Novel Resilient Model Risk Forecasts based on Neuro Volatility Models

Abstract—**Recently, there has been a growing interest in using neuro volatility models in fuzzy forecasting and fuzzy option pricing. Neuro volatility models are used to model and predict financial market volatility by extending the neutral network autoregressive (NNAR) model for nonlinear nonstationary times series data. In financial risk forecasting, various risk forecasting models for volatility are used to obtain the volatility forecasts, and the model risk ratio based on all the models is calculated to assess the stability of the financial system. However, the recently proposed neuro volatility models (based on neural networks such as LSTM, NNAR, etc.) are not used in evaluating the model risk. In this paper, novel 'neuro model risk forecasts' are obtained by including recently proposed neuro volatility models, and the resiliency of the financial system is studied. Unlike the existing model risk ratio forecasting based on linear volatility models, the driving idea is to use more appropriate nonlinear nonstationary neuro volatility forecasting models to obtain the model risk forecasts. The proposed model risk forecasts in this paper have been evaluated through extensive experiments, and it is shown that the model risk ratio can effectively serve as a metric for assessing the resilience and stability of the targeted financial system.**

Index Terms—**Neuro volatility models, neuro model risk ratio, resilient financial system, time series cross-validation**

I. Introduction

Forecasting become a vital component of many research, and forecasting methods are constantly evolving, striving to get an edge that the creator can use to increase profit or advance research. Many different models are available for forecasting purposes, depending on the desired interpretability of the model, the characteristics of the time series, and the versatility desired within the model. Some examples of the most common forecasting models for time series data are ARIMA (autoregressive integrated moving average) models, neural network autoregressive (NNAR) models, Prophet models, recurrent neural networks (RNN), and long short-term memory (LSTM) neural networks. Each method has its benefits and fallbacks, making it important to compare and contrast different models to see which is most suitable.

Neural network models are becoming popular in time series forecasting due to their accuracy and capability of handling more complex real-world problems [1], [2]. The feed-forward neural network is a popular way to approximate a given multivariate nonlinear function. The network consists of an input variable(s), an output variable, and a hidden layer(s), which join the input variable(s) and output variable. The hidden layer(s) of the neural network is created with units called neurons. Neurons are capable of identifying new features from the data. With a sufficient number of neurons in the hidden layer(s), a neural network can approximate any continuous function and translate inputs to desired output signals. One of the popular neural networks specifically designed to handle time series data is proposed in [3]. It can be implemented in R using the `nnetar` function available in the `forecast` package, and it fits a single-layer neural network. Also, it can be modified to consider some feature values when training the models, and thus, in such cases, it fits a neural network dynamic regression model. Even though `nnetar` produces a very cable neural network to forecast time series data, it only fits a neural network with a single hidden layer. There is no definitive guideline for establishing the precise count of hidden layers and neurons in a neural network that guarantees optimal performance across all tasks. The most suitable architecture hinges on diverse factors such as problem complexity, data availability, and computational capabilities. Augmenting the number of hidden layers and neurons within a neural network holds the potential to enhance its effectiveness and precision. When setting a multilayer neural network with several neurons for time series data, one popular approach is to use the `keras` package, which connects with the `tensorflow` packages to fit a neural network model. Here, the neural network interfaces with optimized Python code to build an RNN. Implementing this neural network to forecast asset trading volume on asset return and log volatility is discussed in [4] using R, and some other applications with Python are discussed in [5]. Nevertheless, it is crucial to keep in mind that intricate networks do not invariably ensure superior outcomes. Introducing an excessive number of layers or neurons can result in overfitting, where the network excessively memorizes the training data, potentially hindering its ability to generalize effectively to novel, unseen data. Moreover, this complexity escalation reduces interpretability and demands more resources for both model implementation and execution.

One major issue with RNNs is the vanishing gradient problem. The vanishing gradient problem occurs during the training of RNNs, especially over long sequences, where the gradients of the loss function concerning the weights become extremely small. As a result, the model struggles to learn long-term dependencies in the data, hindering its ability to capture relationships between distant time steps. In order to address this issue, more advanced RNN architectures, such as Long Short-Term Memory (LSTM) networks and Gated Recurrent Units (GRUs), have been developed. These architectures include mechanisms to selectively retain or update information over time, allowing them to better capture long-range dependencies in sequential data. These models have

Preprint

been implemented in finance [6], [7], electricity, agriculture, and also in many other areas.

Modeling and predicting financial risk plays a pivotal role in risk management strategies for investment decisions and capital distribution. Since the Basel Accord of 1996, prioritizing financial risk management has become imperative within the financial services sector. According to the Basel Committee on Banking Supervision's Basel III monitoring report in 2021, financial institutions are mandated to employ increasingly sophisticated credit scoring models to optimize capital allocation and bolster risk management practices.

In financial time series, the financial return has always been used to model financial risk. For a reasonable asset, the price shows exponential behavior in time. However, an exponentially increasing time series is hard to manipulate using most statistical tools, e.g. correlation and regression. Also, the mean value of an exponentially increasing time series has no meaning. Thus, a popular approach is to consider the return (simple or compound) of the price of the asset. Return is a complete and scale-free summary of the investment opportunity. If the value of the return is negative, then that implies the value of the asset has declined and is wise-versa. When the returns of the asset form a stationary series, the time series tool can be applied and carry out statistical analysis.

The objective of risk measures is to help decision-making. Recently, there has been notable growth in research focused on forecasting risk in financial time series data. Three pivotal metrics defining risk include volatility, Value at Risk (VaR), and Expected Shortfall (ES). In financial econometrics, volatility is commonly interpreted as the degree of price variation in the series over time, mathematically expressed as the standard deviation of logarithmic returns. VaR, a widely adopted risk measure, signifies the potential financial loss an investment might face within a given confidence level and timeframe. ES, on the other hand, supplements VaR by estimating the expected loss beyond the VaR threshold through conditional expectation calculations. Volatility is a strong risk measure when the returns are normally distributed. However, in many practical situations assumption of normality is violated, and volatility may lead to misleading conclusions. Thus, there is a need for improved risk volatility forecasting models. In academic writing, forecasts of conditional volatility are obtained by taking the square of the forecasted conditional variance. However, [8] points out that this approach is not very efficient as the asymptotic variance of this estimate is very large. For example, a large movement of the returns (with heavy-tailed distributions) will have a larger influence on the estimate of the volatility via the square root of the variance than the direct estimate of volatility. Moreover, financial data is usually heavy-tailed. In particular, certain heavy-tailed distributions, such as Student's-t with d.f. between two and four, have theoretically infinite kurtosis. Therefore, [8] introduced data-driven exponentially weighted moving average (DDEWMA) models for forecasting volatility directly, instead of forecasting variance. Applications of this data-driven model in price forecasting, model risks, and option pricing can be found in [9], [10]. These studies also discussed the superiority of neuro-volatility models in different contexts. Danielsson et. al [11] first introduced the model risk ratio idea and applied the methodology to market risk measures, VaR and ES, which depend on the volatility based on the forecast of conditional variance. However, other risk metrics such as root mean squared error (RMSE), mean absolute errors (MAE), and mean absolute percentage error (MAPE) can be used to construct model risk.

The novelty of this paper is to obtain direct volatility forecasts using neural network models while taking basic time series models and autoregressive models into account. We introduce neuro model risk forecasts based on direct volatility forecasts which allow more appropriate nonlinear nonstationary neuro volatility forecasting models. The work investigates the stability of the forecasts of different models considering the proposed neuro model risk. Moreover, we investigate the models' performance in price forecasts using adjusted closing price data of Google, CBOE Volatility Index (VIX), and Bitcoin-USD. We also investigate the models' performance on temperature data from Ontario, Canada. When applying autoregression and machine learning models for time series data, a common approach is to evaluate the performance/accuracy of models by considering observed and forecast data. In this study, MAE, RMSE, and MAPE are used as measures of forecast errors and detailed description of the risk measures are given in Section II. Furthermore, the work also incorporates a more sophisticated time series cross-validation procedure to evaluate the forecast accuracy than the traditional evaluation based on training and test sets.

The remainder of the paper is organized as follows. Section II provides the neural network model, neuro volatility model, LSTM, Prophet model, forecast errors, model risk, and time series cross-validation. In Section III, experiments and results are described. Finally, concluding remarks are given in Section IV.

II. Methodology

A. Price and Returns

Generally, in literature, the price of an asset or a stock is denoted by P_t. t represents a particular day or a time (frequency) that interested the researcher. If there are multiple assets price of each asset can be denoted by $P_{t,k}$. Often The relative change in the price of a financial asset over a given time interval is expressed as a percentage. Continuously compounded (CC) returns/log-returns (r_t) calculated on a given asset or a stock is given by

$$r_t = log\frac{P_t}{P_{t-1}} = log(P_t) - log(P_{t-1}).$$

B. Volatility

Volatility, denoted as the standard deviation of returns, serves as the primary risk metric in the majority of financial analyses. It proves adequate as a risk measure when returns adhere to a normal distribution. However, this assumption

Preprint

of normality is frequently violated across various financial returns. Consequently, relying solely on volatility as a risk measure can yield misleading conclusions. The degree of inaccuracy resulting from volatility usage varies across specific applications in practice. In many scenarios, concerns about extreme outcomes are minimal, making the utilization of volatility relatively benign. However, in the realm of financial risk management, volatility tends to systematically underestimate risk, unlike in other applications.

There are two concepts for volatility, Unconditional volatility (σ) and Conditional volatility (σ_t). Unconditional volatility is volatility over an entire time period, and in conditional volatility, volatility is conditional on a given time period, the history, model, and model parameters.

C. Forecasting models

The model risk introduced in this study allows linear as well as more appropriate nonlinear nonstationary neuro-volatility forecasting models. Here, we provide specifics of the forecasting models used in this study to construct model risk.

1) Neural Network Autoregressive (NNAR) Model: Before discussing NNAR, it is crucial first to discuss the basis of neural networks in forecasting contexts. An NNAR model can be written as (p, k), where p denotes the number of input variables (features) in the model and k denotes the number of nodes in the hidden layer. We can visualize neural networks as a collection of nodes organized in multiple layers of linear and non-linear functions fitted to minimize a cost function. For example, we can use the mean squared error (MSE) or RMSE as a cost function. Any function used to assess the forecasted values with the actual values can be used as a potential cost function [3].

The simplest type of neural network is where there are no hidden layers, which simplifies to a simple linear regression model. We can express this model as $\hat{y} = b + w_1 x_1 + ... + w_p x_p$, where the coefficients (weights) on the input variables (features) and b are determined through linear regression. A neural network with no hidden layers of nodes is denoted as NNET$(p, 0)$ where there are p input variables; however no nodes in any hidden layer. Adding hidden layers or layers of nodes transforms the model into a multilayer feed-forward network, which means each layer of nodes receives inputs from the previous layers.

The inputs to each node are combined using a weighted linear combination:

$$z_j = b_j + \sum_{i=1}^{k} w_{i,j} x_i,$$

where z_j is the j^{th} node in a hidden layer, $w_{i,j}$ are the weights assigned using a learning algorithm for the i^{th} input for the j^{th} node, and x_i is the i^{th} input variable. Note that the values of the weights are often restricted to prevent overfitting [3]. Then, the result is modified using a nonlinear function, such as the sigmoid function $s(z) = \frac{1}{1+e^{-z}}$. The result from the nonlinear function is used as inputs to the next layer. Generally, we can express a neural network model as $y = f(\vec{x}) + \epsilon$, where $\vec{x} = (x_1, x_2, ..., x_p)$ and ϵ is an error series assumed to be homoscedastic. Note that when fitting a neural network autoregressive model when the number of nodes (k) is not specified, k is set to $k = (p + P + 1)/2$, where p is the number of nonseasonal lagged values, and P is the number of seasonally lagged values.

A feed-forward neural network differs from a recurrent network, as all inputs go forward in the same direction. The benefits of a feed-forward neural network are the simplicity of the model, low complexity, and thus faster speed. However, it is plagued by potentially losing information from earlier layers (lack of memory) and does not share information between nodes in the same layer. In recurrent neural networks, such as LSTM, inputs can go in both directions, allowing information from previous layers to be retained. There is, however, increased complexity and slower speed. The current work being conducted by [12] focuses on developing guidelines and best practices for the use of recurrent neural networks.

2) Neuro-Volatility model Using NNAR: In finance, stock prices are modeled as a geometric Brownian motion. Where the stock prices are denoted as P_t, $t = 1, \ldots, T$. The first step when modeling volatility is to transform the adjusted closing prices P_t and calculate the log returns. Then calculate the mean of the log returns, denoted $\bar{r}$. We will then compute the correlation between $r_t - \bar{r}$ and $sign(r_t - \bar{r})$ denoted as $\hat{\rho} = corr(r_t - \bar{r}, sign(r_t - \bar{r}))$, then compute the observed volatility $V_t = \frac{|r_t - \bar{r}|}{\hat{\rho}}$. The observed volatility formula stems from the unbiased estimator of the standard deviation introduced by [8]. After all computations are completed, we can model the observed volatility. The observed volatility can then be used as the input time series to fit the neural network model; the fitted neural network model with the volatility series as the training data can forecast future volatility of the stock/cryptocurrency prices. Figure 1 provides a general representation of a neural network with one hidden layer. The network forecasts volatility at time $t+1$, V_{t+1}, using lag values of observed volatility $(V_t, V_{t-1}, \ldots, V_{t-p})$.

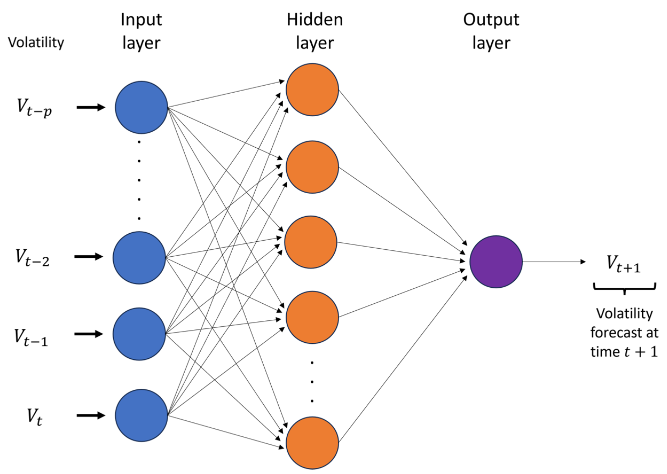

Fig. 1. Neuro-Volatility Model

Preprint

3) Recurrent Neural Networks (RNN): RNNs are a type of artificial neural network (ANN) designed to process time series data by maintaining an internal state, or memory, to capture temporal dependencies. Unlike feedforward neural networks, which process data strictly in one direction, RNNs possess internal memory elements that allow them to inherently tackle both the temporal order and dependencies present within the data [13]. The basic architecture of an RNN consists of a series of interconnected nodes or units, organized into layers. Each unit takes an input vector and produces an output vector, along with a hidden state vector that represents its internal memory. This hidden state is then passed to the next unit in the sequence, allowing the network to capture information from previous inputs. Given an input sequence $x = (x_1, x_2, ..., x_T)$ of length T, the forward propagation process in an RNN unfolds over time steps $t = 1, 2, ..., T$. At each time step t, the input x_t is fed into the network, along with the hidden state vector from the previous time step h_{t-1}. The output y_t and the updated hidden state h_t are then computed using the following equations:

$$h_t = \sigma(W_{ih}x_t + W_{hh}h_{t-1} + b_h)$$

$$y_t = \text{softmax}(W_{ho}h_t + b_o)$$

where W_{ih}, W_{hh}, W_{ho} are weight matrices, b_h, b_o are bias vectors, and σ is an activation function such as the hyperbolic tangent $(\tanh)$ or the rectified linear unit (ReLU).

RNNs have found widespread use in various applications involving sequential data, such as natural language processing (e.g., language modeling, machine translation), speech recognition, time series analysis, and more.

4) Long short-term memory (LSTM): LSTM, a variant of RNNs, differs from Traditional Neural Networks by incorporating internal loops that enable memory retention from previous computations. This architecture is adept at processing time series data due to its ability to retain information over time. Unlike conventional RNNs, LSTMs typically mitigate issues like vanishing and exploding gradients.

Figure 2 illustrates an LSTM cell in which every line conveys a complete vector, transmitting from the output of one node to the inputs of others. Pink circles symbolize pointwise operations, such as vector addition, whereas yellow boxes denote neural network layers undergoing learning. Converging lines indicate concatenation, while diverging lines signify the replication of content, with the duplicates directed to various destinations. Thus, at each step t, LSTM cell contains following components:

- Gating variables include the forget gate f_t, input gate i_t, and output gate o_t, each of which comprises neural networks employing sigmoid activation functions.
- The candidate cell state $\tilde{c}_t$ is a neural network utilizing a tangent activation function.
- Cell and hidden states consist of the memory state c_t and the hidden state h_t.

LSTM models offer a method for predicting non-stationary stock prices, often represented by geometric Brownian motion. The sample autocorrelation of the price series exhibits hyperbolic decay, indicating its non-stationary nature. In this context, LSTM utilizes n historical closing prices $P_1, \ldots, P_n$ as input and predicts D data points using the latest observation P_n. It generates forecasts for $n - D$ one-step ahead points $\hat{P}_{D+1}, \ldots, \hat{P}_n$ corresponding to $P_{D+1}, \ldots, P_n$, as well as D future points forecasts $\hat{P}_{n+1}, \ldots, \hat{P}_{n+D}$. Each forecast $\hat{P}_{t+1}$ depends on the preceding D data points $P_t, \ldots, P_{t-(D-1)}$ obtained through LSTM. This approach can be readily extended to obtain forecasts of the volatility series $(V_1, \ldots, V_n)$ associated with the price.

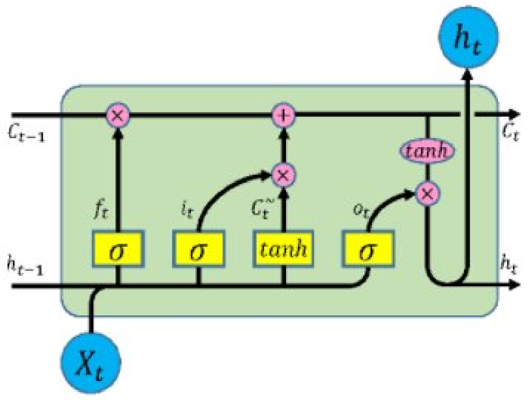

Fig. 2. LSTM cell

5) Prophet Model: Prophet models are simpler in terms of comprehension compared to neural network models. Facebook developed Prophet models, mainly known for forecasting daily data with weekly and yearly seasonality, plus holiday effects, which makes the model very useful for potentially forecasting store sales or other common commodities. Prophet models are best suited for time series with strong seasonality and several seasons of historical data [3].

We can express the prophet model as a nonlinear regression model:

$$y_t = g(t) + s(t) + h(t) + \epsilon_t,$$

where $g(t)$ describes a piecewise-linear trend, $s(t)$ describes the various seasonal patterns, $h(t)$ captures the holiday effects, and ϵ_t is a white noise error term.

Prophet approaches the forecasting problem by reframing it as a curve-fitting exercise, where different components are modeled separately:

- Growth is represented by the function $g(t) = \frac{C}{1+\exp(-k(t-m))}$, where C denotes the saturation level, k represents the slope or base growth rate with $\frac{\partial g}{\partial k} > 0$, m is the time offset, and $C(t)$ can be a function of time incorporating exogenous analyst forecasts.
- Seasonality is captured using a Fourier series, with P indicating the period (365 for annual, 7 for weekly) and N denoting the number of Fourier series components. The seasonality function is expressed as $s(t) = \sum_{n=1}^{N} \left[a_n \cos\left(\frac{2\pi nt}{P}\right) + b_n \sin\left(\frac{2\pi nt}{P}\right)\right]$.

Preprint

- Holidays are modeled by incorporating a change in forecast at time i, represented by κ_i and organized into a vector κ. The function $h(t)$ is determined by the indicator vector $I(t)$ of holiday dummies, resulting in $h(t) = I(t) \cdot \kappa$.

D. Forecast Errors

A forecast error represents the disparity between an observed value and its corresponding forecast. In this context, "error" does not denote a mistake but rather signifies the unpredictable component of an observation. Mathematically, it can be expressed as:

$$e_{t+h} = y_{t+h} - \hat{y}_{t+h|t},$$

where the training data is given by $\{y_1, \ldots, y_t\}$, and the test data is given by $\{y_{t+1}, y_{t+2}, \ldots\}$.

It is important to differentiate forecast errors from residuals in two aspects. Firstly, residuals are computed based on the training set, while forecast errors are derived from the test set. Secondly, residuals are associated with one-step forecasts, whereas forecast errors may involve multi-step forecasts. Various methods can be employed to summarize forecast accuracy by analyzing the forecast errors. Forecast errors share the same scale as the data, rendering accuracy metrics relying solely on e_t scale-dependent. Consequently, such metrics cannot be employed to compare series with disparate units. The two most commonly used scale-dependent measures are based on the absolute errors or squared errors: Mean absolute error (MAE):

$$MAE = \text{mean}(|e_t|),$$

Root mean squared error (RMSE):

$$RMSE = \sqrt{\text{mean}(e_t^2)}.$$

When evaluating forecast methods for either a single time series or multiple time series sharing the same units, the Mean Absolute Error (MAE) is commonly favored due to its simplicity in both comprehension and computation. A forecasting technique minimizing the MAE yields forecasts resembling the median. On the other hand, minimizing the Root Mean Squared Error (RMSE) results in forecasts resembling the mean. Hence, despite its more complex interpretation, the RMSE is also widely utilized.

The percentage error is given by:

$$p_t = 100 e_t / y_t.$$

Percentage errors offer the benefit of being devoid of units, making them commonly employed for comparing forecast performances across different datasets. The most prevalent measure utilized in this context is: Mean absolute percentage error (MAPE):

$$MAPE = \text{mean}(|p_t|).$$

Measures relying on percentage errors present drawbacks such as becoming infinite or undefined if $y_t = 0$ for any t within the period of interest, as well as yielding extreme values if any y_t is close to zero. Additionally, an often overlooked issue with percentage errors is their assumption that the unit of measurement possesses a meaningful zero. For instance, assessing the accuracy of temperature forecasts using percentage errors is nonsensical for both the Fahrenheit and Celsius scales due to temperature's arbitrary zero point.

E. Neuro Model Risk

The model risk of risk models has been introduced by Danielson et. al. [11]. Suppose we want to forecast risk for the day $t+1$ based on the information available on day t. Considering we have N candidate models to forecast risk on day $t+1$, each providing different forecasts as $\{\text{Risk}^n_{t+1}\}_{n=1}^N$. Then the model risk is defined as the ratio of the highest to the lowest risk forecasts.

$$\text{Model risk} = \text{Risk Ratio}_{t+1} = \frac{\max\{\text{Risk}^n_{t+1}\}_{n=1}^N}{\min\{\text{Risk}^n_{t+1}\}_{n=1}^N}. \quad (1)$$

where N indicates the number of different risk measures calculated using forecasts. A model risk ratio close to one indicates that the financial system is stable. Further details about the model risk and how it can be used to understand financial stability using different risk measures can be found in [11], [14].

The existing model risk ratio forecasts are based on linear volatility models only and use VaR and ES as risk measures. However, other risk metrics such as RMSE, MAE, and MAPE from nonlinear nonstationary neuro volatility forecasting models can be used to construct model risk. Hence, we introduce neuro model risk forecasts based on direct volatility forecasts which allow more appropriate nonlinear nonstationary neuro volatility forecasting models and can be used to investigate the stability of the forecasts of different models.

F. Time Series Cross-validation

The quality of the forecasts can be improved by selecting the best models for given data, and the best models can be chosen using cross-validating techniques. It is important to note that traditional cross-validation techniques such as K-Fold and Leave-One-Out can not be used for time series data as we are required to protect the correlation structure of the time series data and bring its information to the model. Thus, choosing data randomly for training and testing data will not work for time series data, and we are required to choose data sequentially to protect the correlation structure. One way to implement cross-validation is by averaging over the test sets. It is commonly known as time-series cross-validation/evaluation on a rolling forecasting origin. In this approach, first, obtain a one-step-ahead forecast using the first few observations (training data). Second, again one-step-ahead forecast is obtained by adding one more observation to the training data. We continuously do this until the second last observation is added to the training data. Here, the training data set gradually grows. This approach can be modified to allow multi-step ahead forecasts, and the performance of a given model can be compared in terms of one-step ahead

Preprint

forecasts or multi-step ahead forecasts. This approach is much better than the single split because, in the single split, we only have one observation for each of the forecast horizons, and here, we have multiple observations. With big data, this requires significant computational power and memory. Thus, we can increase the number of observations added to the training data to reduce the number of times we calculate test errors.

Figure 3 illustrates the process of time series cross-validation for a single series. The process starts with three observations (blue circles) as training data and obtains a one-step-ahead forecast (orange stars). In the next step, four observations are considered as training and the forecast of the fifth observation is obtained. The process continues until the forecast of the last observation is obtained with the selected model. This approach helps to determine which models perform well with given data while considering more than one realization of forecasts.

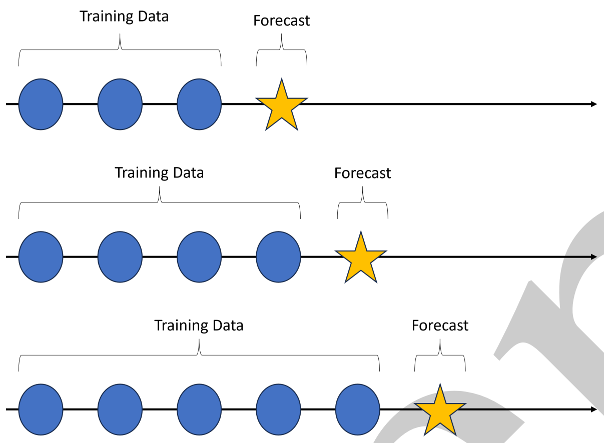

Fig. 3. Time Series Cross-validation (The blue circle represents training data and the orange star represents one-step-ahead forecast)

III. Experimental Results

The experimental results section of this paper consists of three subsections. Subsection one reports and discusses forecasts accuracy and models stability of assets price forecasts considering Google, VIX, and Bitcoin adjusted closing price data. In the second second subsection, the study observes volatility forecasts for Google, VIX, and Bitcoin. The study also reports forecasts accuracy of volatility using RMSE, MAE, and MAPE and discusses market stability using models risk ratio. Adjusted closing prices of Google, VIX, and Bitcoin are obtained from Yahoo! Finance for the study period 2016-01-01 to 2018-12-31. In the last subsection, the study investigates the temperature forecast accuracy of different models and model risk ratios using temperature data from Ontario, Canada for the period of 2018-01-01 to 2019-12-31. The importance of investigating temperature forecasts is their usability in electricity demand forecasts and weather forecasts (see [3] for further details). The study discusses the performance and stability of different models for temperature data and shows the superiority of the NNAR model. The current body of literature highlights the effectiveness of integrating various meteorological factors to improve model accuracy. Our initial examination suggests that incorporating additional meteorological variables, such as snowfall and air density, improves the accuracy of temperature forecasts. Consequently, this study incorporates a range of meteorological influences, including precipitation (mm/hour), snowfall (mm/hour), snow mass (kg/m²), air density (kg/m³), and cloud cover fraction, to enhance temperature forecasting. All the values of meteorological features values including temperature are obtained from Renewables.ninja for the same study period.

A. Forecast Accuracy and Model Risk for Adjusted Closing Prices

In this subsection, the study provides price forecast accuracy and model risks for different models. Note that forecast errors are obtained with time series cross-validation, and reported errors are average values of RMSE, MAE, and MAPE considering one-step-ahead forecasts of price.

TABLE I
Forecast Accuracy of Eight Models for Google

Model	RMSE	MAE	MAPE
NAIVE	0.71	0.48	0.96
Drift	0.71	0.48	0.96
Prophet	2.67	2.01	3.96
ARIMA	0.72	0.49	0.98
NNAR	0.72	0.51	1.01
ETS	0.71	0.48	0.96
LSTM	1.63	1.31	2.46
RNN	0.91	0.68	1.30
Model Risk	3.76	4.19	4.13

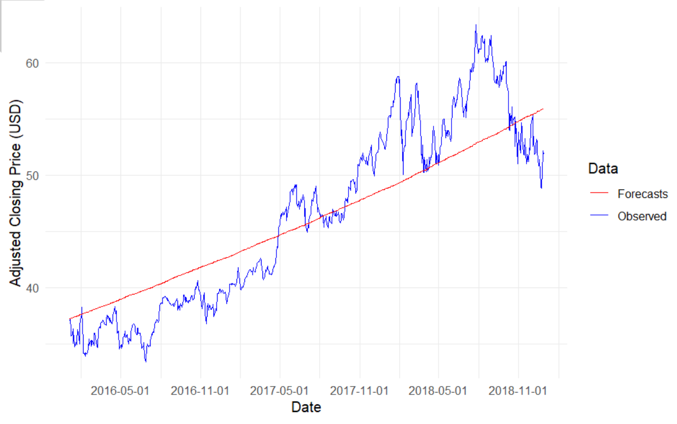

Fig. 4. Point forecasts of Adjusted closing Prices of Google from January 2016 to December 2018 using ETS

Table I provides forecasts accuracy and models risk for adjusted closing prices of Google. It can be observed that Prophet and LSTM have higher errors compared to other models. Thus, it leads to higher model risks under RMSE, MAE, and MAPE. A similar observation can be made with adjusted closing prices of Bitcoin and results are summarized in Table

Preprint

III. However, for VIX, LSTM shows similar performance accuracy compared to other models, and still, Prophet forecast accuracy is low for adjusted closing prices of VIX (Table II). The 'NNAR' model with the specification 'NNAR(9,5)' is being used to model or forecast the adjusted closing prices of Google stock to capture the nonlinear pattern. That indicates the model uses the previous 9 observations ($p = 9$) as inputs and employs a neural network with 5 hidden units ($k = 5$) for the forecast. Adjusted closing prices of VIX are modeled with its previous 5 observations ($p = 5$) and in the hidden layer there are 3 hidden units ($k = 3$). However, for Bitcoin, a more complicated model is suggested, and to model/forecast the adjusted prices of Bitcoin, the previous 29 observations ($p = 29$) are used as inputs and employ a neural network with 15 hidden units ($k = 15$).

TABLE II
Forecast Accuracy of Eight Models for VIX

Model	RMSE	MAE	MAPE
NAIVE	1.55	0.85	5.46
Drift	1.56	0.85	5.46
Prophet	3.70	2.33	14.65
ARIMA	1.59	0.87	5.55
NNAR	1.62	0.88	5.69
ETS	1.56	0.86	5.49
LSTM	1.70	1.04	7.03
RNN	1.72	1.11	7.71
Model Risk	2.39	2.74	2.68

TABLE III
Forecast Accuracy of Eight Models for Bitcoin

Model	RMSE	MAE	MAPE
NAIVE	359.95	174.79	2.88
Drift	360.17	174.41	2.87
Prophet	2003.26	1111.62	16.44
ARIMA	399.75	185.23	2.99
NNAR	593.91	249.85	3.92
ETS	363.21	175.36	2.89
LSTM	515.63	300.22	4.60
RNN	400.91	220.31	3.29
Model Risk	5.56	6.37	5.71

Results from Tables I-III also show neuro model risk forecasts for Google, VIX, and Bitcoin. We focus on the use of the neuro model risk ratio to measure the level of disagreement among the risk measures (RMSE, MAE, and MAPE) from different risk forecasting models. For example, model risk based on RMSE for Google is 3.76 (I) indicates the degree to which different models used here disagree. We also examine the sensitivity of the results on the neuro model risk since one might employ different measures to quantify the disagreement among the alternative models other than RMSE, such as MAE and MAPE. Moreover, Figure 4 demonstrates the one-step-ahead forecasts for adjusted closing prices of Google using the NNAR model.

B. Forecast Accuracy and Model Risk for Volatility

For the volatility of Google stock, an NNAR model with lagged inputs of 14 observations ($p = 14$) and 8 hidden units ($k = 8$) was employed. Similarly, for the VIX index volatility, an NNAR model with lagged inputs of 4 observations ($p = 4$) and 2 hidden units ($k = 2$) was utilized. Additionally, for Bitcoin volatility, an NNAR model with lagged inputs of 25 observations ($p = 25$) and 13 hidden units ($k = 25$) was applied. These models leverage historical volatility data to make forecasts, with each model tailored to the specific characteristics of its respective asset.

TABLE IV
Forecast Accuracy of Six Models of Volatility for Google

Model	RMSE	MAE	MAPE
Prophet	0.22	0.15	356.30
NNAR	0.22	0.15	495.99
ETS	0.21	0.15	439.41
LSTM	0.22	0.16	324.58
RNN	0.08	0.06	135.45
GARCH	0.02	0.01	97.48
Model Risk	11.00	16.00	5.09

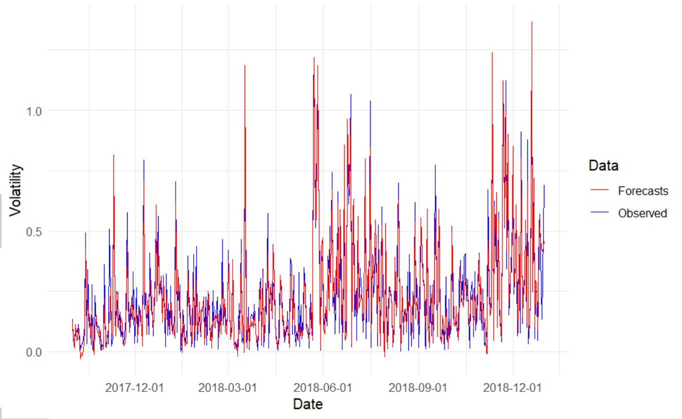

Fig. 5. Point forecasts of Volatility of Google from October 2017 to December 2018 using RNN

TABLE V
Forecast Accuracy of Six Models of Volatility for VIX

Model	RMSE	MAE	MAPE
Prophet	1.51	0.95	409.66
NNAR	1.64	1.00	398.40
ETS	1.54	0.99	419.55
LSTM	1.49	0.95	420.90
RNN	0.51	0.35	209.59
GARCH	0.12	0.08	92.02
Model Risk	13.67	12.50	4.57

As discussed in the introduction section, forecasts of conditional volatility are obtained by taking the square of the forecasted conditional variance. The GARCH models for squared log returns are used to first obtain the forecast of the conditional variance of the log returns, and then its square root is taken to obtain a forecast of the conditional volatility, for the computation of the risk measures such as VaR, ES, RMSE, etc. However, the square root of the variance estimate is an inefficient estimate of the volatility [8]. Hence, based

Preprint

TABLE VI
FORECAST ACCURACY OF SIX MODELS OF VOLATILITY FOR BITCOIN

Model	RMSE	MAE	MAPE
Prophet	0.90	0.64	935.74
NNAR	0.95	0.64	879.88
ETS	0.87	0.60	941.61
LSTM	0.88	0.64	673.31
RNN	0.25	0.18	185.34
GARCH	0.06	0.04	132.23
Model Risk	15.83	16.00	7.12

on the results from Tables IV - VI, even though GARCH has a smaller RMSE/MAE/MAPE, it is not reliable due to huge uncertainty associated with the large variance.

Tables IV-VI present the neuro model risk volatility forecasts for Google, VIX, and Bitcoin which shows the use of the neuro model risk ratio to measure the level of disagreement among the risk measures (RMSE, MAE, and MAPE) from different neuro volatility risk forecasting models. For example, model risk based on RMSE for Google is 11.00 (IV) indicates the degree to which different models used here disagree. We also examine the sensitivity of the results on the neuro model risk since one might employ different measures to quantify the disagreement among the alternative models other than RMSE, such as MAE and MAPE. Moreover, Figure 5 demonstrates the one-step ahead volatility forecasts of Google using the RNN model.

C. Forecast Accuracy and Model Risks for Temperature

We also investigate the performance of neuro model risk in terms of regression context. To that end, we explore the temperature forecast accuracy of different models and model risk ratios using temperature data from Ontario, Canada. It is important to note that since the time series cross-validation (evaluation on a rolling forecasting origin) has not been investigated for regression setup and thus, in this case, we used traditional training and test (say, first 80% as training and last 20% as test) data setup to obtain the forecast accuracy for regression models. In the temperature data, we consider data as the training set from the period 2018-01-01 to 2019-05-31 and the test set from the period 2019-06-01 to 2019-12-31.

Forecasts of a given time series can be obtained using popular base models/benchmark models such as the mean (average) model, naive model, seasonal naive model, and drift model. Even though they are simple and easy to implement with less computational burden, their performances are limited in practice as they fail to capture the complicated correlation structures in real-life data. Thus, to obtain more accurate forecasts, more advanced time models have been taken into account. In autoregression-type models, we forecast the variable of interest (asset price/electricity demand) using a linear combination of past values of the variable and values of some features (trading volume/temperature). Rather than using past values of the forecast variable, moving average models use past forecast errors. If we combine differencing with autoregression and a moving average model, we obtain a non-seasonal Autoregressive Integrated Moving Average (ARIMA) model. ARIMA models have been investigated in stock price predictions extensively. Nevertheless, the efficacy of ARIMA models is constrained since they overlook the seasonality inherent in data and are incapable of integrating additional factors that influence the target time series. Seasonal ARIMA models can be used to model a wide range of seasonal data and to include other relevant information along with information from past data, the dynamic regression models with ARIMA/SARIMA (DRSARIMA) errors are more appropriate and efficient [3]. Another simple model to obtain time series forecasts with the given feature values is the Prophet model [15]. It is an additive model which captures the non-linear trend, seasonality, and holiday effects. Even though it is robust to missing data and handles outliers and shifts in the trend, it requires time series that have strong seasonal effects and several seasons of historical data.

The investigation of temperature accuracy involved the examination of twelve distinct models, encompassing benchmark models such as Naive, Seasonal Naive (SNAIVE), and Drift, with corresponding test errors summarized in Table VII. While simpler models like Naive lack the capability to integrate additional features, more intricate models such as Prophet, ARIMA, DRSARIMA, NNAR, and time series linear model (TSLM) can incorporate regression-type time models tailored to the provided data. For Prophet, NNAR, and TSLM, two sets of models were fitted. For instance, Prophet.1 entails fitting a Prophet model for temperature considering its lag values, while Prophet.2 fits a regression-type model for temperatures by incorporating other meteorological features. Similarly, NNAR.1 and NNAR.2, TSLM.1, and TSLM.2 are defined. Additionally, the ETS model was also employed in this study. Table VII displays the forecast accuracy for all temperature models. Based on the findings in Table VII, NNAR.2 emerges as the most effective model for obtaining one-step-ahead temperature forecasts, described by NNAR$(p = 22, P = 1, k = 14)_7$, which incorporates 14 neurons in the hidden layer with the past 22 observations of temperature $(y_{t-1}, \ldots, y_{t-22})$ as inputs along with meteorological features to derive temperature forecasts. Moreover, the model indicates weekly seasonality $(m = 7)$.

TABLE VII
FORECAST ACCURACY OF TWELVE MODELS FOR TEMPERATURE

Model	RMSE	MAE	MAPE
NAIVE	2.69	2.02	139.42
SNAIVE	4.60	3.50	250.91
Drift	2.70	2.02	140.19
Prophet.1	3.94	3.03	207.51
Prophet.2	5.23	4.03	593.19
ARIMA	2.56	1.92	115.59
DRSARIMA	4.30	3.30	453.35
NNAR.1	3.12	2.32	169.47
NNAR.2	1.27	1.03	80.83
ETS	2.70	2.03	138.49
TSLM.1	11.66	10.28	464.04
TSLM.2	1.68	1.34	69.26
Model Risk	9.18	9.98	6.70

Preprint

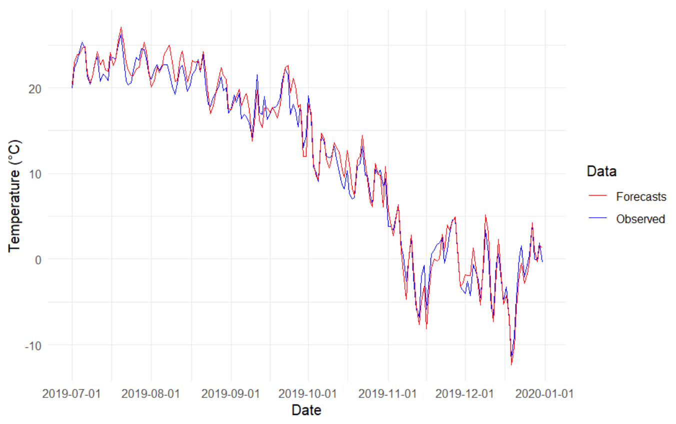

Fig. 6. Point forecasts of Temperature from June 2019 to December 2019 using NNAR.2

Table VII reports the neuro model risk forecasts for temperature which show the use of the neuro model risk ratio to measure the level of disagreement among the risk measures (RMSE, MAE, and MAPE) from different neuro volatility risk forecasting models. For example, model risk based on RMSE for Temperature is 9.18 (VII) indicates the degree to which different models (benchmark models, prophet, NNAR, TSLM, and others) disagree. We also used other risk measures such as MAE and MAPE to examine the sensitivity of the results on the neuro model risk since one might employ different measures to quantify the disagreement among the alternative models other than RMSE. Moreover, Figure 6 demonstrates the one-step ahead temperature forecasts using the NNAR.2 model which incorporates meteorological features such as precipitation, snowfall, snow mass, air density, and cloud cover fraction.

IV. Conclusions

There has been a growing interest in using Neruo volatility models in forecasting problems. Neuro volatility models typically refer to using neural network models to model and predict financial market volatility for nonlinear time series data. In financial risk forecasting, various risk forecasting models are used to obtain the volatility forecasts, and then the model risk ratio is calculated to assess the model risk. However, recently proposed neuro volatility forecasting models (based on neural networks such as LSTM and NNAR) have not been used in evaluating the model risk. In this paper, we propose a novel neuro model risk forecast based on the neuro volatility models and show that the proposed models can be used to check the resiliency of the financial system. The existing model risk ratio forecast is based on linear volatility models, whereas the proposed neuro model risk forecasts allow more appropriate nonlinear nonstationary neuro volatility forecasting models. Extensive experimental studies on price forecasts, volatility forecasts, and temperature forecasts show that the neuro model risk ratio forecast can effectively be used as a metric to assess the resilience and stability of a given financial system.

References

[1] Pala, Z., & Atici, R. (2019). Forecasting sunspot time series using deep learning methods. Solar Physics, 294(5), 50.

[2] Hoque, M. E., Bowala, S., Paseka, A., Thavaneswaran, A., & Thulasiram, R. (2023, June). Fuzzy Option Pricing for Jump Diffusion Model using Neuro Volatility Models. In 2023 IEEE 47th Annual Computers, Software, and Applications Conference (COMPSAC) (pp. 1349-1354). IEEE.

[3] Hyndman, R. J., & Athanasopoulos, G. (2018). Forecasting: principles and practice. OTexts.

[4] James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). An introduction to statistical learning (Vol. 112, p. 18). New York: springer.

[5] Almuammar, M., & Fasli, M. (2019, December). Deep learning for nonstationary multivariate time series forecasting. In 2019 IEEE international conference on big data (Big Data) (pp. 2097-2106). IEEE.

[6] Pirani, M., Thakkar, P., Jivrani, P., Bohara, M. H., & Garg, D. (2022, April). A comparative analysis of ARIMA, GRU, LSTM and BiLSTM on financial time series forecasting. In 2022 IEEE International Conference on Distributed Computing and Electrical Circuits and Electronics (ICDCECE) (pp. 1-6). IEEE.

[7] Thavaneswaran, A., Liang, Y., Das, S., Thulasiram, R. K., & Bhanushali, J. (2022, May). Intelligent probabilistic forecasts of VIX and its volatility using machine learning methods. In 2022 IEEE Symposium on Computational Intelligence for Financial Engineering and Economics (CIFEr) (pp. 1-8). IEEE.

[8] Thavaneswaran, A., Paseka, A., & Frank, J. (2020). Generalized value at risk forecasting. Communications in Statistics-Theory and Methods, 49(20), 4988-4995.

[9] Singh, J., Bowala, S., Thavaneswaran, A., Thulasiram, R., & Mandal, S. (2022, July). Data-Driven and Neuro-Volatility Fuzzy Forecasts for Cryptocurrencies. In 2022 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE) (pp. 1-8). IEEE.

[10] Thavaneswaran, A., Thulasiram, R. K., Zhu, Z., Hoque, M. E., & Ravishanker, N. (2019, July). Fuzzy value-at-risk forecasts using a novel data-driven neuro volatility predictive model. In 2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC) (Vol. 2, pp. 221-226). IEEE.

[11] Danielsson, J., James, K. R., Valenzuela, M., & Zer, I. (2016). Model risk of risk models. Journal of Financial Stability, 23, 79-91.

[12] Hewamalage, H., Bergmeir, C., & Bandara, K. (2021). Recurrent neural networks for time series forecasting: Current status and future directions. International Journal of Forecasting, 37(1), 388-427.

[13] Schäfer, A. M., & Zimmermann, H. G. (2006). Recurrent neural networks are universal approximators. In Artificial Neural Networks–ICANN 2006: 16th International Conference, Athens, Greece, September 10-14, 2006. Proceedings, Part I 16 (pp. 632-640). Springer Berlin Heidelberg.

[14] Danielsson, J. (2011). Financial risk forecasting: The theory and practice of forecasting market risk with implementation in R and Matlab. John Wiley & Sons.

[15] Taylor, S. J., & Letham, B. (2018). Forecasting at scale. The American Statistician, 72(1), 37-45.

V. Appendix

A. *Naive Model*

Consider an observed series denoted by y_t, where $t = 1, \ldots, T$. In the context of Naive forecasts, the approach entails assigning all forecasts the value of the most recent observation. That is,

$$\hat{y}_{t+h|t} = y_t, \;\; h = 1, \ldots, H.$$

A similar approach, referred to as the seasonal naive model, proves advantageous for datasets exhibiting prominent seasonal patterns. In this context, each forecast is aligned with the most recent observed value from the corresponding season. It's worth noting that, in the case of stock price and volatility data, seasonal patterns are seldom observed. Consequently, the

Preprint

paper does not offer accuracy measures for price volatility forecasts.

B. Drift Model

A revised approach to the basic method entails allowing forecasts to gradually increase or decrease over time, with the rate of change, referred to as drift, being determined by the average historical change in the data. Thus, the forecasts of time $t+h$ is given by

$$\hat{y}_{t+h|t} = y_t + \frac{h}{t-1}\sum_{t=2}^{T} y_t - y_{t-1} = y_t + h\left(\frac{y_t - y_1}{t-1}\right).$$

This is the same as drawing a line from the initial observation to the final one and extending it forward to predict future values.

C. Exponential Smoothing (ETS)

The calculation of forecasts employs weighted averages, where the weights diminish exponentially as observations extend further into the past — the oldest observations are associated with the smallest weights:

$$\hat{y}_{T+1|T} = \alpha y_T + \alpha(1-\alpha)y_{T-1} + \alpha(1-\alpha)^2 y_{T-2} + \cdots,$$

where $0 \leq \alpha \leq 1$ is the smoothing parameter. The one-step-ahead forecast for time $T+1$ is a weighted average of all the observations in the series $y_1, \ldots, y_T$. The pace at which the weights diminish is determined by the parameter α. For any α between 0 and 1, the weights assigned to the observations decrease exponentially as we move backwards in time, thus earning the label "exponential smoothing." A small α (i.e., close to 0) allocates more weight to observations from the distant past, while a large α (i.e., close to 1) assigns more weight to recent observations. In the extreme case where $\alpha = 1$, $\hat{y}_{T+1|T} = y_T$, and the forecasts align with the naïve forecasts.

D. GARCH Model

GARCH, or Generalized Autoregressive Conditional Heteroskedasticity, is a statistical model used to analyze and forecast volatility in financial time series data. It is an extension of the ARCH (Autoregressive Conditional Heteroskedasticity) model, introduced by Robert Engle in the 1980s. GARCH models are popular in financial econometrics for their ability to capture time-varying volatility, which is a crucial aspect of asset pricing and risk management. They specify a dynamic process for the conditional variance of a series, where volatility is modeled as a function of past squared residuals (errors) and past conditional variances. This framework allows GARCH models to capture the persistence and clustering of volatility commonly observed in financial markets. GARCH models are widely used in areas such as volatility forecasting, portfolio optimization, and derivative pricing.

Consider the general GARCH (p,q) model for the time series y_t given by

$$\begin{aligned} y_t &= \sqrt{h_t} Z_t, \\ h_t &= \omega + \sum_{i=1}^{p} \alpha_i y_{t-i}^2 + \sum_{j=1}^{q} \beta_j h_{t-j}, \end{aligned} \tag{2}$$

where Z_t is a sequence of independent and identically (i.i.d.) random variables with zero mean and unit variance. The conditional variance of y_t is h_t, a time-varying, positive and measurable function of the information set at time $t-1$. For the conditional variance to be positive, the parameters must satisfy $\omega > 0, \alpha_i \geq 0$ for $i = 1, \ldots, p$, and $\beta_j \geq 0$ for $j = 1, \ldots, q$.

Preprint