University of Tartu
Faculty of Social Sciences
School of Economics and Business Administration

Predicting stock return and volatility with machine learning and econometric models: A comparative case study of the Baltic stock market

Anders Nõu, Darya Lapitskaya,
M. Hakan Eratalay, Rajesh Sharma

Tartu, 2021

ISSN-L 1406-5967
ISSN 1736-8995
ISBN 978-9985-4-1286-2 (pdf)
The University of Tartu FEBA
https://majandus.ut.ee/en/research/workingpapers

Predicting stock return and volatility with machine learning and econometric models — a comparative case study of the Baltic stock market

Anders Nõu[1], Darya Lapitskaya[2], M. Hakan Eratalay[3], Rajesh Sharma[4]

Abstract

For stock market predictions, the essence of the problem is usually predicting the magnitude and direction of the stock price movement as accurately as possible. There are different approaches (e.g., econometrics and machine learning) for predicting stock returns. However, it is non-trivial to find an approach which works the best. In this paper, we make a thorough analysis of the predictive accuracy of different machine learning and econometric approaches for predicting the returns and volatilities on the OMX Baltic Benchmark price index, which is a relatively less researched stock market. Our results show that the machine learning methods, namely the support vector regression and k-nearest neighbours, predict the returns better than autoregressive moving average models for most of the metrics, while for the other approaches, the results were not conclusive. Our analysis also highlighted that training and testing sample size plays an important role on the outcome of machine learning approaches.

Keywords: machine learning, neural networks, autoregressive moving average, generalized autoregressive conditional heteroskedasticity

JEL classification numbers: C23, C45, C51, C52, C53, C58

[1]Institute of Computer Science, University of Tartu

[2]Corresponding author, darya.lapitskaya@ut.ee, School of Economics and Business Administration, University of Tartu

[3]School of Economics and Business Administration, University of Tartu

[4]Institute of Computer Science, University of Tartu

We gratefully acknowledge funding from GrowInPro (Horizon 2020), H2020 Project, SoBigData++, and CHIST-ERA Project, SAI. We are very grateful for the suggestions of Luca Alfieri.

1 Introduction

The active development of the stock markets in the Baltic region started after the Soviet Union collapsed. The first stock exchange that opened in the Baltic region was the Vilnius stock exchange in 1993 (Nikkinen et al., 2012), followed by the Riga stock exchange in the same year (Nikkinen et al., 2012). Two years later, the Tallinn stock exchange was opened in 1995 (Nikkinen et al., 2012). Currently, these three stock exchanges are part of the joint Baltic market that was established to reduce trade barriers between the Baltic countries, and at the moment, they operate under the NASDAQ Baltic index. As of 22 April 2021, the main list of the NASDAQ Baltic includes a total of 32 companies, and the secondary list contains 28 companies[5]. Interestingly, the Baltic stock exchange has recently demonstrated large growth: while in 2019, the total turnover of the stock exchange was approximately 260 million euros, in 2020, the total turnover was approximately 444 million euros[6].

The research is motivated by the fact that recently, in Baltic countries, interest in investing in the stock market has grown due to the popularisation of investment and the favourable trading conditions of banks (Suimets, 2020); however, while other stock exchanges use different techniques, such as automated trading, quantitative analysis, and various econometric models[7,8], upon the preliminary research on current methods used in the Baltic stock market, we did not find the application of automated trading transactions or machine learning models. Moreover, unlike larger markets, there are very few attempts to apply machine learning and neural network models in the studies of the Baltic region.

This paper aims to provide researchers and investors with a comparison of various machine learning and econometric approaches for time series analysis and for predicting stock returns and volatility in the Baltic stock market. The motivation for this research lies in the fact that the Baltic stock market has been studied very little in terms of the application of machine learning and financial models. For instance, examples of related works include the work by Grigoryan et al. (2015) who used the artificial neural network model to predict the daily stock price of the Tallink Group AS (TAL1T); and the study by Ercan (2017) who used the artificial neural network model to predict the OMX Baltic Benchmark GI (OMXBBGI) index value.

In this research, we use machine learning (random forest, KNN, SVR), econometric models (ARMA, GARCH), and a hybrid model of artificial neural networks and an econometric model (GARCH-ANN) for predicting the Baltic stock returns and volatilities. The contribution of this paper to the literature is of empirical nature. We tune each model to get its best predictive performance.

[5]https://nasdaqbaltic.com/statistics/en/shares, retrieved on 22.04.2021

[6]https://nasdaqbaltic.com/statistics/en/statistics, retrieved on 22.04.2021

[7]https://www.cnbc.com/2019/06/28/80percent-of-the-stock-market-is-now-on-autopilot.html, retrieved on 26.02.2021

[8]https://www.mordorintelligence.com/industry-reports/algorithmic-trading-market, retrieved on 30.01.2021

After obtaining the results from the best configurations, we make comparisons across the models for training and testing different sample sizes and using various metrics. In this study we focus on the Baltic market and use the OMX Baltic Benchmark Price Index daily data from 04.09.2001 to 01.03.2021. Our results indicate that for predicting returns, machine learning models, in particular support vector regression and k-nearest neighbour methods, performed better; and, for volatility predictions, the performances of GARCH and GARCH-ANN were comparable, depending on the favoured evaluation metric. From our exhaustive analysis, we can infer that a model's predictive performance (calculated using various evaluation metrics) depends on the choice of the predictors, and on the split size between the training and testing samples. In addition, for identifying the best model it is important not to trust default parameters as in some cases, we found that after tuning the parameters, the results got better (compared to the default parameter settings).

The paper is organised as follows: the first section briefly describes the problem, the goal, and the motivation behind the problem. The second section provides an overview of related works and explains the background of the study. The third section gives an overview of the data used. The fourth section is dedicated to describing the methodology; that is, the programming of machine learning and econometric models. The fifth section summarises the results of the different models and compares the machine learning and econometric models. And, the sixth section concludes the results of the study and provides recommendations for further improvements.

2 Background

In this section, we discuss different approaches that researchers apply for predicting stock returns and for financial analysis, namely, econometric (Section 2.1), machine learning (Section 2.2), and hybrid ones (Section 2.3).

2.1 Econometric models

The researchers extensively use various econometric models to predict the stock price and volatility. For instance, autoregressive–moving-average (ARMA) and generalised autoregressive conditional heteroscedasticity (GARCH) models are actively used in numerous works (Herwartz, 2017; Hu et al., 2020; Oberholzer & Venter, 2015; Rounaghi & Zadeh, 2016). In particular, in the research by Rounaghi and Zadeh (2016), the authors create an ARMA model to forecast the yearly and monthly stock returns of the S&P 500 Index and the London Stock Exchange and propose that this model can be used to predict medium or long horizons for the specified stock exchanges.

In addition, different variations of GARCH models are used in the literature (Herwartz, 2017). For example, Oberholzer and Venter (2015) apply the GARCH model to the South African stock exchange (JSE) and use the Akaike and Schwarz information criteria to evaluate its performance.

Moreover, both ARMA and GARCH models can be used to analyse fitting and prediction effects. For instance, in the research by Hu et al. (2020), the authors show that the GARCH model performs better in the fitting effect, and the ARMA is better for the prediction effect. Interestingly, in another work, the combination of ARMA and GARCH has also been used to predict Dow and S&P 500 Indices, and the results show that such models are effective for financial analysis (Wang et al., 2009).

2.2 Machine Learning based models

In addition to econometric models, in recent years, machine learning techniques have been actively used for financial analysis and forecast (Khan et al., 2020). The main reason behind this trend is that usually, machine learning usage overcomes limitations of traditional econometric models (Rossi, 2018). In addition, we can observe that some machine learning algorithms demonstrate high accuracy and high predictive powers compared to traditional econometric models when used for stock returns predictions (Lapitskaya et al., 2021). In this approach, various learning methods are used by researchers. For example, the support vector machine is a widely applied machine learning algorithm in time series forecasts (Ballings et al., 2015; Choudhry & Garg, 2008; Huang et al., 2005; Kara et al., 2011; Patel et al., 2015), and it provides higher accuracy through cross-validation (Li et al., 2018). Furthermore, logistic regression and random forest models are extensively used in this research area (Ballings et al., 2015; Long et al., 2019; Patel et al., 2015).

In general, the problem has been analysed from classification and regression perspectives. In particular, from the classification side, random forest and support vector machine models are used in several works to predict the direction of the stock price of Samsung, Apple, General Electric, and the Tokyo Stock Exchange index Nikkei 225 (Huang et al., 2005; Khaidem et al., 2016). Furthermore, such machine learning models as the kernel factory, AdaBoost, K-nearest neighbours, logistic regression, random forest, and support vector machine are used to predict the direction of stock prices on the European markets (Choudhry & Garg, 2008; Huang et al., 2005; Kara et al., 2011; Khaidem et al., 2016; Long et al., 2019; Nelson et al., 2017; Patel et al., 2015), where random forest and support vector machine models demonstrate the most accurate results (Ballings et al., 2015). At the same time, hybrid models can also be used in such cases. Interestingly, a hybrid model using a genetic algorithm and support vector machine performs better than the standalone support vector machine model (Choudhry & Garg, 2008).

At the same time, other works model a prediction problem as a regression one. For example, in the study by Trafalis and Ince (2000), the researchers employ support vector regression (SVR) for predicting the prices of IBM, Yahoo, and America Online. And in a different work, the combinations of machine learning algorithms and neural networks were used for the analysis of the Bombay Stock exchange (Patel et al., 2015). There, the authors concluded that the two-stage hybrid models

perform better than the single-stage prediction models.

2.3 Neural networks

Though many studies have implemented numerous machine learning models, some researchers have also used artificial neural networks and their variations to analyse similar problems (Chen et al., 2015; Dash & Dash, 2016; Kara et al., 2011; Long et al., 2019; Nelson et al., 2017). In general, artificial neural networks (ANNs) are among the most useful methods of volatility and price forecasting (Wanjawa & Muchemi, 2015). They present a non-parametric model that facilitates time series modelling (Yeze & Yiying, 2019). These models have an advantage in modelling complex non-linear relationships, and they can relate a set of input variables to one or more output target variables that contain non-linear latent units to achieve significant flexibility (Kim & Enke, 2016).

Researchers use ANNs with the regression problem of stock price or return predictions. For example, Jang and Lee (2017) use Bayesian neural networks to predict Bitcoin prices, and in the study by Ticknor (2013), a Bayesian regularised artificial neural network is used to predict the one-day future stock prices of Microsoft and Goldman Sachs. Moreover, different variations of neural networks are also actively used. For example, the long short-term memory (LSTM) neural network, multi-filters neural network (MFNN), computational efficient functional link artificial neural network (CEFLANN), and ANNs have been used on numerous stock exchanges (Chen et al., 2015; Dash & Dash, 2016; Kara et al., 2011; Long et al., 2019; Nelson et al., 2017).

Interestingly, the combination of the GARCH model and neural networks can also be applied in some cases. For instance, some examples prove that a hybrid GARCH-ANN can perform better than traditional GARCH models in terms of volatility forecasts (Liu & So, 2020). So, from the literature, we can see that although neural networks are used more widely and might generally perform better than machine learning models, there is still less research on the Baltic stock market using such methodologies.

Different methods can be used for evaluating the model's performance. For instance, Rounaghi and Zadeh (2016) use mean absolute error, mean absolute percentage error, median absolute percentage error, symmetric median absolute percentage error and the mean absolute scaled error to evaluate econometric models. In addition, Shah (2007) uses mean absolute error, root mean squared error, relative absolute error, and root relative squared error for evaluating machine learning models.

3 Data and Methods

This section is dedicated to the description of the dataset and the research methodology, where we introduce the data (Subsection 3.1) and the econometric (Subsection 3.2), machine learning

(Subsection 3.3) and hybrid models (Subsection 3.4) for the prediction tasks.

3.1 Data

The OMX Baltic Benchmark Price Index (OMXBBPI) is chosen for this study because it contains the Baltic market's largest and most traded companies. One of the sources that offers information about the Baltic stock market is the website investing.com [9], from where the daily price data is obtained for this research due to the number of entries and the available columns for calculating volatility. The time range of the data in consideration is from 4 September 2001 to 1 March 2021. This data range allows us to consider nearly 20 years, which covers many different financial and economic circumstances, including the 2008 financial crisis and the Covid-19 period. Based on our results, we can conclude that the predictive ability of the models in this paper is robust to these changes in the data.

In this work, the stock returns are predicted by both the econometric models (ARMA) and machine learning models. Next, we describe various equations and notations which are used in this work.

1. Equation 1 shows the return calculation for an observed period, where P_t is the daily adjusted closing price of the stock market index.

$$r_t = (P_t - P_{t-1}) / P_{t-1} \approx log(P_t) - log(P_{t-1}) \tag{1}$$

2. Volatility is measured as either the standard deviation or variance between returns from the same security or market index (Daly, 2008). There are several ways to estimate the volatility of a stock or index. One option to find the volatility of a security is through a volatility index for a specific market. For example, the VIX index tracks the volatility based on S&P 500 index options. However, an index that tracks the volatility of the Nasdaq Baltic market does not exist. Another option is to calculate the realised volatility of the index, which is the sum of squared returns for a period. Calculating the daily realised volatility requires intraday data, which is not available for the OMX Baltic Benchmark Price Index. Therefore, an alternative approach is needed. There are several alternative equations to calculate the estimated volatility. One of them is the Garman and Klass approach (Lebedeva, 2018) that is shown in Equation 2, where $H_{it}, L_{it}, O_{it}, C_{it}$ are high, low, open and close values for a security respectively. In this Equation, *i* represents the series, and *t* represents time.

[9] https://www.investing.com/

$$\begin{aligned}\widetilde{\sigma}_{it}^2 &= 0.511(\log H_{it} - \log L_{it})^2 \\ &- 0.019[(\log C_{it} - \log O_{it})(\log H_{it} + \log L_{it} - 2\log O_{it}) \\ &- 2(\log H_{it} - \log O_{it})(\log L_{it} - \log O_{it})] - 0.383(\log C_{it} - \log O_{it})^2\end{aligned} \tag{2}$$

3. In this paper, we use the return variable for the model estimation. First, we consider z number of previous returns, where z is the number of previous days of the data record. Then, we consider a minimum of 1 and a maximum of 30 for the value of z. For machine learning, several additional features are used for fitting the models. These are the high, low, close and open values. The high and low values are the highest and lowest prices of a specific period respectively; and, the open value is the price at the start of the trading day, while the close value is the price at the end of the trading day.

3.2 Econometric models

3.2.1 ARMA

The autoregressive moving average (ARMA) model, first introduced by Whittle (1951), is widely used in statistical analysis of time series. While constructing the ARMA model, the optimal p and q lags have to be chosen, which can be done with several methods. One way to do that is to calculate the autocorrelation and partial autocorrelations for the time series at hand. The respective plots for partial autocorrelation and autocorrelation lags are shown in figures 1 and 2, respectively.

For partial autocorrelation and autocorrelation, there is a sharp drop after one lag. Consequently, the most optimal orders of p and q are ones according to the plots. However, in both plots, a considerable autocorrelation can be observed for lags between 2 and 14, with some exceptions. In both cases, the correlation drops off, with some exceptions of small spikes for some lags. After the optimal orders of p and q are determined, the best performing model is compared with the best performing machine learning models. Furthermore, the outperforming ARMA model is applied for the sliding window method.

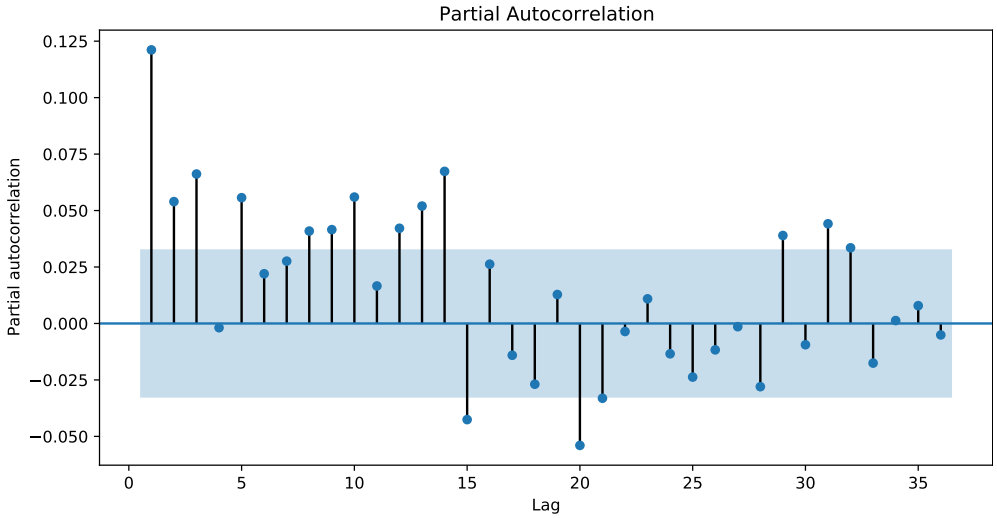

Figure 1. Partial autocorrelation of OMXBBPI time series

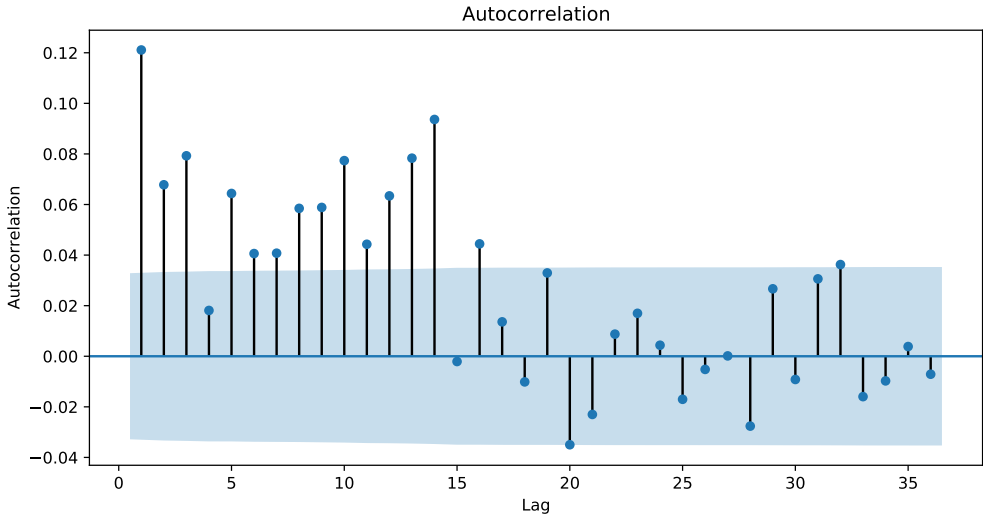

Figure 2. Autocorrelation of OMXBBPI time series

3.2.2 GARCH

The generalised autoregressive conditionally heteroscedastic (GARCH) model is based on the autoregressive conditional heteroscedasticity (ARCH) model generalised by Bollerslev (1986) by including past conditional variances.

For the GARCH model, similar tests as used on the ARMA model (described in Section 3.2.1) are carried out to have a fair comparison. The GARCH model with the most optimal (p,q) orders is also used for the GARCH-ANN hybrid model. Later on, the GARCH model is compared with the GARCH-ANN model.

3.3 Machine learning models

This study uses three different machine learning methods: the random forest, support vector machine (SVM), and K-nearest neighbours (KNN).

3.3.1 Random forest

Random forest is a supervised learning algorithm proposed by Breiman (2001). The algorithm consists of creating several trees that each cast a vote for the most popular class. Essentially, the random forest consists of several decision trees that use randomly selected inputs.

3.3.2 Support vector regression

Support vector regression (SVR) is a supervised learning model based on the Vapnik-Chervonenkis theory (Awad & Khanna, 2015). SVR is used for the regression problem; meanwhile, SVM is the analogue version of the SVR technique and is used for classification problems. The general idea of the SVM is to find a hyperplane in a multidimensional space that separates the classification targets (Noble, 2006).

3.3.3 K-nearest neighbours

KNN is an algorithm that predicts the target value based on the k nearest neighbours of the observed features (García-Laencina et al., 2009; Guo et al., 2003). The prediction is decided on a majority vote, which is similar to the random forest decision tree forecasting explained in Section 3.3.1. The KNN method can be applied for classification and regression problems (Ban et al., 2013; García-Laencina et al., 2009). The KNN is described as a simple and intuitive but low-performance learning method (Ban et al., 2013; Guo et al., 2003). The effectiveness of the model is also heavily dependent on an optimal value for k.

3.4 GARCH-ANN

GARCH-ANN is a hybrid model composed of the GARCH model (described in Section 3.2.2) and an artificial neural network (ANN) model. Generally, the hybrid model is created in the following way: first, the GARCH model is created and fitted, and the predictions of the previously constructed GARCH model are used to fit the ANN model. Then, the fitted ANN model is used to predict the target value, and the target is then evaluated with the actual value.

As for the specific ANN implementation model, first, we test several ANN structures. The model's structure optimisation consists of modifying the number of layers and the dimensionality of

the layer's output space [10]. The dropout layer, which assists in avoiding overfitting, is also included in some cases [11]. The activation layer functions and optimisers are cross-tested to find the best combination of the layer activation functions and the optimiser for the ANN model. Finally, the best performing ANN structure, layer activation function and optimiser combination are used for the sliding window method.

4 Evaluation

In this section, we discuss various evaluation metrics and then present the results.

4.1 Experimental setup

It is important to note that traditionally, machine learning estimation involves shuffling the data and splitting the dataset to train and test sets. However, when dealing with a time series problem, the dataset cannot be split because the order of observations is essential. Therefore, to perform cross-validation methods when dealing with a time series problem, the sliding window method is used (Yu et al., 2014). The sliding window technique means choosing the training and test sets that have constant size throughout the cross-validation process. After each iteration, the test data is added to the training data, and a chunk that has the size of the test data is removed from the start of the training set. Furthermore, the sliding window technique shows whether it is more beneficial to use either the whole dataset or a smaller amount of data to forecast future values. Hence, the sliding window method is carried out for all the models: ARMA, machine learning, GARCH and GARCH-ANN. In this study, the training window sizes are 1000, 500, 365, 300 or 100 and the test window size is 5 or 10 to be able to work with short-term forecasting. Nakajima (2017) uses a similar approach to construct sliding window samples with a training window size of 10 and a test window size of 5.

As for choosing the training and testing test sizes, a 70% / 30% split is used for all the machine learning approaches, since we found that this split size is the most optimal. For example, the most optimal GARCH(p,q) model is compared with the best performing GARCH-ANN model with a 70% / 30% split and then compared in the sliding window method. After the fitting is done, the models are used to predict the returns. Later, the predictions are compared with the actual values, and the prediction errors are calculated.

The overall testing and optimisation procedure consists of three parts. First, the optimal features are chosen for each model out of the two sets of features that are tested:

- z number of previous returns;

[10] https://www.tensorflow.org/api$_d ocs/python/tf/keras/layers/Dense, retrievedon$24.04.2021

[11] https://www.tensorflow.org/api$_d ocs/python/tf/keras/layers/Dropout, retrievedon$24.01.2021

- z number of previous returns, and high, low, close and open values.

Afterwards, the best performing feature set is chosen for each model, and parameter optimisation is carried out to maximise the effectiveness of the models. In addition, the sliding window method is implemented for each model to measure the effects of adjusting the size of the training size and the range of the prediction size. In the next step the machine learning methods are compared to determine the model with the best predictive performance in the standard method and the sliding window method. Finally, the best-performing machine learning model is compared with the ARMA model.

4.2 Metrics

In this paper, we use six evaluation metrics: mean absolute error, mean absolute percentage error, symmetric mean absolute percentage error, mean squared error, root mean square error and the histograms of standardized residuals. In the equations presented below, y_t is the prediction value, x_t is the actual value, and n denotes the forecast horizon. The description of the metrics are also presented below.

1) Mean absolute error (MAE) is a statistical metric used to measure a model's performance (Chai & Draxler, 2014). MAE is also often compared with root mean square error and is considered to be a better metric (Chai & Draxler, 2014; Willmott & Matsuura, 2005). MAE is calculated using the following Equation 3 (Chai & Draxler, 2014):

$$MAE = \frac{1}{n}\sum_{t=1}^{n}|y_t - x_t| \tag{3}$$

2) Mean absolute percentage error (MAPE) offers an interpretation of the relative error, which helps to evaluate the results better. Equation 4 shows the calculation of MAPE (Kim & Kim, 2016):

$$MAPE = \frac{1}{n}\sum_{t=1}^{n}\left|\frac{y_t - x_t}{y_t}\right| \tag{4}$$

3) Symmetric mean absolute percentage error (sMAPE) is also used as one of the metrics. A variation of the original equation introduced by Armstrong (1985) is used in this paper as shown in Equation 5:

$$sMAPE = 1/n\sum_{t=1}^{n}\frac{|y_t - x_t|}{(|y_t| + |x_t|)/2} \tag{5}$$

The bounds of sMAPE are [0, 200%] and the maximum value of the range is 200% because of the division by two in the denominator.

4) Mean squared error (MSE) presents the quantitative level of error or similarity between two values (Wang & Bovik, 2009). The range of MSE is [0, ∞] and Equation 6 shows the calculation of MSE (Wang & Bovik, 2009):

$$MSE = \frac{1}{N}\sum_{t=1}^{n}(y_t - x_t)^2 \tag{6}$$

5) Root mean square error (RMSE) represents the square root of the MSE metric. The scale of the RMSE is [0, ∞] and is presented in the same unit of measurement as the observed values of x and y in the equation. Equation 7 below demonstrates the calculation of RMSE:

$$RMSE = \sqrt{\frac{1}{N}\sum_{t=1}^{n}(y_t - x_t)^2} \tag{7}$$

6) Standardised residuals (SR) show the strength of the difference between the observed value and the actual value (Everitt & Skrondal, n.d.). The goal of using standardised residuals is to compare the overall predictive performance of using the econometric and machine learning models. The closer to zero the standardised residuals, the more effective a model. In this paper, the absolute values of standardised residuals are taken. Then, the mean absolute value is calculated to compare the strength of the chosen models.

Here, we calculate two sets of standardised residuals. We use Equation 8 to calculate the standardised residuals for econometric models.

$$SR1 = \frac{\text{ARMA forecast error}}{\sqrt{\text{GARCH volatility forecast}}} \tag{8}$$

Equation 9 shows how to calculate the standardised residuals for machine learning models.

$$SR2 = \frac{\text{Machine learning forecast error}}{\sqrt{\text{GARCH-ANN volatility forecast}}} \tag{9}$$

Standardised residuals are calculated for the standard method and the sliding window method for all models. Regarding the sliding window method parameters, a training size of 1000 and a test size of 5 are chosen. Also, the mean absolute values of standardised residuals are calculated to compare different approaches and methods.

4.3 Model tuning

In this subsection, we give a brief overview of our approach to interpreting the results. First, we interpret the results of the ARMA model. Here, we choose the optimal model and then apply the sliding window method to determine appropriate training and test sizes. After that, we decide on the optimal set of features for machine learning models. Then, we optimise the models' parameters

and compare them with the default parameter models. Furthermore, the sliding window method is applied to each machine learning model. Based on that, the best performing machine learning method is determined. Later, the ARMA and machine learning models are compared in the standard and sliding window methods to determine which approach is better in predicting the OMX Baltic Benchmark Price Index return.

For the GARCH model, first, different orders of p and q are considered to determine the optimal lags. Second, the GARCH model is used for the sliding window method with the optimal p and q determined from the first step. Finally, the optimal training and test sizes are selected from the sliding window test. In the case of GARCH-ANN, the best performing GARCH model forecasts are used as the neural network model inputs to determine the optimal neural network structure, optimiser, layer activation function, and epoch size. Then, the optimal neural network structure and parameters are determined to construct a GARCH-ANN model for the sliding window method. Finally, we choose the best training and test sizes. After determining the best performing GARCH and GARCH-ANN models, the models are compared in several tests performed on the complete dataset and using the sliding window method. Then, based on the comparisons, we determine which model is better at predicting volatility.

4.4 ARMA

For the ARMA model, first, we need to find the optimal orders of p and q for AR(p) and MA(q). To do that, we perform 25 tests with different sets of p and q. The values considered for p and q are from 1 to 5. Table 22 (see Appendix) gives an overview of these tests. Although the autocorrelation and partial autocorrelation plots indicated that the best model is ARMA(1,1); based on the AIC value, we observe that ARMA(3,3) is the best one. Nevertheless, neither ARMA(1,1) nor ARMA(3,3) outperform other ARMA models based on the five metrics described earlier. On the contrary, with regards to sMAPE, ARMA(4,5) outperforms the others. Although, ARMA(2,1) outperforms the others in regards to the MAPE metric. In addition, based on the MAE results, ARMA(4,4) is the best one, while ARMA(3,2) dominates other ARMA models based on the MSE and RMSE metrics. The best performing models among different metrics and the models stated to be the best performers by the correlation plots and standard method are displayed in Table 1.

Table 1. The best performing ARMA models and the models stated to be best

p	q	MAE	MAPE	sMAPE	MSE	RMSE
1	1	0.003876	126.187%	167.512%	0.00005198	0.0072
2	1	0.003871	124.312%	168.777%	0.00005161	0.0072
3	2	0.003866	125.766%	165.329%	0.00005112	0.0071
3	3	0.003879	137.051%	161.123%	0.00005211	0.0072
4	4	0.003848	128.991%	162.887%	0.00005128	0.0072
4	5	0.003905	167.689%	155.925%	0.0000549	0.0074

Based on the MAPE metric, ARMA(2,1) outperforms the other models. Therefore, ARMA(2,1) is chosen for the sliding window method. When considering MAPE, ARMA(2,1) is the most effective with the sliding window method when the training size is 1000 and the test size is 5 (see Table 2), where MAPE is 122.683%, and sMAPE is 171.973% respectively. Furthermore, when the training size is 1000, the MSE values are similar to the MSE of the standard method in Table 1. In conclusion, we can say that the method's predictive performance is more highly based on the metrics than on the evaluated methods.

Table 2. ARMA(2,1) sliding window results

Training size	Test size	MAE	MAPE	sMAPE	MSE	RMSE
1000	5	0.004323	122.683%	171.973%	0.000052	0.005268
500	5	0.005277	132.907%	170.189%	0.000077	0.006416
365	5	0.005718	131.661%	165.430%	0.000101	0.006930
300	5	0.005699	144.047%	162.609%	0.000097	0.006887
100	5	0.005774	156.280%	159.968%	0.000096	0.006991
1000	10	0.004326	119.845%	172.934%	0.000052	0.005523
500	10	0.005292	130.805%	171.219%	0.000078	0.006741
365	10	0.005672	126.463%	166.578%	0.000094	0.007195
300	10	0.005753	145.816%	164.120%	0.000099	0.007241
100	10	0.005817	157.229%	160.082%	0.000098	0.007348

4.5 Machine learning

For different machine learning models, we use identical data sets and features to build a comparison between them. First, we evaluate the optimal features for each machine learning method and choose the best set of features. Then, for each method, the models with default parameters and optimised

parameters are compared. Afterwards, we choose the best set of model parameters to create a sliding window method. In this study, the best training and test sizes are determined for each model.

4.5.1 Random forest

Random forest's (RF) first model optimisation involves the testing of the features. Initially, we test the model with z number of previous returns, where the minimum value of z is one, and the maximum is thirty. Table 23 (see Appendix) gives an overview of the number of previous returns in the feature testing. As the number of previous returns as features grow, the MAE, MAPE, MSE and RMSE decrease in general. On the other hand, as the number of previous returns increase, the sMAPE value increases. The results for MAE, MAPE, sMAPE and MSE can be confirmed visually from figures 3 - 6. An immense difference in MAPE can be seen (see Table 23 in Appendix) between the results of one previous return and thirty previous returns – 469.225% and 151.448% respectively.

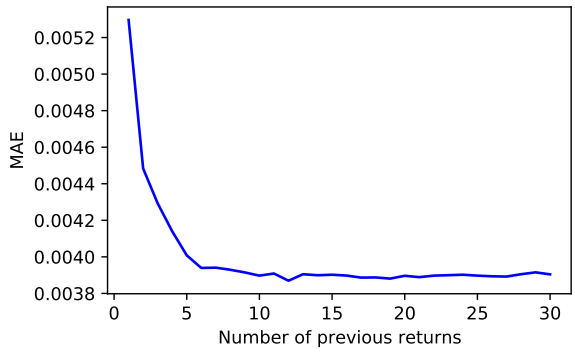

Figure 3. RF number of previous returns MAE results

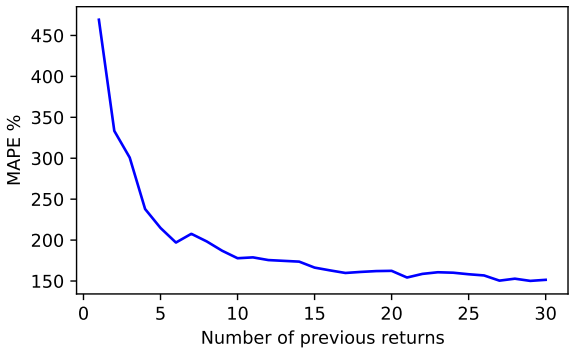

Figure 4. RF number of previous returns MAPE results

Figure 5. RF number of previous returns sMAPE results

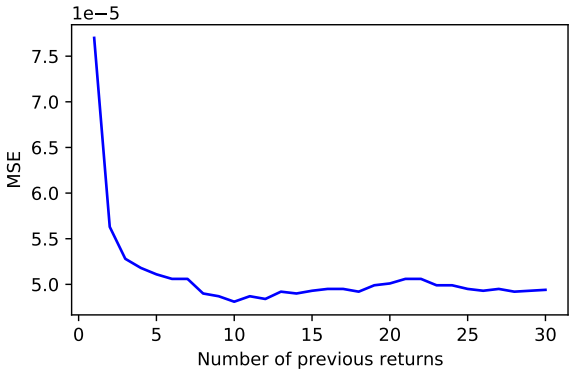

Figure 6. RF number of previous returns MSE results

The second set of model feature tests also includes the high, low, open and close values and the z number of previous returns. Table 24 (see Appendix) gives a numerical overview of the results.

For MAE, MAPE and sMAPE measures, similar results are obtained as seen in figures 7, 8 and 9. However, unlike in the previous set of tests, MSE has a spike between 15 and 25 returns as seen in figure 10.

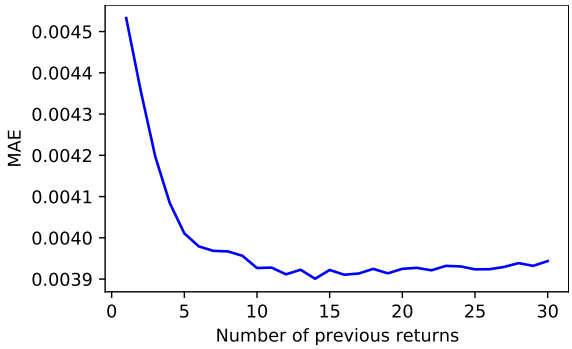

Figure 7. RF all features MAE results

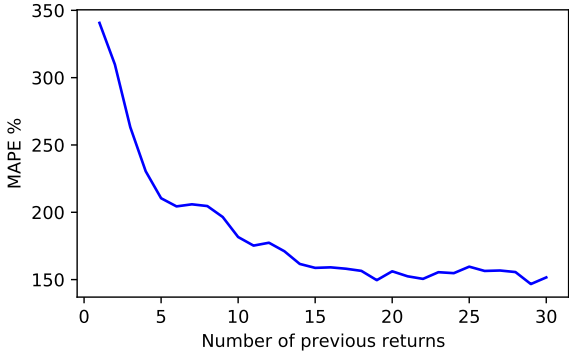

Figure 8. RF all features MAPE results

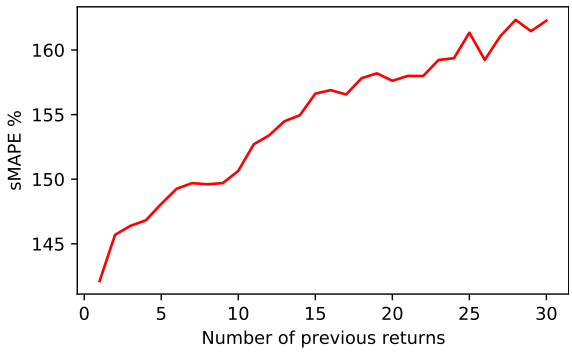

Figure 9. RF all features sMAPE results

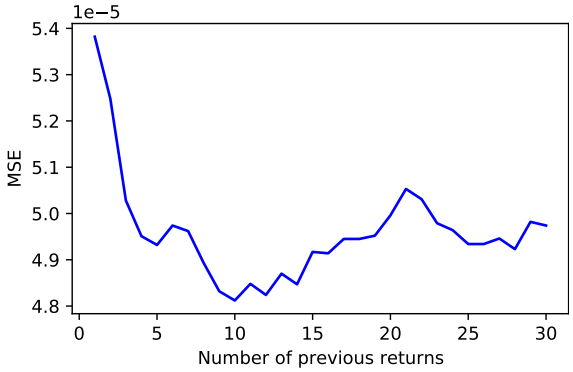

Figure 10. RF all features MSE results

To conclude the feature testing, we can see that the number of previous returns has an extensive impact on the results. The values for the MAE, MAPE, and MSE metrics decrease substantially as the number of previous returns grows. On the contrary, the value for sMAPE increases as the number of previous returns rises. Therefore, if the interest is in the MAE, MAPE or MSE values, it is more reasonable to choose a high number of previous returns. If the sMAPE is considered more substantial, it is more feasible to favour a low number of previous returns. Furthermore, if a low number of previous returns is preferred, it is better to include the extra features. Simultaneously, if a high number of previous returns is chosen, then the high, low, open and close values do not appear to be meaningful as the results are similar. For further RF model optimisation and tests, both options of one previous return and thirty previous returns are studied. In addition, further optimisation includes the optimising the model's parameters.

Tables 3 and 4 show the difference when using default parameters and optimised parameters for the RF model. With one previous return, there is a big increase in the values of the MAE and

Table 3. Random forest results with default parameters

z prev. returns	MAE	MAPE	sMAPE	MSE	RMSE
1	0.004532	340.681%	142.120%	0.000053	0.0073
30	0.003943	151.616%	162.270%	0.000049	0.0071

Table 4. Random forest results with optimised parameters

z prev. returns	MAE	MAPE	sMAPE	MSE	RMSE
1	0.003888	114.516%	170.743%	0.000054	0.0073
30	0.003872	109.421%	182.709%	0.000051	0.0071

MAPE metrics when parameters are optimised, while MSE and RMSE values are fairly similar. However, we can see that the model's effectiveness decreased in the sMAPE metrics. With thirty previous returns, the outcome is similar to one previous return, except that MAE and MAPE do not shrink as much. In conclusion, the optimisation technique does not favour the sMAPE metric, but it is beneficial in terms of the MAE and MAPE metrics.

The sliding window method is performed with two different sets of features – one previous return, high, low, close, and open values, and thirty previous returns, high, low, close, and open values. The overview of the results can be seen in tables 5 and 6. With both sets of features, the metric value patterns are quite identical. A large training data set is beneficial for the MAE, MAPE, MSE and RMSE metrics in both cases. Here, we can observe that the values increase as the training size increases; however, for sMAPE, a small size is favourable.

The results of the sliding window method align with the results of the feature testing and parameter optimisation testing (see Tables 23 and 24 in Appendix). Thirty previous returns as features yield better outcomes than one previous return with MAE, MAPE, MSE, RMSE, and one previous return yields better results with sMAPE.

4.5.2 Support vector regression

For SVR models, the same set of tests is carried out as for the RF models. Tables 25 and 26 (see Appendix) give an overview of the results of the feature testing with SVR default parameters. As seen in figures 12, 14,and 16 and figures 24 and 26 (see Appendix), as the number of previous returns increases gradually, the metric values increase. Therefore, the optimal number of returns for SVR with all the features is one; and for the feature set including only the previous returns, the metric values start out relatively small and then spike between two and five previous returns. However, after the spike, the errors decrease and are somewhat stable until thirty previous returns.

Table 5. Random forest sliding window results with 1 previous return, high, low, close and open values as features

Training size	Test size	MAE	MAPE	sMAPE	MSE	RMSE
1000	5	0.0042292	136.707%	158.989%	0.0000503	0.0054251
500	5	0.0051478	171.140%	147.707%	0.0000797	0.0063650
365	5	0.0056279	183.895%	143.967%	0.0001036	0.0069238
300	5	0.0056305	195.862%	141.303%	0.0001015	0.0069124
100	5	0.0057843	229.950%	136.660%	0.0001015	0.0070492
1000	10	0.0042292	136.708%	158.990%	0.0000503	0.0054251
500	10	0.0052239	177.779%	147.516%	0.0000816	0.0067373
365	10	0.0056816	195.423%	144.304%	0.0000995	0.0072991
300	10	0.0057086	195.899%	140.693%	0.0001041	0.0073129
100	10	0.0058286	228.130%	136.958%	0.0001032	0.0074374

Table 6. Random forest sliding window results with 30 previous returns, high, low, close and open values as features

Training size	Test size	MAE	MAPE	sMAPE	MSE	RMSE
1000	5	0.004331981	136.487%	165.986%	0.0000526	0.0052887
500	5	0.005235231	160.208%	160.844%	0.0000773	0.0064205
365	5	0.005786871	172.378%	158.141%	0.0000994	0.0070673
300	5	0.005771790	182.862%	155.889%	0.0000980	0.0070461
100	5	0.005938070	209.415%	150.579%	0.0000992	0.0072083
1000	10	0.004367577	137.181%	165.987%	0.0000543	0.0055787
500	10	0.005272594	162.311%	161.058%	0.0000789	0.0067350
365	10	0.005809639	167.753%	158.240%	0.0000989	0.0074285
300	10	0.005823662	185.551%	156.302%	0.0000996	0.0073638
100	10	0.005968445	208.415%	150.319%	0.0001008	0.0075318

In contrast to the RF models, using high, low, open, and close features does not yield better results than without them. Also, we can see that the sMAPE value is positively correlated with other metrics, unlike the RF model, where the sMAPE metric is negatively correlated with other metrics.

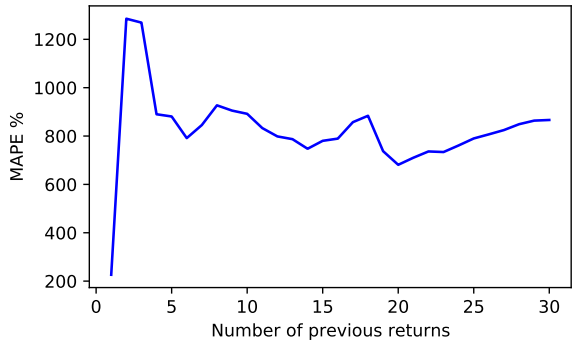

Figure 11. SVR number of previous returns MAPE results

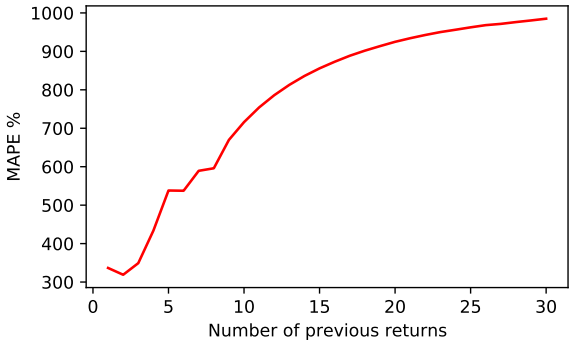

Figure 12. SVR all features MAPE results

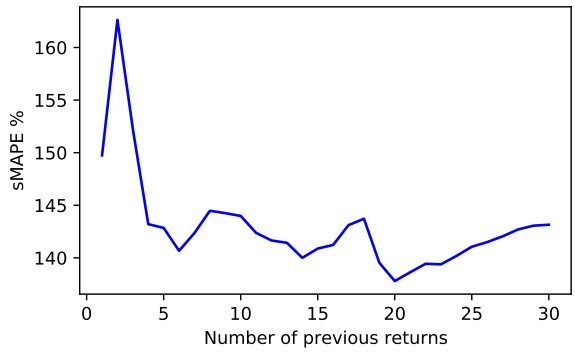

Figure 13. SVR number of previous returns sMAPE results

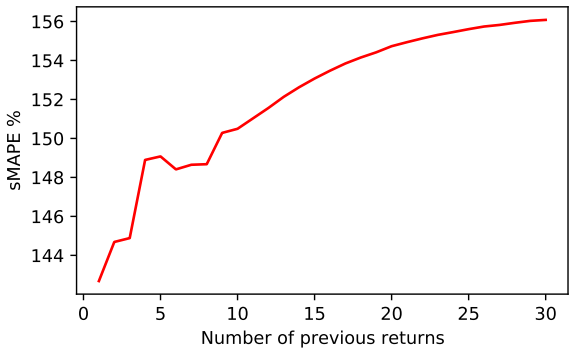

Figure 14. SVR all features sMAPE results

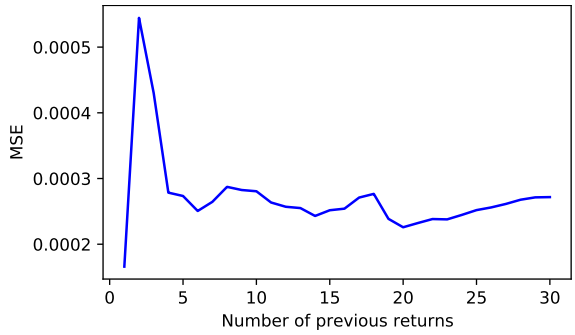

Figure 15. SVR number of previous returns MSE results

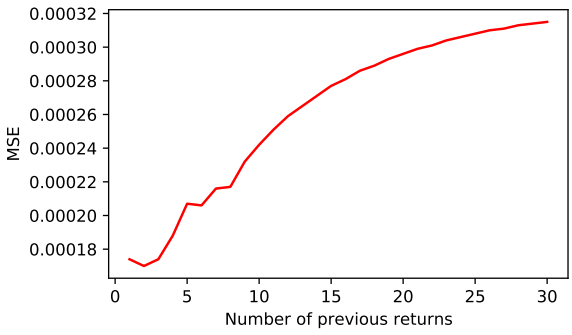

Figure 16. SVR all features MSE results

According to the SVR feature testing, the optimal number of previous returns is one. Two sets of cases are compared, as was done with the RF model. As seen in tables 7 and 8, the optimised parameters greatly improve the MAE, MAPE, MSE and RMSE metrics. We can also observe a large improvement with thirty previous returns. In this case, the MAPE value decreased from 866.428% to 105.440%. Moreover, with one previous return, the value of MAPE demonstrates an almost two times decrease. With optimised parameters, the SVR model is more effective with thirty previous

returns based on the MAE, MAPE, MSE and RMSE values. However, as in the RF model, the SVR's sMAPE largely increases with the optimised variation of the model. Therefore, we can see that the parameter optimisation offers improvements over the default parameters. Besides, it is reasonable to test the sliding window method with both sets of parameters.

Table 7. SVR results with default parameters

z prev. returns	MAE	MAPE	sMAPE	MSE	RMSE
1	0.0071281	226.278%	149.743%	0.0001659	0.0129
30	0.0123163	866.428%	143.149%	0.0002717	0.0165

Table 8. SVR results with optimised parameters

z prev. returns	MAE	MAPE	sMAPE	MSE	RMSE
1	0.006642	112.357%	172.399%	0.000156	0.0125
30	0.006687	105.440%	185.201%	0.000157	0.0125

The sliding window method is implemented with two sets of features for SVR: one and thirty previous returns. Unlike RF, the high, low, open and close values are not used for the SVR model with the sliding window implementation.

The results of the sliding window implementation with one previous return as features can be seen in Table 9. We can see that the results are mainly of the same order of magnitude. For example, the difference between the maximum value and minimum value of sMAPE is 6%. The only outlier in the results is the test with a training size of 100 and test size of 10, where the MAPE, MSE and RMSE are vastly different from other tests. However, the SVR model is more effective with a smaller test size of 5 than with a higher test size of 10.

The overview of the SVR model with sliding window implementation and thirty previous returns is presented in Table 10. We can see that the results between one and thirty previous returns do not differ much, and the same metric growth and decay patterns are also observed. For example, the MAE grows as the training size decreases. A training size of 365 also yields the best results in regards to MAPE. Additionally, sMAPE is the lowest with a training size of 365, with the result of a training size of 300 being a close second. Here, we can observe that the optimal training and test size depends heavily on the favoured metric. However, a training size of 365 with a test size of 5 is the most optimal due to its results across various metrics.

Table 9. SVR sliding window results with 1 previous return

Training size	Test size	MAE	MAPE	sMAPE	MSE	RMSE
1000	5	0.001809	134.904%	66.605%	0.000013	0.002295
500	5	0.001813	127.716%	62.403%	0.000013	0.002309
365	5	0.001887	121.033%	61.806%	0.000015	0.002398
300	5	0.001897	126.677%	61.290%	0.000015	0.002402
100	5	0.002062	140.286%	63.849%	0.000017	0.002566
1000	10	0.001813	133.369%	67.359%	0.000013	0.002459
500	10	0.001878	132.227%	62.436%	0.000015	0.002539
365	10	0.001921	123.939%	62.683%	0.000014	0.002555
300	10	0.002064	141.501%	62.529%	0.000020	0.002738
100	10	0.002505	220.737%	64.919%	0.000095	0.003334

Table 10. SVR sliding window results with 30 previous return

Training size	Test size	MAE	MAPE	sMAPE	MSE	RMSE
1000	5	0.001824	133.317%	66.993%	0.000013	0.002321
500	5	0.001839	127.577%	63.749%	0.000013	0.002324
365	5	0.001865	123.760%	61.886%	0.000013	0.002367
300	5	0.001894	127.980%	62.082%	0.000014	0.002389
100	5	0.002279	140.768%	63.960%	0.000059	0.002816
1000	10	0.001817	128.120%	66.935%	0.000013	0.002472
500	10	0.001853	127.928%	63.806%	0.000013	0.002492
365	10	0.001924	127.943%	63.265%	0.000013	0.002568
300	10	0.002000	129.264%	63.327%	0.000016	0.002663
100	10	0.002344	146.169%	65.194%	0.000032	0.003071

4.5.3 K-nearest neighbours

For KNN, the feature testing overview results are presented in tables 27 and 28 (see Appendix). When the feature set includes only the z number of previous returns, the MAPE, MAE, and MSE values vary moderately in terms of magnitude. At the same time, the sMAPE magnitude is relatively stable. We can see that no correlation is observed between the number of returns and the results as seen in figures 17, 19 and 21 and figures 27 and 29 (see Appendix).

Figure 17. KNN z number of previous returns MAPE results

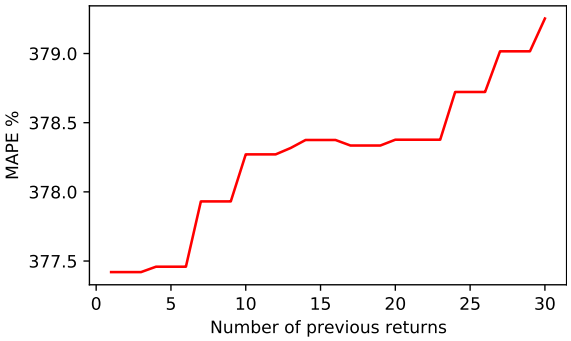

Figure 18. KNN all features MAPE results

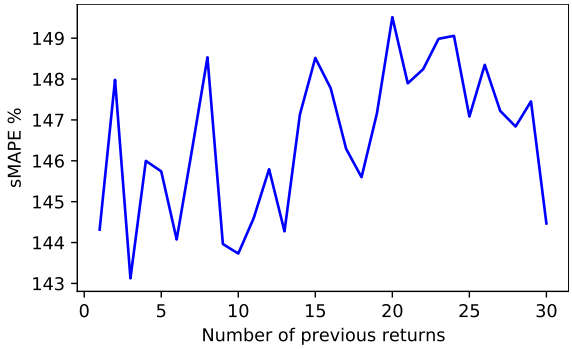

Figure 19. KNN z number of previous returns sMAPE results

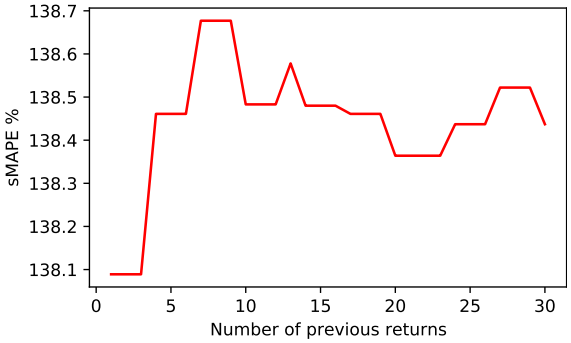

Figure 20. KNN all features sMAPE results

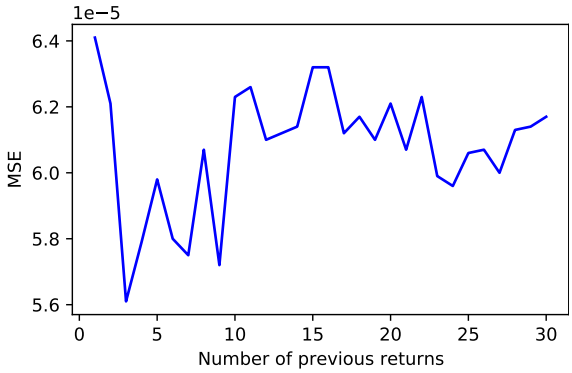

Figure 21. KNN z number of previous returns MSE results

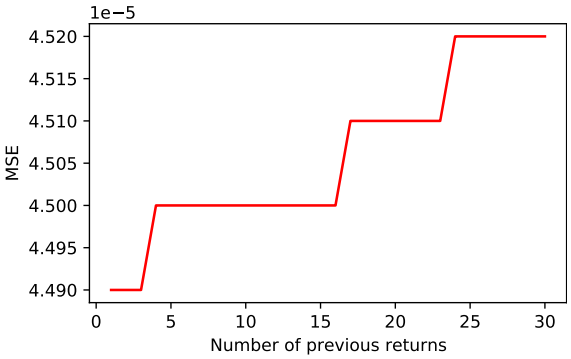

Figure 22. KNN all features MSE results

When the feature set additionally includes high, low, open and close values, the results are relatively stable (See figures 18, 20 and 22, and figures 28 and 30 in Appendix). However, as the number of previous returns increases, generally, the error metrics also increase. From the results, we can see that the most optimal number of previous returns is 1.

In general, including all the features yields better results in almost all the metrics, except the MAPE. However, the test results, including all the features, are relatively stable. Based on that, we can conclude that all the observed features should be included in the further tests for the KNN

model due to the better results.

Parameter optimisation is carried out with z previous returns, high, low, open and close values as features. However, the optimal number of z previous returns is one, according to the feature testing results where z = 1 and z = 30 for comparison with other machine learning models. Based on the parameter optimisation, the optimal k for KNN is three. As seen in Table 11 and Table 12, for a set of features where z = 1, only the MAPE metric is moderately improved with the optimisation – from 138% to 129%. The MSE and RMSE values also decrease by a meagre margin, and MAE and MAPE values increase. We can also see that for a set of features where z = 30, the model improves only regarding sMAPE, and that the values for the other metrics increase with the optimised parameters.

Table 11. KNN results with default parameters

z prev. returns	MAE	MAPE	sMAPE	MSE	RMSE
1	0.0043319	377.420%	138.089%	0.0000449	0.0067
30	0.0043565	379.253%	138.437%	0.0000452	0.0067

Table 12. KNN results with optimised parameters

z prev. returns	MAE	MAPE	sMAPE	MSE	RMSE
1	0.004337	406.823%	129.109%	0.000044	0.0066
30	0.004514	420.124%	129.911%	0.000047	0.0069

Based on the findings, we can observe that the optimised KNN model is improved only regarding sMAPE and deteriorates with regards to the other metrics. Moreover, the optimisation is not as successful for the KNN model as it is with RF and SVR.

The sliding window method is performed with two sets of features: one previous return and thirty previous returns. In both sets high, low, open and close values are also included as features. The results for the two sets are almost equal as seen in tables 13 and 14. However, the second set, where z = 30, has slightly better results. Overall, a higher training size is more favourable for the KNN model. The metrics generally increase as the training size decreases. We can see that the test size has a minor impact on the results, as all the metrics except MAPE are greater (with a test size of 10), although the difference is minor. In conclusion, the KNN performs better with z = 30 and a more extensive training size and a smaller test size of 5.

Table 13. KNN sliding window results with 1 previous return

Training size	Test size	MAE	MAPE	sMAPE	MSE	RMSE
1000	5	0.003074	201.410%	100.715%	0.000031	0.003796
500	5	0.004186	237.488%	106.728%	0.000062	0.005206
365	5	0.004818	237.170%	109.437%	0.000090	0.005949
300	5	0.004857	235.477%	109.942%	0.000087	0.005994
100	5	0.005236	254.503%	115.289%	0.000095	0.006391
1000	10	0.003126	207.538%	101.042%	0.000032	0.004066
500	10	0.004409	250.347%	108.955%	0.000069	0.005701
365	10	0.004946	240.381%	111.768%	0.000086	0.006381
300	10	0.005101	241.137%	112.827%	0.000094	0.006530
100	10	0.005432	252.849%	117.991%	0.000099	0.006909

Table 14. KNN sliding window results with 30 previous return

Training size	Test size	MAE	MAPE	sMAPE	MSE	RMSE
1000	5	0.003058	198.870%	100.521%	0.000031	0.003791
500	5	0.004060	235.659%	107.105%	0.000057	0.005035
365	5	0.004800	229.944%	109.217%	0.000088	0.005906
300	5	0.004873	229.376%	110.175%	0.000087	0.005976
100	5	0.005309	251.451%	115.660%	0.000096	0.006441
1000	10	0.003116	196.041%	101.485%	0.000033	0.004042
500	10	0.004283	242.706%	108.508%	0.000064	0.005539
365	10	0.004997	246.391%	111.847%	0.000089	0.006404
300	10	0.005055	228.944%	112.435%	0.000091	0.006496
100	10	0.005499	250.767%	118.310%	0.000098	0.006990

4.6 ARMA and machine learning comparison

In this section, we compare the ARMA model results with those of the machine learning models in two different ways. The first method is based on taking the complete training and testing on the whole data set. The second is the sliding window method. The ARMA(2,1) model is chosen for both comparisons.

The best performing models are also chosen among the machine learning models. First, we choose the standard method's best-performing machine learning model to compare with the ARMA

model. Here, it is important to note that each machine learning model excels with specific metrics. For example, the RF has the lowest MAE. SVR has the lowest MAPE, but the RF model's MAPE has close results – 105.440% and 109.421%, respectively (see Table 15 below). On the other hand, KNN has the lowest sMAPE, MSE and RMSE values. Generally, the optimal machine learning model to be compared to ARMA depends heavily on the favoured metric. However, to give a complete overview of all the metrics, each metric's best performing models should be compared with the respective ARMA metric value.

The comparison of the models is displayed in Table 15. The best result of each metric is coloured grey. As we can see, each model excels in a specific metric. For example, the ARMA(2,1) model outperforms the machine learning models in MAE, with the RF being a close second. While the SVR model and RF model (following closely) are the most optimal if MAPE is the favoured metric and KNN is the best model to achieve the lowest sMAPE. At the same time, RF exceeds others in regards to MSE and RMSE. Based on these numbers, we can conclude that there isn't a single model that outperforms others in all metrics. On the contrary, the appropriate model should be chosen according to the most favoured metric. Nevertheless, if one should choose to have an all-round model, the RF would have better results in all metrics, except sMAPE, where the KNN model excels.

Table 15. Best performing ARMA and Machine learning model comparison, standard method

Model / Metric	MAE	MAPE	sMAPE	MSE	RMSE
ARMA(2,1)	0.003871	124.312%	168.777%	0.0000516	0.0072
RF	0.003872	109.421%	182.709%	0.00005055	0.0071
SVR	0.006687	105.440%	185.201%	0.000157	0.0125
KNN	0.004514	420.124%	129.911%	0.000047	0.0069

Second, the best performing models of machine learning and ARMA are compared in the sliding window method. For the comparison, only the training and test sizes which yielded the best results in the previous sections are chosen. The following three settings of training/testing sizes are considered:

- training size = 1000 & test size = 5;
- training size = 365 & test size = 5;
- training size = 1000 & test size = 10.

The overview of the results is given in Table 16, where the lowest value of each metric is highlighted in grey. Overall, the best performing model in the sliding window method is the SVR model, which has the lowest value in at least a total of 4 metrics – MAE, sMAPE, MSE and RMSE.

Table 16. ARMA and machine learning results from the sliding window method

	Training size	Test size	MAE	MAPE	sMAPE	MSE	RMSE
ARMA(2,1)	1000	5	0.004324	122.68	171.97	0.000052	0.005269
RF	1000	5	0.004332	136.487	165.986	0.000053	0.005289
SVR	1000	5	0.001824	133.317	66.993	0.000013	0.002321
KNN	1000	5	0.003058	198.870	100.521	0.000031	0.003791
ARMA(2,1)	365	5	0.005718	131.66	165.43	0.000102	0.006931
RF	365	5	0.005787	172.378	158.141	0.000099	0.007067
SVR	365	5	0.001865	123.760	61.886	0.000013	0.002367
KNN	365	5	0.004800	229.944	109.217	0.000088	0.005906
ARMA(2,1)	1000	10	0.004327	119.84	172.93	0.000052	0.005524
RF	1000	10	0.004368	137.181	165.987	0.000054	0.005579
SVR	1000	10	0.001817	128.120	66.935	0.000013	0.002472
KNN	1000	10	0.003116	196.041	101.485	0.000033	0.004042

When using a smaller training size, SVR outperformed the other models based on all 5 metrics. In terms of sMAPE, the SVR model outperforms the other models by at least 33%. Regarding MSE, the difference is more than two times bigger. It seems that the ARMA model performs better according to the MAPE metric when the training set size is larger, while RF and KNN are relatively worse. In those cases, however, the MAPE of SVR method is not far off from that of ARMA. We can also observe that KNN provides the second best results, outperforming the ARMA model in all settings based on 4 metrics.

In conclusion, we can see that the machine learning models, in particular SVR and KNN, exceed the ARMA model in almost all of the metrics. Therefore, machine learning has great potential in predicting the OMX Baltic Benchmark returns when compared to the ARMA approach.

4.7 GARCH

Using a similar approach as for the ARMA model in Section 5.3, we consider different p and q values (from one to five for each setting, resulting in 25 different cases) and use the sliding window method.

Table 29 (see Appendix) gives an overview of the metrics of errors based on the GARCH models with different p and q values. The table consists of 25 models, where p and q have a minimum value of one and a maximum value of five. Overall, the GARCH(1,1) outperformed the other variations of the model. This finding is partially in line with Hansen and Lunde (2005), where the authors

found that GARCH(1,1) outperformed the GARCH model with p and q up to 2 lags. The only metrics where GARCH(1,1) did not outperform others are in MSE and RMSE, where GARCH(2,1) excels. Overall, the results vary when the orders of p and q are changed, although the MAPE, MAE, MSE, RMSE metrics fluctuate considerably more than the sMAPE metric. As the best model, GARCH(1,1) is used for the analysis with the sliding window method.

The sliding window method with the GARCH(1,1) model is executed with training sizes of 1000, 500, 365, 300, 100, and test sizes of 5 and 10 – a total of 10 tests. Table 17 gives an overview of the GARCH(1,1) sliding window results. The MAE values are the lowest when the training size is 1000, as modifying the test size did not impact the results a lot. The MSE and RMSE values also have the lowest values with a higher training size. MAPE is at its lowest with a smaller training size. At the same time, the sMAPE values vary between 60% and 67%. The difference between the lowest and highest sMAPE values does not exceed 8%. However, the best setting for sMAPE is a training size of 100 and a test size of 5, which shows that the recent history has more impact in regards to MAPE, and long history is important for MAE, sMAPE, MSE and RMSE.

Table 17. GARCH(1,1) sliding window results

Training size	Test size	MAE	MAPE	sMAPE	MSE	RMSE
1000	5	3.36E-06	108.772	62.30179	2.04E-11	4.11E-06
500	5	5.57E-06	120.217	60.15201	4.00E-10	6.31E-06
365	5	4.79E-06	115.560	59.88104	3.16E-10	5.42E-06
300	5	4.47E-06	110.575	60.89112	2.90E-10	5.10E-06
100	5	4.45E-06	97.607	66.09664	7.35E-10	6.00E-06
1000	10	3.53E-06	115.261	62.48914	2.32E-11	4.73E-06
500	10	6.17E-06	122.235	60.35259	8.60E-10	8.47E-06
365	10	4.79E-06	115.853	59.74211	3.14E-10	5.57E-06
300	10	4.49E-06	112.073	61.954	2.87E-10	5.65E-06
100	10	4.48E-06	95.786	66.95244	8.34E-10	7.34E-06

To sum up, in the GARCH case, a higher training size is more favourable for MAE, MSE and RMSE, and a smaller training size is more favourable for MAPE. For sMAPE, however, the optimal training size is 365. Also, it is important to note that the test size did not have a large impact on the results.

4.8 GARCH-ANN

For the GARCH-ANN model, we create various structures of the ANN model for evaluation. Six different structures, which vary in terms of the number of layers used and the dimensionality of the layer's output space, are represented in Table 18. It is important to note that here, the default optimiser RMSprop [12] is used, and no layer activation functions are used in the structure testing.

Table 18. GARCH-ANN structure testing

Structure	MAE	MAPE	sMAPE	MSE	RMSE
Structure 1	3.06E-06	58.258	66.791	2.87E-10	1.70E-05
Structure 2	3.56E-06	182.221	79.050	2.80E-10	1.67E-05
Structure 3	2.98E-06	63.332	61.652	2.86E-10	1.69E-05
Structure 4	3.13E-06	57.502	71.571	2.88E-10	1.70E-05
Structure 5	3.05E-06	58.574	65.960	2.87E-10	1.70E-05
Structure 6	3.02E-06	59.805	64.124	2.87E-10	1.69E-05

From Table 18, we can see that most of the structures perform similarly across various metrics. The only exception is Structure 2, which has a higher MAPE and a slightly higher sMAPE. At the same time, Structure 3 and Structure 2 are almost identical, apart from the dropout layer demonstrated by Structure 3. The high error is most likely avoided in the Structure 3 predictions due to the dropout layer. However, Structures 1, 4, 5 and 6 (in addition to Structure 2) do not have a dropout layer, and the results do not show a large rise in errors. On the contrary, the MAPE is lower without a dropout layer, but the sMAPE is slightly higher without the dropout layer than with. Adding more layers and increasing the dimensionality of the layer output does not impact the results a lot. Structure 6 has more layers than others, and the first layers also have higher dimensionality, but the results are still similar. Structure 6 is used to test the optimisers and layer activation functions due to the most outstanding results.

Tables 30-38 (see Appendix) give an overview of the cross-validation tests of the optimisers and the layer activation functions. We can see that cross-testing does not yield better results and that only a few combinations have considerably higher errors. For example, using the ftrl optimiser led to much larger errors in some cases.

Overall, the sigmoid-adam combination has the best balance between MAPE and sMAPE. Even though many combinations yield better MAPE or sMAPE, the other metrics have higher errors. Structure 6 with the sigmoid activation function and adam optimiser is used with the sliding window method. An overview of the GARCH-ANN sliding window method results is given in Table 19. The results indicate that a smaller test size of 5 is in general more effective than having a test size

[12] https://www.tensorflow.org/api$_d ocs/python/tf/keras/Model, retrievedon 25.04.2021$

Table 19. GARCH-ANN sliding window method

Training size	Test size	MAE	MAPE	sMAPE	MSE	RMSE
1000	5	2.64E-06	119.504	60.498	1.75E-10	3.76E-06
500	5	4.04E-06	101.804	55.645	2.47E-10	5.75E-06
365	5	5.02E-06	101.118	56.113	3.61E-10	6.98E-06
300	5	4.99E-06	98.454	56.751	3.50E-10	6.94E-06
100	5	5.07E-06	90.926	56.220	3.47E-10	7.01E-06
1000	10	1.24E-05	1074.851	144.060	4.08E-10	1.36E-05
500	10	1.30E-05	714.963	137.284	4.34E-10	1.47E-05
365	10	5.10E-06	95.357	56.486	3.63E-10	7.78E-06
300	10	5.22E-06	116.604	58.011	3.84E-10	8.06E-06
100	10	5.40E-06	105.967	59.054	3.78E-10	8.27E-06

of 10. When the test size is 5, a smaller training size yields better results for MAPE and sMAPE metrics than a larger training size. For example, with a training size of 1000, the MAPE is almost 30% higher than with a training size of 100. However, the MAE, MSE and RMSE metrics indicate that a higher training size is more beneficial than a smaller one. The values for MAE, MSE and RMSE's are almost double between the tests with a training size of 1000 and a training size of 100.

In summary, it should be noted that if minimising MAPE or sMAPE is the main priority, then a smaller training size should be used. However, if the MAE, MSE or RMSE metrics is the main priority, then a larger training size is preferred. In any case, a smaller test size should be chosen for better effectiveness.

4.9 GARCH and GARCH-ANN comparison

We compare GARCH and GARCH-ANN models using two methods: the standard method and the sliding window method. GARCH(1,1) is the best performing model among the GARCH models and the best performing GARCH-ANN model consists of the GARCH(1,1) model, the sigmoid layer activation function and adam algorithm as the optimiser. The corresponding results for the standard method presented in tables 29 and 31 in the Appendix conclude that GARCH(1,1) outperforms the GARCH-ANN model for sMAPE, MSE and RMSE metrics, but the GARCH-ANN model had lower MAE and MAPE.

The sliding window results displayed in Tables 17 and 19 show that whether the GARCH-ANN model or the GARCH model gives better predictions depends on the training and testing size, and the metric. For example, according to the sMAPE metric, GARCH-ANN outperformed the GARCH model when the test size is 5, but not when it is 10. When looking at MAPE, GARCH did

a better prediction in 4 out of 5 cases when the test size is 10. Similarly, according to the MAE and RMSE metrics, GARCH gave better predictions when test size is 10, but we cannot make a similar conclusion when thetest size is 5.

In conclusion, it is not possible to give a clear answer if GARCH or GARCH-ANN makes a better prediction, when using the sliding window method. However, if we follow the standard method of using the whole dataset to train and test, then the GARCH model is more effective when the favoured evaluation metric is sMAPE, MSE or RMSE. If MAE or MAPE is the preferred evaluation metric, then the GARCH-ANN model would be more effective.

4.10 Standardised residuals

The histograms of the standardised residuals for the standard method are displayed in figure 31 (see Appendix). The histograms show that the econometric models have most of the standardised residuals between zero and one (almost 300). At the same time, machine learning models have a lower number of standardised residuals between zero and one. Overall, the machine learning models have substantially higher standardised residuals than that of the econometric models. Also, as seen in figure 31 (see Appendix), there are almost no standardised residuals above seven for the econometric models, but there are many higher standardised residual values for the machine learning models; for example, the SVR model has the highest amount. Table 20 displays the mean absolute standardised residuals for all methods, where we can see that the econometric approach has the lowest mean – 1.8633, while the SVR approach has the highest mean of the absolute standardised residuals of 5.4157.

Table 20. Mean absolute standardised residuals with the sliding window method

Method	Mean absolute standardised residuals
Econometric	1.8633
Random forest	3.2216
SVR	5.4157
KNN	3.5365

Figure 32 (see Appendix) displays the histograms of the standardised residuals with the sliding window method. Compared to the standard method, the econometrics models have more standardised residuals in the higher range of values. In contrast to the standard method, the SVR and KNN models have substantially better results. Moreover, the SVR approach has a low number of high values of standardised residuals, as seen in figure 32 (see Appendix), while the standardised residuals of the RF do not improve a lot.

Table 21 shows the mean absolute standardised residuals using the sliding window method.

Using the sliding window method, the SVR approach has the best result with a mean of 1.2236. Simultaneously, the RF method has the highest mean, and the econometric models have a mean of 2.6327, which is higher than the standard method.

Table 21. Sliding window method mean absolute standardised residuals

Method	Mean absolute standardised residuals
Econometric	2.6327
Random forest	2.8816
SVR	1.2236
KNN	2.0968

In conclusion, we can say that the machine learning models do better with the sliding window method while the econometric models do worse. Overall, the SVR model with the sliding window method has the lowest standardised residuals.

5 Conclusion

Predicting stock returns and volatilities with higher accuracy is one of the most interesting topics in stock market analysis. That is why the challenge of finding a suitable prediction method is tempting. From the previous works, we can see that both econometric and machine learning methods could be used for prediction purposes. However, in this paper, we found that the choice of a suitable methodology depends heavily on the preferred evaluation technique and the size of the dataset. This paper contributes to the literature by providing a detailed analysis of using different econometric and machine learning techniques based on the example of the NASDAQ Baltic Stock Exchange.

The results of our analysis demonstrate that the machine learning models we consider generally outperform the ARMA approach for different training and testing sample sizes and metrics. In particular, we can say that the support vector regression (SVR) and k-nearest neighbours (KNN) provide better predictions in most of the cases we considered, while the performances of the random forest (RF) and artificial neural network GARCH (GARCH-ANN) versus the corresponding econometric models (ARMA and GARCH, respectively) depended on the training and testing sample sizes and the metric. These results are similar to that of Ballings et al. (2015), where RF and SVR models demonstrated the most accurate results among the machine learning algorithms for predicting the stock prices of European firms. Our results using GARCH and GARCH-ANN is in line with the work of Shaik and Sejpal (2020), who found no clear evidence on which model is better for India's three stock market indices. However, the comparison of GARCH and GARCH-ANN may yield different results depending on the data at hand. Siddiqui et al. (2018) showed that GARCH outperforms GARCH-ANN for net asset values prediction, while Kristjanpoller and Minutolo

(2015) found that the GARCH-ANN model improved the mean average percentage error by 25% when predicting gold price volatility. Based on previous works and our analysis, this paper suggests that when choosing an appropriate machine learning model, one should also pay attention to the metric that evaluates the prediction errors, and at the same time the training and testing sample sizes.

There are several ways to expand on the topic. For example, the dataset could be modified via filtering the outliers to minimise their impact on the dataset and later researching the prediction if the time series is not as volatile. Additionally, other machine learning models, financial models and hybrid models could be implemented, and other features could also be introduced, such as technical indicators and financial statements. Moreover, sentiment analysis could also be used to determine the public's opinion about specific stocks or the market overall, and the sentiment scores can be used as predictive features.

References

Armstrong, J. S. (1985). Long-range forecasting: From crystal ball to computer. *John-Wiley and Sons, New York*, 348.

Awad, M., & Khanna, R. Support vector regression. In: *Efficient learning machines*. Springer, 2015, pp. 67–80.

Ballings, M., Van den Poel, D., Hespeels, N., & Gryp, R. (2015). Evaluating multiple classifiers for stock price direction prediction. *Expert Systems with Applications*, *42*(20), 7046–7056.

Ban, T., Zhang, R., Pang, S., Sarrafzadeh, A., & Inoue, D. Referential knn regression for financial time series forecasting. In: *International conference on neural information processing*. Springer. 2013, 601–608.

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

Breiman, L. (2001). Random forests. *Machine learning*, *45*(1), 5–32.

Chai, T., & Draxler, R. R. (2014). Root mean square error (rmse) or mean absolute error (mae)?–arguments against avoiding rmse in the literature. *Geoscientific model development*, *7*(3), 1247–1250.

Chen, K., Zhou, Y., & Dai, F. A lstm-based method for stock returns prediction: A case study of china stock market. In: *2015 ieee international conference on big data (big data)*. IEEE. 2015, 2823–2824.

Choudhry, R., & Garg, K. (2008). A hybrid machine learning system for stock market forecasting. *World Academy of Science, Engineering and Technology*, *39*(3), 315–318.

Daly, K. (2008). Financial volatility: Issues and measuring techniques. *Physica A: statistical mechanics and its applications*, *387*(11), 2377–2393.

Dash, R., & Dash, P. K. (2016). A hybrid stock trading framework integrating technical analysis with machine learning techniques. *The Journal of Finance and Data Science*, *2*(1), 42–57.

Ercan, H. Baltic stock market prediction by using narx. In: *2017 12th international scientific and technical conference on computer sciences and information technologies (csit). 1*. IEEE. 2017, 464–467.

Everitt, B. S., & Skrondal, A. (n.d.). The cambridge dictionary of statistics. *Cambridge, UK: Cambridge University Press*.

García-Laencina, P. J., Sancho-Gómez, J.-L., Figueiras-Vidal, A. R., & Verleysen, M. (2009). K nearest neighbours with mutual information for simultaneous classification and missing data imputation. *Neurocomputing*, *72*(7-9), 1483–1493.

Grigoryan, H. et al. (2015). Stock market prediction using artificial neural networks. case study of tal1t, nasdaq omx baltic stock. *Database Systems Journal*, *6*(2), 14–23.

Guo, G., Wang, H., Bell, D., Bi, Y., & Greer, K. Knn model-based approach in classification. In: *Otm confederated international conferences" on the move to meaningful internet systems"*. Springer. 2003, 986–996.

Hansen, P. R., & Lunde, A. (2005). A forecast comparison of volatility models: Does anything beat a garch (1, 1)? *Journal of applied econometrics*, *20*(7), 873–889.

Herwartz, H. (2017). Stock return prediction under garch — an empirical assessment. *International Journal of Forecasting*, *33*(3), 569–580. https://doi.org/https://doi.org/10.1016/j.ijforecast.2017.01.002

Hu, Y., Tao, Z., Xing, D., Pan, Z., Zhao, J., & Chen, X. (2020). Research on stock returns forecast of the four major banks based on ARMA and GARCH model. *Journal of Physics: Conference Series*, *1616*, 012075. https://doi.org/10.1088/1742-6596/1616/1/012075

Huang, W., Nakamori, Y., & Wang, S.-Y. (2005). Forecasting stock market movement direction with support vector machine. *Computers & operations research*, *32*(10), 2513–2522.

Jang, H., & Lee, J. (2017). An empirical study on modeling and prediction of bitcoin prices with bayesian neural networks based on blockchain information. *Ieee Access*, *6*, 5427–5437.

Kara, Y., Boyacioglu, M. A., & Baykan, Ö. K. (2011). Predicting direction of stock price index movement using artificial neural networks and support vector machines: The sample of the istanbul stock exchange. *Expert systems with Applications*, *38*(5), 5311–5319.

Khaidem, L., Saha, S., & Dey, S. R. (2016). Predicting the direction of stock market prices using random forest. *arXiv preprint arXiv:1605.00003*.

Khan, W., Ghazanfar, M. a., Azam, M. A., Karami, A., Alyoubi, K., & Alfakeeh, A. (2020). Stock market prediction using machine learning classifiers and social media, news. *Journal of Ambient Intelligence and Humanized Computing*. https://doi.org/10.1007/s12652-020-01839-w

Kim, S., & Kim, H. (2016). A new metric of absolute percentage error for intermittent demand forecasts. *International Journal of Forecasting*, *32*(3), 669–679.

Kim, Y., & Enke, D. (2016). Using neural networks to forecast volatility for an asset allocation strategy based on the target volatility, procedia computer science. *Procedia Computer Science*, *95*, 281–286.

Kristjanpoller, W., & Minutolo, M. C. (2015). Gold price volatility: A forecasting approach using the artificial neural network–garch model. *Expert systems with applications*, *42*(20), 7245–7251.

Lapitskaya, D., Eratalay, H., & Sharma, R. (2021). Predicting stock returns: Armax vs. machine learning. *Advances in Econometrics, Operational Research, Data Science and Actuarial Studies - Techniques and Theories*.

Lebedeva, E. (2018). *Spillovers between cryptocurrencies. network map of cryptocurrencies* (Doctoral dissertation). Master's Thesis, University of Tartu, Tartu, Estonia.

Li, M., Yang, C., Zhang, J., Puthal, D., Luo, Y., & Li, J. Stock market analysis using social networks. In: *Proceedings of the australasian computer science week multiconference*. New York, NY, USA: Association for Computing Machinery, 2018. ISBN: 9781450354363.

Liu, W. K., & So, M. K. P. (2020). A garch model with artificial neural networks. *Information*, *11*(10). https://www.mdpi.com/2078-2489/11/10/489

Long, W., Lu, Z., & Cui, L. (2019). Deep learning-based feature engineering for stock price movement prediction. *Knowledge-Based Systems*, *164*, 163–173.

Nakajima, J. (2017). Bayesian analysis of multivariate stochastic volatility with skew return distribution. *Econometric Reviews*, *36*(5), 546–562.

Nelson, D. M., Pereira, A. C., & de Oliveira, R. A. Stock market's price movement prediction with lstm neural networks. In: *2017 international joint conference on neural networks (ijcnn)*. IEEE. 2017, 1419–1426.

Nikkinen, J., Piljak, V., & Äijö, J. (2012). Baltic stock markets and the financial crisis of 2008–2009. *Research in International Business and Finance*, *26*(3), 398–409.

Noble, W. S. (2006). What is a support vector machine? *Nature biotechnology*, *24*(12), 1565–1567.

Oberholzer, N., & Venter, P. (2015). Univariate garch models applied to the jse/ftse stock indices. *Procedia Economics and Finance*, *24*, 491–500.

Patel, J., Shah, S., Thakkar, P., & Kotecha, K. (2015). Predicting stock market index using fusion of machine learning techniques. *Expert Systems with Applications*, *42*(4), 2162–2172.

Rossi, A. G. (2018). Predicting stock market returns with machine learning. *Georgetown University*.

Rounaghi, M. M., & Zadeh, F. N. (2016). Investigation of market efficiency and financial stability between s&p 500 and london stock exchange: Monthly and yearly forecasting of time series stock returns using arma model. *Physica A: Statistical Mechanics and its Applications*, *456*, 10–21.

Shah, V. H. (2007). Machine learning techniques for stock prediction. *Foundations of Machine Learning | Spring*, *1*(1), 6–12.

Shaik, M., & Sejpal, A. (2020). Comparison of garch and ann model for forecasting volatility: Evidence based on indian stock markets. *Journal of Prediction Markets*, *14*(2).

Siddiqui, M. U., Abbas, A., AbdurRehman, S. M., Jawed, A., & Rafi, M. Comparison of garch model and artificial neural network for mutual fund's growth prediction. In: *2018 international conference on computing, mathematics and engineering technologies (icomet)*. IEEE. 2018, 1–7.

Suimets, A. (2020, September). Kas praegu on õige aeg investeerida tallinna börsile? Retrieved March 3, 2021, from https://kukkur.swedbank.ee/investeerimine/kas-praegu-on-oige-aeg-investeerida-tallinna-borsile

Ticknor, J. L. (2013). A bayesian regularized artificial neural network for stock market forecasting. *Expert Systems with Applications*, *40*(14), 5501–5506.

Trafalis, T. B., & Ince, H. Support vector machine for regression and applications to financial forecasting. In: *Proceedings of the ieee-inns-enns international joint conference on neural networks. ijcnn 2000. neural computing: New challenges and perspectives for the new millennium*. *6*. IEEE. 2000, 348–353.

Wang, W., Guo, Y., Niu, Z., & Cao, Y. Stock indices analysis based on arma-garch model. In: *2009 ieee international conference on industrial engineering and engineering management*. 2009, 2143–2147. https://doi.org/10.1109/IEEM.2009.5373131.

Wang, Z., & Bovik, A. C. (2009). Mean squared error: Love it or leave it? a new look at signal fidelity measures. *IEEE signal processing magazine*, *26*(1), 98–117.

Wanjawa, B., & Muchemi, L. (2015). Ann model to predict stock prices at stock exchange markets. *ArXiv*, *abs/1502.06434*.

Whittle, P. (1951). *Hypothesis testing in time series analysis* (Vol. 4). Almqvist & Wiksells boktr.

Willmott, C. J., & Matsuura, K. (2005). Advantages of the mean absolute error (mae) over the root mean square error (rmse) in assessing average model performance. *Climate research*, *30*(1), 79–82.

Yeze, Z., & Yiying, W. (2019). Stock price prediction based on information entropy and artificial neural network. *2019 5th International Conference on Information Management (ICIM)*, 248–251.

Yu, Y., Zhu, Y., Li, S., & Wan, D. (2014). Time series outlier detection based on sliding window prediction. *Mathematical problems in Engineering*, *2014*.

Appendix

I. Tables

Table 22. ARMA orders of p and q test results

p	q	MAE	MSE	RMSE	MAPE	sMAPE
1	1	0.003876	0.00005198	0.0072	126.187	167.512
1	2	0.003876	0.00005233	0.0072	125.514	167.152
1	3	0.003872	0.00005208	0.0072	128.072	165.458
1	4	0.003873	0.0000519	0.0072	126.398	166.815
1	5	0.003859	0.0000519	0.0072	137.670	160.147
2	1	0.003871	0.00005161	0.0072	124.312	168.777
2	2	0.003861	0.00005133	0.0072	133.233	164.832
2	3	0.003922	0.00005229	0.0072	169.811	159.358
2	4	0.003878	0.0000523	0.0072	126.494	166.440
2	5	0.003892	0.00005354	0.0073	136.976	161.392
3	1	0.003870	0.00005162	0.0072	128.040	165.935
3	2	0.003866	0.00005112	0.0071	125.766	165.329
3	3	0.003879	0.00005211	0.0072	137.051	161.123
3	4	0.003876	0.00005163	0.0072	132.958	165.532
3	5	0.003917	0.00005334	0.0073	148.028	159.532
4	1	0.003872	0.00005176	0.0072	128.919	165.820
4	2	0.003872	0.00005191	0.0072	128.282	165.895
4	3	0.004512	0.00007678	0.0088	204.388	164.656
4	4	0.003848	0.00005128	0.0072	128.991	162.887
4	5	0.003905	0.0000549	0.0074	167.689	155.925
5	1	0.003866	0.00005144	0.0072	133.209	164.453
5	2	0.003868	0.0000515	0.0072	132.255	164.455
5	3	0.003990	0.00007677	0.0088	148.054	161.621
5	4	0.003876	0.00005206	0.0072	134.537	165.062
5	5	0.003897	0.00005279	0.0073	154.040	160.215

Table 23. Random forest results with z number of previous returns as features

z prev. returns	MAE	MAPE	sMAPE	MSE	RMSE
1	0.0052959	469.225	143.818	0.0000770	0.0088
2	0.0044832	333.410	147.209	0.0000563	0.0075
3	0.0042926	300.801	149.292	0.0000528	0.0073
4	0.0041397	237.931	148.567	0.0000518	0.0072
5	0.0040083	214.825	147.881	0.0000511	0.0071
6	0.0039393	196.985	149.634	0.0000506	0.0071
7	0.0039410	207.665	151.329	0.0000506	0.0071
8	0.0039294	198.348	154.204	0.0000490	0.0070
9	0.0039148	186.982	153.364	0.0000487	0.0070
10	0.0038973	177.828	153.518	0.0000481	0.0069
11	0.0039089	178.874	156.561	0.0000487	0.0070
12	0.0038696	175.585	156.070	0.0000484	0.0070
13	0.0039050	174.623	158.025	0.0000492	0.0070
14	0.0038995	173.675	156.893	0.0000490	0.0070
15	0.0039023	166.379	159.744	0.0000493	0.0070
16	0.0038977	162.986	159.815	0.0000495	0.0070
17	0.0038865	159.830	159.335	0.0000495	0.0070
18	0.0038874	161.113	159.119	0.0000492	0.0070
19	0.0038810	162.124	158.474	0.0000499	0.0071
20	0.0038966	162.431	159.554	0.0000501	0.0071
21	0.0038890	154.264	159.386	0.0000506	0.0071
22	0.0038973	158.702	158.886	0.0000506	0.0071
23	0.0038998	160.707	159.739	0.0000499	0.0071
24	0.0039023	160.138	159.998	0.0000499	0.0071
25	0.0038973	158.256	160.553	0.0000495	0.0070
26	0.0038937	156.805	160.826	0.0000493	0.0070
27	0.0038922	150.446	161.143	0.0000495	0.0070
28	0.0039050	152.843	162.439	0.0000492	0.0070
29	0.0039153	150.154	162.433	0.0000493	0.0070
30	0.0039042	151.448	162.099	0.0000494	0.0070

Table 24. Random forest results with z number of previous returns, high, low, open and close values as features

z prev. returns	MAE	MAPE	sMAPE	MSE	RMSE
1	0.00453270	340.681	142.120	0.00005382	0.0073
2	0.00435649	309.805	145.690	0.00005248	0.0072
3	0.00419698	263.201	146.391	0.00005028	0.0071
4	0.00408411	230.473	146.820	0.00004951	0.0070
5	0.00401075	210.429	148.088	0.00004932	0.0070
6	0.00397919	204.374	149.262	0.00004974	0.0071
7	0.00396841	205.916	149.698	0.00004962	0.0070
8	0.00396711	204.630	149.607	0.00004893	0.0070
9	0.00395640	196.495	149.702	0.00004832	0.0070
10	0.00392679	181.626	150.644	0.00004812	0.0069
11	0.00392788	175.247	152.698	0.00004848	0.0070
12	0.00391155	177.379	153.391	0.00004824	0.0069
13	0.00392271	171.068	154.486	0.00004870	0.0070
14	0.00390076	161.630	154.954	0.00004847	0.0070
15	0.00392215	158.702	156.622	0.00004917	0.0070
16	0.00391058	159.098	156.894	0.00004914	0.0070
17	0.00391353	158.071	156.554	0.00004945	0.0070
18	0.00392472	156.476	157.816	0.00004945	0.0070
19	0.00391391	149.576	158.194	0.00004952	0.0070
20	0.00392486	156.166	157.615	0.00004996	0.0071
21	0.00392728	152.449	157.995	0.00005053	0.0071
22	0.00392128	150.529	157.985	0.00005031	0.0071
23	0.00393199	155.495	159.228	0.00004979	0.0071
24	0.00393077	154.785	159.367	0.00004964	0.0070
25	0.00392367	159.604	161.343	0.00004934	0.0070
26	0.00392391	156.427	159.232	0.00004934	0.0070
27	0.00392948	156.751	161.056	0.00004946	0.0070
28	0.00393870	155.605	162.332	0.00004923	0.0070
29	0.00393208	146.696	161.457	0.00004982	0.0071
30	0.00394362	151.616	162.270	0.00004974	0.0071

Table 25. SVR results with high, low, close open and z number of previous returns as features

z prev. returns	MAE	MAPE	sMAPE	MSE	RMSE
1	0.0071281	226.278	149.743	0.0001659	0.0129
2	0.0185368	1285.321	162.623	0.0005445	0.0233
3	0.0170955	1269.167	152.241	0.0004301	0.0207
4	0.0125017	890.102	143.206	0.0002785	0.0167
5	0.0123816	880.922	142.846	0.0002734	0.0165
6	0.0114298	791.442	140.674	0.0002506	0.0158
7	0.0120357	845.513	142.364	0.0002645	0.0163
8	0.0129239	927.041	144.476	0.0002873	0.0169
9	0.0127585	905.226	144.252	0.0002825	0.0168
10	0.0126607	891.926	143.986	0.0002806	0.0167
11	0.0119865	832.990	142.376	0.0002635	0.0162
12	0.0117165	798.489	141.653	0.0002570	0.0160
13	0.0116267	787.114	141.432	0.0002550	0.0160
14	0.0111193	747.410	140.006	0.0002430	0.0156
15	0.0114737	780.215	140.884	0.0002517	0.0159
16	0.0115810	789.367	141.229	0.0002541	0.0159
17	0.0122861	857.448	143.114	0.0002710	0.0165
18	0.0125164	883.829	143.721	0.0002766	0.0166
19	0.0109282	737.403	139.530	0.0002386	0.0154
20	0.0103418	681.046	137.796	0.0002258	0.0150
21	0.0106389	710.246	138.630	0.0002322	0.0152
22	0.0109151	736.288	139.435	0.0002383	0.0154
23	0.0108950	733.836	139.386	0.0002378	0.0154
24	0.0111957	761.162	140.179	0.0002446	0.0156
25	0.0115122	790.003	141.057	0.0002519	0.0159
26	0.0116863	807.025	141.500	0.0002560	0.0160
27	0.0119038	824.726	142.045	0.0002613	0.0162
28	0.0121591	848.943	142.699	0.0002677	0.0164
29	0.0123041	863.794	143.059	0.0002713	0.0165
30	0.0123163	866.428	143.149	0.0002717	0.0165

Table 26. SVR results with z number of previous returns as features

z prev. returns	MAE	MAPE	sMAPE	MSE	RMSE
1	0.0078907	336.644	142.682	0.000174	0.0132
2	0.0077225	318.936	144.688	0.000170	0.0131
3	0.0080221	349.156	144.886	0.000174	0.0132
4	0.0088102	433.743	148.897	0.000188	0.0137
5	0.0097488	538.073	149.077	0.000207	0.0144
6	0.0097202	537.465	148.412	0.000206	0.0143
7	0.0102297	589.310	148.649	0.000216	0.0147
8	0.0102767	595.910	148.678	0.000217	0.0147
9	0.0109943	669.755	150.284	0.000232	0.0152
10	0.0114400	715.953	150.495	0.000242	0.0156
11	0.0118227	754.184	151.028	0.000251	0.0158
12	0.0121488	785.978	151.561	0.000259	0.0161
13	0.0124328	812.874	152.135	0.000265	0.0163
14	0.0126798	836.020	152.629	0.000271	0.0165
15	0.0128941	855.792	153.073	0.000277	0.0166
16	0.0130829	872.963	153.475	0.000281	0.0168
17	0.0132521	888.585	153.846	0.000286	0.0169
18	0.0133995	901.841	154.152	0.000289	0.0170
19	0.0135305	913.591	154.418	0.000293	0.0171
20	0.0136539	924.839	154.732	0.000296	0.0172
21	0.0137587	934.191	154.939	0.000299	0.0173
22	0.0138539	942.619	155.134	0.000301	0.0174
23	0.0139399	950.192	155.313	0.000304	0.0174
24	0.0140201	956.250	155.457	0.000306	0.0175
25	0.0140917	962.508	155.607	0.000308	0.0175
26	0.0141570	968.213	155.742	0.000310	0.0176
27	0.0142170	971.503	155.823	0.000311	0.0176
28	0.0142708	976.186	155.933	0.000313	0.0177
29	0.0143196	980.432	156.032	0.000314	0.0177
30	0.0143594	984.980	156.082	0.000315	0.0178

Table 27. KNN results with z number of previous returns as features

z prev. returns	MAE	MAPE	sMAPE	MSE	RMSE
1	0.0047606	369.925	144.315	0.0000641	0.0080
2	0.0046811	357.278	147.982	0.0000621	0.0079
3	0.0045044	371.885	143.125	0.0000561	0.0075
4	0.0045869	290.439	145.998	0.0000579	0.0076
5	0.0045283	310.523	145.741	0.0000598	0.0077
6	0.0044336	297.080	144.074	0.0000580	0.0076
7	0.0043958	281.959	146.275	0.0000575	0.0076
8	0.0044230	259.937	148.532	0.0000607	0.0078
9	0.0043117	270.017	143.964	0.0000572	0.0076
10	0.0042986	243.104	143.730	0.0000623	0.0079
11	0.0043377	233.440	144.597	0.0000626	0.0079
12	0.0043024	249.190	145.794	0.0000610	0.0078
13	0.0042876	232.070	144.274	0.0000612	0.0078
14	0.0043653	253.888	147.133	0.0000614	0.0078
15	0.0043717	271.345	148.517	0.0000632	0.0079
16	0.0043432	249.885	147.773	0.0000632	0.0079
17	0.0043217	279.719	146.293	0.0000612	0.0078
18	0.0043395	262.353	145.598	0.0000617	0.0079
19	0.0043209	270.228	147.161	0.0000610	0.0078
20	0.0043772	278.542	149.514	0.0000621	0.0079
21	0.0043470	267.713	147.895	0.0000607	0.0078
22	0.0043425	256.855	148.240	0.0000623	0.0079
23	0.0043027	244.428	148.984	0.0000599	0.0077
24	0.0042582	252.819	149.057	0.0000596	0.0077
25	0.0042596	241.657	147.082	0.0000606	0.0078
26	0.0043107	236.553	148.348	0.0000607	0.0078
27	0.0042963	264.332	147.221	0.0000600	0.0077
28	0.0043111	252.394	146.838	0.0000613	0.0078
29	0.0042948	265.302	147.454	0.0000614	0.0078
30	0.0043344	282.993	144.467	0.0000617	0.0079

Table 28. KNN results with z number of previous returns, high, low, open and close values as features

z prev. returns	MAE	MAPE	sMAPE	MSE	RMSE
1	0.0043319	377.420%	138.089%	0.0000449	0.0067
2	0.0043319	377.420%	138.089%	0.0000449	0.0067
3	0.0043319	377.420%	138.089%	0.0000449	0.0067
4	0.0043386	377.459%	138.461%	0.0000450	0.0067
5	0.0043386	377.459%	138.461%	0.0000450	0.0067
6	0.0043386	377.459%	138.461%	0.0000450	0.0067
7	0.0043431	377.931%	138.677%	0.0000450	0.0067
8	0.0043431	377.931%	138.677%	0.0000450	0.0067
9	0.0043431	377.931%	138.677%	0.0000450	0.0067
10	0.0043427	378.271%	138.483%	0.0000450	0.0067
11	0.0043427	378.271%	138.483%	0.0000450	0.0067
12	0.0043427	378.271%	138.483%	0.0000450	0.0067
13	0.0043445	378.317%	138.578%	0.0000450	0.0067
14	0.0043453	378.375%	138.480%	0.0000450	0.0067
15	0.0043453	378.375%	138.480%	0.0000450	0.0067
16	0.0043453	378.375%	138.480%	0.0000450	0.0067
17	0.0043474	378.335%	138.461%	0.0000451	0.0067
18	0.0043474	378.335%	138.461%	0.0000451	0.0067
19	0.0043474	378.335%	138.461%	0.0000451	0.0067
20	0.0043496	378.377%	138.364%	0.0000451	0.0067
21	0.0043496	378.377%	138.364%	0.0000451	0.0067
22	0.0043496	378.377%	138.364%	0.0000451	0.0067
23	0.0043496	378.377%	138.364%	0.0000451	0.0067
24	0.0043529	378.722%	138.437%	0.0000452	0.0067
25	0.0043529	378.722%	138.437%	0.0000452	0.0067
26	0.0043529	378.722%	138.437%	0.0000452	0.0067
27	0.0043569	379.016%	138.522%	0.0000452	0.0067
28	0.0043569	379.016%	138.522%	0.0000452	0.0067
29	0.0043569	379.016%	138.522%	0.0000452	0.0067
30	0.0043565	379.253%	138.437%	0.0000452	0.0067

Table 29. GARCH orders of p and q test results

p	q	MAE	MAPE	sMAPE	MSE	RMSE
1	1	4.16E-06	66.972	51.050	2.04E-10	1.43E-05
1	2	6.16E-06	106.751	57.429	3.92E-10	1.98E-05
1	3	6.33E-06	115.524	59.001	3.91E-10	1.98E-05
1	4	6.40E-06	124.390	60.561	3.72E-10	1.93E-05
1	5	6.46E-06	133.085	61.555	3.84E-10	1.96E-05
2	1	3.97E-06	72.304	52.018	1.33E-10	1.15E-05
2	2	8.43E-06	209.655	71.817	4.28E-10	2.07E-05
2	3	5.89E-06	122.275	60.096	2.67E-10	1.63E-05
2	4	5.94E-06	130.968	61.220	2.50E-10	1.58E-05
2	5	5.97E-06	139.548	62.243	2.44E-10	1.56E-05
3	1	8.02E-06	208.194	72.160	4.14E-10	2.03E-05
3	2	8.78E-06	220.065	73.488	4.74E-10	2.18E-05
3	3	6.51E-06	131.500	62.271	3.31E-10	1.82E-05
3	4	6.51E-06	140.150	63.295	3.10E-10	1.76E-05
3	5	6.54E-06	148.150	64.128	2.99E-10	1.73E-05
4	1	8.22E-06	212.977	72.970	4.41E-10	2.10E-05
4	2	9.04E-06	223.067	74.103	5.16E-10	2.27E-05
4	3	6.84E-06	136.827	63.606	3.81E-10	1.95E-05
4	4	6.81E-06	144.743	64.322	3.59E-10	1.89E-05
4	5	6.86E-06	152.805	65.045	3.47E-10	1.86E-05
5	1	8.39E-06	218.103	73.642	4.62E-10	2.15E-05
5	2	9.26E-06	228.399	74.863	5.38E-10	2.32E-05
5	3	7.03E-06	142.724	64.977	3.93E-10	1.98E-05
5	4	7.00E-06	150.126	65.530	3.74E-10	1.93E-05
5	5	7.00E-06	157.930	66.232	3.58E-10	1.89E-05

Table 30. GARCH-ANN relu layer activation function results

Optimiser	MAE	MAPE	sMAPE	MSE	RMSE
rmsprop	3.12E-06	124.7022	67.33394	2.81E-10	1.68E-05
adam	2.95E-06	88.68045	60.7522	2.83E-10	1.68E-05
adadelta	2.93E-06	80.97395	59.93846	2.84E-10	1.68E-05
adagrad	2.93E-06	78.34828	59.81356	2.84E-10	1.69E-05
adamax	3.01E-06	104.7715	63.38418	2.82E-10	1.68E-05
nadam	2.93E-06	83.11572	60.14456	2.84E-10	1.68E-05
ftrl	4.25E-05	3079.608	200	2.09E-09	4.57E-05

Table 31. GARCH-ANN sigmoid layer activation function results

Optimiser	MAE	MAPE	sMAPE	MSE	RMSE
rmsprop	4.17E-06	115.5875	200	2.97E-10	1.72E-05
adam	2.98E-06	62.8196	61.86965	2.86E-10	1.69E-05
adadelta	2.93E-06	81.31975	60.00108	2.84E-10	1.69E-05
adagrad	2.93E-06	77.43968	59.78577	2.84E-10	1.69E-05
adamax	3.01E-06	60.41674	63.48281	2.87E-10	1.69E-05
nadam	3.05E-06	112.6613	64.87926	2.82E-10	1.68E-05
ftrl	2.95E-06	68.75039	60.20398	2.85E-10	1.69E-05

Table 32. GARCH-ANN softmax layer activation function results

Optimiser	MAE	MAPE	sMAPE	MSE	RMSE
rmsprop	3.79E-06	209.0068	83.89194	2.79E-10	1.67E-05
adam	2.94E-06	69.6599	60.08016	2.85E-10	1.69E-05
adadelta	2.94E-06	86.32181	60.48747	2.84E-10	1.68E-05
adagrad	2.95E-06	88.21622	60.71242	2.83E-10	1.68E-05
adamax	2.94E-06	71.64111	59.85888	2.85E-10	1.69E-05
nadam	2.96E-06	65.20224	60.98233	2.86E-10	1.69E-05
ftrl	2.94E-06	85.69063	60.41954	2.84E-10	1.68E-05

Table 33. GARCH-ANN softplus layer activation function results

Optimiser	MAE	MAPE	sMAPE	MSE	RMSE
rmsprop	3.42E-06	164.3105	75.64893	2.80E-10	1.67E-05
adam	2.96E-06	92.10317	61.2234	2.83E-10	1.68E-05
adadelta	2.93E-06	78.49231	59.82324	2.84E-10	1.69E-05
adagrad	3.02E-06	107.9957	63.98045	2.82E-10	1.68E-05
adamax	2.93E-06	80.29211	59.91874	2.84E-10	1.69E-05
nadam	3.27E-06	145.7434	71.79555	2.81E-10	1.68E-05
ftrl	2.95E-06	90.86379	61.05119	2.83E-10	1.68E-05

Table 34. GARCH-ANN softsign layer activation function results

Optimiser	MAE	MAPE	sMAPE	MSE	RMSE
rmsprop	3.26E-06	58.66021	81.8609	2.90E-10	1.70E-05
adam	2.93E-06	79.36901	59.83047	2.84E-10	1.69E-05
adadelta	2.94E-06	82.87181	60.14216	2.84E-10	1.68E-05
adagrad	2.94E-06	83.9401	60.24264	2.84E-10	1.68E-05
adamax	3.05E-06	112.8718	64.90179	2.82E-10	1.68E-05
nadam	2.93E-06	76.71803	59.73889	2.84E-10	1.69E-05
ftrl	4.26E-05	3086.755	200	2.10E-09	4.58E-05

Table 35. GARCH-ANN tanh layer activation function results

Optimiser	MAE	MAPE	sMAPE	MSE	RMSE
rmsprop	3.05E-06	113.3224	65.00926	2.82E-10	1.68E-05
adam	3.14E-06	57.48152	72.32692	2.88E-10	1.70E-05
adadelta	2.93E-06	81.68106	60.03393	2.84E-10	1.69E-05
adagrad	2.93E-06	72.73026	59.78111	2.85E-10	1.69E-05
adamax	2.94E-06	71.05777	59.89117	2.85E-10	1.69E-05
nadam	3.12E-06	124.7693	67.33987	2.81E-10	1.68E-05
ftrl	4.25E-05	3081.122	200	2.09E-09	4.57E-05

Table 36. GARCH-ANN selu layer activation function results

Optimiser	MAE	MAPE	sMAPE	MSE	RMSE
rmsprop	3.17E-06	57.59447	74.99062	2.89E-10	1.70E-05
adam	2.99E-06	61.72146	62.40456	2.86E-10	1.69E-05
adadelta	2.93E-06	78.83218	59.86649	2.84E-10	1.69E-05
adagrad	2.93E-06	72.7798	59.80047	2.85E-10	1.69E-05
adamax	2.93E-06	83.17015	60.13983	2.84E-10	1.68E-05
nadam	3.28E-06	147.8645	72.22635	2.80E-10	1.67E-05
ftrl	4.21E-05	3047.316	200	2.05E-09	4.53E-05

Table 37. GARCH-ANN elu layer activation function results

Optimiser	MAE	MAPE	sMAPE	MSE	RMSE
rmsprop	3.53E-06	178.9312	78.44511	2.80E-10	1.67E-05
adam	2.94E-06	85.03473	60.31237	2.83E-10	1.68E-05
adadelta	2.94E-06	82.85015	60.13854	2.84E-10	1.68E-05
adagrad	2.93E-06	80.13859	59.91075	2.84E-10	1.69E-05
adamax	3.07E-06	116.181	65.57151	2.82E-10	1.68E-05
nadam	2.94E-06	68.60321	60.18833	2.85E-10	1.69E-05
ftrl	4.24E-05	3069.495	200	2.08E-09	4.56E-05

Table 38. GARCH-ANN exponential layer activation function results

Optimiser	MAE	MAPE	sMAPE	MSE	RMSE
rmsprop	4.06E-06	238.4027	88.72346	2.79E-10	1.67E-05
adam	2.96E-06	93.92799	61.49769	2.83E-10	1.68E-05
adadelta	2.93E-06	82.2518	60.08522	2.84E-10	1.69E-05
adagrad	3.22E-06	58.05996	78.79319	2.89E-10	1.70E-05
adamax	3.00E-06	61.28713	62.76268	2.87E-10	1.69E-05
nadam	2.96E-06	66.34733	60.66262	2.86E-10	1.69E-05
ftrl	2.94E-06	69.03516	60.16305	2.85E-10	1.69E-05

II. Charts

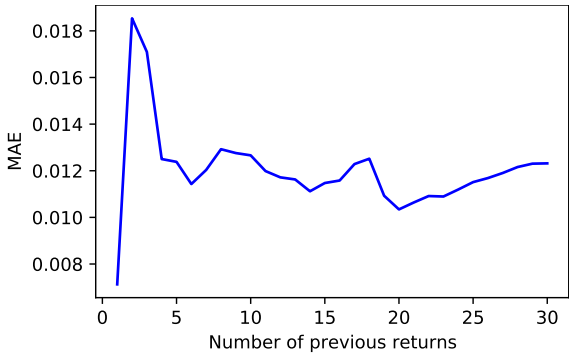

Figure 23. SVR number of previous returns MAE results

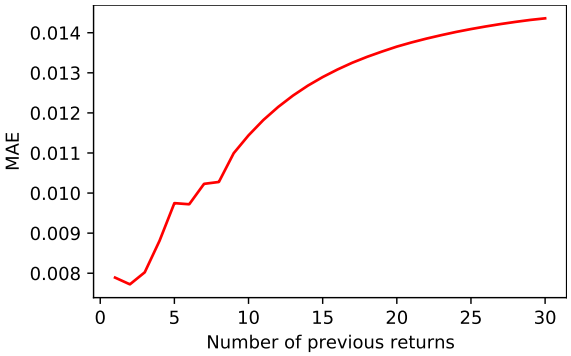

Figure 24. SVR all features MAE results

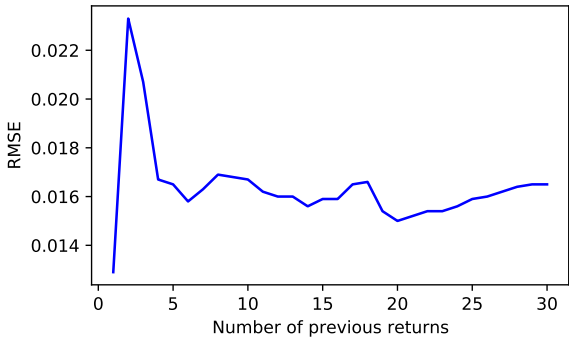

Figure 25. SVR nr of previous returns RMSE results

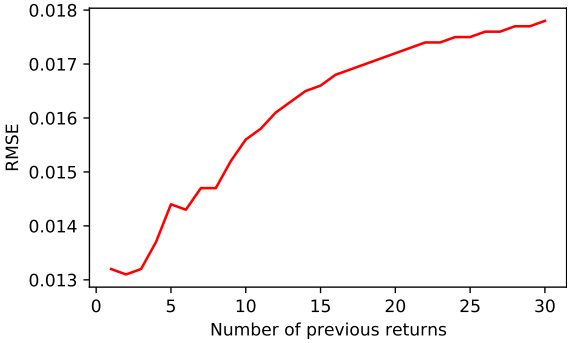

Figure 26. SVR all features RMSE results

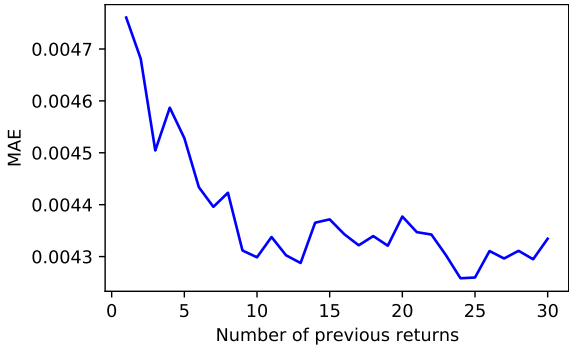

Figure 27. KNN number of previous returns MAE results

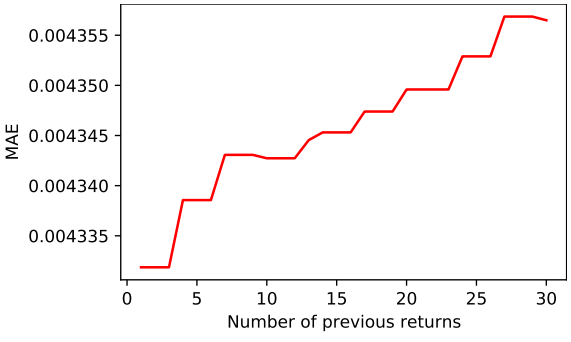

Figure 28. KNN all features MAE results

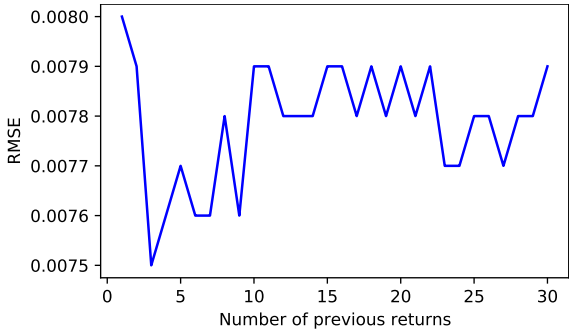

Figure 29. KNN number of previous returns RMSE results

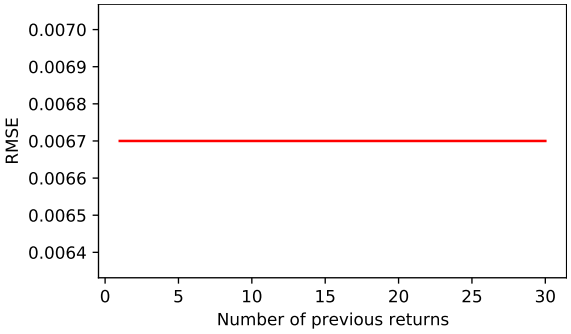

Figure 30. KNN all features RMSE results

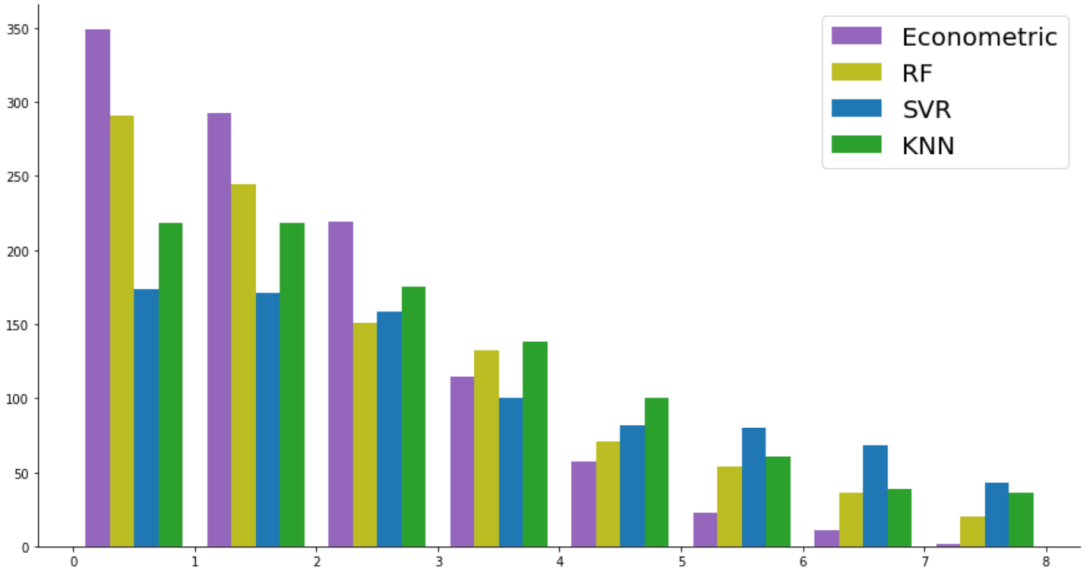

Figure 31. Standardised residuals with standard method

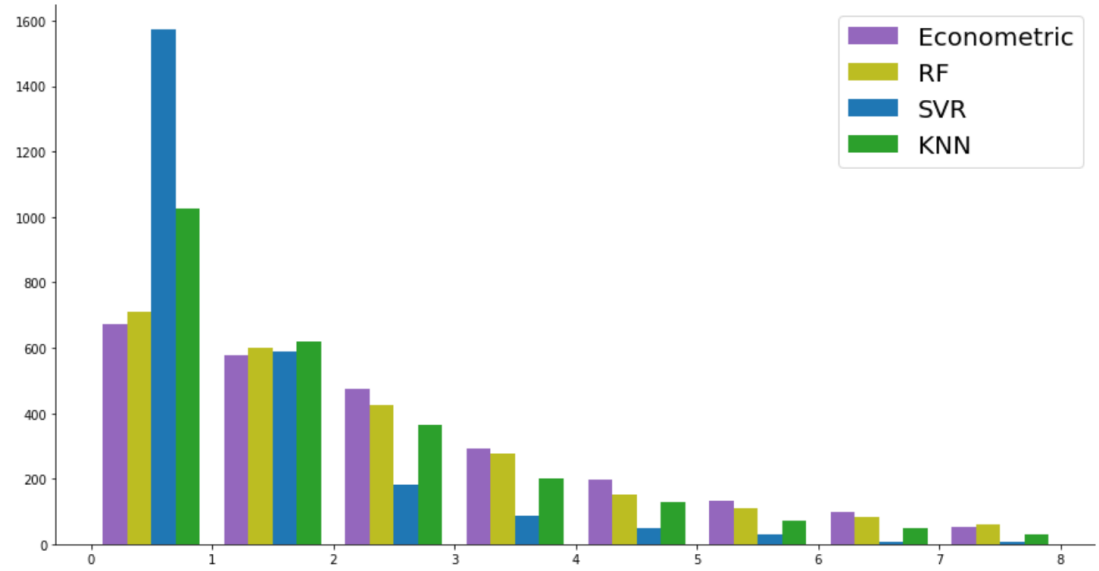

Figure 32. Standardised residuals with sliding window method

KOKKUVÕTE

Aktsia tootluse ja volatiilsuse ennustamine masinõppe ja ökonomeetriliste mudelitega – Balti aktsiaturu võrdlev juhtumianalüüs

Aktsia tootluse ja volatiilsuse suure täpsusega ennustamine on aktsiaturu analüüsi üks huvitavamaid teemasid. Kaks peamist lähenemisviisi aktsiate tootluse prognoosimiseks on ökonomeetrilised mudelid ja masinõpe. Ent antud teema puhul kõige parema lähenemisviisi leidmine ei ole lihtne ülesanne. Töö annab oma panuse kirjandusse, pakkudes detailset analüüsi erinevate ökonomeetriliste ja masinõppe tehnikate kohta NASDAQ Balti börsi näitel.

Analüüsime põhjalikult erinevate masinõppe ja ökonomeetriliste lähenemisviiside ennustustäpsust OMX Baltic Benchmark hinnaindeksi tootluse ja volatiilsuse prognoosimisel. Saime teada, et sobiva metoodika valik sõltub suuresti eelistatud hindamistehnikast ja andmestiku suurusest.

Meie analüüsi tulemused näitavad, et vaadeldavad masinõppe mudelid ületavad üldiselt ARMA lähenemisviisi erinevate treenimis- ja testimisvalimi suuruste ja mõõdikute puhul. Eelkõige võime öelda, et tugivektori regressioonimudel ja k-lähima naabri mudel pakuvad enamikul meie poolt vaadeldud juhtudel paremaid ennustusi, samas kui otsustusmetsa ja tehisnärvivõrgu GARCHi (GARCH-ANNi) jõudlus võrreldes vastavate ökonomeetriliste mudelitega (ARMA ja GARCH) sõltus treenimise ja testimise valimi suurusest ning mõõdikust. Siiski võib GARCHi ja GARCH-ANNi võrdlus anda olenevalt kasutatavatest andmetest erinevaid tulemusi. Eelnevate tööde ja meie analüüsi põhjal soovitab käesolev artikkel sobiva masinõppemudeli valikul pöörata tähelepanu ka ennustusvigu hindavale mõõdikule ning treenimise ja testimise valimi suurustele.