Journal of
Risk and Financial Management

MDPI

Article

Estimating Value-at-Risk in the EURUSD Currency Cross from Implied Volatilities Using Machine Learning Methods and Quantile Regression

Herman Mørkved Blom [1], Petter Eilif de Lange [2,*] and Morten Risstad [3]

[1] Department of Economics, Norwegian University of Science and Technology, Klæbuveien 72, 7030 Trondheim, Norway; hmblom@outlook.com
[2] Department of International Business, Norwegian University of Science and Technology, Klæbuveien 72, 7030 Trondheim, Norway
[3] Department of Industrial Economics and Technology Management, Norwegian University of Science and Technology, Klæbuveien 72, 7030 Trondheim, Norway; morten.risstad@ntnu.no
* Correspondence: petter.e.delange@ntnu.no

Abstract: In this study, we propose a semiparametric, parsimonious value-at-risk forecasting model, based on quantile regression and machine learning methods, combined with readily available market prices of option contracts from the over-the-counter foreign exchange rate interbank market. We aim at improving existing methods for VaR prediction of currency investments using machine learning. We employ two different methods, i.e., ensemble methods and neural networks. Explanatory variables are implied volatilities with plausible economic interpretation. The forward-looking nature of the model, achieved by the application of implied volatilities as risk factors, ensures that new information is rapidly reflected in value-at-risk estimates. To the best of our knowledge, this study is the first to utilize information in the volatility surface, combined with machine learning and quantile regression, for VaR prediction of currency investments. The proposed ensemble models achieve good estimates across all quantiles. The light gradient boosting machine model and the categorical boosting model both yield estimates which are better than, or equal to, those of the benchmark model. In general, neural network models are quite unstable.

Keywords: value-at-risk; over-the-counter foreign exchange (OTC FX) options; quantile regression; machine learning (ML)

Citation: Blom, Herman Mørkved, Petter Eilif de Lange, and Morten Risstad. 2023. Estimating Value-at-Risk in the EURUSD Currency Cross from Implied Volatilities Using Machine Learning Methods and Quantile Regression. *Journal of Risk and Financial Management* 16: 312. https://doi.org/10.3390/jrfm16070312

Academic Editor: Shigeyuki Hamori

Received: 30 April 2023
Revised: 20 June 2023
Accepted: 23 June 2023
Published: 27 June 2023

1. Introduction

The ability to model value-at-risk (VaR) with high accuracy is an important tool for quantifying risk in financial markets (Schaumburg 2012). VaR is an estimate of the loss that will be exceeded with a small probability during a fixed holding period. It measures the worst attainable expected loss over a given time horizon at a given confidence level. The (parametric) VaR measure typically relies on the assumption that the associated portfolio (or investment position) is normally distributed, implicitly assuming normal market conditions. The normal assumption has been criticized in the literature, but does not need to be "too" wrong when portfolios are well diversified. Another limitation with VaR is that if it is used as the objective in an optimization problem, the problem becomes nonconvex, and one must use complicated numerical methods to find VaR efficient portfolios.

In addition to occupying a prominent role in regulatory frameworks, VaR will continue to be important for financial institutions as a measure of market risk. For instance, VaR could be used by banks to compute the amount of assets needed to cover losses.

VaR has been criticized for not explicitly considering tail risk. The expected shortfall (ES) is a risk measure closely related to VaR and is often labeled conditional VaR or tail risk. The VaR metric assigns a 100% weighting to the Qth quantile and zero to other quantiles.

The expected shortfall, however, gives equal weight to all quantiles greater than the Qth quantile and zero weight to all quantiles below the Qth quantile. In certain situations, the ES gives traders in financial markets better incentives to control risk than those of the VaR measure. However, we have elected to employ the VaR measure since, typically, traders are constrained from constructing portfolios with excessive tail risk by detailed mandates of which the VaR metric is but one important component. Some useful studies that have employed both VaR and the ES are: Chaiboonsri and Wannapan (2021) and Yamai and Yoshiba (2005).

This study utilizes market prices for option contracts on the EURUSD exchange rate quoted in the over-the-counter (OTC) foreign exchange (FX) interbank market. The models we employ use implied volatility metrics, i.e., ATM implied volatilities and risk reversals, to forecast VaR. ATM implied volatility is the risk-neutral expectation of spot rate volatility over the remaining life of the option. Risk reversal reflects the difference in the demand for out-of-the money options at high strikes compared to low strikes. Thus, it can be interpreted as a market-based measure of skewness, the most likely direction of the spot movement over the expiry period.

This study extends the work of de Lange et al. (2022). As noted by the authors, machine learning models might further improve their predictions as such models can handle nonlinearities among explanatory variables. The contribution of this study is employing different machine learning models and techniques to improve VaR predictions of currency investments compared to the benchmark quantile regression implied moments (QR-IM) model. From our literature study on machine learning methods presented in Section 2 below, we discovered that neural networks and ensemble methods have both been successfully used for VaR predictions. Therefore, we elected to employ neural networks and ensemble methods for our VaR predictions. For this purpose, recurrent neural network (RNN) and long short-term memory neural network (LSTM) stand out among the neural network models; both these models appear to yield more accurate forecasting results for time series data compared to the feedforward networks, which, however, are still widely used. Amidst the ensemble methods, the random forest model is the most frequently used. Gradient boosting methods have also been used, but less commonly. Ensemble methods and neural networks are considered to be state-of-the-art models in the machine learning community today. We test gradient boosting methods as well as random forest alongside recurrent neural networks, long short-term memory neural networks, and feedforward networks. The various models are created using different model architectures as well as hyperparameter tuning and model stacking for the ensemble models.

We examine several options for tuning and improving the two concepts of ensemble methods and neural networks. In addition, the training and validation datasets are constructed using the QR-IM, XGBoost, and LGBM models. The QR-IM model was proposed by de Lange et al. (2022) for forecasting value-at-risk in currency markets. All models are trained with the same data and validated on the same out-of-sample period.

To the best of our knowledge, this study is the first to utilize information in the volatility surface, more precisely at-the-money volatility and risk reversals as proxies for higher order moments, combined with machine learning and quantile regression to provide accurate VaR estimates.

Our main findings are:

1. The LightGBM model and the categorical boost model yield more accurate VaR estimates than the benchmark QR-IM model.
2. The ensemble models achieve good estimates across all quantiles.
3. The neural network models are, in general, quite unstable and could benefit from more training data and perhaps a better model architecture.
4. Model stacking and hyperparameter tuning overall improved the model predictions.

The remainder of this paper is organized as follows: Section 2 provides an overview of the literature, Section 3 presents the data, Section 4 describes the methodology, Section 5 presents and discusses the results, and Section 6 states the conclusions.

2. Literature Review

Quantile regression methods are often applied to value-at-risk forecasting. Engle and Manganelli (2004) proposed the CAViaR method. This model could directly estimate quantiles instead of modeling the whole distribution. Taylor (2008) proposed the exponentially weighted quantile regression (EWQR) model for estimating value-at-risk. He found that this model outperformed both the GARCH-based methods and the CAViaR models. Chen and Chen (2002) found that calculating VaR at the Nikkei 225 index using quantile regression outperformed the conventional variance-covariance approach. In addition, Shim et al. (2012) employed semiparametric support vector quantile regression (SSVQR) models to estimate VaR on return data on the S&P 500, NIKEI 225, and KOSPI 200 indices. They found that their models outperformed variance-covariance and linear quantile regression models.

A few studies have applied quantile regression in the context of forecasting volatility or VaR of foreign exchange rates. Taylor (1999) found that a quantile regression approach provided a better fit to multi-period data when forecasting volatility compared to variations of the GARCH(1,1) model. Huang et al. (2011) used quantile regression to forecast foreign exchange rate volatility. Jeon and Taylor (2013) included implied volatility as an additional regressor in CAViaR models and obtained increased precision of FX VaR estimates.

Effective explanatory variables are essential to make accurate forecasts of value-at-risk for currency crosses. Chang et al. (2013) provided an outline of different forecasting objectives using options data, including option-implied individual moments. Barone-Adesi et al. (2019) and Huggenberger et al. (2018) showed how to use options data to compute forward looking VaR and conditional VaR measures. The information content of option combinations, such as risk reversal, was studied in the context or arbitrage-free option pricing and hedging in studies by Bossens et al. (2010) and Sarma et al. (2003). de Lange et al. (2022) forecasted value-at-risk in foreign exchange markets using OTC at-the-money option contracts from the foreign exchange interbank market to model volatility, and risk reversals as a proxy for higher order moments. Their QR-IM model outperformed benchmark models such as GARCH and CAViaR-SAV for VaR forecasts.

Several previous studies have confirmed the forecasting ability of a plain vanilla feedforward neural network over traditional statistical models. However, standard neural networks have limitations. Most notably, these models rely on the assumption of independent data observations, which presents a problem when data points are related through time. In order to overcome this problem, Bijelic and Ouijjane (2019) used a gated recurrent unit type of neural network to produce one-step-ahead volatility forecasts of the EURUSD exchange rate. Their model was outperformed by a GARCH (1,1) model for the VaR 95%. Xu et al. (2016a) recognized that multi-period VaR was a complex nonlinear function of the holding period and the one-step ahead volatility forecast. They employed support vector exponentially weighted quantile regression (SVEWQR), which incorporated the SVQR model as a special case, by considering an exponentially weighted quantile regression via SVM to estimate multi-period VaR. They discovered that their model outperformed several traditional methods including the volatility models, filtered historical simulation, and linear quantile regression, on three stock indices. Further, Heryadi et al. (2021) modeled value-at-risk of foreign exchange rates using tree models, support vector machines, and ensemble models. Another study which applied neural networks for predicting foreign exchange rates was by He et al. (2018).

Neural networks have been used in a vast variety of studies that aimed to model value-at-risk. Petneházi (2021) used convolutional neural networks to forecast value-at-risk. By modifying the algorithm slightly, the convolutional networks could estimate arbitrary quantiles of the distribution, not only the mean, thus allowing the network to be applied to VaR forecasting. Pradeepkumar and Ravi (2017) forecasted financial time series volatility using a developed particle swarm optimization trained quantile regression neural network, named SPOQRNN. They compared their model to three traditional forecasting models including GARCH, multilayer perceptron, general regression neural network, and random forest, and found that the SPOQRNN outperformed the other models.

A few authors have used artificial neural networks (ANNs) for forecasting value-at-risk. Taylor (1999) applied a quantile regression neural network approach to estimate the conditional density of multi-period returns. The model was compared to GARCH-based quantile estimates on daily exchange rates. Xu et al. (2016b) developed a new quantile autoregression neural network (QARNN) model based on an artificial neural network architecture. By optimizing an approximate error function and standard gradient-based optimization algorithms, the QARNN output conditional quantile functions recursively. This allowed the QARNN to explore nonlinearity in a financial time series. Yen et al. (2009) used ANN to forecast VaR on a stock index. The authors stated that the model benefited from adding more exogenous parameters in the estimation process, such as interest rates.

Yan et al. (2015) used long short-term memory neural networks in a quantile regression framework to learn the tail behavior of financial asset returns. Their model captured both the time-varying characteristic and the asymmetrical heavy-tail property of financial time series. The authors combined the sequential neural network with a self-constructed parametric quantile function to represent the conditional distribution of asset returns. Kakade et al. (2022) proposed a hybrid model that combined LSTM and a bidirectional LSTM with GARCH to forecast volatility. The model was evaluated on periods with extreme volatility, i.e., the 2007–2009 global financial crisis and the covid recession of 2020–2021. The proposed model provided significant improvement in the quality and accuracy of VaR forecasts compared to benchmark GARCH models.

Ensemble methods have been used by a number of studies for estimating VaR. Andreani et al. (2022) introduced the use of mixed-frequency variables in a quantile regression framework by merging the quantile regression forest algorithm and a mixed-data-sampling model. The empirical application of the model delivered adequate VaR forecasts, and, in terms of quantile loss, it outperformed popular existing models used for VaR forecasting, such as the GARCH model. Jiang et al. (2017) proposed a hybrid semiparametric quantile regression random forest approach to evaluate value-at-risk. The model was used to explain the nonlinear relationship in multi-period VaR measurement. Görgen et al. (2022) used a generalized random forest (GRF) for predicting value-at-risk for cryptocurrencies. They found that random forest outperformed quantile regression methods, including GARCH-type and CAViaR models, when tailored to conditional quantiles. The authors stated that the adaptive nonlinear form of GRF appeared to capture time variations of volatility and spike behavior in cryptocurrency return especially well, in contrast to more conventional financial econometric methods. Gradient boosting methods were tested by Cai et al. (2020) for modeling VaR. They found the method to be effective at capturing risk.

3. Data

We applied our ensemble and neural network models to empirical data for the EURUSD spot exchange rate, taking implied volatility quotes as input data, and examined the models' predictive properties. Daily exchange rates and implied volatility quotes were sourced from Bloomberg and covered the period from January 2009 to December 2020.

3.1. Spot-Rate Returns

The daily returns are calculated as $r_t = ln\left(\frac{S_t}{S_{t-1}}\right)$, where S_t is the spot exchange rate at time t. In our case, this is the daily return of a dollar measured in euros. Table 1 displays descriptive statistics for daily log-returns from January 2009 to December 2020 and Figure 1 plots the time series correspondingly. The return series exhibit the stylized facts that have been widely documented in the financial economics literature, with unconditional means close to zero, clustering of volatility, and fat-tailed return distributions.

Table 1. Descriptive statistics for daily EURUSD log-returns. Time period, January 2009–September 2020 (source, Bloomberg).

n	**2930**
Mean	−0.0001
Std. dev	0.0053
Skewness	0.0330
Kurtosis	4.7508
Min	−0.0229
Max	0.0295

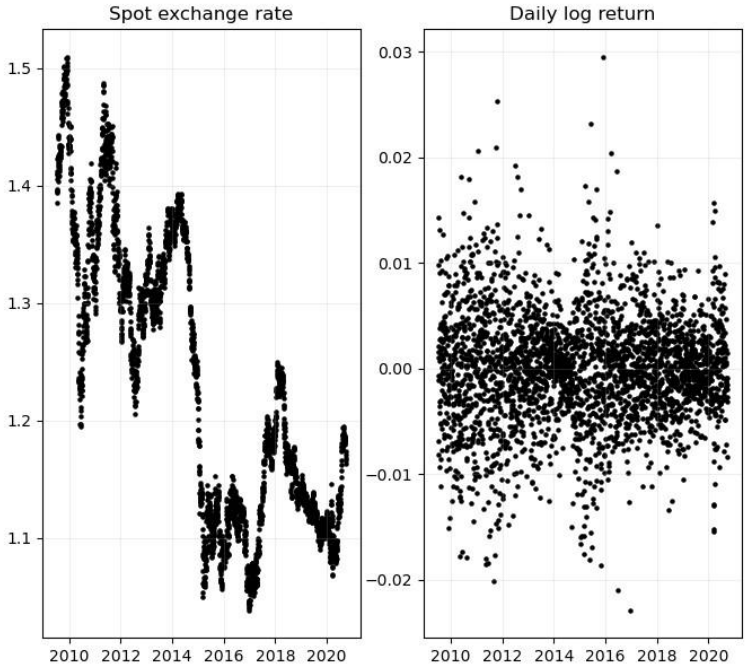

Figure 1. EURUSD spot exchange rates (**left panel**) and daily log-returns (**right panel**). Time period, January 2009–September 2020 (source, Bloomberg).

3.2. Examining the Data

Figure 2 displays time series for at-the-money (ATM) volatilities and 25-delta risk reversals (RRs) for European EURUSD options with one week to expiry and reveals the stochastic nature of the volatility surface. ATM volatility levels spiked around important economic events, such as the Brexit vote in June 2016 and the COVID-19 outbreak in March 2020. Figure 2c shows that the sign of the RR has changed over time and taken both positive and negative values, which in itself is an interesting observation. If the risk reversal reflects the relative probability of depreciation and appreciation of a currency, time-varying sign and the magnitude of the risk reversal can be interpreted as an indication of time varying probability of tail events. Figure 2b,d display empirical distributions, which are skewed and leptokurtic for both variables. ATM volatility is naturally bounded below by zero, but spikes during periods of market turmoil which causes a heavy right tail. The risk reversal displays a highly non-normal, heavy-tailed empirical distribution.

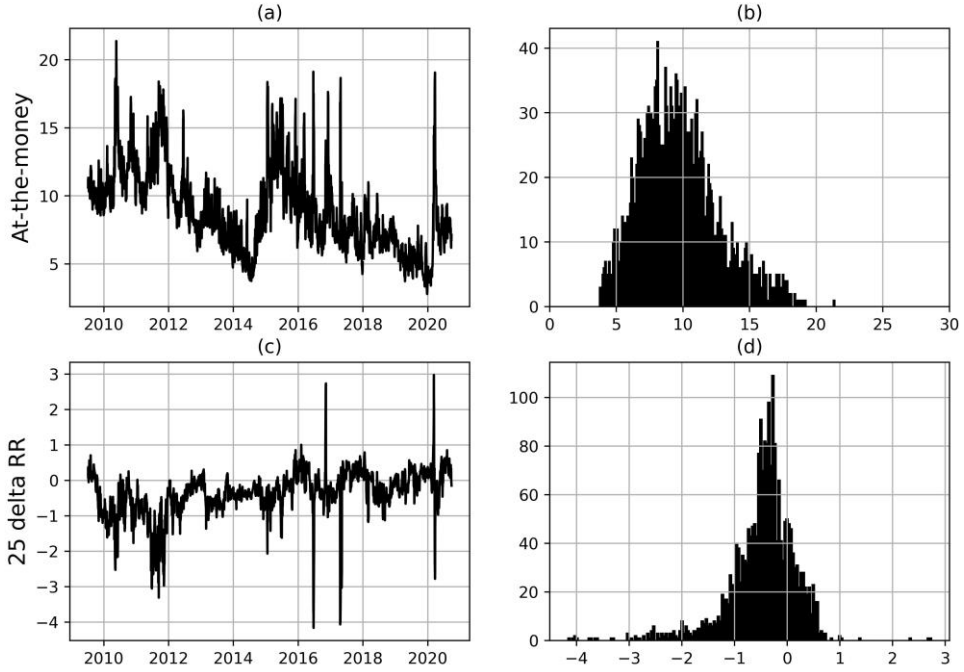

Figure 2. Time series and empirical distributions for implied moments of 1-week to expiry options: (**a**,**b**) At-the-money volatilities; (**c**,**d**) 25-delta risk reversals. Time period, January 2009:01–September 2020 (source, Bloomberg).

3.3. Testing the Dataset

In order to test the data for stationarity, we performed an augmented Dickey–Fuller unit root test and concluded that our data were stationary. Further, to rule out multicollinearity, we ran a variance inflation factor (VIF) test. No sign of multicollinearity was discovered. We also tested our data for heteroskedasticity using the Breusch–Pagan test and serial correlation between the explanatory variables using a Breusch–Godfrey test. The Breusch–Pagan test indicated some sign of heteroskedasticity, whereas no sign of serial correlation was found.

4. Methods

In this section, we provide a brief outline of the models we have used to produce our VaR forecast. We demonstrate how the dataset is generated, as well as briefly and generically explain the QR-IM model, ensemble methods, and neural networks. We also provide a short note on prepossessing of data and the implementation of models. A subsection is devoted to the specific ensemble models and neural networks that we implemented on our data.

4.1. Generating the Training Data

In order to examine the different machine learning methods, we needed to create a proper dataset containing the required quantiles. In this study, we used two methods to create the training dataset. First, we employed the methods proposed by de Lange et al. (2022) using the QR-IM model. Our second approach was to use gradient boosting to generate quantiles based on the generated predictions. This was achieved by using the light gradient boosting machine model and the gradient boosting model; both models have built-in options for quantile regression. The QR-IM is explained below.

The purpose of this study is to investigate the ability of a set of machine learning models to provide accurate VaR estimates out-of-sample. This requires a training dataset from which the ML models can learn the relationship between the conditional return distribution for the Y_q variables and the explanatory variables, the latter being at-the-money volatility and the risk reversal in this study. We applied two different approaches to

generate the conditional return distribution. First, we employed the methods proposed by de Lange et al. (2022). Here, we used the QR-IM model to estimate the return quantiles, conditional on at-the-money volatility and risk reversal values in the training sample. This ensured that the ML methods were trained on a basis directly comparable to the linear QR-IM model. By virtue of this particular approach, we ensured that any relative performance advantage of the ML methods was due to their ability to capture nonlinearities. We refer to this as the QR-IM generated dataset. It is fully conceivable that the optimized ML models contain nonlinear weights and biases for a given minimum MSE, the latter depicted by the linear training set. Indeed, judging from our out-of-sample results, this is a likely explanation for the ML models outperforming the linear QR-IM benchmark model. In addition, there are well known potential problems with linear quantile regression models. One problem is the quantile crossing problem, which implies a non-monotonically increasing cumulative density function. Another problem is that of sensitivity to outliers, which leads to higher variance of estimators and is of particular relevance when estimating conditional return quantiles out-of-sample, as we did in this study. Compared to the linear quantile regression model, the more flexible ML models might very well be more robust with regards to these well-known model risks, if they are present in the training set to any extent.

Second, we relied on gradient boosting; referred to as the gradient boost and LGBM datasets, respectively. The rational for the second method of generating data, i.e., through gradient boosting, is mostly practical, i.e., the light gradient boosting machine model and the gradient boosting model both have built-in options for quantile regression, which make them well suited for generating conditional return distributions.

The Quantile Regression Implied Moments Model

The quantile regression implied moments (QR-IM) model was proposed by de Lange et al. (2022). We employed this model for generating the benchmark in sample quantiles for our currency data. The model is based on the hypothesis that the volatility surface of OTC FX options contains information that can be utilized to improve the accuracy of VaR estimates. Similar to de Lange et al. (2022), we studied data for at-the-money (ATM) options and risk reversals (RR).

At-the-money options are struck at the FX forward rate. ATM options have an initial delta of 50%. The ATM implied volatility is the risk-neutral expectation of spot rate volatility over the remaining life of the option.

Risk reversals involve the simultaneous sale of a put option and purchase of a call option. The two options are struck at the same delta. At the outset, a 25% delta risk reversal will have a combined delta of 50% and very little sensitivity to gamma and vega because of the offsetting effect of the long and short position. Risk reversals are usually quoted as the difference in implied volatility of similar call and put options with the equal delta. The risk reversal reflects the difference in the demand for out-of-the money options at high strikes compared to low strikes. Thus, it can be interpreted as a market-based measure of skewness, the most likely direction of the spot movement over the expiry period.

In the general case, the simple linear quantile regression model is given by:

$$Y_q = \alpha + \beta X + \epsilon_q \tag{1}$$

where Y_q is the qth quantile of the random variable Y, X are regressors, and the distribution of ϵ_q is left unspecified. The expression for the conditional q quantile, $0 < q < 1$, is defined as any solution to the minimization problem:

$$\underset{\alpha,\ \beta}{min} \sum_{t=1}^{T} \left(q - I_{Y_t \leq \alpha + \beta X_t}\right)\left(Y_t - (\alpha + \beta X_t)\right) \tag{2}$$

where t denotes time, and

$$I_{Y_{t \leq \alpha + \beta X_t}} = \begin{cases} 1 \; if \; Y_t \leq \alpha + \beta X_t \\ 0 \; otherwise \end{cases}. \tag{3}$$

In the QR-IM model, the conditional quantile function can be expressed as:

$$\widehat{VaR_{q,t+1}} = \widehat{\alpha_q} + \widehat{\beta_q^{ATM}} ATM_t + \widehat{\beta^{RR}} RR_t + \epsilon_{q,t} \tag{4}$$

A vector of regression parameters $\left[\widehat{\alpha_q}, \widehat{\beta_q^{ATM}}, \widehat{\beta^{RR}}\right]$ can be obtained for each quantile of interest, and the whole return distribution can be found, given observed values for the at-the-money volatility (ATM) and the risk reversal (RR).

4.2. Value-at-Risk

VaR is a statistical risk measure of potential losses and summarizes, in a single number, the worst loss over a target horizon that will not be exceeded with a given level of confidence. Formally, the VaR at a level α of the profit and loss distribution X is defined as:

$$\text{VaR}_\alpha = min\{m : P(L \leq m) \geq 1 - \alpha\} \tag{5}$$

The most common methods for calculating VaR are usually divided into parametric, semiparametric, and nonparametric approaches.

4.3. Ensemble Learning

In this study, we employed the following two base methods for estimating VaR: ensemble learning and neural networks. Ensemble learning is the process of using multiple models, often called weak learners, trained over the same data and combined to obtain better results. Weak learners, or base models, are models that, on average, perform slightly better than random chance. This is either because they have a high bias or because they have too much variance to be robust. The idea of ensemble methods is to attempt reducing bias and/or variance of such weak learners by combining several of them to create a strong learner that achieves better performance.

Low bias and low variance are two of the most desirable features of a model. There is also a trade-off between degrees of freedom and the variance of a model. Introducing too many degrees of freedom can cause high variance (bias–variance trade off).

When combining weak learners, we need to assure that our choice of weak learners is consistent with the way we aggregate the models. If we choose base models with low bias but high variance, we should employ an aggregating method that tends to reduce variance and vice versa. In general, there are three major types of algorithms that aim at combining weak learners: bagging, boosting, and stacking.

Bagging learns homogeneous weak learners independently and combines them following some type of deterministic averaging process. Boosting, which also often considers homogeneous weak learners, learns them sequentially and combines them following a deterministic strategy. Stacking learns weak learners in parallel and combines them by training a meta-model which outputs a prediction based on the different weak model predictions. In general, both boosting and stacking mainly try to produce strong models with less bias than their components, whereas bagging mainly focuses on getting an ensemble model with less variance than its components.

4.4. Deep Learning

Deep learning refers to machine learning methods using deep neural networks to approximate some unknown function based only on inputs and expected outputs. Below, we briefly explain the concepts of artificial neural networks including some activation

functions relevant to this study. Furthermore, the learning process of the neural network is explained along with optimizers and the concept of batch and epoch.

4.4.1. Artificial Neural Network (ANN)

ANNs, hereafter referred to as neural networks (NNs), recognize underlying relationships in a dataset through a process that mimics the way the human brain operates. NNs are nonlinear functional approximators that can be tuned to approximate an unknown function based on observations and target outputs. NNs are structured in consecutive layers. Each layer takes an input, applies a transformation, and returns an output. The perception, originally described by Rosenblatt (1958), acts as the inspiration for the neuron, the foundational building piece of the layer. There is an associated weight for each element in the input vector to the neuron. The bias of the neuron can optionally be added to the input. Then, the neuron determines an output by adding the inputs and the weights assigned to each input. An arbitrary positive number of neurons, each of which provides an output, can make up a layer in an NN. Input can be a vector of size m and output can be a vector of size n, equal to the number of neurons in the layer. Calculating the output vector involves multiplying the input vector by the weight matrix of the layer:

$$y = xW + b \tag{6}$$

where x is the input vector, y is the output vector, W is a $m \times n$ matrix containing the weights for the n neurons in the layer, and b is the vector of biases.

4.4.2. Activation Functions

As mentioned in Section 4.4.1, neural networks are organized in successive layers; each layer computes an output given an input as described in Equation (6). The output of one layer is then subjected to a nonlinear transformation before being passed as input to the following layer. This is referred to as an activation function. Commonly used activation functions are:

The Logistic Function

The logistic function, also known as the sigmoid activation function, transforms the values into the range [0, 1]:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{7}$$

Tanh

The hyperbolic tangent function, simply referred to as tanh, is similar to the sigmoid activation function, but instead outputs values in the range $[-1, 1]$:

$$\sigma(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{8}$$

ReLU

The rectified linear unit (ReLU) activation function outputs values in the range [0, 1]. It does this by taking the maximum of the input and zero:

$$\sigma(x) = max(0, x) \tag{9}$$

All the above-mentioned activation functions are displayed in Figure 3, in blue, with their derivatives in yellow.

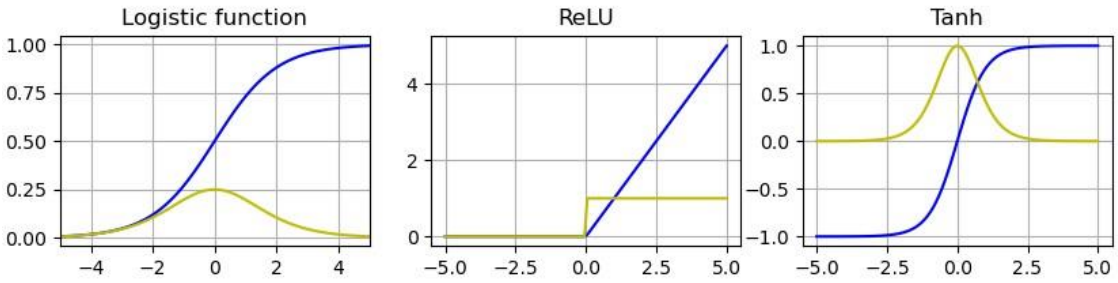

Figure 3. Illustration of activation functions.

The process of adjusting the layer weights until the network satisfactorily approximates the target function is known as training the neural network. The function that the neural network is attempting to approximate is (arbitrarily) denoted by F. An objective function, frequently referred to as the loss function, is used to measure how closely the network approximates F. When the neural network accurately approximates F, the loss function has a global minimum. It calculates the distance between the output $\widehat{y}$ to the target output y. One of the most commonly used loss functions, which was also employed in this study, is the mean squared error (MSE). It is defined as:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \widehat{y}_i)^2$$

4.5. Implementation

We used Keras to construct the neural network, which serves as an interface for the TensorFlow library. TensorFlow was developed by Google and is both a free and open-source library for machine learning. Keras offers both ordinary neural network layers, recurrent layers, and LSTM layers, in addition to a multitude of different machine learning methods.

4.6. The Chosen Ensemble Models

According to the literature review in Section 2, the random forest algorithm has more frequently been employed to estimate VaR compared to the gradient boosting methods. Therefore, in this study, for predicting VaR, we tested different gradient boosting methods as well as the random forest.

Random Forest

The random forest model is an ensemble of many weak learners, in this case decision trees. It can be applied to classification and regression problems. The regression procedure using random forest starts by splitting of features, and then creates decision trees. Every tree makes its individual decision based on the data. The average value of predictions from all trees becomes the final prediction.

Gradient Boosting

Gradient boosting on decision trees is a form of machine learning that works by progressively training more complex models to maximize the accuracy of predictions. Gradient boosting is particularly useful for predictive models that analyze ordered (continuous) data and categorical data. Gradient boosting benefits from training on huge datasets. Gradient boosting is one of the most efficient ways to build ensemble models. The combination of gradient boosting with decision trees provides state-of-the-art results in many applications with structured data.

Extreme Gradient Boosting

Extreme gradient boosting (XGBoost) is an improved version of the gradient boosting algorithm. This algorithm creates decision trees sequentially. Different weights are assigned

to all the independent variables, which are then fed into the decision tree that predicts results. The weight of wrongly predicted variables by the tree is increased and the variables are then fed to the second decision tree. This is known as a greedy algorithm. Then, these individual classifiers/predictors ensemble to give a strong and more precise model. This works for regression, classification, ranking, and user-defined prediction problems.

Light Gradient Boosting Machine

Light gradient boosting machine, known as LightGBM, is a gradient lifting framework which is based on the decision tree algorithm. It can be used in classification, regression, and many more machine learning tasks. Compared to XGBoost it splits leaf-wise and chooses the maximum delta value to grow, rather than level-wise. LightGBM has a significant advantage when performing hyperparameter optimization because the different input parameters are easily controlled.

Category Boosting

Categorical boosting, in short CatBoost, provides a way of performing classifications and rankings of data by using a collection of decision-making mechanisms. The results generated by the learners are weighted and classified based on the strengths and weaknesses of each learner.

4.7. Hyperparameter Architecture

Hyperparameter tuning is a little different for ensemble methods and neural networks. Nevertheless, the general aim is to minimize the loss function and maximize accuracy while reducing bias and overfitting. For both the ensemble methods and the neural networks, choosing the correct architecture is not trivial. When it comes to the ensemble methods, the chosen architecture is mostly based on the method in use, whether this is ordinary random forest or some gradient boost technique. Choosing a method also restricts a lot of the possible tuning. In this study, the process of hyperparameter tuning for the ensemble methods was carried out by using software (if it existed), and trial and error. Typically, the tuning parameters are the number of leaves, number of trees, and the learning rate.

In a neural network there are a lot of possible variations. The hyperparameters that need to be adjusted in the neural network are the number of layers, number of nodes in each layer, the lookback period, activation functions, learning rate, batch size, and number of epochs. This was achieved through an iterative approach, optimizing one parameter at a time.

Another possibility for tuning the models is the training and validation period. The training data consists of data from 7 July 2009 to 29 December 2017. For the neural network, there is also the training period. We split the training period 80/20 percent, into the neural network training set and the test set, respectively. The validation set was equal for both periods and consisted of data from 3 January 2018 to 25 September 2020. Too short a training set can make the data prone to outliers, however it may be less overfitted. Too long a training period makes the data prone to overfitting, and the model may be useless for its actual task because one cannot test the model on enough data to detect anomalies in the model.

A neural network needs many input parameters to find any patterns in the data. In our data, there were only two parameters, which could be too few for a neural network to work properly. Thus, a third parameter was created which was an interaction term between the two already existing parameters.

$$Interaction\ term:\ x_3 = ATM\ volatility \cdot 25\%\ delta\ Risk\ Reversal$$

4.8. The Considered Neural Networks

In this study, we employ the recurrent neural network, the long short-term memory, and the feedforward neural network.

Feedforward Neural Network

The feedforward neural network (FFNN) is an artificial neural network. Connections between the nodes do not form any type of cycle. The information is passed only in one direction through the hidden nodes to the output nodes. Feedforward neural networks with a single hidden layer is the most widely used neural network for forecasting (Zhang et al. 1998).

Recurrent Neural Network

A recurrent neural network (RNN) is a class of artificial neural networks where connections between nodes can create a cycle, allowing output from some nodes to affect subsequent input to the same nodes.

Long Short-Term Memory

The long short-term memory (LSTM) can consolidate information from far in the past with that which is more recent, see Figure 4. A forget gate, an input gate, and an output gate make up the LSTM cell. While the gates govern the information flow in and out of the cell, the fundamental function of the LSTM cell is to recall information over time intervals. The gates essentially decide which information is remembered and which is forgotten. The information that the LSTM cell remembers is kept in its cell state C_t and computed based on the input x_t. The results of the previous output h_{t-1} is kept in the forget gate.

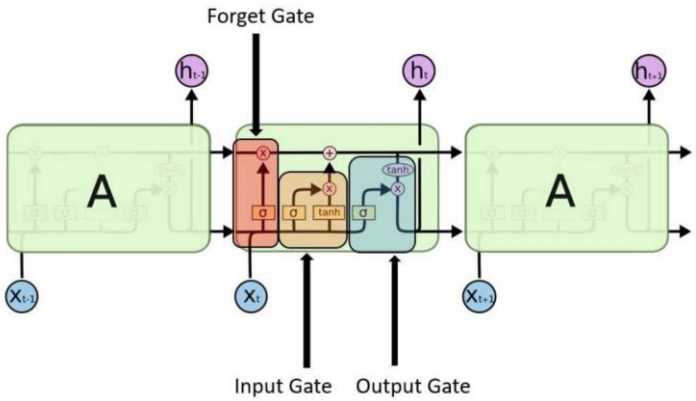

Figure 4. The structure of a long short-term memory (LSTM) layer.

4.9. Explainable AI

We introduce Shapley values to enhance comprehension of how the machine learning models operate. Shapley (SHAP) values originate from game theory and are frequently used for post hoc explainable modeling in machine learning models. The idea is to see the impact each feature has on the target. For further explanation see the original paper by Shapley (1951).

4.10. Developed Neural Network Models

For all models, a batch size of 5 was used. Each model was allowed to run for 120 epochs, but the final number varied based on whether early stopping was activated. All models had their weights randomly initialized and used the Adam optimizer.

4.10.1. Feedforward Neural Network

The feedforward neural network (Figure 5) has three dense layers consisting of 12 nodes, and a dropout frequency of 0.2. After the three dense layers comes a suppress layer, with a dropout frequency of 0.2. This layer suppresses the data and makes the predictions more stable. All layers have a ReLU activation function. The model output are the predicted quantiles: α: [0.01, 0.025, 0.05, 0.95, 0.975, 0.99].

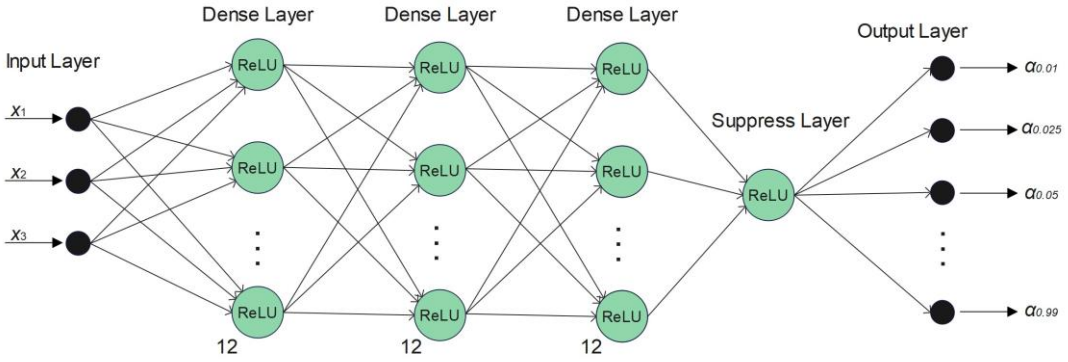

Figure 5. The feedforward neural network used in this study.

4.10.2. Recurrent Neural Network

The recurrent neural network (Figure 6) has two dense layers consisting of 12 nodes and between them a recurrent layer consisting of 8 nodes. All layers have dropout frequencies of 0.2 and a ReLU activation function. The model output is the predicted quantiles: α: [0.01, 0.025, 0.05, 0.95, 0.975, 0.99].

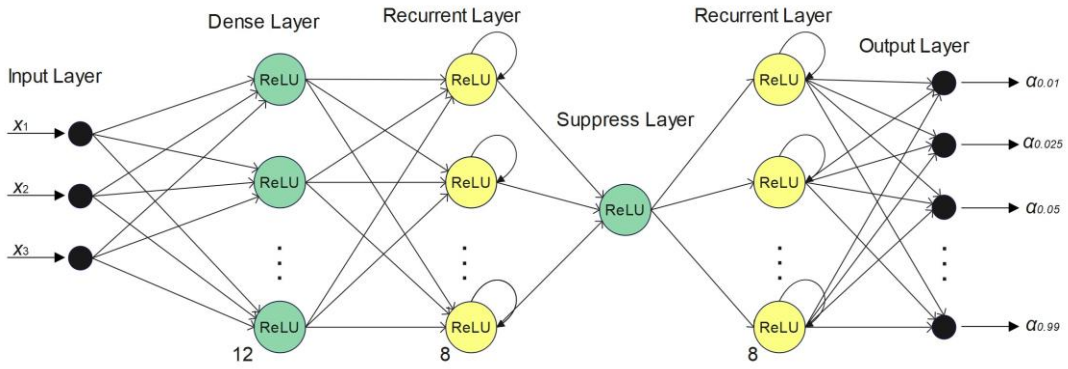

Figure 6. The recurrent neural network used in this study.

4.10.3. Long Short-Term Memory Neural Network

The LSTM neural network (Figure 7) starts with a dense layer of 12 nodes, followed by an LSTM layer with 8 nodes. Then, comes a suppress layer consisting of one node needed to make the model more stable, followed by a LSTM layer with 8 nodes. All layers have dropout frequencies of 0.2 and a ReLU activation function. Outputs from the model are the predicted quantiles: α: [0.01, 0.025, 0.05, 0.95, 0.975, 0.99].

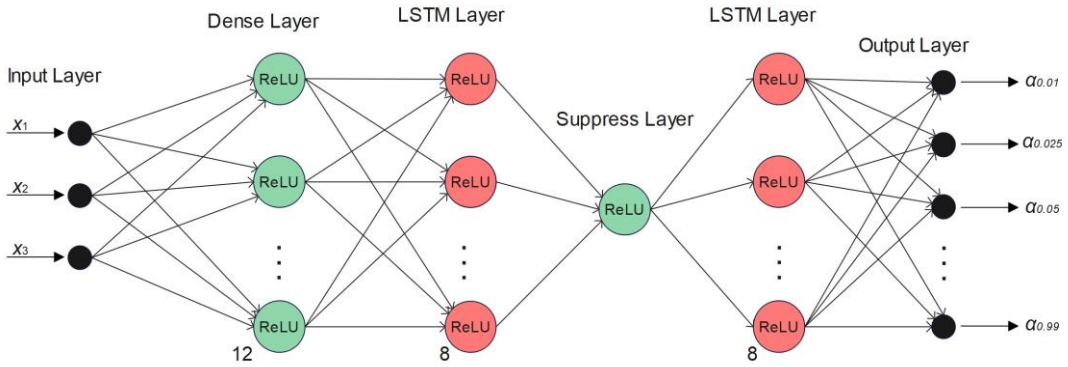

Figure 7. The LSTM neural network used in this study.

4.11. Evaluating the Models

We let $I(A)$ denote the indicator function, i.e., a function which returns 1 if the event A occurs and 0 if not, represented by (11). The realized return is given by R, while the predicted return (VaR estimate) is given by $\widehat{R_q}$. N provides the total number of predictions. We introduce the following metrics with this notation, which are used to examine superiority among the VaR models:

$$I_t(A) = \begin{cases} 1 \; if \; the \; event \; A \; occurs \\ 0 \; if \; A \; does \; not \; occur \end{cases} \tag{10}$$

If the VaR is specified at confidence level α, the breach level should roughly be equivalent to α. The *Breach ratio* is given as:

$$Breach \; Ratio = 100 \cdot \frac{\sum_{t=1}^{T} I(\widehat{R_q} < R_t)}{T} (\%) \tag{11}$$

Calculating the nominal breach of the model over the total number of predictions is one method of evaluating performance. The model with smaller *Sum if breach* is favored given models with comparable breach ratios since this would suggest that the model is at least closer to the true value of the loss. The *Sum if breach* is given as:

$$Sum \; if \; breach = \sum_{t=1}^{T} I(\widehat{R_q} < R_t)(R_t - \widehat{R_q}) \tag{12}$$

It is also interesting to know how close a prediction is to the predicted medium when no breach occurs. Thus, the *Sum if no breach* is given as:

$$Sum \; if \; no \; breach = \sum_{t=1}^{T} I(R_t < \widehat{R_q}) \left(\widehat{R_q} - R_t \right) \tag{13}$$

The formulas are both expressed for the higher quantiles. The same formulas, however, can be used for the lower quantiles. In addition, the following 4 metrics are used for further comparison:

$$Total \; sum \; if \; breach = \sum_{q=\alpha} |Sum \; if \; breach_\alpha|$$

$$Total \; sum \; if \; no \; breach = \sum_{q=\alpha} |Sum \; if \; no \; breach_\alpha|$$

$$Max. \; VaR = \max\left(\widehat{R_1}, \ldots, \widehat{R_T} \right)$$

$$Min. \; VaR = \min\left(\widehat{R_1}, \ldots, \widehat{R_T} \right)$$

5. Results

In this section we present the results. Throughout all estimates, a fixed training window from 1 July 2009 to 31 December 2017 is used, and the out-of-sample period is from 1 January 2018 to 28 September 2020.

5.1. Estimating the Quantile Regression VaR Model

Table 2 displays the estimated quantile regression coefficients for the ATM variables and the 25-delta RR variables. The model is estimated on daily Bloomberg data during the in-sample period (July 2009–December 2017). The coefficients are scaled by 100 for readability. Regarding the sign, magnitude, shape, and statistical significance of regression coefficients, the results are consistent for both the ATM variable and the RR variable.

Table 2. Quantile regression coefficients for the QR-IM model, along with the hit percentage. (i) and (ii) denote statistical significance at the 1% and 5% confidence levels, respectively.

Quantile—α	1.0%	2.5%	5.0%	95.0%	97.5%	99.0%
Constant	−0.61 (i)	−0.21	−0.07	0.09	0.05	0.12
ATM	−0.06 (i)	−0.09 (i)	−0.08 (i)	0.09 (i)	0.12 (i)	0.15 (i)
25-delta RR	0.21 (i)	0.16 (i)	0.10 (ii)	0.16 (i)	0.30 (i)	0.38 (i)
Breach Ratio	1.04	2.48	4.97	94.99	97.43	99.01

The breach ratios of the estimated quantiles of the in-sample data are also shown in Table 2. The breach ratio displays good coverage throughout the data with respect to the estimated quantiles, as would be expected as the coefficients are statistically significant.

The estimated ATM coefficients support the view that the ATM is an indicator of overall market risk. The upper panel of Figure 8 depicts a nonlinear, increasing relationship between the quantiles and the ATM regression coefficients. The coefficients project the shape of the tails of the conditional return distribution. It can be observed that the coefficients are negative for the lower quantiles and positive for the higher quantiles. In addition, the absolute value of ATM coefficients is slightly higher in the right tail compared to the left tail. This indicates a nonlinear relationship between implied volatility and returns.

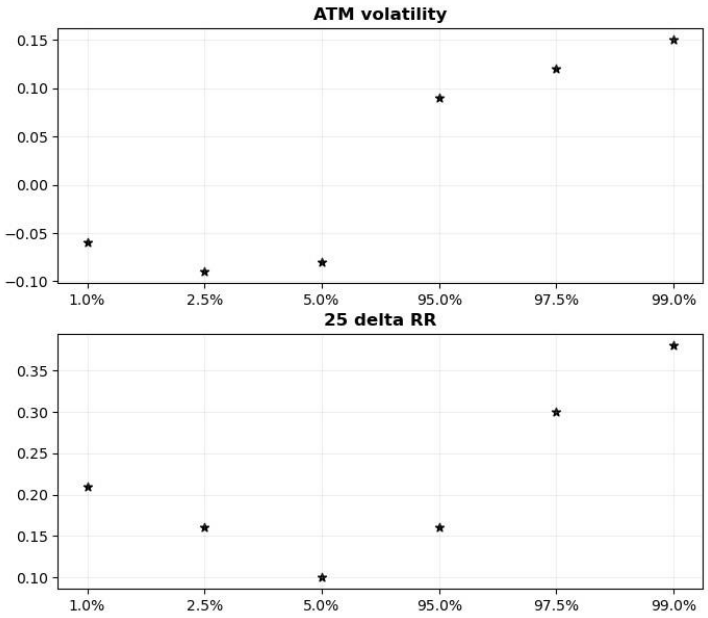

Figure 8. Quantile regression coefficients for the regressors in the QR-IM model across tail quantiles. At-the-money (ATM) volatility (**top panel**) and 25-delta risk reversal (**lower panel**). In-sample period. Coefficient values on the y-axes, quantiles along the x-axis.

The lower panel of Figure 8 reveals that the RR coefficients form a U-shaped pattern and that all are positive. This indicates that implied volatility is more important when estimating both ends of the tails. However, since all coefficients are statistically significant, all quantile estimates of the return distribution will improve from employing implied volatility.

5.2. General Scheme for Evaluating the Machine Learning Models

We tested different uses of the algorithms for ensemble models: random forest, extreme gradient boosting, light gradient boosting, and categorical boosting.

The data were split in two for training and validation. The training set started at the beginning of the sample period and lasted until the end of 2017. The validation period started in 2018 and last until the end of the dataset. This part was used for validation and

testing the performance of the model. Figure 9 illustrates the concept. To illustrate the scheme in full detail, we present the results for the random forest model in the next section.

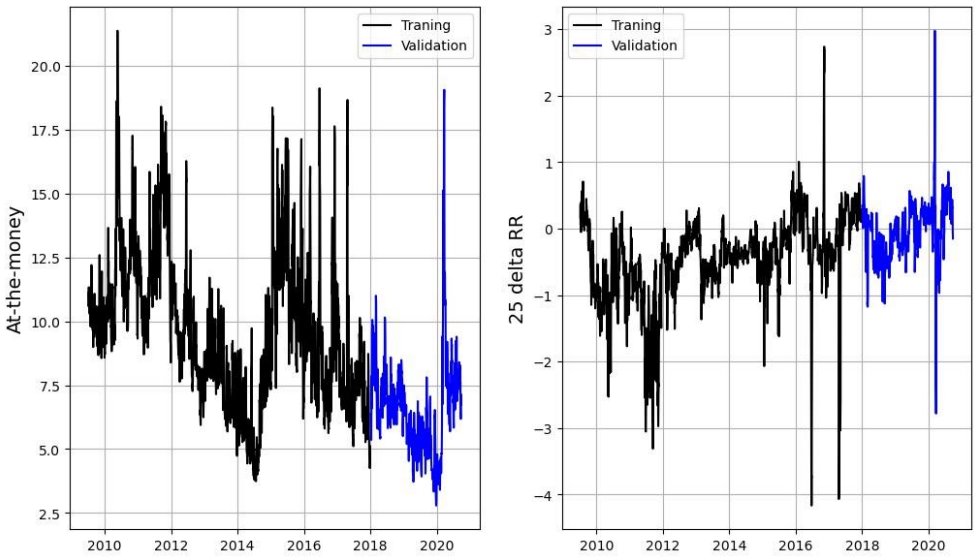

Figure 9. Time series dynamics for at-the-money volatility (**left panel**) and risk reversal (**right panel**) during the training period and the validation period.

Illustrating the Scheme: Random Forest

In this section, we describe how the random forest algorithm was tested. The data used in the training originated from the data generated by the QR-IM model in Equation (4), discussed in the previous section. The model was trained individually on each quantile. The in-sample data for RR and ATM volatility, along with the in sample estimated quantiles in Section 5.1, were used for training the model. For testing, only the sampling data for RR and ATM volatility were used. The results are listed in Table 3.

Table 3. Random forest performance, out-of-sample period.

Quantile—α	**1.0%**	**2.5%**	**5.0%**	**95.0%**	**97.5%**	**99.0%**
Breach Ratio	0.7	2.1	5.75	95.09	97.62	99.44
Sum if breach	−0.005	−0.018	−0.054	0.067	0.022	0.005
Sum if no breach	−7.421	−5.641	−4.483	4.729	5.982	7.650
Min. Var	−0.023	−0.022	−0.019	0.004	0.005	0.006
Max. VaR	−0.008	−0.005	−0.004	0.016	0.023	0.029

The first thing to notice is the increased performance for estimates of higher-order quantiles; 95%, 97.5%, and 99%. Second, the sum if no breach increases further away from the mean. This makes sense as these estimates should be further away from the sampling data compared to quantiles closer to the mean. Third, as expected, the sum if breach decreases towards the tails of the distribution. Fourth, the sum if breach is almost equal to zero at the quantiles in the tails. This suggests that the model yields promising results when predicting spike behavior.

The graph shown in Figure 10 displays all the point estimates for the different quantiles, produced by the random forest algorithm. One thing to notice is how closely the predictions move in line with the underlying data. This gives a good indication of how well the model works with the EURUSD data.

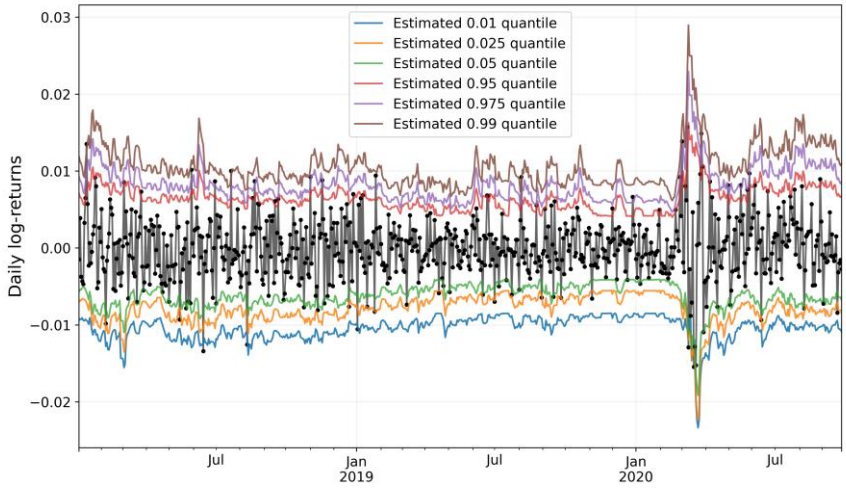

Figure 10. Out-of-sample VaR estimates from the random forest model. Realized daily log-returns represented by black dots.

Figure 11 shows the distribution of the SHAP values calculated from the random forest ensemble method. The plot indicates symmetry in the estimation of the quantiles. Further, risk reversal seems to be the more significant model parameter, as it displays both the greatest density and magnitude in the figure. In this model, high values for the risk reversal provide a positive contribution to the prediction, and low values provide a negative contribution. Excluding outliers in the implied volatility, the high/low values have the same contribution as seen for the risk reversal.

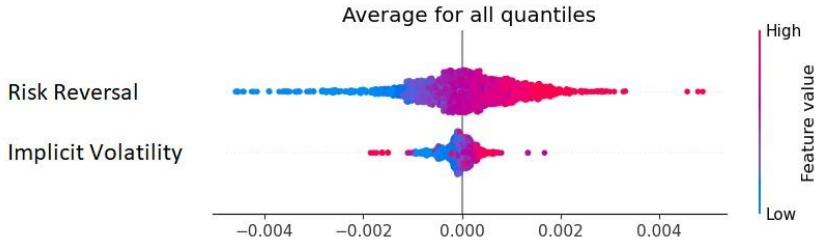

Figure 11. Variable importance for the random forest model, expressed as Shapley values. The figure shows that high values of explanatory variables increase VaR estimates. Furthermore, the risk reversal has significant explanatory power for the tails of the return distribution.

Basically, we followed the scheme outlined above for evaluating all our machine learning models. In addition to the random forest model, we further tested two XGBoost models, three LightGBM models, one categorical boost (CatBoost) model, and three neural networks. We also created a stacking model, combining the LGBM 2 model with the CatBoost model. Stacking (similar to hyperparameter tuning) is a tool used for improving model performance.

The difference in performance of the two XGBoost models (and the three LightGBM models) stems from the difference in the datasets constructed to train them (Table 4). We also tested three different neural networks: a feedforward neural network (FFNN), a recurrent neural network (RNN) and a long-short term memory neural network (LTSM). The different datasets used for training the model are summarized in Table 4.

To save space, we have not reported details from examining all models such as we did with the random forest model above. Instead, we provide a summary comparing the performance of the different models to the QR-IM model, which we have chosen as the benchmark for this study.

Table 4. The different datasets used for training the models. (*) A new movement variable was constructed, which captures the interaction between the ATM variable and the RR variable.

Model	Dataset Used for Training
XGBoost (1)	QR-IM generated dataset
XGBoost (2)	Gradient boost generated dataset
LGBM (1)	QR-IM generated dataset
LGBM (2)	QR-IM generated dataset with new movement variable (*)
LGBM (3)	LGBM generated dataset
Categorical Boosting	QR-IM generated dataset
FFNN	QR-IM generated dataset with new movement variable (*)
RNN	QR-IM generated dataset with new movement variable (*)
LTSM	QR-IM generated dataset with new movement variable (*)

However, first, a brief note on the neural networks:

For training and validation of the neural networks, we split the data in three parts. The first part starts at the beginning of the dataset and lasts until the end of 2017. This part is used for training and testing the model. From the first part, 80% of the data is used for training and the remaining for testing. The second part starts in 2018 and lasts until the end of the dataset, and this part is used for the model validation. All three periods are illustrated in Figure 12.

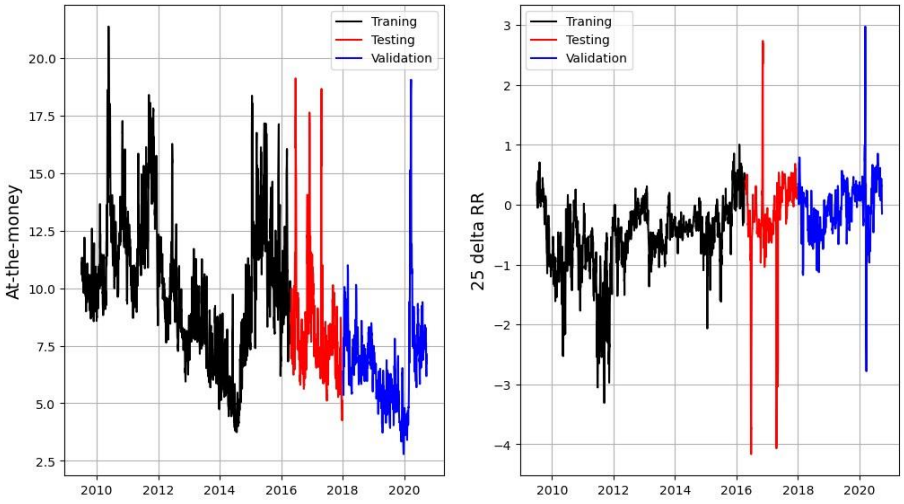

Figure 12. Time series dynamics for at-the-money volatility (**left panel**) and risk reversal (**right panel**) during training, testing, and validation period for the neural networks (FFNN, RNN, and LSTM).

5.3. Summary of Results: Comparing the ML Models to the QR-IM Model

We validate the out-of-sample performance of the different ML methods against the QR-IM model developed by de Lange et al. (2022).

In general, the performance of the ensemble methods is more stable and consistent than that of the neural networks. The ensemble methods handle the periods with outliers better than the neural networks, whose predictions tend to explode around these events.

The ensemble models perform well compared to the baseline QR-IM model. Three models stand out compared to the baseline model: LightGBM 2, CatBoost, and the stacking model between the two. These models all perform better or equal to that of the QR-IM model on the lower quantiles, i.e., 1%, 2.5%, and 5%. At the higher quantiles, i.e., 95%, 97.5%, and 99%, the QR-IM model is better or equal to the three models.

The neural network models are, in general, quite unstable and are probably not particularly well suited for this task. The models produce worse or equal quantile predictions compared to the QR-IM model for all quantiles.

All models are summarized with their breach ratios for the out-of-sample data in Table 5.

Table 5. Summary of all model breach ratios during the out-of-sample period.

Quantile—α	**1.0%**	**2.5%**	**5.0%**	**95.0%**	**97.5%**	**99.0%**
Baseline model			Breach Ratio			
QR-IM	0.7	2.1	6.3	95	97.5	99.3
Ensemble methods						
Random Forest	0.7	2.1	5.75	95.09	97.62	99.44
XGB (1)	0.7	2.1	5.47	95.09	97.9	99.44
XGB (2)	1.26	3.65	7.01	95.37	98.46	99.16
LightGBM (1)	0.55	1.82	5.33	95.09	97.62	99.44
LightGBM (2)	0.7	2.1	5.05	95.23	97.76	99.44
LightGBM (3)	1.4	4.49	8.42	95.65	96.63	99.3
CatBoost	0.7	2.23	5.47	94.95	97.62	99.44
Stacking Models						
LGBM2 & CatBoost	0.7	2.18	5.31	95.05	97.67	99.44
Neural network						
FFNN	2.38	4.35	8.27	90.74	96.21	98.74
RNN	2.42	4.27	7.68	90.75	96.02	98.72
LTSM	3.13	4.97	8.68	96.16	96.16	98.44

We note that, for the sum if breach parameter, less is better. By having a small sum, the model is better at predicting the outcome when a breach happens. However, if a small sum if breach value is combined with a high value for sum if no breach, then the model may be too conservative in its predictions. In Table 6, both the if breach (i.b.) and if no breach (i.n.b.) sums are listed for all models. It is difficult to distinguish between the models and find a superior model. The random forest, XGB model 1, and CatBoost algorithms are all very similar when comparing the i.b. scores, all having an approximate score of 0.17. However, the random forest is significantly better when comparing the i.n.b scores, having the lowest score of 35.91. Thus, the random forest may be considered to be superior amongst the three. It is more difficult to distinguish between the LBGM 1 and the LGBM 2, each having its one superior trait. Thus, the choice of model depends on which feature, i.e., sum if breach or sum if no breach, one deems to be more important. The LBGM model 1 has the lowest i.b. score (0.164) among all models.

In Table 7, all the models are summarized with their Christoffersen test p-values and DQ test p-values with four lags. For further comparison, we have reported results for a set of benchmark models frequently employed in the literature. GARCH models can accommodate a wide range of assumptions with regards to the distribution of residuals. Filtered historical simulation (FHS) was introduced by Barone-Adesi et al. (2008). Here, the conditional distribution of residuals is derived from the empirical distribution of standardized returns. McNeil and Frey (2000) suggested fitting a GARCH model to the time series or returns, and then applying EVT to the standardized residuals. We refer to these approaches as FHS-GARCH and EVT-GARCH, respectively. The CAViaR methodology from Engle and Manganelli (2004) models VaR as an autoregressive process. We apply the symmetric absolute value specification (CAViaR-SAV), in which the estimated VaR responds symmetrically to the absolute value of realized returns. Furthermore, we compute VaR estimates from the GJR-GARCH model from Glosten et al. (1993), which allows for the asymmetric response of conditional volatility to negative and positive returns through the leverage parameter.

Table 6. Breach values, i.b., and if not breach values, i.n.b., out-of-sample period.

Quantile—α		1.0%	2.5%	5.0%	95.0%	97.5%	99.0%	|Sum|
Ensemble methods								
Random Forest	i.b.	−0.005	−0.018	−0.054	0.067	0.022	0.005	0.171
	i.n.b	−7.421	−5.641	−4.483	4.729	5.982	7.650	35.91
XGB (1)	i.b.	−0.006	−0.019	−0.052	0.064	0.023	0.005	0.169
	i.n.b.	−7.442	−5.680	−4.521	4.756	6.028	7.643	36.07
XGB (2)	i.b.	−0.025	−0.056	−0.184	0.112	0.040	0.020	0.437
	i.n.b.	−5.860	−5.152	−3.612	4.155	5.477	6.391	30.65
LightGBM (1)	i.b.	−0.004	−0.017	−0.053	0.066	0.022	0.004	0.166
	i.n.b.	−7.445	−5.668	−4.500	4.748	6.005	7.691	36.06
LightGBM (2)	i.b.	−0.005	−0.019	−0.055	0.060	0.020	0.005	0.164
	i.n.b.	−7.445	−5.682	−4.526	4.816	6.046	7.712	36.23
LightGBM (3)	i.b.	−0.020	−0.054	−0.104	0.067	0.033	0.010	0.288
	i.n.b.	−6.255	−4.860	−4.110	4.929	5.657	7.164	32.97
CatBoost	i.b.	−0.006	−0.020	−0.056	0.064	0.022	0.005	0.173
Neural network	i.n.b.	−7.411	−5.641	−4.496	4.789	6.017	7.688	36.04
FFNN	i.b.	−0.052	−0.098	−0.188	0.186	0.086	0.038	0.648
	i.n.b	−9.477	−8.139	−6.752	6.420	7.917	9.873	48.57
RNN	i.b.	−0.069	−0.157	−0.208	0.201	0.088	0.047	0.770
	i.n.b	−9.120	−8.486	−6.287	6.274	8.406	10.064	47.00
LTSM	i.b.	−0.052	−0.095	−0.200	0.213	0.110	0.046	0.716
	i.n.b	−9.303	−8.114	−6.529	6.132	7.346	9.242	46.66

Table 7. The *p*-values for the Christoffersen test and DQ test (with four lags), out-of-sample. High values indicate high accuracy.

Christoffersen Test (*p*-Value)	1.0%	2.5%	5.0%	95.0%	97.5%	99.0%
Random forest	0.398	0.489	0.364	0.364	0.846	0.200
XGB (1)	0.398	0.489	0.564	0.917	0.489	0.200
XGB (2)	0.175	0.000	0.000	0.000	0.157	0.001
LightGBM (1)	0.200	0.343	0.810	0.917	0.846	0.200
LightGBM (2)	0.398	0.489	0.945	0.781	0.660	0.200
LightGBM (3)	0.306	0.002	0.000	0.419	0.157	0.398
CatBoost	0.398	0.660	0.564	0.945	0.846	0.200
FHS-GARCH	0.210	0.000	0.380	0.310	0.380	0.670
EVT-GARCH	0.430	0.000	0.380	0.030	0.380	0.670
CAViaR-SAV	0.070	0.000	0.200	0.170	0.000	0.930
GJR-GARCH (Student *t*)	0.430	0.000	0.500	0.090	0.380	0.670
DQ Test (*p*-Value)	**1.0%**	**2.5%**	**5.0%**	**95.0%**	**97.5%**	**99.0%**
Random forest	0.622	0.929	0.191	0.727	0.255	0.965
XGB (1)	0.806	0.944	0.192	0.280	0.858	0.965
XGB (2)	0.128	0.000	0.000	0.000	0.084	0.001
LightGBM (1)	0.673	0.938	0.319	0.648	0.717	0.964
LightGBM (2)	0.617	0.942	0.491	0.381	0.201	0.963
LightGBM (3)	0.000	0.001	0.000	0.951	0.506	0.991
CatBoost	0.692	0.943	0.371	0.678	0.204	0.964
FHS-GARCH	0.820	0.000	0.690	0.620	0.710	0.000
EVT-GARCH	0.900	0.000	0.700	0.230	0.700	0.000
CAViaR-SAV	0.710	0.000	0.700	0.230	0.700	0.000
GJR-GARCH (Student *t*)	0.880	0.000	0.930	0.410	0.710	0.000

The XGB model 2 and LightGBM model 3 both reject the null hypotheses. Thus, the observed number of VaR breeches is significantly different from the expected number of breeches for the two models. The remaining ML models, on the one hand, generally display high p-values, indicating high forecasting performance across the quantiles. The conventional benchmark models, on the other hand, cannot deliver consistent results.

6. Conclusions

In this study, we employ ensemble methods and neural networks to forecast value-at-risk for daily exchange rates. Our aim is to improve VaR predictions of currency investments compared to a benchmark quantile regression implied moments (QR-IM) model. The ensemble methods, which are forward looking and utilize directly observable option prices as explanatory variables, show great potential at predicting the daily exchange rate. The neural network, though also forward looking and utilizing directly observable option prices as explanatory variables, does not perform at a desired prediction level.

We emphasize six key outtakes from our study:

- The ensemble methods are better at predicting the spike behavior of the EURUSD than the neural networks. The ensemble methods are well suited for predicting the daily EURUSD currency cross distribution, including its VaR.
- The second LightGBM model, categorical boost, and the stacking model between the two, stand out in terms of comparing the breach values to the base line QR-IM model. These models all perform better or equal to the latter at the lower quantiles and have poorer or equal performance at the higher quantiles.
- The random forest and LightGBM 1 and 2 have the best performances among the stand-alone models in terms of i.b. and i.n.b. values.
- Neural networks can improve a lot compared to the ensemble methods. One way of doing this might be to introduce more layers and more nodes. However, by doing so, the models become more of a black box.
- The advantage of neural networks compared to ensemble methods is the way they estimate the quantiles. On the one hand, neural networks are constructed such that all quantiles can be estimated simultaneously. On the other hand, ensemble methods forecast one quantile at a time. For huge datasets, the possibility of computing all quantiles simultaneously can significantly improve computational time.
- Model stacking and hyperparameter tuning significantly improved the models in terms of the overall performance of the breach ratio.

Future Research

Several possible directions can be taken to further examine the models explored in this study. The models' performances should be tested in less efficient markets than the EURUSD. Including more explanatory variables, such as macroeconomic variables, might improve the models. In addition, one could try different types of combinations of layers in the neural networks. This should lead to better forecasts and more stable results, at the cost of transparency of these networks. Lastly, ensemble methods might be improved by further exploring hyperparameter tuning and stacking methods.

Author Contributions: Suggesting research problem, P.E.d.L. and M.R.; methodology, H.M.B., P.E.d.L. and M.R.; software, H.M.B.; validation, H.M.B.; formal analysis, H.M.B.; data curation, H.M.B. and M.R.; writing—original draft preparation, H.M.B. and P.E.d.L.; writing—review and editing, H.M.B., P.E.d.L. and M.R. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: All data used for this study are publicly available. The authors retrieved the data from Bloomberg.

Acknowledgments: The authors want to thank three anonymous reviewers for valuable comments to our manuscript.

Conflicts of Interest: The authors declare no conflict of interest.

References

Andreani, Mila, Vincenzo Candila, and Lea Petrella. 2022. Quantile Regression Forest for Value-at-Risk Forecasting via Mixed-Frequency Data. Paper presented at 4th International Conference on Information and Communications Technology (ICOIACT), Virtual, August 30–31; Cham: Springer International Publishing. ISBN 978-3-030-99638-3.

Barone-Adesi, Giovanni, Chiara Legnazzi, and Carlo Sala. 2019. Option-implied risk measures: An empirical examination on the S&P 500 index. *International Journal of Finance & Economics* 24: 1409–28. [CrossRef]

Barone-Adesi, Giovanni, Robert F. Engle, and Loriano Mancini. 2008. A GARCH option pricing model with filtered historical simulation. *Review of Financial Studies* 21: 1223–58. [CrossRef]

Bijelic, Anna, and Tilila Ouijjane. 2019. Predicting Exchange Rate Value-at-Risk and Expected Shortfall: A Neural Network Approach. Available online: https://lup.lub.lu.se/student-papers/search/publication/8989138 (accessed on 22 June 2023).

Bossens, Frédéric, Grégory Rayée, Nikos S. Skantzos, and Griselda Deelstra. 2010. Vanna-volga methods applied to fx derivatives: From theory to market practice. *International Journal of Theoretical and Applied Finance* 13: 1293–324. [CrossRef]

Cai, Xiaoting, Yang Yang, and Guangxin Jiang. 2020. Online risk measure estimation via natural gradient boosting. Paper presented at 2020 Winter Simulation Conference, Orlando, FL, USA, December 14–18.

Chaiboonsri, Chukiat, and Satawat Wannapan. 2021. Applying Quantum Mechanics for Extreme Value Prediction of VaR and ES in the ASEAN Stock Exchange. *Economies* 9: 13. [CrossRef]

Chang, Bo Young, Peter Christoffersen, and Kris Jacobs. 2013. Market skewness risk and the cross section of stock returns. *Journal of Financial Economics* 107: 46–48. [CrossRef]

Chen, Mei-Yuan, and Jau-Er Chen. 2002. Application of quantile regression to estimation of value at risk. *Review of Financial Risk Management* 1: 15. Available online: https://www.jcic.org.tw/upload/download/7e0d92a1-1d70-48a7-8d21-fee5317e9ce3.pdf (accessed on 22 June 2023).

de Lange, Petter E., Morten Risstad, and Sjur Westgaard. 2022. Estimating value-at-risk using quantile regression and implied moments. *The Journal of Risk Model Validation* 16: 53–76.

Engle, Robert F., and Simone Manganelli. 2004. Caviar: Conditional autoregressive value at risk by regression quantiles. *Journal of Business & Economic Statistics* 22: 367–81.

Glosten, Lawrence R., Ravi Jagannathan, and David E. Runkle. 1993. On the relation between the expected value and the volatility of the nominal excess return on stocks. *Journal of Finance* 48: 1779–801. [CrossRef]

Görgen, Konstantin, Jonas Meirer, and Melanie Schienle. 2022. Predicting Value at Risk for Cryptocurrencies with Generalized Random Forests. Available online: https://ssrn.com/abstract=4053537 (accessed on 22 June 2023). [CrossRef]

He, Kaijian, Lei Ji, Geoffrey K. F. Tso, Bangzhu Zhu, and Yingchao Zou. 2018. Forecasting exchange rate value at risk using deep belief network ensemble-based approach. *Procedia Computer Science* 139: 25–32. [CrossRef]

Heryadi, Yaya, Antoni Wibowo, Sudimanto, and Lucas. 2021. Foreign exchange prediction using machine learning approach: A pilot study. Paper presented at International Conference on Information and Communications Technology, Yogyakarta, Indonesia, August 30–31.

Huang, Alex YiHou, Sheng-Pen Peng, Fangjhy Li, and Ching-Jie Ke. 2011. Volatility forecasting of exchange rate by quantile regression. *International Review of Economics & Finance* 20: 591–606.

Huggenberger, Markus, Chu Zhang, and Ti Zhou. 2018. Forward-Looking Tail Risk Measures. SSRN Papers. Available online: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=2909808 (accessed on 22 June 2023).

Jeon, Jooyoung, and James W. Taylor. 2013. Using CAViaR Models with Implied Volatility for Value-at-Risk Estimation. *Journal of Forecasting* 32: 62–74. [CrossRef]

Jiang, Feng, Wenjun Wu, and Zijun Peng. 2017. A semi-parametric quantile regression random forest approach for evaluating muti-period value at risk. Paper presented at 2017 36th Chinese Control Conference (CCC), Dalian, China, July 26–28; pp. 5642–46. Available online: http://resolver.scholarsportal.info/resolve/19341768/v2017inone/5642_asqrrffemvar.xml (accessed on 22 June 2023).

Kakade, Kshitij, Ishan Jain, and Aswini Kumar Mishra. 2022. Value-at-risk forecasting: A hybrid ensemble learning garch-lstm based approach. *Resources Policy* 78: 102903. [CrossRef]

McNeil, Alexander J., and Rüdiger Frey. 2000. Estimation of tail-related risk measures for heteroscedastic financial time series: An extreme value approach. *Journal of Empirical Finance* 7: 271–300. [CrossRef]

Petneházi, Gábor. 2021. Quantile convolutional neural networks for value at risk forecasting. 2021. *Machine Learning with Applications* 6: 100096. [CrossRef]

Pradeepkumar, Dadabada, and Vadlamani Ravi. 2017. Forecasting financial time series volatility using particle swarm optimization trained quantile regression neural network. *Applied Soft Computing* 58: 35–52. [CrossRef]

Rosenblatt, Frank. 1958. The Perceptron—A Perceiving and Recognizing Automaton. Available online: https://blogs.umass.edu/brain-wars/files/2016/03/rosenblatt-1957.pdf (accessed on 22 June 2023).

Sarma, Mandira, Susan Thomas, and Ajay Shah. 2003. Selection of value-at-risk models. *Journal of Forecasting* 22: 337–58. [CrossRef]

Schaumburg, Julia. 2012. Predicting extreme value at risk: Nonparametric quantile regression with refinements from extreme value theory. *Computational Statistics and Data Analysis* 56: 4081–96. [CrossRef]

Shapley, Lloyd S. 1951. Notes on the N-Person Game—II: The Value of an N-Person Game. Available online: https://www.rand.org/content/dam/rand/pubs/research_memoranda/2008/RM670.pdf (accessed on 22 June 2023).

Shim, Jooyong, Yongtae Kim, Jangtaek Lee, and Changha Hwang. 2012. Estimating value at risk with semiparametric support vector quantile regression. *Computational Statistics* 27: 685–700. [CrossRef]

Taylor, James W. 1999. A quantile regression approach to estimating the distribution of multiperiod returns. *Journal of Derivatives* 7: 64–78. [CrossRef]

Taylor, James W. 2008. Using exponentially weighted quantile regression to estimate value at risk and expected shortfall. *Journal of financial Econometrics* 6: 382–406. [CrossRef]

Xu, Qifa, Cuixia Jiang, and Yaoyao He. 2016a. An exponentially weighted quantile regression via SVM with application to estimating multiperiod VaR. *Statistical Methods & Applications* 25: 285–320.

Xu, Qifa, Xi Liu, Cuixia Jiang, and Keming Yu. 2016b. Quantile autoregression neural network model with applications to evaluating value at risk. *Applied Soft Computing* 49: 1–12. [CrossRef]

Yamai, Yasuhiro, and Toshinao Yoshiba. 2005. Value at risk versus Expected Shortfall: A practical perspective. *Journal of Banking and Finance* 29: 997–1015. Available online: https://www.sciencedirect.com/science/article/abs/pii/S0378426604001499 (accessed on 22 June 2023). [CrossRef]

Yan, Xing, Weizhong Zhang, Lin Ma, Wei Liu, and Qi Wu. 2015. Parsimonious quantile regression of financial asset tail dynamics via sequential learning. *Advances in Neural Information Processing Systems*. Available online: https://proceedings.neurips.cc/paper_files/paper/2018/file/9e3cfc48eccf81a0d57663e129aef3cb-Paper.pdf (accessed on 22 June 2023).

Yen, Jerome, Xiaoliang Chen, and Kin Keung Lai. 2009. A statistical neural network approach for value-at-risk analysis. Paper presented at International Joint Conference on Computational Sciences and Optimization, Sanya, China, April 24–26. Available online: https://www.researchgate.net/profile/Kin-Keung-Lai/publication/221187237_A_Statistical_Neural_Network_Approach_for_Value-at-Risk_Analysis/links/5be7d2d192851c6b27b5ffdf/A-Statistical-Neural-Network-Approach-for-Value-at-Risk-Analysis.pdf (accessed on 22 June 2023).

Zhang, Guoqiang, B. Eddy Patuwo, and Michael Y. Hu. 1998. Forecasting with artificial neural networks: The state of the art. *International Journal of Forecasting* 14: 35–62. [CrossRef]