a traditional regulated stock market (CAC40) being a de facto multivariate extension of [25].

The rest of the paper is structured as follows: Sect. 2 provides an oveview of the Dynamic Factor Machine Learner approach. Section 3 introduces the different tested multivariate models as well as the considered datasets and the formulation of the relevant forecast quantities. Section 4 concludes the paper with a discussion of the results and the future research directions.

2 Dynamic Factor Machine Learner

A Dynamic Factor Model (DFM) is a technique for multivariate forecasting originating in the economic forecasting community [14]. The basic idea of DFM is that a small number of unobserved series (or factors) can account for the temporal behavior of a much larger number of variables. If we are able to obtain accurate estimates of these factors, the forecasting endeavor can be made simpler by using the estimated dynamic factors for forecasting instead of using all series. In equations:

$$\mathbf{Y}_{t+1} = \mathbf{W}\mathbf{Z}_{t+1} + \epsilon_{t+1} \tag{1}$$

$$\mathbf{Z}_{t+1} = \mathbf{A}_t\mathbf{Z}_t + \cdots + \mathbf{A}_{t-m+1}\mathbf{Z}_{t-m+1} + \eta_{t+1} \tag{2}$$

where $\mathbf{Y}_t$ is a multivariate time series vector at time t, $\mathbf{Z}_t$ is the vector of unobserved factors of size q ($q \ll n$), $\mathbf{A}_i$ are $q \times q$ coefficient matrices, $\mathbf{W}$ is the matrix $(n \times q)$ of dynamic factor loadings and the vectors of disturbances terms η are assumed to be uncorrelated. As shown in Eq. 2, in the original DFM, the latent factors follow a VAR time series process. For a detailed review of DFM models, the interested reader could refer to [28].

Here, we propose to employ a machine learning extension of the DFM (called DFML - Dynamic Factor Machine Learner). The DFML, first proposed by Bontempi et al. [4] and further discussed in [9], relies on dimensionality reduction techniques to extract the factors. Then, the factors are forecast using a nonlinear model. Finally, the forecasts of the factors are transformed back to the original

values by inverting the dimensionality reduction process. The basic architecture of the DFML is depicted in Fig. 1, along with the description of the different variants. Concerning dimensionality reduction, both linear (PCA) and nonlinear (autoencoder) techniques are employed in the DFML. Linear dimensionality reduction by PCA transforms the n original time series $\mathbf{Y}[1], \ldots, \mathbf{Y}[n]$ into q new variables $\mathbf{Z}[1], \ldots, \mathbf{Z}[q]$ (called *principal components* or *factors*) such that the new variables are uncorrelated with each other and account for decreasing portions of the variance of the original variables. The q principal components are then expressed as weighted sums of the elements of $\mathbf{Y}$ with maximal variance, where the weights are normalized and constrained to ensure orthogonality:

$$\mathbf{Z}[p] = \sum_{j=1}^{n} w_{jp} \mathbf{Y}[j], \qquad p = 1, \ldots, q \tag{3}$$

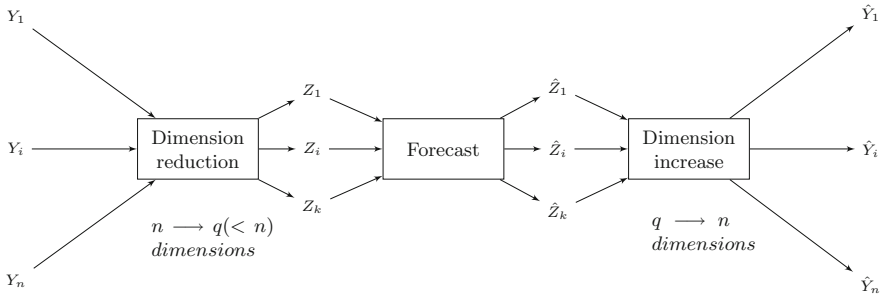
Given the multivariate time series matrix $\mathbf{Y}$, $\mathbf{Z} = \mathbf{YW}$ represents the projection of the series onto the pth principal components and $\hat{\mathbf{Y}} = \mathbf{ZW}^T$ represent the reconstruction $\hat{\mathbf{Y}}$ of the values of $\mathbf{Y}$, based on the factors $\mathbf{Z}$. On the other hand, nonlinear dimensionality reduction is performed through the use of autoencoders. Autoencoders are neural networks trained to learn identity mapping from inputs to outputs [31], through a constrained architecture to enforce dimensionality reduction. As such their input and output layer have the same number of neurons n as the number of input time series but their hidden layers contain a reduced number of neurons q. Autoencoders are composed of two stacked multi-layer networks: an *encoder*:

$$\mathbf{Z}_t = f_\theta(\mathbf{Y}_t) \tag{4}$$

that transforms inputs $\mathbf{Y}_t$ into some latent (encoded) representation $\mathbf{Z}_t$, and a *decoder*:

$$\hat{\mathbf{Y}}_t = g_{\theta'}(\mathbf{Z}_t) \tag{5}$$

that reconstructs an approximation $\hat{\mathbf{Y}}_t$ of the input $\mathbf{Y}_t$ on the basis of the latent feature $\mathbf{Z}_t$ and where the mappings f_θ and $g_{\theta'}$ are non-linear. The network is usually trained using gradient descent techniques such as backpropagation, with the objective of minimizing the mean-squared error between the input $\mathbf{Y}_t$ and the output (its reconstruction $\hat{\mathbf{Y}}_t$) [31]. Concerning the forecasting part, the original DFML paper [4] proposes to forecast each factor independently (given their orthogonality) using a nonlinear model (lazy learning [5]) and a univariate multi-step-ahead forecasting strategy. In addition to the basic forecaster, the paper also proposes an optimized version (DFML′), performing a joint selection of the hyperparameters (number of factors for the dimensionality reduction, predictor and multi-step-ahead strategy for the forecaster) using out-of-sample assessment. Although we consider lazy methods for the forecaster, the modular architecture of this framework easily allows the replacement of the aforementioned technique with alternative supervised machine learning approaches (e.g. SVM, RNN).



	Dimensionality reduction	**Forecast**
DFM	PCA	VAR
DFML$_{\mathbf{PCA}}$	PCA	Lazy-learning
DFML$_{\mathbf{A}}$	Autoencoder	Lazy-learning
DFML'$_{\mathbf{PCA}}$	Optimized PCA	Optimized Lazy-learning

Fig. 1. Schema of the DFML architecture with a summary of the different components as implemented in the different proposed methods.

3 Methodology

3.1 Multivariate Forecasting Methods

Multiple Univariate Techniques - {Naive, UNI}: In the case of a multivariate time series $\mathbf{Y}$, *univariate* approaches are still of interest since the multivariate forecasting task can be decomposed in a number of single-output multi-input tasks (or equivalently in a set of NARX tasks with exogenous variables)

$$\begin{cases} Y_{t+1}[1] & = f_1(Y_t[1], \ldots, Y_{t-m+1}[1], \ldots, \\ & \quad Y_t[n], \ldots, Y_{t-m+1}[n]) + w_t[1] \\ \vdots & \\ Y_{t+1}[n] & = f_n(Y_t[1], \ldots, Y_{t-m+1}[1], \ldots, \\ & \quad Y_t[n], \ldots, Y_{t-m+1}[n]) + w_t[n] \end{cases} \tag{6}$$

In this case the training set is used to learn the n mapping functions f_i, $i = 1, \ldots, n$, with $w_t[i]$ being uncorrelated disturbances. For large n the problem of large input dimensionality can be addressed by adopting a feature selection technique, selecting a reduced number q of most correlated features For these univariate techniques, we will also consider a *naive* method in which $\forall i \in \{1, \ldots, n\}, f_i(t) = Y_{t-1}[i]$, i.e. for every series, the forecast for the following H steps is given by the last available value. These are the baseline methods against which we will compare the performances of our forecaster.

Partial Least Squares - PLS: Partial Least Squares [15] allows the joint forecasting of the H steps ahead of the multivariate time series on the basis of the

lagged vectors $\mathbf{Y}_t, \ldots, \mathbf{Y}_{t-m}$. This is a multi-input multi-output regression task where the number of inputs amounts to nm and the number of outputs to Hn respectively, with n being the number of variables, m the embedding order of the model and H being the forecasting horizon. The benefit of PLS is that it allows at the same time a dimensionality reduction of the inputs and a joint prediction of the outputs, taking then into consideration the dependency between the future steps. An example of application of PLS in financial time series forecasting can be found in [22].

Recurrent Neural Networks - {RNN, LSTM}: Recurrent Neural networks (RNN) form a class of predictive models based on neural networks, in which recurrent connections to the network inputs allow to model dynamic temporal dependencies. In their simple form (also known as simple RNN) [17,23], the recurrent connections come from a *hidden state* H_t, which is also used for predicting future values Y_t:

$$\mathbf{H}_t = \sigma(\mathbf{W}_{HY}\mathbf{Y}_{t-1} + \mathbf{W}_{HH}\mathbf{H}_{t-1} + \mathbf{B}_H), \tag{7}$$

$$\mathbf{Y}_t = \mathbf{W}_{YH}\mathbf{H}_t + \mathbf{B}_Y \tag{8}$$

The matrices $\mathbf{W}_{HY}$, $\mathbf{W}_{HH}$, $\mathbf{W}_{YH}$, $\mathbf{B}_H$ and $\mathbf{B}_Y$ are the parameters (weights and biases) of the network, typically learned by gradient descent algorithms such as backpropagation through time [17]. A sigmoid activation function σ allows the modeling of nonlinear dependencies, while the recurrent connections allow the modeling of long-term temporal dependencies. Research on RNNs has recently been boosted by the advent of General Programming Graphic Processing Units (GPGPU), and improved design of the memory cell (*Long-Short Term Memory cells* [20]). These have allowed much more efficient RNN implementations, and effective training over multiple layers (*deep RNNs*). RNNs architectures have reached state-of-the-art performances for volatility either as part of an LSTM-GARCH hybrid model [21,33] or as standalone model [26].

3.2 Datasets Description

CAC40: The available data consists of 1645 data points of the 40 time series composing the french stock market index CAC40 from 02/01/2012 to 08/06/2018 (approximately 6 years and 5 months) in OHLC (Opening, High, Low, Closing) format.

Cryptocurrencies: The available data comes from the Kaggle dataset "Every Cryptocurrency Daily Market Price"[1] constituted of 785,024 observation of 1644 different cryptotokens from 28/04/2013 to 06/06/2018. However the number of available datapoints per cryptotoken is inversely proportional to the lifespan of the token itself. In other words, the further we go into the past, the fewer values we have for our analysis, as depicted in Fig. 2. For these reasons, we restricted our analysis to the period from 28/01/2017 to 06/06/2018 for which we have complete OHLC data for 291 tokens.

[1] https://www.kaggle.com/jessevent/all-crypto-currencies.

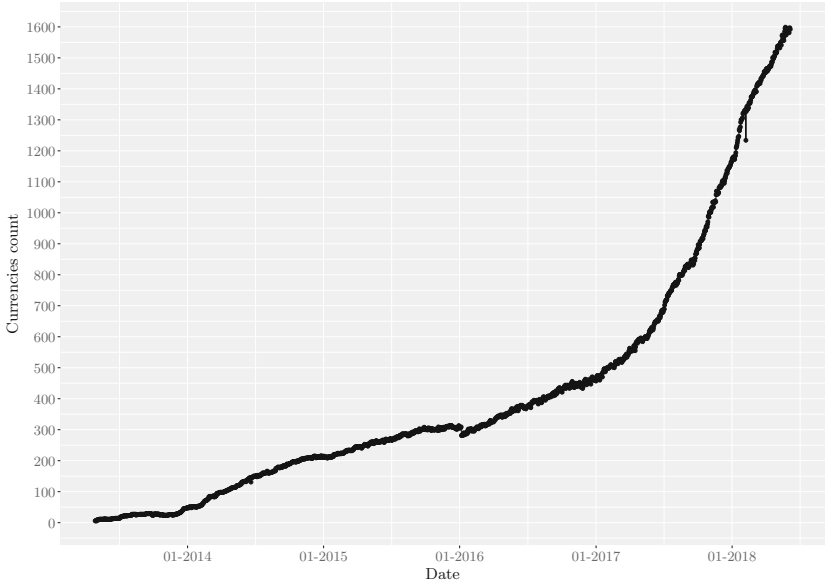


Fig. 2. Number of available datapoints for the cryptocurrencies dataset as a function of time

3.3 Volatility Proxies

The OHLC available data is composed of several 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. In the absence of detailed information concerning the price movements within a given trading days, stock volatility becomes directly unobservable [30]. To cope with such problem, several different measures (also called proxies) have been proposed in the econometrics literature [16,19,24,27] to capture this information. However, there is no consensus in the scientific literature upon what volatility proxy should be employed for a given purpose. Nevertheless, for an empirical analysis of the use of volatility proxies in the case of univariate forecasting, the interested reader could find more details in [8].

Volatility as Variance. The first family of proxies corresponds to the natural definition of volatility [27], that is, a rolling standard deviation of a given stock's continuously compounded returns over a past time window of size n:

$$\sigma_t^{SD,w} = \sqrt{\frac{1}{w-1}\sum_{i=0}^{w-1}(r_{t-i} - \bar{r}_w)^2} \tag{9}$$

where

$$r_t = \ln\left(\frac{P_t^{(c)}}{P_{t-1}^{(c)}}\right) \tag{10}$$

represents the daily continuously compounded return for day t computed from the closing prices $P_t^{(c)}$ and $\bar{r}_w$ represents the returns' average over the period $\{t, \cdots, t-w\}$. In this formulation, w represents the degree of smoothing that is applied to the original time series. We will consider here $w \in \{5, 10, 21\}$, representing respectively one week, two weeks and one month of trading.

Volatility as a Proxy of the Coarse Grained Intraday Information. The second family of proxies that we will consider is the σ_t^i one, analytically derived by [16] 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. Among all the defined proxies, we will focus on:

$$\sigma_t^0 = \left[\ln\left(\frac{P_{t+1}^{(c)}}{P_t^{(c)}}\right)\right]^2 = r_t^2 \tag{11}$$

$$u = \ln\left(\frac{P_t^{(h)}}{P_t^{(o)}}\right) \qquad d = \ln\left(\frac{P_t^{(l)}}{P_t^{(o)}}\right) \qquad c = \ln\left(\frac{P_t^{(c)}}{P_t^{(o)}}\right) \tag{12}$$

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

$$\sigma_t^4 = 0.511(u-d)^2 - 0.019[c(u+d) - 2ud] - 0.383c^2 \tag{13}$$

$$\sigma_t^6 = \underbrace{\frac{a}{f} \cdot \log\left(\frac{P_{t+1}^{(o)}}{P_t^{(c)}}\right)^2}_{\text{Nightly volatility}} + \underbrace{\frac{1-a}{1-f} \cdot \hat{\sigma}_4(t)}_{\text{Intraday volatility}} \tag{14}$$

The value of $f \in [0, 1]$ represents the fraction of the trading day in which the market is closed. It is by definition bounded in the interval $[0, 1]$, In the case of CAC40, we have that $f > 1 - f$, since trading is only performed of roughly one third of the day. Here, a is a weighting parameter, whose optimal value, according to [16] is shown to be 0.17, regardless of the value of f.

After a preprocessing phase of the datasets, involving removal of missing values and proxy calculation for each time series, the data is restructured in a multivariate time series matrix form $\mathbf{Y}$ having N (number of observations) rows and n (number of variables/time series) columns. For each proxy, this matrix is such that each row $\mathbf{Y}_t$ represent a n-dimensional vector containing the value of the given proxy for of the n variables at the time t, and the scalar value $Y_t[j]$ represent the value of jth ($j = 1, \ldots, n$) variable at time t.

4 Experimental Results

The experimental study assessed and compared the methods previously discussed in the article. The methods are listed below together with the software used for the experiments. Note that, for the sake of assessment, we set the lag $m = 2$ and the maximum number of latent factors to $q = 3$ for all methods, unless specified otherwise.

1. NAIVE: univariate baseline method using the last observed value for each time series as prediction for the following H steps.
2. UNI: univariate multi-step-ahead Direct forecasting of each individual series (Eq. 6) with a feature selection process based on correlation.
3. PLS: partial-least-squares forecasting (Sect. 3.1) implemented by the function `mvr` of the R package `pls`. The optimal values for the size of the input space and the number of principal components q is determined through an out-of-sample criterion.
4. RNN: recurrent neural network implemented by the `keras_predict` function of `kerasR`[2], the R keras interface to the `keras` Deep Learning library[3] for Theano. The network is a fully-connected RNN with 10 hidden units. Since an automated setting of the number of units would not have been feasible due to an excessive computational time, this number has been set on the basis of trial and error over a small number of synthetic series.
5. LSTM: As RNN, the model is a fully connected RNN, with 10 hidden units implemented using `kerasR`. It differs from RNN as it employs LSTM cells [20] in the hidden layer, instead of regular neurons.
6. DFM: linear Dynamic Factor Model where PCA is used for factor estimation, the number of factors is set to q and the forecasting of the factors is carried out with a VAR method implemented by the `estBlackBox` function of the R package `dse`. The batch PCA is computed using the base R `eigen` function.
7. DFML$_{PCA}$: Dynamic Factor Machine Learner where PCA is used for factor estimation, the number of factors is set to q and the forecasting of each factors is carried out in a univariate manner using a local learning predictor (lazy learning [5]) and a multi-step-ahead Direct strategy.
8. DFML$_A$: it differs from DFML$_{PCA}$ because of the use of an autoencoder instead of PCA in the process of factor estimation.
9. DFML'$_{PCA}$: it differs from DFML$_{PCA}$ because of the automatic selection strategy (described in [4]): the number of factors (in the range $[1, q]$) and the multi-step-ahead strategy (among Direct, Iterated and MIMO) and the lag m are selected by an out-of-sample strategy carried out on the training set.

4.1 Results Discussion

For each multivariate dataset we performed time series cross-validation following a rolling origin strategy [29]. The size of the training set is $2N/3$ and a sequence of 50 different test sets of length H is considered.

[2] https://github.com/statsmaths/kerasR.
[3] https://keras.io.

For each test set, all methods are assessed in terms of the average Normalized Mean Squared Error:

$$\text{NMSE} = \frac{\sum_{j=1}^{n} \text{NMSE}[j]}{n}$$

where

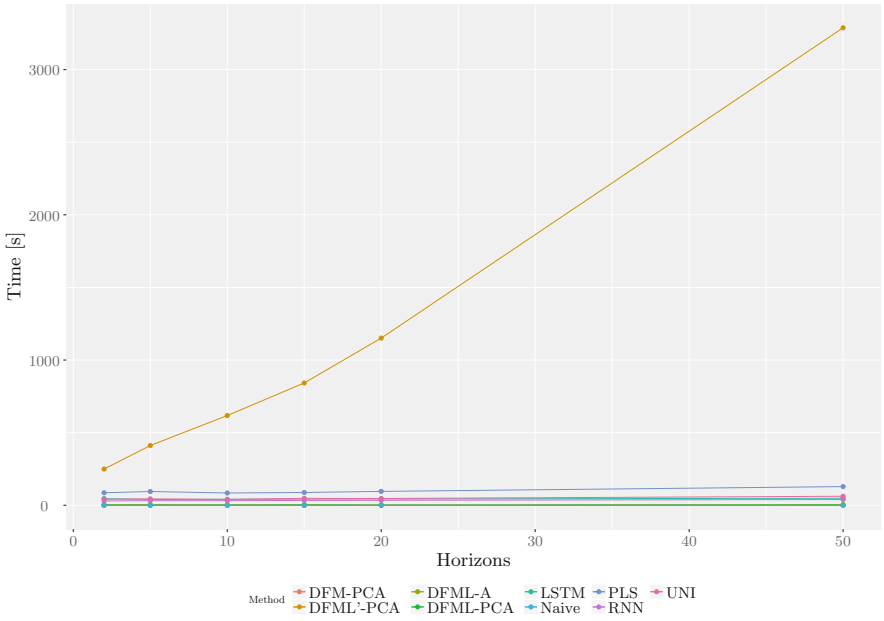
$$\text{NMSE}[j] = \frac{\sum_{h=1}^{H} (Y_{T+h}[j] - \hat{Y}_{T+h}[j])^2}{V[j]H}$$

$V[j]$ is the variance of the series $Y[j]$ and $T + 1$ is the starting index of the continuation set.

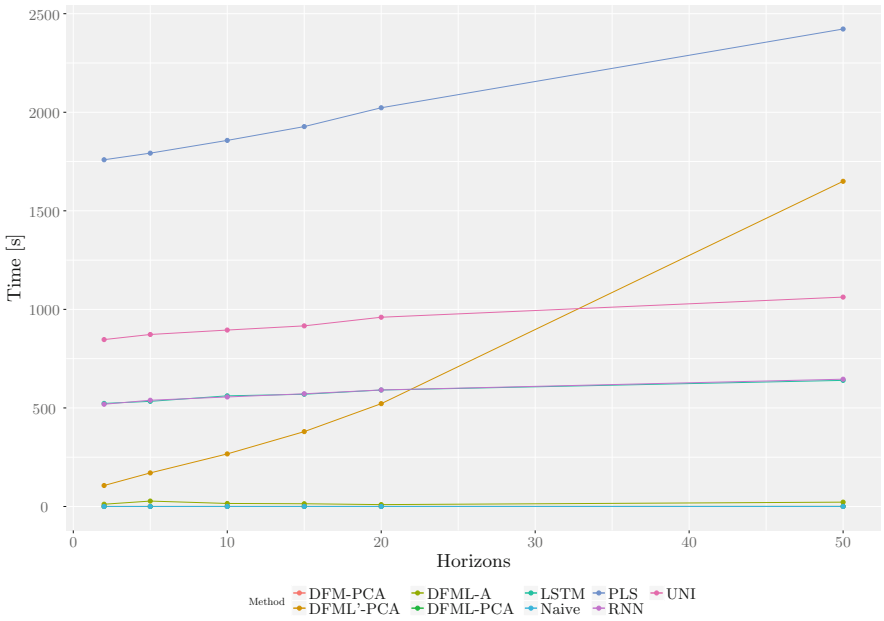
While dealing with high dimensionality ($n = 291$) coupled with a relatively low number of observations ($N = 495$), as in the case of the Cryptocurrency dataset (Table 1), using the σ_t^i family of proxies, the DFML, even without hyperparameter optimisation, clearly outperforms all the concurrent methods. It should also be noted that some methods tested in the original DFML paper [4] (i.e. VAR, DSE, SSA) could not be tested due to numerical problems related to the limited number of available observations. The performances of DFML are mitigated while using proxies coming from the $\sigma_t^{SD,w}$ family, where the performance of the Naive method improves, even for forecasting horizons up to 20 steps ahead, as the smoothing provided by the window size parameter w increases. In both the cases, a linear dimensionality reduction technique with no optimization (DFM, DFML$_{PCA}$) is shown to improve the performances of the forecaster, compared to nonlinear (DFML$_A$) and optimized (DFML'$_{PCA}$) ones.

A similar ranking among the methods is observed in the case of the CAC40 dataset (Table 2), characterized by a lower dimensionality ($n = 40$) but an higher number of points ($N = 1641$). Here we can observe a generally higher average normalized NMSE, indicating a higher complexity of the forecasting problem. For the σ_t^i family, PLS and DFM appears as competitive alternatives of the DFML, especially for longer horizons ($h > 15$). As in the previous case, for the $\sigma_t^{SD,w}$ family of proxies, the performances of the DFML family are affected by the value of the smoothing factor w, where, the higher the smoothing factor is, the less effective the DFML becomes for shorter horizons, with the Naive method becoming the best one, while still maintaining good forecasting accuracy for longer horizons.

In addition to forecasting accuracy, we also analyzed the total computational time required to produce a forecast. The total computational time is obtained by summing up the time required to train the considered model and the time needed to generate a forecast. Figure 3a shows that, for low dimensionalities ($n = 40$) the total computational time of the different techniques is comparable, and independent of the forecasting horizon, except for the optimized DFML'$_{PCA}$, where the comparison of different forecasting strategies require a computational time proportional to the length of the forecasting horizon. On the other hand, for higher dimensionalities ($n = 291$), the computational time required to train multiple univariate models (UNI), neural based models (RNN and LSTM) and PLS increases considerably due to the increase of both dimensionality and forecasting



(a)



(b)

Fig. 3. Total computational time (model training + forecast) of the tested methods on the CAC40 - σ^4 (a) ($n = 40$) and cryptocurrencies - σ^4 (b) ($n = 291$) dataset-proxy combination.

Table 1. Cryptocurrencies - volatility time series: NMSE (averaged over all the continuation sets) of the different forecasting methods. The bold notation is used to highlight all techniques which are not significantly worse (pv = 0.05) than the one with the lowest NMSE score.

Dataset	H	Naive	UNI	PLS	DFM	$DFML_A$	$DFML_{PCA}$	$DFML'_{PCA}$	LSTM	RNN
σ_t^0	2	0.988	0.660	0.630	0.595	0.631	**0.594**	0.596	0.630	0.670
	5	0.982	0.646	0.613	0.579	0.613	**0.576**	0.588	0.605	0.656
	10	1.042	0.608	0.581	0.543	0.575	**0.539**	0.538	0.570	0.615
	15	1.172	0.602	0.584	0.540	0.569	**0.537**	0.547	0.563	0.599
	20	1.247	0.579	0.555	0.515	0.544	**0.512**	0.514	0.540	0.593
	50	1.024	0.517	0.503	**0.451**	0.483	**0.451**	0.466	0.479	0.521
σ_t^4	2	0.831	0.607	0.602	0.540	0.611	**0.528**	0.543	0.585	0.647
	5	0.816	0.598	0.585	0.521	0.580	**0.510**	0.522	0.559	0.638
	10	0.945	0.582	0.579	0.505	0.564	**0.491**	0.494	0.542	0.590
	15	0.924	0.582	0.592	0.508	0.565	**0.495**	0.498	0.551	0.580
	20	1.061	0.578	0.584	0.501	0.554	**0.489**	10.969	0.539	0.575
	50	0.950	0.553	0.563	0.474	0.524	**0.472**	0.476	0.510	0.543
σ_t^6	2	0.946	0.587	0.588	0.528	0.580	**0.512**	0.527	0.547	0.613
	5	1.103	0.561	0.578	0.507	0.551	**0.479**	0.480	0.531	0.587
	10	1.101	0.583	0.590	0.516	0.579	**0.499**	0.500	0.553	0.591
	15	1.041	0.592	0.616	0.525	0.574	**0.505**	0.509	0.554	0.591
	20	1.000	0.589	0.592	0.522	0.568	**0.507**	0.509	0.551	0.586
	50	1.185	0.557	0.582	0.481	0.530	0.481	**0.474**	0.514	0.560
$\sigma_t^{SD,5}$	2	**0.269**	0.351	0.648	0.499	0.662	0.500	0.524	0.647	0.739
	5	**0.511**	0.533	0.674	0.519	0.668	0.514	0.534	0.647	0.717
	10	0.719	0.612	0.669	0.523	0.647	**0.516**	0.534	0.635	0.790
	15	0.818	0.627	0.662	0.527	0.638	**0.520**	0.523	0.616	0.794
	20	0.852	0.636	0.653	0.517	0.629	**0.514**	0.526	0.614	0.771
	50	0.974	0.611	0.636	**0.484**	0.577	0.497	0.481	0.558	0.748
$\sigma_t^{SD,10}$	2	**0.113**	0.258	0.754	0.491	0.756	0.494	0.534	0.722	0.796
	5	**0.238**	0.415	0.769	0.501	0.751	0.504	0.541	0.728	0.838
	10	**0.466**	0.598	0.781	0.513	0.746	0.507	0.543	0.719	1.027
	15	0.606	0.668	0.780	0.526	0.737	**0.514**	0.558	0.713	0.898
	20	0.668	0.706	0.777	0.523	0.741	**0.522**	0.574	0.701	0.911
	50	0.891	0.726	0.778	**0.514**	0.683	0.547	0.533	0.657	0.907
$\sigma_t^{SD,21}$	2	**0.052**	0.203	0.989	0.493	0.992	0.493	0.570	0.899	1.034
	5	**0.108**	0.316	0.986	0.501	0.977	0.504	0.571	0.864	1.144
	10	**0.199**	0.522	0.987	0.513	0.958	0.514	0.573	0.891	1.065
	15	**0.295**	0.658	0.988	0.523	0.963	0.528	0.572	0.848	1.340
	20	**0.397**	0.748	0.990	0.535	0.874	0.544	0.571	0.863	1.117
	50	0.775	0.833	1.014	**0.586**	0.882	0.649	0.612	0.808	1.252

Table 2. CAC40 - volatility time series: NMSE (averaged over all the continuation sets) of the different forecasting methods. The bold notation is used to highlight all techniques which are not significantly worse (pv = 0.05) than the one with the lowest NMSE score.

Dataset	H	Naive	UNI	PLS	DFM	$DFML_A$	$DFML_{PCA}$	$DFML'_{PCA}$	LSTM	RNN
σ_t^0	2	1.332	1.047	0.972	0.969	0.987	**0.962**	1.010	1.018	1.006
	5	2.177	1.916	1.826	1.857	1.872	1.838	**1.822**	1.849	1.865
	10	1.438	1.246	**1.157**	1.173	1.184	1.164	1.155	1.164	1.184
	15	2.499	1.304	1.220	1.220	1.227	**1.209**	1.222	1.219	1.242
	20	1.566	1.227	1.155	1.153	1.163	1.174	**1.146**	1.160	1.160
	50	2.026	1.221	1.136	1.135	1.144	1.134	**1.120**	1.160	1.164
σ_t^4	2	0.585	0.504	0.463	**0.433**	0.521	0.434	0.450	0.564	0.496
	5	2.295	1.347	1.318	1.292	1.356	**1.268**	1.275	1.328	1.346
	10	1.047	1.003	0.948	0.936	0.991	**0.911**	0.946	1.014	1.018
	15	1.372	1.132	1.078	1.067	1.118	**1.048**	1.071	1.126	1.120
	20	1.272	1.023	0.948	0.926	0.977	**0.908**	0.933	1.010	1.007
	50	1.111	1.036	0.936	0.942	0.987	**0.919**	0.981	1.052	1.042
σ_t^6	2	1.780	0.854	0.805	0.776	0.859	0.767	**0.758**	0.852	0.822
	5	1.859	1.800	1.750	**1.741**	1.809	1.747	1.715	1.781	1.770
	10	1.264	1.171	1.106	1.102	1.154	**1.083**	1.118	1.149	1.139
	15	1.222	1.074	**1.001**	**0.999**	1.049	**1.001**	1.011	1.093	1.046
	20	1.332	1.185	**1.103**	1.107	1.156	1.108	1.116	1.172	1.170
	50	1.280	1.188	1.112	1.098	1.139	**1.089**	1.126	1.206	1.177
$\sigma_t^{SD,5}$	2	**0.276**	0.649	0.834	0.783	0.877	0.787	0.769	0.823	0.864
	5	**1.122**	1.275	1.304	1.289	1.355	1.242	1.215	1.329	1.352
	10	1.329	1.199	1.163	1.139	1.167	**1.095**	1.162	1.131	1.201
	15	1.408	1.149	1.095	1.068	1.113	**1.064**	1.066	1.111	1.134
	20	1.576	1.215	1.154	1.133	1.166	**1.141**	1.150	1.203	1.182
	50	2.584	1.292	1.316	1.444	**1.184**	1.243	1.192	1.229	1.273
$\sigma_t^{SD,10}$	2	**0.453**	0.667	0.901	0.805	0.964	0.805	0.788	0.827	0.881
	5	**0.698**	0.886	1.018	0.932	1.073	0.934	0.927	0.910	1.009
	10	1.133	1.010	1.044	**0.970**	1.073	1.005	1.005	1.000	1.104
	15	1.495	1.065	1.140	1.292	1.271	1.092	**1.013**	1.066	1.032
	20	1.642	1.141	1.181	1.340	1.223	1.145	**1.078**	1.108	1.178
	50	1.916	1.258	1.233	1.256	1.158	**1.144**	1.171	1.310	1.338
$\sigma_t^{SD,21}$	2	**0.033**	0.306	0.747	0.509	0.772	0.510	0.561	0.776	0.725
	5	**0.123**	0.372	0.732	0.530	0.736	0.595	0.566	0.867	0.716
	10	**0.346**	0.520	0.808	0.660	0.853	0.682	0.673	0.992	0.932
	15	**0.608**	0.680	0.862	0.771	0.893	0.795	12.315	0.970	0.868
	20	**0.827**	**0.827**	0.923	0.905	0.890	0.840	0.777	1.010	1.256
	50	1.603	1.259	1.210	1.357	1.109	**1.076**	1.311	1.282	1.585

horizons, while DFML models, thanks to the dimensionality reduction component, maintain a reduced computational time regardless of the forecasting horizon.

5 Conclusion and Future Work

The empirical analysis shows that DFML is able to produce accurate volatility forecasts, especially in the case of high-dimensional noisy series (i.e. Cryptocurrencies dataset) with non-smoothed volatility proxies σ^i, by summarizing well the intrinsic market correlations in a reduced number of factors. However, the presence of a smoothing factor (as in the $\sigma^{SD,w}$ proxies family) is shown to worsen the performances of the DFML methods. Moreover, we have shown that, thanks to the dimensionality reduction component, DFML methods can produce multi-step ahead forecasts with the same accuracy as concurrent methods with a great reduction in terms of computational cost. In order to further improve this framework we foresee different possible extensions. On one hand we believe that the use of additional volatility proxies, together with an automated variable selection process could further improve the forecasting performances. On the other hand, the use of incremental dimensionality reduction techniques could further improve the computational efficiency of the method at the expense of a small reduction in forecasting accuracy.

References

1. Alessandretti, L., ElBahrawy, A., Aiello, L.M., Baronchelli, A.: Machine learning the cryptocurrency market. arXiv preprint arXiv:1805.08550 (2018)
2. Andersen, T.G., Bollerslev, T.: ARCH and GARCH models. Encyclopedia of Statistical Sciences (1998)
3. Bollerslev, T., Patton, A.J., Quaedvlieg, R.: Multivariate leverage effects and realized semicovariance GARCH models (2018). https://doi.org/10.2139/ssrn.3164361
4. Bontempi, G., Le Borgne, Y.A., De Stefani, J.: A dynamic factor machine learning method for multi-variate and multi-step-ahead forecasting. In: 2017 IEEE International Conference on Data Science and Advanced Analytics (DSAA), pp. 222–231. IEEE (2017)
5. Bontempi, G., Taieb, S.B.: Conditionally dependent strategies for multiple-step-ahead prediction in local learning. Int. J. Forecast. **27**(3), 689–699 (2011)
6. Catania, L., Grassi, S., Ravazzolo, F.: Forecasting cryptocurrencies financial time series. In: CAMP Working Paper Series 3, BI Norwegian Business School (2018)
7. Catania, L., Grassi, S., Ravazzolo, F.: Predicting the volatility of cryptocurrency time-series. In: CAMP Working Paper Series 5, BI Norwegian Business School (2018)
8. De Stefani, J., Caelen, O., Hattab, D., Bontempi, G.: Machine learning for multi-step ahead forecasting of volatility proxies. In: 2nd Workshop on MIning DAta for Financial Applications (MIDAS). CEUR Workshop Proceedings, Aachen, vol. 1941, pp. 17–28 (2017). http://ceur-ws.org/Vol-1941/MIDAS2017_paper3.pdf

9. De Stefani, J., Le Borgne, Y.A., Caelen, O., Hattab, D., Bontempi, G.: Batch and incremental dynamic factor machine learning for multivariate and multi-step-ahead forecasting. https://doi.org/10.1007/s41060-018-0150-x
10. Degiannakis, S.: Multiple days ahead realized volatility forecasting: single, combined and average forecasts. Glob. Financ. J. **36**, 41–61 (2018)
11. ElBahrawy, A., Alessandretti, L., Kandler, A., Pastor-Satorras, R., Baronchelli, A.: Evolutionary dynamics of the cryptocurrency market. Roy. Soc. Open Sci. **4**(11), 170623 (2017)
12. Engle III, R.F., Ito, T., Lin, W.L.: Meteor showers or heat waves? Heteroskedastic intra-daily volatility in the foreign exchange market (1988)
13. Fengler, M.R., Herwartz, H., Raters, F.: Multivariate volatility models. In: Härdle, W.K., Hautsch, N., Overbeck, L. (eds.) Applied Quantitative Finance, pp. 25–37. Springer, Heidelberg (2017). https://doi.org/10.1007/978-3-540-69179-2_15
14. Forni, M., Hallin, M., Lippi, M., Reichlin, L.: The generalized dynamic factor model. J. Am. Stat. Assoc. **100**(471), 830–840 (2005). https://doi.org/10.1198/016214504000002050
15. Franses, P., Legerstee, R.: A unifying view on multi-step forecasting using an autoregression. J. Econ. Surv. **24**(3), 389–401 (2010)
16. Garman, M.B., Klass, M.J.: On the estimation of security price volatilities from historical data. J. Bus. **53**(1), 67–78 (1980)
17. Graves, A.: Supervised Sequence Labelling with Recurrent Neural Networks. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-24797-2
18. Hafner, C.M., Herwartz, H.: Structural analysis of portfolio risk using beta impulse response functions. Statistica Neerlandica **52**(3), 336–355 (1998)
19. Hansen, P.R., Lunde, A.: A forecast comparison of volatility models: does anything beat a garch (1, 1)? J. Appl. Econometrics **20**(7), 873–889 (2005)
20. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)
21. Kim, H.Y., Won, C.H.: Forecasting the volatility of stock price index: a hybrid model integrating LSTM with multiple GARCH-type models. Expert Syst. Appl. **103**, 25–37 (2018)
22. Kim, J.M., Jung, H.: Time series forecasting using functional partial least square regression with stochastic volatility, GARCH, and exponential smoothing. J. Forecast. **37**(3), 269–280 (2018)
23. Lipton, Z.C., Berkowitz, J., Elkan, C.: A critical review of recurrent neural networks for sequence learning. arXiv preprint arXiv:1506.00019 (2015)
24. Parkinson, M.: The extreme value method for estimating the variance of the rate of return. J. Bus. **53**(1), 61–65 (1980)
25. Peng, Y., Albuquerque, P.H.M., de Sá, J.M.C., Padula, A.J.A., Montenegro, M.R.: The best of two worlds: forecasting high frequency volatility for cryptocurrencies and traditional currencies with support vector regression. Expert Syst. Appl. **97**, 177–192 (2018)
26. Petneházi, G., Gáll, J.: Exploring the predictability of range-based volatility estimators using rnns. arXiv preprint arXiv:1803.07152 (2018)
27. Poon, S.H., Granger, C.W.: Forecasting volatility in financial markets: a review. J. Econ. Lit. **41**(2), 478–539 (2003)
28. Stock, J., Watson, M.: Dynamic factor models. In: Clements, M., Hendry, D. (eds.) Oxford Handbook of Economic Forecasting. Oxford University Press, Oxford (2010)
29. Tashman, L.J.: Out-of-sample tests of forecasting accuracy: an analysis and review. Int. J. Forecast. **16**(4), 437–450 (2000). The M3-Competition

30. Tsay, R.S.: Analysis of Financial Time Series, vol. 543. Wiley, Hoboken (2005)
31. Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., Manzagol, P.A.: Stacked denoising autoencoders: learning useful representations in a deep network with a local denoising criterion. J. Mach. Learn. Res. **11**, 3371–3408 (2010)
32. Walther, T., Klein, T.: Exogenous drivers of cryptocurrency volatility - a mixed data sampling approach to forecasting (2018). https://doi.org/10.2139/ssrn.3192474
33. Yu, S.L., Li, Z.: Forecasting stock price index volatility with LSTM deep neural network. In: Tavana, M., Patnaik, S. (eds.) Recent Developments in Data Science and Business Analytics. SPBE, pp. 265–272. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-72745-5_29