Master Thesis

Econometrics and Management Science - Quantitative Finance

Incorporating Realized Volatililty In Bitcoin Volatility Forecasting

Ries Schoot Uiterkamp (470973)

Supervisor :	Dr. Rutger-Jan Lange
Second assessor :	Dr. Onno Kleen
AMDAX Supervisor :	Marcel Burger, MSc.

September 26, 2022

Abstract

This paper aims to forecast Bitcoin volatility using two GARCH type models, two Bayesian Markov-switching models and two long short-term memory neural networks. Where for each model type we use one model utilizing squared daily returns and one that incorporates realized variance. These models are used to perform a series of one-step ahead forecasts of the realized volatility of Bitcoin. These are evaluated based on RMSE, MAE, daily VaR forecasts and corresponding expected shortfall. The results show that out of the six models that we test, the realized GARCH model performs the best for all metrics. It has the lowest RMSE and MAE, the best exceedance ratio for value at risk forecasts, and the second lowest expected shortfall. So the more complex and computationally expensive models do not manage to perform better than the relatively simple realized GARCH. We find that the models that incorporate realized variance outperform their respective counterpart in terms of the VaR and expected shortfall forecasts, while for two of the three model types it also improves the RMSE and MAE.

Contents

1 Introduction

Risk plays an important part in finance. It is one of the most thought about concepts while also being one of the least tangible. For example, the capital asset pricing model (CAPM) states a direct relationship between the (required) return on an asset and its risk. Similarly, the most important factor in option pricing is the future risk of the underlying asset, its implied volatility. A good volatility forecast is useful in a business sense, in that it is used in risk management, derivative pricing and portfolio management among others. In each, it is the predictability of volatility that is required. A portfolio manager wants to know the volatility of the portfolio constituents to obtain portfolio variance. An option trader will want to know the volatility that can be expected over the future life of the contract, and be able to make trades concerning the implied volatility in the market. A risk manager might want to get volatility forecasts in order to calculate value at risk (VaR) for a certain position or portfolio. Therefore, having a good volatility model is beneficial in many different aspects of finance.

While volatility forecasting is important for basically all asset classes, it might be even more important in the case of cryptocurrencies. Over the last decade, cryptocurrencies as Bitcoin (Nakamoto, 2008) and Ethereum (Buterin, 2013) have gone from obscure concepts for cryptography researchers and enthusiasts, to one of the most widely known financial assets. While one can very well argue that the market of Bitcoin and ether have matured the last years, it still demonstrates high levels of volatility and big intraday market movements. As illustration, since 2021 we have seen an increase of over 100% followed by a drop of over 50% twice. In such a volatile market, good risk management risk management should play an very important role as the consequences of being on the wrong side of things is severe. A good example of this is the crypto 'hedge fund' Three Arrows Capital, whose CEO quoted the now infamous phrase: "Those who do not manage their risk will have the market manage it for them." (Zhu, 2021), just eight months before they went bankrupt. All in all, there is a big incentive to find what volatility models perform well on this new asset class.

The goal of this paper is to find the effect of incorporating realized measures, specifically realized volatility, into different volatility models on the forecasting power of said models on Bitcoin. We look at three different types of models with two different specifications, one using 'normal' squared daily returns, and one using realized variance.

The first model type we will be using will be two simple GARCH type models. These types of models are very easy to estimate and interpret and are therefore widely used. This makes them very well suited to be used to compare with more complex models as a benchmark. We use the GJR-GARCH specification of Glosten, Jagannathan, and Runkle (1993), which is able to model the leverage effect that is present in most financial assets. Next to that we use the Realized GARCH model of Hansen, Huang, and Shek (2012), that

allows for different realized measures, as well as being able to model the leverage effect.

Secondly we use two Markov-switching models, a type of regime switching model that assumes that future states only depend on the current state. We implement a two-state Markov-switching GJR-GARCH model as well as a Markov-switching realized variance model, where the returns and realized variance are jointly modeled. These models divide the sample into two different regimes, in this case a high volatility and a low volatility regime. Then based on the regime parameters get estimated. This is attractive to volatility modeling as it could help with volatility clustering, the observation that high volatility tends to be followed by high volatility, and low volatility by low volatility. This would suggest the presence of two distinct regimes in volatility behavior.

The last two models we use are long short-term memory (LSTM) neural networks. This is a type of recurrent neural network that exhibits long-term memory, something that also is present in volatility through high persistence. Compared to the different models we have mentioned so far LSTM and neural networks in general are less interpretable. We feed panel data of certain pre-specified features into the model combined with a target variable. Then during the training process the model tweaks it internal functions in order to produce an output that minimizes the loss function. This allows for non-linear relations between variables that other models might not be able to model. We actually combine these LSTM models with the two GARCH models mentioned above. By adding GARCH estimates as parameters we hope to add extra information to our feature set. The first LSTM model contains rolling GJR-GARCH estimates as well as squared daily returns as unique features. The other LSTM model uses rolling Realized GARCH estimates and realized variance as unique features. Both models also use a set of shared features. In this way we can clearly see the effect of using realized variance on volatility forecasts.

To evaluate the different models and their volatility forecasts we will be looking both at metrics to determine the quality of the forecasts itself, as well as using value at risk and expected shortfall for looking at the forecasts from a portfolio management perspective. There can be a severe disconnect between the two, as from a portfolio management perspective one cares mostly about whether the actual volatility is higher than a forecast. On the other hand, one that would use these forecasts to price derivatives would care about the difference to the actual volatility, both when its positive and negative. We find that the Realized GARCH model scores the best for both mean absolute error and root mean squared error. The other models have very similar values to each other for the exception of the LSTM model using squared daily returns, which performs a lot worse than the others. We find that with these two metrics the 'simple' benchmark models perform better than the more complex and computationally expensive models.

When we look at value at risk and expected shortfall, we find that both GJR-GARCH and LSTM with

squared daily returns both provide a too conservative VaR forecasts, with exceedance ratios clearly below the intended 5%. LSTM with realized variance has the VaR estimates that are exceeded the most, with an exceedance ratio of 9.4%. Its expected shortfall is the lowest, but this is partly caused by the high amount of exceedances, which mostly happens at not very negative returns. For expected shortfall the three realized variance models all outperform the models using squared daily returns. This suggests that even though the VaR gets exceeded more often, this happens at relatively small negative returns instead of big negative returns. The three models utilizing squared daily returns have the highest expected shortfall, which confirms one of the disadvantages of GARCH models, that they are not great at modeling spikes in volatility. We conclude that utilizing realized variance over squared daily returns can improve the volatility forecasting ability of different model types. We tested four models that were more complex and computationally expensive, but none of them were able to outperform the Realized GARCH model.

In the next section, we discuss the literature pertaining to the models, methodology, and the problem field as a whole. In section 3, we elaborate on the data set that will be used and what sample we use. Section 4, describes the methodology and model specifications. In section 5 we show our results, and finally, we present our conclusions in section 6.

2 Literature

After the seminal paper of Engle (1982) that introduced ARCH (Autoregressive Conditional Heteroskedastic) models, and the generalizing paper of Bollerslev (1986) introducing GARCH, we have seen a large amount of attention on time-varying volatility and how to measure, model and forecast it. There has been nothing less than an arms race to develop new and better models to get grip on financial market volatility, such as EGARCH (Nelson, 1991), TGARCH (Zakoian, 1994) and IGARCH (Engle & Bollerslev, 1986).

While conditional variance models like ARCH and GARCH do quite well in practice, they all have a flaw in their specification. Standard GARCH models use daily returns (often squared returns) to obtain information about the current level of volatility. And this information is used to generate expectations about next period's volatility. Unfortunately, squared returns are a very noisy volatility measure. In response to this different realized measures of volatility have been introduced such as realized variance and the realized kernel see Andersen, Bollerslev, Diebold, and Ebens (2001), Barndorff-Nielsen and Shephard (2002) and Barndorff-Nielsen, Hansen, Lunde, and Shephard (2008) among others. As these measures use higher frequency data to obtain their value, they inherently contain more available information than squared daily returns. We have seen different models that incorporate realized variance or a different realized metric in order to improve volatility forecasts. From the fractionally integrated ARFIMA models of Andersen, Bollerslev, Diebold, and Labys (2003) and long-memory Heterogeneous AR (HAR) model of Corsi (2009) to the Realized GARCH

model of Hansen et al. (2012).

We have also seen an increase in the usage of neural networks to model financial time series. These types of models do not have an as extensive history in financial literature, but follow from breakthroughs in computer science. The last decades has seen major breakthroughs in deep learning, such as Alexnet (Krizhevsky, Sutskever, & Hinton, 2012) which introduced convolutional neural networks. The advent of graphical processing units made it so large models could efficiently be trained, while at the same time the amount of data that is generated everyday has skyrocketed. All in all it has been a perfect storm for deep learning to blossom. This eventually found its way to the world of finance, where people tried incorporating different models. Vidal and Kristjanpoller (2020) uses a Long Short-Term Memory (LSTM) network in order to forecast the volatility for gold prices, while Xiong, Nichols, and Shen (2015) uses deep learning to forecast volatility using google search data. Others like Sirignano and Cont (2019) use deep learning models to find a relation between order flow and price movements.

While the above-mentioned models have seen much testing on a variety of financial assets like stocks, stock indices, commodities and currencies, the amount of literature covering the results of applying these models to cryptocurrencies is naturally quite low. Volatility modeling for cryptocurrencies has relied mostly on the GARCH framework, Dyhrberg (2016) uses an asymmetric GARCH model to research Bitcoin's hedging capabilities. Chu, Chan, Nadarajah, and Osterrieder (2017) fit a selection of 12 GARCH models on different seven different cryptocurrencies to find out which models perform the best. There have been some papers that researched the use of Markov-switching GARCH models on Bitcoin, such as Ardia, Bluteau, and Rüede (2019) and Tan, Koh, Ng, and Ng (2021). There has been little research into the effects of using realized volatility to forecast the volatility of Bitcoin returns. This is the part where this paper will add to the literature, by giving a clear comparison between models using daily returns and models utilizing realized volatility.

3 Data

In this paper we will be focusing on Bitcoin. Bitcoin is the first cryptocurrency, after being first deployed in 2009 (Nakamoto, 2008). It is currently the biggest and most well known cryptocurrency out there, with at the time of writing a market cap of $440,020,465,493, which constitutes 40% of the entire cryptocurrency market cap. The data that we will be using is obtained from Glassnode, a cryptocurrency data provider. We use both daily closing price data as well as 10 minute prices to construct the realized variance. Liu, Patton, and Sheppard (2015) argues that 5 minute data is the best in terms of calculating the realized variance, unfortunately we do not have that data available, so we use 10 minute data to construct the realized volatility.

Our sample is from 01-01-2016 to 24-06-2022, and contains 2367 daily observations. As Bitcoin is still a relatively young asset we have to choose from what time we think the price movement of Bitcoin is useful and representative of what it is now. For any earlier than 2016 it will be quite hard to argue that the price action is similar to what it is now, as there was little liquidity and did not have many participants. For this reason we do not consider the data before 2016 as we do not think this data is representative and useful. We will divide this sample into a training set and a test set. Training of the models will be done on the sample that is from 01-01-2016 up to 01-06-2020, which consists of 1613 daily observations. The models are then tested on data from 01-06-2020 to 24-06-2022. Figure 1 shows the price series over our sample. This graph

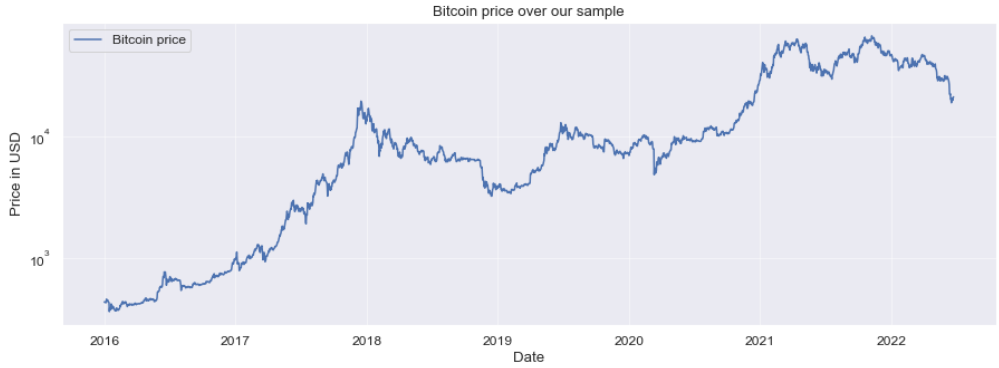

Figure 1: Daily Bitcoin closing prices from 01-01-2016 to 24-06-2022

shows the volatile nature of Bitcoin, we see the meteoric rise in value in 2016 and 2017, and the following downturn in 2018. In total we have four periods of major growth, with the main one being 2017.

Now we will look deeper into the dynamics of Bitcoin returns during our sample. First we transform the daily prices into log returns:

$$y_t = \log\left(\frac{P_t}{P_{t-1}}\right). \tag{1}$$

In this formula P_t denotes the closing price of Bitcoin at time t.

Figure 2 shows these log returns for our sample, combined with the density plot of the log returns. We can see one major negative return, which corresponds with 12th of march 2020, when Bitcoin's price dropped nearly 40% in one day in response to the corona virus outbreak. Other than that the distribution of log returns shows some negative skew and clear presence of long tails. This is further confirmed by the results reported in Table 1, where we can see the summary statistics of Bitcoin's log returns. The table also shows the value of the Jarque-Bera test for normality, we can see that normality of the log returns is rejected.

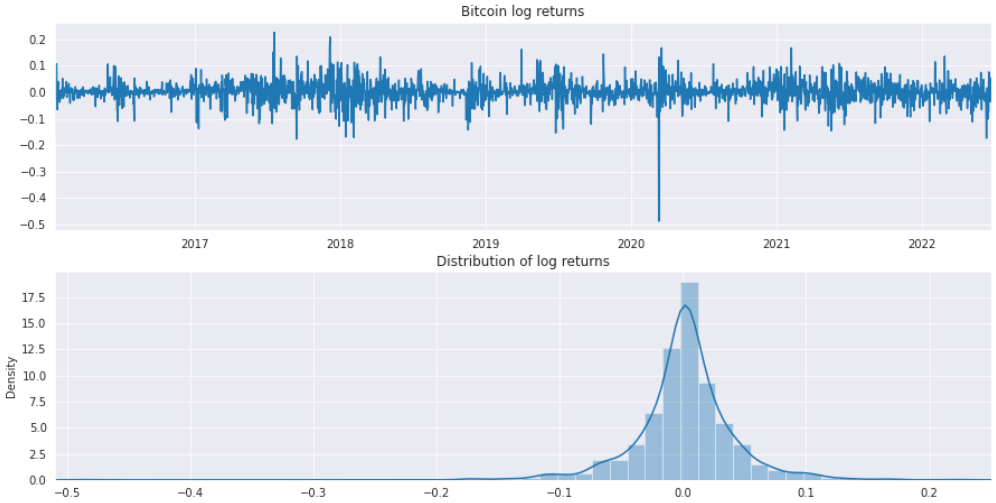

Figure 2: Bitcoin log returns from 01-01-2016 to 24-06-2022 and the density plot of those log returns

Table 1: Descriptive statistics of the log returns of Bitcoin

	BTC
Mean	0.002
St Dev	0.040
Minimum	-0.487
Maximum	0.226
Skewness	-0.804
Kurtosis	12.805
Jarque-Bera	15760
p-value	0.000

4 Methodology

4.1 GJR-GARCH

GARCH type models are one of the base models of volatility modeling. They are easily estimated, well interpretable and have a useful and easy closed form forecast. As a relatively simple model GARCH has shown good results in modeling volatility for different asset classes. This generalized model got first introduced in Bollerslev (1986) and is defined as

$$\begin{aligned} y_t &= \mu + z_t\sqrt{h_t}, \\ h_t &= \omega + \alpha\varepsilon_{t-1}^2 + \beta h_{t-1}, \end{aligned} \tag{2}$$

where y_t is the log return,$z_t = y_t/\sqrt{h_t} \sim i.i.d(0,1)$ and $\varepsilon_t = z_t\sqrt{h_t}$. While being a generally good model, GARCH has its shortcomings. One of the characteristics that GARCH is not able to capture is the leverage effect, where negative returns have a greater effect on future volatility than positive returns of the same

magnitude. Therefore we will be using the GJR-GARCH model of Glosten et al. (1993), which specification is

$$\begin{aligned} y_t &= \mu + z_t\sqrt{h_t}, \\ h_t &= \omega + (\alpha + \gamma I[\varepsilon_{t-1} < 0])\varepsilon_{t-1}^2 + \beta h_{t-1}. \end{aligned} \tag{3}$$

In this model an extra parameter γ is introduced that can capture the leverage effect. This is a characteristic of most financial assets where negative losses have bigger impact on volatility than gains of the same magnitude, such that we expect $\gamma > 0$. We will use this GJR-GARCH model as the benchmark to compare the other models to, since it is very easy to estimate and relatively simple. So more complex or computationally expensive models should be able to outperform this model to be worth the added complexity. Forecasting for this model is relatively straightforward and will be a series of one-step ahead forecasts.

4.2 Realized GARCH

The first realized variance model we will be looking at is the Realized GARCH model of Hansen et al. (2012). In this model a second measurement equation is added to the GARCH of Bollerslev (1986) to allow for the incorporation of different realized measures. We will be using a log-linear Realized-GARCH(1,1) specification that uses realized variance as its realized measure. We define realized variance as

$$RV_t = \sum_{i=1}^{n} r_{t+i/n}^2. \tag{4}$$

$$\text{where } r_{t+i/n} = p_{t+i/n} - p_{t+i-1/n}$$

Here n is the amount of parts a day gets divided, so for 10 minute realized variance this would be 144, p_t is the log price at time t. So the realized variance is defined as the sum of intraday squared returns. The specification of the Realized GARCH model is:

$$\begin{aligned} y_t &= z_t\sqrt{h_t}, \\ \log h_t &= \omega + \beta \log h_{t-1} + \psi \log x_{t-1}, \\ \log x_t &= \xi + \varphi \log h_t + \tau(z_t) + u_t. \end{aligned} \tag{5}$$

Here y_t is the log return, x_t is the realized measure, $z_t = y_t/\sqrt{h_t} \sim i.i.d(0,1)$ and $u_t \sim i.i.d(0,\sigma_u)$. In the third equation we link the observed realized measure to the latent volatility. Almost all different GARCH specifications are nested within this Realized GARCH framework. A standard GARCH model can be constructed by setting $x_t = y_t$, this way the measurement equation is reduced to a simple identity and is therefore redundant. The function $\tau(z_t)$ is called the leverage function, it captures the dependence structure between the returns and the volatility. By changing this function we are able to capture the leverage effect

similar to an EGARCH specification. This leverage function takes on the form

$$\tau\left(z_t\right)=\eta_1 z_t+\eta_2\left(z_t^2-1\right), \tag{6}$$

where η_1 and η_2 are variables that will also be estimated. This specification is useful as it ensures that $E(\tau\left(z_t\right))=0$ as long as $E(z_t)=0$ and $var(z_t)=1$. Closely related to this leverage function is the so called news impact curve, see Engle and Ng (1993). This curve shows how volatility reacts to positive and negative shocks in the price. This curve is defined as

$$v(z)=E(\log h_{t+1}|z_t=z)-E(\log h_{t+1}). \tag{7}$$

$100\cdot v(z)$ measures the percentage impact on the volatility as a function of z. From Equation 5 we can derive that we can calculate the news impact curve by multiplying the leverage function with the parameter ψ. To estimate this model we will be using the R package rugarch Ghalanos (2022). Using the estimated model we will forecast the realized volatility of Bitcoin using a series of one-step ahead forecasts. Here we use the parameters that were estimated based on the training set, while using the real return and realized variance value of the previous days.

4.3 MS-GARCH

In the GJR-GARCH specification the effect an observation has on the volatility change according to the value of ε_{t-1}, as we have different behavior for $\varepsilon_{t-1}<0$. This can be seen as a regime switching model where the regime is determined by ε_{t-1}. In contrast to this one could assume that the regime is not determined by observable variable ε_{t-1} but by a hidden Markov process. To capture this Markov process we will be using a Markov-switching model

4.3.1 Specification

This brings us to a Markov-switching GJR-GARCH (MS-GARCH) model, that permits regime switching in the parameters. To also get a better grasp at how the leverage effect behaves over different regimes. The general specification for the MS-GARCH model, given it is in state s_t, is

$$\begin{aligned} y_t &= z_t\sqrt{h_t}, \\ h_t &= \omega_{s_t}+(\alpha_{s_t}+\gamma_{s_t}I[\varepsilon_{t-1}<0])\varepsilon_{t-1}^2+\beta_{s_t}h_{t-1}. \end{aligned} \tag{8}$$

where y_t is the log return,$z_t=y_t/\sqrt{h_t}\sim i.i.d(0,1)$ and $\varepsilon_t=z_t\sqrt{h_t}$. This model is subject to the parameter restrictions $\omega_{s_t}>0$, $\alpha_{s_t}\geq 0$, $\beta_{s_t}\geq 0$ for $s_t\in\{1,2,..,n\}$ in order to ensure that h_t is non-negative. z_t can follow different distributions and we denote shape parameters for a specific distribution with ν. We will be using a two-regime model with Students-t distribution for z_t, shape parameter ν will be estimated for both

regimes.

4.3.2 Estimation and forecasting

Estimation of the Markov-switching GARCH models can be done by either frequentist or Bayesian approaches. Both approaches require the evaluation of the likelihood function. In this paper we will be estimating the model using a Bayesian framework.

Let Θ be the vector containing all model parameters, and let θ_1 be the model parameters in regime 1, namely $(\omega_1, \alpha_1, \beta_1, \gamma_1)$. So let $\Theta = (\theta_1, \theta_2, \nu_1, \nu_2, \mathbf{P})$ be the vector of model parameters for a two regime Markov-switching GARCH model, where ν is the shape parameter of the conditional distribution of y_t, if applicable. $\mathbf{P}$ is the transition probability matrix. Then the likelihood function is

$$\mathcal{L}(\Theta|\mathcal{I}_T) = \prod_{t=i}^{T} f(y_t|\Theta, \mathcal{I}_{t-1}), \tag{9}$$

here $f(y_t|\Theta, \mathcal{I}_{t-1})$ is the density function of y_t given the previous observations, $\mathcal{I}_{t-1}$, and the model parameters. For the Markov-switching GARCH model with two regimes, this density function conditional on parameters Θ and past observations $\mathcal{I}_T$ is

$$f(y_t|\Theta, \mathcal{I}_{t-1}) = \sum_{i=1}^{2}\sum_{j=1}^{2} p_{i,j} m_{i,t-1} f(y_t|s_t = j, \Theta, \mathcal{I}_{t-1}). \tag{10}$$

The transition probability for going from state i to state j is given by $p_{i,j}$. $m_{i,t-1} = P(s_{t-1} = i|\Theta, \mathcal{I}_{t-1})$ is the filtered probability of being in state i at time $t-1$, which is obtained via the Hamilton filter (Hamilton, 1989). $f(y_t|s_t = j, \Theta, \mathcal{I}_{t-1})$ is the conditional density of y_t conditional on being in state j, the model parameters, and the previous observations. So we get the likelihood function

$$\mathcal{L}(\Theta|\mathcal{I}_T) = \prod_{t=1}^{T} \left[\sum_{i=1}^{2}\sum_{j=1}^{2} p_{i,j} m_{i,t-1} f(y_t; \nu_j|s_t = j, \Theta, \mathcal{I}_{t-1}) \right]. \tag{11}$$

This likelihood function is combined with prior $f(\Theta)$ to obtain the posterior kernel $f(\Theta|\mathcal{I}_T)$. For our priors we use the diffuse prior specifications as used in Ardia, Bluteau, Boudt, and Catania (2018). For a two

regime model, the priors are:

$$f(\Theta) \propto f(\theta_1, \nu_1) f(\theta_2, \nu_2) f(\mathrm{P}) \mathbb{1}\{h_{1,1} < h_{2,1}\}, \tag{12}$$

$$f(\theta_k, \nu_k) \propto f(\theta_k) f(\nu_k) \mathbb{1}\{(\theta_k, \nu_k) \in CSC_k\}, \qquad (k = 1, 2) \tag{13}$$

$$f(\theta_k) \propto f_{\mathcal{N}}(\theta_k; \mathbf{0}, 1000 \times \mathbf{I}) \mathbb{1}\{\theta_k > 0\}, \qquad (k = 1, 2) \tag{14}$$

$$f(\nu_k) \propto f_{\mathcal{N}}(\nu_k; \mathbf{0}, 1000 \times \mathbf{I}) \mathbb{1}\{\nu_{k,1} > 0, \nu_{k,2} > 2\}, \qquad (k = 1, 2) \tag{15}$$

$$f(\mathbf{P}) \propto \prod_{i=1}^{2} \prod_{j=1}^{2} p_{i,j} \mathbb{1}\{0 < p_{i,j} < 1\}. \tag{16}$$

Here we denote with $\mathbf{0}$ and $\mathbf{I}$ a vector of zeros and an identity matrix respectively. CSC_k is the covariance-stationarity condition for regime k as defined in Trottier and Ardia (2016). $f_{\mathcal{N}}(x; \mu, \mathbf{\Sigma})$ denotes a multivariate normal density with mean vector μ and covariance matrix $\mathbf{\Sigma}$, evaluated at value x. Finally, ν_k denotes the degrees of freedom of the Student-t distribution. These prior specifications are used to guarantee specific behavior of the estimators, like positivity of the GARCH parameters, as well as identification, by ordering regimes based on the variance of returns associated with that specific regime. The prior of transition matrix $\mathbf{P}$ assumes that the rows in this matrix are independent and follow a Dirichlet((1,1)) prior with $K = 2$.

In order to estimate the parameters of our models we will be using a Gibbs sampler (Geman & Geman, 1984) which consists of the following sampling steps:

1. $s^{(m)} \sim p(s|\theta_1^{(m-1)}, \theta_2^{(m-1)}, \nu^{(m-1)}, P^{(m-1)}, y)$,
2. $P^{(m)} \sim p(P|s^{(m)})$,
3. $\theta_1^{(m)} \sim p(\theta_1|\nu_1^{(m-1)}, s^{(m)}, y)$,
4. $\theta_2^{(m)} \sim p(\theta_2|\nu_2^{(m-1)}, s^{(m)}, y)$,
5. $\nu_1^{(m)} \sim p(\nu_1|\theta_1^{(m)}, s^{(m)}, y)$,
6. $\nu_2^{(m)} \sim p(\nu_2|\theta_2^{(m)}, s^{(m)}, y)$.

Of these different steps we can do the first two using established methods, we obtain the state probabilities using the Hamilton filter of Hamilton (1989). From the state probabilities we can estimate the state for each observation using the Viterby algorithm of Viterbi (1967). Combining the conjugate prior for each row of transition matrix $\mathbf{P}$: $P_j \sim Dir(1, 1)$ and the likelihood function, we will get the posterior $Dir(1+n_{i,1}, 1+n_{i,2})$ for row i. Here $n_{i,j}$ is the amount in the sample that we transition from state i to state j. Since the other posterior densities we obtain from combining these truncated normal priors with the likelihood function are from an unknown form, we will be using a different sampling techniques to generate their draws. We apply the adaptive random-walk Metropolis-Hastings sampler introduced by Vihola (2012) to sample from there posteriors. This Metropolis-Hastings algorithm consists of the following steps:

1. Compute $\theta^* \coloneqq \theta^{(m-1)} + S_{m-1}U_m$, where U_m is a draw from a spherically symmetric proposal density q. S_{m-1} is a non-singular shape matrix.

2. Accept the proposal draw with probability $\alpha_m \coloneqq \min\{1, \pi(\theta^*)/\pi(\theta^{(m-1)})\}$, where $\pi(\bullet)$ is a probability density function. Set

$$\theta^{(m)} \coloneqq \theta^* \qquad \text{if proposal draw is accepted,} \tag{17}$$

$$\theta^{(m)} \coloneqq \theta^{(m-1)} \qquad \text{if proposal draw is denied.} \tag{18}$$

3. Compute the lower diagonal matrix S_m with positive diagonal elements, that satisfies

$$S_m S_m^T = S_m \left(I + \eta_m(\alpha_m - \alpha_*) \frac{U_m U_m^T}{\|U_m\|^2} \right) S_{m-1}^T. \tag{19}$$

   Here $\{\eta_n\}_{n \geq 1} \subset (0, 1]$ is a series of step sizes decaying to 0, and α_* is the target mean acceptance rate of the algorithm.

This third step is where the algorithm of Vihola (2012) differs from a traditional Metropolis-Hastings or Adaptive Metropolis-Hastings algorithms. The reason one wants to have the matrix S be a random variable instead of a constant, is that if chosen right an evolving shape matrix S_m can reduce the amount of samples needed before we are sampling from the target density function. The basic idea behind this approach is that when the acceptance probability is lower than desired ($\alpha_m < \alpha_*$) the proposal distribution is shrunk in the direction of the current proposal, and vice versa. As S_m in Equation 19 is the Cholesky factor of the right hand side of the equation, it always exists. Vihola (2012) provides proof that this resulting matrix is symmetric and positive definite. This result will lower the amount of samples we need to reach convergence, which lowers the computation time needed to fully estimate the Markov-switching model. Since Bayesian models are known to be computationally expensive, every change to improve time to convergence is a useful one. To obtain the posterior results that follow from these methods to estimate the different parameters, we will be using the R package MSGARCH (Ardia, Bluteau, Boudt, et al., 2019).

Using these posterior results we will be forecasting the realized variance using a series of one-step ahead forecasts. Ideally we would be retraining the model in order to have it be estimated with the most recent available information, but we are not able to do that due to the computational expense. This also keeps it in line with the method of forecasting for the other models. For each one step ahead forecasts we calculate the prediction step, the probabilities for being in the two states based on the estimated parameters and past observations. We can then calculate the one-step ahead volatility forecast for each state separately. Multiplying these forecasts with the probabilities from the prediction step gives us the forecasted volatility. For the next one-step ahead forecast the predicted probabilities will be updated using the value of that current observation.

4.4 Markov-switching realized variance

4.4.1 Specification

The fourth model we use is a model that jointly models returns and a realized measure in a Markov-switching framework, a Markov-switching realized variance model. This model was introduced by Liu and Maheu (2018). As with the realized GARCH model of Section 4.2 we will be using realized variance as our realized measure in this model, but in further research it could be interesting to examine the effect of other ex-post volatility measures. The model specification from Liu and Maheu (2018) is

$$\begin{aligned} y_t|s_t &\sim N(\mu_{s_t}, \sigma^2_{s_t}), \\ RV_t|s_t &\sim IG(\omega+1, \omega\sigma^2_{s_t}), \\ p_{i,j} &= p(s_t = j|s_{t-1} = i). \end{aligned} \tag{20}$$

The main assumption and idea of this model is that both returns and realized variance are subjected to the same regimes, and that they are connected by the shared parameter $\sigma^2_{s_t}$. This seems a rather fair assumption to make, since realized variance is a function of intraday returns. As the conditional distribution of the realized variance, given the current regime, is an inverse gamma distribution with shape parameter $\omega + 1$ and scale parameter $\omega\sigma^2_{s_t}$, it has expected value

$$E(RV_t|s_t) = \frac{\omega\sigma^2_{s_t}}{(\omega+1)-1} = \sigma^2_{s_t}, \tag{21}$$

and variance

$$\text{Var}(RV_t|s_t) = \frac{(\omega\sigma^2_{s_t})^2}{(\omega^2)(\omega-1)} = \frac{(\sigma^2_{s_t})^2}{\omega-1}. \tag{22}$$

So RV_t is centered around the regime dependent variance, with the variance of its distribution being positively correlated with this regime dependent variance. So in high a variance regime we get higher expected realized variance as well as a larger variance of its distribution.

4.4.2 Estimation and forecasting

Like the MS-GARCH model of Section 4.3, the estimation of this joint Markov-switching model will be done using Bayesian inference. We will be using different MCMC methods in order to simulate from the conditional posterior distributions. For a two state Markov-switching model we have the parameters $\theta = \{\mu_1, \sigma_1, \mu_2, \sigma_2\}$, ω and transition matrix $\mathbf{P}$. We will use different priors for μ_1 and μ_2 to avoid labeling issues, as well as

restricting $\mu_1 < \mu_2$. The priors we use follow those used by Liu and Maheu (2018), and are

$$\mu_1 \sim N(-1, 1), \tag{23}$$

$$\mu_2 \sim N(1, 1), \tag{24}$$

$$P_j \sim Dir(1, 1), \qquad j = (1, 2) \tag{25}$$

$$\sigma_j^2 \sim G(\overline{RV}_T, 1), \qquad j = (1, 2) \tag{26}$$

$$\omega \sim IG(2, 1). \tag{27}$$

Here $\overline{RV}_T$ is the sample mean of the in-sample realized variance. Using these priors we iterate over the following MCMC steps in order to estimate the variables.

1. $s_{1:T}|y_{1:T}, RV_{1:T}, \theta$

   In the first step we estimate the state probabilities based on the return and realized variance data, as well as the current model parameters. In this step we use the forward filter backward sampler method of Chib (1996). This method contains the following steps:

   (a) $p(s_1 = j|y1, RV_1, \theta, \omega, \mathbf{P}) = \pi_j$ for $j = 1, 2$

   We initiate the distribution of s_1 by setting it equal to the left eigenvector that corresponds to the eigenvalue of 1.

   (b) $p(s_t|y_{1:t-1}, RV_{1:t-1}, \theta, \omega, \mathbf{P}) \propto \sum_{j=1}^{2} \mathbf{P}_{j,s_t} \cdot p(s_t = j|y_{1:t-1}, RV_{1:t-1}, \theta, \omega, \mathbf{P})$

   In this prediction step we multiply the current distribution of s_t with the transition matrix, in order to generate a prediction for the distribution of s_t at the next time period.

   (c) $p(s_t|y_{1:t}, RV_{1:t}, \theta, \omega, \mathbf{P}) \propto p(y_t, RV_t|s_t, \theta, \omega) \cdot p(s_t|y_{1:t-1}, RV_{1:t-1}, \theta, \omega, \mathbf{P}) \qquad \rightarrow$
   $p(s_t|y_{1:t}, RV_{1:t}, \theta, \omega, \mathbf{P}) \propto \phi(y_t; \mu_{s_t}, \sigma_{s_t}) \cdot IG(RV_t; \omega + 1, \omega\sigma_{s_t}^2) \cdot p(s_t|y_{1:t-1}, RV_{1:t-1}, \theta, \omega, \mathbf{P})$

   In this updating step we update the distribution of the prediction step by multiplying with the likelihood function, updating the prediction using the current observation. We get to the second equation by plugging in the likelihood function that corresponds to the specification in Equation 20. Here $\phi(\cdot)$ is the normal density valued at y_t with parameters μ_{s_t} and σ_{s_t}.

2. $\mu_j|y_{1:T}, RV_{1:T}, s_{1:T}, \sigma_j^2$

   For this second step we will be sampling μ_j from its conditional distribution, where we treat σ_j as a known variable. We do this for both regimes separately. Given the priors of Equations 23 and 24 the conditional posterior distributions of μ_1 and μ_2 are

$$\mu_1 \sim N\left(\frac{\sum_{s_t=1} y_t - 1 \cdot \sigma_1^2}{\sigma_1^2 + n_1},\ \frac{\sigma_1^2}{\sigma_1^2 + n_1}\right) \tag{28}$$

and

$$\mu_2 \sim N\left(\frac{\sum_{s_t=2} y_t + 1 \cdot \sigma_2^2}{\sigma_2^2 + n_2},\ \frac{\sigma_2^2}{\sigma_2^2 + n_2}\right) \tag{29}$$

respectively. Here n_j denotes the number of observations that we are in state j according to the series $S_T = s_1, s_2, \dots s_T$

3. $\sigma_j^2 | y_{1:T}, RV_{1:T}, \mu_j, s_{1:T}, \omega$

   To get the conditional posterior of σ_j^2 we multiply the likelihood function with the prior $\sigma_j \sim G(\overline{RV}_T, 1)$ and take all values that depend on σ_j^2. We get the following conditional posterior:

$$p(\sigma_j^2 | y_{1:T}, RV_{1:T}, \mu_j, s_{1:T}, \omega) \propto \sigma_j^{-n_j} \cdot (\omega\sigma_j^2)^{n_j \cdot (\omega+1)} \prod_{t=1}^{T} \left\{ \exp\left(\left(-\frac{(y_t - \mu_j)^2}{2\sigma_j} - \frac{\omega\sigma_j^2}{RV_t} \right) \cdot \mathbb{1}\{s_t = j\} \right) \right\} \tag{30}$$

$$\cdot (\sigma_j^2)^{\overline{RV}_T - 1} \exp(-\sigma_j^2).$$

   As this conditional posterior distribution is not of a known form we will be using a Metropolis-Hastings sampler to sample σ_j^2. The candidate density that we use is the likelihood function from the RV component combined with the prior,

$${}^{(c)}\sigma_j^2 \sim G\left(n_j(\omega + 1) + \overline{RV}_T,\ \omega \sum_{s_t=j} \frac{1}{RV_t} + 1 \right). \tag{31}$$

   In this equation $\sum_{s_t=j} \frac{1}{RV_t}$ denotes a summation of $\frac{1}{RV_t}$ over all periods t for which $s_t = j$.

4. $\omega | y_{1:T}, RV_{1:T}, s_{1:T}, \sigma_1^2, \sigma_2^2$

   With the prior $\omega \sim IG(2, 1)$ we obtain the following posterior:

$$p(\omega | y_{1:T}, RV_{1:T}, s_{1:T}, \sigma_1^2, \sigma_2^2) \propto \prod_{t=1}^{T} \left(\frac{(\omega\sigma_j^2)^{(\omega+1)}}{\Gamma(\omega + 1)} RV_t^{-v-2} \exp\left(-\frac{\omega\sigma_{s_t}^2}{RV_t} \right) \right) \cdot \omega^{-1} \exp\left(-\frac{1}{\omega} \right). \tag{32}$$

   To sample ω we use a Metropolis-Hastings sampler with random walk proposal, where the negative draws get rejected. This candidate density is a normal distribution centered around the current value of ω.

5. $\mathbf{P} | s_{1:T}$

   Combining the conjugate prior for each row of transition matrix $\mathbf{P} : P_j \sim Dir(1, 1)$ and the likelihood function, we will get the posterior $Dir(1 + n_{i,1}, 1 + n_{i,2})$ for row i. Here $n_{i,j}$ is the amount in the sample that we transition from state i to state j.

Generating forecasts with this model is similar to the way we forecast using the MS-GARCH. We perform

a series of one step ahead forecasts, where we first calculate the prediction for the probabilities for being in a specific state. We can then calculate the forecast density for RV by multiplying the state probabilities with the conditional distribution of RV, which is given in Equation 20. To generate point forecasts we draw samples from this distribution and take the mean of these draws.

4.5 LSTM-GARCH

The final type of model we are going to implement will be a LSTM (Long Short Term Memory) model that utilizes estimated variables from GARCH models. We will be implementing two different LSTMs, one using squared daily returns, and one utilitzing the realized variance.

Artificial neural networks are models developed in the discipline of computer science, inspired by the biological neural network in the brain. It constitutes of different layers of so called neurons which are connected to the other neurons. Each neuron gets a signal and processes it, and puts it through to the neurons that are connected to it. This output is in the form of a real number. To get an output of a layer and with that a neural network, a non-linear function is calculated with the outputs of the neurons. This output is then scored using a loss function. During the training process the score of this loss function is minimized by changing the weight of different neurons and edges between neurons.

4.5.1 Recurrent Neural Networks

Standard feedforward neural networks only allow signals to flow one way, from input to output. There is no feedback, so the signals of a certain layer are not able to affect that same layer. This is good to estimate short memory processes where there is not a lot of dependence between different observations. For other tasks like speech recognition and handwriting generation this does not work as there is temporal dependence between observations. This is also the case for volatility forecasting, as volatility seems to be a long memory process (Breidt, Crato, & De Lima, 1998).

In these cases it is needed to have some sort of feedback loop in the architecture of the neural network, such that past signals influence the current signal. This is the main idea behind recurrent neural networks (RNN), an idea that came up in research as early as Minsky and Papert (1969) and based on Rumelhart, Hinton, and Williams (1986). By introducing memory into the architecture of the network RNNs can process and learn sequences in the observations of input sequences. This makes it ideal to deal with sequence problems like time series. Instead of just mapping inputs to outputs, the network is capable of learning how the so called context affects the output, not just the input that has been presented. To train an RNN, backpropagation through time is often the method of choice. In this method one 'unrolls' a recurrent network and represents it as N duplicates of the original network, each representing the network at a different point in time, as illustrated in Figure 3.

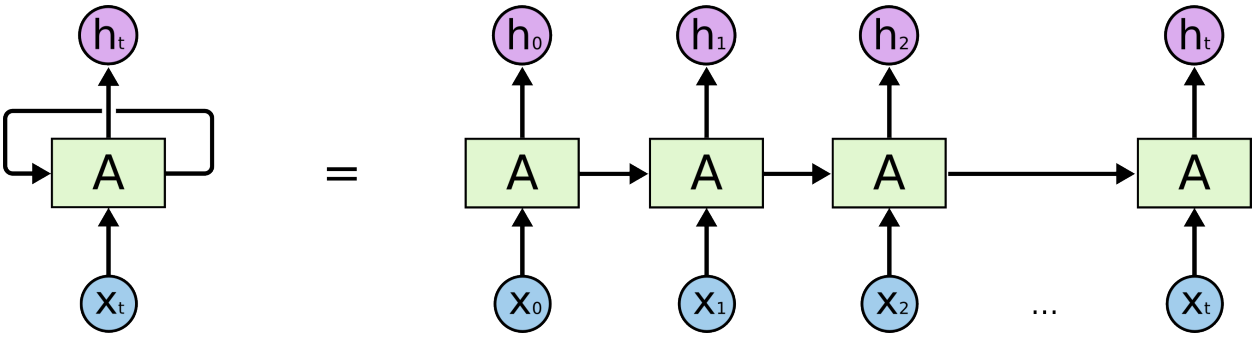

Figure 3: Unrolling a recurrent neural network. (Olah, 2015)

In the calculation of the gradient, which is necessary for backpropagation, the weights of every following layer are included, so for a normal feedforward network this would be $w_1 \cdot \alpha_1 \cdot \ \ldots \ \cdot w_n \cdot \alpha_n$. But as in the recurrent network all layers represent the same network at a different point in time, the weights are all the same, so we get $w^n \cdot (\alpha_1 \cdot \ \ldots \ \cdot \alpha_n)$. This often leads to an exploding or vanishing gradient for a large n. To overcome this challenge Hochreiter and Schmidhuber (1997) introduced Long Short Term Memory (LSTM), a special kind of RNN that is explicitly designed to learn long-term dependencies and deal with the exploding or vanishing gradient problem.

4.5.2 Long Short Term Memory

A usual LSTM unit consists of a cell state, an input gate, an output gate and a forget gate. The different gates are used to filter the amount of information. The key idea of LSTMs is that the cell state that keeps track of the current information, the upper horizontal line in Figure 4. As we can see it interacts with the gates to either add or subtract information, these gates are a way to optionally let information through. At each gate a sigmoid function is used that gives a vector of values between 0 and 1 based on the incoming data and learned weights. The result of this sigmoid function decides the amount of information that is let through. The workings of the memory cell that is at the core of the LSTM model is determined by the following equations:

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f), \tag{33}$$

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i), \tag{34}$$

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o), \tag{35}$$

$$\tilde{c}_t = tanh(W_c x_t + U_c h_{t-1} + b_c), \tag{36}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t, \tag{37}$$

$$h_t = o_t \odot tanh(c_t). \tag{38}$$

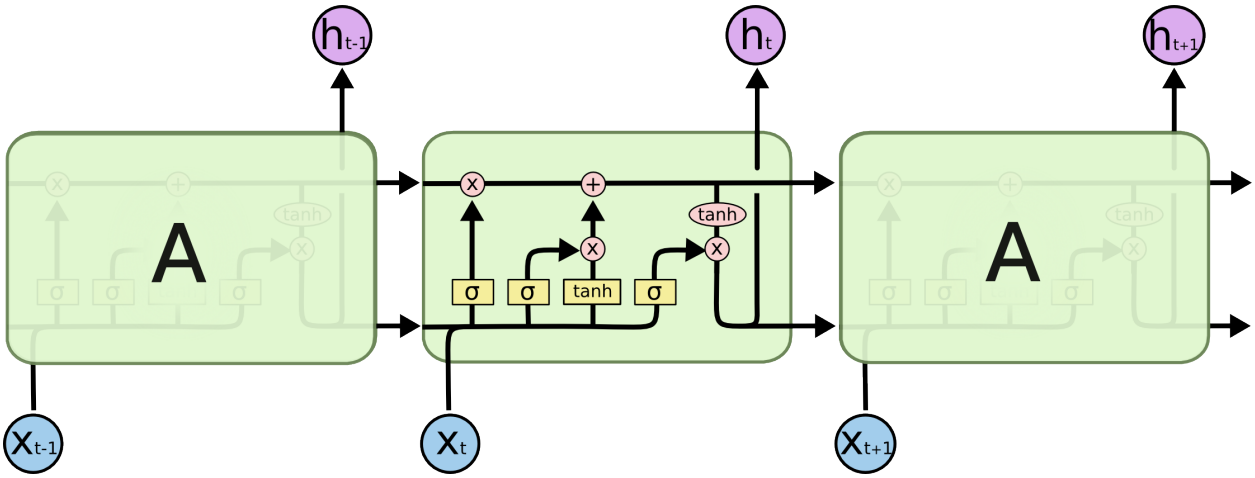

Figure 4: Layout of a memory cell in an LSTM network. (Olah, 2015)

Here σ denotes a sigmoid function, $\odot$ denotes element-wise multiplications. In these equations, $U_{\_}$ and $W_{\_}$ are weight matrices that are gate specific and are learned during the training of the model, $b_{\_}$ are bias vectors. The first three equations describe how the three different gates work. A linear combination of the input and the previous output go into the sigmoid function to return a vector of values between 0 and 1. Equation 33 describes the forget gate, denoted by the first upwards line in Figure 4. This gate determines the amount of information of the previous cell state is kept. The second gate is the input gate, that determines the amount of current information that is added to the cell state. Equation 37 shows how the forget and input gate interact with the previous cell state and the transformed input data (Equation 36) to form the current cell state. This current cell state is then used as an input in the next time step. To get an output for this time, the cell state goes through a hyperbolic tangent function to push the values between -1 and 1. It is then multiplied by the value of the output gate in Equation 38, which decides the amount of information that is actually in the output.

4.5.3 Our model

Instead of using this LSTM network as a standalone model we will be combining it with a GARCH model. Studies as Roh (2007), Kristjanpoller, Fadic, and Minutolo (2014) and Kristjanpoller and Hernández (2017) have integrated time series models and feedforward networks as a way to improve the models. By using estimates from a GARCH model as features for our LSTM, we hope to be able to capture some characteristics of those GARCH models. Characteristics like the leverage effect and volatility clustering. We will have two separate LSTM networks, with slightly different features. The first neural network will be using squared daily returns and estimated parameters of a GJR-GARCH model as unique features. The second model will use estimated parameters from the Realized GARCH model as well as the realized volatility as unique features. To get the parameter estimates for the two GARCH models we use a rolling window of 60 days. So for each observation we use the past 60 days in order to estimate the volatility model and get the parameter estimates.

Besides these model unique features we use several common features for both networks, these are:

- Difference between the daily high and low price,
- Daily closing price of BTC,
- Daily trading volume in BTC,
- Returns over the past 7 days.

Both models will be trained on realized volatility data with a root mean squared error (RMSE) loss function,

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\widehat{RV}_i - RV_i)^2}. \tag{39}$$

By using two different models we can see the impact that incorporating realized volatility has on the predictive performance versus using squared daily returns. We divide the test set into both a test set and a validation set, which of the last 200 days of the original test set. By doing this we can tune the hyperparameters of the two models based on the RMSE on the validation set. This reduces the probability that we overfit on the training set. So we train the model on the training set, get the optimal parameters using the validation set and test its forecasts on the test set. One of the hyperparameters that we will be tuning is the amount of LSTM layers in the model, either 2, 3 or 4 layers, consisting of 32-16, 64-32-16 or 128-64-32-16 neurons respectively. The other two are the the activation function and the dropout value. The models contain dropout layers that randomly sets input units to 0 with a frequency of the dropout rate at each training step. This form of regularization is a cheap and effective way to reduce overfitting.

4.6 Evaluating Results

To evaluate the performance of our different models we will be comparing the forecasts to the actual realized volatility with several loss functions, namely root mean squared error, and mean absolute error (MAE)

$$MAE = \frac{1}{N}\sum_{i=1}^{N}\left|\widehat{RV}_i - RV_i\right|, \tag{40}$$

as well as evaluating value at risk estimates and corresponding expected shortfall. While the results for the two loss functions are relevant and interesting most if you would be trading volatility or pricing derivatives, from a portfolio and risk management perspective the value at risk and expected shortfall are the more interesting metrics. To calculate the VaR estimates we will be using the Filtered Historical Simulation method proposed by Hull and White (1998) and Barone-Adesi, Bourgoin, and Giannopoulos (1998). In this method the historical returns are standardized using the estimated conditional mean and estimated volatility,

$\hat{z}_t = (y_t - \hat{\mu}_t)/\hat{\sigma}_t$. Using the series of standardized return we can then forecast the value at risk using the formula

$$VaR^{\alpha}_{T+1} = \hat{\mu} + \hat{\sigma}_{T+1}\text{Quantile}(\{\hat{z}_t\}^{T}_{t=1}, \alpha). \tag{41}$$

So for every day in our sample set we calculate the 5% value at risk by adding the average return of the model to the volatility forecasts times the 5% quantile of returns. We calculate this quantile based on the training data. This VaR evaluation based on the different volatility models gives us insight into whether the forecasted volatilities are either too conservative or not conservative enough. Preferably a model would have the amount of exceedances that corresponds to the used quantile, where an exceedance occurs when the daily return is lower than the value at risk. So we would like to see and exceedance ratio of around 5% for each of the models. Even though value at risk has become a standard metric in financial risk management, it has several downsides. First of all, the value at risk does not say anything about the possible size of loss that exceeds the value at risk level. This may lead to optimizing such that the value at risk stays acceptable, while the underlying potential losses can be extreme. Secondly, VaR is not subadditive, the VaR of a portfolio can exceed the sum of the VaR of its constituents. In our case this is not a relevant problem as we consider only 1 asset, but for broader research this is definitely important to keep track of. One metric that has neither of these liabilities, is the Expected Shortfall (ES). It says something about the losses exceeding the VaR and is subadditive. Expected shortfall is the average return conditional on the return exceeding the value at risk on the downside, and is calculated as

$$ES^{\alpha} = \frac{\sum_{t=T}^{T+N-1} y_t \cdot \mathbb{1}\{y_{t+1} > VaR^{\alpha}_{t+1}\}}{\sum_{t=T}^{T+N-1} \mathbb{1}\{y_{t+1} > VaR^{\alpha}_{t+1}\}}. \tag{42}$$

So it can be used to determine whether most of the exceedances occur at 'higher' low returns or that they mostly happen at the extremes. Ideally we would like to have an as low as possible expected shortfall, as this signals that when we exceed the value at risk, the loss is not that big on average. But for clear conclusions we need both the VaR and the expected shortfall, as they paint the most complete picture when combined.

5 Results

In this chapter we will discuss the results of the model estimations and their forecast. We start by taking a look at the estimated models before comparing the volatility forecasts for all the models in Section 5.2.

5.1 In-Sample results

5.1.1 GJR-GARCH

We estimated a GJR-GARCH(1,1) model as our benchmark, as it is simple and performs rather well. We use a t-distribution for the errors. Table 2 shows the results of the GJR-GARCH estimation. Something

that catches the eye is the negative estimate for γ. This signals there to be some sort of an inverted leverage effect. Although his effect is not significant at a 5% confidence level. So there is no significant leverage effect present in our sample. Therefore we can conclude that there is no significant difference in the way positive and negative returns affect the volatility.

Table 2: Parameter estimates for the GJR-GARCH model. Estimates significantly different from 0 at a 5% significance level are indicated by *.

	Estimates	Standard Errors
μ	0.002*	< 0.000
ω	0.000	< 0.000
α	0.127*	0.019
γ	-0.038	0.024
β	0.892*	0.021

5.1.2 Realized GARCH

In this Realized GARCH model we add the incorporation of a measurement equation that allows the incorporation of different realized measures. We use the realized variance that is calculated by summing the squared 10 minute returns over a day. In Table 3 we see the parameter estimates, where significant parameters are shows with an *. We can see that β has a higher estimated value than ψ. This indicates that the

Table 3: Parameter estimates for the Realized GARCH model. Estimates significantly different from 0 at a 5% significance level are indicated by *.

	Estimates	Standard Errors
ω	-1.550*	0.164
β	0.524*	0.028
ψ	0.424*	0.035
ξ	2.644*	0.469
φ	0.967*	0.072
σ_u	0.385*	0.007
η_1	-0.016	0.010
η_2	0.026*	0.002

lagged value h_t, so h_{t-1} has a larger impact on h_t than x_{t-1}, the realized variance in period $t-1$. Using the estimates for η_1, η_2 and ψ we can construct the News impact curve, this is shown in Figure 5. We can see that lower values for z_{t-1} leads to higher increases in volatility. Whether the skew in the effect of positive and negative returns on the volatility is positive or negative is determined by the estimate of η_1. From Table 3 we know that this estimate is not significantly different than zero, which in turn signals that there is no significant leverage effect present in our sample. A result that matches what we have seen with the GJR-GARCH model.

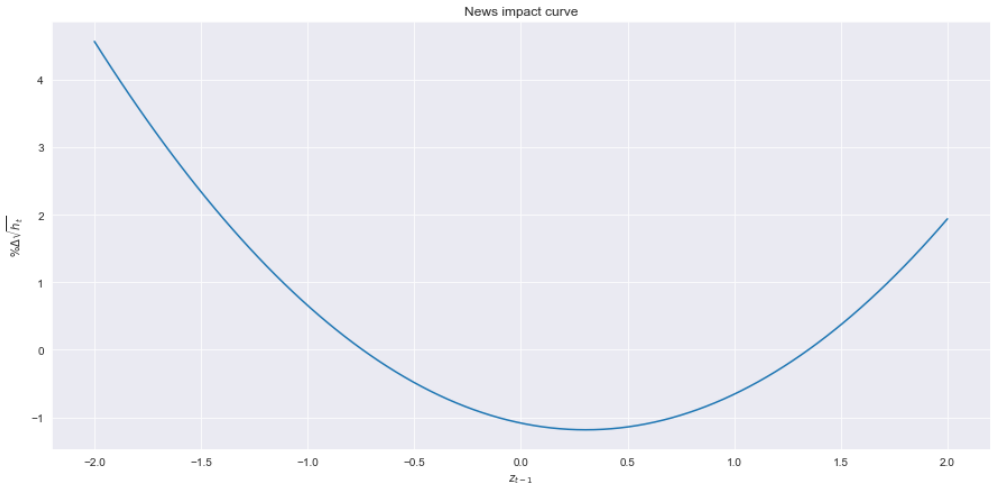

Figure 5: News impact curve, introduced by Engle and Ng (1993), for our Realized GARCH model showing the percentage change in $\sqrt{h_t}$ for different values of z_{t-1}

5.1.3 MS-GARCH

We have a 2 regime GJR-GARCH specification with t-distribution for the errors. To estimate the values of the parameters, we run 55,000 iterations with a burn-in of 5,000 and a thinning value of 10. Therefore we end up with a posterior sample of 5000 draws. In Table 4 we can see the posterior mean and standard deviation for our parameter estimates. The model recognizes a low and a high volatility regime. Regime

Table 4: Posterior means and standard deviation for the 2 regime GJR-MS-GARCH model. The GJR specification in regime s_t is $h_t = \omega_{s_t} + (\alpha_{s_t} + \gamma_{s_t} I[\varepsilon_{t-1} < 0])\varepsilon_{t-1}^2 + \beta_{s_t} h_{t-1}$. The estimates are obtained from the posterior sample of 1000 draws. The value in brackets is the posterior standard deviation.

Regime 1	Posterior mean	Regime 2	Posterior mean
ω_1	0.000	ω_2	0.002
	(0.000)		(0.001)
α_1	0.083	α_2	0.139
	(0.020)		(0.120)
γ_1	0.014	γ_2	0.286
	(0.016)		(0.232)
β_1	0.889	β_2	0.308
	(0.031)		(0.207)
ν_1	3.400	ν_2	50.678
	(1.880)		(27.528)
P_{11}	0.905	P_{22}	0.567
	(0.035)		(0.106)

1 is the low volatility regime, with an in-sample unconditional volatility of 25.37%. Regime 2 is the high volatility regime with an in-sample unconditional volatility of 151.77%. One interesting difference between the two different regimes is the value of γ_{s_t}. We see that in the second regime the reaction to negative returns is more than 10 times as strong as in regime 1. Besides the unconditional volatility and reaction to negative

returns, the volatility persistence in the two regimes is very different. In regime 1 it is $\alpha_1 + \frac{1}{2}\gamma_1 + \beta_1 = 0.979$ and for regime 2 it is $\alpha_2 + \frac{1}{2}\gamma_2 + \beta_2 = 0.59$. So the two different regimes are characterized by:

- **Regime 1**
- Low unconditional volatility,
- Weak reaction to negative returns,
- High volatility persistence.

- **Regime 2**
- High unconditional volatility,
- Strong reaction to negative returns,
- Low volatility persistence.

5.1.4 Markov-switching realized variance

For this Bayesian Markov-switching Realized variance model we implemented the model of Liu and Maheu (2018). We run the model for 30000 iterations, with a burn-in of 1000 iterations and a thinning value of 10. We end up with a posterior sample of 2900 draws. For the internal Metropolis-Hastings samplers we have 25,000 iterations with a burn-in of 5,000 values. By tweaking these values for the iterations we can generate posterior samples of high quality with relatively low amount of iterations. Table 5 shows the posterior results for the model. We can see that regime 1 corresponds with a low volatility regime with an unconditional volatility of $\sqrt{365} * 2.27 = 43.45\%$. The second regime is the high volatility regime, with an unconditional volatility of 113.91%. We see that for this high volatility regime the mean of the returns is much higher,

Table 5: Posterior means and standard deviation for the 2 regime Markov-switching realized variance model. In this model the returns and realized variance get jointly modeled. Given that we are in state s_t, the realized variance is then distributed as $RV_{s_t} = IG(\nu + 1, \sigma^2_{s_t})$. P_{ii} denotes the posterior transition probability from state i to state i. The amount of observations that are in state i is denoted with n_i. The value in brackets is the posterior standard deviation.

Regime 1	Posterior mean	Regime 2	Posterior mean
μ_1	0.192	μ_2	1.653
	(0.192)		(1.466)
σ_1^2	2.270	σ_2^2	5.963
	(0.039)		(0.189)
P_{11}	0.984	P_{22}	0.953
	(0.004)		(0.012)
n_1	1205.750	n_2	407.250
	(18.169)		(18.169)

almost 10 times as high as in the low volatility regime. Both of the regimes are very persistent with 98.4% and 95.3% average probability respectively to stay in the same state. This is different than the result for the MS-GARCH model where the high volatility regime was not very persistent.

This difference can clearly be seen in Figure 6, which shows the in-sample probabilities of being in the high volatility regime for both the Markov-switching GARCH model (MS-GARCH) and the Markov-switching realized variance model (MS-RV). While the MS-GARCH model mostly has spikes into high probabilities

for the second regime, the MS-RV model has prolonged periods where the probability of being in the high volatility regime is almost 1. Another interesting observation one can make from this figure is that the spikes for MS-GARCH during high levels of realized variance are as high as those in periods where the realized variance is rather low. This is the result of MS-GARCH using squared daily returns, which does not behave the same as realized variance. For example, a day of steady upwards price movement would have high squared daily return, while the realized variance is low.

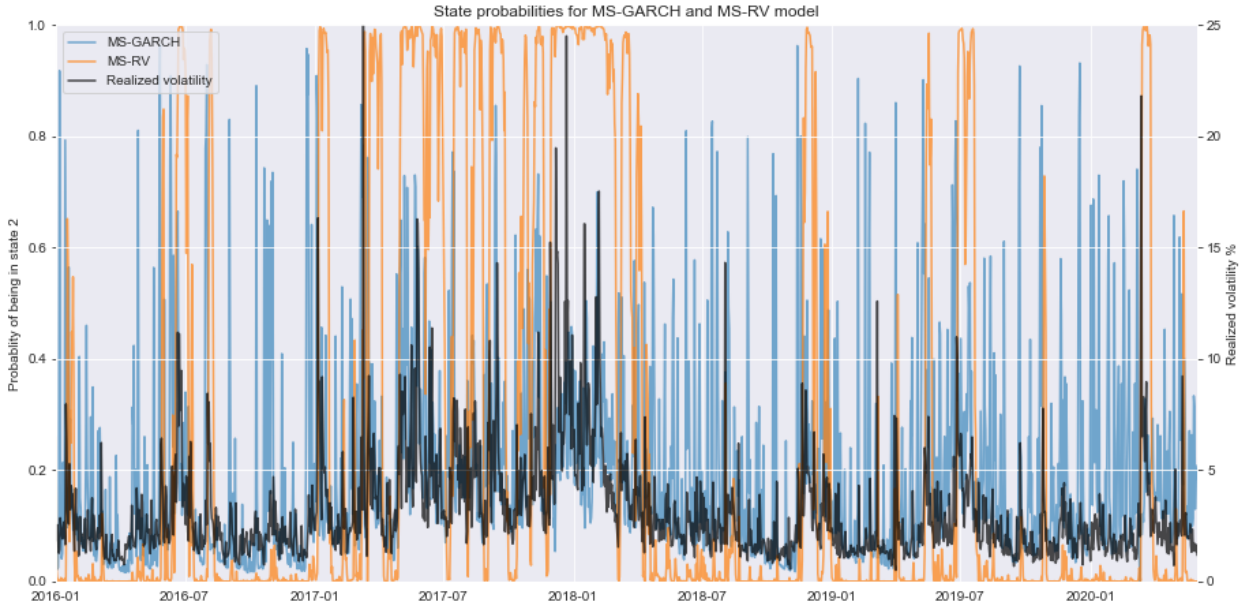

Figure 6: State probability to be in the high volatility regime for our two Markov-switching models in the test sample, along with the realized variance which is shown in black

5.1.5 LSTM with GARCH

As LSTM neural networks do not have interpretable parameter weights, we will be discuss at the results of the hyperparameter tuning. We tuned three different parameters:

- Number of LSTM layers,
- Dropout value,
- Activation function.

We found the optimal values for each model by doing a grid search over the different values and choosing the combination with the lowest RMSE on the validation set. For both models we got an optimal dropout value of 0.05, so at every training step an input has a 5% chance to be set to 0, this reduces overfitting. Both models also had 'sigmoid' as the best activation function. The difference between the two is in the amount of LSTM layers. For the model using squared daily returns 2 LSTM layers was optimal, one with

32 neurons and one with 16 neurons. For the other model another added layer of neurons was optimal, so it has 3 layers. One with 64 neurons, one with 32 neurons and one with 16 neurons.

5.2 Out-of-sample results

The real goal of this research is to forecast the volatility using the different volatility models, and determine whether the utilization of realized variance improves the forecasts of different types of models. Figure 8 shows the out-of-sample volatility forecasts for all six of our models, compared against the realized volatility. The left three figures show the results for the models utilizing squared daily returns, while the right three are from the models incorporating realized variance. The result that catches the eye is for LSTM with squared

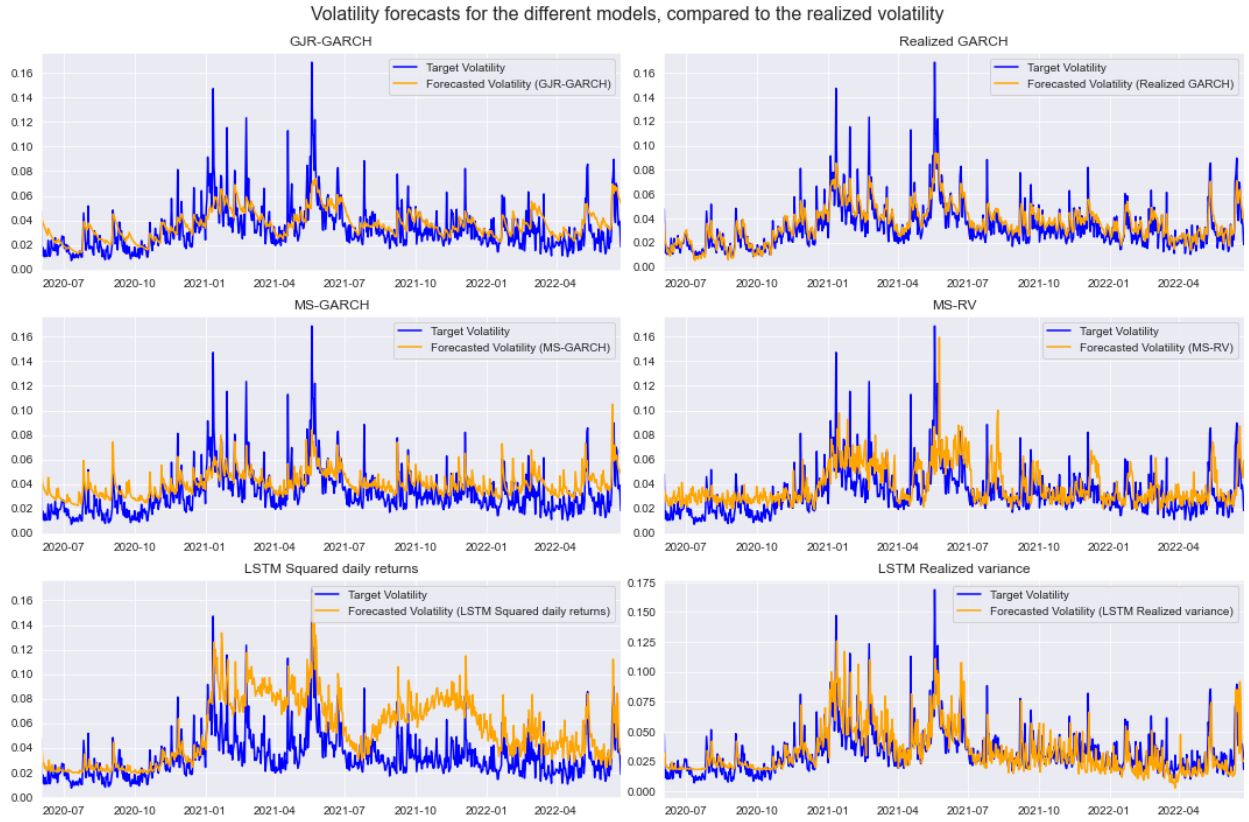

Figure 7: Out-of-sample volatility forecasts compared to the realized volatility, for our six different models

daily returns, as its out-of-sample forecasts are quite poor, especially compared to the other models. So our feature set combined with daily squared returns is not able to capture the dynamics of realized variance when using an LSTM model. The GJR-GARCH forecast follows the realized variance pretty well, but is not able to capture the larger spikes in volatility. This is something that is known for GARCH type models. For the Realized GARCH we can see a clear improvement over GJR-GARCH by adding the realized variance to the model. The forecast seems way more responsive, and is also better in capturing larger moves up or down in the volatility.

The MS-GARCH forecasts look quite accurate, but the model clearly has problems with forecasting periods of low volatility. Besides periods of high realized volatility with some spikes, the volatility estimates from this model are mostly higher than the realized volatility. For applications like VaR forecasting this is not as bad, as you would rather be too conservative than to underestimate the risk. But for an application like option pricing this would not be great, as you would consistently price options too high. MS-RV forecasts are the most volatile, and are also not that great at capturing the low volatility periods, this is especially clear at the start of our test sample. One reason for this can be deduced from Figure 6, as for MS-RV the state probabilities are rather binary. There are not a lot of observations where the probability of being in the high volatility regime is between 0.2 and 0.8. This has the effect that observations in low volatility periods all have the same expected value. So as the realized volatility changes quite a bit, the probability of being in the high volatility regime goes from 0 to 0.05 for example, which causes the forecast stay similar.

The biggest difference in forecasts between using squared daily returns and realized variance is in the LSTM models. Where adding using estimates from Realized GARCH and the realized variance itself show a clear improvement over using GJR-GARCH estimates and squared daily returns. The latter is not able to capture the dynamics of the realized variance for most of the test sample, with only some good results at the end. In Table 6 we can see the forecast valuations of the different models using four metrics, RMSE, MAE,

Table 6: Forecast evaluations for our different models, exceedance ratio is defined as the percentage of observations that exceeds the value at risk estimate

	RMSE	MAE	Exceedance ratio	Expected shortfall
GJR-GARCH	0.014	0.010	3.851%	-8.452%
Realized GARCH	0.009	0.007	4.914%	-7.319%
MS-GARCH	0.015	0.012	5.312%	-7.983%
MS-RV	0.015	0.013	5.445%	-7.858%
LSTM Squared daily returns	0.030	0.024	3.320%	-7.912%
LSTM Realized variance	0.016	0.011	9.429%	-5.917%

exceedance ratio and expected shortfall. We can see that indeed LSTM with squared daily returns is clearly the worst performer. Realized GARCH seems to be the best performer with both the lowest RMSE and MAE. GJR-GARCH also performs rather well for both metrics. To evaluate the value at risk forecasts we look at both the exceedance ratio and the expected shortfall. These metrics need to be both looked at to get a clear indication how the model performed, as the metrics on their own do not tell the whole story.

As we forecast the VaR at a 5% level we would expect around 5% of the observations to exceed the estimate. The exceedance ratio shows the percentage of observations that indeed exceeded the VaR estimate. We can see that both the GJR-GARCH and LSTM using squared daily returns provide too conservative VaR forecasts. This is expected for the LSTM model as for large parts of the test sample the volatility forecast is

way too high, which in turns causes the VaR estimate to be lower. Like said before, GARCH models perform quite good in volatility forecasts, except in forecasting spikes in volatility. So we would expect a exceedance ratio around 5%, while having high expected shortfall. Its exceedance ratio is a bit lower than 5% but the expected shortfall is indeed high. This signals that of the observations that exceed the VaR estimate, most of them are in high volatility periods. Compare this to the LSTM model where the expected shortfall is 50 percentage points lower, signaling that some of those exceedances happened at lower volatility periods.

The three models that are closest to the expected exceedance ratio are Realized GARCH, MS-GARCH and MS-RV. When comparing the expected shortfall of these models we notice that Realized GARCH has the lowest value, while also having the lowest exceedance ratio of the three. This shows that are objectively better, as the lower expected shortfall is not caused by a larger number of exceedances bringing down the average. This is confirmed by the values in Table 7, which shows the summary statistics of the observations where the return exceeds the VaR estimate. We an see that out of Realized GARCH, MS-GARCH and MS-RV the former has the bast values for the 25th, 75th and 100th percentiles. MS-RV slightly outperforms MS-GARCH in terms of the different percentile values of the exceedances, but overall their forecasting performance is rather similar in quality.

The highest value for expected shortfall in Table 6 is for the LSTM model utilizing realized variance. This model has also the worst exceedance ratio of all, almost twice as high as expected. These two results go somewhat hand in hand, since a higher exceedance ratio often lowers the expected shortfall, as more exceedances happen at not that negative returns. This idea is somewhat confirmed in Table 7, as we can see that the 75th percentile for this LSTM model is higher than the max for 2 models. This shows that a lot of the exceedances for this model happen for smaller negative returns. This is not a great result, but one would rather have a model that underperforms at forecasting VaR in lower volatility environment than one that does not perform in high volatility regimes. To get an even better idea how the exceedances are distributed for the different models, Appendix A shows the densities of the exceedances.

Table 7: Summary statistics of the value at risk exceedances for the different models. These exceedances are defined as the daily returns that are lower than the forecasted value at risk. If exactly 5% of the observations exceed the VaR estimates, it would correspond to a count of 37.65.

	Count	Mean	Min	25%	75%	Max
GJR-GARCH	29	-8.452	-15.964	-10.481	-5.811	-4.352
Realized GARCH	37	-7.319	-15.964	-9.205	-5.089	-2.523
MS-GARCH	40	-7.983	-15.964	-9.870	-5.724	-4.352
MS-RV	41	-7.858	-15.964	-9.744	-5.591	-3.422
LSTM Squared daily returns	25	-7.912	-15.964	-10.481	-5.535	-3.321
LSTM Realized variance	71	-5.917	-15.964	-6.932	-3.750	-2.058

6 Conclusion

In this paper we introduced six different approaches to modeling the realized volatility of Bitcoin. We can divide those models in different groups, first of all we have the divide between the three models that use squared daily returns as the volatility proxy and the three models that incorporate realized variance. Over those two groups of models we have used three model types. We started off with two GARCH type models, GJR-GARCH and realized GARCH. These are the two simplest models and serve as a benchmark for the other models. Secondly we used two Markov-switching models that were estimated using a Bayesian framework, which allows for model parameters to switch based on the regime that we are in. Finally we used two long short-term memory neural networks, a type of recurrent neural network that is well suited to model time series data. We want to find what the effects of incorporating realized variance into different model types are on the quality of volatility forecasts.

Using four different metrics we look at the forecasting ability of the different models. We look at both RMSE and MAE to determine the quality of the forecast, while looking at the value at risk and expected shortfall to determine how these forecasts would do in a risk management setting. We find that the realized GARCH model has the lowest forecast error as measured by both RMSE and MAE, while the LSTM model with squared daily returns give the worst forecasts by quite a large margin. We find that incorporating realized variance into the different model types improves the volatility forecasts for the standard GARCH models and the LSTM models. The two Markov-switching models perform very similarly with the same value for the RMSE and a small difference in MAE in favor of MS-GARCH. Out of the three model types, the standard GARCH models perform the best in terms of RMSE and MAE, with the LSTM models performing the worst. Comparing the VaR estimates and the corresponding expected shortfall is more difficult as the results of those two metrics need to be looked at collectively. We find that realized GARCH once again performs the best overall, with an exceedance ratio close to the expected value and a low expected shortfall. For these two metrics the models incorporating realized variance outperform their opposite number for all three model types. We thus conclude that utilizing realized variance over squared daily returns can improve volatility forecasting ability of different model types. While we tested four models that were much more complex and computationally expensive, none of them were able to outperform the realized GARCH model.

There are multiple limitations to our research. First is the availability of data, Bitcoin has existed for just 13 years, and we deem most of that price history not very usable as it was a market with very low liquidity and due to that big price movements. This has improve in the last 7 years, but it leaves us with less data than desired. We would also have preferred to use 5 minute data to construct the realized variance as research has shown that that seems to be the optimal time frame, unfortunately we did not have that data available. Secondly, concerning our methodology is in the prediction for our Bayesian Markov-switching

models. Estimation of these models is quite computationally expensive. To improve the forecasting ability of these models we would like to re-estimate our models with every new point in our out-of-sample dataset, in this way we would be able to incorporate all new data into our parameter estimates as it becomes available. Unfortunately this was not feasible for this paper. It would be interesting how these models would perform if they were re-estimated and if that would significantly improve their forecasting ability. Another improvement could be made in the LSTM models. We chose to use a relatively simple feature set as we wanted the focus to be more on the model than on the feature engineering. If one would incorporate more potentially interesting and useful features, the model performance would almost certainly improve. One can think of other realized measures like the realized quarticity, but also of Bitcoin specific features like certain on-chain metrics, data that is derived from blockchain data.

References

Andersen, T. G., Bollerslev, T., Diebold, F. X., & Ebens, H. (2001). The distribution of realized stock return volatility. *Journal of Financial Economics*, *61*(1), 43–76.

Andersen, T. G., Bollerslev, T., Diebold, F. X., & Labys, P. (2003). Modeling and forecasting realized volatility. *Econometrica*, *71*(2), 579–625.

Ardia, D., Bluteau, K., Boudt, K., & Catania, L. (2018). Forecasting risk with Markov-switching GARCH models: A large-scale performance study. *International Journal of Forecasting*, *34*(4), 733–747.

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

Ardia, D., Bluteau, K., & Rüede, M. (2019). Regime changes in Bitcoin GARCH volatility dynamics. *Finance Research Letters*, *29*, 266–271.

Barndorff-Nielsen, O. E., Hansen, P. R., Lunde, A., & Shephard, N. (2008). Designing realized kernels to measure the ex-post variation of equity prices in the presence of noise. *Econometrica*, *76*(6), 1481–1536.

Barndorff-Nielsen, O. E., & Shephard, N. (2002). Econometric analysis of realized volatility and its use in estimating stochastic volatility models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, *64*(2), 253–280.

Barone-Adesi, G., Bourgoin, F., & Giannopoulos, K. (1998). Don't look back. *Risk*, *11*(8), 100–103.

Bollerslev, T. (1986). Generalized autoregressive conditional heteroskedasticity. *Journal of Econometrics*, *31*(3), 307–327.

Breidt, F. J., Crato, N., & De Lima, P. (1998). The detection and estimation of long memory in stochastic volatility. *Journal of Econometrics*, *83*(1-2), 325–348.

Buterin, V. (2013). A Next-Generation Smart Contract and Decentralized Application Platform. *White paper*.

Chib, S. (1996). Calculating posterior distributions and modal estimates in Markov mixture models. *Journal of Econometrics*, *75*(1), 79–97.

Chu, J., Chan, S., Nadarajah, S., & Osterrieder, J. (2017). GARCH modelling of cryptocurrencies. *Journal of Risk and Financial Management*, *10*(4), 17.

Corsi, F. (2009). A simple approximate long-memory model of realized volatility. *Journal of Financial Econometrics*, *7*(2), 174–196.

Dyhrberg, A. H. (2016). Hedging capabilities of bitcoin. Is it the virtual gold? *Finance Research Letters*, *16*, 139–144.

Engle, R. F. (1982). Autoregressive conditional heteroscedasticity with estimates of the variance of United Kingdom inflation. *Econometrica: Journal of the Econometric Society*, 987–1007.

Engle, R. F., & Bollerslev, T. (1986). Modelling the persistence of conditional variances. *Econometric reviews*, *5*(1), 1–50.

Engle, R. F., & Ng, V. K. (1993). Measuring and testing the impact of news on volatility. *The Journal of Finance*, *48*(5), 1749–1778.

Geman, S., & Geman, D. (1984). Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*(6), 721–741.

Ghalanos, A. (2022). Rugarch: Univariate GARCH models [Computer software manual]. (R package version 1.4-7.)

Glosten, L. R., Jagannathan, R., & Runkle, D. E. (1993). On the relation between the expected value and the volatility of the nominal excess return on stocks. *The Journal of Finance*, *48*(5), 1779–1801.

Hamilton, J. D. (1989). A new approach to the economic analysis of nonstationary time series and the business cycle. *Econometrica: Journal of the Econometric Society*, 357–384.

Hansen, P. R., Huang, Z., & Shek, H. H. (2012). Realized GARCH: a joint model for returns and realized measures of volatility. *Journal of Applied Econometrics*, *27*(6), 877–906.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, *9*(8), 1735–1780.

Hull, J., & White, A. (1998). Incorporating volatility updating into the historical simulation method for value-at-risk. *Journal of Risk*, *1*(1), 5–19.

Kristjanpoller, W., Fadic, A., & Minutolo, M. C. (2014). Volatility forecast using hybrid neural network models. *Expert Systems with Applications*, *41*(5), 2437–2442.

Kristjanpoller, W., & Hernández, E. (2017). Volatility of main metals forecasted by a hybrid ANN-GARCH model with regressors. *Expert Systems with Applications*, *84*, 290–300.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, *25*.

Liu, J., & Maheu, J. M. (2018). Improving Markov switching models using realized variance. *Journal of Applied Econometrics*, *33*(3), 297–318.

Liu, L., Patton, A. J., & Sheppard, K. (2015). Does anything beat 5-minute RV? A comparison of realized measures across multiple asset classes. *Journal of Econometrics*, *187*(1), 293–311.

Minsky, M., & Papert, S. (1969). *Perceptrons*. MIT press.

Nakamoto, S. (2008). Bitcoin: A peer-to-peer electronic cash system. *Decentralized Business Review*, 21260.

Nelson, D. B. (1991). Conditional heteroskedasticity in asset returns: A new approach. *Econometrica: Journal of the Econometric Society*, 347–370.

Roh, T. H. (2007). Forecasting the volatility of stock price index. *Expert Systems with Applications*, *33*(4), 916–922.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, *323*(6088), 533–536.

Sirignano, J., & Cont, R. (2019). Universal features of price formation in financial markets: Perspectives from deep learning. *Quantitative Finance*, *19*(9), 1449–1459.

Tan, C.-Y., Koh, Y.-B., Ng, K.-H., & Ng, K.-H. (2021). Dynamic volatility modelling of Bitcoin using

time-varying transition probability Markov-switching GARCH model. *The North American Journal of Economics and Finance*, *56*, 101377.

Trottier, D.-A., & Ardia, D. (2016). Moments of standardized Fernandez–Steel skewed distributions: Applications to the estimation of GARCH-type models. *Finance Research Letters*, *18*, 311–316.

Vidal, A., & Kristjanpoller, W. (2020). Gold volatility prediction using a CNN-LSTM approach. *Expert Systems with Applications*, *157*, 113481.

Vihola, M. (2012). Robust adaptive Metropolis algorithm with coerced acceptance rate. *Statistics and Computing*, *22*(5), 997–1008.

Viterbi, A. (1967). Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE transactions on Information Theory*, *13*(2), 260–269.

Xiong, R., Nichols, E. P., & Shen, Y. (2015). Deep learning stock volatility with Google domestic trends. *arXiv preprint arXiv:1512.04916*.

Zakoian, J.-M. (1994). Threshold heteroskedastic models. *Journal of Economic Dynamics and Control*, *18*(5), 931–955.

A Exceedance Densities

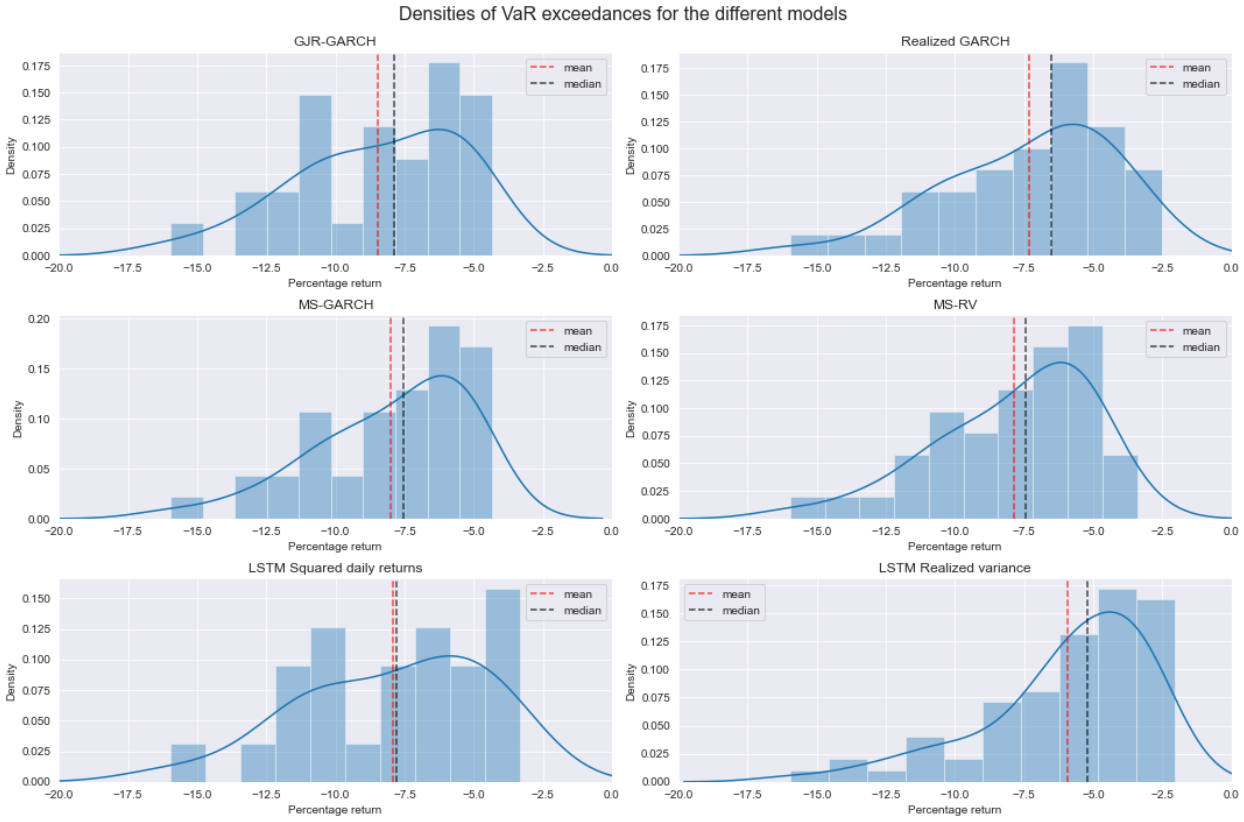

Figure 8: Densities of the returns that exceeded the value at risk observation, for the different models.