Using Machine Learning Algorithms in Realized Volatility Forecasting in Presence of Jumps

Iakov Grigoryev[*]
New Economic School

May 14, 2021

Abstract

Recently, machine learning methods have become widespread, and many researchers have begun to use them to model and predict various time series. This paper compares the predictive power of some machine learning algorithms, including neural networks and random forest, with standard econometric approaches (such as the HAR model) while modeling realized volatility. A distinctive feature of the work is the inclusion of jumps in the modeling process, which makes it possible to analyze high-frequency data. The result of the work is that machine learning methods are not always able to outperform econometric approaches in predictive power, and also the addition of jump component cannot always guarantee the improvement of predictability performance.

[*]Master's Student at NES. Email: igrigoryev@nes.ru

Contents

1 Introduction

Volatility forecasting is important not only in academia, but also in business, in investment banking, in trading, and even in government decision making. Moreover, option prices, Value at Risk, Sharpe ratio and other indicators are highly dependent on volatility, which reveals the necessity of its accurate forecasts. For example, volatility is used by option traders as an input for option pricing using Black–Scholes' (Black & Scholes (1973)) and Heston's (Heston (1993)) models.

When high-frequency data appeared, the realized volatility (RV) measure was introduced by Andersen et al. (2003). Unlike GARCH-like measures, RV is model-free, and it is observable. In this paper we compare some machine learning algorithms (such as neural networks and random forests) and a classical econometric approach (the HAR model) in forecasting RV. Moreover, we will consider a pair (C_t, J_t), where C_t is a continuous part of RV and J_t is its jump component (as defined in Andersen et al. (2012)) and try to use the same machine learning algorithms and the HAR-CJ model to forecast this pair.

Luong & Dokuchaev (2018) researched the possibility to forecast RV using the random forest algorithm. Vortelinos (2017) compared the HAR model against the GARCH model, the feedforward neural network model and the principal components combining model (which will not be considered in this paper). Bucci (2020) compared different recurrent neural network (RNN) architectures applied to RV in terms of predictability performance.

The goals of this paper can be formulated as follows:

1. Compare the HAR model with the random forest algorithm and the LSTM model without adding jumps.

2. Compare the HAR-CJ model with the random forest algorithm and the LSTM model with added jumps.

3. Compare the results of the previous two points with each other, checking whether the addition of jumps improves predictability performance.

The empirical results of this paper are slightly correlated with the results of the above papers:

- In Vortelinos (2017) it was shown that the HAR model outperforms the feedforward neural network in terms of predictive power. In our paper the LSTM model was used instead of feedforward NN, and yet it still cannot beat the HAR model.

- In Luong & Dokuchaev (2018) instead of comparing the random forest algorithm with the HAR model, they were merged. In particular, after having constructed the functional form of the HAR model, its parameters are fitted using the random forest regression algorithm.

The results are that this merge can decrease the forecasting error of the model. However, in our paper these two models are compared standalone, and the results are the following: the random forest model predicts RV slightly better than the HAR-CJ model.

- In Bucci (2020) different neural network architectures were compared, whereas in our paper there is only one neural network model (the LSTM model) which turned out to be the best in many experiments of the above paper.

The remainder of this paper is structured as follows. In Section 2 we provide a literature review on all topics we cover in this paper: volatility, models. In Section 3 we give a definition of RV. In Section 4 we introduce description of all models we used to forecast realized volatility. In Section 5 we provide a data description used in the paper. In Section 6 there are results of our empirical research and in Section 7 there is a conclusion of the paper.

2 Literature Review

Volatility has been estimated and predicted a lot (see the references in Andersen & Bollerslev (1997), Dokuchaev (2014)).

The ARCH and the GARCH models were proposed by Engle (1982) and Bollerslev (1986), respectively. These models were extended in many directions: EGARCH (Nelson (1991)), GJR-GARCH (Glosten et al. (1993)), AGARCH (Engle (1990)), and TGARCH (Zakoian (1994)). But these models cannot be appropriate estimators for RV since they were developed for low-frequency data and cannot capture intradaily information well. And Vortelinos (2017) showed that the family of the GARCH models cannot indeed compete with the HAR model in terms of forecasting. So, we will not consider GARCH-like models in this paper.

As we already mentioned in introduction, Andersen et al. (2003) proposed the RV measure. The HAR model, which was introduced in Corsi (2003), was developed specifically for estimating and predicting RV. After that Barndorff-Nielsen & Shephard (2004) proposed to use so-called realized bi-power variation as a measure of continuous part of RV, and in Andersen et al. (2007) the HAR-CJ model was introduced which uses both continuous and jump components of RV to predict new values of RV. More recently, Andersen et al. (2012) proposed new measure for continuous component of RV called Median Realized Variance estimator (MedRV) which has better efficiency properties than the bi-power variation measure and displays better finite-sample robustness to jumps and small returns. We decided to use this exact estimator to divide RV into continuous and jumps components.

The algorithm of random forests was introduced by Breiman (2001) and it is basically an ensemble of some number of decision tree algorithms, which were presented in Breiman et al.

(1984). The random forest algorithm will be used in the paper.

The feedforward neural network model, according to Vortelinos (2017), is not a perfect choice for the RV forecasting. On the contrary, the recurrent neural network (RNN) architectures are applied for time series estimation and prediction not only in academia, but also in business. The LSTM neural network architecture (presented in Hochreiter (1997)) is one of the most popular RNNs, and it will be used in this paper.

3 Realized Volatility

Denote P_t the asset price at time $t \in [0, T]$, $p_t = \log P_t$. The asset return is $r_{t,\Delta t} = p_{t+\Delta t} - p_t$ for some Δt such that $t + \Delta t \leq T$. The assumption is that p_t can be represented by the following diffusion Ito equation:

$$dp_t = \mu_t dt + \sigma_t dW_t + \kappa_t dq_t, \quad 0 \leq t \leq T,$$

where W_t is Brownian motion, μ_t and σ_t are predictable processes with σ_t being the standard deviation of dp_t and independent of dW_t, and q_t is the number of jumps with time-varying intensity κ_t. Firstly, assume that jump component $\kappa_t dq_t$ is absent. The term $\mu_t dt$ is called drift. From the diffusion equation it follows that

$$r_{t,\Delta t} = p_{t+\Delta t} - p_t = \int_0^{\Delta t} \mu_{t+\tau} d\tau + \int_0^{\Delta t} \sigma_{t+\tau} dW(t+\tau).$$

It is true not only for the segment $[t, t+\Delta t]$, but also for any its subsegment $[\bar{t}, \bar{t} + \delta t] \subset [t, t+\Delta t]$. If δt is small enough, the drift term can be excluded, and we have

$$r_{\bar{t},\delta t} \approx \int_0^{\delta t} \sigma_{\bar{t}+\tau} dW(\bar{t}+\tau) = \sigma_{\bar{t}+\epsilon\delta t}(W_{\bar{t}+\delta t} - W_{\bar{t}}),$$

where $0 < \epsilon < 1$ (it follows from the first mean theorem).

Since W_t is Brownian motion, $W_{\bar{t}+\delta t} - W_{\bar{t}} \sim N(0, \delta t)$, thus, $(W_{\bar{t}+\delta t} - W_{\bar{t}})^2 \sim \delta t \times \chi^2(1)$, which has a mean of δt. We can then take δt as an approximate value of $(W_{\bar{t}+\delta t} - W_{\bar{t}})^2$ when δt is small enough. So, we have

$$r^2_{\bar{t},\delta t} \approx \sigma^2_{\bar{t}+\epsilon\delta t}\delta t = \sigma^2_{\bar{t}+\epsilon\delta t} \int_0^{\delta t} 1 d\tau = \frac{\sigma^2_{\bar{t}+\epsilon\delta t}}{\sigma^2_{\bar{t}+\gamma\delta t}} \int_0^{\delta t} \sigma^2_{\bar{t}+\tau} d\tau, \tag{3.1}$$

where $0 < \gamma < 1$ (again, from the first mean theorem).

Consider even splitting $\left\{t_i = t + \frac{\Delta t}{m} \times i\right\}_{i=0}^{m-1}$ of the segment $[t, t+\Delta t]$. Then realized variance can be calculated as follows:

$$\mathrm{RV}_t^2 = \sum_{i=0}^{m-1} r_{t_i, \frac{\Delta t}{m}}^2 .$$

Why is it important? From equation (3.1) it follows that

$$\mathrm{RV}_t^2 \xrightarrow[m\to\infty]{p} \int\limits_0^{\Delta t} \sigma_{t+\tau}^2 d\tau,$$

as for each $i = 0, \ldots, m-1$

$$\frac{\sigma^2_{t_i+\epsilon_i \frac{\Delta t}{m}}}{\sigma^2_{t_i+\gamma_i \frac{\Delta t}{m}}} \xrightarrow[m\to\infty]{} 1.$$

Integrated variance (IV) is defined as follows:

$$\mathrm{IV}_t = \int\limits_0^{\Delta t} \sigma_{t+\tau}^2 d\tau,$$

and we showed that RV_t^2 is a consistent estimator for IV_t (we assume the absence of jumps). Integrated variance is of a particular interest in academia and in business since it measures the variation of an asset within the time period Δt. Therefore, the realized variance prediction is important in research and in applications.

Now let us incorporate jump component. And in the presence of jumps, one can prove that

$$\mathrm{RV}_t^2 \xrightarrow[m\to\infty]{p} \underbrace{\int\limits_0^{\Delta t} \sigma_{t+\tau}^2 d\tau}_{\mathrm{IV}_t} + \underbrace{\sum_{t<\tau\le t+\Delta t} \kappa_\tau^2}_{K_t} .$$

So, realized variance is a consistent estimator of the sum of integrated variance IV_t and jump component K_t. The problem is that both IV_t and K_t are not observable. And our goal is to find consistent estimators for them. In other words, we need to find observable processes C_t and J_t such that

$$\mathrm{RV}_t^2 = C_t + J_t, \quad C_t \xrightarrow[m\to\infty]{p} \mathrm{IV}_t, \quad \text{and } J_t \xrightarrow[m\to\infty]{p} K_t.$$

Barndorff-Nielsen & Shephard (2004) proposed so-called bi-power variation measure:

$$\mathrm{BV}_t = \frac{2}{\pi} \cdot \left(\frac{m}{m-2}\right) \cdot \sum_{i=2}^{m-1} \left|r_{t_{i-2}, \frac{\Delta t}{m}}\right| \left|r_{t_i, \frac{\Delta t}{m}}\right| \xrightarrow[m \to \infty]{p} \mathrm{IV}_t.$$

The fact that $\mathrm{BV}_t \xrightarrow[m \to \infty]{p} \mathrm{IV}_t$ is proven in the same paper. Thus, we could take $C_t = \mathrm{BV}_t$ and $J_t = \mathrm{RV}_t^2 - C_t$.

However, the calculation of BV_t greatly depends on the sampling frequency m. Therefore, due to market microstructure, BV_t cannot converge to IV_t when $m \longrightarrow \infty$, and, thus, it cannot be a jump-robust estimator of IV_t. And Andersen et al. (2012) proposed new estimator MedRV_t:

$$\mathrm{MedRV}_t = \frac{\pi}{6 - 4\sqrt{3} + \pi} \cdot \left(\frac{m}{m-2}\right) \cdot \sum_{i=1}^{m-2} \mathrm{Med}\left(\left|r_{t_{i-1}, \frac{\Delta t}{m}}\right|, \left|r_{t_i, \frac{\Delta t}{m}}\right|, \left|r_{t_{i+1}, \frac{\Delta t}{m}}\right|\right)^2.$$

In that paper it is proven that MedRV estimator dampens the effect of jumps at a faster asymptotic rate than any multi-power (including bi-power) variation estimator. And we decided to use it as C_t process. Overall, in our paper:

$$C_t = \mathrm{MedRV}_t, \quad J_t = \mathrm{RV}_t^2 - C_t.$$

Finally, realized volatility (RV) is simply a square root of realized variance (we denote it with RV_t). Starting from the next section let us assume that $\Delta t = 1$ day (so that each day we calculate RV based on intradaily data).

4 Models

We took the typical examples of the models within their classes: the HAR model (econometric approach), the Random Forest model (ensemble) and the LSTM model (neural network).

4.1 HAR Model

Denote:

$$\mathrm{rv}_t = \log \mathrm{RV}_t^2, \quad \mathrm{rv}_t^n = \frac{\sum_{i=0}^{n-1} \mathrm{rv}_{t-i}}{n}.$$

Then, the daily Heterogeneous Autoregressive (HAR) model can be expressed as follows:

$$\mathrm{rv}_t = \beta_0 + \beta_d \mathrm{rv}_{t-1}^1 + \beta_w \mathrm{rv}_{t-1}^5 + \beta_m \mathrm{rv}_{t-1}^{22} + \epsilon_t, \ t \in [22, T],$$

where subscripts "d", "w", and "m" stand for "day", "week", and "month", respectively. $\mathbb{E}(\epsilon_t|I_{t-1}) = 0$ for $t \in [22, T]$, where $I_{t-1} = \{\mathrm{rv}_0, \ldots, \mathrm{rv}_{t-1}\}$.

Not let us introduce the HAR-CJ model. In this model, opposite to the HAR model, we use continuous and jump components of RV instead of RV itself as regressors. Denote:

$$c_t = \log C_t, \quad j_t = \log(J_t + 1),$$

$$c_t^n = \frac{\sum_{i=0}^{n-1} c_{t-i}}{n}, \quad j_t^n = \frac{\sum_{i=0}^{n-1} j_{t-i}}{n}.$$

We use +1 in the definition of j_t to deal with very small (yet positive) values of J_t. Here we follow Andersen et al. (2007).

The HAR-CJ model can be expressed as follows:

$$\mathrm{rv}_t = \beta_0^c + \beta_{cd} c_{t-1}^1 + \beta_{cw} c_{t-1}^5 + \beta_{cm} c_{t-1}^{22} + \beta_{jd} j_{t-1}^1 + \beta_{jw} j_{t-1}^5 + \beta_{jm} j_{t-1}^{22} + \epsilon_t, \; t \in [22, T].$$

Again, $\mathbb{E}(\epsilon_t|I_{t-1}) = 0$ for $t \in [22, T]$.

Finally, we introduce a modified version of the HAR-CJ model, which predicts both continuous and jumps parts of RV:

$$c_t = \beta_0^c + \beta_{cd}^c c_{t-1}^1 + \beta_{cw}^c c_{t-1}^5 + \beta_{cm}^c c_{t-1}^{22} + \beta_{jd}^c j_{t-1}^1 + \beta_{jw}^c j_{t-1}^5 + \beta_{jm}^c j_{t-1}^{22} + \epsilon_t^c, \; t \in [22, T],$$

$$j_t = \beta_0^j + \beta_{cd}^j c_{t-1}^1 + \beta_{cw}^j c_{t-1}^5 + \beta_{cm}^j c_{t-1}^{22} + \beta_{jd}^j j_{t-1}^1 + \beta_{jw}^j j_{t-1}^5 + \beta_{jm}^j j_{t-1}^{22} + \epsilon_t^j, \; t \in [22, T].$$

Here $\mathbb{E}(\epsilon_t^c|I_{t-1}) = \mathbb{E}(\epsilon_t^j|I_{t-1}) = 0$ for $t \in [22, T]$. Then, to predict rv_t, we use the following formula: $\mathrm{rv}_t = \log(\exp(c_t) + \exp(j_t) - 1)$.

In all three models we use ordinary least squares to estimate parameters. Therefore, we can report the results of the estimation in the table and analyze them.

4.2 Random Forest Model

For this algorithm, as well as for the next LSTM algorithm, we will build a dependency of rv_t (or (c_t, j_t)) on some number k of previous observations $\mathrm{rv}_{t-1}, \ldots, \mathrm{rv}_{t-k}$ ($c_{t-1}, \ldots, c_{t-k}, j_{t-1}, \ldots, j_{t-k}$, respectively), where k is a hyperparameter. In this paper we take $k = 10$, this value was found using cross-validation with $65 : 5 : 30$ proportion of train, val and test sets and a grid $(1, 3, 10, 30, 100)$ of possible values of the hyperparameter. So, let us call rv_t ((c_t, j_t), respectively) a target and $\mathrm{rv}_{t-1}, \ldots, \mathrm{rv}_{t-k}$ ($c_{t-1}, \ldots, c_{t-k}, j_{t-1}, \ldots, j_{t-k}$, respectively) features. For each $t \geq k$ we have a separate object with a target and features. Denote our sample as X, it consists of vectors $((\mathrm{rv}_{t-1}, \ldots, \mathrm{rv}_{t-k}), \mathrm{rv}_t)$

$(((c_{t-1}, \ldots, c_{t-k}, j_{t-1}, \ldots, j_{t-k}), (c_t, j_t))$, respectively), $t \in [k, T]$.

To fully describe the random forest algorithm, we first should define the decision tree method:

1. Let N be the root node of the tree with all available data.
2. Check whether some stop condition holds. If so, make N a leaf node and assign to it the average target of the objects in this node (meaning that while forecasting, if an out-of-sample object ends up being in this node, than we predict this value as a target). Otherwise, go ahead to the next step.
3. Select the feature F and a threshold T, such that all data is separated into two subsets: X_{TRUE} and X_{FALSE}, where data with $F < T$ is going to X_{TRUE} and otherwise in X_{FALSE}. And we select F and T maximizing some criterion $Q(X, F, T)$.
4. Assign a pair (F, T) to the node N, make two new nodes N_{TRUE} and N_{FALSE}, passing to them X_{TRUE} and X_{FALSE}, respectively.
5. Repeat steps 2-4 with both nodes each considered as a new root of a respective subtree.

To finish the definition, we should set a particular stop condition and a criterion $Q(X, F, T)$. The latter is defined as follows:

$$Q(X, F, T) = H(X) - \frac{|X_{TRUE}|}{|X|} H(X_{TRUE}) - \frac{|X_{FALSE}|}{|X|} H(X_{FALSE}),$$

where $H(X) = \min\limits_{c \in \mathbb{Y}} \frac{1}{|X|} \sum\limits_{((x_1^i, \ldots, x_k^i), y^i) \in X} L(y^i, c)$, and $L(y, c)$ is some loss function we choose. $\mathbb{Y}$ is a set of possible targets, in our case it is $\mathbb{R}$. In this paper the standard MSE loss function was used for random forests: $L(y, c) = (y - c)^2$. The stop condition is the following: stop whenever the best splitting into two subsets yields negative criterion value $Q(X, F, T)$. When it is positive, then the process of subtree construction continues. Actually, for checking this condition we need to calculate $Q(X, F, T)$ for each pair of (F, T) to take the best one, so we can use these calculations to select the optimal pair on step 3 (of course, if stop condition does not hold).

Now we can define an algorithm of random forest. Hyperparameters (besides those for decision trees) are n_{tree} (number of trees in the ensemble) and m (the number of features in each tree). The algorithm is the following:

1. Use bootstrap to generate n_{tree} samples from the original one (and use each of them while training a separate decision tree).

2. For i_{tree} from 1 to n_{tree}: train a decision tree on i_{tree}'s generated bootstrap sample, but when selecting the best splitting of each node (i.e. on step 3 of tree construction) use only m features, whose numbers are randomly chosen before each splitting.

3. Return the average of the predictions of all trees (i.e. while forecasting, we get the predictions from each tree and after that the averaged value will be a prediction from a random forest).

Random forest makes the variance of the overall algorithm smaller, which prevents overfitting. However, it does not change the bias, so it cannot help with underfitting (if it exists). Therefore, we should use decision trees with large depth (so that they are complex enough not to be underfitted). In fact, in this paper we do not restrict the depth of trees at all (using the only stop condition described above).

In this paper we chose $n_{tree} = 100$ and $m = 10$. Again, the same cross-validation process was used for that (with the same proportion and the same set of possible hyperparameter values).

We will also calculate feature importance after estimating the model. To do that, there exist two methods:

1. Feature importance based on mean decrease in impurity. It is computed as the mean and standard deviation of accumulation of the impurity decrease within each tree. However, this method has a bias toward high-cardinality features and cannot be computed on out-of-sample data.

2. Feature importance based on feature permutation. Features are shuffled n times and the model is refitted to estimate the importance of features. This method overcomes limitations of the impurity-based feature importance, and therefore, we use this second approach.

4.3 LSTM Model

The general neural network consists of the following elements:

- neurons (or nodes), grouped in
- layers, connected to each other;
- weights of neurons and
- (non-linear) activation functions for each layer.

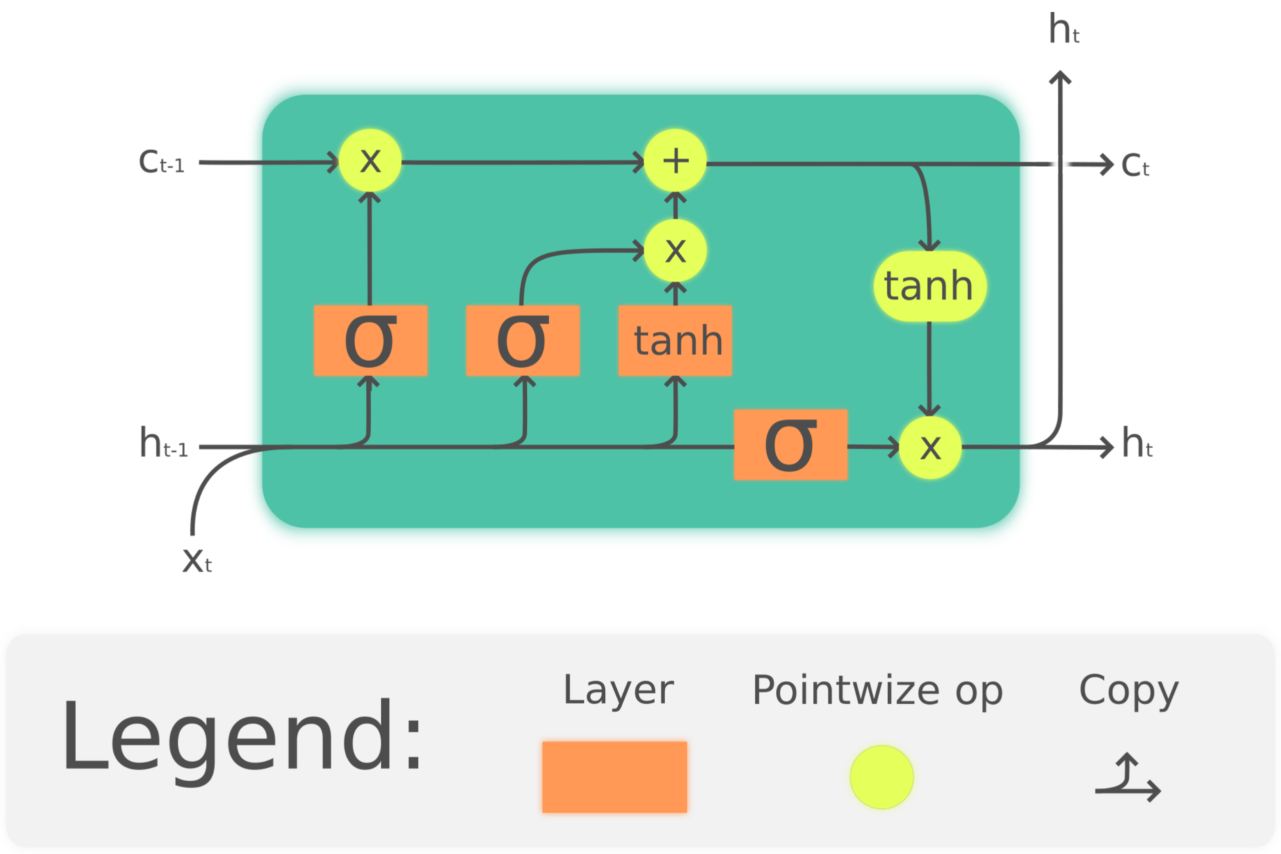
Layers are numerated from 0 (the input of the neural network) to L (the output of the neural network), all layers except the input and the output are called hidden layers. Each node in layer $l-1$ is connected to each node in layer l via arrows and weights under arrows. Each layer l is specified by an activation function $a_l(\cdot)$, so that the value in the node of layer $l+1$ is an activation function applied to the linear combination of values of nodes of layer l with weights under arrows entering this node, including a bias term. Activation functions need to be non-linear so that neural network can approximate non-linear dependencies. We will use the only hidden layer in this paper since Donaldson & Kamstra (1996) showed that a single hidden layer neural network can approximate any non-linear function given enough number of neurons in this hidden layer. Overall, the output function of one hidden layer neural network is

$$f(X, W, b) = a_1 \left(b^1 + \sum_{j=1}^{q} a_0(b_j^0 + XW^0)W_j^1 \right),$$

where $k, q, 1$ are sizes of layers $0, 1, 2$ respectively, W^0 (of size (q, k)) is a matrix consisting of weights connecting the input layer and hidden layer, and W^1 (of size $(1, q)$) is a row vector consisting of weights connecting the hidden layer and the output layer, b^0 (of size $(q, 1)$) is a column vector consisting of bias terms in the hidden layer, and b^1 (scalar) is a bias term in the output layer. X (of size (m, k)) is an input matrix consisting of m examples, so that $f(X, W, b)$ has size $(m, 1)$ approximating the target values on these examples. Remember (from the previous point) that k is the number of features (regressors). In this paper we have $k = 10$ and $q = 100$.

As already mentioned in Section 2, feedforward neural networks are not ideal for estimating and forecasting time series. And we will use the Long Short-Term Memory (LSTM) model for it.The hidden layer in this model (which is the only as we already discussed) is represented by the so-called memory block. Each block includes one or more self-connected memory cells and is equipped with three multiplicative units called input, forget and output gates.

Figure 1: LSTM memory cell



Define the following activation functions:

$$\sigma(z) = \frac{1}{1+e^{-z}},$$

$$\tanh(z) = \frac{e^{z}-e^{-z}}{e^{z}+e^{-z}}.$$

The illustration of the LSTM memory cell is given on Figure 1 on page 11. x_t denotes the input vector at time t, c_t is a cell state at time t, h_t is a hidden state or an output state at time t. The input gate is i_t, whereas f_t is the forget gate and o_t is the output gate. The entire process can be described by the following equations:

$$f_t = \sigma\left(W_f h_{t-1} + U_f x_t + b_f\right),$$

$$i_t = \sigma\left(W_i h_{t-1} + U_i x_t + b_i\right),$$

$$\bar{c}_t = \tanh\left(W_c h_{t-1} + U_c x_t + b_c\right),$$

$$c_t = f_t \circ c_{t-1} + i_t \circ \bar{c}_t,$$

$$o_t = \sigma\left(W_o h_{t-1} + U_o x_t + V_o c_t + b_o\right),$$

$$h_t = o_t \circ \tanh(c_t),$$

$$\hat{y}_t = h_t.$$

Here $\circ$ is the Hadamard product function. As well as for the HAR model, we will use the standard MSE loss function to estimate parameters of the model.

Unfortunately, for the LSTM model there are no well-known measures of feature importance.

5 Data

We used RV of S&P 500 Index, calculated by Oxford-Man Institute of Quantitative Finance (ticks of 5 minutes were used to calculate RV of each day). MedRV estimator is calculated there as well. The period is from 1/3/2000 to 1/14/2020, which is splitted into an in-sample and an out-of-sample periods with a proportion of $7:3$.

All the code and data can be found here.

6 Results

The following models were estimated:

1. HAR for rv_t;
2. HAR-CJ for rv_t;
3. HAR-CJ (modified) for (c_t, j_t);
4. LSTM for rv_t;
5. LSTM for (c_t, j_t);
6. Random Forest for rv_t;
7. Random Forest for (c_t, j_t).

Start with the HAR model. Here are the results of its estimation:

Dep. Variable:	rv_t	**R-squared:**	0.704
Model:	HAR	**Adj. R-squared:**	0.704
Method:	Least Squares	**F-statistic:**	2911.
Date:	Fri, 14 May 2021	**Prob (F-statistic):**	0.00
Time:	15:05:14	**Log-Likelihood:**	-3200.3
No. Observations:	3674	**AIC:**	6409.
Df Residuals:	3670	**BIC:**	6433.
Df Model:	3		

	coef	**std err**	**t**	**P> |t|**	**[0.025**	**0.975]**
const	-0.4778	0.104	-4.576	0.000	-0.682	-0.273
$\mathbf{rv}_{t-1}^{1}$	0.2837	0.020	14.292	0.000	0.245	0.323
$\mathbf{rv}_{t-1}^{5}$	0.4699	0.032	14.768	0.000	0.408	0.532
$\mathbf{rv}_{t-1}^{22}$	0.1972	0.026	7.591	0.000	0.146	0.248

Omnibus:	100.465	**Durbin-Watson:**	2.041
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	202.082
Skew:	0.173	**Prob(JB):**	1.31e-44
Kurtosis:	4.096	**Cond. No.**	185.

Here we can see that R^2 is quite big, all parameters are statistically significant, and rv_{t-1}^{5} has the most (in absolute value) influence on the target variable.

Here are the results of the HAR-CJ model estimation:

Dep. Variable:	rv_t	**R-squared:**	0.715
Model:	HAR-CJ	**Adj. R-squared:**	0.714
Method:	Least Squares	**F-statistic:**	1530.
Date:	Fri, 14 May 2021	**Prob (F-statistic):**	0.00
Time:	15:05:14	**Log-Likelihood:**	-3133.9
No. Observations:	3674	**AIC:**	6282.
Df Residuals:	3667	**BIC:**	6325.
Df Model:	6		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
const	-0.9834	0.161	-6.119	0.000	-1.299	-0.668
c_{t-1}^1	0.3140	0.019	16.702	0.000	0.277	0.351
c_{t-1}^5	0.3396	0.031	10.947	0.000	0.279	0.400
c_{t-1}^{22}	0.1692	0.028	6.144	0.000	0.115	0.223
j_{t-1}^1	294.5850	91.844	3.207	0.001	114.515	474.655
j_{t-1}^5	95.4613	185.510	0.515	0.607	-268.252	459.175
j_{t-1}^{22}	217.7981	211.996	1.027	0.304	-197.844	633.440

Omnibus:	182.787	**Durbin-Watson:**	1.899
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	358.815
Skew:	0.358	**Prob(JB):**	1.21e-78
Kurtosis:	4.353	**Cond. No.**	5.12e+05

As we can see, not all parameters are statistically significant (among jump variables only a parameter near daily jump is statistically significant), R^2 slightly increased and a big condition number might indicate that there are strong multicollinearity or other numerical problems.

Finally, here are the results of the estimation of modified HAR-CJ model:

Dep. Variable:	c_t	**R-squared:**	0.736
Model:	HAR-CJ (modified)	**Adj. R-squared:**	0.735
Method:	Least Squares	**F-statistic:**	1702.
Date:	Fri, 14 May 2021	**Prob (F-statistic):**	0.00
Time:	15:05:14	**Log-Likelihood:**	-3292.1
No. Observations:	3674	**AIC:**	6598.
Df Residuals:	3667	**BIC:**	6642.
Df Model:	6		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
const	-0.7012	0.168	-4.179	0.000	-1.030	-0.372
c_{t-1}^1	0.3786	0.020	19.288	0.000	0.340	0.417
c_{t-1}^5	0.3836	0.032	11.846	0.000	0.320	0.447
c_{t-1}^{22}	0.1734	0.029	6.029	0.000	0.117	0.230
j_{t-1}^1	293.6056	95.887	3.062	0.002	105.609	481.602
j_{t-1}^5	-45.8729	193.676	-0.237	0.813	-425.597	333.851
j_{t-1}^{22}	-26.5368	221.328	-0.120	0.905	-460.475	407.402

Omnibus:	73.843	**Durbin-Watson:**	2.023
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	131.618
Skew:	-0.148	**Prob(JB):**	2.63e-29
Kurtosis:	3.879	**Cond. No.**	5.12e+05

Dep. Variable:	j_t	**R-squared:**	0.416
Model:	HAR-CJ (modified)	**Adj. R-squared:**	0.416
Method:	Least Squares	**F-statistic:**	436.2
Date:	Fri, 14 May 2021	**Prob (F-statistic):**	0.00
Time:	15:05:14	**Log-Likelihood:**	27984.
No. Observations:	3674	**AIC:**	-5.595e+04
Df Residuals:	3667	**BIC:**	-5.591e+04
Df Model:	6		

	coef	**std err**	**t**	**P> |t|**	**[0.025**	**0.975]**
const	0.0001	3.37e-05	3.861	0.000	6.41e-05	0.000
c_{t-1}^1	1.805e-05	3.94e-06	4.578	0.000	1.03e-05	2.58e-05
c_{t-1}^5	2.502e-05	6.51e-06	3.846	0.000	1.23e-05	3.78e-05
c_{t-1}^{22}	-3.203e-05	5.78e-06	-5.544	0.000	-4.34e-05	-2.07e-05
j_{t-1}^1	0.2198	0.019	11.413	0.000	0.182	0.258
j_{t-1}^5	0.0998	0.039	2.564	0.010	0.023	0.176
j_{t-1}^{22}	0.4980	0.044	11.199	0.000	0.411	0.585

Omnibus:	7235.671	**Durbin-Watson:**	2.037
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	34845663.699
Skew:	15.200	**Prob(JB):**	0.00
Kurtosis:	479.132	**Cond. No.**	5.12e+05

For continuous part we can see that statistical significance of the parameters are the same as in the previous model, R^2 slightly increased again, and again, we have a big condition number. As for jump part j_t, all parameters are statistically significant and R^2 is less than it was in all previous HAR models and in continuous part of this model.

Now let us turn to RF models. Start with the simple model, where RV is predicted using the previous $k = 10$ values of the RV process. A plot of feature importance for this model (details are provided in Section 4) is represented on Figure 2 on page 16.

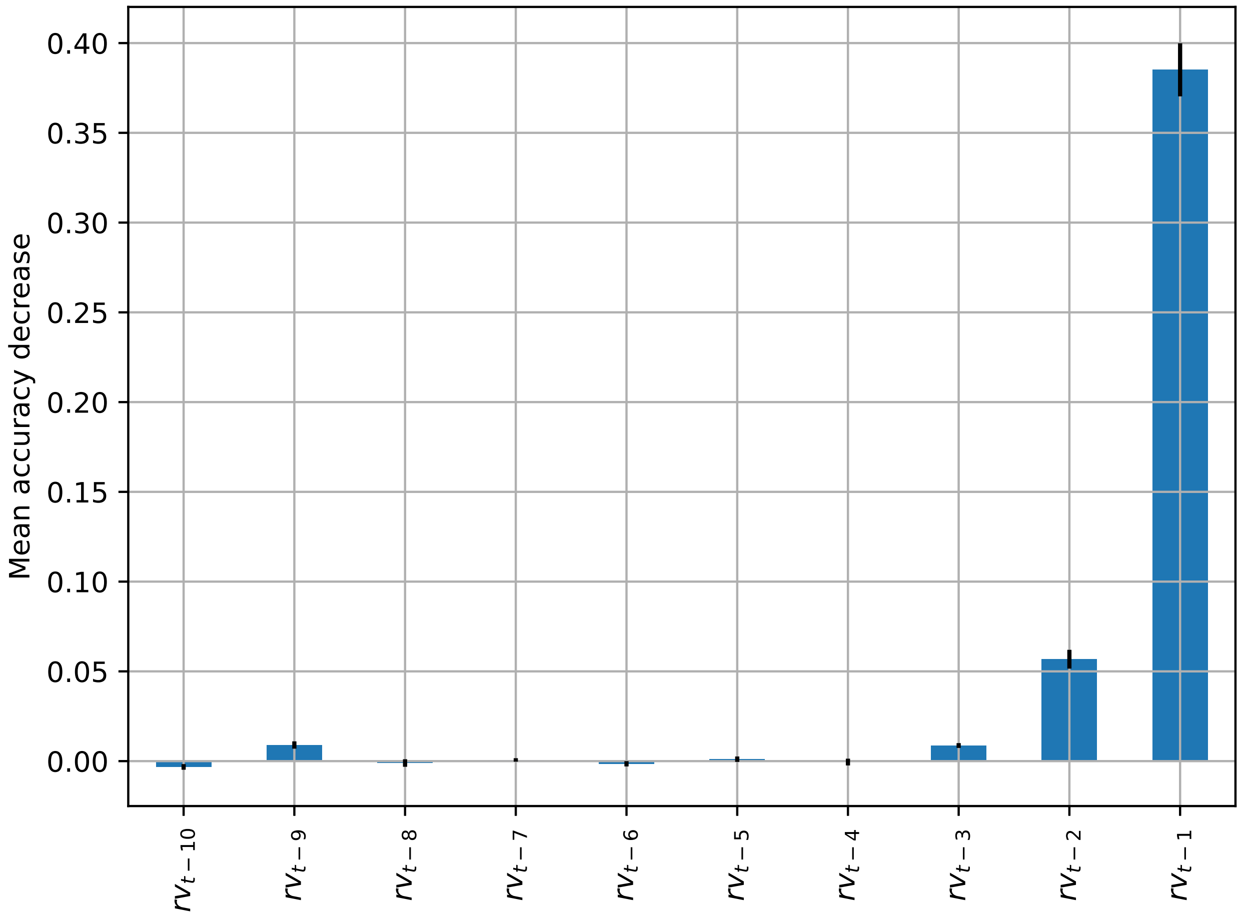


Figure 2: Feature Importance for the simple RF model

Results are straightforward: the most important feature is rv_{t-1} which is the previous value of RV. It is reasonable since this value is the closest to the one we predict.

Plots of feature importance for the modified RF model (predicting a pair of continuous and jump components of RV) are represented on Figures 3–4.

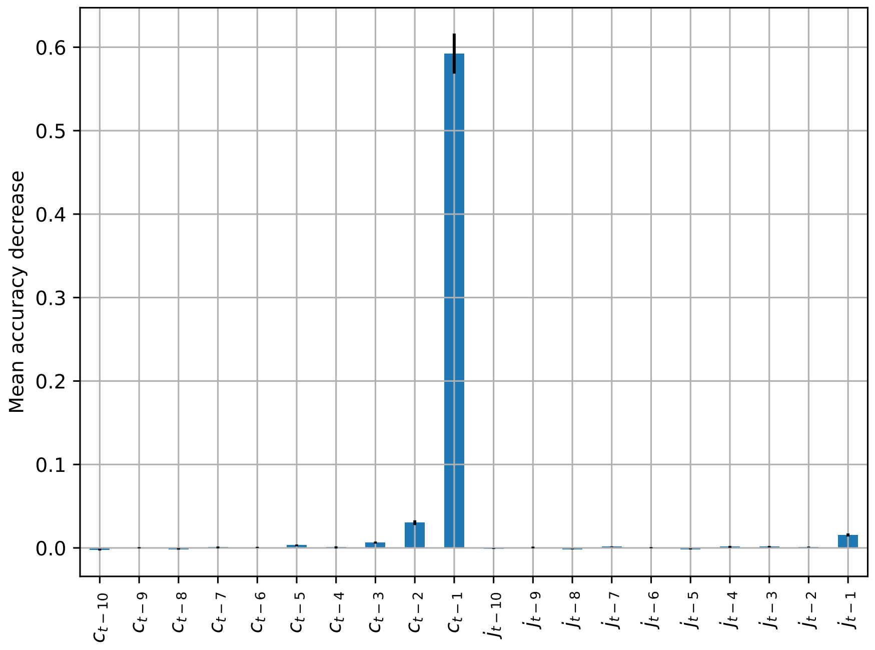


Figure 3: Feature Importance for the modified RF model (continuous part)

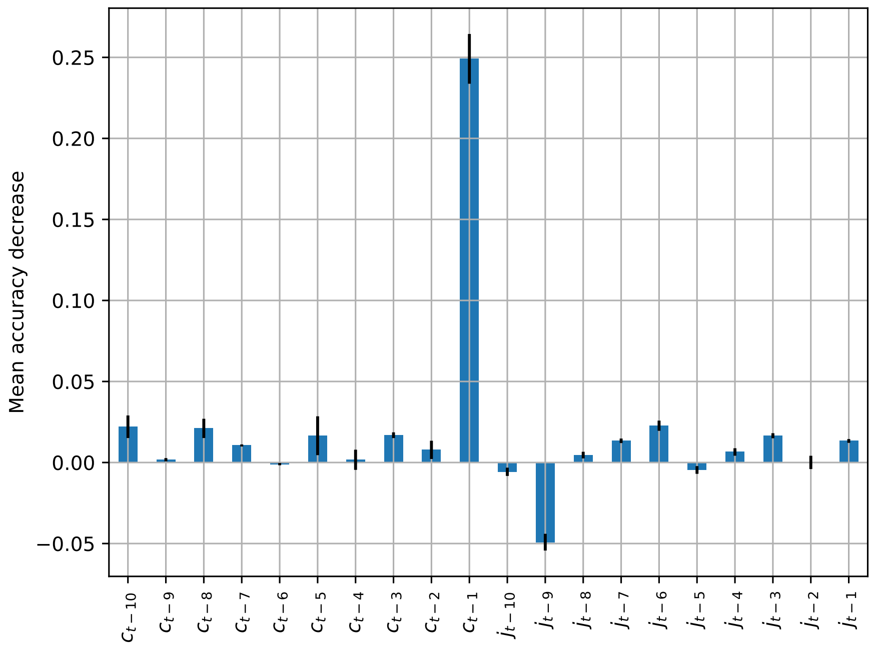


Figure 4: Feature Importance for the modified RF model (jump part)

For continuous part the results are again reasonable: the most important feature is the previous value of a continuous component. However, for jump part the same previous value of a continuous component is the most important. And also some features decrease accuracy (those which have negative bars on the plot).

Now let us move to predictability performance. We can achieve the first and the third goals from the introduction looking at one table (predictions are one-step-ahead and made on out-of-sample data):

Error/Model	HAR	HAR-CJ	HAR-CJ (modified)	LSTM
MSE for rv_t	0.441	0.466	0.723	1.848
MSE for RV_t^2	3.73×10^{-8}	3.67×10^{-8}	3.32×10^{-8}	6.98×10^{-8}

Error/Model	LSTM (modified)	RF	RF (modified)
MSE for rv_t	1.497	0.478	0.641
MSE for RV_t^2	6.90×10^{-8}	3.10×10^{-8}	3.36×10^{-8}

The following table compares errors in separate components (continuous and jump) of RV:

Error/Model	HAR-CJ	LSTM	RF
MSE for c_t	0.513	1.780	0.520
MSE for j_t	1.90×10^{-8}	2.77×10^{-8}	2.07×10^{-8}
MSE for C_t	4.69×10^{-9}	1.38×10^{-8}	4.87×10^{-9}
MSE for J_t	1.91×10^{-8}	2.77×10^{-8}	2.08×10^{-8}

What we can see at these tables:

1. If we take error on rv_t as the main criterion, than the best model is the simplest HAR model, and it is not worth adding jumps and use machine learning algorithms.

2. However, if we take error on RV_t^2 as the main criterion (and this is usually the case), the best model is simple RF model, and again it is not worth adding jumps.

3. But if we look at the HAR model, after the addition of jumps the error (of RV_t^2) decreased. Moreover, when we started to predict continuous and jumps components of RV separately, the error decreased again.

4. The LSTM model is the worst in all comparisons, but it is worth mentioning that addition of jumps decreased the error.

5. If we look at continuous and jump components separately, the HAR model predicts them slightly better than the RF model.

Finally, let us plot graphs of one-step-ahead predictions on out-of-sample data of different models. They are represented on Figures 5–12.

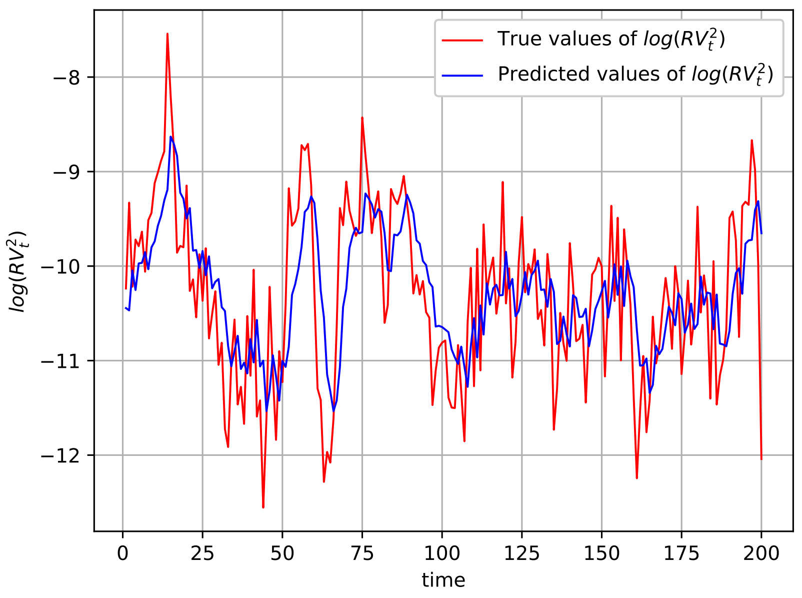


Figure 5: The HAR model, one-step-ahead predictions

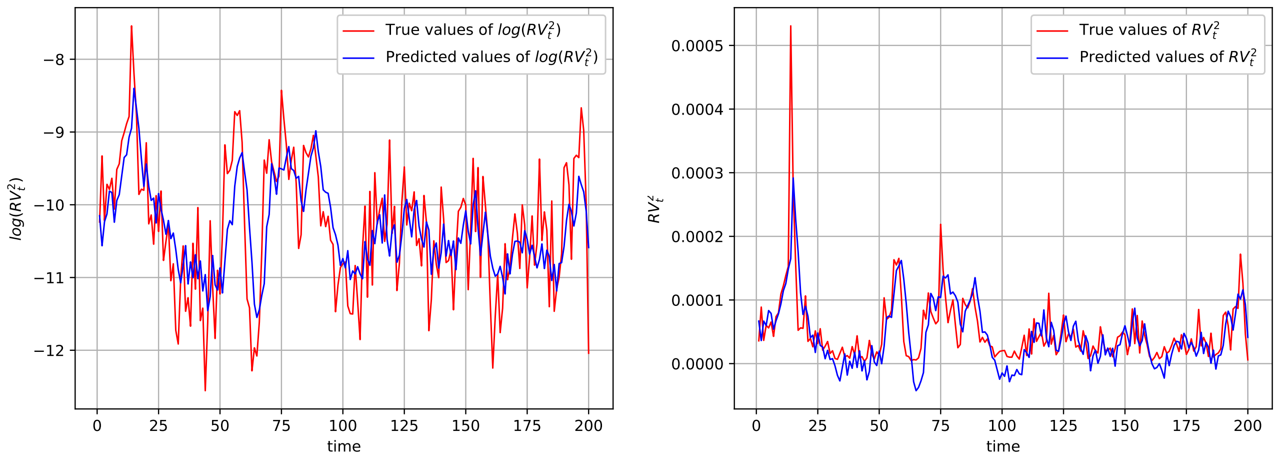


Figure 6: The HAR-CJ model, one-step-ahead predictions

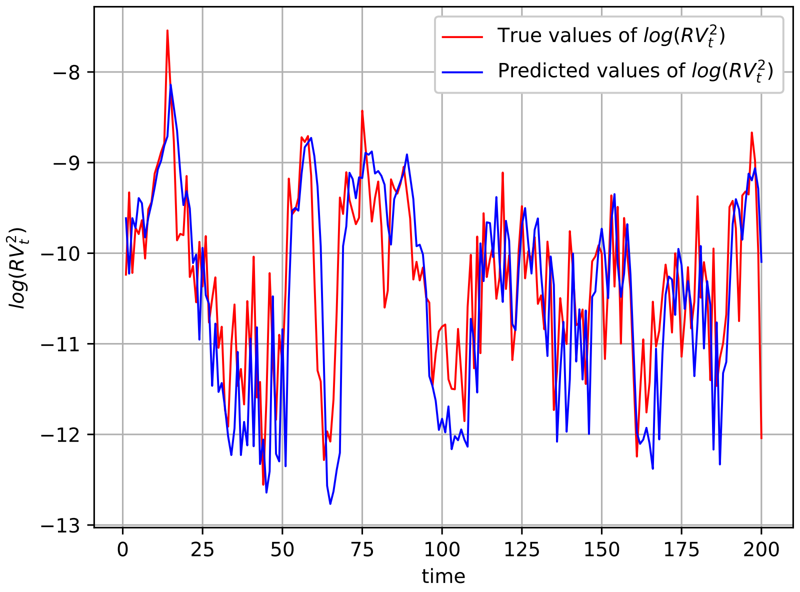


Figure 7: The HAR-CJ model (modified), one-step-ahead predictions

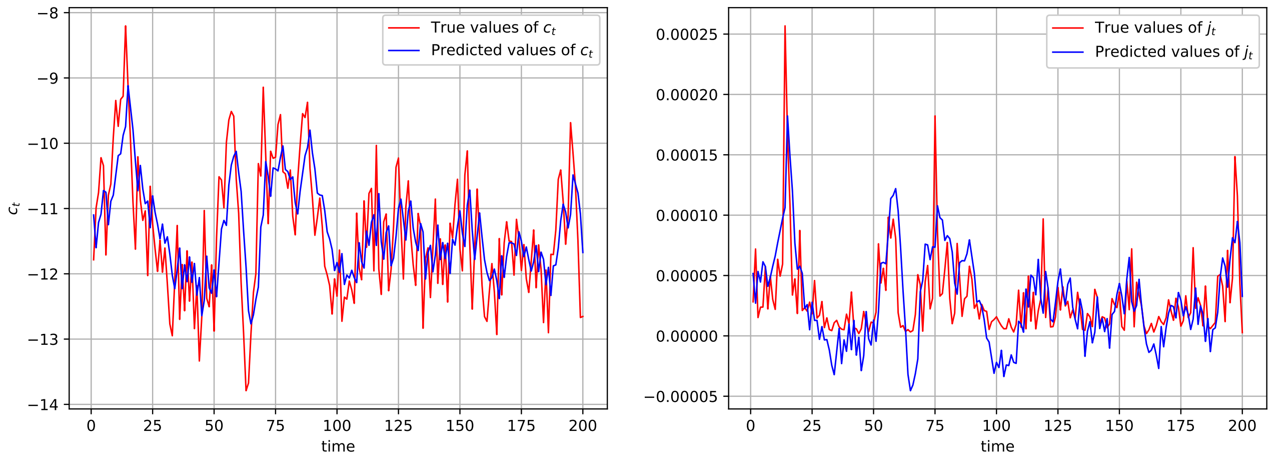
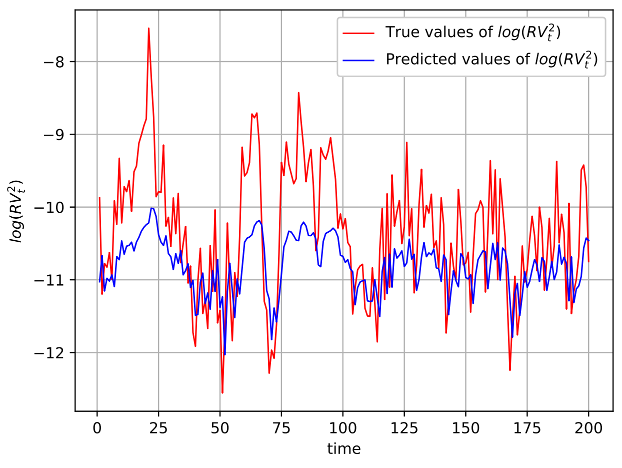


Figure 8: The HAR-CJ model (modified), continuous and jump component, one-step-ahead predictions



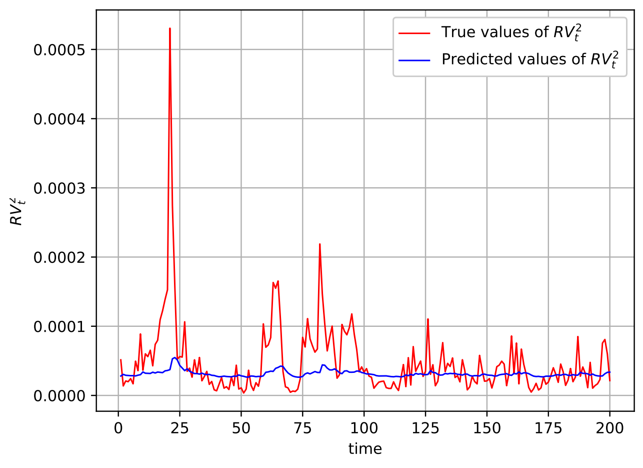
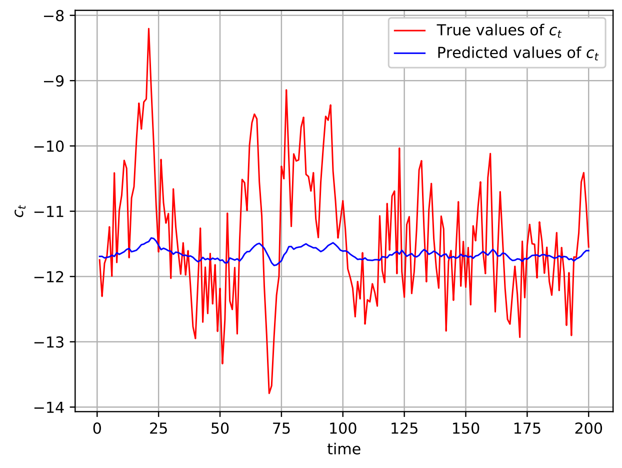


Figure 9: The LSTM model, one-step-ahead predictions



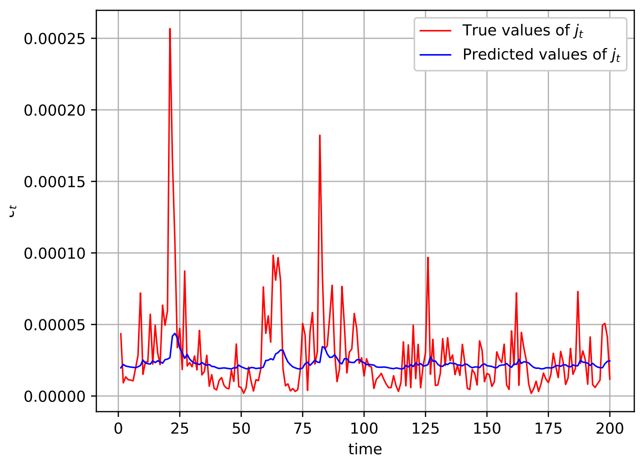


Figure 10: The LSTM model (modified), continuous and jump component, one-step-ahead predictions

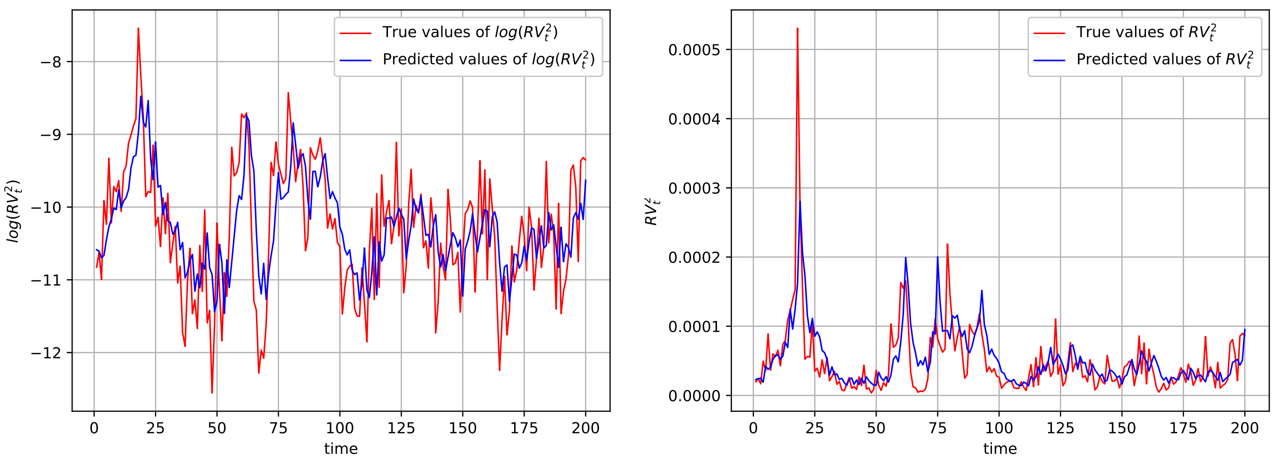


Figure 11: The RF model, one-step-ahead predictions

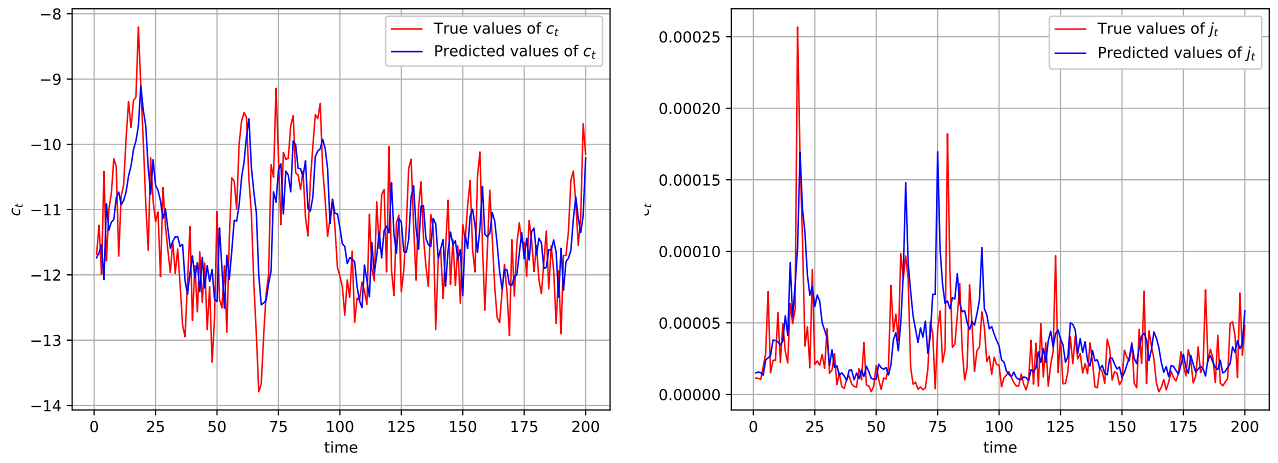


Figure 12: The RF model (modified), continuous and jump component, one-step-ahead predictions

Note that such poor results of the LSTM model are predictable since this model (as well as all neural networks in general) need much more data points than we have (more than 100,000 points is acceptible, but we have less than 10,000 points).

7 Conclusion

To summarize the results of the paper:

1. We proved that not always machine learning algorithms are better that standard econometric approaches. Indeed, in most comparisons the HAR model was better than both LSTM and RF model.
2. We proved that not always the addition of jumps improves the accuracy of predictions: for the RF model it is not true, for the HAR model it is true when we use one criterion and false if we use another.
3. We saw that neural networks can perform badly, and it is due to the small dataset size.

References

Andersen, T. G. & Bollerslev, T. (1997). Heterogeneous information arrivals and return volatility dynamics: Uncovering the long run in high frequency data. *Journal of Finance*, 52, 975–1005.

Andersen, T. G., Bollerslev, T., & Diebold, F. X. (2007). Roughing it up: including jump components in the measurement, modeling, and forecasting of return volatility. *The Review of Economics and Statistics*, 89(4), 701–720.

Andersen, T. G., Bollerslev, T., Diebold, F. X., & Labys, P. (2003). Modeling and forecasting realized volatility. *Econometrica*, 71, 529–626.

Andersen, T. G., Dobrev, D., & Schaumburg, E. (2012). Jump-robust volatility estimation using nearest neighbor truncation. *Journal of Econometrics*, 169, 75–93.

Barndorff-Nielsen, O. E. & Shephard, N. (2004). Power and bipower variation with stochastic volatility and jumps. *Journal of Financial Econometrics*, 2, 1–37.

Black, F. & Scholes, M. (1973). The Pricing of Options and Corporate Liabilities. *Journal of Political Economy*, 81, 637–54.

Bollerslev, T. (1986). Generalized Auto Regressive Conditional Heteroskedasticity. *Journal of Econometrics*, 31, 307–27.

Breiman, L. (2001). Random Forests. *Machine Learning*, 45, 5–32.

Breiman, L., Friedman, J. H., Olshen, R. A., & Stone, C. J. (1984). *Classification and Regression Trees*. Monterey, CA: Wadsworth and Brooks.

Bucci, A. (2020). Realized Volatility Forecasting with Neural Networks. *Journal of Financial Econometrics*, 18(3), 502–531.

Corsi, F. (2003). A Simple Approximate Long-Memory Model of Realized Volatility. *Journal of Financial Econometrics*, 7, 174–96.

Dokuchaev, N. (2014). Volatility estimation from short time series of stock prices. *Journal of Nonparametric Statistics*, 26, 373–84.

Donaldson, G. & Kamstra, M. (1996). Forecast Combining with Neural Networks. *Journal of Forecasting*, 15, 49–61.

Engle, R. F. (1982). Autoregressive conditional heteroskedasticity with estimates of variance of UK inflation. *Econometrica*, 50, 987–1008.

Engle, R. F. (1990). Discussion: Stock market volatility and the crash of 87. *Review of Financial Studies*, 3, 103–6.

Glosten, L. R., Jagannathan, R., & Runkle, D. E. (1993). On the relationship between the expected value and the volatility of the nominal excess return on stocks. *Journal of Finance*, 46, 1779–801.

Heston, S. L. (1993). A closed-form solution for options with stochastic volatility with applications to bond and currency options. *The Review of Financial Studies*, 6, 327–43.

Hochreiter, S. (1997). Long Short-Term Memory. *Neural Computation*, 9(8), 1735–1780.

Luong, C. & Dokuchaev, N. (2018). Forecasting of Realised Volatility with the Random Forests Algorithm. *Journal of Risk and Financial Management*, 11, 61.

Nelson, D. B. (1991). Conditional Heteroskedasticity in Asset Returns: A New Approach. *Econometrica*, 59, 347–70.

Vortelinos, D. I. (2017). Forecasting realized volatility: HAR against Principal Components Combining, neural networks and GARCH. *Research in International Business and Finance*, 39, 824–839.

Zakoian, J.-M. (1994). Threshold Heteroscedastic Models. *Journal of Economic Dynamics and Control*, 18, 931–55.