الجمهورية الجزائرية الديمقراطية الشعبية
The People's Democratic Republic of Algeria
وزارة التعليم العالي والبحث العلمي
Ministry of Higher Education and Scientific Research
جامعة محمد بوضياف بالمسيلة
University Mohamed Boudiaf of M'sila

كلية الرياضيات والإعلام الآلي
Faculty of Mathematics and Informatics

قسم الإعلام الآلي
Department of Computer Science

Domain: Mathematics and Computer Science

Thesis Presented to Fulfill the Partial Requirement
for **Master's Degree** in Computer Science

Specialty: Artificial Intelligence

Prepared By: Amari Mohammed Elamin

Supervised By:

Prof. Sayad Lamri

ENTITLED

Forecasting Volatility of Cryptocurrencies Using Machine Learning and Deep Learning

Jury Members

Mustapha Bourahla	President
Lamri Sayad	Supervisor
Salah Guesmia	Examiner

Academic Year 2024/2025

DEDICATION

"*Without a family, man, alone in the world, trembles with the cold*"—André Maurois.

To my parents and family.

To my grandmother.

One more milestone reached thanks to you.

Hopefully not the last.

ACKNOWLEDGEMENT

I would like to express my gratitude to my supervising teacher **Prof. Lamri Sayad,** for his guidance, patience, and insightful feedback—All of which were vital for the completion of this work.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

GENERAL INTRODUCTION

In financial markets, the ability to predict the market movements, at least to an acceptable degree of accuracy, is crucial for investors and financial institutions to make well-informed decisions that could increase gains or reduce losses. The use of traditional statistical models has proven fruitful for years in the context of traditional financial assets. However, the introduction of cryptocurrencies, and the subsequent emergence of cryptocurrency market, proved a great challenge for these traditional models.

Cryptocurrencies are digital assets that rely on blockchain technology to record and verify transactions across a network of computers. Unlike fiat currencies, cryptocurrencies are not subject to the oversight and control of official financial institutions, like central banks, for their issuance or traffic. This independent, decentralized nature of these novel assets gave them some advantages that appealed to the public, as well as investors and financial institutions: financial inclusion—anyone can access them through only simple internet connection, low transaction fees, transaction speed—they can be settled in a matter of seconds, transparency and security—they are immutable and publicly verifiable, little to no inflation risk because of market cap, and true ownership—users hold private keys to their assets with no risk of fund seizure or freezing. But even when they are called crypto-*currencies*: "*Some scholars consider them just speculative assets rather than to be called currencies at all. Yermack D (2015)*". [1]

Nevertheless, after the successful release of **Bitcoin** in 2009, and the subsequent introduction of several other cryptocurrencies—each with unique use cases and technological characteristics, the new emerging market, that grew rapidly into a global market despite its novelty, has proven to be highly erratic and unstable in comparison with the traditional financial market. This became a concern for traders and investors —the traditional models developed specifically to forecast the traditional financial assets couldn't handle the highly volatile nature of the new market with the same degree of success. However, this also presented an opportunity to achieve great investment success, because "*the riskier the market, the higher the reward*". [1] New, more accurate models to handle the challenges of this novel market have become crucial for these investors and traders, and this coincided with the rise of **Artificial Intelligence (AI)** with more sophistication and potential.

Machine learning (ML) and deep learning (DL) techniques, which both fall under the umbrella of (AI), have gained increasing attention in the financial domain due their proven ability to discover patterns and relationships, especially non-linear and complex ones in financial datasets.

Given the distinct nature of cryptocurrency market, these abilities hold promise. But, while several studies have already demonstrated the ability of ML and DL models in predicting the prices, only few were concerned primarily with forecasting the cryptocurrencies market volatility. Accurate volatility predictions are critical for informed decision making concerning future investments and economic ventures by financial actors. This require an intensive demonstration of both ML and DL tools' ability to handle cryptocurrencies highly volatile market nature and provide accurate forecasts, or at least accurate enough: "*A trader does not need to make perfectly accurate forecast to have a positive expectation when participating in the markets, he/she just needs to make a forecast that is more correct than the consensus*". [2]

While they are both AI tools, ML and DL models differ in their strategy, architecture, complexity, and even interpretability. The research undertaken in this thesis addresses a central question: How do machine learning and deep learning models, specifically tree-based machine learning and RNN deep learning models, perform in forecasting cryptocurrency volatility and Which approach—tree-based or RNN—offers superior prediction performance for forecasting volatility for major cryptocurrencies (Bitcoin and Ethereum) 24-hours ahead? Answering this question involves the examination of several models, both ML and DL, their accuracy, generalizability, and hyperparameter optimization impact on their performance, using evaluation metrics (MAE, MSE, RMSE, R^2) and diagnostic checks.

The core hypothesis of this thesis is that deep learning models, particularly Recurrent Neural Network models such as LSTM and GRU, will outperform the traditional tree-based machine learning models, represented by Random Forest and XGBoost. The hypothesis is grounded in the premise that RNN models are structured to capture temporal dependencies and are well-suited to handle rich, sequential, and vast datasets.

This thesis is organized into five main sections:

- **Introduction**: Where the research context, problematic, objective, and the plan are announced.
- **Literature Review:** Clarifies main concepts and summarizes the previous works on the subject.
- **Methodology:** Details the steps and procedures followed in the research from data collection and preprocessing to model testing and evaluation.
- **Results and Discussion:** Presenting the results, analysis, interpretation, comparison with other studies, diagnostic checks, discussion of findings, and limitations of the research.
- **Conclusion:** Recaps the research objective, summarizes the findings, acknowledges the limitations, and suggests future research directions.

CHAPTER 1: LITERATURE REVIEW

1.1. Introduction:

This chapter begins with a succinct definition of cryptocurrencies, characteristics of cryptocurrency market, and the concept of cryptocurrency volatility. Then, it outlines the traditional or classical statistical models, reviews some studies about statistical models, delves into theoretical aspects of the machine/deep learning models of this thesis, then reviews studies that compares the traditional and the ML models. Finally, it delves into a review of studies that focused mainly on ML/DL models in forecasting the volatility of cryptocurrencies.

1.2. Definition of Cryptocurrencies:

The term *Cryptocurrency* refers to a set of digital, virtual coins, that are based on blockchain technology and secured by cryptography, created as an alternative to fiat currencies (traditional money). Unlike traditional financial assets that are backed by real companies, governments, and physical assets, these digital assets "*does not have any intrinsic value*" [1], except value based on supply, demand, and peer-to-peer trust. They are "*a medium of exchange, created and stored electronically on the blockchain, using cryptographic techniques to verify the transfer of funds and an algorithm to control the creation of monetary units*". [53]

As for blockchain, it is a digital ledger that uses **consensus mechanisms** to verify and secure transactions, and create new coins. Two most common consensus mechanisms are **proof of work** and **proof of stake**. [54] Proof of work (PoW) relies on the process of solving complex mathematical problems that require computational intensity, which enforces security and discourages fraud, while in proof of stake (PoS), validators stake an amount of cryptocurrency as collateral in order to validate transactions. [55]

1.3. Characteristics of Cryptocurrency Market:

As a novel asset class, cryptocurrencies exhibit unique characteristics that set it apart from traditional financial assets. Among those distinct features we find:

- **High Volatility:** Describing cryptocurrencies volatility as 'high' is an understatement. This extreme fluctuation both poses a threat and provides an opportunity to investors. Daily 5-10% or more price swings are not rare due to several factors like: speculation, lack of

regulation, market novelty and therefore immaturity, and high sensitivity to news and social media.

- **Non-regulation and Decentralization:** Unlike traditional financial assets, cryptocurrencies are not issued by official or governmental institutions like central banks, which significantly reduces oversight and leads to an unstable and sometimes erratic market.
- **Sentiment sensitivity:** the cryptocurrency market can be easily acted upon from media, social media platforms, and public sentiment. News, rumors, and opinions can cause big price surges and drops in a small timeframe.
- **24/7 Trading:** conventional markets have defined trading hours, while cryptocurrency markets operate 24 hours a day, 7 days a week, potentially affecting volatility patterns. [10]
- **Relatively Young Market:** The novelty of this market results in a limited historical data.

1.4. Cryptocurrency Volatility:

1.4.1. Definition of Volatility:

"*In quantitative finance, volatility refers to the conditional standard deviation (or conditional variance) of the underlying asset returns (Lahmiri et al., 2018)"*. [3] Alternatively, it is the degree of variation in asset prices over time. It is a central concept in financial risk management and other aspects of financial markets. High volatility periods indicate significant price fluctuations and increased market uncertainty, while low volatility suggests more stable and predictable price movements. [4]

Typically, the volatility of cryptocurrencies is measured using:

- **Realized or Historical Volatility:** Measures past price movements during a period of time, which provides an insight into how volatile an asset has been.
- **Implied Volatility:** Reflects the expectations of volatility in the future.

1.4.2. Why Are Cryptocurrencies So Volatile:

Compared to traditional financial assets, cryptocurrencies are highly volatile, which makes accurate fluctuation predictions essential for investors and institutions: "*Forecasting volatility in finance is a complex task, and capturing the sensitive nature of cryptocurrencies is hard in both finance and economics*". [1] "*Giudici and Abu-Hashish (2019) confirmed cryptocurrency markets as more volatile than traditional foreign exchange markets. Chu et al. (2019) also reported considerable volatility in the price of cryptocurrencies. Likewise, Bouri et al. (2017) validated the footprints of greater volatility in crypto-currencies than their counterparts, i.e., the traditional foreign exchange market*". [1]

This high volatility is attributed to the following key factors:

- **Speculation:** Buying and selling cryptocurrencies is based on price expectations rather than the asset's intrinsic value. This market is reactive to news, hype, and fear which causes sharp swings driven by sentiment and emotion.
- **Illiquidity:** "*Liquidity contributes to a stable market by buffering against extreme price swings and market manipulation. Because many cryptocurrencies are generally illiquid, they are also more vulnerable to global news events that impact on risk-on assets"*. [56]
- **Market Novelty:** The relatively novel state of cryptocurrency markets means they are more susceptible to large price swings due to lower liquidity and market depth. [57]
- **Regulations:** No global regulatory body of cryptocurrencies, which means that regulations, or the lack of thereof, varies from nation to nation. This affects the volatility when, for example, a large nation decides to restrict or totally ban cryptocurrency trading. [56][57]
- **Macroeconomic Events:** "*such as geopolitical tensions, economic crises, or natural disasters, can fuel volatility in the cryptocurrency market. Events like these often create uncertainty and fear, leading investors to re-evaluate their risk appetite and potentially withdraw capital from cryptocurrencies*". [57]

We distinguish two types of approaches to volatility modeling: traditional, such as ARCH/GARCH, and machine learning techniques.

1.5. Traditional Volatility Forecasting Methods:

1.5.1. ARCH and GARCH Models:

Autoregressive Conditional Heteroskedasticity (ARCH) proposed by **Robert F. Engle** (1982) and its generalization, namely (GARCH) proposed by **Tim Bollerslev** (1986), are among the most popular models for forecasting time series.

Autoregressive means the model looks at past data to predict current data. **Conditional** means it adjusts its predictions based on recent information. **Heteroskedasticity** refers to the fact that the volatility of data isn't constant—it changes over time.

1.5.1.1. How ARCH/GARCH Works:

The main concept that needs to be explored is the concept of **Shock**. It is the difference between the actual return and the expected return.

$$\text{Shock} = \text{Actual return} - \text{Expected return} \tag{1.1}$$

In other words, Shock is the unexpected part of returns—the surprise component that wasn't predicted by the model. It is also called **Residual** or **Error**.
Volatility is, in its part, a function of squared shocks.

$$\sigma_t^2 = \alpha_0 + \alpha_1 . \varepsilon_{t-1}^2 + \alpha_2 . \varepsilon_{t-2}^2 + \cdots + \alpha_q . \varepsilon_{t-q}^2 \quad (1.2)$$

Where:

- σ_t^2 = forecasted variance (volatility) at time t
- ε_{t-i}^2 = **squared shock** from the past
- $\alpha_0, \alpha_1, \ldots, \alpha_q$ = parameters estimated by the model
- q = number of past shocks we look at. [5]

But in order to have a good forecast, q needs to be very large—many past shocks are needed. This makes the model complicated, and on top of that, it doesn't account for the effect of volatility itself. This shortcoming was fixed by introducing GARCH.

GARCH takes past volatility into consideration, which adds a **memory** component to the model, making it more efficient and accurate.

$$\sigma_t^2 = \alpha_0 + \alpha_1 . \varepsilon_{t-1}^2 + \beta_1 . \sigma_{t-1}^2 \quad (1.3)$$

Where:

- σ_{t-1}^2 = last period's variance (volatility)
- β_1 = new parameter for how persistent volatility is. [5]

1.5.1.2. GARCH-type Models:

GARCH-type models are a basic class of econometric models used in modeling and forecasting volatility of financial time series. The strength of GARCH-type models is their ability to capture volatility clustering—periods of high (or low) volatility tend to be followed by periods of high (or low) volatility. GARCH-type models are efficient and include structures for conditional variance that include both past squared shocks and past volatility.

Since the basic GARCH (1,1) model was developed, many different extensions of the GARCH-type model have since been proposed to help mitigate shortcomings of the original model. The original GARCH-type model does not consider asymmetric effects from shocks (for example, leverage effect), so EGARCH (Exponential GARCH) has been introduced. GJRGARCH (Glosten-Jagannathan-Runkle GARCH) has also been introduced to address the issue of modeling the different effects of both positive and negative shocks on volatility.

All of these modifications enhance the suitability of GARCH-type models for containing features of financial data found in the real-world, for example; asymmetric returns, and volatility clustering.

1.5.2. Stochastic Volatility Models:

While volatility in ARCH/GARCH models is treated as a deterministic function of past squared shocks, it is modeled in SV models as a **latent stochastic process.** This means that while it cannot be directly measured (hidden and randomly changing value), we can infer its existence from directly observing its effects. [6]

Because of this, estimating the model parameter is more challenging than that of GARCH models, demanding the implementation of techniques like **Markov Chain Monte Carlo** (MCMC). [7]

The canonical stochastic volatility model is: [8]

$$y_t = e^{h_t/2}\varepsilon_t \tag{1.4}$$

with the latent log-volatility process defined by:

$$h_t = \mu + \varphi h_{t-1} + \tau\eta_t \tag{1.5}$$

Where:

- y_t = observed return.
- h_t = log (σ_t^2): latent log-volatility process ($e^{h_t/2} = \sigma_t^2$).
- ε_t = Standard normal shock.
- μ = The long-term mean of the log-volatility.
- φ = Persistence parameter ($|\varphi| < 1$).
- τ = The volatility of the volatility.
- η_t = standard normal random shock, applied to the log-volatility.

SV models' approach to volatility helps them to be a better match to how real markets work. Especially in cryptocurrency market, where studies have proven that they excel over GARCH-type models due to their ability to capture complex volatility dynamics: "*The excellence of forecasting power in our SV model provides implications that it can be used as a better risk management tool than other GARCH family models. Moreover, the forecasting errors of the SV model, compared with the GARCH models, tend to be more accurate as forecast time horizons are longer*". [9]

1.6. The Application of Traditional Volatility Forecasting Models to Cryptocurrencies:

While traditional models have been proven to be successful in forecasting the volatility of traditional financial assets, it's not necessarily true that the same should apply to cryptocurrencies. Several studies have been conducted in the past several years applying traditional volatility models to cryptocurrencies.

For example, Chu et al. [11] concluded that traditional volatility models (especially IGARCH and GJRGARCH) are useful in predicting cryptocurrency volatility. Also, the paper supports increased regulation, noting growing global oversight, especially by the U.S. SEC, China, and South Korea.

Kim et al. [9] analyzed 10 cryptocurrencies and found out that the SV model outperformed GARCH in forecasting accuracy, especially over longer horizons and in highly volatile conditions.

Although these studies and others have concluded that traditional volatility models can capture some aspects of cryptocurrency's volatility, the unique features exhibited by this new financial asset may require the use of enhanced versions or even the introduction of entirely new methods.

1.7. Limitations of Traditional Volatility Forecasting Models:

Maciel [12] assessed that GARCH models do not account for regime shifts that occur due to external factors like regulatory changes or macroeconomic events, which results in poor forecasts during such periods.

Sun and Krištoufek [13] found out that GARCH models need high-frequency data for better volatility prediction. The unavailability of such data may result in these models underperforming even in comparison with simpler methods like Garman-Klass estimator. Pruser [14] argues that SV models perform better than GARCH models, but they require more intensive computation and may depend on Bayesian estimation methods.

Chen et al. [15] and Khaldi et al. [16] determined that standard GARCH models are not reliable in capturing the leverage effect, where negative returns lead to higher volatility. Modified versions like EGARCH and TGARCH can address this, but they require careful parameter selection and may sometimes underperform when compared to simpler models.

Wang et al. [17] demonstrated that machine learning models, such as LSTM (Long Short-Term Memory) and Random Forest, often outperform GARCH models in volatility forecasting. This is attributed to the ability of machine learning models to capture non-linear patterns and complex interactions in the data. While Derbentsev et al. [18] noted that a primary distinguishing feature

separating machine learning from traditional statistical modelling is the concept of how data is used. Specifically, machine learning algorithms assume that the algorithms can explore raw (or seemingly unstructured) data, extract patterns and associations, and produce useful inferences without the need to be decomposed beforehand or conduct hypothesis testing. This means that machine learning algorithms assume that they can extract predictive logic directly from the data itself, which facilitates the modelling process.

"GARCH models, while designed for volatility, in most cases cannot cope with such magnitude and frequency of the fluctuation in cryptocurrency markets (Sumon et al.,2024)". [19]

Table 1.1 Summary of Studies on Bitcoin Volatility Forecasting with Their Best Score Traditional Model

AUTHOR(S)	TARGET	DATASET	MODELS	EVALUATION METRICS
Chi Bui [2]	Realized Volatility	BTC 2014-2021 1-day interval	TARCH (1,2,0)	RMSPE: 0.2015 RMSE: 0.0668
Kim et al. [9]	Realized Volatility	BTC 01-2018 to 01-2019 1-day interval	TARCH Horizon = 3 days	MSE: 5.288
			SV Horizon = 3 days	MSE: 1.646
Amirshani & Lahmiri [3]	Realized Volatility	BTC 2018 - 2022 1-day interval	GARCH (GED distribution)	RMSE: 0.00458
Pratas et al. [20]	Realized Volatility	BTC 2014-2022 1-day interval	ARCH (4) Horizon = 3 days	MAE: 0.00155 MAPE: 194.45%
Wang et al. [17]	Realized Volatility	BTC 1-11-2017 to 31-07-2022 30-min interval	GARCH Horizon = 1 day	RMSE: 0.014 MAPE: 38.02

1.8. Machine Learning/Deep Learning Volatility Forecasting Methods:

Machine learning and deep learning techniques are increasingly being used for financial time series forecasting. Although these techniques are not yet fully adept at forecasting volatility in time series where the underlying variables are highly nonlinear, non-stationary, and noisy, they still hold promise. With machine learning and deep learning, there is an opportunity to utilize a wide range of models. From classical machine learning algorithms (such as Random Forests, Support Vector Machines, and Gradient Boosting Machines) to deep learning architectures (including Recurrent Neural Networks (RNN), Long Short-term Memory (LSTM) Networks, Gated Recurrent Units (GRU), and Temporal Convolutional Networks (TCN), to name but a few. These algorithms have the benefit of being flexible, data-driven, can capture the complexities of temporal dependencies and nonlinear relationships which have been difficult to capture using conventional econometric approaches (such as GARCH-type approaches).

"*Some of the key benefits of integrating machine learning-driven predictions into financial systems are better accuracy, speed, and scalability. Unlike the traditional models, which are often adjusted manually, the machine learning models automatically self-adjust with new data. This ensures that they remain relevant even when market conditions change. Such systems can also process large volumes of data in real time, including unstructured sources such as social media sentiment or blockchain transaction data, to provide a far more holistic view of the market*". [19]

We are keen to highlight this rapidly growing area of research into the application of machine learning and deep learning approaches to volatility forecasting in the following review of the literature and key methodologies, datasets, evaluation metrics and performance in comparison to conventional methods.

1.8.1. Traditional Machine Learning Models:

The machine learning paradigm selected for this thesis is **Ensemble Leaning**. It is the process of combining multiple base or weak learners to produce a better or more accurate prediction compared to individual models. It utilizes techniques like bagging, boosting, and stacking. Multiple studies have confirmed that this paradigm is advantageous in the field of forecasting cryptocurrency market. [21] [22] [23] Two models are selected: Random Forest and XGBoost.

1.8.1.1. Random Forest:

Random Forest (RF) is a powerful ensemble learning method for time-series forecasting that works by building multiple decision trees using bagging and random subspace selection in order to reduce prediction variance and prevent overfitting by increasing model diversity. RF uses block

bootstrap sampling to account for temporal dependencies. The final output is computed by averaging the weighted predictions of trees across bootstrap samples. [17]

"*Bagging is especially useful in combination with tree models that are sensitive to changes in training data. In the RF algorithm, bagging is combined with the method of random subspaces: that is, each tree is built on different randomly selected subsets of features—this process is called subspace sampling. The random subspaces method reduces the correlation between trees and avoids retraining because the basic algorithms are trained on different subsets of traits, which are also randomly selected*". [24]

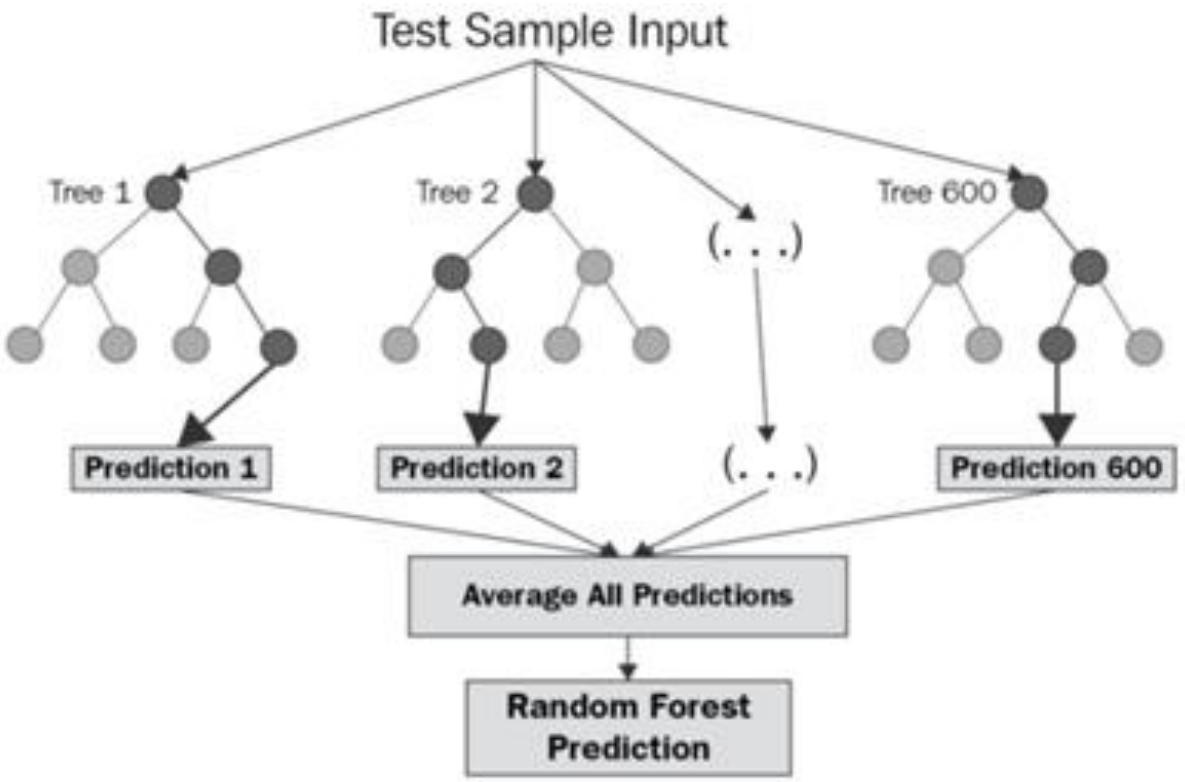

Figure 1.1 Ensemble Prediction Process in Random Forest Regression [25]

The equation is like this:

$$\hat{y} = \frac{1}{B}\sum_{b=1}^{B} T_b(x) \tag{1.6}$$

Where:

- $\hat{y}$ is the final predicted value,
- B is the total number of trees in the forest,
- $T_b(x)$ is the prediction of the b^{th} decision tree for input x.

While The formula for $T_b(x)$ is:

$$T_b(x) = \frac{1}{N_b(x)} \sum_{i \in L_b(x)} y_i \quad (1.7)$$

Where:

- $L_b(x)$ is the set of training instances in the leaf node where input x ends up in tree b,
- y_i is the target value of training instance i,
- $N_b(x)$=$|L_b(x)|$ is the number of training instances in that leaf.

1.8.1.2. XGBoost (Gradient Boosting):

Despite being both Ensemble Learning methods, the difference between Random Forest and Gradient Boosting lays in the process of prediction. While the output of Random Forest is the average of all tree models' predictions, Gradient Boosting gradually and repeatedly trains models so the next one learns from the errors of the one before. In the end, all the models are combined into a stronger one with much better predictions.

XGBoost is a version of Gradient Boosting, and was designed to be faster and more accurate especially when dealing with large datasets, or when overfitting might be a concern.

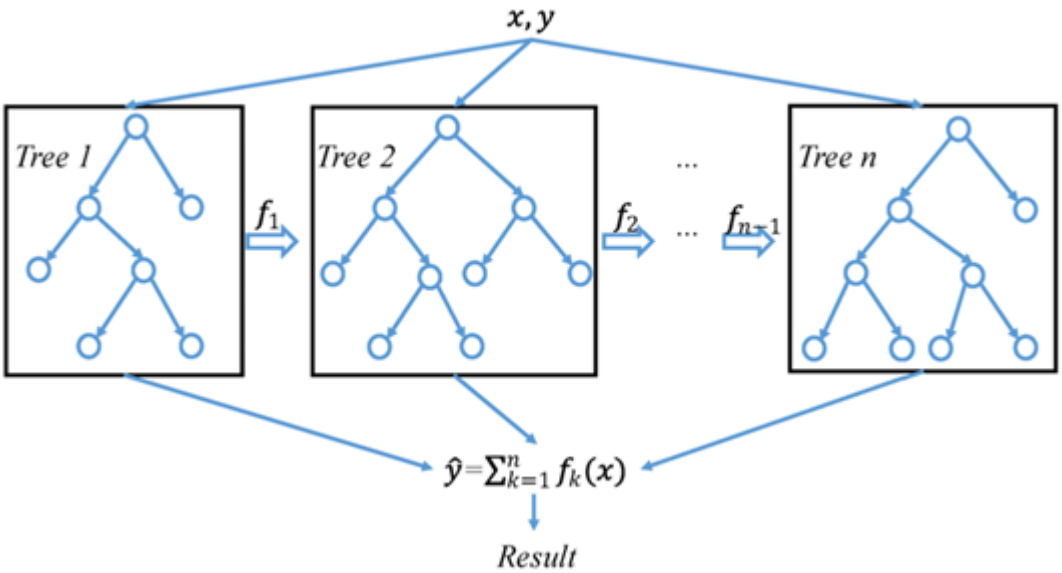

Figure 1.2 A general architecture of XGBoost [27]

The output is expressed generally as such: [26]

$$y = \sum_{k=1}^{N} f_k(x_t) \quad (1.8)$$

Where:

- y predicted output,
- f_k the score from the k-th regression tree,
- x_t the input feature at time t,

- N is the total number of trees used in the model.

1.8.2. Deep Learning Models:

RNNs, or Recurrent Neural Networks are a type of artificial neural networks (ANNs) that have succeeded in domains where temporal context is important, which is the case with forecasting based on time series data. LSTM and GRU are both advanced RNN variants that are commonly used and extensively researched and validated in different studies. [28] [29]

1.8.2.1. LSTM (Long Short-Term Memory):

It is a RNN that was designed to work with sequential data and to handle the vanishing gradient issue described by Bengio et al. (1994). [3] An LSTM consists of a **cell state (C)**, a **hidden state (h)**, and three gates: **input gate** that decides what information to keep, **forget gate** that decides what information to discard, and **output gate** that determines the hidden state based on the cell state. [17]

These gates operate on a sigmoid activation function to manage the flow of information:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{1.9}$$

Where the cell candidate state activation function is:

$$tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{1.10}$$

LSTM equations per each time step:

Given:

- x_t: input at time t,
- h_{t-1}: previous hidden state,
- c_{t-1}: previous cell state,
- W, U, b: weights and biases.

The equations are:

1. Forget gate:

$$f_t = \sigma\left(W_f x_t + U_f h_{t-1} + b_f\right) \tag{1.11}$$

2. Input gate:

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \tag{1.12}$$

3. Candidate cell state:

$$\tilde{c}_t = tanh(W_c x_t + U_c h_{t-1} + b_c) \tag{1.13}$$

4. New cell state:

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \quad (1.14)$$

5. Output gate:

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o) \quad (1.15)$$

6. Hidden state:

$$h_t = o_t \odot \tanh(c_t) \quad (1.16)$$

Where $\odot$ denotes element-wise multiplication.

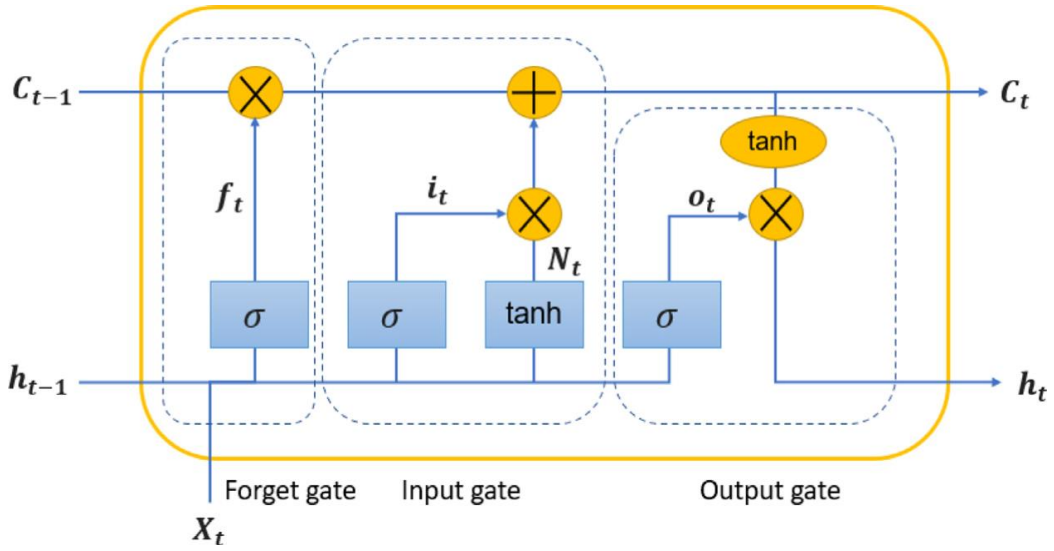

Figure 1.3 LSTM Unit Structure [3]

1.8.2.2. GRU (Gated Recurrent Unit):

The Gated Recurrent Unit (GRU) is a simplified alternative to LSTM that was introduced by Cho et al. (2014). While LSTM has three gates: input, forget, and output; GRU has only two: **update** and **reset**, where update merges both input and forget gates in LSTM. It also has one hidden state and no cell state. This simplification renders GRU computationally faster and less complex, but often on bar with LSTM in terms of performance. [30]

GRU equations per time step are:

Given:

- x_t: input at time t,
- h_{t-1}: previous hidden state,
- W, U, b: weights and biases.

The equations are:

1. Update gate:

$$z_t = \sigma(W_z x_t + U_z h_{t-1} + b_z) \quad (1.17)$$

2. Reset gate:

$$r_t = \sigma(W_r x_t + U_r h_{t-1} + b_r) \tag{1.18}$$

3. Candidate hidden state:

$$\tilde{h}_t = tanh(W_h x_t + U_h(r_t \odot h_{t-1}) + b_h) \tag{1.19}$$

4. Final hidden state:

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t \tag{1.20}$$

Where:

- σ: sigmoid activation function
- tanh: hyperbolic tangent activation
- ⊙: element-wise multiplication

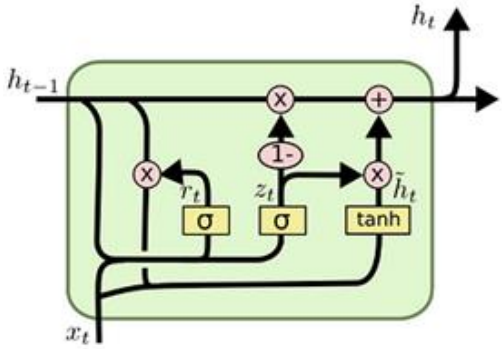

Figure 1.4 GRU structure [31]

1.8.3. Comparative Studies of Traditional vs Machine Learning Models:

Pratas et al. [20] concluded that generally, deep learning models outperform traditional models, especially over short time horizons. By comparing classical volatility models (ARCH, GARCH) with DL models (MLP, RNN, LSTM) for Bitcoin forecasting, the study found that ARCH (4) and GARCH (4,2) models achieved competitive accuracy when evaluated using MAPE. However, the MLP and RNN models proved to be superior in capturing nonlinear dynamics, but required greater computational resources. Interestingly, the LSTM model was the worst performing model of the DL models in this study.

Wang et al. [17] examined machine learning methods, like Random Forest and LSTM models optimized with Artificial Bee Colony methods (ABC), employed to forecast cryptocurrency volatility using both internal (e.g: lagged volatility) and external (e.g: financial, technological, and

policy uncertainty) determinants. The results lead to the conclusion that ML models outperformed traditional GARCH models in a notable way, with ABC-optimized LSTM giving the best accuracy. Also, it is important to note that models incorporating data from multiple cryptocurrencies performed better than models using a single asset, and internal determinants were the most influential in driving forecasts.

The study conducted by Dudek et al. [32] comparatively evaluated statistical and ML models for forecasting both the daily and weekly volatility of a set of coins: Bitcoin, Ethereum, Litecoin, and Monero. The models performances differed due to the variety of cryptocurrencies, metrics, and forecasting horizons. Therefore, no universally superior model was apparent. Interestingly, Random Forest and LSTM did not outperform simpler traditional models in a notable way, while Linear SVR (Support Vector Regression) ranked among the best for the daily forecast and both FNM (Fuzzy Neighborhood Model) and ridge regression were the leading models in weekly forecast. Among the findings of this study is that traditional GARCH models were less effective than ML models, and that using ensembling and hybridization techniques didn't provide significant improvement.

A comparative analysis by Nadarajah et al. [33] of volatility forecasts for Bitcoin, Ethereum and Litecoin showed that, of the models used: all GARCH-type, tree-based ensemble methods and ANFIS (Adaptive Neuro-Fuzzy Inference System) model, that XGBoost produced the most consistently accurate forecasts on both the in-sample and out-of-sample periods. While GARCH methods are relatively simple and interpretable, they are a relatively strong competitive benchmark and captured some key aspects of volatility. Also, the hybrid ANFIS model does have potential, but further refinement and enhancement using advanced neural architecture and enhanced fuzzy systems is required. Overall, the results of this study reaffirm the effectiveness of ensemble learning in forecasting cryptocurrency volatility while emphasizing, at the same time, the continued relevance of classical approaches.

Huang et al. [34] explore the effectiveness of machine learning models in forecasting Bitcoin volatility by comparing traditional models, such as GARCH and HAR, to neural network-based models including LSTM and a hybrid CNN-LSTM model. Using high-frequency data (2011–2021), they calculate realized volatility at 10-minute intervals. Their models predict volatility in different horizons (from 1 day to 60 days). The study finds that despite the good performance of the HAR model in short-term forecasts, the CNN-LSTM model consistently outperforms GARCH-type models and simple LSTM model, particularly in 7-day forecasts. The results underline the potential of hybrid deep learning methods in capturing complex, nonlinear volatility patterns in cryptocurrency markets.

"*The main difference between ML and classical modeling is that the Machine Learning algorithms interpret the data themselves, so there is no need to perform their initial decomposition. Depending on the purpose of the analysis, these algorithms "build" logic modeling based on the available data. This avoids the complex and lengthy pre-model stage of statistical testing of various hypotheses*". [24]

1.8.4. Studies of Machine Learning Application to Cryptocurrency Volatility Forecasting:

Zheng [35] used machine learning methods in Python to examine Bitcoin's volatility and its economic impact and underlined its potential to disrupt finance as we know it. The study confirmed that Bitcoin is very volatile, due primarily to its decentralized structure and blockchain basis, and while it presents opportunities for investment, Bitcoin also represents systemic risks. Bitcoin created modern ways to transact business, but it also brought with it the capacity to undermine current regulations in finance and the overall stability of the sector. The study highlighted the need for further adaptive regulatory measures and research to balance risks and investment rewards in the evolving cryptocurrency sector.

Khan et al. [1] examined the performance of three machine learning models: the cubic smoothing spline (CSS), neural network autoregressive (NNETAR), and group method of data handling neural network (GMDH-NN) for forecasting return volatility on Bitcoin, Ethereum, XRP, and Tether using daily data from 2017-2020. The overall results show that no single model performed the best for all cryptocurrencies involved. CSS had the best accuracy when predicting return volatility for Bitcoin and XRP, while NNETAR and GMDH-NN outperformed others in prediction of Ethereum and Tether volatility respectively using Mean Absolute Error (MAE) and Root Mean Square Error (RMSE). This study illustrates that the development of volatility forecasting models must be tailored to the characteristics of each individual cryptocurrency asset.

The capabilities of deep learning algorithms, particularly DFFNN (Deep Feed Forward Neural Network) and LSTM, paired with GARCH-type models was the subject of Amirshahi & Lahmiri [3] study. The objective was to forecast volatility for 27 cryptocurrency assets as well as average pricing across all cryptocurrencies. By using the forecasted outputs for GARCH, EGARCH, and APGARCH as input features for the deep learning models, the resulting hybrid model outperformed purely parametric or deep learning models by themselves.

Brauneis & Sahiner [26] explored the integration of AI based investor sentiment from crypto-specific news with machine learning models to forecast 6-hour realized volatility for the major eight cryptocurrencies. While traditional HAR models did not improve with the inclusion of sentiment, some of the ML models (LightGBM, XGBoost, and LSTM) improved predictive

performance in 54.17% of cases. This research concluded that combining sentiment analysis and ML has significant promise for risk management and forecasting performance.

Mjoska et al. [36] investigates the integration of blockchain technology and machine learning by utilizing the Random Forest algorithm to predict Bitcoin volatility. The model is trained by utilizing real-time Bitcoin blockchain. The model achieved low mean absolute errors in both training and testing which indicates a good predictive performance. This study proves that features derived from blockchain data can improve accuracy when forecasting volatility, outperforming autoregressive (AR) and vector autoregressive (VAR) models. The authors also notes that prediction error increases with higher volatility and over longer forecast horizons. They suggest testing other ML algorithms, tree depths, and forecast windows in the future could help further enhance model performance.

Tang et al. [37] found that the predictive power of RFSV+QRH (Rough Fractional Stochastic Volatility + Quantile Regression Hybrid) model using only a few parameters, was equivalent to that of LSTM models. Additionally, The LSTM model was demonstrated to be robust in capturing asymmetric volatility patterns.

Table 1.2 Summary of Studies that Applied ML Models to Forecast Cryptocurrencies Volatility

AUTHORS	DATASET*	MODELS**	EVALUATION METRICS***
Amirshahi & Lahmiri [3]	From 7-2-2018 to 7-2-2022	E-DFFNN + EGARCH	RMSE: 0.00737
		AP-LSTM + APGARCH	RMSE: 0.01115
Dudek et al. [32]	2017-2022	MLP	R^2: 0.202
			MAE: 0.00103
		RF	R^2: 0.111
			MAE: 0.00106
		LSTM	R^2: 0.143
			MAE: 0.00114
Wang et al. [17]	1-11-2017 to 31-07-2022 30 min- interval	RF	RMSE: 0.012
		ABC-LSTM	RMSE: 0.013
Chi Bui [2]	9-2014 to 8-2021	Bi-LSTM	RMSE: 0.0483

Pratas et al. [20]	07-09-2014 to 01-05-2022	MLP	MAE: 0.00003
		RNN	MAE: 0.0002
		LSTM	MAE: 0.00043
Khan et al. [1]	14-04-2017 to 30-10-2020	NNETAR	MAE: 0.002
			RMSE: 0.015
Nadarajah et al. [33]	01-01-2017 to 31-01-2024	XGBM	RMSE: 0.000306
Mjoska et al. [36]	14-01-2019 to 12-01-2022	RF	MAE: 0.015
Brauneis & Sahiner [26]	21-12-2021 to 22-12-2022 5 min- interval	XGBM	MAE: 0.0053
			RMSE: 0.0072
		LSTM	MAE: 0.0049
			RMSE: 0.0069

* Datasets may include multiple cryptocurrencies, but only Bitcoin was considered. And unless specified otherwise, the interval is 1-day.

** Only best performing models and/or models selected for this thesis were included.

*** Studies may also use other metrics, but only metrics used in this thesis were included.

1.9. **Conclusion:**

Traditional methods like GARCH are still useful, but machine learning models—particularly when combined with optimization, hybrid architectures, or enriched data sources—will almost always outperform traditional models in forecasting cryptocurrencies volatility. The model performance may vary considerably based on selected cryptocurrency, forecast horizon, and features, which underlines the importance of customized model specifications.

CHAPTER 2: METHODOLOGY

2.1. Introduction:

The aim of this research is to predict the volatility of cryptocurrencies, namely Bitcoin (BTC) and Ethereum (ETH), using machine learning and deep learning models. This chapter presents the methodological framework employed in the process.

The chapter starts with data collection, the source for the historical market data and timeframes, as well as the data preprocessing steps taken to clean and prepare the raw data. Then it delves into the process employed for features engineering, including the lag variables and rolling statistics. After that, the forecasting models that have been tested in this study; this included established ensemble modeling methods and recurrent neural networks (RNN). The chapter concludes with the evaluation metrics and diagnostic checks used for assessing models performance.

The figure below is the Model pipeline or the workflow diagram that summarizes the steps taken:

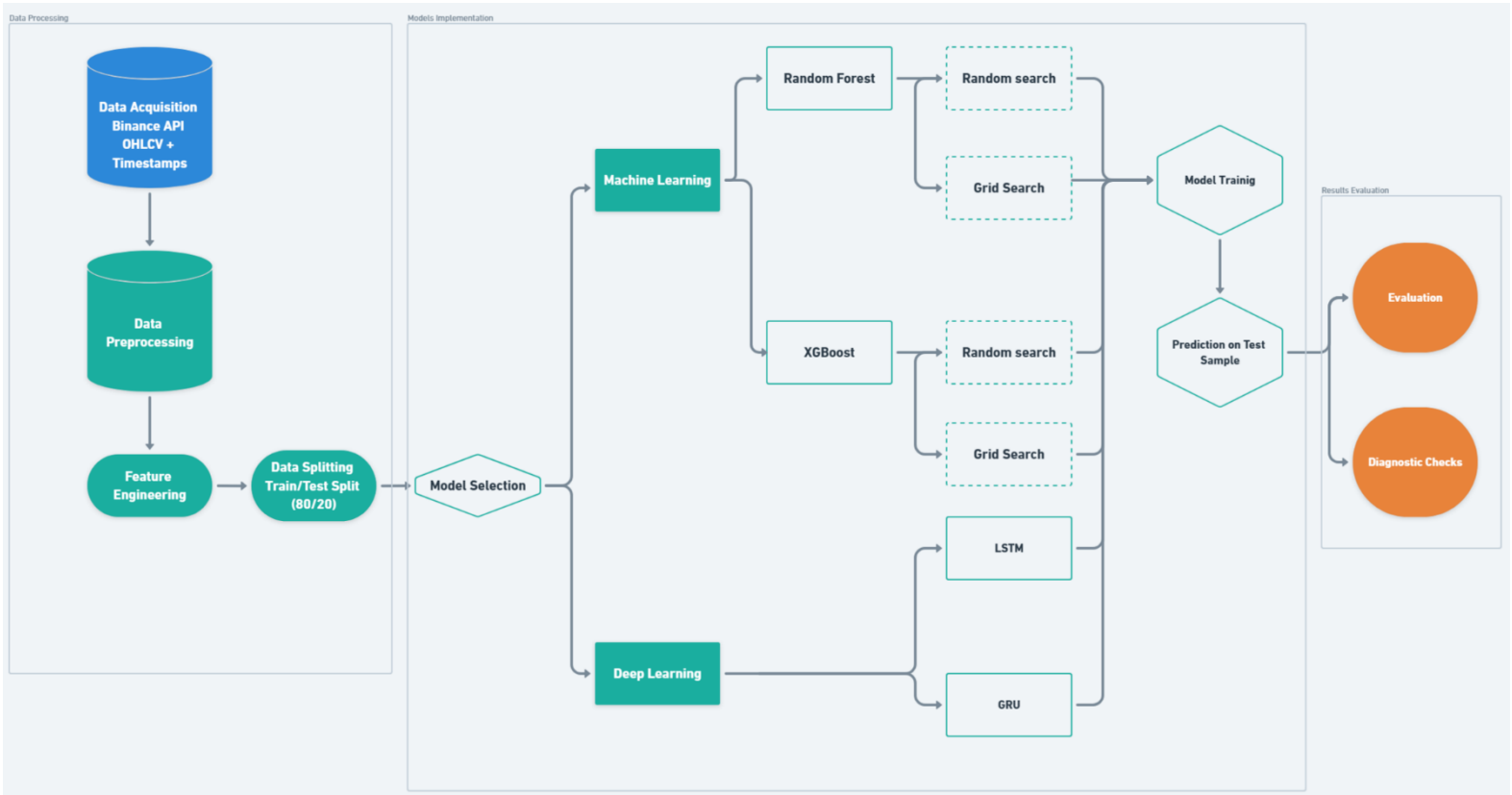

Figure 2.1 Model Pipeline (Workflow Diagram)

2.2. Data collection:

2.2.1. Time Series Data:

In financial forecasting context, time series data is a sequence of observations recorded in chronological order, and at equal time intervals, for example: minute-to-minute, hourly, daily…etc.

Cryptocurrency time series data has unique features that makes it significantly different from traditional financial time series data:

- **Non-stationarity:** The statistical characteristics of the data, like the mean, the variance…etc., change over time. This is a problem for models that assume stationarity. One important aspect of this is **heteroskedasticity**, or **volatility clustering**, which means that volatility changes over time.
- **Regime Shifts:** They are when the behavior of the underlying feature of the data (in this case: volatility) suddenly changes. they represent a real challenge for the models to accurately predict them.
- **High-Frequency Noise:** Rapid and random movements in data that don't represent a real trend, especially in high-frequency data. They can lead models to overfitting.
- **Non-Linearity:** Crypto markets behave in a complex, unpredictable ways. In simpler words: the relationship between inputs and the output is not a simple, straight line.
- **Extreme Volatility:** Self-explanatory and evident in the skewness score that will be seen later on in this chapter.
- **24/7 Trading Behavior:** No closing time or rest periods. Volatility can change anytime of the day. This requires the addition of time-based features later on in feature engineering.
- **Lack of Historical Depth:** Compared to traditional assets like bonds, stocks, and commodities, cryptocurrencies don't have a long history of trading data. This means that the data available for model training is limited, which in turn, affects the reliability of the results.

2.2.2. Selected Cryptocurrencies:

The data that was used in this study consists of two separate datasets. The first one is Bitcoin dataset, and the other one is Ethereum.

Bitcoin was first introduced by **Satoshi Nakamoto** (an alias for an unknown individual or individuals) in 2008 and entered trade in 2009 as a decentralized digital currency used in transactions on the internet as an alternative to traditional money that flow through banks. "*Bitcoin*

is a dominant figure and the most famous digital currency which is traded in more than 16,000 markets around the globe. Narayanan A, Clark J (2017)". [1] As for Ethereum, it differs from Bitcoin in that it is actually a platform for building apps that run on a blockchain. It has its own cryptocurrency called **Ether**, which is used to pay for the use of the network. [38]

The reasons those two cryptocurrencies were selected are:

1- Market dominance: The two largest cryptocurrencies by market cap [39], which makes their volatility patterns more interesting for research.
2- Long trading history (BTC since 2009 ETH since 2015), which means rich, reliable, and high frequency data.
3- The most preferred "new" financial assets among financial institutions. [40]

2.2.3. Data Provider:

The provider of the data is Binance API client. Binance is the largest cryptocurrency exchange in the world [41], which makes it more reliable and immune to manipulation. Binance API is public, free, and actively maintained. It offers both historical and live streaming data, and it is widely used in academic research.

The data span around 7 years, from 01-01-2018 at 00:00 to 27-01-2025 at 08:00. In this period, the cryptocurrencies in question reached high liquidity and institutional participation, and Binance started providing high quality, high frequency data via its API. Also, this period knew multiple market cycles, like post 2017 crush, crypto winter of 2022, and increased regulations and market recovery of 2023-2024. Not to mention major global events such as COVID-19. This all helps model generalization and robustness.

Both cryptocurrencies were paired against Tether (USDT), which is a stablecoin representing the US Dollar (USD), where 1 USDT ≈ 1 USD. So, we have two pairs: (BTCUSDT) and (ETHUSDT), both represent the **symbol** used to fetch the data of the specific coin through the Binance API.

The data frequency is hourly, which makes it well suited and sufficient for forecasting daily volatility with machine learning models. This also helps capturing the volatility more effectively than the daily data, giving the 24/7 open market nature, and avoids the noise and computational intensiveness of the high frequent, minute-by-minute data.

However, Binance API provides only up to 1000 hours for each request (less than 42 days). To bypass this, a function (get_fully_hourly) was implemented in python that relied on **pagination** in order to fetch the whole hourly data of the datasets' period (Figure 2.2 below).

The **OHLCV** structure ('open', 'high', 'low', 'close', 'volume') is the standard practice in cryptocurrency datasets intended for financial research. [42] And as such, it was adopted for these datasets. *"The central entities of the dataset are the historic prices of cryptocurrencies, including attributes like opening and closing prices, daily highs and lows, and trading volumes that give a fine-grained view of market trends and enable the identification of patterns and their correlation in time"*. [19]

```python
from binance.client import Client
import time
client = Client()

def get_full_hourly_data(symbol, start_date, end_date):
    """
    Fetch full hourly data from Binance using pagination.
    """
    interval = Client.KLINE_INTERVAL_1HOUR
    limit = 1000
    start = pd.to_datetime(start_date)
    end = pd.to_datetime(end_date)
    data = []

    while start < end:
        print(f"Fetching: {start} to {start + timedelta(hours=limit)}")
        temp = client.get_historical_klines(symbol, interval, start.strftime("%d %b, %Y %H:%M:%S"),
                                            (start + timedelta(hours=limit)).strftime("%d %b, %Y %H:%M:%S"))
        if not temp:
            break
        data += temp
        start += timedelta(hours=limit)
        time.sleep(0.5)  # to avoid rate limits

    df = pd.DataFrame(data, columns=[
        'timestamp', 'open', 'high', 'low', 'close', 'volume', 'close_time',
        'quote_asset_volume', 'num_trades', 'taker_buy_base_volume',
        'taker_buy_quote_volume', 'ignore'
    ])
    df['timestamp'] = pd.to_datetime(df['timestamp'], unit='ms')
    df.set_index('timestamp', inplace=True)
    df = df[['open', 'high', 'low', 'close', 'volume']].astype(float)
    return df
```

Figure 2.2 get_fully_hourly_data Python function

Each of the two datasets are comprised of 61898 observations, which is exactly the number of hours between the start and end dates and times (timestamps) of the datasets. All data was complete with no null entries, which validates the selection of the data provider.

We notice in both **Table 2.1** and **Table 2.2** below that the minimum volume is 0 of both BTC and ETH, which is not normal. After query, we notice that the entries in both datasets that correspond to the volume of value 0 have the same timestamps.

Table 2.1 Statistical Description of The BTC Dataset

	open	high	low	close	volume
count	61898.000000	61898.000000	61898.000000	61898.000000	61898.000000
mean	28799.930164	28930.994572	28661.857041	28801.313277	2967.584087
std	23158.331864	23254.195540	23058.839604	23159.979238	4152.140014
min	3172.620000	3184.750000	3156.260000	3172.050000	0.000000
25%	9071.517500	9108.482500	9023.917500	9071.572500	968.490852
50%	22622.465000	22756.790000	22495.340000	22622.750000	1646.683100
75%	43269.417500	43484.242500	43080.007500	43272.405000	3145.004153
max	108320.000000	109588.000000	107780.510000	108320.010000	137207.188600

Table 2.2 Statistical Description of The ETH Dataset

	open	high	low	close	volume
count	61898.000000	61898.000000	61898.000000	61898.000000	61898.000000
mean	1529.898012	1538.474826	1520.677956	1529.933712	23002.617004
std	1238.874432	1245.421312	1231.804043	1238.885177	26660.370660
min	82.160000	82.950000	81.790000	82.170000	0.000000
25%	264.912500	266.912500	262.582500	264.880000	7967.438122
50%	1525.230000	1535.330000	1514.860000	1525.340000	14868.732825
75%	2475.140000	2488.550000	2461.982500	2475.132500	27980.733598
max	4846.940000	4868.000000	4833.190000	4846.710000	493227.882820

Table 2.3 Output That Shows the Entries That Correspond with Volume = 0

```
                        open      high       low     close  volume
timestamp
2019-06-07 21:00:00   7930.85   7930.85   7930.85   7930.85     0.0
2021-02-11 03:00:00  44582.07  44582.07  44582.07  44582.07     0.0
2023-03-24 12:00:00  28080.00  28080.00  28080.00  28080.00     0.0
Number of entries with zero volume in BTC dataset: 3
```

```
                        open     high      low    close  volume
timestamp
2019-06-07 21:00:00   247.70   247.70   247.70   247.70     0.0
2021-02-11 03:00:00  1721.86  1721.86  1721.86  1721.86     0.0
2023-03-24 12:00:00  1789.52  1789.52  1789.52  1789.52     0.0
Number of entries with zero volume in ETH dataset: 3
```

The timestamps that **Table 2.3** shows correspond to the following events:

1- **2019-06-07 21:00:00 UTC:** No known public record of an event that might cause the anomaly. But it is likely a downtime or a record issue.
2- **2021-02-11 03:00:00 UTC:** Technical issues that caused delay and API problems. [43]
3- **2023-03-24 12:00:00 UTC:** An outage due to a software bug. [44]

There are three possible approaches to deal with such anomaly:

- **Remove them**. Although a common practice, this would disrupt the continuity of the datasets and possibly negatively affect the models' performance, especially LSTM and GRU.
- **Impute them,** by interpolation or forward-fill for missing values. But this can distort or misrepresent market behavior.
- **Flag them**. Might be the best conduct, since they represent real-world events that affected the market flow. This would also be a chance for models to learn special market conditions that might affect volatility.

So, the approach selected for these datasets anomalies is flagging. This is done by creating a new column (zero_volume_flag), where the rows with (volume == 0) will take the value 1, and the others will take the value 0.

Table 2.4 Summary of The BTC and ETH Datasets

BTC dataset summary:

Column	Non-Null Count	Dtype
open	61898	float64
high	61898	float64
low	61898	float64
close	61898	float64
volume	61898	float64
zero_volume_flag	61898	int32

Start Timestamp: 2018-01-01 00:00:00
End Timestamp: 2025-01-27 07:00:00
Total Rows: 61898
Total Columns: 6
Memory Usage (MB): 3.07

ETH dataset summary:

Column	Non-Null Count	Dtype
open	61898	float64
high	61898	float64
low	61898	float64
close	61898	float64
volume	61898	float64
zero_volume_flag	61898	int32

Start Timestamp: 2018-01-01 00:00:00
End Timestamp: 2025-01-27 07:00:00
Total Rows: 61898
Total Columns: 6
Memory Usage (MB): 3.07

From **Table 2.4** we can see that both of the datasets are in a well-structured and consistent state, with each comprised of 61898 rows that represent the hourly entries in the period from January 1,

2018, to January 27, 2025. The “Non-Null Count” of the columns is equal to the total number of the entries (Total Rows), which ensures completeness and reliability. The data types are appropriate and the memory usage of 3.07 MB each assures efficient machine learning workflow.

Figure 2.3 BTC Hourly Close Prices Over Time

Examining the figure above, we notice that starting from 2021, the close price exhibit sharp surges followed by somewhat steep decline in short time spans. Bitcoin’s price doubles and halves in a matter of months, which indicates a highly volatile market.

Figure 2.4 ETH Hourly Close Prices Over Time

The same is noticeable for Ethereum, with even more sharp swings in short time periods. But the difference between them is in the magnitude; while ETH capped at almost 5k $, BTC reached an all-time high surpassing 100k $.

2.3. Volatility Calculation:

In this research, volatility is represented by **the standard deviation of log returns**. It is calculated through the following formula: [45]

$$\sigma_t = \sqrt{\frac{1}{N-1}\sum_{i=0}^{N-1}(r_{t-i} - \bar{r}_t)^2} \quad (2.1)$$

Where:

- σ_t is the estimated volatility at time t,
- N is the window size (24 hourly log returns = 1 day),
- r_{t-i} is the log return at hour t-i,
- $\bar{r}_t = \frac{1}{N}\sum_{i=0}^{N-1} r_{t-i}$ the mean of the N log returns.

But before this, we need to calculate the simple returns and log returns:

a. Simple Return:

$$R_t = \frac{P_t - P_{t-1}}{P_{t-1}} = \frac{P_t}{P_{t-1}} - 1 \quad (2.2)$$

Where:

- R_t is the **simple return** at time t,
- P_t is the closing price at time t,
- P_{t-1} is the closing price at time t-1.

b. Log Return:

$$r_t = \ln\left(\frac{P_t}{P_{t-1}}\right) \quad (2.3)$$

Where r_t, P_t, P_{t-1} are as defined above.

```
#return and log return

df['return'] = df['close'].pct_change()
df['log_return'] = np.log(df['close'] / df['close'].shift(1))
```

Figure 2.5 return and log_return features addition

Two additional columns are introduced to the datasets (Figure 2.5):

The first is (return), which represents the calculated simple return for each observation. It is computed through '**percentage change**' method of the **Pandas** python library (pct_change()) using the (close) column.

The second one is (log_return), which represents the calculated log return for each observation. It is computed through the (np.log()) function of the **NumPy** library, also using the (close) column.

```
#Volatility (Rolling Standard Deviation of Returns)

df['volatility_12h'] = df['log_return'].rolling(window=12).std()
df['volatility_24h'] = df['log_return'].rolling(window=24).std()
df['volatility_7d'] = df['log_return'].rolling(window=24*7).std()
df['volatility_30d'] = df['log_return'].rolling(window=24*30).std()
```

Figure 2.6 Volatility features addition

Now, as Figure 2.6 shows, we add the volatility column(s). The choice of this thesis is to forecast the 24 hours volatility. This is the standard practice in risk management and Value-at-Risk computation, and daily volatility forecast is more useful than shorter (hourly) or longer (weekly) horizons for financial institutions' decision making. [46] Also, daily volatility forecasting is in compliance with reporting standards that require daily metrics. [47]

The volatility is computed using '**standard deviation**' function of the **Pandas** python library (std()), and (rolling()) tool that helps calculate in a particular predefined window.

The datasets will also include the 12 hours, 7 days, and 30 days volatility calculations. The inclusion of these values is to help the models capture short-term and long-term volatility trends and patterns. So overall, we will have 4 new columns.

2.4. Feature Engineering:

To ensure that the machine learning models perform at their optimum, the data provided must contain enough features that represent the underlaying patterns and relationships to be learned by the algorithms. These features can be derived from the raw data, and are very crucial in the process of implementing the model pipeline.

The datasets have already been enriched through the addition of few engineered features: return, log return, and different volatility measures.

Other features added are as follow:

- **Time-based Features:** This includes the hour (0 to 23), the day of the week (0 = Monday to 6 = Sunday), day of the month (1 to 28 or 29 or 30 or 31), month (1 to 12), week of the year (1 to 52), and weekend (0 if no, 1 if yes (Saturday/Sunday)). Also, cyclic feature of

the hour of the day, which helps identify 23:00 hours as close to 00:00 rather than far away in a linear representation. These features are especially important for LSTM and GRU to learn temporal dependencies.

- **Rolling Statistics:** This includes the mean, the standard deviation, the lowest, the highest, the skew, and the kurtosis of the close prices in a window of 24 hours to align with the target (24-hour volatility). These features capture important market dynamics like volatility clustering and regime shifts.
- **Lag Features** of both return and 24-hour volatility. The lags are in this range [1, 6, 12, 24] hours. "*Lagging means going some steps back in time. To predict the future, the past is our best resource — it is not surprising that lagged values of the target variable are quite often used as inputs for forecasting*." [48] They are important for tree models to remember past values since they lack the memory component of RNN models.
- **Technical Indicators:** Moving averages (MA), Exponential moving averages (EMA), the momentum, Relative Strength Index (RSI), Moving Average Convergence Divergence (MACD), and Bollinger Bands. All in the window of 24 hours. They indicate trends, overbought/oversold signals, and measure volatility itself.

The Null entries need to be dropped because the new features created a lot of (NaNs) as the **Table 2.5** shows:

Table 2.5 Number of Null Rows in Each Column Containing Them in The Datasets After Adding the New Features

	Column	Non-null Count	NaN Count
0	return	61897	1
1	log_return	61897	1
2	volatility_12h	61886	12
3	volatility_24h	61874	24
4	volatility_7d	61730	168
5	volatility_30d	61178	720
6	close_roll_std_24	61897	1
7	close_roll_skew_24	61896	2
8	close_roll_kurt_24	61895	3
9	return_lag_1	61896	2
10	volatility_lag_1	61873	25
11	return_lag_6	61891	7
12	volatility_lag_6	61868	30
13	return_lag_12	61885	13
14	volatility_lag_12	61862	36
15	return_lag_24	61873	25
16	volatility_lag_24	61850	48
17	ma_24h	61875	23
18	momentum_24h	61874	24
19	rsi	61875	23
20	macd	61873	25
21	macd_signal	61865	33
22	bb_upper	61875	23
23	bb_lower	61875	23
24	bb_width	61875	23

This will leave the datasets with 61178 rows each.

2.4.1. Target Variable:

```
#target variable:

horizon = 24  # Predicting next 24-hour volatility
df['future_volatility'] = df['volatility_24h'].shift(-horizon)
df.dropna(inplace=True)
```

Figure 2.7 future_volatility Feature Creation

Since the goal is forecasting the next 24-hour volatility, a feature (future_volatility) that reflects the real calculated volatility after 24 hours is created by shifting the values of the (volatility_24h) column up by 24 rows. This will leave us with 24 rows with null entries that must be dropped. The final number of rows in both of the datasets is **61154**, where the start date is shifted to **31-01-2018 01:00**, and the end date is **26-01-2025 08:00**.

The plot of the distribution of the (future_volatility) column of the BTC dataset is shown in **figure 2.8** below:

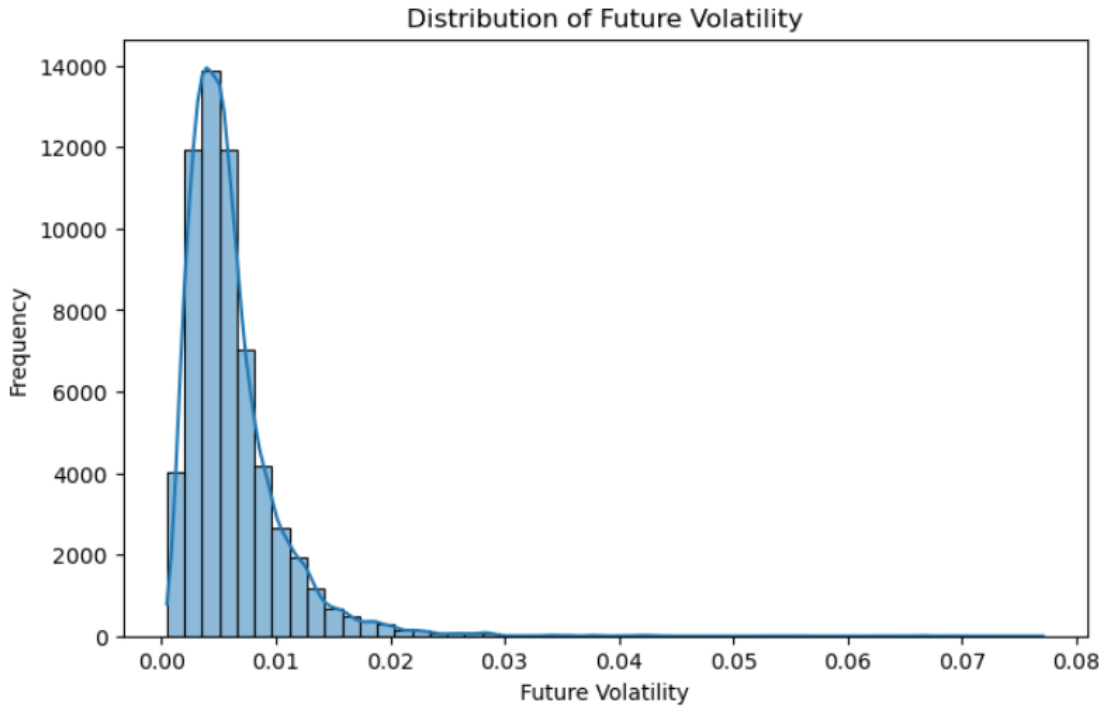

Figure 2.8 The Distribution of Future Volatility of BTC

The distribution is highly **right-skewed**: a long tail stretches to the right toward higher values while the majority of data points are on the left (lower values). This means that most volatility is

on the lower side, but few large ones pull the tail to the right. This is important because this can make it harder for models to learn patterns and relationships.

Using the (skew()) function in **Pandas**, we calculate the skewness. The result is 3.384 which is way greater than 0 and is compatible with the right-skewed plot. Then, it is recommended to apply log transformation on (future_volatility). As Figure 2.9 below indicates, using the (np.log1p()) function, the results are put in a new feature: (target).

```
#the histogram is heavily right-skewed -> a log-transform is often beneficial.

df['target'] = np.log1p(df['future_volatility'])
```

Figure 2.9 Log transforming Target Feature

After computing the skewness of the new target feature, we find a slight improvement at 3.271, albeit still right-skewed.

The same goes for the ETH dataset with a slight difference in skewness results (2.9 before log transformation and 2.813 after), but with the same implications.

2.4.2 Train Test Split:

```
df = df.sort_index()
# Define features and target
X = df.drop(['future_volatility', 'volatility_24h', 'target'], axis=1)
y = df['target']

# Split into training and test sets (no shuffling for time series)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, shuffle=False)
```

Figure 2.10 Train Test Split Code

The first step in getting the data properly set for model training and evaluation is chronologically sorting the datasets by the datetime index to preserve and ensure the temporal order soundness, which is critical for time series forecasting.

After that, the feature matrix (X) and the target variable (y) are both created from the datasets, were (X) is assigned all the independent features except the target and the features corresponding to the target (volatility_24h and future_volatility in this case) to prevent data leakage during training, and (y) is assigned only the target feature. The resulting (X) matrix is comprised of 42 columns and 61154 rows, where (y) is just one column with the same number of rows.

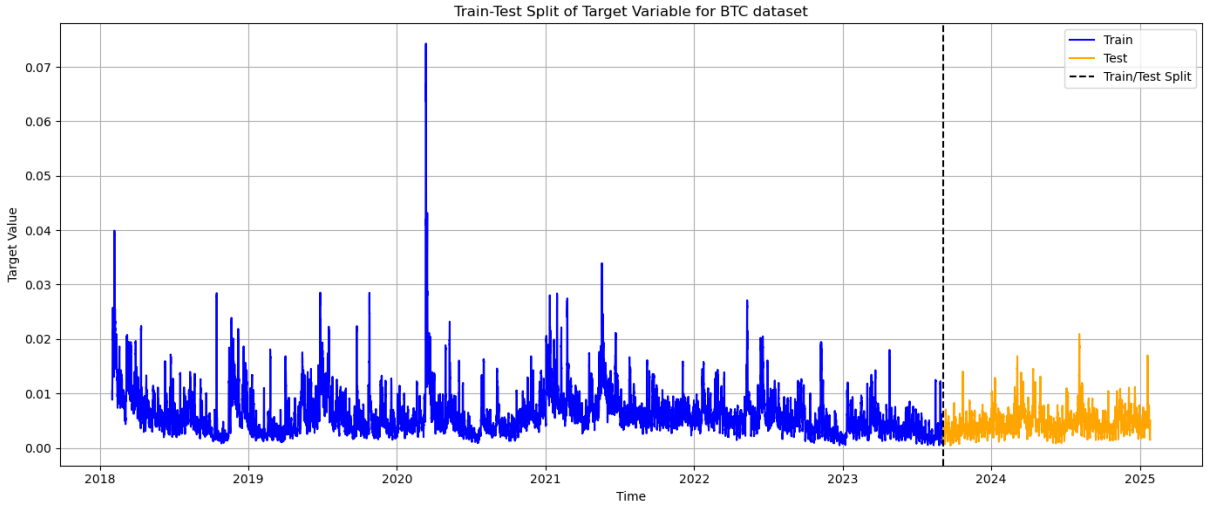

Figure 2.11 Temporal Train-Test Split of Log-Transformed Volatility (BTC, 2018–2025)

Then, they were split into training and testing data at an 80/20 ratio, by keeping the last 20% of observations for test data. Furthermore, shuffling was disabled to preserve the temporal order of the split and retain the most recent data for testing and evaluating which is only valid on future unseen data in time series forecasting.

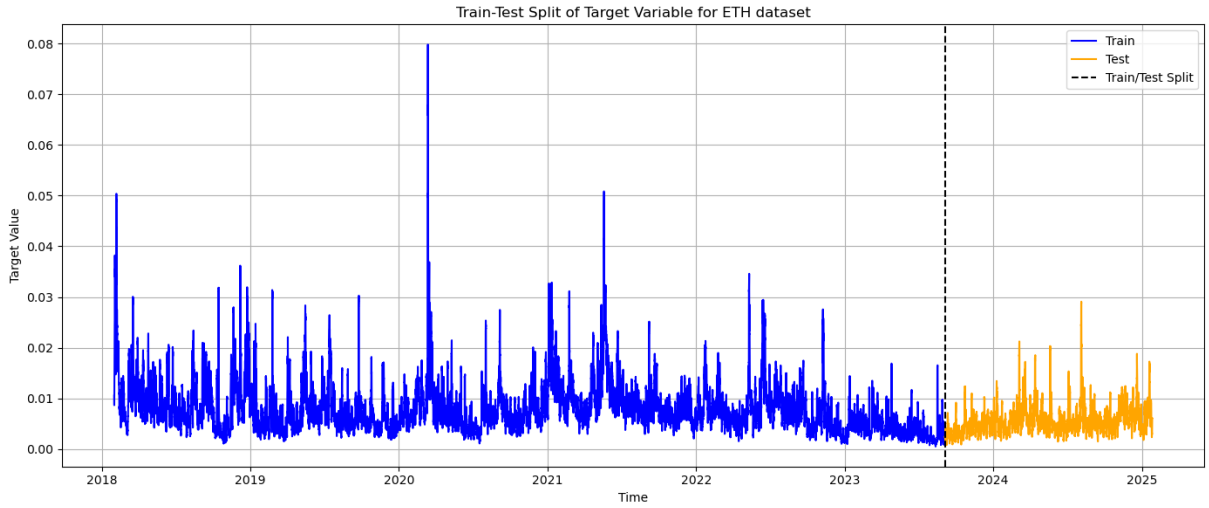

Figure 2.12 Temporal Train-Test Split of Log-Transformed Volatility (ETH, 2018–2025)

The break point between the train and test data is at **04-09-2023 17:00**, which makes the train data consisting of **48923** rows and the test data of **12231** rows.

2.5. Machine Learning Models:

2.5.1. Traditional Machine Learning Models:

2.5.1.1. Random Forest:

The Random Forest Regressor employed in this study was tuned with particular attention to the sequential nature of the time series data:

```
from sklearn.model_selection import TimeSeriesSplit

tscv = TimeSeriesSplit(n_splits=3) #avoid data leakage in time series.
```

Figure 2.13 Cross Validation for Time Series

1. **Cross-Validation:** As Figure 2.13 above shows, **TimeSeriesSplit** from **sklearn** was utilized to ensure that the train and test sets respect chronological order, with 3 splits to preserve temporal integrity.
2. Both **Random search** and **Grid search** techniques were used for comparative purposes. The key difference between the two is that while the grid search technique is deterministic that exhaustively searches all the possible combinations of the specified hyperparameters to select only the best scoring one, random search is probabilistic that randomly samples a fixed number of combinations to evaluate.

```
rf = RandomForestRegressor(random_state=42)
param_dist_rf = {
    'n_estimators': [100, 200, 300],
    'max_depth': [3, 5, 10, None],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'bootstrap': [True, False]
}

search_rf = RandomizedSearchCV(rf, param_distributions=param_dist_rf,
                               n_iter=20, cv=tscv, scoring='neg_mean_squared_error', random_state=42, n_jobs=-1, verbose=1)
```

Figure 2.14 RF Random Search Hyperparameters

```
# Use a smaller, more focused parameter grid for fine-tuning
param_grid_rf = {
    'n_estimators': [100, 200],
    'max_depth': [5, 10],
    'min_samples_split': [2, 5],
    'min_samples_leaf': [1, 2],
    'bootstrap': [True]
}

# GridSearchCV instead of RandomizedSearchCV
grid_search_rf = GridSearchCV(
    rf,
    param_grid=param_grid_rf,
    cv=tscv,
    scoring='neg_mean_squared_error',
    n_jobs=-1,
    verbose=1
)
```

Figure 2.15 RF Grid Search Hyperparameters

3. **Hyperparameters:** Summarized in **Table 2.6** below as shown in **Figure 2.14** and **Figure 2.15**:

"*Hyperparameter tuning in machine learning is the process of making a model perform optimally by selecting the best parameters. Unlike model parameters, which are equipped during training, hyperparameters must be set before the model training begins and control aspects such as model complexity and learning efficiency*". [19]

Table 2.6 Hyperparameters of Random Forest Model

Component	Hyperparameter Value	Explanation
Ensemble Size (Number of trees)	Random search: [100, 200, 300]	Increasing the number of trees is generally beneficial for the performance, but it is computationally costly.
	Grid search: [100, 200]	
Max Tree Depth	Random search: [3, 5, 10, None]	Maximum depth is basically the number of levels of each

	Grid search: [5, 10]	tree. It is intended to control model complexity and prevent overfitting.
Node Splitting (Min samples to split)	Random search: [2, 5, 10]	Minimum sample number required to split an internal node (during tree construction). This is important to prevent both overfitting and underfitting.
	Grid search: [2, 5]	
Leaf Size (Min samples at leaf)	Random search: [1, 2, 4]	Larger values reduce tree complexity.
	Grid search: [1, 2]	
Bootstrap Sampling	Random search: [True, False]	Controls the variance and diversity of the trees.
	Grid search: [True]	
Model Evaluation	Scoring = Negative MSE	To evaluate each model during cross-validation.

```
# Fit the grid search
grid_search_rf.fit(X_train, y_train)

# Best model
best_rf = grid_search_rf.best_estimator_
```

Figure 2.16 Best Performing Grid Search RF Model Selection

Then, the best model we retrieve after training (for Random and Grid search in both RF and XGBoost) is the one used to make the prediction using the test sample, as shown in Figure 2.16 in the case of RF grid search.

2.5.1.2. XGBoost (Gradient Boosting):

The model used in this study was tuned with the same values for both grid and random search:

1. **Cross-Validation:** Same as RF model.

2. Also, Both **Random search** and **Grid search** techniques were used.

```
#Gradient boosting

#using RandomizedSearchCV

from xgboost import XGBRegressor

xgb = XGBRegressor(objective='reg:squarederror', random_state=42)

param_dist_xgb = {
    'n_estimators': [100, 200, 300],
    'max_depth': [3, 5, 7],
    'learning_rate': [0.01, 0.05, 0.1],
    'subsample': [0.6, 0.8, 1.0],
    'colsample_bytree': [0.6, 0.8, 1.0]
}

search_xgb = RandomizedSearchCV(xgb, param_distributions=param_dist_xgb,
                                n_iter=20, cv=tscv, scoring='neg_mean_squared_error', random_state=42, n_jobs=-1)
```

Figure 2.17 XGBoost Random Search Hyperparameters

```
# Define grid of parameters (same as before)
param_grid_xgb = {
    'n_estimators': [100, 200, 300],
    'max_depth': [3, 5, 7],
    'learning_rate': [0.01, 0.05, 0.1],
    'subsample': [0.6, 0.8, 1.0],
    'colsample_bytree': [0.6, 0.8, 1.0]
}

# Set up GridSearchCV
grid_search_xgb = GridSearchCV(
    estimator=xgb,
    param_grid=param_grid_xgb,
    scoring='neg_mean_squared_error',
    cv=tscv,
    n_jobs=-1,
    verbose=1
)
```

Figure 2.18 XGBoost Grid Search Hyperparameters

4. **Hyperparameters:** Summarized in the table below as shown in Figure 2.17 and Figure 2.18:
    - **The Objective Function:** The model was initialized with the objective function set to 'reg:squarederror' to minimize the mean squared.
    - The rest of hyperparameters are summarized in table 2.7:

Table 2.7 Hyperparameters of XGBoost Model

Component	Hyperparameter Value	Explanation
Boosting Rounds	[100, 200, 300] (Random & Grid)	Each round trains a tree that tries to correct the errors of the previous trees. Too many rounds might lead to overfitting. Too few may cause underfitting.
Max Tree Depth	[3, 5, 7]	Maximum depth is basically the number of levels of each tree. It is intended to control model complexity and prevent overfitting.
Shrinkage Factor	learning_rate = [0.01, 0.05, 0.1]	Controlled with learning_rate. It prevents the model from quickly overfitting and improves generalization.
Subsample Ratio	subsample = [0.6, 0.8, 1.0]	Controls how much of the train data is used per boosting round.
Feature Sampling	colsample_bytree = [0.6, 0.8, 1.0]	Features are randomly selected to train each tree. More features might cause overfitting; fewer may reduce the risk of overfitting.
Model Evaluation	Scoring = Negative MSE	To evaluate each model during cross-validation.

After fitting the models to the training data, the best model is selected to make the forecast on the test data, ensuring the best performance.

2.5.2. Deep Learning Models:

2.5.2.1. LSTM (Long Short-Term Memory):

```
# Scale X
scaler_X = MinMaxScaler()
X_scaled = scaler_X.fit_transform(X)

# Scale y
scaler_y = MinMaxScaler()
y_scaled = scaler_y.fit_transform(y.to_numpy().reshape(-1, 1)).flatten()

# Sequence generation
def create_sequences(X, y, time_steps=30):
    Xs, ys = [], []
    for i in range(len(X) - time_steps):
        Xs.append(X[i:(i + time_steps)])
        ys.append(y[i + time_steps])
    return np.array(Xs), np.array(ys)

X_seq, y_seq = create_sequences(X_scaled, y_scaled, time_steps=30)

# Train-test split
split = int(0.8 * len(X_seq))
X_train_seq, X_test_seq = X_seq[:split], X_seq[split:]
y_train_seq, y_test_seq = y_seq[:split], y_seq[split:]

# Manual time-aware validation split (last 10% of training data)
val_size = int(0.1 * len(X_train_seq))

X_val_seq = X_train_seq[-val_size:]
y_val_seq = y_train_seq[-val_size:]

X_train_final = X_train_seq[:-val_size]
y_train_final = y_train_seq[:-val_size]
```

Figure 2.19 LSTM (and GRU) Data Preparation

As **Figure 2.19** indicates, before running the model, extra data preparation is required:

- **Feature scaling**: Both (X) and (y) are scaled to the range of [0,1] using **MinMaxScaler** of **sklearn**. This is required for better neural networks performance.
- **Sequence generation:** Using a function that creates sequences from the data. This is important because LSTM requires 3D input, and this function structures the data as such.
- **Train test split:** sequences from the step before are split 80/20 for training and testing.

- **Manual validation split:** By default, the selection of validation sample is random and based on shuffling, and this is not ideal for time series data. In order to preserve temporal soundness, a manual split that respects the temporal order is required.

```
from keras.layers import Dropout

model_lstm = Sequential([
    LSTM(64, return_sequences=True, input_shape=(X_train_seq.shape[1], X_train_seq.shape[2])),
    Dropout(0.2),
    LSTM(32),
    Dropout(0.2),
    Dense(1)
])

from keras.callbacks import EarlyStopping

early_stop = EarlyStopping(monitor='val_loss', patience=10, restore_best_weights=True)

model_lstm.compile(loss='mse', optimizer='adam')
model_lstm.fit(X_train_final, y_train_final, epochs=100, batch_size=32, validation_data=(X_val_seq, y_val_seq),
               callbacks=[early_stop])
```

Figure 2.20 LSTM Model Hyperparameters

Table 2.8 below summarizes the hyperparameters of the LSTM model used in this study as shown in **Figure 2.20**:

Table 2.8 LSTM model hyperparameters:

Component	Hyperparameter	Explanation
LSTM layer 1	Units = 64	Memory cells needed to learn patterns
	Return sequence = true	Return the sequence to layer 2 to further process
Dropout 1	Rate = 0.2	Randomly drop 20% of neurons to prevent overfitting
LSTM layer 2	Units = 32	No more layers after this, so no need to return sequence.
	Return sequence = false	
Dropout 2	Rate = 0.2	Randomly drop 20% of neurons to prevent overfitting
Dense layer	Units = 1	For output
Compilation	Loss function = mse	Default LSTM settings, common for regression forecasting. Model
	Optimizer = adam	

		validation is based on MSE score.
Training	Epochs = 100	Train iterations, high to learn complex patterns
	Batch size = 32	Smaller helps generalization
	Validation data = 0.1	10% of training data to validate (manually split in respect of temporal order)
Early stopping	Monitor = validation loss	Model stops when no improvement on validation. Protects from overfitting and unnecessary training time.
	Patience = 10 epochs	Avoids stopping too early just because of an MSE score drop.
	Restore best weight = true	Rolls back to the best model.

2.5.2.2. GRU (Gated Recurrent Unit):

The same extra data preparation for LSTM shown in **Figure 2.19** is also required for GRU (Feature scaling, sequence generation, train test split, manual validation split).

```
#GRU

model_gru = Sequential([
    GRU(64, return_sequences=True, input_shape=(X_train_seq.shape[1], X_train_seq.shape[2])),
    Dropout(0.2),
    GRU(32),
    Dropout(0.2),
    Dense(1)
])


model_gru.compile(loss='mse', optimizer='adam')

early_stop = EarlyStopping(monitor='val_loss', patience=10, restore_best_weights=True)

model_gru.fit(X_train_final, y_train_final, epochs=100, batch_size=32, validation_data=(X_val_seq, y_val_seq),
              callbacks=[early_stop])
```

Figure 2.21 GRU Model Hyperparameters

Table 2.9 summarizes the hyperparameters of the GRU model used in this study as shown in **Figure 2.21**:

Table 2.9 GRU model hyperparameters:

Component	Hyperparameter	Explanation
GRU layer 1	Units = 64	Memory cells needed to learn patterns
	Return sequence = true	Return the sequence to layer 2 to further process
Dropout 1	Rate = 0.2	Randomly drop 20% of neurons to prevent overfitting
GRU layer 2	Units = 32	Final layer before dense output.
Dropout 2	Rate = 0.2	Randomly drop 20% of neurons to prevent overfitting
Dense layer	Units = 1	For output
Compilation	Loss function = mse	Default LSTM settings, common for regression forecasting. Model validation is based on MSE score.
	Optimizer = adam	
Training	Epochs = 100	Train iterations, high to learn complex patterns
	Batch size = 32	Smaller helps generalization
	Validation data = 0.1	10% of training data to validate (manually split in respect of temporal order)
Early stopping	Monitor = validation loss	Model stops when no improvement on validation. Protects from overfitting and unnecessary training time.
	Patience = 10 epochs	Avoids stopping too early just because of an MSE score drop.
	Restore best weight = true	Rolls back to the best model.

2.6. Evaluation Metrics:

The following evaluation metrics were applied in this study to evaluate the models in both error measuring and performance. The selection of more than one metric was intentional in order to calibrate between advantages and disadvantages of each metric:

2.6.1. Mean Absolute Error (MAE):

Mean Absolute Error (MAE) is the average of the absolute differences between predicted and actual values: [49]

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|\hat{y}_i - y_i| \tag{2.4}$$

Where N is the total number of data points, y is the actual value and $\hat{y}$ is the predicted value.
It is less sensitive to outliers and easy to interpret. However, it does not penalize wrong predictions as much as other metrics. [50]

2.6.2. Mean squared Error (MSE):

It is the average of the squared differences between predicted and actual values: [49]

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2 \tag{2.5}$$

It penalizes large errors due to the squaring. But this can be disadvantageous if outliers are present, and it makes it less interpretable than MAE. [50]

2.6.3. Root Mean squared Error (RMSE):

It is the squared root of the mean squared error (MSE):

$$RMSE = \sqrt{\boldsymbol{MSE}} \tag{2.6}$$

It penalizes large errors due to the squaring, and it is easily interpretable. However, it is sensitive to outliers. [50]

2.6.4. R Squared (R^2):

It is the coefficient of determination, or the proportion of variance in the target variable explained by the model: [51]

$$R^2 = \frac{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{N}(y_i - \bar{y})^2} \tag{2.7}$$

Where $\bar{y}$ is the mean of actual values.

R^2 is more of a performance metric than an error metric. It indicates goodness of fit, where the closer the value is to 1 the better. However, it is recommended to use it with caution in the context of time series forecasting. [52]

2.7. Diagnostic Checks:

In order to validate the results of the evaluation metrics and make sure of their reliability, diagnostic checks are important to perform. These are a collection of plots that help better understand the model's performance: where it succeeds and where it fails, not just the overall score that might not provide a detailed insight into the model's behavior across different contexts.

2.7.1. Actual vs. Predicted Volatility Plot:

It is a line plot of actual against predicted volatility overlay across time, or time-sorted rows index. It reveals how well a model tracks volatility, especially spikes and regime shifts.

2.7.2. Histogram of Residuals:

It displays the distribution of residuals:

$$\text{Residual} = \text{Actual Value} - \text{Predicted Value} \tag{2.8}$$

Good characteristics of this histogram are typically: center around zero, symmetric bell shape, narrow spread, and absent or light tail.

2.7.3. Residuals vs Predicted Values Scatter Plot:

Residual points should be scattered near (x = 0) axis with no recognizable pattern or shape. Points above this axis indicate underestimation, and below they indicate overestimation.

2.7.4. Error vs Actual Volatility Scatter Plot:

Here:

$$\text{Error} = |\text{Residual}| \tag{2.9}$$

Points should cluster in the lower part of the plane, which indicates low error through all regimes (Low, Medium, and High Volatility). U shape indicates a model that fails to model low and high volatility, and only works in medium (mean).

2.7.5. Predictions vs Actual Volatility Scatter plot:

The closer the points are to the 45° line (y = x) the better the model is. Above this line indicates overestimation, and below it underestimation.

2.7.6. Residual Heatmap Overtime:

Neutral colors indicate stability and consistency. Red indicates high residuals (positive), which means underestimations. Blue means overestimation because it indicates low residuals (negative).

2.8. Software and Tools:

2.8.1. Programming Language:

The table below contain information about the software used in this study:

Table 2.10 Software Used in This Thesis

PC	Language	Version	IDE	IDE Version
PC 1 (Laptop)	Python	3.9.13	Jupyter Notebook	6.4.12
PC 2 (Desktop)	Python	3.12.7	Jupyter Notebook	7.2.2

2.8.2. Libraries:

Table 2.11 Libraries of Python and Their Content/Functionality Used in This Thesis

Library	Content/Functionality
pandas	DataFrames, series, time series handling, data cleaning
numpy	N-dimensional arrays, numerical operations
Matplotlib	Static 2D plotting: line charts, bar plots, histograms
seaborn	Statistical data visualization
ta	technical analysis
sklearn	GradientBoostingRegressor, RandomForestRegressor, mean_squared_error, mean_absolute_error, MinMaxScaler, train_test_split, r2_score,

	RandomizedSearchCV, TimeSeriesSplit, GridSearchCV
tensorflow	Sequential, LSTM, GRU, Dense, EarlyStopping, Dropout
xgboost	Gradient boosting model
datetime	date, datetime, timedelta
binance	Client
tabulate	tabulate

2.8.3. Computing Environment:

Two machines were used in the making of this thesis. The specs of each are shown in the table below:

Table 2.12 Summary of Computing Environment

Machine	Type	CPU	GPU	RAM	STORAGE	OS
PC1	Laptop	i7-6500U 4CPUs 2.6 GHz	Intel(R) HD 520	32 GB 2133 MHz DDR4	512 GB SSD + 1 TB HDD	Windows 10 Pro 64 bit
PC2	Desktop	i5-12400f 12CPUs 2.5 GHz	RTX 3070ti 8 GB	16 GB 3200 MHz DDR4	500 GB SSD	Windows 11 Pro 64 bit

PC1 was used mainly in thesis drafting, notebook files viewing, web research, review of related literature, and preliminary code testing. PC2, having the most powerful hardware among the two machines—especially the powerful GPU that was especially important for DL models that require high computational resources, served as the main computing environment where the core code of the project was executed and from which the results were obtained.

2.9. Conclusion:

This thesis is about forecasting the 24-hour future volatility of Bitcoin and Ethereum. The data was collected from Binance API covering the period 2018-2025 with hourly frequence. Data cleaning involved handling anomalies like zero-volume entries and feature engineering such as returns, log returns, volatility in different windows, time variables, technical indicators, and lagged features. The target variable is log transformed 24-hour volatility due to heavy right-skewed distribution. The datasets were split into 80% training and 20% testing respecting the chronological order that is critical in time series forecasting. Four models were tested: Random Forest and XGBoost (Ensemble learning) with hyperparameters selection to use both random and grid search. The other two are LSTM and GRU (RNN) with manual validation split and early sopping. Evaluation metrics used included: MAE, MSE, RMSE, and R^2 in addition to several plots that represent diagnostic checks. the chapter ends with software, tools, and machine details.

CHAPTER 3: RESULTS AND DISCUSSION

3.1. Introduction:

In this chapter, the results from the forecasting models built to forecast 24-hour cryptocurrency volatility are presented. Forecasting results from traditional machine learning models (Random Forest, XGBoost), and deep learning models (LSTM, GRU) will be shown using error metrics, plots, and charts. Alongside presenting the results, a comparative analysis is conducted with the aim of highlighting the advantages and disadvantages of each model, followed by diagnostic checks to evaluate the validity and reliability of the results. Then, the discussion delves into interpreting the findings in context with the existing literature and the nature of financial time series data. Finally, it ends with the acknowledging the limitations of this work.

3.2. Evaluation Metrics Summary:

3.2.1. Prediction on Train Sample Analysis:

Table 3.1 Summary of evaluation metrics scores on BTC and ETH (train sample)

Model	BTC					ETH				
	Train time	**MAE**	**MSE***	**RMSE**	R^2	**Train time**	**MAE**	**MSE***	**RMSE**	R^2
RF (random)	840 s (14 min)	0.0021	1.1148	0.0033	0.4321	865 s (14.4 min)	0.0027	1.628	0.0040	0.4148
RF (grid)	458 s (7.6 min)	0.0019	**0.9261**	**0.0030**	**0.5282**	464 s (7.7 min)	0.0024	1.286	0.0035	**0.5377**
XGB (random)	**16 s**	0.0022	1.2273	0.0035	0.3748	**17 s**	0.0027	1.681	0.0041	0.3957
XGB (grid)	149 s (2.5 min)	0.0022	1.2266	0.0035	0.3752	153 s (2.5 min)	0.0025	1.415	0.0037	0.4914
LSTM	254 s (4.2 min)	**0.0018**	0.9649	0.0031	0.4942	264 s (4.4 min)	**0.0022**	**1.248**	**0.0035**	0.5368
GRU	208 s (3.5 min)	0.0019	1.0457	0.0032	0.4518	167 s (2.8 min)	0.0023	1.251	0.0035	0.5357

(*) Indicates MSE value to 10^{-5}.

As the **table 3.1** above shows, Random Forest (Grid Search) gave the best performance overall in R^2, and the best in MSE and RMSE on the BTC dataset. While the best performer overall in MAE was LSTM, which also exceeded the other models in MSE and RMSE on the ETH dataset. The worst on every metric and dataset was XGBoost.

3.2.2. Prediction on Test Sample Analysis:

Table 3.2 Summary of evaluation metrics scores on BTC and ETH (test sample)

Model	BTC				ETH			
	MAE	**MSE***	**RMSE**	R^2	**MAE**	**MSE***	**RMSE**	R^2
RF (random)	0.00169	0.5019	0.00224	0.1202	0.00204	0.7966	0.00282	0.0698
RF (grid)	0.00160	0.4572	0.00213	0.1986	0.00187	0.6992	0.00264	0.1836
XGB (random)	0.00180	0.5205	0.00228	0.0876	0.00226	0.8065	0.00284	0.0582
XGB (grid)	0.00180	0.5182	0.00227	0.0917	0.00194	0.6810	0.00260	0.2048
LSTM	**0.00147**	**0.4157**	**0.00203**	**0.2614**	0.00167	**0.5950**	**0.00243**	**0.2928**
GRU	0.00149	0.4171	0.00204	0.2590	**0.00164**	0.6250	0.00250	0.2572

(*) Indicates MSE value to 10^{-5}.

Figure 3.1 and **Figure 3.2** present a comparative analysis of model performance across four machine learning approaches—Random Forest, XGBoost, LSTM, and GRU on both datasets.

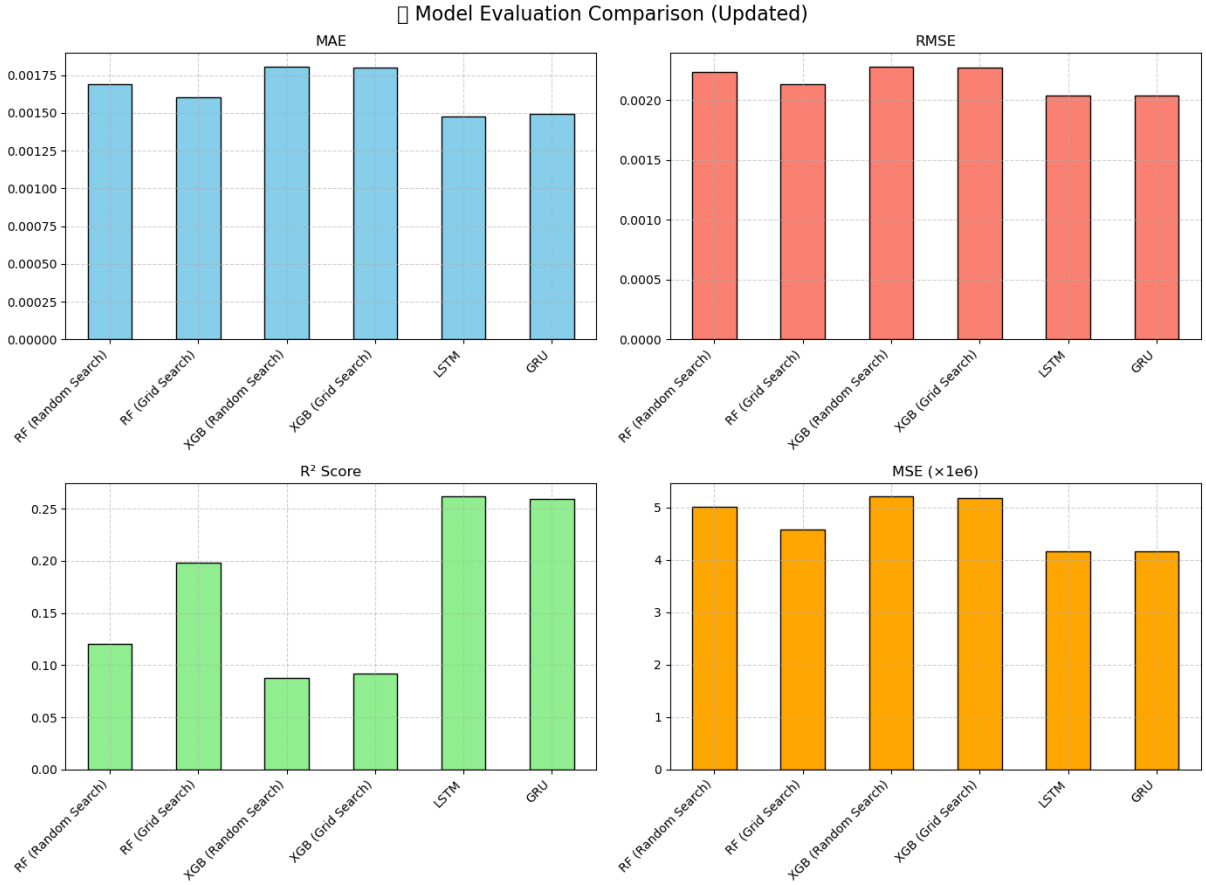

Figure 3.1 Model Evaluation Comparison for BTC dataset

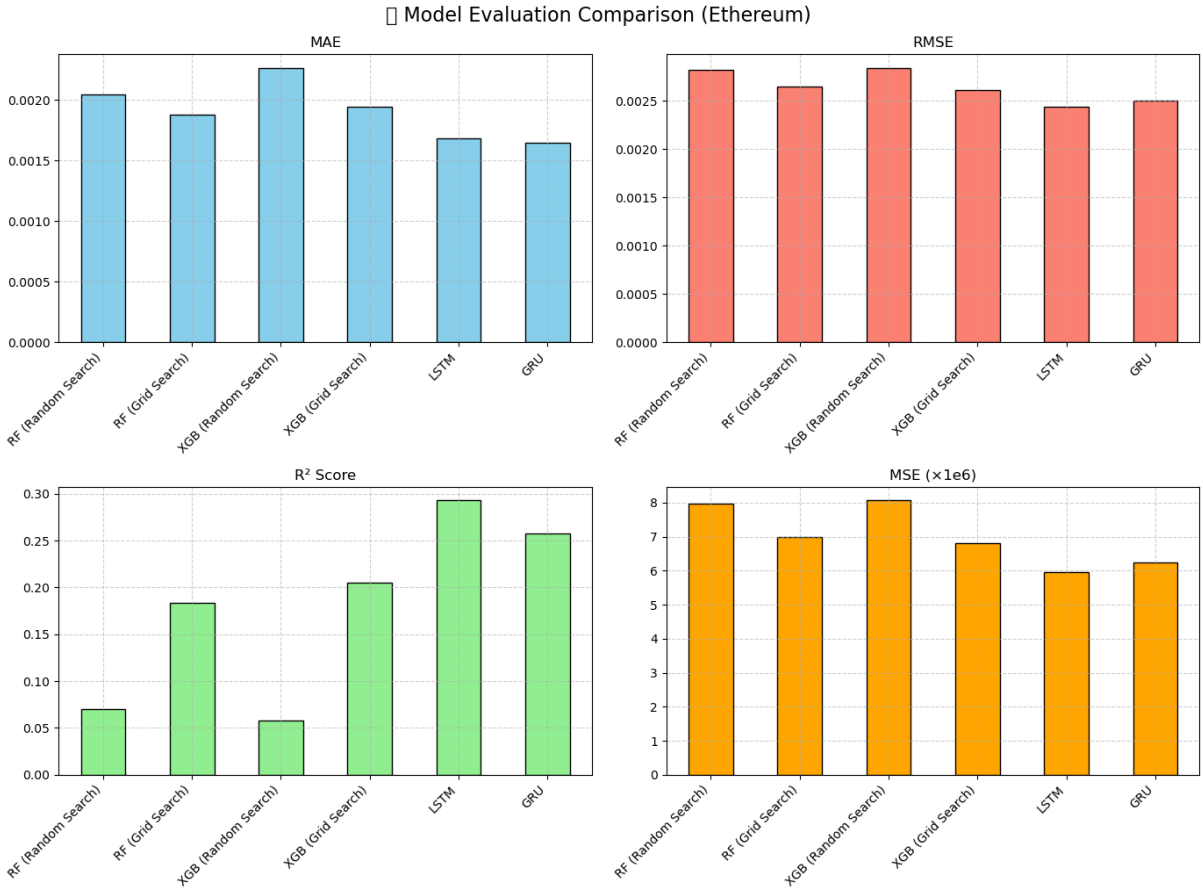

Figure 3.2 Model Evaluation Comparison for ETH dataset

On the test sample, LSTM outperformed every other model on both datasets and on every metric, except MAE of the ETH dataset where GRU was the lead. In general, deep learning models excelled over machine learning models on unseen data, which indicates better generalization.

3.2.3. General Insights and Inferences from The Results:

From the two tables above, we find noteworthy observations and what they could mean:

- RF (Grid) performed strongly on the training data, but was outperformed by deep learning models on the test data. This indicates that deep learning models (LSTM, GRU) exhibit superior generalization to unseen data and are better equipped to capture the non-linear patterns in cryptocurrency data.
- Both RF and XGBoost exhibit an improvement in grid search in comparison with random search across all contexts. Finding the best combination of hyperparameters is inevitable in deterministic exhaustive grid search while it is not in random search.
- For RF, Training time is shorter for grid search than for random search, which is unusual and counter-intuitive for the first glance. Also, RF has the significantly slowest training

time compared to all other models, while XGBoost is much faster than RF and neural networks, taking only seconds for training.

- GRU trains faster than LSTM, but performs slightly behind.
- All models exhibited an improvement in their error metrics scores on the test data when compared to the training data. This can be attributed to the fact that the test data is less volatile than the train data. But at the same time, they all got worse R^2 scores, which might be an indication of overfitting to train data. Though, as stated in the previous chapter, R^2 score, in the context of forecasting time series data, is not an absolutely reliable metric.
- All models have worse scores on the ETH dataset when compared to the BTC dataset, both on train and test. It appears that ETH is more volatile and harder to forecast than BTC. This is supported by the Close price overtime graphs of the two assets presented in chapter 2. (**Figure 2.3** and **Figure 2.4**)
- The best performing model across all metrics and contexts is LSTM. This is due to the fact that LSTM structure is designed for sequential data, which is the case for time series. While the worst model is XGBoost: The reason is that it is a tree-based model, which are prone to noise and overfitting to train data. Also, it doesn't have the "memory" component like LSTM, and it uses boosting instead of bagging like RF, which makes it more sensitive to noise and overfitting.

3.2.4. Comparison with Results from Other Studies:

Table 3.3 Comparing Results with Other Studies

Study	Model	Currency	Metric	Score	This study's score
Dudek et al. [32]	RF (Random Search)	BTC	MAE	**0.00106**	0.00169
			R^2	0.111	**0.1202**
		ETH	MAE	**0.00152**	0.00204
			R^2	**0.160**	0.0698

	LSTM	BTC	MAE	**0.00114**	0.00147
			R^2	0.143	**0.2614**
		ETH	MAE	0.00178	**0.00167**
			R^2	0.165	**0.2928**
Wang et al. [17]	RF (Grid Search)	BTC	RMSE	0.012	**0.00213**
		ETH	RMSE	0.015	**0.00264**
	LSTM	BTC	RMSE	0.014	**0.00203**
		ETH	RMSE	0.017	**0.00243**
Pratas et al. [20]	LSTM	BTC	MAE	**0.00043**	0.00147
Nadarajah et al. [33]	XGB (Assumed Random Search)	BTC	RMSE	**0.000306**	0.00228
		ETH	RMSE	**0.000306**	0.00284

The comparison with other studies shows that the models of this study exhibit a competitive performance, particularly in R^2 scores. For example, LSTM model outperforms Dudek et al. for ETH (MAE: 0.00167 vs. 0.00178, R^2: 0.2928 vs. 0.165), and achieves notably higher R^2 for BTC (0.2614 vs. 0.143). Compared to Wang et al., the models show significantly lower RMSE values. However, XGBoost results from Nadarajah et al. are better than those of XGBoost of this study in terms of RMSE, and LSTM of Pratas et al. exceedingly outperformed the LSTM of this study in MAE score, which implies that some improvements are needed in tuning or feature engineering.

3.3. Diagnostic Checks:

Here, we show plots that are related to the performance of the models across both datasets and we analyze them:

3.3.1. Actual vs Predicted Volatility:

As **Figure 3.3** and **Figure 3.4** shows, LSTM and GRU tracks volatility well on both BTC and ETH datasets, especially when compared to RF and XGB. This goes hand in hand with the results from the evaluation metrics in **Table 3.2**.

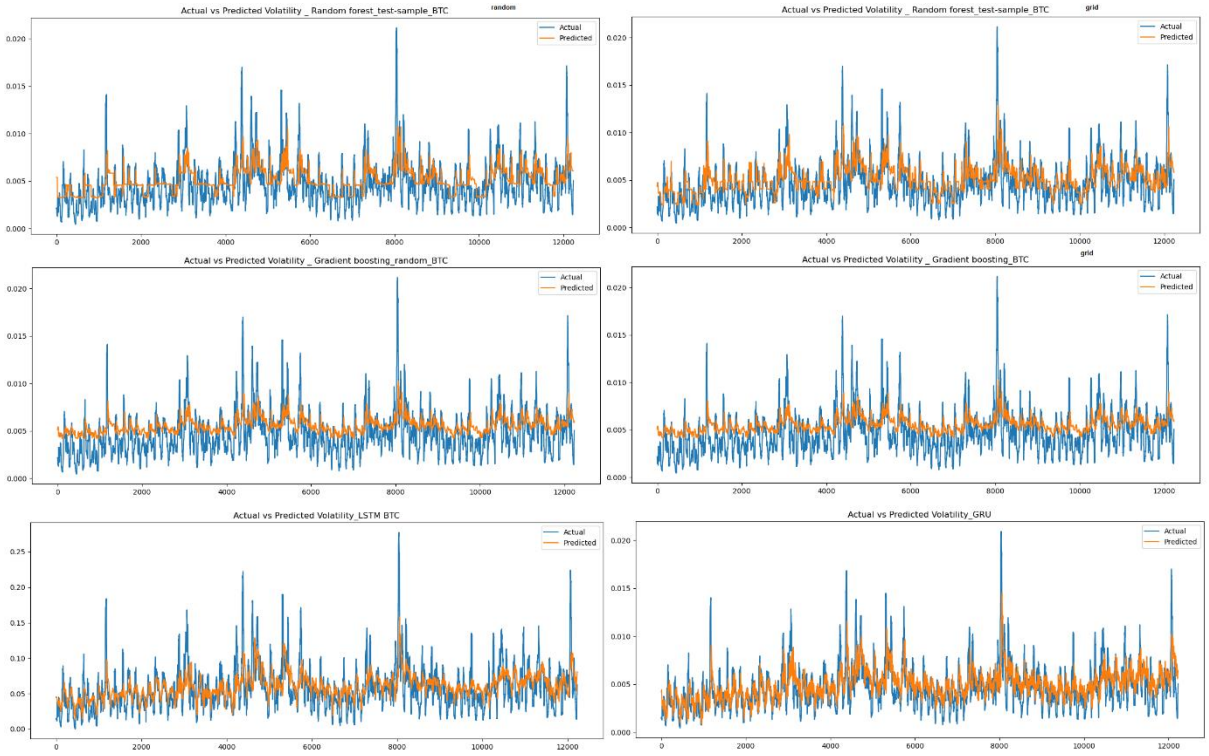

Figure 3.4 Actual vs Predicted Volatility (BTC)

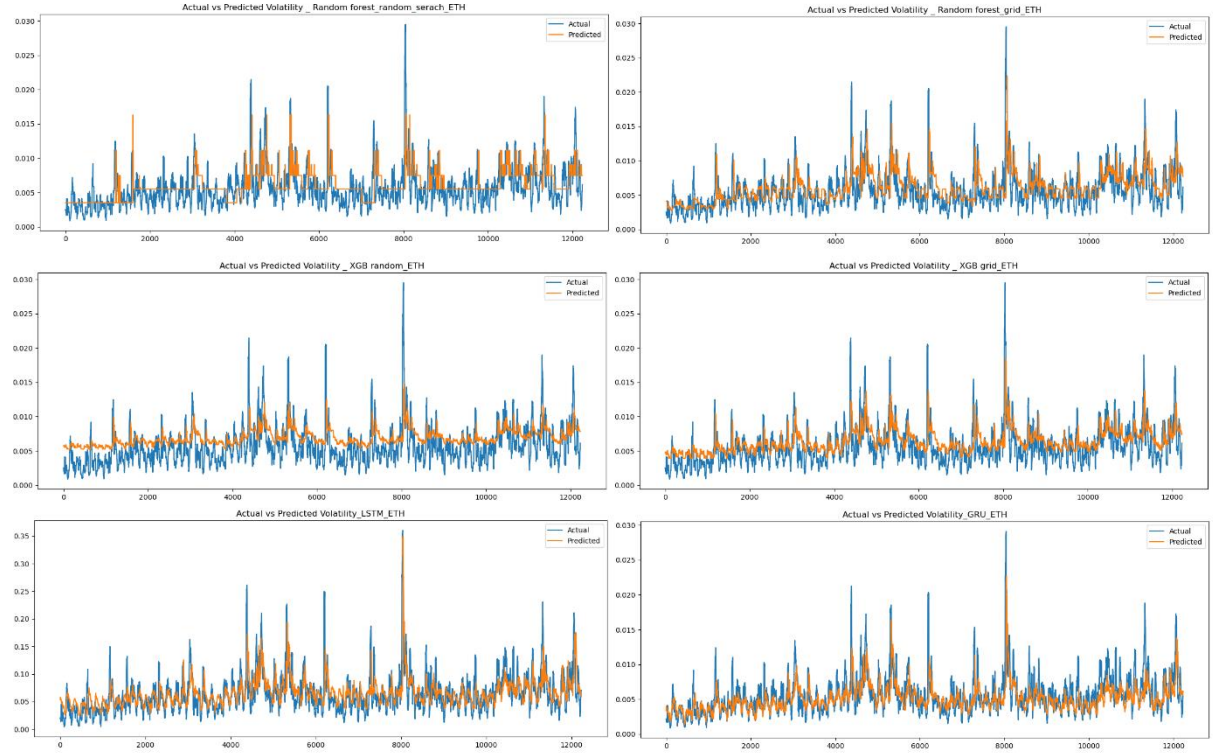

Figure 3.3 Actual vs Predicted Volatility (ETH)

3.3.2. Histograms of Residuals:

Figure 3.5 and **Figure 3.6** depicts the histogram of residuals for both datasets. According to them, LSTM exhibits the best characteristics: center near zero, symmetric bell shape, and narrow spread, though with a tail that suggests large errors during volatility surges, which is expected in time series. Followed by GRU with a slightly heavier tail. While FR and XGB don't show the same good attributes.

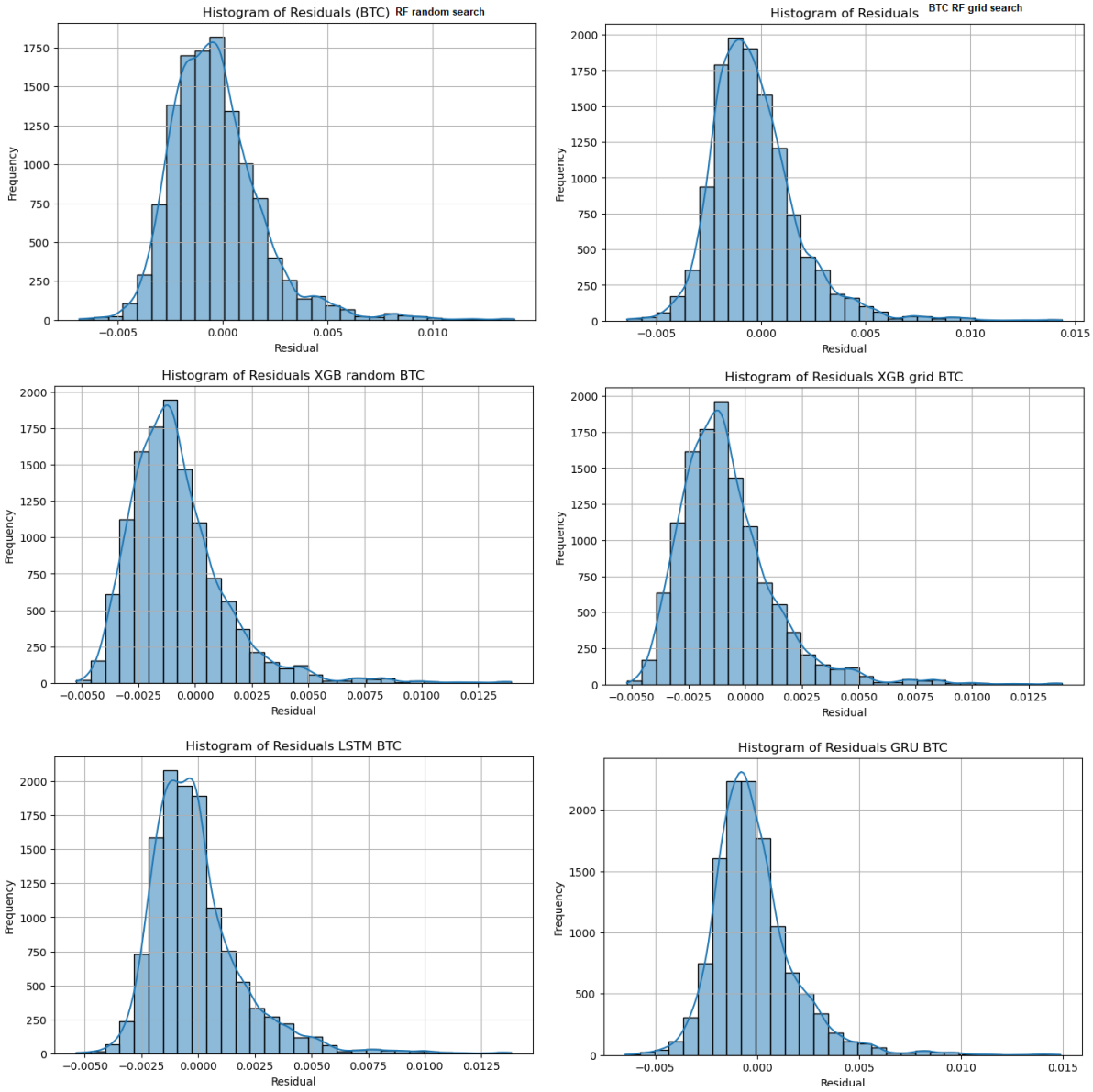

Figure 3.5 Histograms of Residuals (BTC)

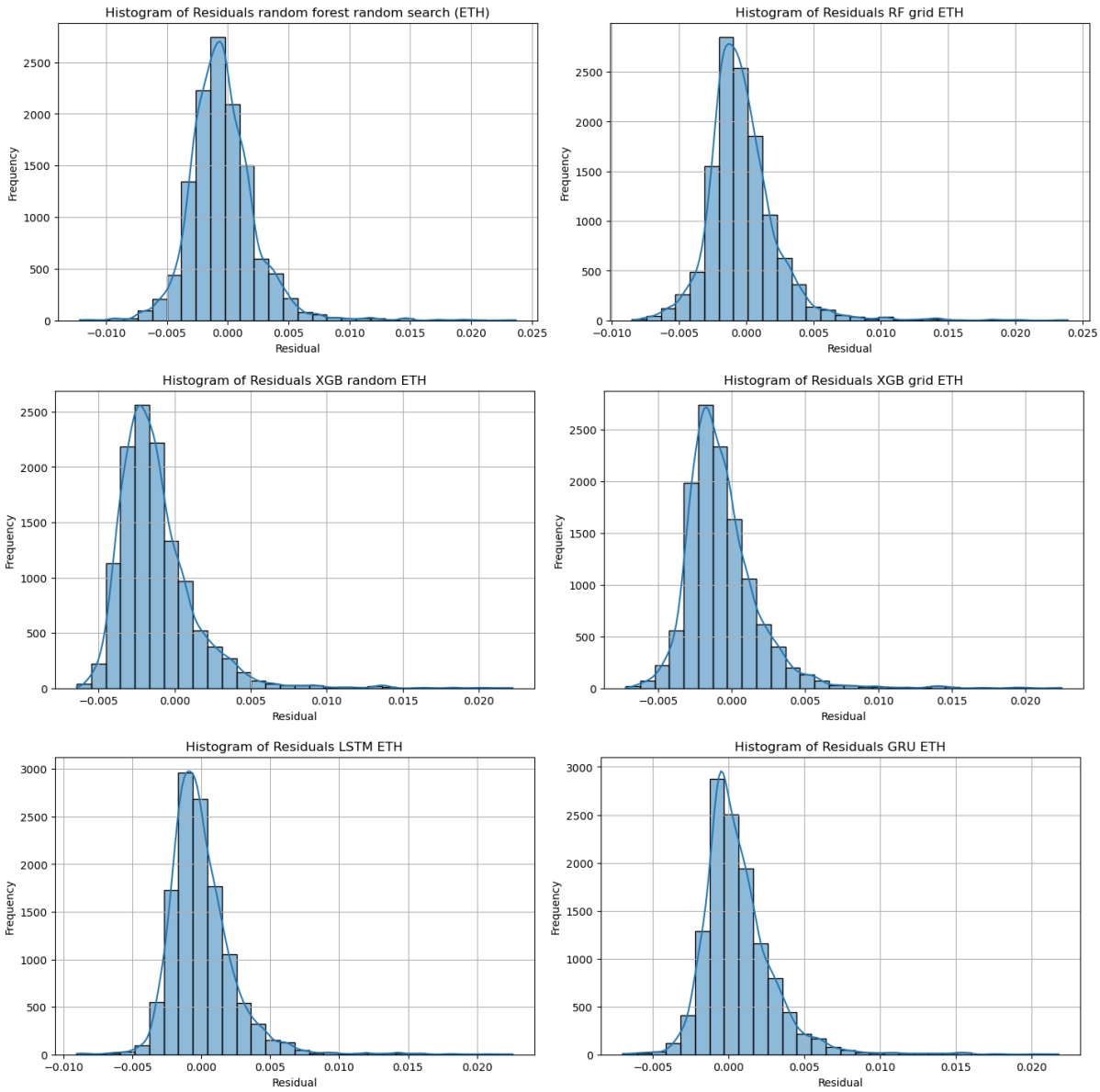

Figure 3.6 Histograms of Residuals (ETH)

3.3.3. Residuals vs Predicted Values:

Figure 3.7 and **Figure 3.8** validate as well the evaluation metrics results: LSTM is the best performing model with most residuals centered around zero with no pattern or shape. At higher predicted volatility (x axis) there is higher scattered residuals in BTC dataset, which indicates underestimation, and the opposite with ETH (overestimation). It is also followed tightly with GRU, with more errors at high volatility on both datasets (both underestimation and overestimation). On the other hand, RF shows less randomness in the scatter shape, especially random search of ETH dataset, and more outliers. XGB is overall better than RF but still not as good as DL models.

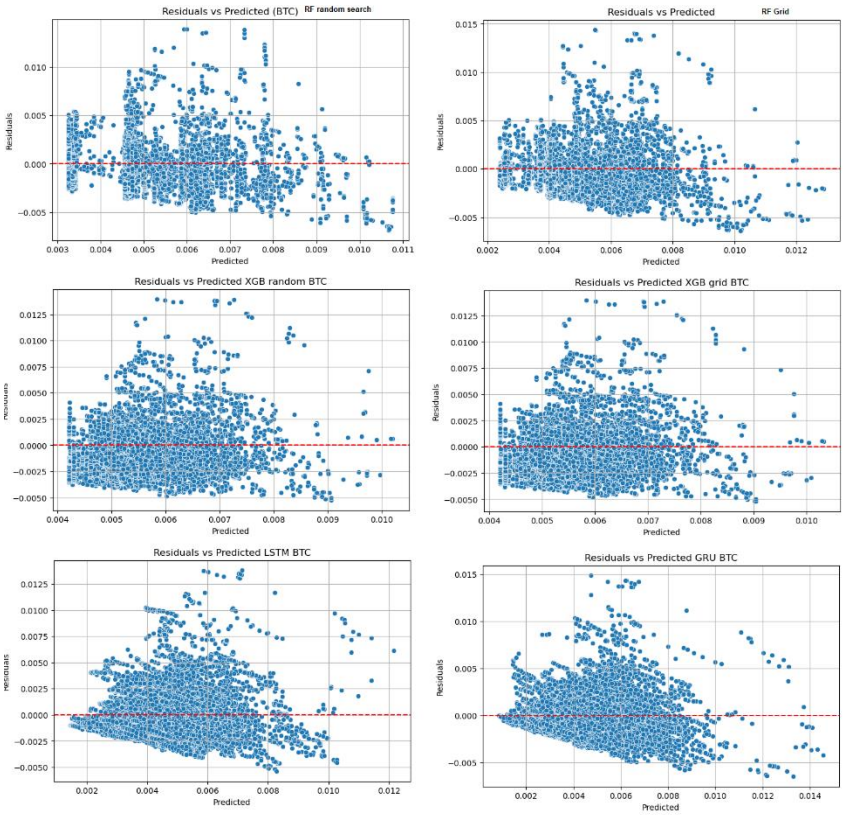

Figure 3.7 Residuals vs Predicted Values (BTC)

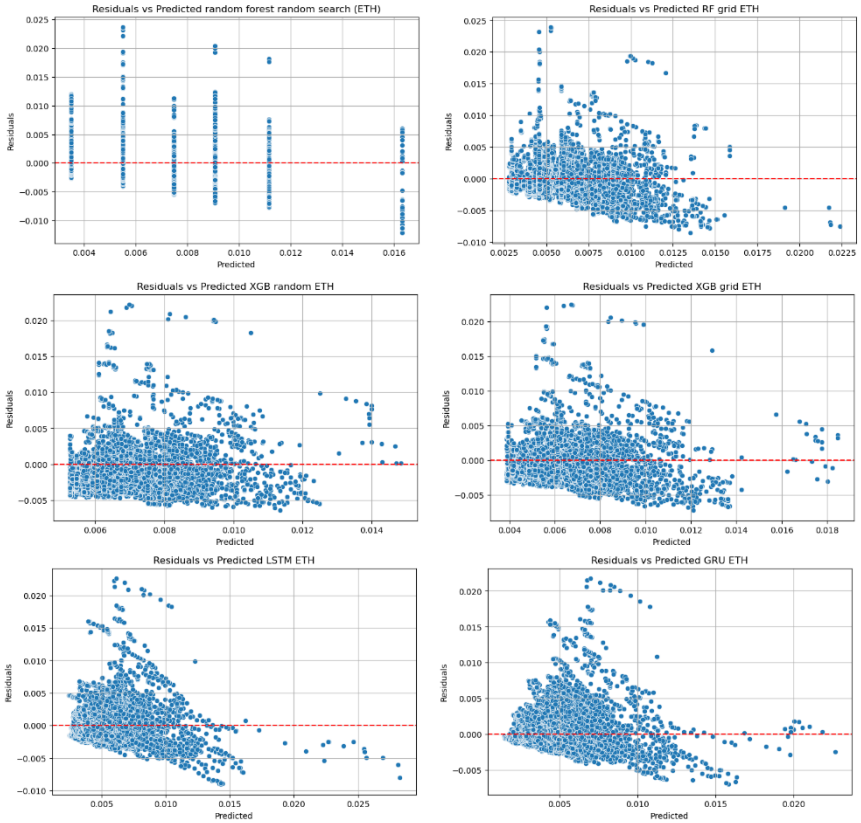

Figure 3.8 Residuals vs Predicted Values (ETH)

3.3.4. Error vs Actual Volatility:

Figure 3.9 and **Figure 3.10** represents error vs actual volatility scatter plot. LSTM shows tight error clusters in low to mid-range actual volatility values with no strong U shape and some error increase in value at high volatility which is not uncommon in data series forecasting. GRU exhibits the same, but with slightly higher errors as usual. RF/XGB only perform well at mid-range (stable

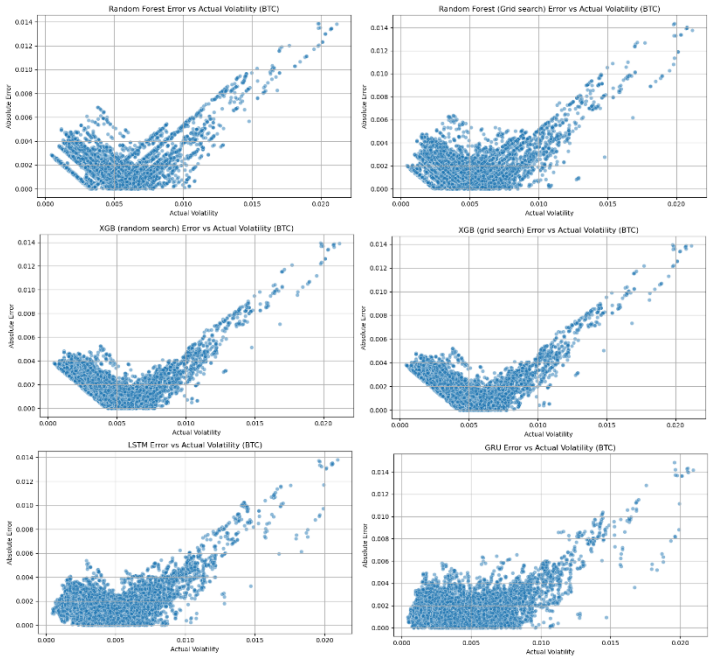

Figure 3.9 Error vs Actual Values (BTC)

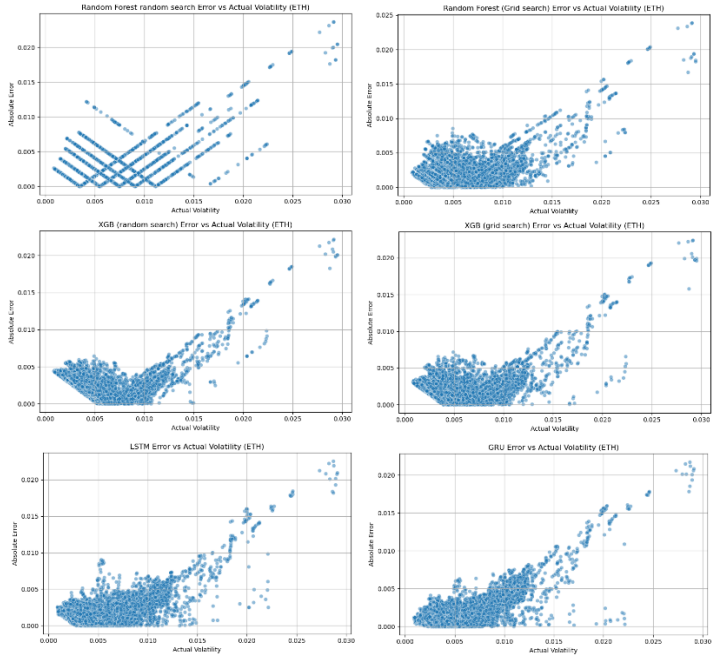

Figure 3.10 Error vs Actual Values (ETH)

3.3.5. Predictions vs Actual Volatility Scatter plot:

Figure 3.11 and **Figure 3.12** show that across the two datasets of BTC and ETH, LSTM comes out as the best performing model followed tightly by GRU. Both models points are tightly clustered around the 45° red line with slight underestimation from LSTM in BTC data and both underestimation and overestimation in ETH data at high actual volatility. XGB performs moderately but still noticeably worse than DL models. Random search RF is the worst with wider scattering points, but with noticeable improvement with grid search.

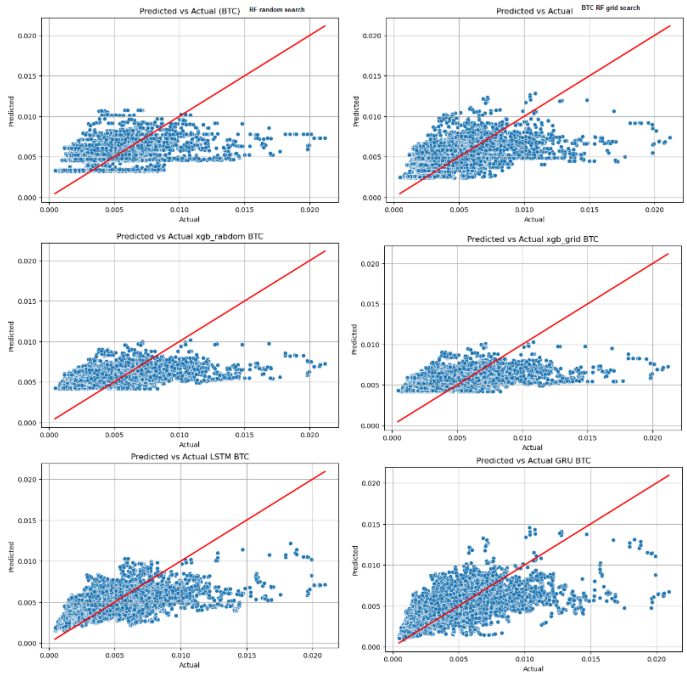

Figure 3.12 Predicted vs Actual (BTC)

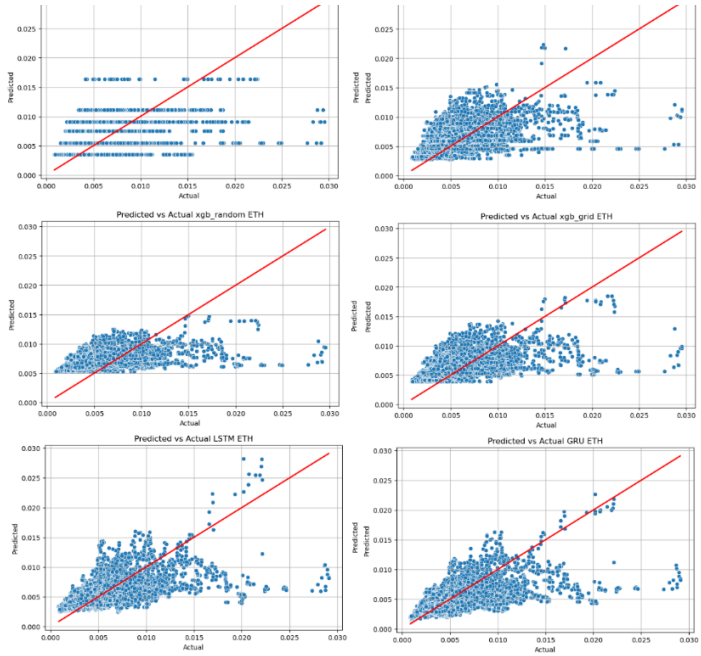

Figure 3.11 Predicted vs Actual (ETH)

3.3.6. Residual Heatmap:

In **Figure 3.13** and **Figure 3.14**, LSTM and GRU show more neutral coloring than RF and XGB which indicates more stability and consistency, except for error spikes during high volatility periods. On the other hand, RF and XGB are more "colorful" with red and blueish spots, suggesting worse handling of volatility shifts. This, again, validates the results of the evaluation metrics, and confirms its reliability.

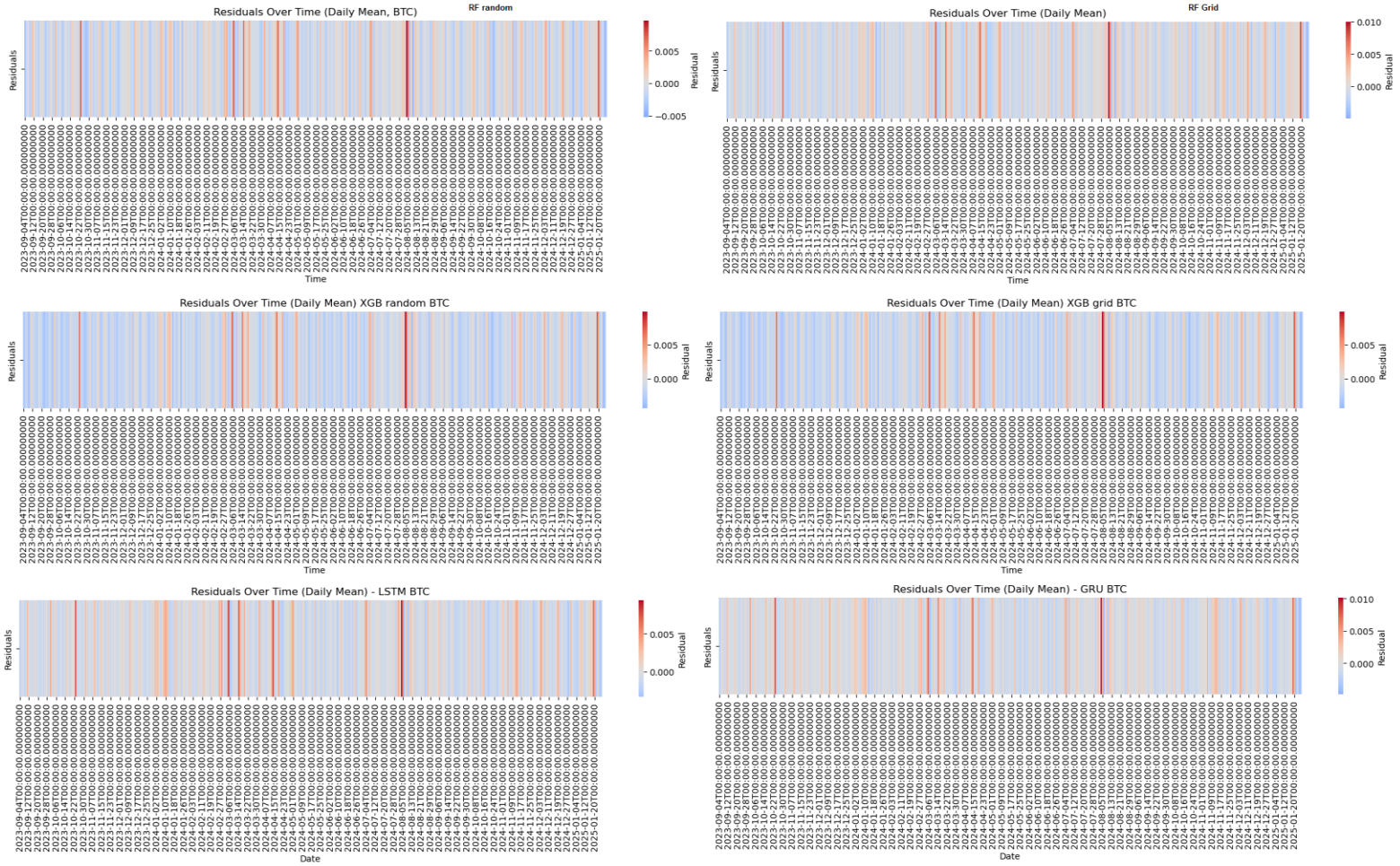

Figure 3.13 Residual Heatmap Overtime (BTC)

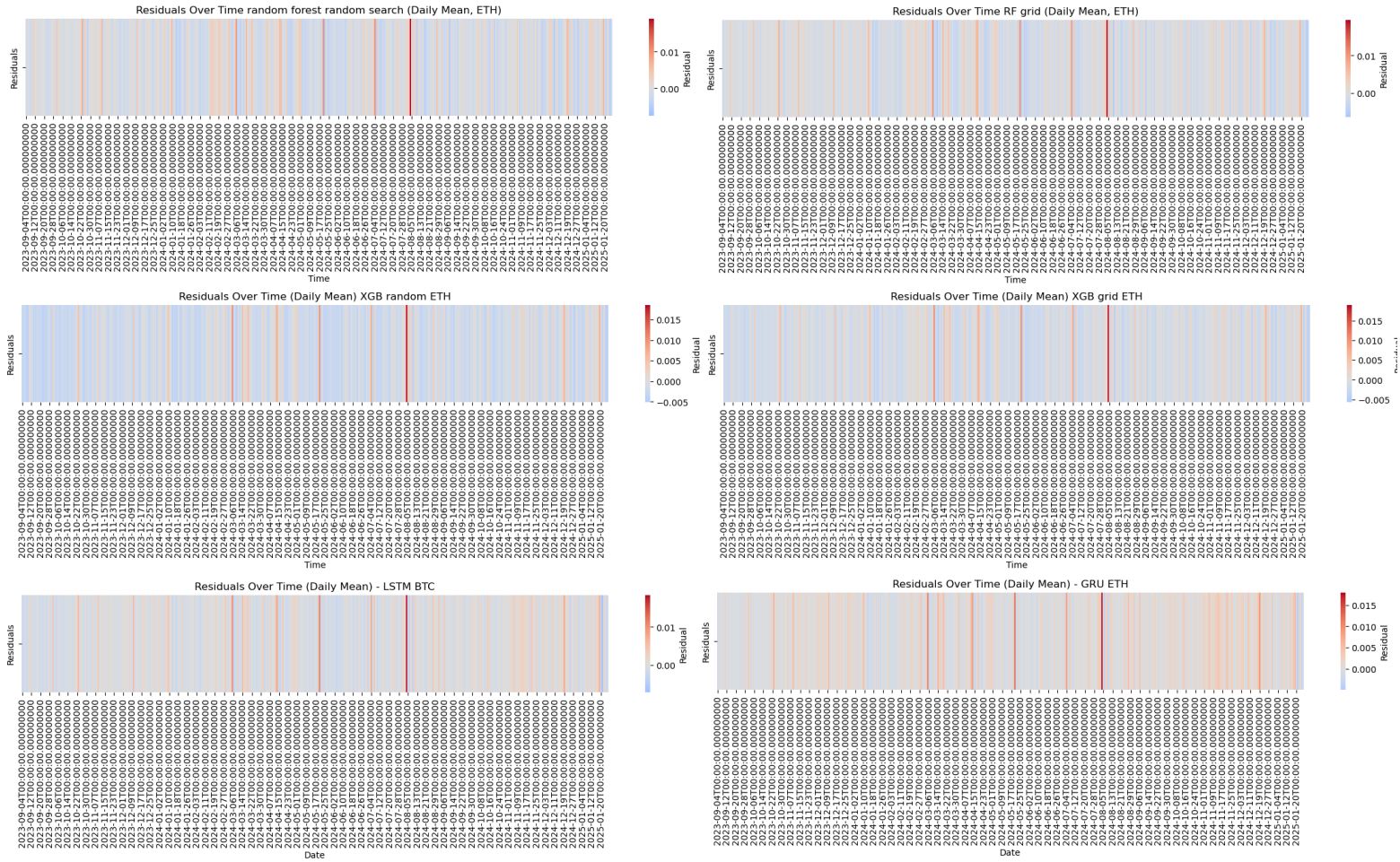

Figure 3.14 Residual Heatmap Overtime (ETH)

3.4. Discussion:

3.4.1. Interpretation of Results:

The results demonstrate the superiority of deep learning models, LSTM and GRU, over traditional machine learning models, namely: Random Forest and XGBoost in forecasting 24-hour cryptocurrency volatility, specifically Bitcoin and Ethereum. This is most evident in LSTM's low MAE, MSE, and RMSE scores on test data across both BTC and ETH datasets, which indicates its ability to generalize on unseen data.

This superiority is attributed to these RNN models' architecture, which gives them the ability to capture temporal dependencies in time series data, unlike tree-based models which lack the memory component and struggle in capturing dynamic, non-linear relations. This is most evident in **Figure 3.15** and **Figure 3.16**, where it is noticeable that the most important features for LSTM and GRU are time-based features like is_weekend, day_of_week, hour, month, weekofyear…etc.

This is absent in tree-based models, where they rely heavily on lagged volatility features, i.e.: direct past numerical values.

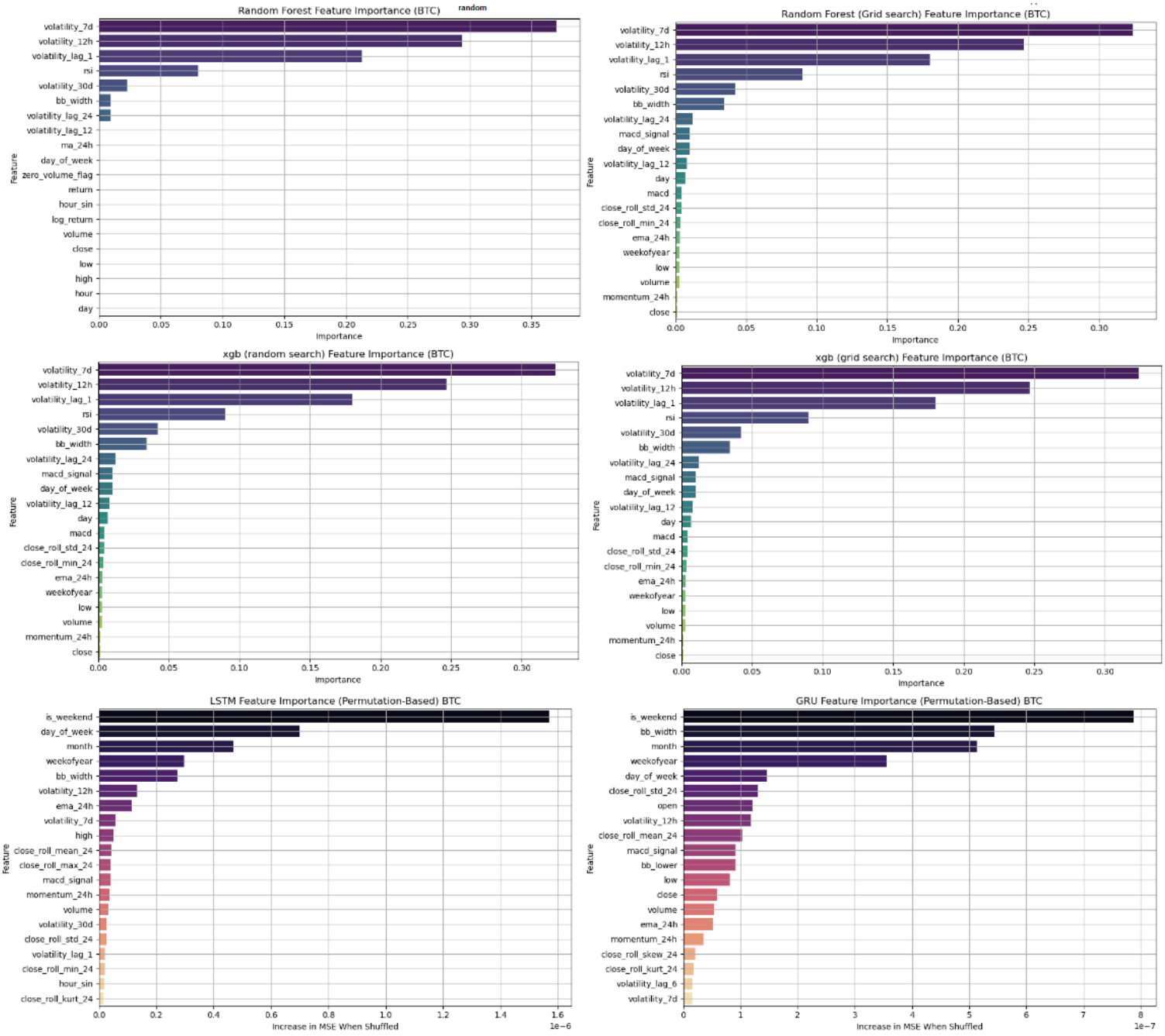

Figure 3.15 Feature Importance (BTC)

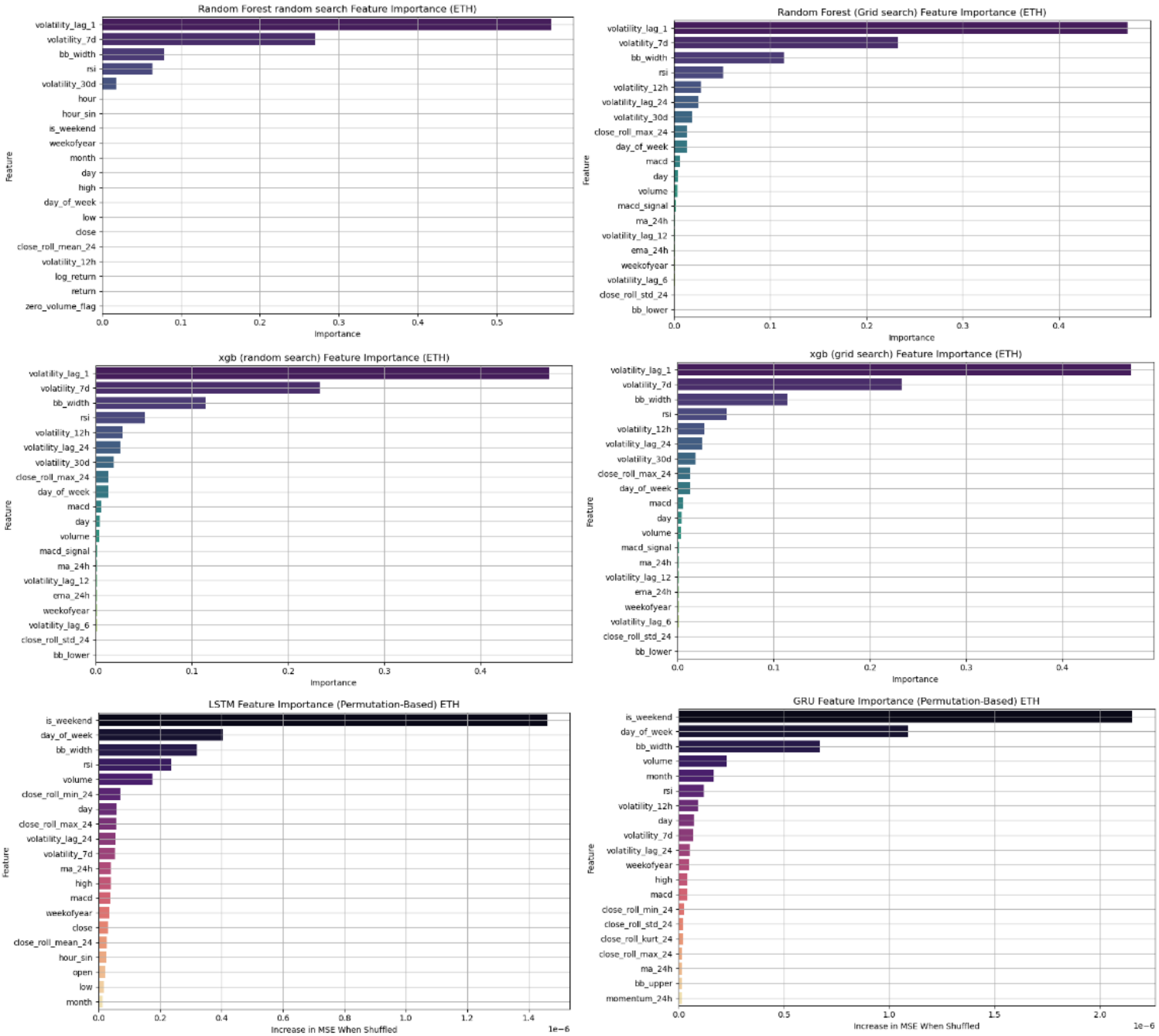

Figure 3.16 Feature Importance (ETH)

These features allow the deep learning models to capture volatility behavior tied to human activities and market dynamics, which are often cyclic. In contrast, tree-based models rely heavily on lagged volatility and some rolling stats, while almost entirely ignore the time-based features or give them only minor importance. This also explains the more consistent and robust performance of LSTM and GRU across time periods and volatility regimes in all visualized plots compared to RF and XGB that meanwhile suffer from noticeable error increase.

While both LSTM and GRU perform well, LSTM slightly outperform GRU in both datasets and across almost all metrics. This is due to the more complex architecture of LSTM, which allows it to treat data more thoroughly due to higher parameters count, but on the expense of computational cost.

There is also the difference of models' performance between the BTC and ETH datasets. Almost all models perform worse on the ETH dataset across all metrics. This is likely due to the fact that the ETH dataset is more volatile and thus harder to forecast than the BTC dataset.

Finally, the reason why RF training time is shorter in grid search than it is in random search is that there are fewer parameters in grid search than random search as **Table 2.6** shows. Additionally, RF is significantly slower than other models because it only uses the CPU—it doesn't have access to GPU acceleration like LSTM and GRU, and lacks the computational efficiency and optimization of XGBoost. The latter despite being the fastest model in training, it was also the worst performing on all metrics and datasets.

3.4.2. Limitations of Work:

There are certain limitations to this work that should be acknowledged:

- **Data Limitations:** The first limitation is the datasets themselves. The models of this study where trained on data of just two cryptocurrencies: Bitcoin and Ethereum, and for a specific period of time. This might not make them useful in forecasting other cryptocurrencies, especially smaller ones with totally different market behavior. Additionally, the datasets are not exactly of a High-frequency nature: data with minute-to-minute or even second-to-second time intervals or ticks would have been better at providing more patterns and relations that might help the models performance of forecasting the volatility, despite the more noise it would exhibit and higher computational cost that it would certainly require. Also, it might help the traders who are interested in forecasting over shorter horizons, like 1-hour horizon. Finally, market sentiment was not included in the data. Several studies have concluded that sentiment has a direct impact on volatility, and by that, on models' performance: "*Market sentiment, informed by the news, social media, and influential voices, often drives price movements. In the case of highly influential people, like Elon Musk, one tweet can have high effects on virtual currencies such as Dogecoin and Bitcoin*". [19]
- **Model Limitations:** Despite deep learning models outperforming other models in this study, they have their own shortcomings: an important one is their **Black Box** nature. This means that while they may produce good predictions, it's often very hard to understand

why they made those predictions. This is because deep learning models involve complex layered build with complex mathematical operations that make it a challenging task to understand the reasoning behind those outputs. This might be described as a **lack of transparency**, and it might be a problem for risk managers and financiers to make decisions based on the output of such models: "*Advanced algorithms, such as deep learning models, often behave like "black boxes" in which end-users do not understand how decisions are made. This lack of transparency deters investors and regulators from fully embracing or trusting such tools*". [19] Another problem is the computational cost required by these models to be trained. They often need more resources and time, especially with high frequency data. They also require retraining on a regular basis due to the ever-changing nature of the market; this is an aspect of what is called **Alpha Decay**: *"...since financial time series data are constantly evolving, no model would be able to consistently forecast with high accuracy level forever. The average lifetime of a model is between 6 months to 5 years, and there's a phenomenon in quant trading that is called **alpha decay**, which is the loss in predictive power of an alpha model over time*". [2] Finally, the study did not test hybrid models; whether they were a combination of statistical models such as GARCH-type models and machine learning/deep learning models, or a combination of machine learning models themselves. Some studies suggested the superiority of such models over single models, and it would have been interesting and informative to compare a model of this type with the models of this study: "*Nowadays combined classical econometric methods as well as methods of machine learning (Albuquerque et al. 2018; Wang et al. 2018) and those which take into consideration the spirit of social networks regarding the state and tendency of cryptocurrency dynamics (Kennis 2018) are becoming more popular*". [24]

- **Generalizability:** The findings of this study may not apply to different cryptocurrencies, or different forecasting horizons. The models were not tested over longer or shorter time horizons than 1 day or 24-hours. Also, they were not tested on datasets of other cryptocurrencies.

3.5. Conclusion:

This chapter provided a thorough assessment of multiple volatility forecasting models using BTC and ETH datasets. Overall, LSTM was the standout model, showing the best predictive performance, accuracy, and robustness as the evaluation measures and diagnostic checks indicated. GRU was a close second, having scored less, but just with a small margin. Random Forest and

XGBoost, though improved on some metrics with grid search as a strategy in comparison with random search, they were unable to keep up with deep learning models in modeling expected sequential dependencies and volatility regime shifts. All the diagnostic plots provided strong evidence in support of the evaluation metrics results: LSTM and GRU were shown to have the capacity to generalize and remain stable over time, and manage high volatility regime shifts much more effectively than the tree-based models. But these results remain limited to their source datasets, models selected to be utilized, and forecast horizon.

GENERAL CONCLUSION

In this study, LSTM was shown to be the best performing model out of the models tested, on every metric and dataset, followed closely by GRU. In general, deep learning models, specifically RNN models, outperformed machine learning tree-based models (Random Forest and XGBoost) in forecasting BTC and ETH volatility 24-hours ahead. This is attributed to the design of RNN models which is well-suited to capture temporal dependencies in sequential time series data, and which possesses a memory component that allows these models to retain useful information and discard other information. In contrast, Random Forest and XGBoost that couldn't keep up with DL models, especially on unseen test data, are not structured to recognize the sequential nature of data naturally, and rely heavily on engineered features, mainly the ones derived from the target variable. Overall, while the current results are promising, there is room for improvement through parameter tuning, feature engineering, and optimization techniques.

However, it is important to acknowledge the limitations of this study. First, the datasets include only two cryptocurrencies, lack a higher frequency, and do not contain sentiment information. Second, the best performing DL models have a black-box nature that affects the transparency of the results and poses challenges to financial actors that might rely on them. They also have a high computational cost and are susceptible to alpha decay—they need regular retraining. Third, hybrid models that combine traditional and ML approaches were not explored. Lastly, the findings may not generalize to other cryptocurrencies or forecasting horizons longer the 24-hour.

Research in the future should involve datasets of a higher frequency nature, enriched with sentiment data, and contain more cryptocurrencies. Also, it should explore hybrid models, assess performance in different timeframes, both shorter and longer, and address the issue of computational efficiency.

BIBLIOGRAPHY & WEBOGRAPHY

[1] Farman Ullah Khan, F. Khan, and Parvez Ahmed Shaikh, “Forecasting returns volatility of cryptocurrency by applying various deep learning algorithms,” Future Business Journal, vol. 9, no. 1, Jun. 2023, doi: https://doi.org/10.1186/s43093-023-00200-9.

[2] C. Bui, “chibui191/bitcoin_volatility_forecasting: GARCH and Multivariate LSTM forecasting models for Bitcoin realized volatility with potential applications in crypto options trading, hedging, portfolio management, and risk management,” GitHub, 2025. https://github.com/chibui191/bitcoin_volatility_forecasting (accessed Apr. 23, 2025).

[3] B. Amirshahi and S. Lahmiri, “Hybrid deep learning and GARCH-family models for forecasting volatility of cryptocurrencies,” Machine Learning with Applications, vol. 12, p. 100465, Apr. 2023, doi: https://doi.org/10.1016/j.mlwa.2023.100465.

[4] T. G. Andersen, T. Bollerslev, P. F. Christoffersen, and F. X. Diebold, “Chapter 15 Volatility and Correlation Forecasting,” Handbook of Economic Forecasting, pp. 777–878, 2006, doi: https://doi.org/10.1016/s1574-0706(05)01015-3.

[5] C. Brooks, Introductory Econometrics for Finance, 4th ed. Cambridge, United Kingdom ; New York, Ny: Cambridge University Press, 2019, pp. 507–541.

[6] A. Harvey, E. Ruiz, and N. Shephard, “Multivariate Stochastic Variance Models,” *The Review of Economic Studies*, vol. 61, no. 2, pp. 247–264, Apr. 1994, doi: https://doi.org/10.2307/2297980.

[7] E. Lundgren, F. Viitanen, and K. Podgórski, “Performance of Stochastic Volatility and GARCH Models in Different Market Regimes,” 2022. Accessed: Apr. 12, 2025. [Online]. Available: https://lup.lub.lu.se/student-papers/record/9071849/file/9071851.pdf

[8] H. Freitas Lopes, Stochastic Volatility Models . University of Chicago Booth School of Business. Accessed: Apr. 20, 2025. [Online]. Available: https://hedibert.org/wp-content/uploads/2013/12/sv-models.pdf

[9] J.-M. Kim, C. Jun, and J. Lee, “Forecasting the Volatility of the Cryptocurrency Market by

GARCH and Stochastic Volatility," Mathematics, vol. 9, no. 14, p. 1614, Jul. 2021, doi: https://doi.org/10.3390/math9141614.

[10] A. H. Dyhrberg, "Bitcoin, gold and the dollar – A GARCH volatility analysis," Finance Research Letters, vol. 16, no. 1544–6123, pp. 85–92, Feb. 2016, doi: https://doi.org/10.1016/j.frl.2015.10.008.

[11] J. Chu, S. Chan, S. Nadarajah, and J. Osterrieder, "GARCH Modelling of Cryptocurrencies," Journal of Risk and Financial Management, vol. 10, no. 4, p. 17, Oct. 2017, doi: https://doi.org/10.3390/jrfm10040017.

[12] L. Maciel, "Cryptocurrencies value-at-risk and expected shortfall: Do regime-switching volatility models improve forecasting?," International Journal of Finance & Economics, Aug. 2020, doi: https://doi.org/10.1002/ijfe.2043.

[13] W. Sun and Ladislav Kristoufek, "Beyond GARCH in cryptocurrency volatility modelling: superiority of range-based estimators," Applied Economics Letters, pp. 1–8, Jun. 2024, doi: https://doi.org/10.1080/13504851.2024.2363295.

[14] J. Prüser, "Forecasting the Risk of Cryptocurrencies: Comparison and Combination of GARCH and Stochastic Volatility Models," SSRN Electronic Journal, 2023, doi: https://doi.org/10.2139/ssrn.4359827.

[15] Z. Chen, J. Liu, and X. Hao, "Can the 'good-bad' volatility and the leverage effect improve the prediction of cryptocurrency volatility?—Evidence from SHARV-MGJR model," Finance Research Letters, vol. 67, p. 105757, Sep. 2024, doi: https://doi.org/10.1016/j.frl.2024.105757.

[16] R. Khaldi, A. El Afia, and R. Chiheb, "Forecasting of BTC volatility: comparative study between parametric and nonparametric models," Progress in Artificial Intelligence, vol. 8, no. 4, pp. 511–523, Jul. 2019, doi: https://doi.org/10.1007/s13748-019-00196-w.

[17] Y. Wang, G. Andreeva, and B. Martin-Barragan, "Machine learning approaches to forecasting cryptocurrency volatility: Considering internal and external determinants," International Review of Financial Analysis, vol. 90, p. 102914, Nov. 2023, doi: https://doi.org/10.1016/j.irfa.2023.102914.

[18] V. Derbentsev, N. Datsenko, O. Stepanenko, and V. Bezkorovainyi, “Forecasting cryptocurrency prices time series using machine learning approach,” *SHS Web of Conferences*, vol. 65, p. 02001, 2019, doi: https://doi.org/10.1051/shsconf/20196502001.

[19] M. Z. Islam et al., “Evaluating The Effectiveness of Machine Learning Algorithms In Predicting Cryptocurrency Prices Under Market Volatility: A Study Based On The Usa Financial Market,” The American Journal of Management and Economics Innovations, vol. 06, no. 12, pp. 15–38, Dec. 2024, doi: https://doi.org/10.37547/tajmei/volume06issue12-03.

[20] T. E. Pratas, F. Ramos, and L. Rubio, “Forecasting bitcoin volatility: exploring the potential of deep learning,” Eurasian Economic Review, vol. 13, pp. 285–305, Jun. 2023, doi: https://doi.org/10.1007/s40822-023-00232-0.

[21] A. M. Rather, “A new method of ensemble learning: case of cryptocurrency price prediction,” Knowledge and Information Systems, Dec. 2022, doi: https://doi.org/10.1007/s10115-022-01796-0.

[22] V. Derbentsev, N. Datsenko, V. Babenko, O. Pushko, and O. Pursky, “Forecasting Cryptocurrency Prices Using Ensembles-Based Machine Learning Approach,” 2020 IEEE International Conference on Problems of Infocommunications. Science and Technology (PIC S&T), Oct. 2020, doi: https://doi.org/10.1109/picst51311.2020.9468090.

[23] A. Bouteska, M. Z. Abedin, P. Hajek, and K. Yuan, “Cryptocurrency price forecasting – A comparative analysis of ensemble learning and deep learning methods,” International Review of Financial Analysis, vol. 92, p. 103055, Mar. 2024, doi: https://doi.org/10.1016/j.irfa.2023.103055.

[24] V. Derbentsev, A. Matviychuk, and V. N. Soloviev, “Forecasting of Cryptocurrency Prices Using Machine Learning,” *Advanced Studies of Financial Technologies and Cryptocurrency Markets*, pp. 211–231, 2020, doi: https://doi.org/10.1007/978-981-15-4498-9_12.

[25] Keboola, “The Ultimate Guide to Random Forest Regression,” www.keboola.com, Sep. 17, 2020. https://www.keboola.com/blog/random-forest-regression (accessed May 04, 2025).

[26] A. Brauneis and M. Sahiner, “Crypto Volatility Forecasting: Mounting a HAR, Sentiment,

and Machine Learning Horserace," Asia-Pacific Financial Markets, Dec. 2024, doi: https://doi.org/10.1007/s10690-024-09510-6.

[27] Y. Wang, Z. Pan, J. Zheng, L. Qian, and M. Li, "A hybrid ensemble method for pulsar candidate classification," Astrophysics and Space Science, vol. 364, no. 8, Aug. 2019, doi: https://doi.org/10.1007/s10509-019-3602-4.

[28] I. Nasirtafreshi, "Forecasting cryptocurrency prices using Recurrent Neural Network and Long Short-term Memory," Data & Knowledge Engineering, p. 102009, Mar. 2022, doi: https://doi.org/10.1016/j.datak.2022.102009.

[29] P. L. Seabe, C. R. B. Moutsinga, and E. Pindza, "Forecasting Cryptocurrency Prices Using LSTM, GRU, and Bi-Directional LSTM: A Deep Learning Approach," Fractal and Fractional, vol. 7, no. 2, p. 203, Feb. 2023, doi: https://doi.org/10.3390/fractalfract7020203.

[30] H. Hewamalage, C. Bergmeir, and K. Bandara, "Recurrent Neural Networks for Time Series Forecasting: Current status and future directions," International Journal of Forecasting, vol. 37, no. 1, pp. 388–427, Jan. 2021, doi: https://doi.org/10.1016/j.ijforecast.2020.06.008.

[31] Yashwanth S, "Deep Learning Series 13: GRU(Gated Recurrent Unit) - Yashwanth S - Medium," Medium, Dec. 2024. https://medium.com/%40yashwanths_29644/deep-learning-series-13-gru-gated-recurrent-unit-7374776329c7 (accessed May 05, 2025).

[32] G. Dudek, Piotr Fiszeder, P. Kobus, and Witold Orzeszko, "Forecasting Cryptocurrencies Volatility Using Statistical and Machine Learning Methods: A Comparative Study," Applied Soft Computing, vol. 151, pp. 111132–111132, Jan. 2024, doi: https://doi.org/10.1016/j.asoc.2023.111132.

[33] S. Nadarajah, J. C. Mba, P. Rakotomarolahy, and H. T. J. E. Ratolojanahary, "Ensemble Learning and an Adaptive Neuro-Fuzzy Inference System for Cryptocurrency Volatility Forecasting," Journal of Risk and Financial Management, vol. 18, no. 2, p. 52, Jan. 2025, doi: https://doi.org/10.3390/jrfm18020052.

[34] Z.-C. Huang, I. Sangiorgi, and A. Urquhart, "Forecasting Bitcoin volatility using machine learning techniques," *Journal of International Financial Markets Institutions and Money*, vol. 97,

pp. 102064–102064, Oct. 2024, doi: https://doi.org/10.1016/j.intfin.2024.102064.

[35] Z. Zheng, “A Machine Learning Approach to Cryptocurrency Volatility and Financial Implications,” Advances in Economics Management and Political Sciences, vol. 151, no. 1, pp. 133–139, Jan. 2025, doi: https://doi.org/10.54254/2754-1169/2024.19417.

[36] Mimoza Mjoska, Blagoj Ristevski, Snezana Savoska, and Vladimir Trajkovik, “Predicting Bitcoin Volatility Using Machine Learning Algorithms and Blockchain Technology,” Information Systems and Grid Technologies (ISGT 2022), May 2022, Available: https://www.researchgate.net/publication/363045700_Predicting_Bitcoin_Volatility_Using_Machine_Learning_Algorithms_and_Blockchain_Technology

[37] Siu Hin Tang, M. Rosenbaum, and C. Zhou, “Forecasting volatility with machine learning and rough volatility: example from the crypto-winter,” Digital Finance/Digital finance, vol. 6, pp. 639–655, Mar. 2024, doi: https://doi.org/10.1007/s42521-024-00108-1.

[38] Ethereum, “What is Ethereum?,” ethereum.org. https://ethereum.org/en/what-is-ethereum/ (accessed May 01, 2025).

[39] “Bitcoin Dominance | CoinMarketCap,” CoinMarketCap, 2024. https://coinmarketcap.com/charts/bitcoin-dominance/ (accessed May 01, 2025).

[40] B. Strack, “74% of Institutions Plan To Buy Crypto: Fidelity Survey,” Blockworks, Oct. 27, 2022. https://blockworks.co/news/74-of-institutions-plan-to-buy-crypto-fidelity-survey (accessed May 01, 2025).

[41] S. P. Lee, “Crypto Trading Volumes Reached $18.83T in 2024, Still Below 2021’s $25.21T Peak,” CoinGecko, Feb. 12, 2025. https://www.coingecko.com/research/publications/largest-centralized-crypto-exchanges (accessed May 01, 2025).

[42] “CoinAPI Blog - Understanding OHLCV in market data analysis,” www.coinapi.io. https://www.coinapi.io/blog/understanding-ohlcv-in-market-data-analysis (accessed May 01, 2025).

[43] “Reddit - The heart of the internet,” Reddit.com, 2021.

https://www.reddit.com/r/binance/comments/lhd3qk/temporary_system_maintenance_complete_20210211 (accessed May 01, 2025).

[44] X (formerly Twitter), Mar. 24, 2023. https://x.com/binance/status/1639230280119775234 (accessed May 01, 2025).

[45] Lev Borodovsky and M. Lore, Professional's Handbook of Financial Risk Management. Elsevier, 2000. Accessed: May 03, 2025. [Online]. Available: https://books.google.dz/books?hl=en&lr=&id=V4KLbnIOQdQC&oi=fnd&pg=PA42&dq=volatility+defined+as+the+standard+deviation+of+log+returns&ots=fwKmIZhHTi&sig=NnWdcFqlBTwz3xdl1-hTFV-9qN4&redir_esc=y#v=onepage&q&f=false

[46] T. Angelidis and S. Degiannakis, "Volatility forecasting: Intra-day versus inter-day models," Journal of International Financial Markets, Institutions and Money, vol. 18, no. 5, pp. 449–465, Dec. 2008, doi: https://doi.org/10.1016/j.intfin.2007.07.001.

[47] D. P. Louzis, S. Xanthopoulos-Sisinis, and A. P. Refenes, "The Role of High-Frequency Intra-daily Data, Daily Range and Implied Volatility in Multi-period Value-at-Risk Forecasting," Journal of Forecasting, vol. 32, no. 6, pp. 561–576, Jun. 2013, doi: https://doi.org/10.1002/for.2249.

[48] G. Petneházi, "Recurrent Neural Networks for Time Series Forecasting," *arXiv:1901.00069 [cs, stat]*, Dec. 2018, Available: https://arxiv.org/abs/1901.00069

[49] R. Agrawal, "Know The Best Evaluation Metrics for Your Regression Model !," Analytics Vidhya, May 19, 2021. https://www.analyticsvidhya.com/blog/2021/05/know-the-best-evaluation-metrics-for-your-regression-model/?utm_source=chatgpt.com (accessed May 05, 2025).

[50] D. Andrés, "Error Metrics for Time Series Forecasting - ML Pills," Jun. 22, 2023. https://mlpills.dev/time-series/error-metrics-for-time-series-forecasting/ (accessed May 05, 2025).

[51] S. Taylor, "R-Squared," Corporate Finance Institute. https://corporatefinanceinstitute.com/resources/data-science/r-squared/ (accessed May 05, 2025).

[52] “What is the problem with using R-squared in time series models? | ResearchGate,” ResearchGate, 2020. https://www.researchgate.net/post/What-is-the-problem-with-using-R-squared-in-time-series-models (accessed May 05, 2025).

[53] S. Likens, “Making sense of bitcoin and blockchain,” *PwC*, 2023. https://www.pwc.com/us/en/industries/financial-services/fintech/bitcoin-blockchain-cryptocurrency.html (accessed Jun. 04, 2025).

[54] C. T. Nguyen, D. T. Hoang, D. N. Nguyen, D. Niyato, H. T. Nguyen, and E. Dutkiewicz, “Proof-of-Stake Consensus Mechanisms for Future Blockchain Networks: Fundamentals, Applications and Opportunities,” *IEEE Access*, vol. 7, pp. 85727–85745, 2019, doi: https://doi.org/10.1109/access.2019.2925010.

[55] G. Nicolini and S. Intini, *The Cryptocurrency Phenomenon*. Abingdon, Oxon New York, NY: Taylor & Francis, 2023. Accessed: Jun. 04, 2025. [Online]. Available: https://books.google.dz/books?id=Y7zHEAAAQBAJ&lpg=PT53&pg=PP1#v=onepage&q&f=false

[56] “Crypto market volatility: What it is and how to navigate it | Kraken,” *Kraken.com*, 2022. https://www.kraken.com/learn/crypto-market-volatility (accessed Jun. 04, 2025).

[57] Flipster, “Flipster Blog | What Causes Crypto Market Volatility: How to Mitigate Risk in Volatile Markets,” *Flipster*, Jan. 31, 2025. https://flipster.io/en/blog/what-causes-crypto-market-volatility-how-to-mitigate-risk-in-volatile (accessed Jun. 04, 2025).

ABSTRACT

This thesis investigates the application of machine learning and deep learning methods into forecasting cryptocurrencies volatility. Using Bitcoin and Ethereum datasets, the performance of the following models was analyzed: Random Forest, XGBoost, LSTM, and GRU. The results show that deep learning models, specifically LSTM, outperforms machine learning models on every metric. These findings suggest that RNN models are better equipped to handle temporal dependencies in time series data that have sequential nature.
Keywords: Machine learning, Deep learning, cryptocurrency, volatility, Bitcoin, Ethereum, Random Forest, XGBoost, LSTM, GRU.

ملخص

تتناول هذه الأطروحة تطبيق طرق تعلم الآلة والتعلم العميق في التنبؤ بتقلبات العملات المعماة. وباستخدام مجموعة بيانات خاصة بكل من بيتكوين وإيثيريوم، تم تحليل أداء النماذج التالية: الغابة العشوائية Random Forest، وXGBoost، وLSTM، وGRU. وقد أظهرت النتائج أن نماذج التعلم العميق، وتحديدا LSTM، تتفوق على نماذج تعلم الآلة حسب كل المقاييس. وتشير هذه النتائج إلى امتلاك نماذج الشبكات العصبية المتكررة لخصائص تمكنها من التعامل بشكل أفضل مع التبعيات الزمنية في البيانات المتسلسلة زمنيا ذات الطبيعة التتابعية.
الكلمات المفتاحية: تعلم الآلة، التعلم العميق، عملة معماة، تقلب، بيتكوين، إيثيريوم، الغابة العشوائية، XGBoost، LSTM، GRU.

Résumé

Cette Thèse explore l'application des méthodes d'apprentissage automatique et d'apprentissage profond à la prévision de la volatilité des cryptomonnaies. À l'aide de datasets sur le Bitcoin et l'Ethereum, la performance des modèles suivants a été analysée : Random Forest, XGBoost, LSTM et GRU. Les résultats montrent que les modèles d'apprentissage profond, en particulier le LSTM, surpassent les modèles d'apprentissage automatique selon toutes les métriques. Ces résultats suggèrent que les réseaux de neurones récurrents (RNN) sont mieux adaptés pour gérer les dépendances temporelles dans les données de série temporelle de nature séquentielle.
Mots-clés : Apprentissage automatique, Apprentissage profond, Cryptomonnaie, Volatilité, Bitcoin, Ethereum, Random Forest, XGBoost, LSTM, GRU.