GARCH Type and Machine Learning Models for Volatility Forecasting

Alessandro Ciniltani

Financial Econometrics - January 2023

Abstract

The objective of this study is to investigate and implement two distinct approaches for forecasting volatility. The first approach involves utilizing GARCH-type models from the traditional literature of financial econometrics, while the second approach employs Machine Learning models, specifically Neural Networks. The data utilized for this study was obtained from Yahoo Finance, and includes a 10-year daily dataset of the NASDAQ 100 (NDX).

The results of this study suggest that Neural Networks are able to achieve superior performance when compared to Generalized Autoregressive Conditional Heteroskedasticity models. However, it should be noted that this improved performance is accompanied by a reduction in interpretability and explainability of the predicted values.

1 Introduction

The aim of this study is to evaluate and compare various models for time series forecasting, with a specific focus on volatility forecasting. Volatility is a crucial variable in financial markets, as it is a key indicator of market behavior and is essential for managing risk, creating investment strategies, assessing the performance of financial assets, and pricing financial instruments such as options.
To accomplish this aim, two main approaches for volatility prediction will be examined in this paper. The first approach involves utilizing standard Time Series Modeling techniques, specifically GARCH (Generalized Autoregressive Conditional Heteroskedasticity) type models. The second approach employs Machine Learning models, specifically Neural Networks, specifically LSTM Networks.
The data used for this study was obtained from Yahoo Finance, and includes a 10-year daily dataset of the NASDAQ 100 (NDX). The results of this study suggest that while LSTM networks are able to achieve superior performance when compared to GARCH models, this improved performance comes at the cost of reduced interpretability and explainability of the predicted values.
The next sections of the paper will firstly introduce and describe the data used, followed by an overview of the GARCH type models used. The fourth section will explain the Machine Learning techniques used and their behavior. The results will be then analyzed and discussed, and finally, a summary and an intuition about possible practical implementations will be provided.

2 Exploratory Data Analysis

2.1 Data description

The data used in this study is a time series of the NASDAQ 100 Index ("NDX"), which was obtained from Yahoo Finance. The dataset includes a 10-year time series (from January 1st, 2013) of daily open, high, low, and close prices, volume, dividends, and stock splits, resulting in a total of 2485 observations.
Given that the time series of closing prices is non-stationary, daily returns and daily log returns were calculated. The Augmented Dickey-Fuller test was applied to both series and stationarity was confirmed. The distribution of returns and log returns was investigated and is presented in Figure 1. The results indicate that both series do not conform to the standard normal distribution, exhibiting positive skewness and higher kurtosis.

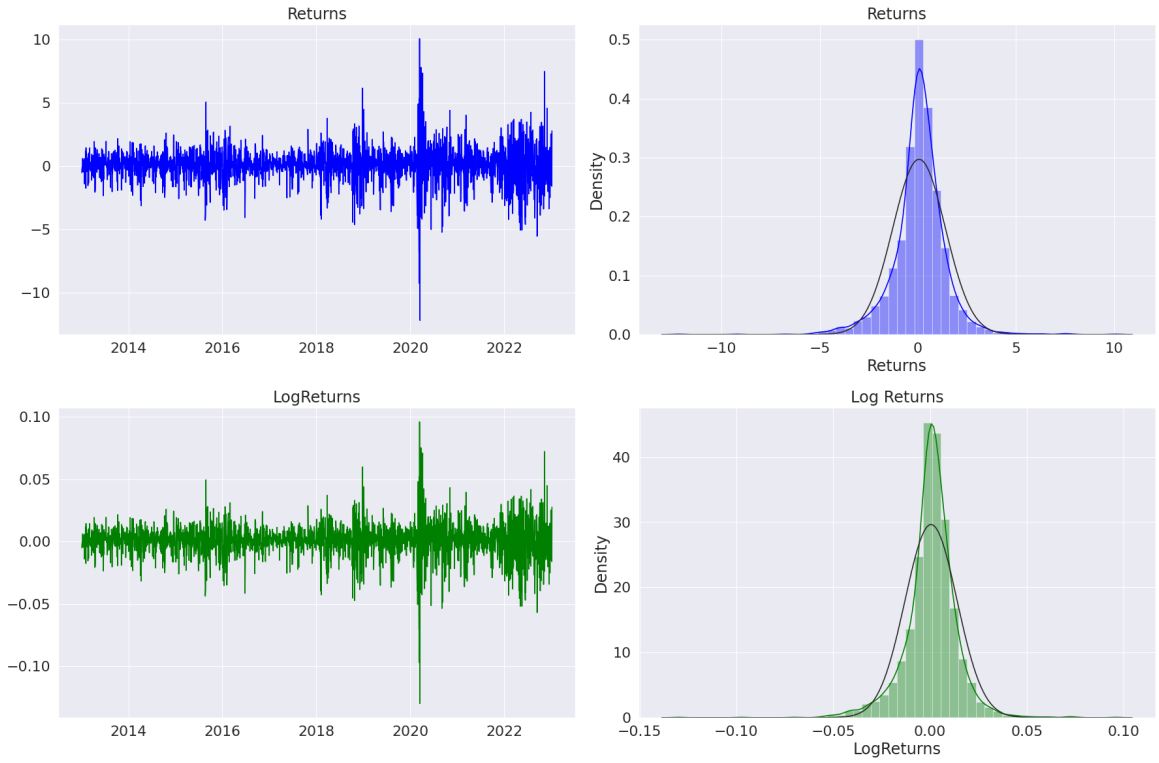

Figure 1

Before proceeding with any computational or statistical analysis, it is essential to evaluate the Autocorrelation Function (ACF) and Partial Autocorrelation Function (PACF) as a proxy for volatility. This step is crucial in order to determine the appropriate modeling and implementation strategies.
As illustrated in Figure 2, the series of daily log returns exhibits almost no significant autocorrelation, whereas in the same series, when calculated in absolute value, a high level of persistent autocorrelation is detected. This suggests that a time series approach may be appropriate for modeling volatility.
Based on this understanding, realized volatility was computed as follows:

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n}(r_i - \bar{r})^2}{n}}$$

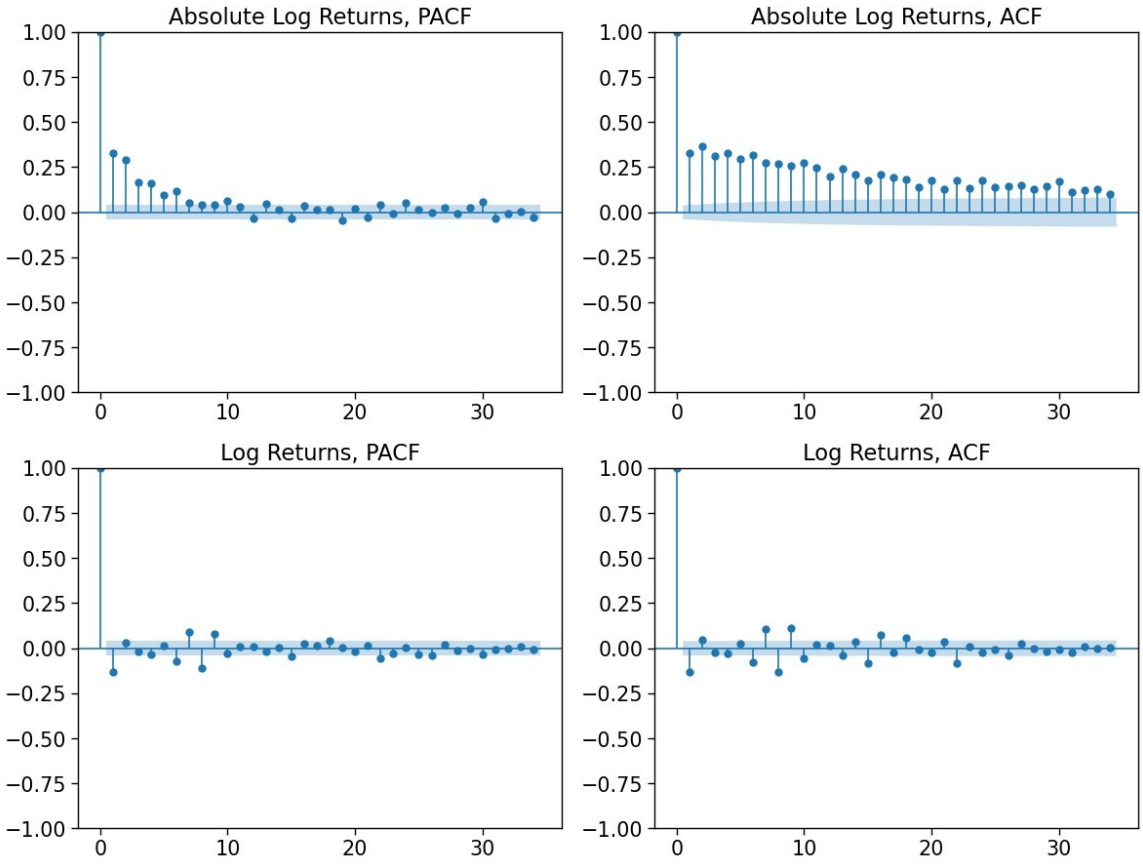
Absolute Log Returns, PACF
Absolute Log Returns, ACF
Log Returns, PACF
Log Returns, ACF
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
10
20
30

Figure 2

Where the interval, n, could be daily, monthly, or semiannual (1, 30, 180 respectively). The results are presented in Figure 3. For the purpose of this study, the target variable for the models will be the 30-day realized volatility, as it is less noisy than the daily volatility and reverts to the mean more slowly than the semiannual volatility.

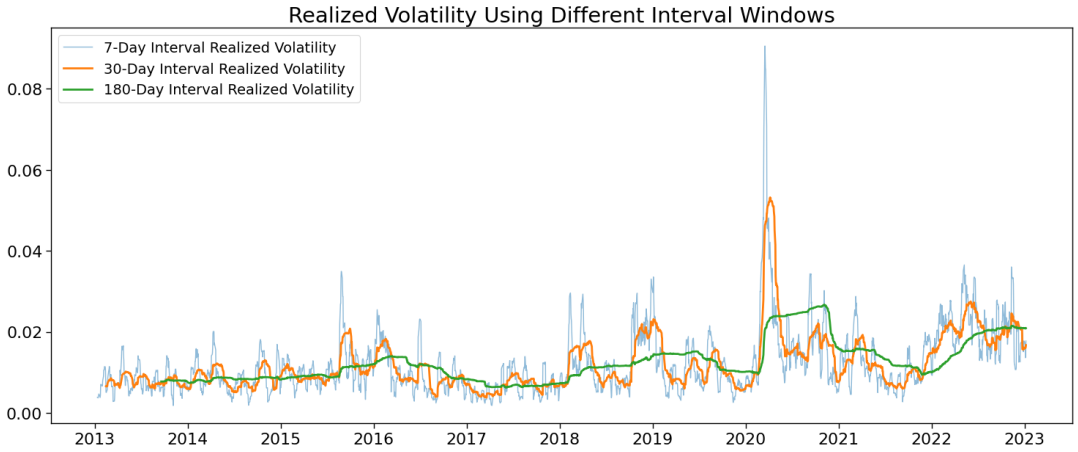

Figure 3

In order to implement two different types of models, the data were normalized to a range between 0 and 1. This was done to ensure that the models were not affected by any scaling or normalization differences in the data.
To conclude this section, the final representation of the time series for the 30-day realized volatility is presented. The distinction between the training, testing and validation datasets is clearly indicated.

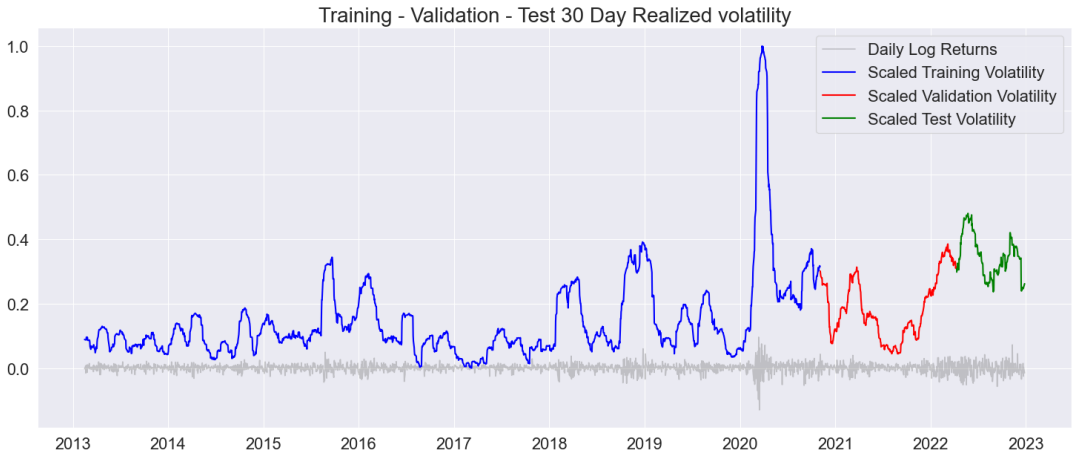

Figure 4

The train-validation-test split is a widely used method for dividing the dataset into three subsets for training, hyperparameter tuning, and testing the model. This ensures that the model's performance is evaluated on unseen data, preventing overfitting, and providing a more robust assessment of the model's performance.

3 Methodologies

In this section, the techniques used in the study will be discussed in detail. Specifically, GARCH-type models and Machine Learning models will be addressed. The methodology employed in this study is based on a logical progression, starting with a classic $GARCH(1,1)$ model, and then building upon the evidence obtained from the parameters to construct more complex models, such as a $GJR-GARCH$ model and two $TARCH$ models.
In the second part of this section, machine learning techniques will be discussed, beginning with a plain Neural Network, followed by Univariate LSTM (Long Short-Term Memory) models, and finally, Multivariate LSTM Neural Networks will be implemented.

3.1 GARCH-Type Models

GARCH-type models, introduced by Bollerslev (reference), are widely used and easy to implement in financial time series analysis. These models, which are extensions of the ARCH model, are used to describe and predict volatility for processes that exhibit volatility clustering. A GARCH model specifies the conditional variance of a time series as a function of both its past values and past residuals. This characteristic enables the model to capture both long-term and short-term dependencies in the data, making it a powerful tool for modeling volatility in financial time series data.

3.1.1 GARCH(1,1)

In the GARCH model class, the GARCH(1,1) model is considered to be the simplest and most straightforward. The GARCH(1,1) model calculates the conditional variance as shown in Equation 1:

$$\sigma_t^2 = \omega + \alpha\epsilon_{t-1}^2 + \beta\sigma_{t-1}^2 \tag{1}$$

Two parameters, α and β, must be estimated in order to use the model. α represents the weight given to past residuals, while β represents the weight given to past conditional variances. These parameters can be estimated using maximum likelihood methods, allowing the model to be fit to historical data and used to make forecasts.
One of the key advantages of the GARCH(1,1) model is its simplicity. However, the model can also be generalized to the GARCH(p,q) specification as shown in Equation 2:

$$\sigma_t^2 = \omega + \sum_{i=1}^{p} \alpha_i\epsilon_{t-i}^2 + \sum_{j=1}^{q} \beta_j\sigma_{t-j}^2 \tag{2}$$

However, as the specification becomes more complex, the increase in prediction ability may not always justify the increased mathematical complexity.
In this study, the GARCH(1,1) model was primarily used for inferential purposes and to decide on the next steps to take.

3.1.2 Threshold GARCH Model and GJR-GARCH

The second model implemented in this study is the Threshold GARCH (TARCH) model, which aims to incorporate asymmetric behavior in volatility, a commonly observed pattern in practice. The specific form of the TARCH model implemented in this study is the TARCH(1,1) model, which is specified as shown in Equation 3:

$$\sigma_t^2 = \omega + \alpha|r_{t-1}| + I_{t-1}^{-}\gamma|r_{t-1}| + \beta\sigma_{t-1} \tag{3}$$

Where: σ_t^2 is the conditional variance at time t; $\omega, \alpha, \gamma, \beta$ are model parameters; r_{t-1} is the residual at time t-1; σ_{t-1}^2 is the conditional variance at time t-1; I is the indicator function, which is equal to 1 if the residual at time t-i is negative.

In practice, these models make different predictions depending on whether the residual at the considered timestamp was negative or positive. If the estimated parameter γ is positive, it can be observed that negative returns are followed by higher volatility than when positive returns are observed.

A particular case of the TARCH model is the GJR-GARCH model (Glosten, Jaganathan, and Runkle, 1993). The main difference between the GJR-GARCH and TARCH models is that the GJR-GARCH model uses r^2 instead of the absolute value of r in the equation. The GJR-GARCH model is specified as shown in Equation 4:

$$\sigma_t^2 = \omega + \sum_{i=1}^{p}(\alpha_i r_{t-i}^2 + I_{t-i}^{-}\gamma_1 r_{t-i}^2) + \sum_{j=1}^{q}\beta_j\sigma_{t-j}^2 \tag{4}$$

As in the case of the TARCH model, the simplest implementation, the GJR-GARCH(1,1) has been chosen. Moreover, hyperparameter tuning was implemented for the TARCH model. All possible combinations with $1 < p < 3$, $0 < q < 3$ and $0 < o < 3$ have been fitted to the data and used for forecasting. The performance of these models was then evaluated using the Bayesian Information Criterion (BIC) and Root Mean Squared Percentage Error (RMSPE). The RMSPE was given priority in the selection process. From the grid search, it was determined that the best implementation was the TARCH(1,2).

3.2 Machine Learning Models

The machine learning models implemented in this study include Neural Networks (NNs). NNs are inspired by the design and function of the human brain, and are categorized as deep learning models. They were first developed in the 1950s and gained popularity after 2005. NNs have been applied to a wide range of tasks, including hand-written recognition and image recognition, and have also found applications in finance.

Each type of NN consists of an input layer and an output layer, with one or more intermediate layers between them. These intermediate layers receive input from the nodes of the previous layer, perform mathematical computations, and transmit information to the next layer.

In this project, NNs were used as supervised learning techniques, where both input and output are provided and used to train the model. Two types of models were implemented:

1. Fully Connected Neural Networks
2. Long Short-Term Memory (LSTM)

The choice of these particular models is based on the idea that they are the most known and used in this field, and they are expected to be suitable for volatility forecasting.

3.2.1 Fully Connected NN

In this study, a Fully Connected Neural Network model was implemented, which serves as a simple linear regression model. The architecture of the model consists of a single dense layer with a single output unit

and no activation function. The dense layer receives the input and performs linear combination of the inputs with the learned weights, and the output unit produces the prediction. This architecture is simple and can be used to model linear relationships between the input and output variables.
A visual intuition of its design is given below:

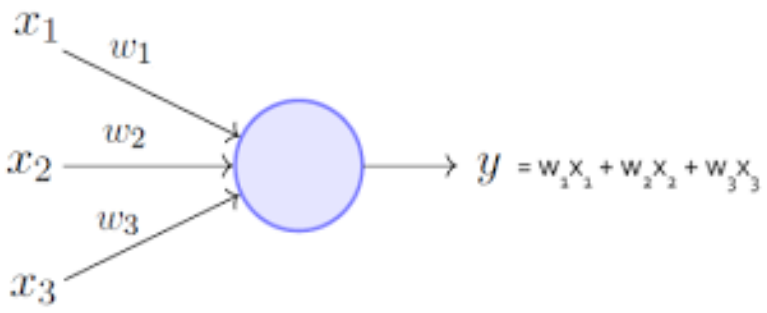

Figure 5: Design of a Fully Connected Neural Network

3.2.2 Long Short Term Memory

The second type of Neural Network model implemented in this study is the Long Short-Term Memory (LSTM) model. LSTM is a type of Recurrent Neural Network (RNN) which, at each step, utilizes both the inputs and outputs of the previous step in order to make a more accurate prediction. At the initial step, the input is processed and generates an output, which is then processed alongside new inputs. This process is repeated recurrently for each iteration.
One of the main challenges of RNNs is their effectiveness decreases as the amount of available information increases over time. To address this issue, an internal state known as a "cell" is introduced into the network. This cell is divided into three parts: a "Forget Gate", "Input Gate" and an "Output Gate". These gates are used to selectively forget, store and output information from the previous time steps, allowing the LSTM to maintain a long-term memory of the inputs and make more accurate predictions.

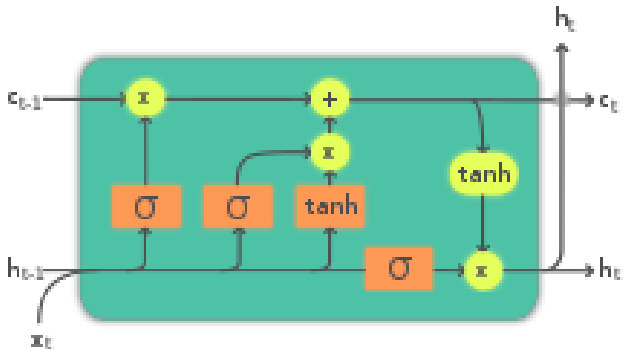

Figure 6: Design of a LSTM Neural Network

This design is particularly useful for time series problems, as not all past information may be useful for a one-period-ahead forecast.
This peculiar design is very useful for time series problems, since not all the past information could be useful for the 1-period ahead forecast.

In addition to the standard LSTM, bidirectional LSTMs (Figure 7 are also employed in time series analysis to enhance model performance and efficiency. A bidirectional LSTM trains two neural networks if all realizations for each timestamp are available. The first network is trained on the original dataset and the second network is trained on a reversed copy of the provided time series.
This allows the model to take into account both past and future temporal dependencies in the data, leading to improved performance.

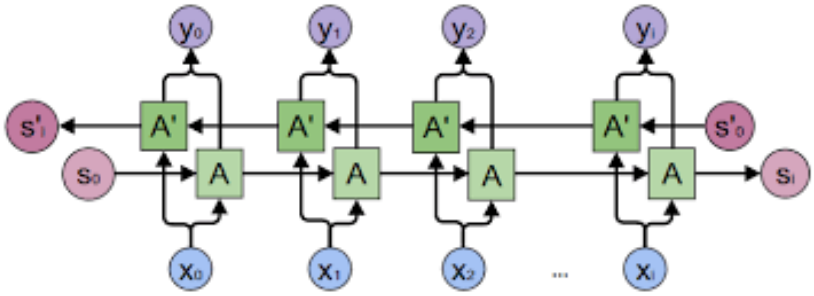

Figure 7: Design of a Bidirectional LSTM Neural Network

An extension to the LSTM model is the Multivariate LSTM. To implement the LSTM in a multivariate framework, several metrics were computed for each day using the available data. In particular, the following metrics were computed:

1. High - Low Spread
2. Open - Close Spread
3. Logarithm of the Volumes traded

This multivariate approach allows for the inclusion of multiple factors, such as the spread between high and low prices, the spread between opening and closing prices, and the volume traded, in the model, this is done with the aim of improving predictive ability of the model.
Within the Neural Network framework, four different models were implemented.

3.3 Metrics

In this paragraph evaluation metrics are discussed.
Two main performance criteria have been used:

- Root Mean Squared Error (RMSE)
- Root Mean Squared Percentage Error (RMSPE)

Which are computed as follows:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{5}$$

$$RMSPE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\frac{(y_i - \hat{y}_i)^2}{y_i}} \tag{6}$$

The RMSE represents the square root of the second sample moment of the differences between predicted values and observed values or the quadratic mean of these differences. It is non-negative and the closest it is to zero, the better the model's performance.
The RMSPE is pretty similar to the RMSE, the only difference is that each deviation is divided by the overall mean.

4 Results

In this section, the results for each model are presented along with their specifications. First, the GARCH type models are discussed, followed by a discussion of the Neural Network models. Finally, a comparison of the models is provided.
To evaluate the models, the 30-day Realized Volatility was computed using the forecasted values from the model and compared with the validation window of the hold-out dataset. This allows for a fair comparison of the models and their ability to accurately forecast volatility.

4.1 GARCH Type Models

The forecasts for the GARCH type models were generated using a rolling one-step forecasting approach. All data points until the end of the training sample were used to fit the model. After fitting, a one-step ahead forecast was computed as the average of the 7-day ahead predicted values. The model was then refitted, and the process was repeated until the end of the validation period. All data were normalized to fall within the range of 0 and 1 for consistency and comparability.

4.1.1 GARCH(1,1)

The first and simplest model implemented is the GARCH(1,1) model, which serves as a starting point for further improvement. A visual representation of the Out-of-Sample forecast is provided in Figure 8.

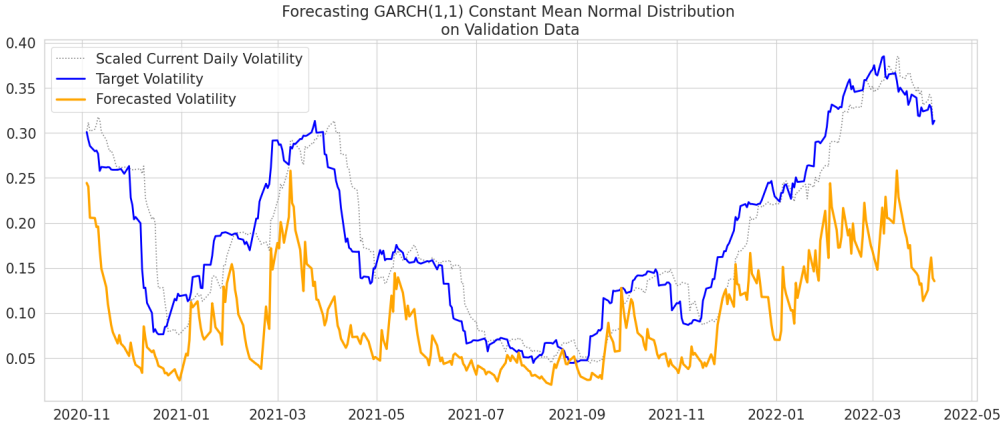

Figure 8

The GARCH(1,1) model achieved a Validation RMSPE of 0.503 and a Validation RMSE of 0.1054. To determine if a better model could be constructed, the residuals were analyzed. As can be seen in Figures 9 and 10, the residuals of the GARCH(1,1) process do not come from a standard normal distribution. As a result, asymmetric models were implemented to better capture the distribution of the residuals.

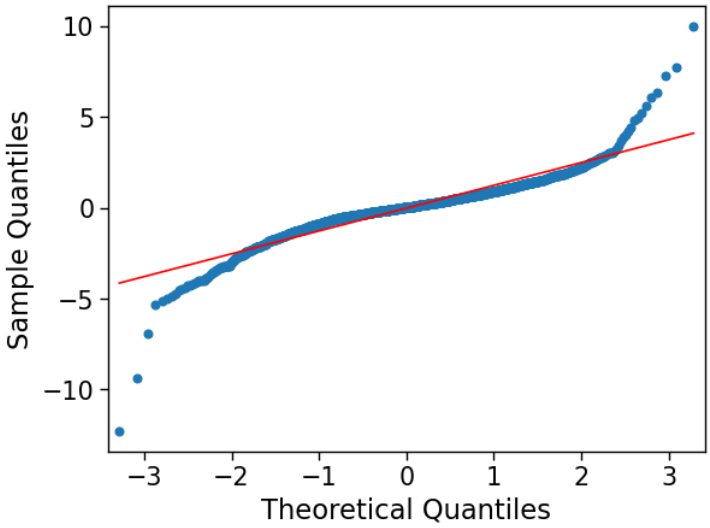

Figure 9

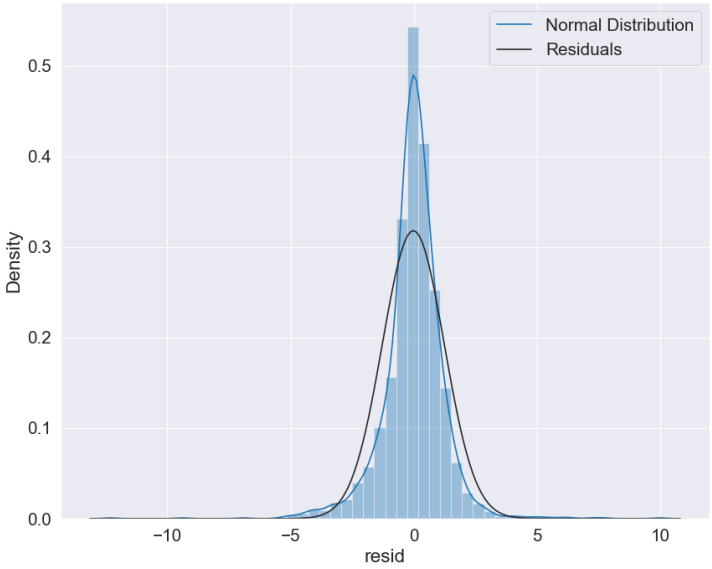

Figure 10

4.1.2 Asymmetric Models

Given the non-normality of GARCH residuals, asymmetric models have been implemented to improve prediction accuracy.

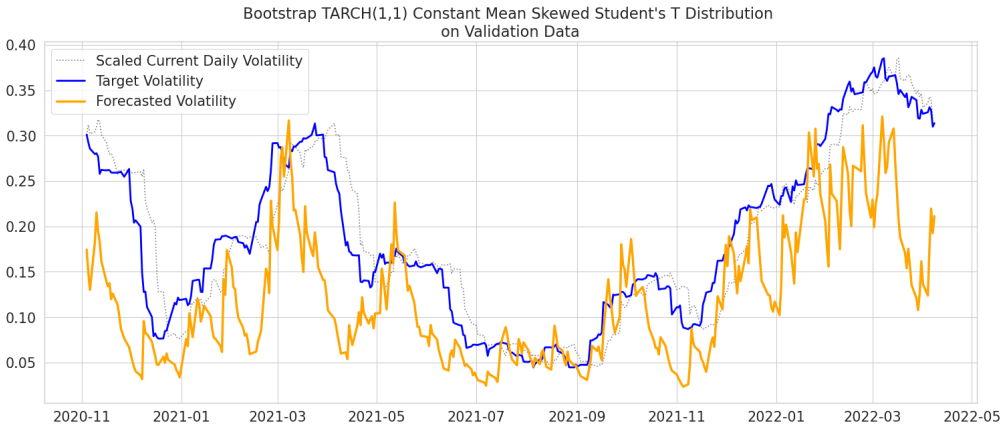

Figure 11

A plain TARCH(1,1) model was implemented, which provided a substantial improvement in terms of the Root Mean Squared Percentage Error (RMSPE) and the Root Mean Squared Error (RMSE) as shown in Table 1. The model was then fine-tuned in an attempt to further improve prediction accuracy.
It is worth noting that, as TARCH models do not have closed-form solutions for more than one-period-ahead forecasts, these models required the implementation of bootstrapping in order to be estimated.
As can be seen from Figure 11, the systematic underestimation of the model was somewhat improved and the forecasted values closely tracked the target process.
Given the substantial improvement provided by the TARCH model, the values of p, q, and o were fine-tuned in an effort to further minimize the RMSE and RMSPE. The best TARCH model was found to be the TARCH(1,2) as shown in Figure 12. Figure 13 shows the forecasted values for the best TARCH(1,2).

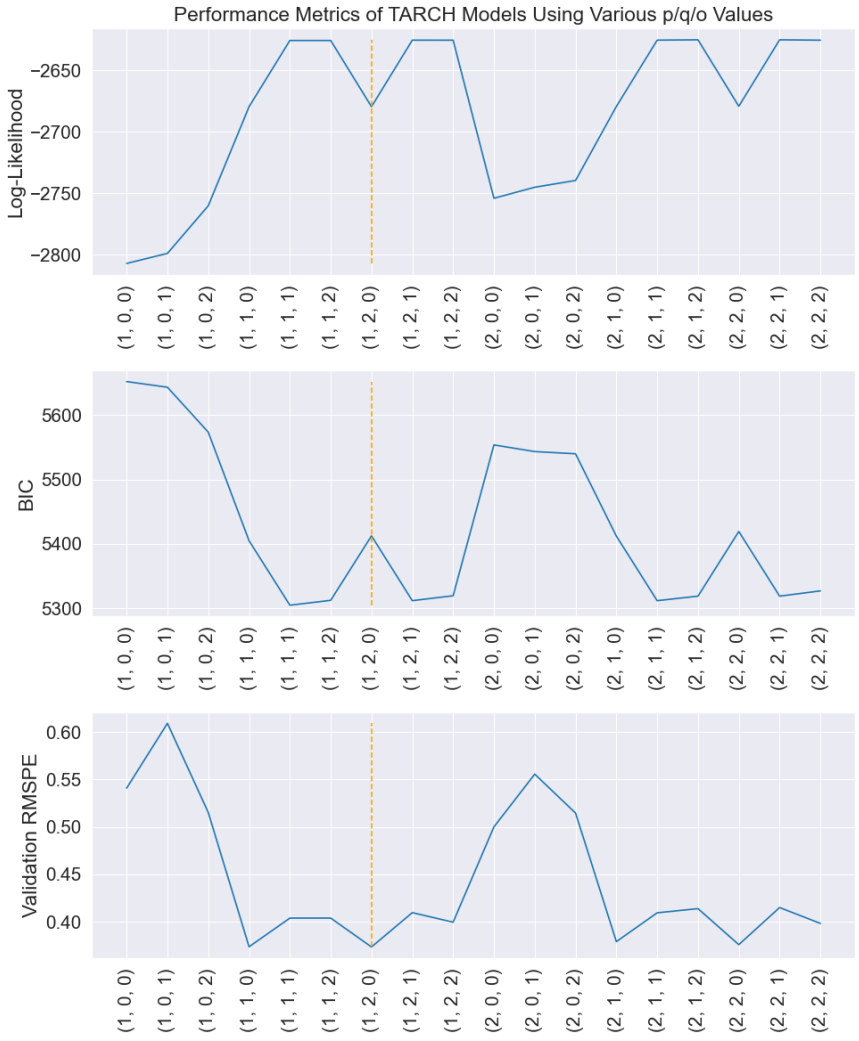

Figure 12

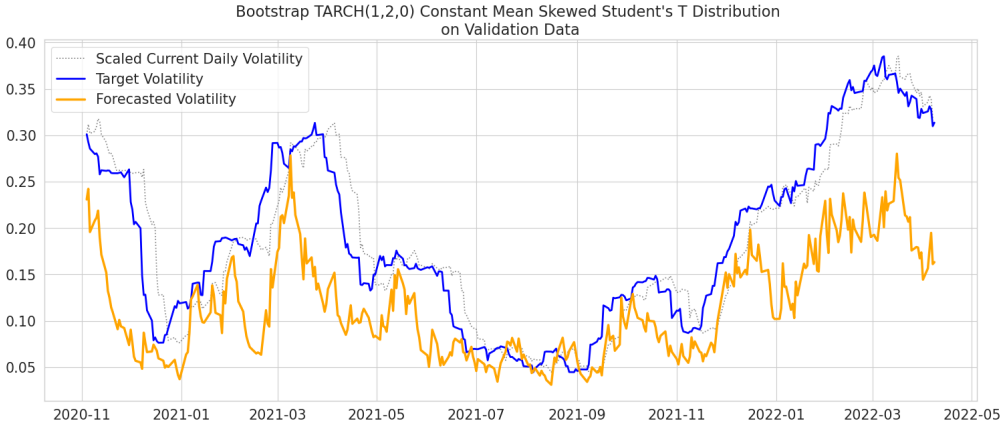

Figure 13

The GJR-GARCH(1,1) model, a specific type of TARCH model, was also implemented.
However, when compared to the TARCH models, the GJR model did not achieve better performance in the validation set (as can be seen in Table 1). Additionally, the GJR model still exhibited issues of underestimation in its forecasts (as shown in Figure 14).

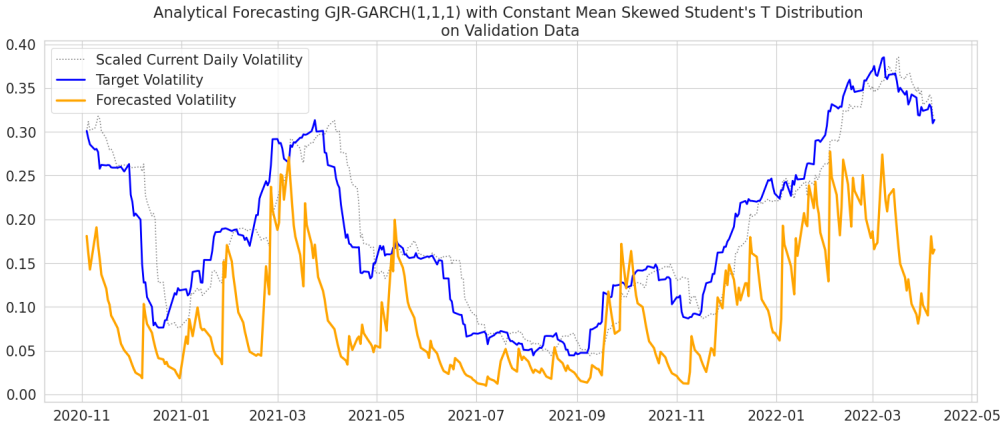

Figure 14

4.1.3 Final Comparison

As shown in Table 1, the TARCH(1,2) model exhibits the best overall performance.
The results demonstrate that the TARCH models provide a significant improvement in performance when compared to the baseline GARCH(1,1) model. The GJR-GARCH model, which also allows for asymmetric shocks, did not show improvement in forecasting performance compared to the GARCH(1,1) model.

Table 1: GARCH Models' Out-of-Sample results

Model	**Validation RMSPE**	**Validation RMSE**
Bootstrap TARCH(1,2,0)	0.393214	0.088186
Bootstrap TARCH(1,1)	0.404801	0.085647
GARCH(1,1)	0.502673	0.105367
GJR-GARCH(1,1,1)	0.543954	0.106587

4.2 Machine Learning Models

The results from the Neural Network models consistently reduce the Root Mean Squared Percentage Error (RMSPE) and Root Mean Squared Error (RMSE) compared to the other models. Additionally, a simple Linear Regression model is able to outperform the TARCH(1,2) model, as shown in Table **??**.
Furthermore, a Long Short Term Memory (LSTM) Neural Network with a simple design (one hidden layer and 20 neurons) is able to outperform the aforementioned Linear Regression model (see Table 2). This model was found to be the best performing model in this context, as shown in Figure 15. The LSTM model was able to provide a substantial increase in performance compared to the other models.
Surprisingly, the worst performing model was the Multivariate LSTM, as shown in Table 2. However, it is worth noting that a more precise fine-tuning of its design could be adopted to improve its performance.

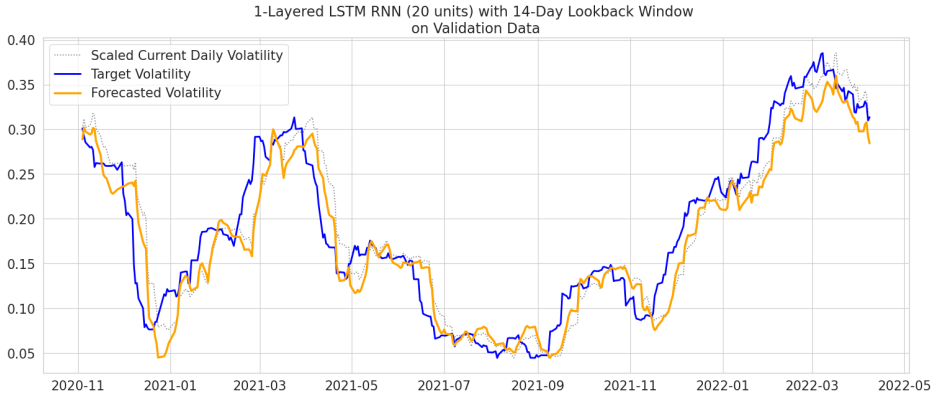

Figure 15

Table 2: Overall Performances

Model	Validation RMSPE	Validation RMSE
LSTM layer 20 units	0.237995	0.031979
LSTM layer 32 units	0.240466	0.031194
Bidirectional LSTM layer 64/32/16 units	0.273216	0.030455
Fully Connected NN (Linear Regression)	0.339261	0.040553
Bootstrap TARCH(1,2)	0.392713	0.088061
Bootstrap TARCH(1,1)	0.403949	0.085424
GARCH(1,1)	0.502673	0.105367
GJR-GARCH(1,1)	0.543954	0.106587
Multivariate Bidirect LSTM 3 layers	0.547389	0.069597

5 Conclusions

The results of this study indicate that the Machine Learning models significantly outperform the GARCH type models. Despite the expectation that the Multivariate LSTM would perform better than the other LSTM networks, this was not the case. However, further optimization of the Multivariate LSTM model may lead to improved results.

It should be noted that the training and fine-tuning of these models requires a significant computational effort and can be costly. Additionally, as the models become more complex, the need for frequent retraining becomes increasingly time-consuming.

While the GARCH models may not have performed as well, they have the advantage of interpretability. In certain applications, the ability to explain the reasoning behind a decision may be more important than the absolute accuracy of the model. In contrast, Neural Networks are often considered "black box" models, providing little interpretability or explanation of the results obtained.

This can make it difficult to understand the reasoning behind the model's predictions and decisions. In addition, it can also be challenging to ensure that the model is making decisions that align with regulatory and compliance requirements. These factors can limit the adoption and trust of machine learning models in finance, despite their potential for improved performance.

References

[1] Bollerslev, T. (1986). Generalized autoregressive conditional heteroskedasticity. Journal of Econometrics, 31(3), pp.307–327.

[2] Bucci, A. (2020). Realized Volatility Forecasting with Neural Networks. Journal of Financial Econometrics. 10.1093/jjfinec/nbaa008.

[3] GLOSTEN, L.R., JAGANNATHAN, R. and RUNKLE, D.E. (1993). On the Relation between the Expected Value and the Volatility of the Nominal Excess Return on Stocks. The Journal of Finance, 48(5), pp.1779–1801.

[4] NASDAQ100 (NDX) (2023). Stock Prices. Yahoo!Finance.

[5] Kevin Sheppard (2021, March 3). bashtage/arch: Release 4.18 (Version v4.18). Zenodo.

[6] Hochreiter, S. and Schmidhuber, J. (1997). Long Short-Term Memory. Neural Computation, 9(8), pp.1735–1780. 10.1162/neco.1997.9.8.1735.