MODELING VOLATILITY DYNAMICS IN FINANCIAL TIME SERIES: AN ANALYSIS OF ECONOMETRIC MODELS AND MACHINE LEARNING METHODS

by

Hakan Erce

B.S., Mathematics, Boğaziçi University, 2020

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Computational Science and Engineering

Boğaziçi University

2023

ABSTRACT

MODELING VOLATILITY DYNAMICS IN FINANCIAL TIME SERIES: AN ANALYSIS OF ECONOMETRIC MODELS AND MACHINE LEARNING METHODS

In this thesis we present an analysis of different volatility dynamics in financial and cryptocurrency markets. We focus on the Borsa Istanbul stock exchange index BIST 100 (XU100), NASDAQ index, and Bitcoin/USD exchange rate (BTCUSD). Our aim is to understand the risk profiles of the data, forecast future risk using historical data, and statistically analyze structural characteristics of the volatility in the markets. In our analysis we used logarithmic returns of each financial asset we considered.

Our first step of the investigation was to determine the best fitting ARIMA models, and analyze the parameters of these models using tools such as ADF-test, ACF- and PACF-plots, QQ-plots, and Kolmogorov-Smirnov test. Our working hypothesis is that if a model successfully explains the behaviour of an asset then the residual signal must be close to white noise. However, we found significant autocorrelations in the residuals which indicates that the ARIMA models we used did not fully capture all underlying patterns.

In the next step, we used GARCH models to analyze the dynamics of the volatility inherent in each asset. While the risk profiles of the datasets varied, the structure coefficients of the models were consistent across all datasets. We finish the thesis by outlining how one might expand our study using endogeneous and exegeneous data, and different machine learning methods.

ÖZET

FINANSAL ZAMAN SERILERINDE OYNAKLIK DINAMIKLERININ MODELLENMESI: EKONOMETRIK MODELLERIN VE MAKINE ÖĞRENIMI YÖNTEMLERININ ANALIZI

Bu tezde, finansal ve kripto para piyasalarındaki farklı volatilite dinamiklerini analiz ediyoruz. Borsa Istanbul endeksi BIST 100 (XU100), NASDAQ endeksi ve Bitcoin/USD döviz kuru (BTCUSD) üzerine odaklanıyoruz. Amacımız, verilerin risk profillerini anlamak, geçmiş verileri kullanarak gelecekteki riskleri tahmin etmek ve piyasalardaki volatilite yapısının istatistiksel özelliklerini analiz etmektir. Analizimizde, ele aldığımız her finansal varlığın logaritmik getirilerini kullandık.

Araştırmamızın ilk adımı, en iyi uyumlu ARIMA modellerini belirlemek ve bu modellerin parametrelerini ACF ve PACF grafikleri, QQ grafikleri ve Kolmogorov-Smirnov testi gibi araçlarla analiz etmek oldu. Çalışma hipotezimiz, bir modelin bir varlığın davranışını başarılı bir şekilde açıklarsa, kalıntı sinyalinin beyaz gürültüye yakın olması gerektiğidir. Ancak, kalıntılarda önemli otokorelasyonlar bulduk, bu da kullandığımız ARIMA modellerinin tüm temel desenleri tam olarak yakalayamadığını göstermektedir.

Sonraki adımda, her varlıkta içsel olarak bulunan volatilite dinamiklerini analiz etmek için GARCH modelleri kullandık. Veri kümesinin risk profilleri farklılık gösterse de, modellerin yapı katsayıları tüm veri kümelerinde tutarlıydı. Tezi, endojen ve ekzojen verileri ve farklı makine öğrenimi yöntemlerini kullanarak çalışmamızı nasıl genişletebileceğimizi belirterek tamamlıyoruz.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

c_0	Constant term in the MA model equations
i	Index representing the lag order in the model equations
j	Index representing the lag order in the GARCH model equations
t	Referring to a specific time index
y_t	White noise series used in the model equations
α_0	Constant term in the model equations
α_1	Coefficient associated with lagged values in the ARMA and GARCH models
$\beta_1, \beta_2, \cdots, \beta_q$	Coefficients associated with lagged values in the MA and GARCH models
ϵ_t	Sequence of independently and identically distributed (iid) random variables
σ_t	Conditional standard deviation (volatility) at time t

LIST OF ACRONYMS/ABBREVIATIONS

AR	Autoregressive Models
ARMA	Autoregressive and Moving Average Models
ARCH	Autoregressive Conditionally Heteroskedasticity Models
BTCUSD	Bitcoin/USD exchange rate
MA	Moving Average Models
XU100	Borsa Istanbul stock exchange index

1. INTRODUCTION

Understanding and predicting the behavior of various economic and financial phenomena relies heavily on time series analysis. Although investors, financial institutions, and policymakers assign significant importance to the prediction and estimation of volatility and risk in financial data over time, accurate forecasting remains challenging due to the intricate temporal associations and volatility patterns observed in financial markets. Our primary hypothesis in this thesis is that the dynamics and subsequent risk assessment of markets, and issues of portfolio management may all be understood better through a thorough and rigorous statistical analysis of *the inherent volatility* in financial time series data.

Volatility is defined as the degree of variation and uncertainty in asset prices or returns. Researchers and practitioners can evaluate the possible risks associated with financial assets and decide on investment strategies, hedging techniques, and risk management by comprehending and modeling volatility. In order to understand and describe the dynamics and volatility of financial data, a number of time series models have been evolved over the years. The three commonly used models for time series are Autoregressive Integrated Moving Average (ARIMA), Autoregressive Conditional Heteroscedasticity (ARCH), and Generalized Autoregressive Conditional Heteroscedasticity (GARCH). While ARIMA models aim to structurally investigate the time series data, the latter two classes of models aim to investigate the inherent volatility within such time series data.

According to Box and Jenkins [1], ARIMA models are among the most well-liked and often used methods for predicting time series. ARIMA combines several elements, including moving average (MA), autoregressive (AR), and differencing (I), to capture the temporal dependencies and stationary properties of a time series. While the MA component depicts the dependence between the current observation and the lagged error terms, the AR component models the linear relationship between the current

observation and its lagged values. By removing trends and seasonality, the differencing component stabilizes the series.

Financial asset returns frequently show high heteroscedasticity which is a measure of volatility of the returns that vary across time. Engle observed that in financial time series there is also a tendency for higher volatility to be followed by higher volatility while lower volatility typically follows lower volatility, which is referred to as volatility clustering. In [2] he introduced ARCH models to capture this volatility clustering phenomena. ARCH models successfully capture the time-varying volatility patterns by allowing the conditional variance to change over time. This conditional variance equation in ARCH models, which is a function of previous squared residuals, expands the ARIMA framework. In [3] Bollerslev introduced GARCH models to better capture the financial volatility through the inclusion of a lagged conditional variance factor to the ARCH framework. By doing so, GARCH models flexibly capture any short-term and long-term dependencies in volatility. They can also better depict the clustering phenomena and capture the persistence of volatility shocks by taking into account the lagged conditional variance. GARCH models are frequently used in a variety of financial applications, such as option pricing, risk management, and asset pricing. In order to better capture of volatility clustering and the modeling of risk in financial markets, Glosten et al. developed the GARCH-M model to represents volatility dynamics more thoroughly by incorporating both the moving average conditional heteroscedasticity (GARCH) and the autoregressive conditional heteroscedasticity (ARCH) components [4].

In this thesis, in addition to ARIMA and (G)ARCH models, we also investigated the use of widely used machine learning models such as XGBoost and Decision Tree Regression to model volatility since XGBoost and decision tree regression have proven to be effective techniques in modeling financial time series and capturing complex patterns [5, 6]. XGBoost, which is an ensemble learning method based on gradient boosting, has gained popularity for its ability to handle large-scale datasets and deliver high predictive accuracy, and has been successfully used in various financial domains,

including stock market prediction and risk modeling [7]. The decision tree regression models, on the other hand, are non-parametric models that recursively splits the data depending on feature values, can be used for the discovery of complicated decision boundaries. These models have been used in financial applications to model asset prices, estimate option prices, and forecast market volatility. Regression trees first described by Breiman et al. [6] can also handle high-dimensional data and capture nonlinear correlations.

The main objectives of this thesis are two fold: first, to develop forecasting models for the Borsa Istanbul 100 Index (BIST 100), the NASDAQ market index, and Bitcoin price changes; second, to develop (G)ARCH models to analyze dynamic of volatility of these assets and indices. However, rather than forecasting price changes, our main focus is on modeling volatility and risk. First, we employ classical time series models in Section 4.1 for all datasets. Then we use lagged rolling variance characteristics as input variables into the XGBoost and Decision Tree Regression models in order to capture the intrinsic temporal patterns and volatility dynamics in these datasets in Section 4.3. Our models seek to enhance their comprehension and prediction of market volatility by utilizing these lagged rolling variance features, which offer useful insights into historical price and return volatility, allowing for a more precise evaluation of the risks related to financial assets.

2. DATASET

2.1. Financial Time Series Dataset

2.1.1. BIST 100

BIST100 (XU100) is the benchmark stock market index of Istanbul Stock Exchange (Borsa Istanbul). It consists the 100 stocks with the highest market and trading volume of the exchange that represent a diverse sectors of the Turkish economy. BIST100 is regarded as a trustworthy indication of the performance of the Turkish stock market, and the economy as a whole. Figure 2.1 displays a sample of the closing price of the BIST 100 from September 18, 2014 to January 1, 2020. The composition of the BIST 100 index is determined by the Istanbul Stock Exchange board. The exchange has two other indices designated as BIST30 and BIST50 along with BIST100. The stocks in the BIST 100 index are chosen from the A and B group shares. The most active stocks on the Borsa Istanbul exchange are in Group A. Group B is used to evaluate shares with a modest quantity of actual circulation. BIST100 is redetermined every 3 months by the exchange.

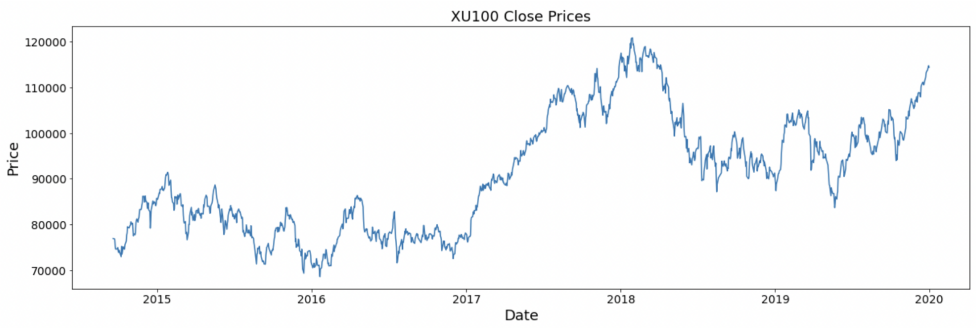

Figure 2.1. BIST100 closing price from September 18, 2014 to January 1, 2020.

2.1.2. NASDAQ

NASDAQ is an American stock exchange that specializes on technology-related listings. It was formed in 1971 and is now the second-largest stock exchange of the world by market capitalisation, trailing only the New York Stock Exchange (NYSE). NASDAQ Composite Index tracks 3,300 equities traded on the exchange. The term "NASDAQ" is commonly utilized to denote the NASDAQ Composite Index. The NASDAQ Composite is frequently regarded as a gauge of the success of the technology sector. The NASDAQ 100 Index, on the other hand, tracks the 100 largest companies by market capitalization traded on the NASDAQ. Although the majority of these companies are typically associated with technology, they also contain a number of businesses from other industry sectors. Depending on their market value, companies might be added to or deleted from this index each year.

In this thesis we focus on NASDAQ 100 Index, and we will use *NASDAQ* to denote NASDAQ 100. Figure 2.2 displays a sample of the closing prices plot for NASDAQ index.

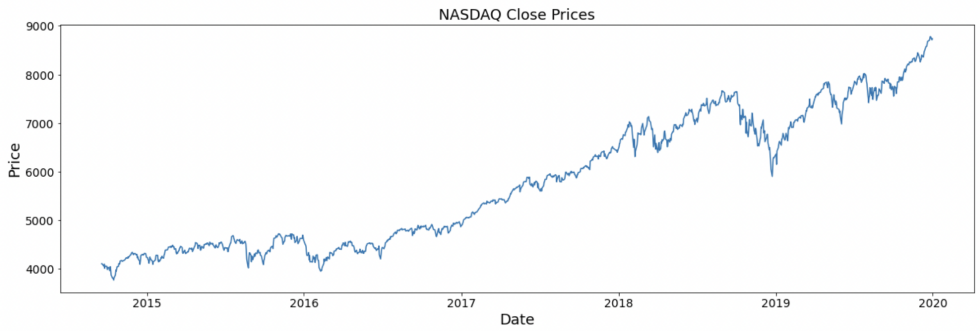

Figure 2.2. NASDAQ close price from September 18, 2014 to January 1, 2020.

2.1.3. Bitcoin

Bitcoin (BTCUSD) is a digital currency that was launched in 2009 by an unknown person using the alias Satoshi Nakamoto. It was built on decentralized blockchain

technology, which enables secure, transparent, and rapid transactions without the use of any intermediaries such as banks, or centralized ledgers. Bitcoin is frequently referred to as a sort of digital gold since it is supposed to be scarce, with a supply limit of 21 million coins. Figure 2.3 displays a sample of the closing prices for BTCUSD.

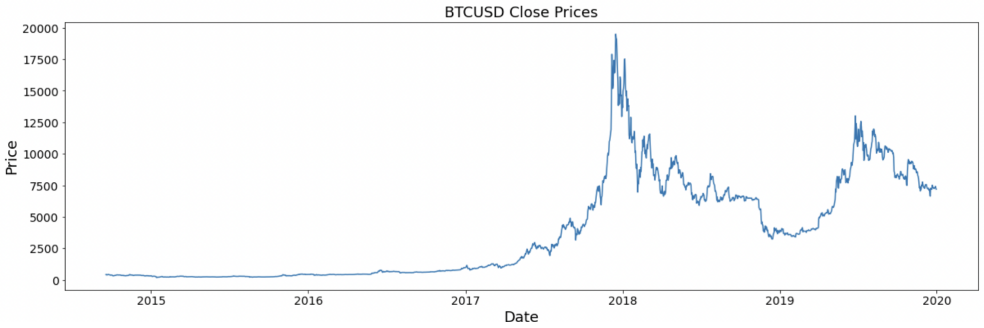

Figure 2.3. BTCUSD close price from September 18, 2014 to January 1, 2020.

2.1.4. Our Combined Dataset

Our dataset for this study consists of daily stock price of BIST100, NASDAQ and Bitcoin, and was obtained from the Yahoo Finance using their API from September 18, 2014 to January 1, 2020. Each row in the dataset demonstrate a single trading day and consists of four columns. A sample of the data for BTCUSD is shown in Figure 2.4. We explain these columns below.

Close price: The closing price of a securities is the final price at which it is traded on a given day. It is the final price reached by buyers and sellers before the market shuts. Traders and investors value this price since it indicates the overall sentiment of the market for the day.

High price: The highest price at which an asset traded during a specific time period, such as a day, week, or month. It indicates the highest level of security need during that time period.

Low price: The lowest price at which a security traded during a certain period is referred to as the low price. It represents the lowest level of demand of the security throughout that time period.

Open price: The first price at which a security trades at the start of a trading session, such as a day. It is usually seen as significant since it might set the tone for the rest of the day's trade.

Date	Open	High	Low	Close
2014-09-18	456.859985	456.859985	413.104004	424.440002
2014-09-19	424.102997	427.834991	384.532013	394.795990
2014-09-20	394.673004	423.295990	389.882996	408.903992
2014-09-21	408.084991	412.425995	393.181000	398.821014
2014-09-22	399.100006	406.915985	397.130005	402.152008
2014-09-23	402.092010	441.557007	396.196991	435.790985
2014-09-24	435.751007	436.112000	421.131989	423.204987
2014-09-25	423.156006	423.519989	409.467987	411.574005
2014-09-26	411.428986	414.937988	400.009003	404.424988
2014-09-27	403.556000	406.622986	397.372009	399.519989

Figure 2.4. A sample of BTCUSD data.

2.2. Return Series

In this thesis, we are going to examine the time series data for the BIST100 and NASDAQ indices, and the Bitcoin to USD exchange rates. It is crucial to emphasize that instead of using the nominal price data, we will be focusing on the return and logarithmic return of closing prices.

2.2.1. Simple Return

Return is a term used in finance to describe the percentage change in an value of an asset over a given time period. It is calculated by taking the final and initial values of asset, dividing them by each other, and then displaying the result as a percentage. Assume that V_t reflects the as at price of an asset at time index. The following definition applies to the appropriate one-period simple net return, denoted as

$$\mathbf{R}_t = \frac{V_t - V_{t-1}}{V_{t-1}}. \tag{2.1}$$

2.2.2. Logarithmic Return

The logarithmic net return for a single period, stated as R_t, can be defined as

$$\mathbf{R}_t = \ln(V_t) - \ln(V_{t-1}), \tag{2.2}$$

where V_t represents the price of an asset at time index t. One can also think of the logarithmic return as the first (discrete) derivative of the logarithm of the price series.

2.3. Variance Series

In order to apply classical machine learning algorithms such as XGBoost and decision tree regression, we needed to generate *rolling variance* series for the financial assests that we are going to investigate in this thesis. In this section, we are going to explain how the rolling variance datasets were prepared.

Recall that the Yahoo Finance data we retrieved for the BIST100, NASDAQ, and BTCUSD tickers from September 18, 2014 to January 1, 2020 contain closing and opening prices. Using the opening and closing prices, as we explained in the previous section, we calculated logarithmic return series for these assets by calculating the first discrete derivatives of the natural logarithms of the price series. In our next step, we calculate the rolling variances to further extract useful data from the dataset.

Given a time series $x_1, \ldots, x_n$ the derived series of variation $v_i := \mathrm{Var}(x_i, \ldots, x_{i+k})$ over a rolling window $x_i, \ldots, x_{i+k}$ for $i = 1, \ldots, n-k+1$ is referred to as the rolling variance. The rolling variance features are added to the dataset as new columns. The rolling variances of the logarithmic returns at various time lags for $k = 1, 2, 3, 4, 5$ are represented by the lagged features $x_1, x_2, \cdots, x_5$. A sample of the datasets can be found in Figure 2.5 and Figure 2.6.

Date	lreturn_rol_var	x1	x2	x3	x4	x5
2014-10-08	0.000221	0.000160	0.000132	0.000137	0.000105	0.000099
2014-10-09	0.000209	0.000221	0.000160	0.000132	0.000137	0.000105
2014-10-10	0.000196	0.000209	0.000221	0.000160	0.000132	0.000137
2014-10-13	0.000293	0.000196	0.000209	0.000221	0.000160	0.000132
2014-10-14	0.000287	0.000293	0.000196	0.000209	0.000221	0.000160
...	...	...	...	...	...	...
2019-12-24	0.000036	0.000031	0.000034	0.000034	0.000032	0.000034
2019-12-26	0.000030	0.000036	0.000031	0.000034	0.000034	0.000032
2019-12-27	0.000029	0.000030	0.000036	0.000031	0.000034	0.000034
2019-12-30	0.000046	0.000029	0.000030	0.000036	0.000031	0.000034
2019-12-31	0.000043	0.000046	0.000029	0.000030	0.000036	0.000031

Figure 2.5. NASDAQ dataset with rolling variance shift.

Date	lreturn_rol_var	x1	x2	x3	x4	x5	x6	x7	x8	x9	x10
2014-10-24	0.000228	0.000232	0.000233	0.000233	0.000216	0.000209	0.000202	0.000227	0.000226	0.000225	0.000180
2014-10-27	0.000232	0.000228	0.000232	0.000233	0.000233	0.000216	0.000209	0.000202	0.000227	0.000226	0.000225
2014-10-28	0.000184	0.000232	0.000228	0.000232	0.000233	0.000233	0.000216	0.000209	0.000202	0.000227	0.000226
2014-10-30	0.000135	0.000184	0.000232	0.000228	0.000232	0.000233	0.000233	0.000216	0.000209	0.000202	0.000227
2014-10-31	0.000073	0.000135	0.000184	0.000232	0.000228	0.000232	0.000233	0.000233	0.000216	0.000209	0.000202
...	...	...	...	...	...	...	...	...	...	...	...
2019-12-24	0.000047	0.000047	0.000059	0.000064	0.000065	0.000067	0.000068	0.000068	0.000083	0.000058	0.000057
2019-12-26	0.000045	0.000047	0.000047	0.000059	0.000064	0.000065	0.000067	0.000068	0.000068	0.000083	0.000058
2019-12-27	0.000045	0.000045	0.000047	0.000047	0.000059	0.000064	0.000065	0.000067	0.000068	0.000068	0.000083
2019-12-30	0.000047	0.000045	0.000045	0.000047	0.000047	0.000059	0.000064	0.000065	0.000067	0.000068	0.000068
2019-12-31	0.000041	0.000047	0.000045	0.000045	0.000047	0.000047	0.000059	0.000064	0.000065	0.000067	0.000068

Figure 2.6. BIST100 dataset with rolling variance shift.

The sample dataset shown in Figure 2.5 comprises of six columns. For the sample, the window size for computing the variance was set at $k = 10$. Additionally, the fact that we have columns `x1`, through `x5` indicates that we took lags x_{t-1} through x_{t-5} for the rolling variance data. On the other hand, the dataset shown in Figure 2.6 has 10 columns. In this case, we set the window size to $k = 14$. In addition, the columns `x1` through `x10` indicate that the number of lagged rolling variance features is 10.

2.4. Synthetic Time Series Dataset

The investigation and comparison of model performance on our datasets requires the use of synthetic data. The basic idea is that one may evaluate the capacity of models to identify and explain risk factors in financial data by making comparisons with purely random time series. This comparison might shed light on the advantages and disadvantages of a particular model as well as their capacity to generalize to new data. In particular, in this thesis we investigate and assess the effectiveness of XGBoost and Decision Tree Regression models in describing the inherent risk in financial time series using synthetic datasets.

First, let us explain the main parameters for the samples we have taken. Brownian motion is often used to create synthetic datasets. The volatility or diffusion rate of the Brownian motion is determined by the Wiener process parameter denoted by δ which we varied for the samples we have taken. This parameter controls the variation of the sampled time series. We then selected the complete time span (abbreviated as T) which controls the length of the time series. The granularity of the observations is determined by dividing the time series into a predetermined number of steps, denoted by N. The ratio $\frac{T}{N}$ is then used to determine the time step size, or dt, which denotes the interval between successive observations.

After setting the parameters, we created our synthetic time series. To simulate the random variations in the time series, we iteratively calculated the value x_{t+1} at each time point by adding to x_t a random increment starting from an initial value.

The random increment is produced by sampling a random number from the normal distribution with a mean of 0 and a standard deviation determined as δ multiplied by the square of dt. x_{t+1} can be defined as

$$x_{t+1} = x_t + \epsilon_t \quad \text{where} \quad \epsilon_t \sim N(0, \delta dt^2). \tag{2.3}$$

A sample of our synthetic time series for the same T, δ and N may be found in Figure 2.7.

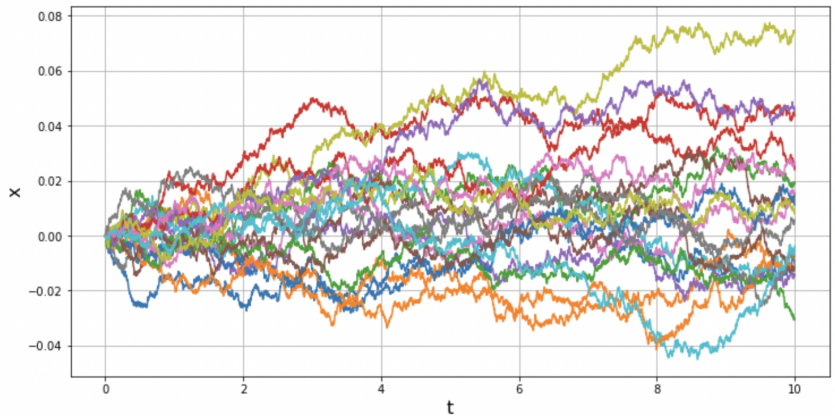

Figure 2.7. Synthetic time series.

3. METHODOLOGY

3.1. Machine Learning Models

3.1.1. Decision Tree Regression

When using continuous target variables in regression situations, decision tree regression is an effective machine learning technique. It is a non-parametric technique that develops a decision tree-like model based on the attributes of dataset. Each leaf node of the tree represents a predicted continuous value, whereas each internal node represents a test on a particular attribute.

By dividing the dataset into smaller subsets based on feature values, decision trees are built iteratively. Finding the splits that effectively divide the values of the target variable is the objective. The CART (Classification and Regression Trees) algorithm was developed for building decision trees and was first proposed by Breiman et al. in 1984 [6]. The method determines the optimum characteristic and value for splitting by evaluating several splitting criteria, such as the Gini impurity or information gain.

The CART simply asks yes-or-no questions. "Is credit amount greater than 200 USD?" or "Is sex female?" are two potential questions. The ideal question that divides the data into two halves with the maximum degree of similarity is determined by carefully examining each variable and its corresponding values using the CART algorithm. This is known as the best split. Each of the generated data fragments is then subjected to the same procedure once more. Furthermore, CART handles both categorical and numerical variables with ease.

Let the parent, left child, and right child nodes be identified as n_p, n_l, and n_r, respectively. A variable matrix A with Q variables (a_j) and R observations makes up the learning sample. The Y class vector is made up of R observations that represent C different classes.

A splitting rule is used to build the classification tree in an effort to segment the learning sample into more manageable subsets. The best potential uniformity in the divide is what is desired. The parent node, left child node, and right child node are referred to in this context as N_p, N_l, and N_r, respectively. In addition, a_j stands for the j-th variable, and a_j^S stands for the ideal splitting value.

The degree of homogeneity inside the child nodes is determined by the impurity function, written as $f(t)$.

We want to maximize the change in the impurity function, given as $\Delta f(t)$, in order to maximize the homogeneity of both the left and right child nodes. No matter the possible splits, $a_j < a_j^S$, where $j = 1, \cdots, Q$, the impurity of parent node,n_p, stays constant. $\Delta f(t)$ is expressed as

$$\Delta f(t) = f(n_p) - E[f(n_c)]. \tag{3.1}$$

The left and right child nodes of the parent node are referred to here as t_c. Assuming P_l and P_r stand for the probabilities of the left and right nodes respectively, one can write $\Delta f(t)$ as change in the impurity

$$\Delta f(t) = f(n_p) - P_l f(n_l) - P_r f(n_r). \tag{3.2}$$

As a result, for each node, the CART algorithm resolves the maximization problem

$$\underset{x_j \leq x_R, j=1,\ldots,M}{\arg\max} [f(n_p) - P_l f(n_l) - P_r f(n_r)]. \tag{3.3}$$

In order to find the splitting question that maximizes the change in the impurity measure, $\Delta f(t)$, with the condition $a_j \leq a_j^S$, CART evaluates all possible values for each variable in the matrix A, according to Equation (3.3).

Although there are various impurity functions in theory, in order to comprehend decision tree regression, we will just focus on Gini Impurity.

3.1.1.1. Gini Impurity. In decision tree algorithms, the Gini index, also known as the Gini splitting rule, is frequently used. The impurity function $f(t)$, defined as

$$f(t) = \sum_{c_1 \neq c_2} p(c_1|t)p(c_2|t). \tag{3.4}$$

Here, the class indices c_1 and c_2 range from 1 to C, and $p(c_1|n)$ denotes the conditional probability of being a member of class c_1 under the assumption that we are in node n.

Equation (3.5) demonstrates the change in impurity measure resulting from applying the Gini impurity function from the Equation (3.4) to the maximization problem in Equation (3.3) where

$$\Delta f(t) = -\sum_{c=1}^{C} p^2(c|n_p) + P_l \sum_{c=1}^{C} p^2(c|n_l) + P_r \sum_{c=1}^{C} p^2(c|n_r). \tag{3.5}$$

3.1.1.2. Regression Trees. There is no classification of classes in regression trees. Instead, the response vector Y shows the response values for each observation in the variable matrix A. The application of Gini impurity in the classification splitting processes covered in Equation (3.4) is not appropriate since regression trees lack preset classes.

The splitting method used in regression trees is based on minimizing the squared residuals. As shown in the objective is to reduce the estimated total of variances for the resulting nodes such as

$$\underset{a_j \leq a_j^S, j=1,\ldots,Q}{\arg\min} \; [P_l \mathrm{Var}(Y_l) + P_r \mathrm{Var}(Y_r)] \tag{3.6}$$

where the response vectors for left and right child nodes are $\mathrm{Var}(Y_l)$, $\mathrm{Var}(Y_r)$ respectively. Here, $a_j < a_j^S$, $j = 1, \cdots, Q$ is the best splitting that satisfies Equation (3.6).

3.1.2. XGBoost

A popular and highly efficient machine learning technique is tree boosting, and a most frequently used scalable and regularized library implementation for tree boosting is XGBoost [5]. Model generalization capabilities of the library are enhanced by the use of sophisticated L1 and L2 regularization. In most applications, XGBoost outperforms Gradient Boosting in terms of performance, but the success of XGBoost is primarily due to its scalability. On a single machine, the system operates more than an order of magnitude faster than other tree boosting implementations, and in distributed or memory-constrained environments, it scalable to billions of samples. Its training may be parallelized across clusters and is extremely quick.

Through a combination of necessary system improvements and algorithmic innovations, XGBoost achieves scalability. One such innovation is its unique method of tree learning, which is made to deal with sparse data. To manage instance weights during approximate tree learning, XGBoost also uses a weighted quantile sketch technique that is theoretically supported. Through the use of parallel and distributed computing, which vastly speeds up model exploration, the learning process is further accelerated. We shall explore the specifics of tree boosting in the section that follows.

3.1.2.1. Regularized Learning Objective. Additive functions $f_1, \ldots, f_K$ are used by a tree ensemble model to forecast the results for a given data set $D = \{(x_i, y_i)\}_{i=1}^{n}$ where $x_i \in \mathbb{R}^m$ and $y_i \in \mathbb{R}$ with n examples and m features as

$$\hat{y_i} = \varphi(x_i) = \sum_{k=1}^{K} f_k(x_i), \quad f_k \in F \tag{3.7}$$

where $F = \{f(x) = w_q(x)\},\ q : \mathbb{R}^m \to T,\ w \in \mathbb{R}^T$ is the space of regression trees (also known as CART). In this case, q stands for the structure of each tree that associates a leaf index with an example, and T denotes how many leaves there are on the tree. An independent tree structure q and leaf weights w is chosen for each f_k. We use w_i to denote the score on the i-th leaf because, unlike decision trees, regression trees have continuous scores on each leaf. For a particular example, we will classify the example

into the leaves using the decision rules in the trees (provided by q), and then total the scores in the associated leaves (provided by w) to determine the final forecast. We minimize the subsequent regularized objective in order to learn the set of functions employed in the model in Equation (3.8) where

$$L(\varphi) = \sum_i \ell(\hat{y}_i, y_i) + \sum_k \Omega(f_k) \tag{3.8}$$

and $\Omega(f) = \gamma T + \frac{1}{2}\lambda\|w\|^2$. The difference between the forecast $\hat{y}_i$ and the goal y_i is measured by the differentiable convex loss function ℓ in this instance. The second term Ω (i.e., the functions of the regression tree) penalizes the complexity of models. In order to prevent over-fitting, the additional regularization term helps to smooth the final learned weights.

3.1.2.2. Gradient Tree Boosting. Equation (3.8) can not be optimized using traditional optimization methods in Euclidean space. The model is instead trained in an additive way. Formally, to minimize the next objective, we need to add f_t, where $\hat{y}_i^t$ represents the prediction of the i-th instance at the t-th iteration. The objective is

$$L^{(t)} = \sum_{i=1}^{n} \ell(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)) + \Omega(f_t). \tag{3.9}$$

This means that we greedily add the f_t that Equation (3.8) indicates that best improves our model. The aim can be quickly optimized in the general setting using second-order approximation

$$L^{(t)} \approx \sum_{i=1}^{n} \left[\ell(y_i, \hat{y}(t-1)) + g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)\right] + \Omega(f_t) \tag{3.10}$$

where g_i and h_i are first and second order gradient statistics respectively. At step t, we can eliminate the constant terms to achieve the next, more straightforward goal, which can be expressed as

$$\hat{L}^{(t)} = \sum_{i=1}^{n} \left[g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)\right] + \Omega(f_t). \tag{3.11}$$

Set $I_j = \{i \,|\, q(x_i) = j\}$ as the leaf j instance set. Equation (3.11) may be rewritten by extending Ω as

$$\hat{L}^{(t)} = \sum_{i=1}^{n} \left[g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)\right] + \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2 \tag{3.12}$$

$$= \sum_{j=1}^{T} \left[\left(\sum_{i \in I_j} g_i \right) w_j + \frac{1}{2} \left(\sum_{i \in I_j} h_i + \lambda \right) w_j^2 \right] + \gamma T. \tag{3.13}$$

We can calculate the ideal leaf weight w_j^* for a given structure $q(x)$ using

$$w_j^* = -\frac{\sum_{i \in I_j} g_i}{\sum_{i \in I_j} h_i + \lambda} \tag{3.14}$$

and determine the related optimum value by

$$\hat{L}^{(t)}(q) = -\frac{1}{2} \sum_{j=1}^{T} \frac{\left(\sum_{i \in I_j} g_i \right)^2}{\sum_{i \in I_j} h_i + \lambda} + \gamma T. \tag{3.15}$$

The quality of a tree structure q can be evaluated using Equation (3.15) as a scoring function. The difference between this score and the impurity score for decision trees is that this formulation can be computed for a larger variety of objective functions.

3.2. Time Series Analysis

A time series is a collection of data points that were collected over time at progressively equal intervals. The sequence of the observations is crucial because it depicts the development of a variable or phenomenon across time. We can predict future behavior and patterns within a time series, spot anomalies, and gain understanding of the underlying dynamics and causes driving the data by looking at historical behavior and patterns within the time series. To extract important information from the data, time series analysis offers a variety of potent tools and approaches.

In this section we are going to pay attention to time series analysis, a statistical method for examining data points gathered over time. Numerous disciplines, including finance, economics, meteorology, and engineering, frequently use time series data. Making forecasts and wise decisions is made possible by the analysis of time series data, which enables us to find patterns, trends, and relationships within the data.

In time series analysis, stationarity is a key notion. When statistical characteristics, such as the mean and covariance of a time series remain constant across time,

it is said to be stationary. However, in reality, weak stationarity where the mean and covariance of time series do not change with time, so it is frequently more realistic. Small variations around a stable mean and a constant covariance structure across the series are permitted by weak stationarity. Weak stationarity allows us to make certain assumptions and apply a variety of statistical techniques to efficiently examine and model time series data.

Next, we cover the correlation and autocorrelation functions. A time series and its lag values are measured by autocorrelation, whereas correlation measures the linear relationship between two random variables.

In this section we also introduce the ideas of linear time series and white noise. We will refer a series of independently distributed random variables with no correlation as *white noise*. Another important concept we are going to introduce is the weighted linear combination of lagged values of a white noise series which is referred as a *linear time series*. Wold's Theorem shows that every stationary time series is linear [8].

3.2.1. Stationarity

Let a_i be a time series. We say that a_i is *stationary* if the joint distribution of any window $(a_{i_1},...,a_{i_k})$ and any of t-iterates $(a_{i_1+t},...,a_{i_k+t})$ are identical for every t and arbitrary window size k. The joint distribution of $(a_{i_1},...,a_{i_k})$ must be time independent to be considered stationary. This is a strong condition that is challenging to test statistically. Thus one can often consider a weaker notion of stationarity.

Consider the case where we have actually observed N data points with a_n, $n = 1, \ldots, N$. With weak stationarity, the time plot would demonstrate that the N values oscillate constantly around a fixed level. Asset return series are frequently assumed to be weakly stationary in financial literature.

For example, given a time series a_n, the lag-autocovariance of a_n is the covariance defined as $\gamma_m = \text{Cov}(a_n, a_{n-m})$. It has two crucial characteristics such as (a) $\gamma_0 = \text{Var}(a_n)$ and (b)$\gamma_{-m} = \gamma_m$. Because $\text{Cov}(a_n, a_{n-(-m)}) = \text{Cov}(a_{n-(-m)}, a_n) = \text{Cov}(a_{n+m}, a_n) = \text{Cov}(a_{n_1}, a_{n_1-m})$ and $n_1 = n + m$, the second property stays true.

3.2.1.1. Correlation. Let A and B be two random variables. The correlation between A and B is defined as

$$q_{A,B} = \frac{\text{Cov}(A, B)}{\sqrt{\text{Var}(A)\text{Var}(B)}} = \frac{E[(A - \mu_A)(B - \mu_B)]}{\sqrt{E(A - \mu_A)^2 E(B - \mu_B)^2}} \tag{3.16}$$

where the variances are presummably present and μ_A and μ_B represent the means of A and B, respectively. This coefficient indicates how strongly random variables A and B correlate with one another. One can show that $q_{A,B} = q_{B,A}$ and that $-1 \leq q_{A,B} \leq 1$. A value close to 1 indicates that if A increases or decreases, then B also tends to have same behaviour. On the other hand, a value close to -1 indicates that A and B move in the opposite directions.

When the sample $\{(a_n, b_n)\}_{n=1}^N$ is provided, the correlation can be reliably approximated by its sample counterpart

$$\hat{q}_{A,B} = \frac{\sum_{n=1}^N (A_n - \bar{A})(B_n - \bar{B})}{\sqrt{\sum_{n=1}^N (A_n - \bar{A})^2 \sum_{n=1}^N (B_n - \bar{B})^2}} \tag{3.17}$$

where $\bar{A} = \sum_{n=1}^N A_n/N$ and $\bar{B} = \sum_{n=1}^N B_n/N$ represent, respectively, the sample means of A and B.

3.2.1.2. Autocorrelation Function (ACF). Let a_n be weakly stationary series. The concept of correlation is expanded to include autocorrelation when it comes to the linear relationship between a_n and its historical values a_{n-i}. The lag-i-autocorrelation of a_n, also known as the correlation coefficient between a_n and a_{n-i}., is represented by the symbol q_i and, according to the weak stationarity assumption, is only a function of i. We define

$$q_i = \frac{\text{Cov}(a_n, a_{n-i})}{\sqrt{\text{Var}(a_n)\text{Var}(a_{n-i})}} = \frac{\text{Cov}(a_n, a_{n-i})}{\text{Var}(a_n)} = \frac{\gamma_i}{\gamma_0} \tag{3.18}$$

since we assume a_n is weakly stationary, $Var(a_n) = Var(a_{n-i})$. Also we get, $q_0 = 1$, $q_i = q_{-i}$ and $-1 \leq q_i \leq 1$ from the definition of correlation. Additionally, $q_i = 0$ for all $i > 0$, if and only if a weakly stationary series a_n is not serially correlated.

Let $\bar{A}$ be the sample mean for a given sample of returns $\{A_n\}_{n=1}^{N}$. Then, lag-1 sample autocorrelation of a_n is

$$\hat{q_1} = \frac{\sum_{n=2}^{N}(A_n - \bar{A})(A_{n-1} - \bar{A})}{\sum_{n=1}^{N}(A_n - \bar{A})^2}. \tag{3.19}$$

$\hat{q_1}$ is a reliable estimate of q_1 under some general circumstances.

The lag-i-sample autocorrelation of A_t is usually defined as

$$\hat{q_i} = \frac{\sum_{n=i+1}^{N}(A_n - \bar{A})(A_{n-i} - \bar{A})}{\sum_{n=1}^{N}(A_n - \bar{A})^2}, \tag{3.20}$$

where $0 \leq i < N - 1$.

3.2.2. White Noise, Linear Time Series and Unit Root

3.2.2.1. White Noise. If $\{a_t\}$ is a sequence of independent random variables with finite mean and variance that are all independently and identically distributed (iid), then a_t is referred to as white noise. The series is referred to as Gaussian white noise if a_t has a normal distribution with mean zero and variance σ^2. All of the ACF values for a white noise series are 0. In reality, we take the series to be a white noise series if all sample ACF values are close to zero.

3.2.2.2. Linear Time Series. A time series a_t is called linear, if it defined as

$$a_t = \mu + \sum_{i=0}^{\infty} w_i a_{t-i}, \tag{3.21}$$

where μ is the average of a_t, w_0 is equal to 1, and $\{a_t\}$ is a series of random variables that are independent and identically distributed and have a mean of zero.

3.2.2.3. Unit Root. The mathematical definition of a unit root can be expressed as follows: Consider the following equation, which represents an autoregressive model of

order p denoted by AR(p) in Equation (3.25):

$$a_t = \alpha_0 + \alpha_1 a_{t-1} + \ldots + \alpha_p a_{t-p} + y_t, \tag{3.22}$$

where α_0 is constant term, y_t is white noise series and $\alpha_0, \alpha_1, \cdots \alpha_p$ are coefficients of autoregressive terms.

We say that a *unit root* exists if there exist $i = 1, 2, \cdots, p$ such that $|\alpha_i| = 1$. This concept is commonly used to test for stationarity in a time series.

3.3. Time Series Models

3.3.1. AR Models

3.3.1.1. Simple autoregressive model AR(1). A simple AR(1) model is

$$a_t = \alpha_0 + \alpha_1 a_{t-1} + y_t \tag{3.23}$$

where y_t is white noise series with zero mean and variance σ_y^2. Model (3.22) is called AR(1). This model is the same as simple linear regression with dependent variable is a_t, while the explanatory variable is a_{t-1}. For AR(1) model, it is worth noting that

$$E(a_t|a_{t-1}) = \alpha_0 + \alpha_1 a_{t-1} \quad \text{Var}(a_t|a_{t-1}) = \text{Var}(y_t) = \sigma_y^2. \tag{3.24}$$

That is, the current return is centered around $\alpha_0 + \alpha_1 a_{t-1}$ with variability σ_y^2.

3.3.1.2. AR(p) models. A generalization for AR(p) model is

$$a_t = \alpha_0 + \alpha_1 a_{t-1} + \ldots + \alpha_p a_{t-p} + y_t \tag{3.25}$$

where y_t is white noise series with zero mean and variance σ_y^2. Model (3.25) is called autoregressive(AR) model with order p or simply AR(p).

It pays out to study fundamental features of AR models for efficient application. We discuss the traits of the AR(1) and AR(2) models in considerable detail and then show the results for the general AR(p) model.

3.3.1.3. Partial Autocorrelation Function (PACF). To identify the appropriate order p for an autoregressive model, an effective approach is to utilize the partial autocorrelation function of a time series. The PACF is a function that is based on the autocorrelation function. In order to provide a clear explanation of the PACF concept, let's consider the following examples of AR models:

$$\begin{aligned}
a_t &= \alpha_{0,1} + \alpha_{1,1}a_{t-1} + e_{1,t} \\
a_t &= \alpha_{0,2} + \alpha_{1,2}a_{t-1} + \alpha_{2,2}a_{t-2} + e_{2,t} \\
a_t &= \alpha_{0,3} + \alpha_{1,3}a_{t-1} + \alpha_{2,3}a_{t-2} + \alpha_{3,3}a_{t-3} + e_{3,t} \\
a_t &= \alpha_{0,4} + \alpha_{1,4}a_{t-1} + \alpha_{2,4}a_{t-2} + \alpha_{3,4}a_{t-3} + \alpha_{4,4}a_{t-4} + e_{4,t} \\
&\vdots,
\end{aligned}$$

where $\alpha_{0,j}$ is constant term, $\alpha_{i,j}$ is the coefficient of a_{t-i}, and $\epsilon_{j,t}$ is the error term of an AR(j) model in the Equation (3.25). The least squares method can be used to estimate these models, which take the form of multiple linear regressions. We called $\alpha_{i,i}$ as lag-i sample partial autocorrelation of a_t. For instance, $\alpha_{1,1}$ is lag-1 sample partial autocorrelation of a_t whereas $\alpha_{2,2}$ is lag-2 sample partial autocorrelation of a_t. For example, the lag-2 partial autocorrelation $\alpha_{2,2}$ illustrates the additional contribution of a_{t-2} to a_t above the AR(1) model $a_t = \alpha_0 + \alpha_1 a_{t-1} + e_{1,t}$ in Equation (3.23), as indicated by the definition. The lag-4 partial autocorrelation illustrates the addition of a_{t-4} to a_t over an AR(3) model, and so on.

3.3.1.4. Expectation of AR(1) Models. We start with assuming the series a_t is weakly stationary. So we have $E(a_t) = \mu$, $\text{Var}(a_t) = \theta_0$, $\text{Cov}(a_t, a_{t-j}) = \theta_j$, where μ and θ_0 are constants. Taking expectation for both sides of the equation (3.23) we have

$$E(a_t) = E(\alpha_0) + E(\alpha_1 a_{t-1}) + E(y_t) = \alpha_0 + \alpha_1 E(a_{t-1}) \tag{3.26}$$

since $E(y_t) = 0$, $E(\alpha_0) = \alpha_0$, $E(\alpha_1) = \alpha_1$ and $E(\alpha_1 a_{t-1}) = E(\alpha_1)E(a_{t-1})$. Under stationarity condition we have $E(a_t) = E(a_{t-1}) = \mu$. Then the Equation (3.26) becomes

$$\begin{aligned}
\mu &= \alpha_0 + \alpha_1 \mu \\
E(a_t) = \mu &= \frac{\alpha_0}{1 - \alpha_1}.
\end{aligned}$$

As we can see the mean of a_t does not exists when $\alpha_1 = 1$ and equal to 0 when $\alpha_0 = 0$. It also means that the mean of a_t depending on both α_0 and α_1.

3.3.1.5. Expectation of AR(2) Models. An AR(2) model can be defined as

$$a_t = \alpha_0 + \alpha_1 a_{t-1} + \alpha_2 a_{t-2} + y_t. \tag{3.27}$$

The identical method used in the AR(1) situation is used to obtain

$$E(a_t) = \mu = \frac{\alpha_0}{1 - \alpha_1 - \alpha_2}. \tag{3.28}$$

The mean of a_t does not exists when $\alpha_1 + \alpha_2 = 1$ and equal to 0 when again $\alpha_0 = 0$ as same as for model AR(1). It also means that the mean of a_t depends on all α_0, α_1, and α_2.

3.3.1.6. Expectation for AR(p) Models. The broad AR(p) model in Equation (3.25) can be easily used to generalize the results of the autoregressive models with p=1,2 in previous sections. The mean of a stationary series is

$$E(a_t) = \frac{\alpha_0}{1 - \alpha_1 - \cdots - \alpha_p}, \tag{3.29}$$

provided $\alpha_1 + \cdots + \alpha_p \neq 1$.

3.3.2. MA Models

We now move on to a different category of straightforward models that can be applied to the modeling of financial return series. These models are known as Moving-average (MA) models. MA models can be introduced in a variety of ways. An effective approach to analyzing a model involves considering it as a direct expansion of a white noise series. Alternatively, the model can be approached as an AR model with an infinite order, subject to specific parameter constraints. We choose the second strategy.

We start with an AR model with infinite order as

$$a_t = \alpha_0 + \alpha_1 a_{t-1} + \alpha_2 a_{t-2} + \cdots + y_t. \tag{3.30}$$

However, because it has an infinite number of parameters, such an AR model is not

useful. Assuming that the coefficients of α_i for all $i \geq 0$ satisfy certain requirements and are thus determined by a finite set of parameters is one technique to make the model workable. A unique instance of this concept is

$$a_t = \alpha_0 - \beta_1 a_{t-1} - \beta_1^2 a_{t-2} - \beta_1^3 a_{t-3} - \cdots + y_t \tag{3.31}$$

when only one parameter determines the coefficients β_1 with $\beta_i = -\beta_1^i$ for all $i > 0$.

In order to achieve stationarity in Equation (3.31), it is crucial for the absolute value of β_1 to be below one. Failing to meet this criterion will result in a non-converging series. In simpler terms, when $|\beta_1| < 1$, the value of β_1^i tends to zero as i approaches infinity. As the value of i increases, the influence of a_{t-i} on a_t decreases exponentially.

The Equation (3.31) can be also written as

$$a_t + \beta_1 a_{t-1} + \beta_1^2 a_{t-2} + \cdots = \alpha_0 + y_t. \tag{3.32}$$

Then the model for a_{t-1}:

$$a_{t-1} + \beta_1 a_{t-2} + \beta_1^2 a_{t-3} + \cdots = \alpha_0 + y_{t-1}. \tag{3.33}$$

Multiplying Equation (3.33) with β_1 and substracting from the Equation (3.32), we get

$$a_t = \alpha_0(1 - \beta_1) + y_t - \beta_1 y_{t-1}. \tag{3.34}$$

Equation (3.34) states that the the series a_t is constant term plus weighted average shocks of y_t and y_{t-1}. As a result, the model is often known as an MA model of order 1 or MA(1) model. A standard form of model MA(1) is

$$a_t = c_0 + y_t - \beta_1 y_{t-1}, \tag{3.35}$$

where y_t is a white noise series and c_0 is a constant. The form of an MA(2) model that is given in the Equation (3.36) below is also similar:

$$a_t = c_0 + y_t - \beta_1 y_{t-1} - \beta_2 y_{t-2}. \tag{3.36}$$

A general MA(q) model is

$$a_t = c_0 + y_t - \beta_1 y_{t-1} - \beta_2 y_{t-2} - \cdots - \beta_q y_{t-q}, \tag{3.37}$$

where $q > 0$.

3.3.2.1. Expectation of MA(1) Model. In Equation (3.35), the MA(1) model is presented. Taking the expectation of the equation, we have

$$E(a_t) = E(c_0) + E(y_t) - E(\beta_1 y_{t-1}) = c_0 + 0 - \beta_1 E(y_{t-1}) = c_0 \tag{3.38}$$

since c_0 is constant and y_t and y_{t-1} are white noise series with mean 0. That means $E(y_t) = 0$ and $E(y_{t-1}) = 0$.

3.3.2.2. Expectation of MA(2) Model. In Equation (3.36), the MA(2) model is presented. Taking the expectation of the equation, we have

$$E(a_t) = E(c_0) + E(y_t) - E(\beta_1 y_{t-1}) - E(\beta_2 y_{t-2}) = c_0 + 0 - \beta_1 E(y_{t-1}) - \beta_2 E(y_{t-2}) = c_0 \tag{3.39}$$

since y_t, y_{t-1} and y_{t-2} are white noise series with mean zero.

3.3.2.3. Expectation of MA(q) Model. In general, we have also have same expectation value for MA(q) model

$$E(a_t) = c_0, \tag{3.40}$$

which is time invariant.

3.3.3. ARMA Models

To accurately capture the dynamic properties of the data, complicated models with many parameters must sometimes be used. When applying the AR or MA models mentioned before, several problems arise. The idea of autoregressive moving-average models is presented to get around this restriction. ARMA models produce a more concise representation with fewer parameters by combining the advantages of autoregressive and moving average models. Although ARMA models are not frequently used in finance to analyze return series, they are quite useful for modeling volatility.

We first study simple ARMA(1,1) model. A time series a_t is said an ARMA(1, 1) model if it is

$$a_t - \alpha_1 a_{t-1} = \alpha_0 + y_t - \beta_1 y_{t-1}, \tag{3.41}$$

where y_t is a white noise series. The AR and MA components of the model are given on the left and right sides, respectively, of the Equation (3.41). α_0 is the constant term. We need $\alpha_1 \neq \beta_1$ for this model to make sense; otherwise, the process degenerates into a white noise series due to an equation cancellation.

A generic ARMA(p, q) model has the following equation

$$a_t = \alpha_0 + \sum_{i=1}^{p} \alpha_i a_{t-i} + y_t - \sum_{i=1}^{q} \beta_i y_{t-i} \tag{3.42}$$

where p and q are non-negative integers, and y_t is a white noise series. The ARMA(p, q) model includes the AR and MA models as special instances.

3.3.3.1. Expectation of ARMA(1,1) Model. Taking the expectation of (3.41) we have,

$$E(a_t) - \alpha_1 E(a_{t-1}) = \alpha_0 + E(y_t) - \beta_1 E(y_{t-1}) \tag{3.43}$$

$$\mu - \alpha_1 \mu = \alpha_0 \tag{3.44}$$

since $E(y_i) = 0$ for all $i > 0$. So we get

$$E(a_t) = \mu = \frac{\alpha_0}{1 - \alpha_1}. \tag{3.45}$$

3.4. Risk Models

In this section, we are going to examine risk models concentrating on the ARCH (Autoregressive Conditional Heteroskedasticity) and GARCH (Generalized ARCH) models. To comprehend volatility dynamics and capture characteristics like volatility clustering, these models use historical squared error terms.

In mathematical analysis of asset and security prices, ARCH models are used heavily. Unfortunately, ARCH models have some drawbacks as well especially when it comes to modeling the inherent volatility in financial asset price data. The volatility behaviour of each financial asset is different in their responses to positive and negative

shocks in the market. ARCH models provide a statistical explanation for conditional variance, but they fail to explain or model what drives volatility in the first place.

As an extension of ARCH models, we present GARCH models to solve these drawbacks. To effectively represent asset return volatility, GARCH models include autoregressive and moving average components. As we shall see, by introducing parameters β_j to quantify the effect of previous conditional variances on current volatility, the GARCH(m, s) model, given by Equation (3.54), expands the ARCH model.

3.4.1. ARCH

Robert F. Engle introduced Autoregressive Conditional Heteroskedasticity (ARCH) models in 1982 to address the drawbacks of conventional models and to represent the time-varying volatility patterns seen in financial data [2] which is widely viewed as an important innovation in econometric modeling. By using historical squared error terms as explanatory variables, ARCH models offer an adaptable framework for understanding volatility dynamics. The basic principle of ARCH models is that current volatility depends on both the recent information found in prior squared errors as well as a constant term. ARCH models are well-suited for capturing the stylized characteristics commonly observed in financial time series, including volatility clustering, skewness, and fat tails. Notably, ARCH models can effectively model volatility clustering, which refers to the tendency for periods of high volatility to be followed by other periods of high volatility, and similarly for low volatility periods.

On a few basic presumptions, the ARCH models are built. First, it is assumed that the mean-corrected asset return r_t is dependent and serially uncorrelated. Second, the dependence of r_t can be explained by a straightforward quadratic function of its lagged values. In an ARCH(m) model, it is assumed

$$r_t = \sigma_t \epsilon_t, \quad \sigma_t^2 = b_0 + b_1 r_{t-1}^2 + \ldots + b_m r_{t-m}^2 \tag{3.46}$$

where ϵ_t is a series of independently and identically distributed (iid) random variables with a mean of 0 and a variance of 1, with $b_0 > 0$ and $b_i \geq 0$ for all $i > 0$. To

guarantee that the unconditional variance of r_t is finite, the coefficients b_i must adhere to a set of regularity requirements. In practice, ϵ_t is frequently thought to follow a typical Student-t distribution or the normal distribution.

From the ARCH model structure in the Equation (3.46), we can conclude that large shock r_{t-i}^2 for $i = 1, \ldots, m$ implies large conditional variance σ_t^2. This also means that, according to the ARCH model, major shocks frequently lead to major shocks.

3.4.1.1. Properties of ARCH Models. The ARCH(1) model is given by

$$r_t = \sigma_t \epsilon_t, \quad \sigma_t^2 = b_0 + b_1 r_{t-1}^2, \tag{3.47}$$

where $b_0 > 0$ and $b_1 \geq 0$. From Equation (3.47), we have

$$E(r_t) = E[E(r_t|F_{t-1})] = E[\sigma_t E(\epsilon_t)] = 0 \tag{3.48}$$

where F_{t-1} stands for the collection of all available information, including past observations and any relevant variables, up until the time period immediately preceding t. Secondly, it is possible to determine the unconditional variance of r_t as

$$\text{Var}(r_t) = E(r_t^2) = E[E(r_t^2|F_{t-1})] = E(b_0 + b_1 r_{t-1}^2) = b_0 + b_1 E(r_{t-1}^2). \tag{3.49}$$

Since r_t is stationary, we have $E(r_t) = 0$ and $\text{Var}(r_t) = \text{Var}(r_{t-1}) = E(r_{t-1}^2)$. We also have $\text{Var}(r_t) = b_0 + b_1 \text{Var}(r_t)$ and $\text{Var}(r_t) = b_0/(1 - b_1)$. We require $0 < b_1 < 1$ because the variance of r_t must be positive. Higher order moments of r_t are required in some applications, therefore b_1 must also adhere to extra restrictions. For instance, we need the fourth moment of r_t to be finite in order to analyze its tail behavior. Using normality assumption for ϵ_t of Equation (3.47), we have

$$E(r_t^4|F_{t-1}) = 3[E(r_t^2|F_{t-1})]^2 = 3(b_0 + b_1 r_{t-1}^2)^2. \tag{3.50}$$

If $m_4 = E(r_t^4)$ and t is fourth-order stationary, then we get

$$m_4 = 3[b_0^2 + 2b_0 b_1 \text{Var}(r_t) + b_1^2 m_4] \tag{3.51}$$

$$m_4 = 3b_0^2 \left(1 + \frac{2b_1}{1 - b_1}\right) + 3b_1^2 m_4 \tag{3.52}$$

$$m_4 = 3b_0^2 \frac{(1 + b_1)}{(1 - b_1)(1 - 3b_1^2)}. \tag{3.53}$$

Considering that the fourth moment of r_t is positive, it implies that the coefficient b_1 must satisfy the condition $1 - 3b_1^2 > 0$. In particular, it is necessary for b_1 to fall within the range $0 \leq b_1^2 < \frac{1}{3}$.

3.4.1.2. Difficulties with ARCH Models. The model, which assumes that volatility is influenced by the squared magnitude of previous shocks, implies that both positive and negative shocks have an equal impact on volatility. However, in practical observations, it is well-known that the price of a financial asset tends to respond differently to positive and negative shocks.

The ARCH model imposes a lot of limitations. For instance, if the series is to have a finite fourth moment, b_1^2 of an ARCH(1) model must be in the range $[0, 1/3]$ as we showed in Equation (3.53). Higher order ARCH models complicate the constraint.

The reasons producing fluctuations in a financial time series are not clearly identified by ARCH models. Instead, they provide a methodical explanation for conditional variance's behavior. Without addressing the particular variables or causes that may contribute to such behavior, these models concentrate on the mechanics of how the conditional variance behaves. As a result, the underlying causes determining the volatility patterns seen in financial time series are not explicitly disclosed by ARCH models.

3.4.2. GARCH

Although the ARCH model is straightforward, it frequently needs a number of parameters to accurately capture the volatility of an asset return. A helpful extension suggested by Bollerslev (1986) is the generalized ARCH (GARCH) model. We assume that an ARMA model can adequately explain the mean equation of the process for a log return series r_t. If r_t follows to the GARCH(m, s) model

$$r_t = \sigma_t \varepsilon_t, \quad \sigma_t^2 = b_0 + \sum_{i=1}^{m} b_i r_{t-i}^2 + \sum_{j=1}^{s} c_j \sigma_{t-j}^2, \tag{3.54}$$

where again ϵ_t is a sequence of iid random variables with mean 0 and variance 1, $b_0 > 0$, $b_i \geq 0$, $c_j \geq 0$ and $\sum_{i=1}^{\max(m,s)}(b_i + c_i) < 1$. Also we have, $b_i = 0$ for $i > m$ and $c_j = 0$ for $j > s$. The unconditional variance of r_t is finite and its conditional variance σ_t^2 varies over time by the constraint on $b_i + c_i$. It is also important to note that Equation (3.54) reduces to a pure ARCH(m) model if $s = 0$.

3.4.2.1. Properties of GARCH Model. We begin with GARCH(1,1) model in order to understand behaviours of GARCH model. A simple definition of GARCH(1,1) is given

$$\sigma_t^2 = b_0 + b_1 r_{t-1}^2 + c_1 \sigma_{t-1}^2, \quad 0 \leq b_1, c_1 \leq 1, \quad (b_1 + c_1) < 1. \tag{3.55}$$

Equation (3.55) indicates that when r_{t-1}^2 or σ_{t-1}^2 is large, it leads to an increase in σ_t^2. This relationship is responsible for the widely observed phenomenon of volatility clustering in financial time series. In other words, a period characterized by a large squared return (r_{t-1}^2) tends to be followed by another period with a similarly large squared return (r_t^2). This clustering of high volatility periods is a distinct characteristic often observed in financial markets.

GARCH model forecasts can be derived using techniques similar to ARMA model techniques. Assume that the forecast origin is time t and take into account the GARCH(1, 1) model in Equation (3.55). For the one step ahead forecast, we have

$$\sigma_{t+1}^2 = b_0 + b_1 r_t^2 + c_1 \sigma_t^2, \tag{3.56}$$

where, at time index t, r_t and σ_t^2 are known. Consequently, the one step ahead forecast is

$$\sigma_t^2(1) = b_0 + b_1 r_t^2 + c_1 \sigma_t^2. \tag{3.57}$$

We use $r_t^2 = \sigma_t^2 \epsilon_t^2$ for multistep ahead forecasts and rewrite the volatility equation in Equation (3.55) as

$$\sigma_{t+1}^2 = b_0 + (b_1 + c_1)\sigma_t^2 + b_1 \sigma_t^2(\epsilon_t - 1). \tag{3.58}$$

Using $t = h + 1$, Equation (3.58) becomes

$$\sigma_{h+2}^2 = b_0 + (b_1 + c_1)\sigma_{h+1}^2 + b_1 \sigma_{h+1}^2(\epsilon_{h+1}^2 - 1). \tag{3.59}$$

The equation is satisfied by the two-step ahead volatility forecast at the forecast origin t as a consequence of the condition $E(\epsilon_{h+1}^2 - 1|F_h) = 0$ and $\sigma_h^2(2) = b_0 + (b_1 + c_1)\sigma_h^2(1)$. In general, we have

$$\sigma_h^2(m) = b_0 + (b_1 + c_1)\sigma_h^2(m-1), \quad m > 1. \tag{3.60}$$

Using several replacements in (3.60), we are able to write the m-step ahead forecast as

$$\sigma_h^2(m) = \frac{b_0[1 - (b_1 + c_1)^{m-1}]}{1 - b_1 - c_1} + (b_1 + c_1)^{m-1}\sigma_h^2(1). \tag{3.61}$$

Therefore, we have

$$\sigma_h^2(m) \to \frac{b_0}{1 - b_1 - c_1}, \quad \text{as} \quad m \to \infty, \tag{3.62}$$

provided that $b_1 + c_1 < 1$. With an increasing forecast horizon that tends towards infinity, the GARCH(1,1) model's multistep forward volatility projections gradually approach and converge to the unconditional variance of r_t.

4. EXPERIMENTS

4.1. ARIMA Models

In this Section, we analyze three different financial time series datasets–BTCUSD, NASDAQ, and BIST100–in order to test the performances of ARIMA models. One may find the details on our datasets in Sections 2.1 and 2.3. We will specifically use the closing prices of each dataset and compute their logarithmic returns. Our hypothesis is that the log returns, which show the percentage change in prices over time, are going to give us some clues on the structural characteristics of these assets to better understand them.

4.1.1. Plots

First, we plot the log returns for each dataset to visualize their distribution and check if they are stationarity. We plot these log-returns in Figures 4.1, 4.2, and 4.3 below.

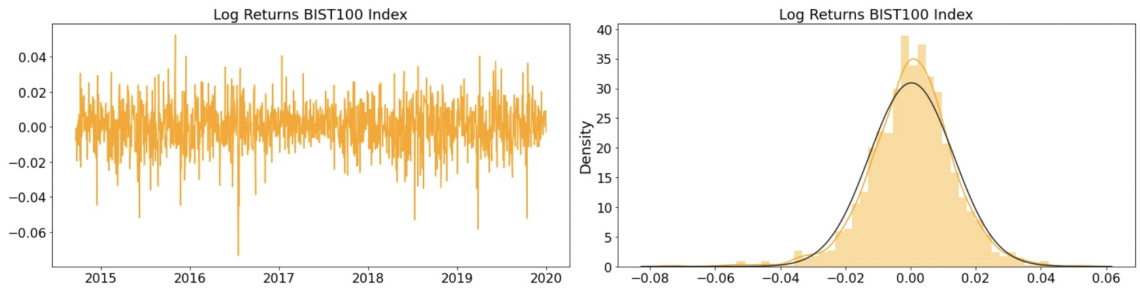

Figure 4.1. BIST100 log return closing price.

As can be seen from the plot, the log returns follow a roughly normal distribution with a mean close to zero. We can also see that there are a few extreme values with large positive or negative returns. For each of the dataset we get similar distributions and their charts for each dataset are shown below.

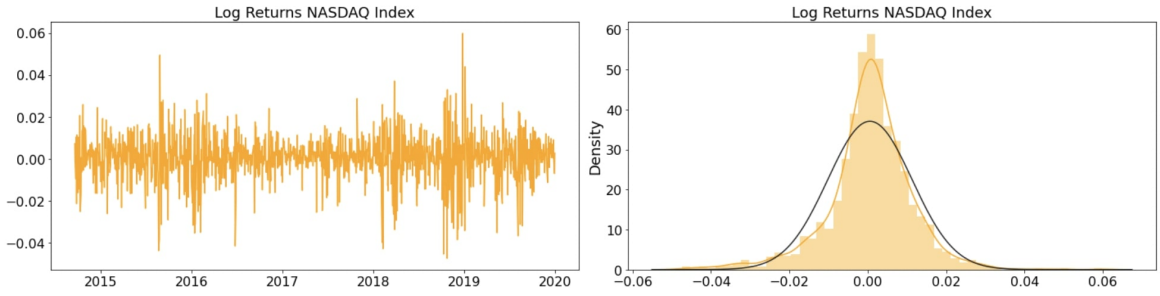

Figure 4.2. NASDAQ log return closing price.

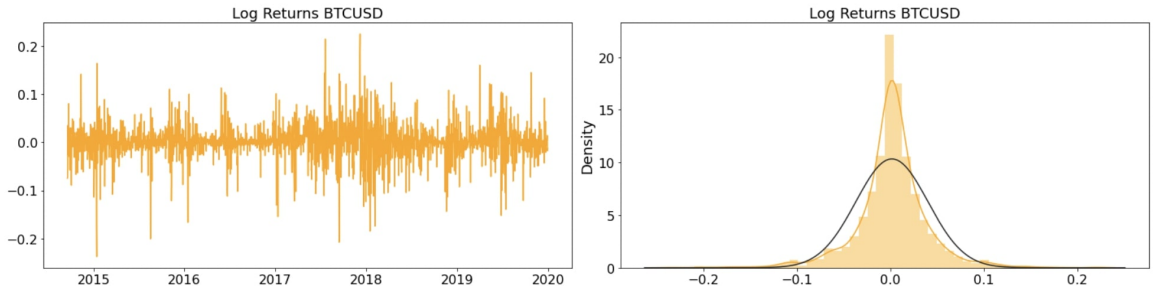

Figure 4.3. BTCUSD log return closing price.

4.1.2. Stationarity

The Augmented Dickey-Fuller (ADF) test is used in the following stage of our study to determine whether the log returns for each dataset are stationary [9]. Specifically, the ADF test is used to determine whether a time series dataset contains *a unit root*. When a time series has a unit root, it is non-stationary, i.e. main statistical characteristics like mean and variance shift over time. Our analyses are done using the python programming language [10] and its library ecosystem. For the experiments for this section, we use the `statsmodels` library. For the ADF test we used the `adfuller` function from the `tsa.stattools` module of the library. The ADF statistic value for each dataset are listed in the Table 4.1 below.

Table 4.1. Augmented Dickey–Fuller test results for all datasets

Assets	ADF Statistics	p-value
BTCUSD	-13.17	0.0002
NASDAQ	-14.29	0.0001
BIST100	-10.89	0.0004

4.1.3. Autocorrelations

Next, we analyze the ACF and PACF charts for each dataset. These charts are shown in Figure 4.4 and 4.5. The correlation between a time series and its lagged values at various time lags is displayed on the ACF plot. The PACF plot, on the other hand, after accounting for the effects of intermediate delays, displays the partial correlation between the time series and its lagged values. The appropriate order of differencing d, the order of the autoregressive term p, and the order of the moving average term q can be determined using these charts for each ARIMA model.

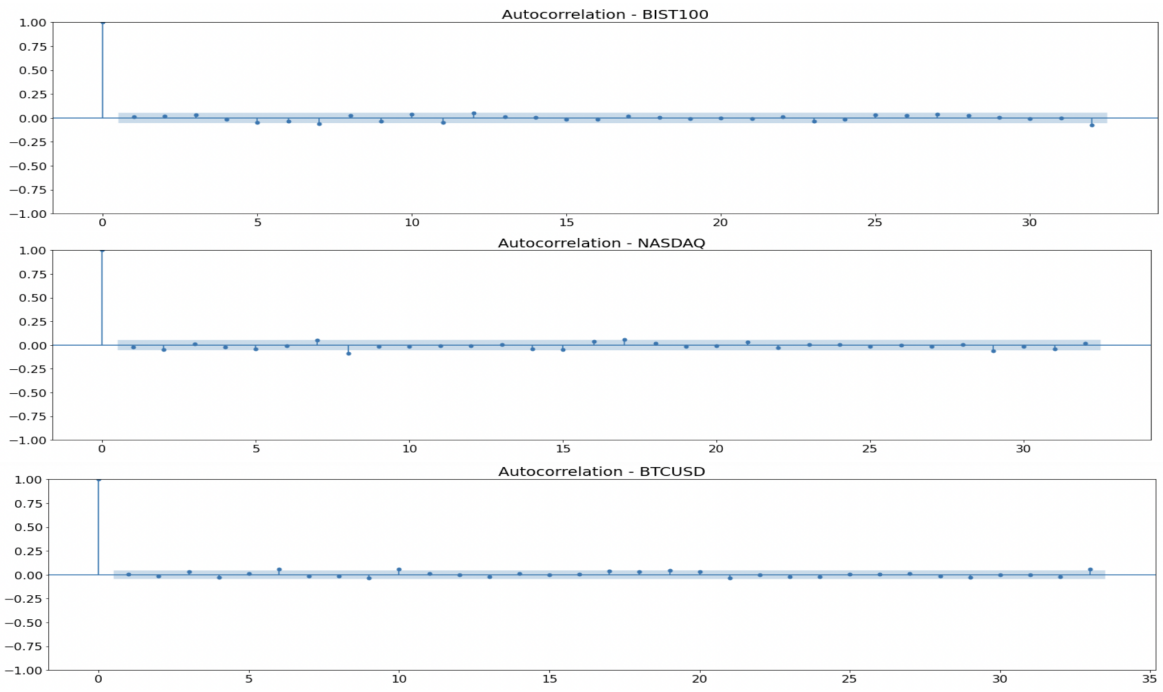

Figure 4.4. ACF plots.

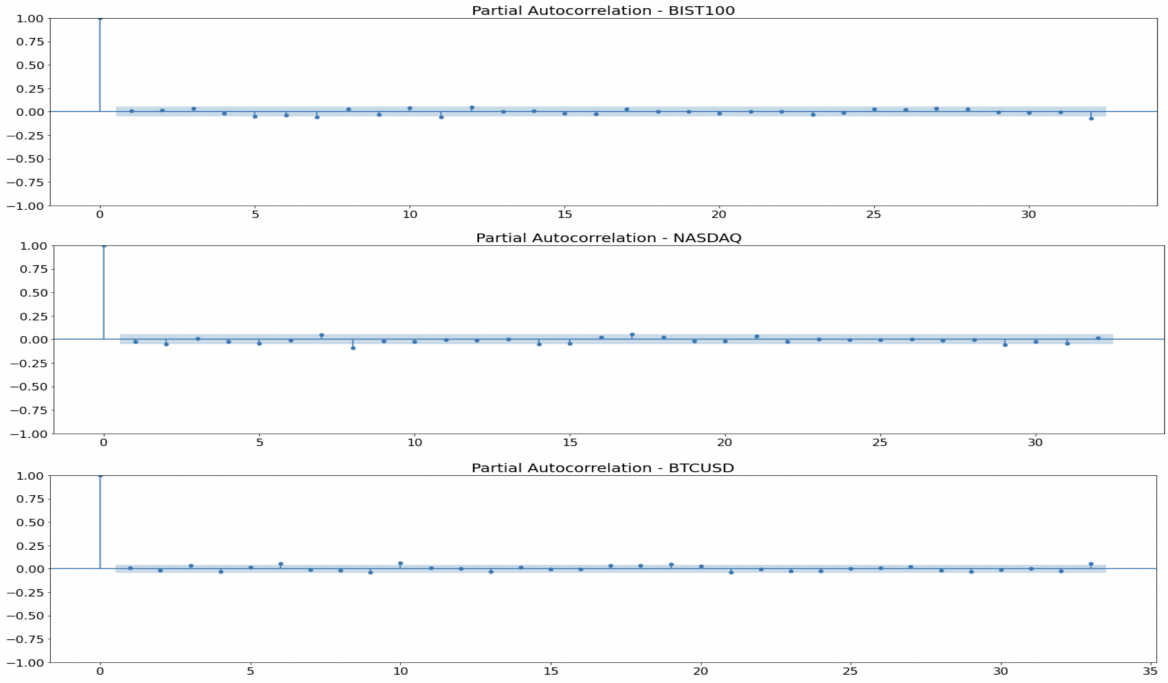

Figure 4.5. PACF plots.

4.1.4. ARIMA Parameters

After creating the plots, we did a hyperparameter tuning by grid searching appropriate values for p, d and q on our three other datasets using the `pmdarima` package. [11]. The library finds the the best values for p, d and q based on the Akaike Information Criterion (AIC). In our grid search, for the BIST100, NASDAQ, and BTCUSD, respectively, we specified the maximum value for p and q as 6, and the maximum value for m of as 5, 5, and 7 respectively since the crypto market is open 7 days including weekends while BIST100 and NASDAQ are only open on 5 business days. We also set the value of d to zero because there is no need for differencing in the log results. The chosen ARIMA parameters are listed in Table 4.2 below.

Table 4.2. Augmented Dickey–Fuller test results for all datasets

Assets	p	d	q
BTCUSD	2	0	0
NASDAQ	0	0	2
BIST100	0	0	1

4.1.5. Analysis of Residuals

We then prepared a summary of the model parameters and diagnostics after fitting the auto ARIMA model to each dataset. These diagnostics comprised a residuals histogram, a Q-Q plot, a correlogram, and a residuals plot over time. These diagnostics assist us in determining the accuracy of our model's fit to the data and the validity of our data-related hypotheses. Residuals plot and residuals histogram are shown in Figures 4.6, 4.7 and 4.8 for each dataset.

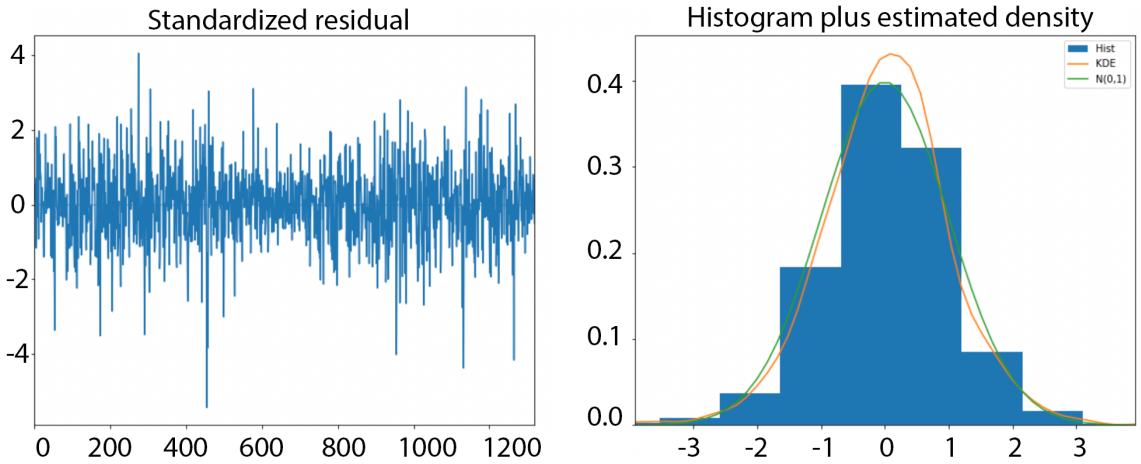

Figure 4.6. Residuals and residual histogram for BIST100.

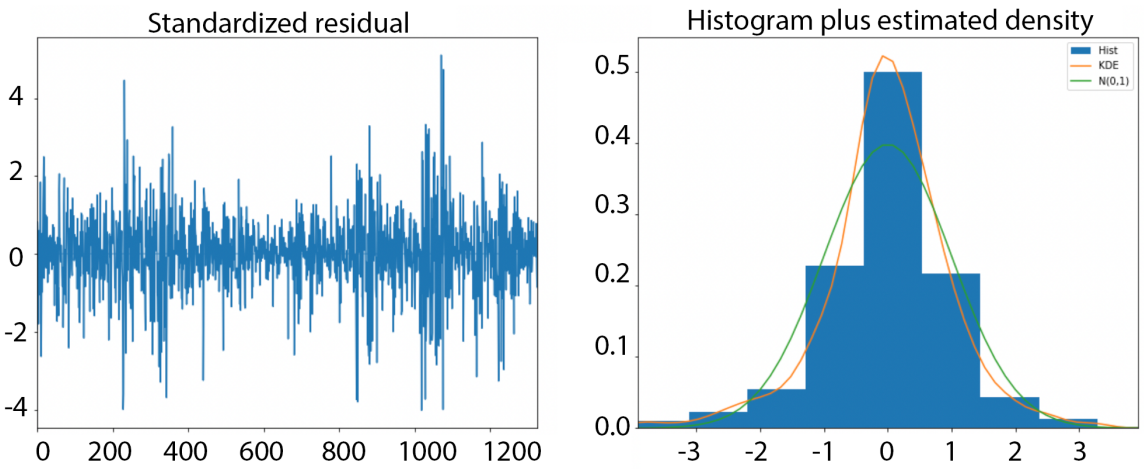

Figure 4.7. Residuals and residual histogram for NASDAQ.

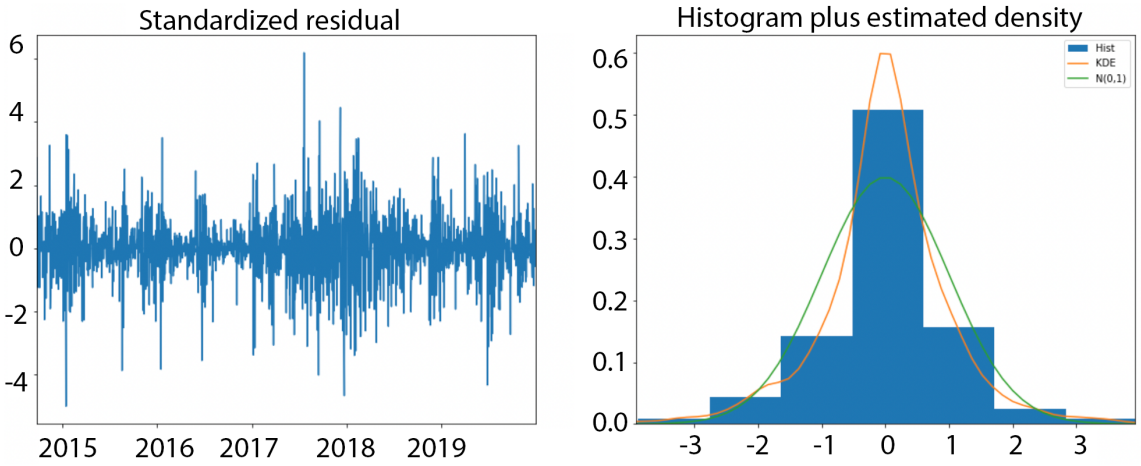

Figure 4.8. Residuals and residual histogram for BTCUSD.

In order to formally check for normality, we used the Kolmogorov-Smirnov test (KS-test) on the residuals. The KS-test evaluates the correspondence between the empirical residual distribution and normal distribution. If the p-value of the test exceeds a significance level of 0.05, the null hypothesis stating that the residuals are normally distributed cannot be rejected. In such cases, it is not determined that the residuals are not normally distributed. However, if the p-value is less than or equal to 0.05, the null hypothesis is rejected, indicating that the residuals are not normally distributed.

When we examined the residuals we observed that none of the residuals had a normal distribution. The KS-test, which revealed that the residuals are not Gaussian and that the null hypothesis was rejected at a significance level of 0.05, indicated this conclusion.

To analyze normality for residuals for each dataset, we also created QQ-plots and correlograms. The QQ-plots display the theoretical quantiles and the sample quantiles corresponding to residuals. The autocorrelation of the residuals at various lags is displayed in the correlograms. These QQ-plots are shown in Figure 4.9.

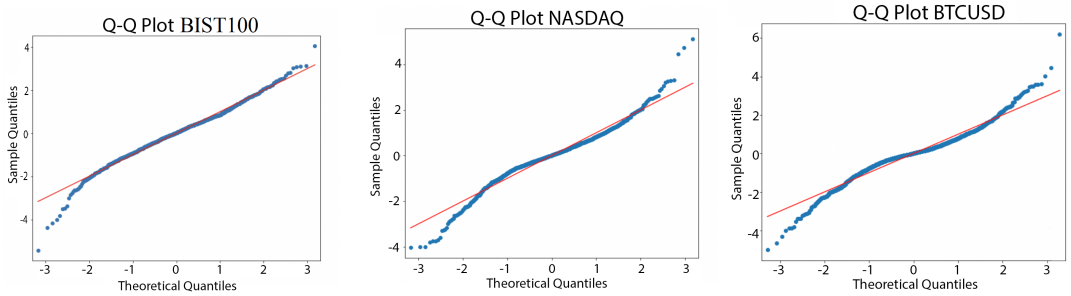

Figure 4.9. QQ plots of residuals for all datasets.

The QQ-plot shows that the residuals have heavier tails than a normal distribution, which means it deviates from the normal distribution at its tails.

We determined structural AR parameters of our datasets by investigating the autocorrelation patterns in the time series data. Strong autocorrelation is seen at lags 5 and 7 in the BIST100, negative correlation is seen at lags 2 and 8 in the NASDAQ, and significant autocorrelation is seen at lags 6 and 10 in the BTCUSD according to the ACF and PACF plots.

Correlogram for residuals of each dataset is shown in Figure 4.10. Correlogram analysis of the residuals also reveals strong autocorrelation at lags 2 and 5 for BIST100, lags 5 and 8 for NASDAQ, and lags 5, 6, and 10 for BTCUSDT, showing the existence of underlying patterns that were not captured by the model.

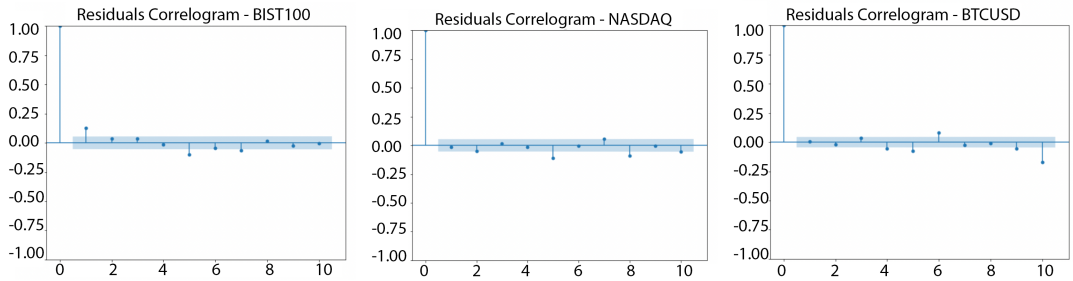

Figure 4.10. Correlogram of residuals for all datasets.

4.2. GARCH Models

In this section, our objective is to assess the performance of the Generalized Autoregressive Conditional Heteroskedasticity (GARCH) model on three distinct datasets: BTCUSD, NASDAQ, and BIST100. We want to determine how well the GARCH model recognizes the volatility for each dataset.

The log returns of each dataset, which are calculated by taking first derivative of the natural logarithm of the closing prices, will be modeled using the GARCH model. The distribution of logarithmic returns for each dataset was previously discussed in Section 4.1, and the plots are shown in Figures 4.1, 4.2, and 4.3. Additionally, we displayed the ADF test results in Section 4.1 and Table 4.1. In this experiment, the identical outcomes from Section 4.1 were utilized.

In order to asses the GARCH models, we are going to use a different strategy. We are going to treat these models as *forecasting models* and assess their performances as such. Before, since our main objective for ARIMA models was not to forecast the log-return series but understand structural ARIMA constants, we used a hyperparameter tuning approach using AIC scores as the comparison measure. So, for the GARCH models we are going to first split our data into training and test sets. We uniformly used the data pertaining the last year in each dataset as our test set while the rest is used for training our GARCH models. In Figure 4.11, we provide one instance of such a separation. The training set of log returns are displayed in blue, and the test set log returns are displayed in green. We displayed the ADF results for our datasets in Section 4.1.2. We now display the ADF results for the training sets in Table 4.3, 4.4 and 4.5.

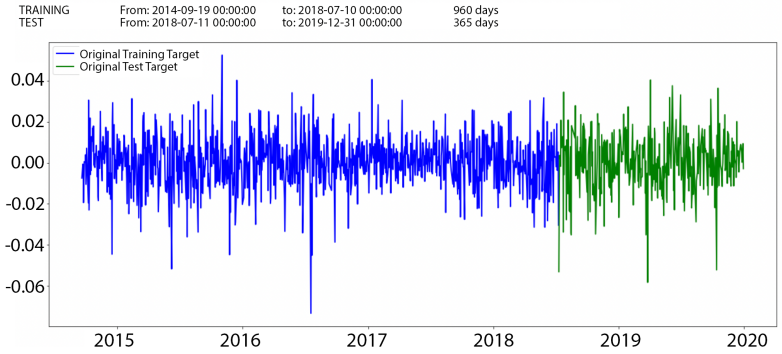

Figure 4.11. NASDAQ log return train test split.

Table 4.3. Augmented Dickey–Fuller test result for BIST100 training set.

ADF Statistics	-37.1894
p-value	0.0
Critical Values	1%: -3.4353 5%: -2.8637 10%: -2.5679

Table 4.4. Augmented Dickey–Fuller test result for NASDAQ training set.

ADF Statistics	-10.8985
p-value	1.1749963660103428e-19
Critical Values	1%: -3.4353 5%: -2.8637 10%: -2.5679

Table 4.5. Augmented Dickey–Fuller test result for BTCUSD training set.

ADF Statistics	-13.1724
p-value	1.22232195266054e-27
Critical Values	1%: -3.4338 5%: -2.8630 10%: -2.5676

Next, we investigate the autocorrelation and partial autocorrelation of the squared log returns for each financial dataset in our training sets. There are two main reasons why we choose squared of log returns instead of just log returns. One of them, it is

well known that financial markets display non-linear behavior including a phenomena known as the leverage effect. According to the leverage effect, larger negative returns are typically accompanied by higher volatility than larger positive returns. Positive and negative returns are included while squaring the log returns, which enables the GARCH model to detect this non-linear relationship and adjust volatility accordingly. Second, the squared log returns tend to exhibit better properties for stationarity and autocorrelation compared to the original log returns. The squared log returns can often exhibit better stationarity properties, making them more suitable for GARCH modeling. We use the ACF and the PACF to analyze the correlation between the squared log returns at different lags, and plot the ACF and PACF results for the squared log returns of each financial dataset. The results are shown in Figures 4.12 and 4.13 below.

The ACF plot for the BIST100 dataset reveals considerable positive autocorrelation at lags 3 and 7, as well as significant partial autocorrelation at lags 3 and 7. Only at lag 3 does the ACF plot for the NASDAQ dataset reveal a significant positive autocorrelation. Significant spikes can also be seen at lag 3 in the PACF plot. Last but not least, the ACF plot for the BTCUSD dataset reveals a strong positive autocorrelation from lag 1 to lag 7. However, the positive autocorrelation is excessive when compared to other lags. On the other hand, the PACF map reveals a considerable peak at lags 1, 3, and 5. Lag 1 is significantly higher than other lags for PACF once again.

Using the `archmodel` function of the `arch` library [12], we fit GARCH models with the desired number of the moving average (q) and autoregression (p) terms. We chose specific values of p and q based on the ACF and PACF plots of the logarithmic returns data. As a result, we determined that a GARCH(3,3) model for the BIST100 dataset, a GARCH(3,3) model for the NASDAQ dataset, and a GARCH(5,5) model for the BTCUSD dataset is most appropriate.

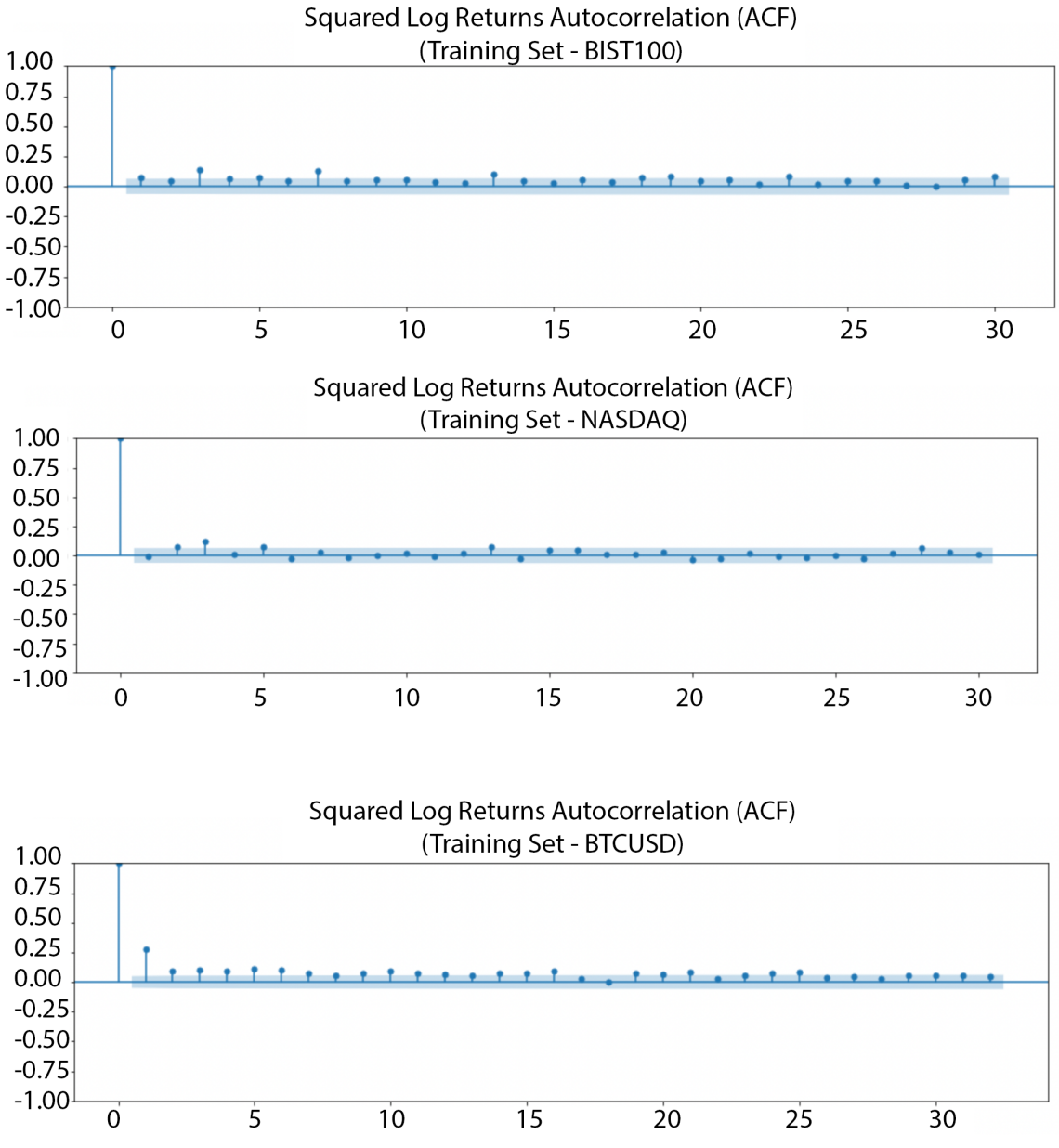

Figure 4.12. Squared log-return train autocorrelation for all datasets.

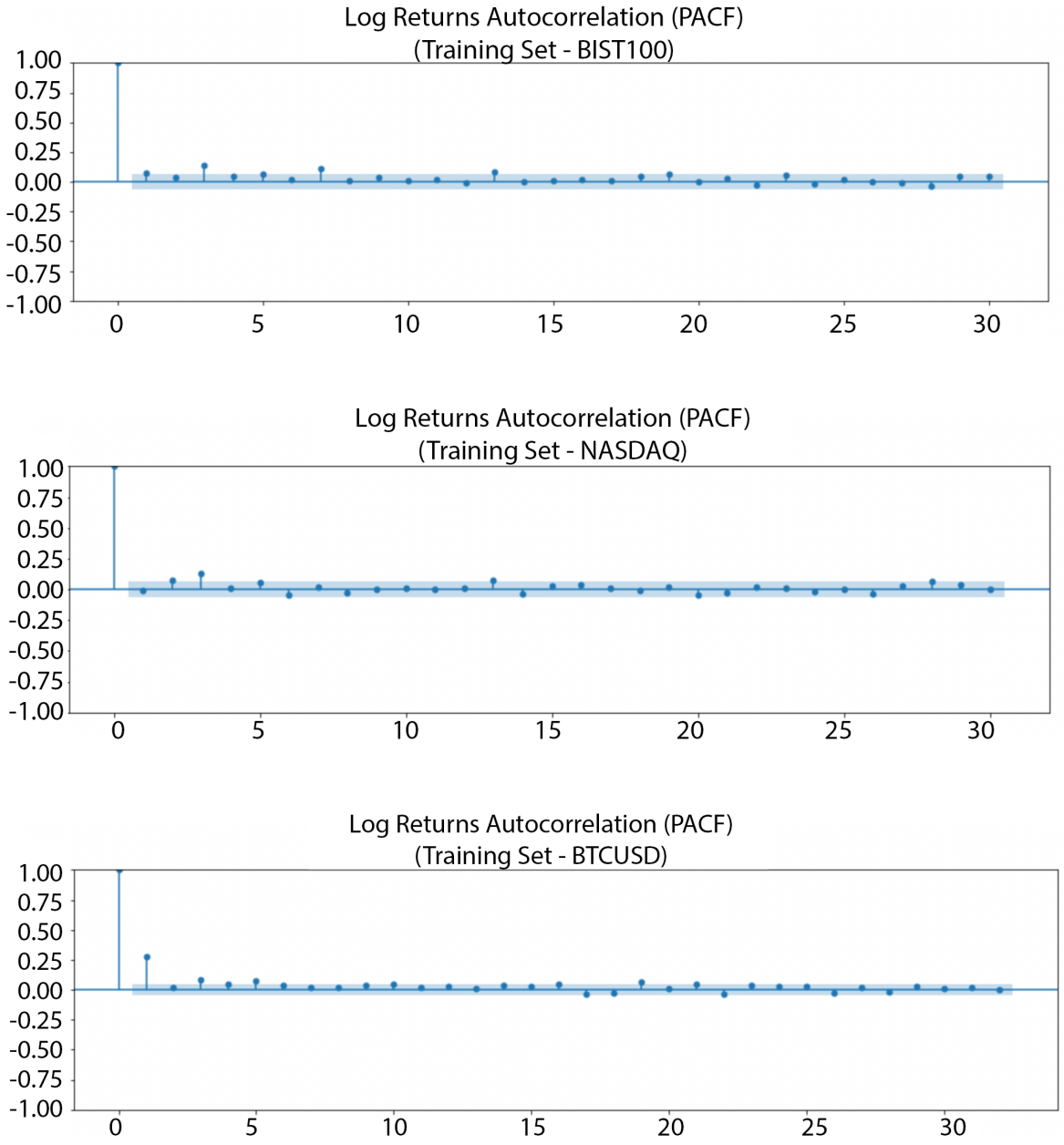

Figure 4.13. Squared log-return train partial autocorrelation for all datasets.

We then evaluated the relevance of the coefficients after fitting the GARCH model to the logarithmic returns training data for each dataset. The coefficient estimations show how the prior volatility shock affected the level of volatility at the moment. The findings from the GARCH models for each dataset are shown in Figures 4.14, 4.15, and 4.16.

Constant Mean - GARCH Model Results

Dep. Variable:	log_returns	R-squared:	0.000
Mean Model:	Constant Mean	Adj. R-squared:	0.000
Vol Model:	GARCH	Log-Likelihood:	2983.11
Distribution:	Normal	AIC:	-5950.23
Method:	Maximum Likelihood	BIC:	-5911.24
		No. Observations:	966
Date:	Thu, May 11 2023	Df Residuals:	965
Time:	10:34:15	Df Model:	1

Mean Model

	coef	std err	t	P>|t|	95.0% Conf. Int.
mu	7.2562e-04	4.267e-04	1.701	8.902e-02	[-1.107e-04,1.562e-03]

Volatility Model

	coef	std err	t	P>|t|	95.0% Conf. Int.
omega	2.6289e-06	1.968e-11	1.336e+05	0.000	[2.629e-06,2.629e-06]
alpha[1]	0.0167	1.149e-02	1.451	0.147	[-5.852e-03,3.919e-02]
alpha[2]	0.0167	0.113	0.147	0.883	[-0.205, 0.239]
alpha[3]	0.0167	0.132	0.127	0.899	[-0.242, 0.275]
beta[1]	0.3100	0.487	0.637	0.524	[-0.644, 1.264]
beta[2]	0.3100	1.694	0.183	0.855	[-3.010, 3.630]
beta[3]	0.3100	2.015	0.154	0.878	[-3.639, 4.259]

Figure 4.14. BIST100 GARCH(3,3).

Constant Mean - GARCH Model Results

Dep. Variable:	log_returns	R-squared:	0.000
Mean Model:	Constant Mean	Adj. R-squared:	0.000
Vol Model:	GARCH	Log-Likelihood:	2852.23
Distribution:	Normal	AIC:	-5688.46
Method:	Maximum Likelihood	BIC:	-5649.52
		No. Observations:	960
Date:	Thu, May 11 2023	Df Residuals:	959
Time:	10:34:15	Df Model:	1

Mean Model

	coef	std err	t	P>|t|	95.0% Conf. Int.
mu	5.0018e-04	3.939e-04	1.270	0.204	[-2.718e-04,1.272e-03]

Volatility Model

	coef	std err	t	P>|t|	95.0% Conf. Int.
omega	3.1473e-06	1.069e-10	2.945e+04	0.000	[3.147e-06,3.148e-06]
alpha[1]	0.0167	3.670e-02	0.454	0.650	[-5.527e-02,8.861e-02]
alpha[2]	0.0167	2.491e-02	0.669	0.503	[-3.216e-02,6.549e-02]
alpha[3]	0.0167	2.305e-02	0.723	0.470	[-2.851e-02,6.184e-02]
beta[1]	0.3100	0.337	0.920	0.358	[-0.350, 0.970]
beta[2]	0.3100	0.428	0.724	0.469	[-0.530, 1.150]
beta[3]	0.3100	0.438	0.707	0.479	[-0.549, 1.169]

Figure 4.15. NASDAQ GARCH(3,3).

```
                     Constant Mean - GARCH Model Results
==============================================================================
Dep. Variable:            log_returns   R-squared:                       0.000
Mean Model:             Constant Mean   Adj. R-squared:                  0.000
Vol Model:                      GARCH   Log-Likelihood:                3023.45
Distribution:                  Normal   AIC:                           -6022.90
Method:            Maximum Likelihood   BIC:                           -5958.62
                                        No. Observations:                 1566
Date:                Thu, May 11 2023   Df Residuals:                     1565
Time:                        10:34:15   Df Model:                            1
                            Mean Model
==============================================================================
                 coef    std err          t      P>|t|       95.0% Conf. Int.
------------------------------------------------------------------------------
mu         1.1848e-03  8.398e-04      1.411      0.158 [-4.613e-04,2.831e-03]
                          Volatility Model
==============================================================================
                 coef    std err          t      P>|t|       95.0% Conf. Int.
------------------------------------------------------------------------------
omega      1.5453e-04  8.616e-04      0.179      0.858 [-1.534e-03,1.843e-03]
alpha[1]       0.0400  2.801e-02      1.428      0.153 [-1.489e-02,9.490e-02]
alpha[2]       0.0400  3.760e-02      1.064      0.287   [-3.370e-02,  0.114]
alpha[3]       0.0400  1.464e-02      2.733  6.283e-03  [1.131e-02,6.869e-02]
alpha[4]       0.0400      0.460  8.691e-02      0.931      [ -0.862,  0.942]
alpha[5]       0.0400      0.132      0.302      0.763      [ -0.220,  0.300]
beta[1]        0.1400      1.812  7.725e-02      0.938      [ -3.412,  3.692]
beta[2]        0.1400      1.175      0.119      0.905      [ -2.163,  2.443]
beta[3]        0.1400      2.850  4.912e-02      0.961      [ -5.446,  5.726]
beta[4]        0.1400      2.758  5.076e-02      0.960      [ -5.266,  5.546]
beta[5]        0.1400      0.235      0.595      0.552      [ -0.321,  0.601]
==============================================================================
```

Figure 4.16. BTCUSD GARCH(5,5).

We noticed that several of the model coefficients are not statistically significant after fitting the GARCH models with the initial p and q values. We made the decision to gradually reduce the orders of p and q values by 1 until all coefficients are significant at a 0.05 threshold in order to generate better-fitting models.

Finding the best choices for p and q that capture the volatility patterns in the data was necessary to adapt the GARCH model to our dataset. In order to do this, we used a stepwise method, repeatedly lowering the orders of p and q values. We began by fitting the GARCH model to the logarithmic return training dataset using the maximum values for p and q. Then, using a threshold p-value of 0.05, we looked at the coefficients and eliminated any that were not statistically significant. The p and q values were reduced by 1 each time we went through this process, doing so until all coefficients were significant. Using this method, we were able to fine-tune the GARCH model for each dataset and make sure that it accurately reflected the volatility patterns in data.

We concluded that the GARCH(1,1) model is suitable for all dataset since all of the coefficients appear to be meaningful in it and the results can be found in Figures 4.17,4.18 and 4.19 below.

```
                     Constant Mean - GARCH Model Results
==============================================================================
Dep. Variable:            log_returns   R-squared:                       0.000
Mean Model:             Constant Mean   Adj. R-squared:                  0.000
Vol Model:                      GARCH   Log-Likelihood:               2983.30
Distribution:                  Normal   AIC:                          -5958.60
Method:            Maximum Likelihood   BIC:                          -5939.11
                                        No. Observations:                  966
Date:                Thu, May 11 2023   Df Residuals:                      965
Time:                        10:57:50   Df Model:                            1
                            Mean Model
==============================================================================
                 coef    std err          t      P>|t|      95.0% Conf. Int.
------------------------------------------------------------------------------
mu         7.3374e-04  3.238e-04      2.266  2.343e-02 [9.917e-05,1.368e-03]
                          Volatility Model
==============================================================================
                 coef    std err          t      P>|t|      95.0% Conf. Int.
------------------------------------------------------------------------------
omega      2.6287e-06  7.859e-12  3.345e+05      0.000 [2.629e-06,2.629e-06]
alpha[1]       0.0500  5.380e-03      9.293  1.497e-20 [3.945e-02,6.054e-02]
beta[1]        0.9300  2.150e-03    432.656      0.000     [  0.926,  0.934]
==============================================================================
```

Figure 4.17. BIST100 GARCH(1,1).

```
                     Constant Mean - GARCH Model Results
==============================================================================
Dep. Variable:            log_returns   R-squared:                       0.000
Mean Model:             Constant Mean   Adj. R-squared:                  0.000
Vol Model:                      GARCH   Log-Likelihood:               2849.56
Distribution:                  Normal   AIC:                          -5691.13
Method:            Maximum Likelihood   BIC:                          -5671.66
                                        No. Observations:                  960
Date:                Thu, May 11 2023   Df Residuals:                      959
Time:                        10:57:50   Df Model:                            1
                            Mean Model
==============================================================================
                 coef    std err          t      P>|t|      95.0% Conf. Int.
------------------------------------------------------------------------------
mu         4.0972e-04  4.354e-05      9.411  4.908e-21 [3.244e-04,4.950e-04]
                          Volatility Model
==============================================================================
                 coef    std err          t      P>|t|      95.0% Conf. Int.
------------------------------------------------------------------------------
omega      1.0139e-05  1.258e-12  8.063e+06      0.000 [1.014e-05,1.014e-05]
alpha[1]       0.0339  1.483e-02      2.287  2.218e-02 [4.854e-03,6.299e-02]
beta[1]        0.9021  1.506e-02     59.909      0.000     [  0.873,  0.932]
==============================================================================
```

Figure 4.18. NASDAQ GARCH(1,1).

```
                     Constant Mean - GARCH Model Results
==============================================================================
Dep. Variable:             log_returns   R-squared:                       0.000
Mean Model:              Constant Mean   Adj. R-squared:                  0.000
Vol Model:                       GARCH   Log-Likelihood:                3049.14
Distribution:                   Normal   AIC:                           -6090.29
Method:            Maximum Likelihood   BIC:                           -6068.86
                                        No. Observations:                 1566
Date:                Thu, May 11 2023   Df Residuals:                     1565
Time:                        10:57:50   Df Model:                            1
                                 Mean Model
============================================================================
                 coef    std err          t      P>|t|       95.0% Conf. Int.
----------------------------------------------------------------------------
mu         1.2994e-03  7.083e-04      1.834  6.659e-02 [-8.889e-05,2.688e-03]
                            Volatility Model
============================================================================
                 coef    std err          t      P>|t|      95.0% Conf. Int.
----------------------------------------------------------------------------
omega      3.1030e-05  5.043e-06      6.153  7.611e-10 [2.115e-05,4.091e-05]
alpha[1]       0.1040  2.008e-02      5.180  2.219e-07   [6.467e-02,  0.143]
beta[1]        0.8760  1.907e-02     45.932      0.000     [  0.839,  0.913]
============================================================================
```

Figure 4.19. BTCUSD GARCH(1,1).

Using these GARCH models, we try to forecast next 365 days variance. In all cases, after some point, variance outputs of model converges. One example is shown from BIST100 dataset for daily period in Figure 4.20.

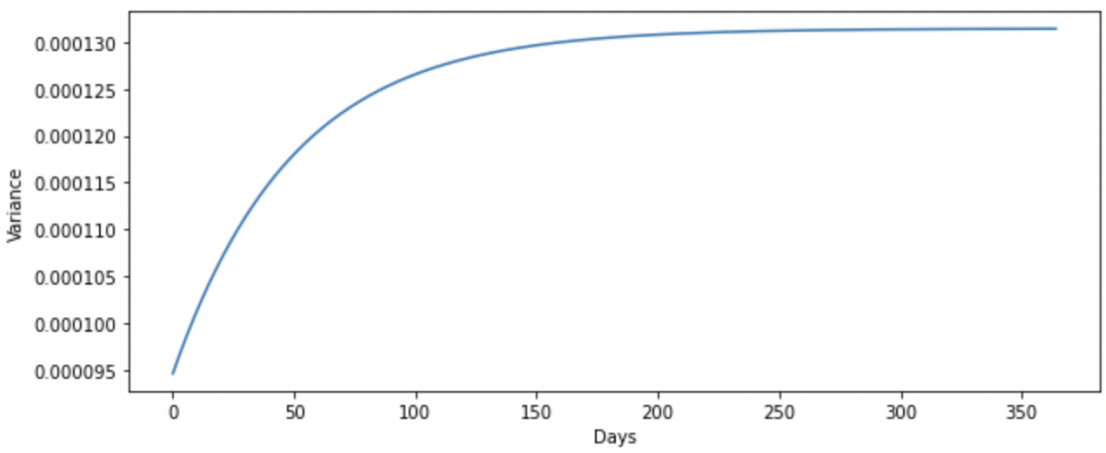

Figure 4.20. An example of a model variance forecast result for 1 year.

We then analyze the standardized residuals and the conditional volatility derived from fitting these GARCH models to the log returns in order to better understand the dynamics of volatility and the behavior of the residuals over time. Residual plots for each training dataset can be found in Figures 4.21, 4.22, and 4.23. These graphs offer crucial data insights and can be used to spot patterns and trends in the volatility of

three datasets. The standardized residuals plots reflect how well the model captures patterns in the data. We expect residuals to be random, with no systematic patterns. Yet, a model that is not well fitted will result in residuals with patterns that show poor fit. On the other hand, annualized conditional volatility plots reflect the estimated volatility of the data over time. These plots for our training dataset are shown in Figures 4.24, 4.25, and 4.26. These plots help us identify periods of high and low volatility and how it develops over time.

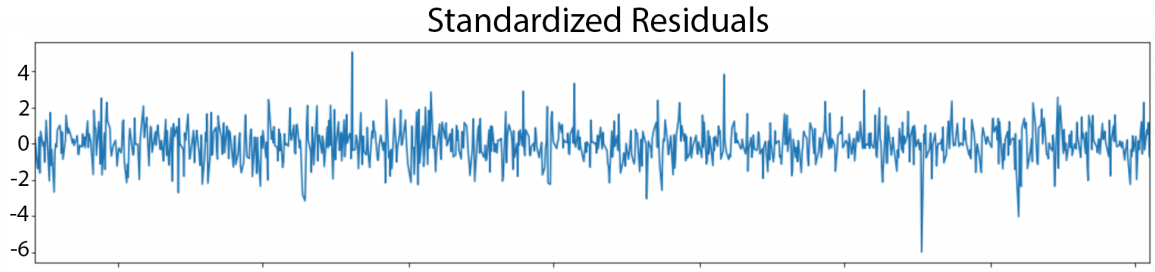

Figure 4.21. BIST100 residuals plot.

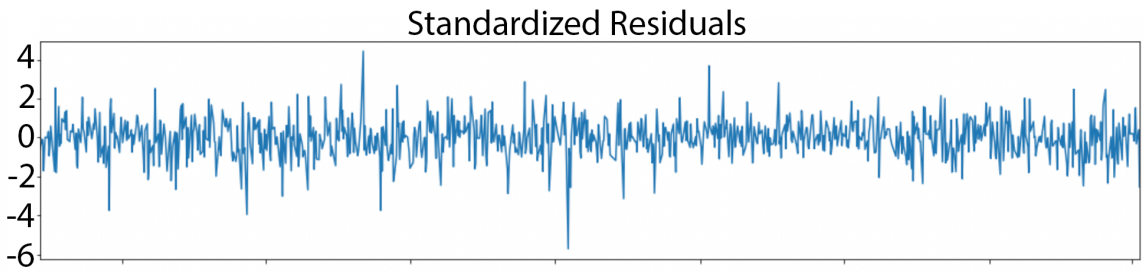

Figure 4.22. NASDAQ residuals plot.

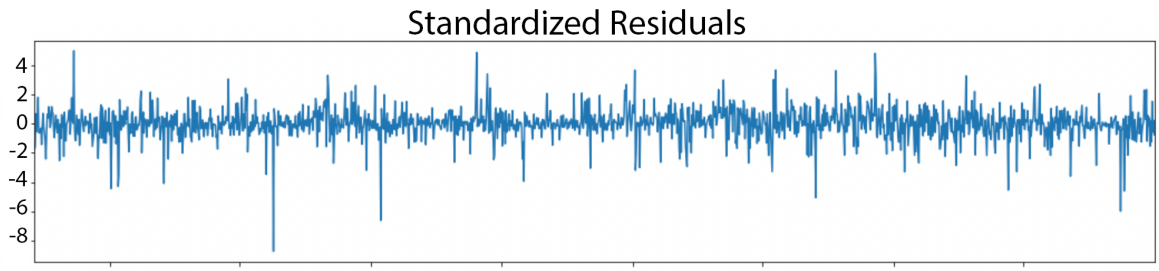

Figure 4.23. BTCUSD residuals plot.

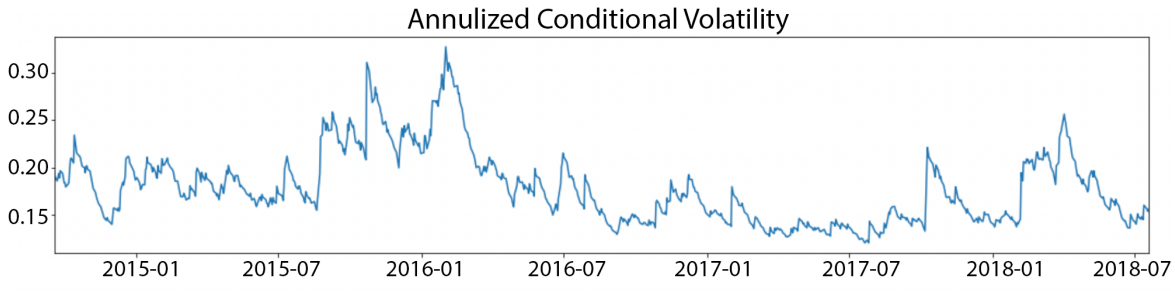

Figure 4.24. BIST100 conditional volatility plot.

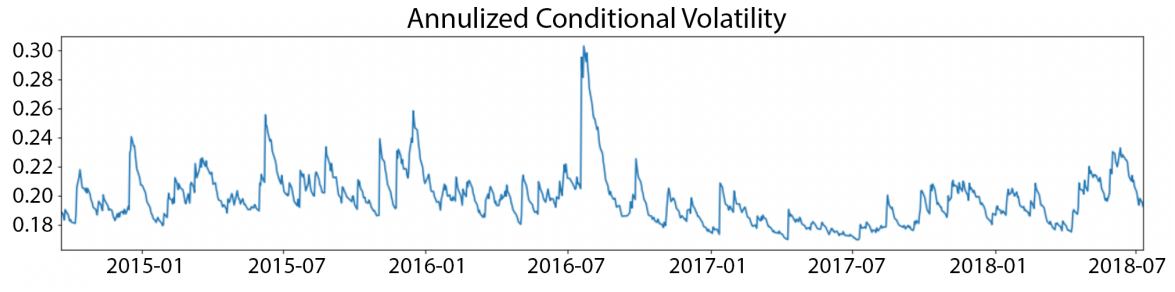

Figure 4.25. NASDAQ conditional volatility plot.

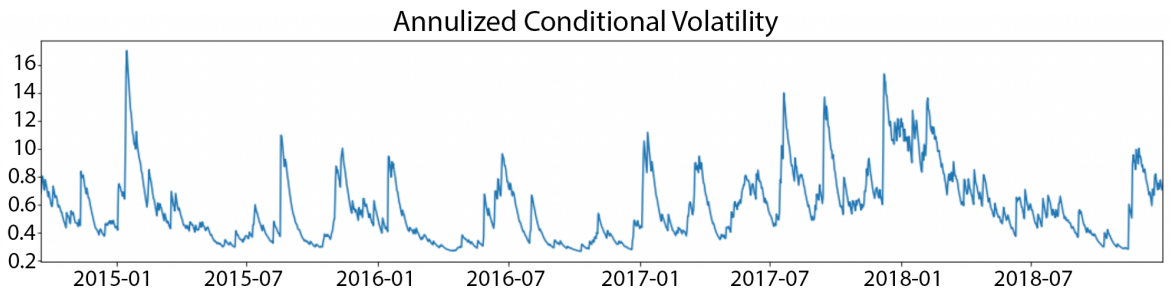

Figure 4.26. BTCUSD conditional volatility plot.

4.3. Rolling Window Models on Variance

We ran number of tests to evaluate the performance of models that we described earlier. We trained multiple classical machine learning models on our datasets with Lagged Variable Generation as we discussed in Section 2.4. We analyzed the inherent volatility in our datasets using the rolling windows approach. For the ARCH models,

each rolling window is fed to the variance model. For the XGBoost and Decision Tree Regression models variances of each window is calculated before they are fed into the models. To assess the efficiency of model, we used the explained variance score.

4.3.1. ARCH Models

We first start with a straight forward ARCH models on rolling windows of each dataset. Predicted variance for log-returns and actual log-returns plotted for each model at Figures 4.27, 4.28, and 4.29.

In this experiment we train the model on a portion of historical data that is updated every step. In each step, we train a new model on all of the past data points and forecast 1 day ahead. Observe that at each iteration the size of the training dataset is increasing.

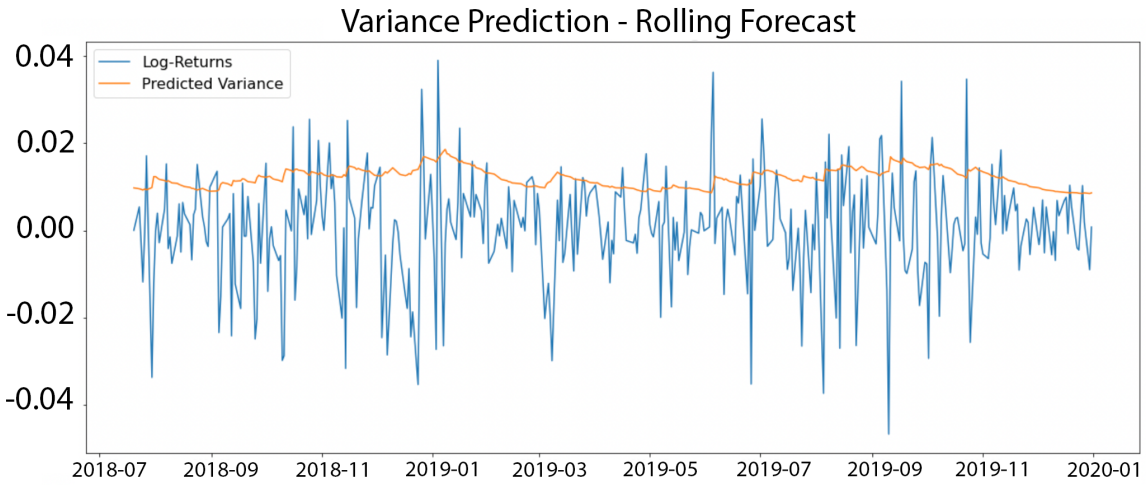

Figure 4.27. Variance forecast with rolling forecast approach - BIST100.

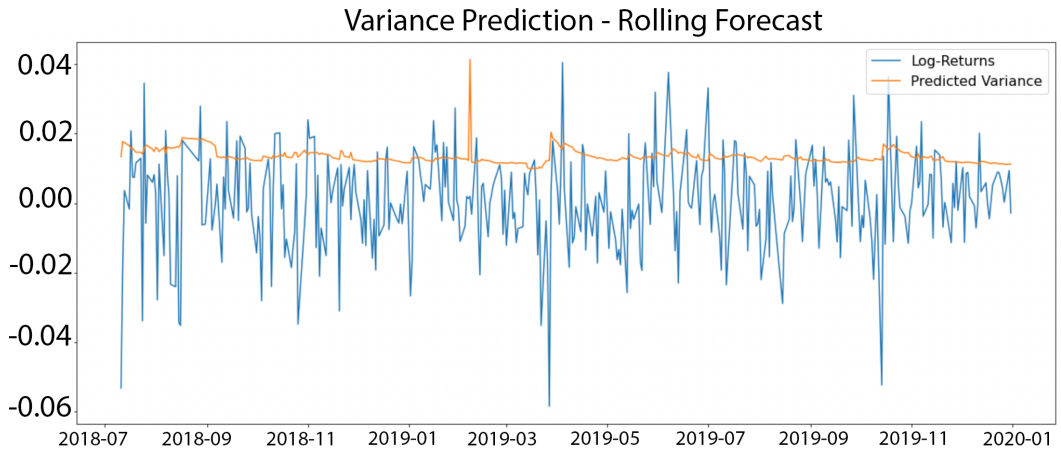

Figure 4.28. Variance forecast with rolling forecast approach - NASDAQ.

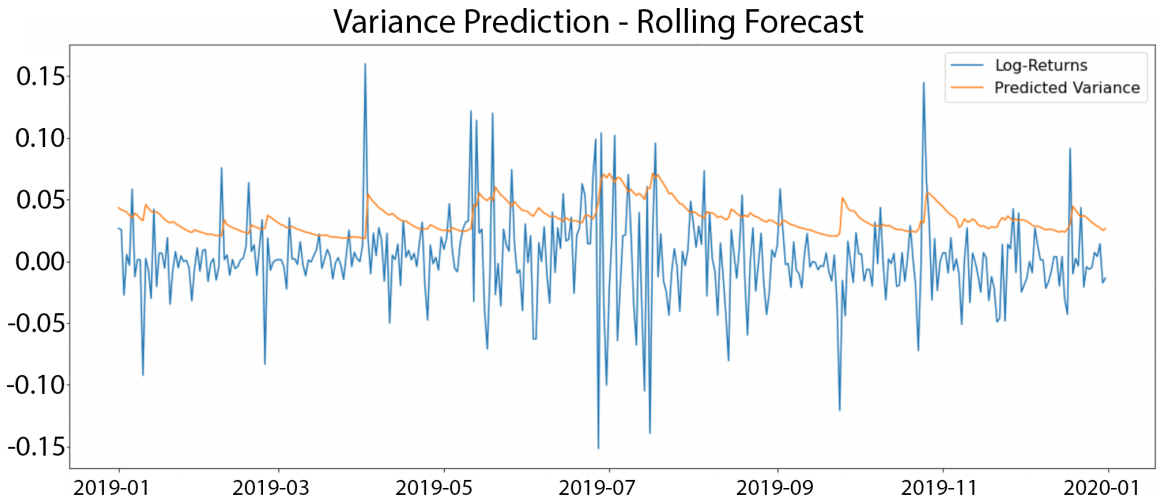

Figure 4.29. Variance forecast with rolling forecast approach - BTCUSD.

To assess the accuracy of the model, we compare the predicted variance with the actual values of log-returns. To quantify the extent to which the predicted variance captures the log returns data, we introduce the concept of the *betweenness*. The *betweenness* is the percentage of log returns that lie in between the predicted band. In Figure 4.30, one instance is given for visualization purposes for BTCUSD dataset on daily period. This percentage gives an indication of how well the GARCH model captures the variations in the log-returns and how well the model can estimate the

volatility of the asset. We calculate that the betweenness are approximately 72%, 70% and 79% for BIST100, NASDAQ and BTCUSD, respectively.

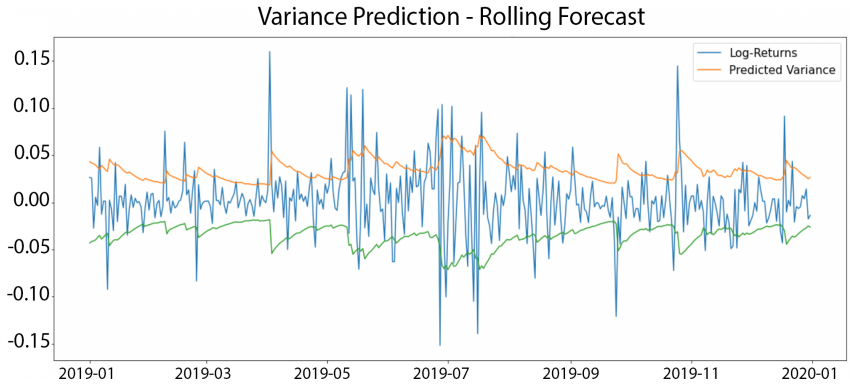

Figure 4.30. Betweenness - BTCUSD

4.3.2. XGBoost

In this section, we outline our experiments in modeling the variance of the BIST100, NASDAQ, and Bitcoin datasets. To do this, we use the XGBoost method, a well-liked and effective gradient boosting framework [5]. XGBoost are frequently used for numerous prediction tasks, particularly those in the financial industry, usually better represent the temporal patterns and volatility dynamics.

We first preprocessed the datasets by generating lagged rolling variance features as we described in Section 2.3 before we started the model development. These lagged features offer insights on the past price and return volatility of the stocks. By varying the number of lag parameter N and the rolling window size parameter k, we produced a number of lagged rolling variance features and we then calculated variaces of these rolling windows of the logarithmic returns at different time intervals. The generated datasets were used as inputs for training the XGBoost models by feeding both the original features and the engineered lagged rolling variance features.

We evaluated model performances using a grid search and cross-validation tech-

niques by varying model parameters, and we choose the optimum set of parameters that produce the maximum prediction accuracy. These parameters included the rolling window size k and the number of lagged features N, the learning rate, maximum tree depth, number of boosting rounds, and subsample ratio. For each dataset, this procedure is carried out independently to guarantee that the models are adapted to the unique properties of the data. Next, to test the performance of the model with the most optimum parameters, we divided the datasets into training and test sets maintaining the correct temporal order. We then trained a XGBoost model on each dataset using the most optimal parameters chosen for that dataset. The rolling variance of the logarithmic returns served as the target variable for the supervised learning strategy of models which used lagged rolling variance features as input variables. To determine their predictive power, we analyzed the performance of the trained XGBoost models using the mean absolute percentage error (MAPE) score and R^2 . The results of our experiments on our real datasets are shown in Tables 4.6, 4.7, and 4.8. Table 4.9 shows the model performance on the synthetic dataset for comparison purposes.

Table 4.6. XGBoost model performances for BIST100 dataset.

(k, N)	MAPE	R^2
$(k = 14, N = 5)$	0.26	0.8582
$(k = 14, N = 10)$	0.27	0.8581
$(k = 14, N = 15)$	0.27	0.8574
$(k = 20, N = 5)$	0.26	0.8764
$(k = 20, N = 10)$	0.26	0.8758
$(k = 20, N = 15)$	0.27	0.8742

Table 4.7. XGBoost model performances for NASDAQ dataset.

(k, N)	MAPE	R^2
$(k = 14, N = 5)$	0.48	0.7857
$(k = 14, N = 10)$	0.48	0.7861
$(k = 14, N = 15)$	0.49	0.7804
$(k = 20, N = 5)$	0.31	0.8357
$(k = 20, N = 10)$	0.31	0.8359
$(k = 20, N = 15)$	0.30	0.8248

Table 4.8. XGBoost model performances for BTCUSD dataset.

(k, N)	MAPE	R^2
$(k = 14, N = 5)$	0.23	0.9244
$(k = 14, N = 10)$	0.22	0.9218
$(k = 14, N = 15)$	0.22	0.9297
$(k = 20, N = 5)$	0.16	0.9602
$(k = 20, N = 10)$	0.17	0.9611
$(k = 20, N = 15)$	0.17	0.9611

Table 4.9. Model performances for synthetic dataset.

(k, N)	MAPE	R^2
$(k = 14, N = 5)$	0.24	0.5728
$(k = 14, N = 10)$	0.24	0.5660
$(k = 14, N = 15)$	0.24	0.5764
$(k = 20, N = 5)$	0.20	0.4477
$(k = 20, N = 10)$	0.20	0.4490
$(k = 20, N = 15)$	0.20	0.4502

Notice that increasing the rolling window size or the number of lags put into the model improved the performance of the model slightly but most likely, not significantly for BIST100. Again, the model NASDAQ series performed significantly better when we increased the rolling window size. Bitcoin showed a similarity with BIST100 in that the model performance improved slightly better as we increased the window size. However, we observe that increasing the lag terms did not result in significant improvements. The models showed the worst performance on synthetic datasets, as expected. However, curiously, increasing rolling window size resulted in the worse results on synthetic datasets only.

4.3.2.1. Testing autocorrelations for the XGBoost models. In addition to assessing how well the XGBoost models predicted the future variance, we tested the significance of various lagged rolling variance values. One can think of this analysis as the analogue of the ACF and PACF for the XGBoost models. For this purpose we used the feature importance analysis [13].

The feature importance analysis might give us clues into the underlying dynamics and temporal patterns in each dataset by assessing the relative significance of several lagged rolling variance characteristics. We might determine which historical features had a bigger impact on a volatility model by evaluating the relative relevance of the lagged rolling variance features. We plot feature importance values of the best models we selected for each dataset.

First we picked model with window size $k = 20$ and with $N = 5$ lagged features $x_1, x_2, \ldots, x_5$ for the BIST100 dataset. Here, the only significant lagged term for the model is the first lagged feature x_1.

The feature significance plot for the NASDAQ dataset is shown in Figure 4.31. For this dataset, we selected a model with the window size $k = 20$ and with $N = 10$ lagged features. Again, the first lagged feature x_1 has the highest relevance score of the chosen parameters. Additionally, we observe that x_{10} displayed a dramatically lower but still significant feature relevance importance score. This suggests that the lagged rolling variance at a further time point, shown by x_{10}, also makes a considerable contribution to the predictive power of the model. The other significant features x_9, x_5, and x_2 were also significant with relatively low relevance ratings on compare with x_{10}.

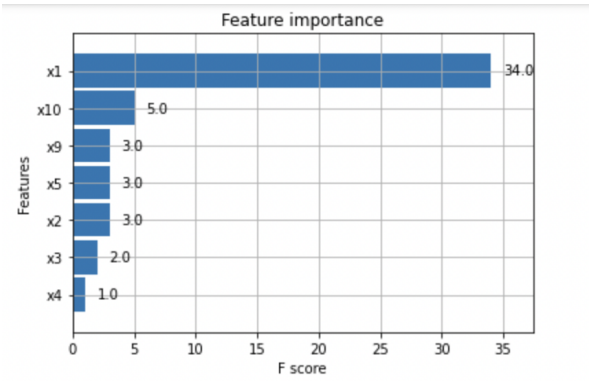

Figure 4.31. NASDAQ XGBoost feature importance.

The feature significance plot for the Bitcoin dataset is shown in Figure 4.32. We selected a model with the window size $k = 20$ and with $N = 10$ lagged features

depending on the MAPE and R^2 results. Again, the first lagged feature x_1 stood out as the most important feature. Additionally, we noticed that x_2 and x_6 had a much smaller but significant importance scores. Although they may have a less substantial effect on the forecast of model than x_1, they nevertheless add to our understanding of the connection between lagged rolling variance and the target variable.

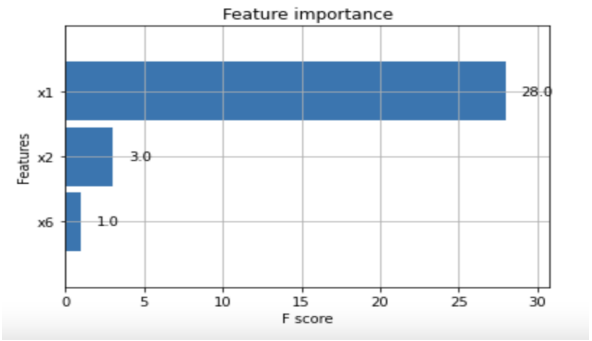

Figure 4.32. BTCUSD XGBoost feature importance.

It is really difficult to theoretically decide which model is the best fit for synthetic dataset. For convenience, we chose the model with parameters $k = 14$ and $N = 15$ depending on R^2 values. The result indicates that the only feature with significant, but comparatively weak, importance is x_1 as expected.

4.3.3. Decision Tree Regression

For our last set of experiments, we replaced XGBoost models with Decision Tree Regression models as an alternative method for developing accurate prediction models for rolling variances. Decision Tree Regression models are well-known machine learning algorithm class and they make predictions using a binary tree structure and a set of decision rules.

We preprocessed the datasets by creating lagged rolling variance features as stated in Section 2.3, using the same rolling window sizes k and number of lagged features N as we did for the XGBoost models in order to decrease complexity, and make comparison feasable. We next use the preprocessed datasets, which include the lagged rolling variance features as input variables, to train the Decision Tree Regression models on

each dataset. Again, the configuration for the training and test split were the same as in Section 4.3.2.

We carried out a rigorous hyper-parameter tuning process to identify the best configurations for the Decision Tree Regression models. As before, we investigated a range of parameter options, including the maximum tree depth, the bare minimum of samples needed to divide a node, and the splitting criterion. Finding the parameter combinations that produce the best prediction accuracy for each dataset is the goal. We evaluated the performance of various parameter combinations and choose the optimum set of parameters using grid search and cross-validation approaches. We assess the performance of models after training using the R-squared (R2) score and the mean absolute percentage error (MAPE) as we did for XGBoost models. The results are shown in Tables 4.10, 4.11, 4.12, and 4.13.

Consistent with what we observed in XGBoost models, we again observe for the decision tree regression models higher R-squared scores and lower mean absolute percentage error values by increasing the rolling window size k across the board. Again as before, increasing the lag parameter N did not impact the performances significantly. The change of performance is most dramatic for the NASDAQ dataset while our models for the BIST100 dataset appears not to respond in increasing the window size. The performance gains for our models on the bitcoin dataset, on the other hand, were modest at best. Again consistent with the XGBoost results, the synthetic dataset showed the weakest model performance for the decision tree regression models, as expected. Moreover, again as expected and as before, the performance of model was unaffected by adjusting the number of lags N, and model performance is slightly affected when the rolling window size k was increased.

4.3.3.1. Testing autocorrelations for the decision tree models. We use the Decision Tree Regression models, just like the XGBoost models, to examine the feature importance of the lagged rolling variance features to test autocorrelations similar to ACF and PACF analysis we performed on ARIMA models. The results of this analysis determines the effect of various lagged rolling variance values in forecasting the inherent variance of the

Table 4.10. Decision Tree Regression model performances for BIST100 dataset.

(k, N)	MAPE	R^2
(k=14, N=5)	0.14	0.8658
(k=14, N=10)	0.14	0.8713
(k=14, N=15)	0.13	0.8889
(k=20, N=5)	0.10	0.9141
(k=20, N=10)	0.10	0.9133
(k=20, N=15)	0.11	0.9024

Table 4.11. Decision Tree Regression model performances for NASDAQ dataset.

(k, N)	MAPE	R^2
(k=14, N=5)	0.18	0.8460
(k=14, N=10)	0.17	0.8598
(k=14, N=15)	0.20	0.7885
(k=20, N=5)	0.08	0.9517
(k=20, N=10)	0.09	0.9269
(k=20, N=15)	0.08	0.9553

Table 4.12. Decision Tree Regression model performances for BTCUSD dataset.

(k, N)	MAPE	R^2
(k=14, N=5)	0.22	0.9198
(k=14, N=10)	0.22	0.9176
(k=14, N=15)	0.23	0.9167
(k=20, N=5)	0.16	0.9334
(k=20, N=10)	0.16	0.9341
(k=20, N=15)	0.16	0.9286

Table 4.13. Decision Tree Regression model performances for synthetic dataset.

(k, N)	MAPE	R^2
(k=14, N=5)	0.25	0.5460
(k=14, N=10)	0.25	0.5380
(k=14, N=15)	0.25	0.5482
(k=20, N=5)	0.22	0.4258
(k=20, N=10)	0.22	0.4281
(k=20, N=15)	0.20	0.4311

dataset at hand. We also analyze the relative relevance of the lagged rolling variance characteristics and plot the feature importance for each dataset.

We picked a model with window size $k = 20$ and with $N = 5$ lagged rolling variance features for the BIST100 dataset, as we did for the XGBoost models. Again, consistent with the XGBoost models, the only statistically significant lagged feature is x_1 for the decision tree models.

For NASDAQ dataset we choose the model with parameters $k = 20$ and $N = 15$ that we chose for the XGBoost models. x_1 has the only importance score. This is again consistent with what we have observed with the XGBoost models. However, this time we found no other lagged variance makes a significant contribution to the rolling variance forecast model.

Finally, for the BTCUSD dataset, we selected the model with the same parameters $k = 20$ and $N = 10$ as in XGBoost models. Again, the only significant contribution comes from the first lagged feature x_1.

4.3.4. A Comparison of XGBoost and Decision Tree Regression

In this part of the section, we compare best models for each dataset.

Table 4.14. A comparison of model performances.

Dataset	Model	Best (k, N)	MAPE	R^2
BIST100	XGBoost	(k=20, N=5)	0.26	0.8764
	Decision Tree	(k=20, N=5)	0.10	0.9141
NASDAQ	XGBoost	(k=20, N=5)	0.26	0.8764
	Decision Tree	(k=20, N=5)	0.10	0.9141
BTCUSD	XGBoost	(k=20, N=10)	0.31	0.8359
	Decision Tree	(k=20, N=15)	0.08	0.9553
Synthetic	XGBoost	(k=14, N=15)	0.24	0.5764
	Decision Tree	(k=14, N=15)	0.25	0.5482

The Table 4.14 compares the results of the model for each dataset, including the NASDAQ index, the BTCUSD cryptocurrency pair, the BIST100 stock exchange index, and a synthetic dataset.

Except for synthetic dataset, Decision Tree Regression is more successful than XGBoost for other three datasets depending on both R^2 values and mean absolute percentage error metrics.

In the best models found for XGBoost and Decision Tree Regression, the variable k is equal to 20 for all. This shows that increasing the k value gives better results for both XGBoost and Decision Tree Regression.

4.3.5. Betweenness Comparison of ARCH, XGBoost and Decision Tree Regression

For the last part of this section we calculate *betweenness* also for XGBoost and Decision Tree Regression models. We choose the best models with parameter k and N for each dataset and model as shown in Table 4.14.

In Section 4.3.1 we predicted variance with ARCH(1,1) model and defined *betweenness* in order to understand how well the predicted variance captures volatility in log returns. In order to compare ARCH, XGBoost and Decision Tree Regression model, we must determine a rolling window size k and autoregression delay parameter N for our ARCH(1,1) models. We can say that $N = 1$, since $p = 1$ in every updated ARCH model. This means we have only one autoregressive term which corresponds to our rolling window number $N = 1$. Unfortunately, since the rolling window is the whole dataset that comes before the current point, there is no fixed length.

Table 4.15 shows a comparison of best models depending on *betweenness* and parameters k,N for each datasets.

Table 4.15. A comparison of betweenness.

Dataset	Model	Best (k, N)	Betweenness
BIST100	XGBoost	(k=20, N=5)	82.69
	Decision Tree	(k=20, N=5)	75.00
	ARCH(1,1)	(N=1)	72 .00
NASDAQ	XGBoost	(k=20, N=5)	80.84
	Decision Tree	(k=20, N=5)	73.94
	ARCH(1,1)	(N=1)	70.69
BTCUSD	XGBoost	(k=20, N=10)	95.00
	Decision Tree	(k=20, N=15)	93.93
	ARCH(1,1)	(N=1)	79.00
Synthetic	XGBoost	(k=14, N=15)	75.42
	Decision Tree	(k=14, N=15)	70.53
	ARCH(1,1)	(N=1)	69.00

Here it is important to note that, betweenness with high value 95 of XGBoost model for BTCUSD dataset does not imply high success. From Figure 4.33, we see that the reason why betweenness is high, predicted variances from XGBoost model have much larger values than actual log returns in some high variance regions.

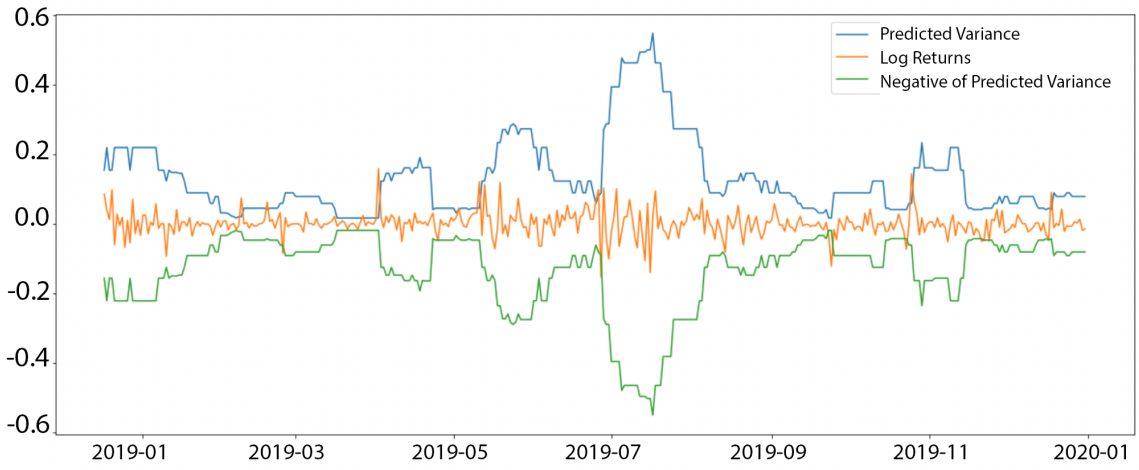

Figure 4.33. XGBoost-BTCUSD betweenness.

5. ANALYSIS

The overall results we obtained in this thesis confirms that logaritmic returns of the datasets BIST 100, NASDAQ, and BTCUSD are almost stationary with no major trends or regular patterns.

We first consider the best-fitting ARIMA models and determined the parameters of the best-fitting ARIMA models for each dataset using a hyper-parameter optimization (Figure 4.2). The only significant parameters for the ARIMA models for the stock exchange data (BIST 100 and NASDAQ) are the moving average (MA) terms. This fact suggests that only the short-term fluctuations have significant impact on future risk in BIST 100 and NASDAQ. On the other hand, for the Bitcoin data the only significant term is the autocorrelation term $p = 2$. This means that the logarithmic return of the BTCUSD series depends on its historical logarithmic return values. This suggests that the persistence and autocorrelation of returns in the bitcoin market are taken into account by the model. Short-term fluctuations in logarithmic returns does not appear to have a substantial impact on forecasting future values according to the absence of any MA or differencing parameters.

We then analyzed autocorrelations using ACF and PACF plots. The BIST100 dataset showed substantial autocorrelations at lags 5 and 7 according to both the ACF and PACF plots. On the NASDAQ dataset, the ACF and PACF plots both showed a strong negative association at delays 2 and 8. This negative correlation suggests that the values at lags 2 and 8 have an impact on the current logarithmic return of the NASDAQ series in a negative way. The BTCUSD dataset study of the ACF and PACF plots found significant autocorrelation at delays 6 and 10, respectively. It is crucial to remember that BTCUSD is a cryptocurrency pair, and unlike other datasets that run for five trading days, its market is open 24/7. For each dataset, these various lag values might correspond to a particular market structure.

The risk profiles displayed by each dataset vary, reflecting the unique volatility patterns of each market or asset. Interesting enough, the parameter values for the best GARCH model are the same across all datasets. This suggests that the GARCH(1,1) model accurately reflects the volatility dynamics present in each dataset due to its constant coefficients across datasets. Table 5.1 shows $alpha[1]$ and $beta[1]$ coefficients for each GARCH(1,1) model.

Table 5.1. GARCH models coefficients.

Dataset	Model	$alpha[1]$	$beta[1]$
BIST100	GARCH(1,1)	0.05	0.93
NASDAQ	GARCH(1,1)	0.034	0.9021
BTCUSD	GARCH(1,1)	0.104	0.876

The GARCH(1,1) model coefficients are $alpha[1] = 0.05$ and $beta[1] = 0.93$ for the BIST 100 dataset. The effect of the past squared residual on the conditional variance of the BIST 100 dataset is represented by the *alpha* coefficient (0.05). The effect of the historical variance on the conditional variance may be seen in the *beta* coefficient, which is 0.93. A higher beta value denotes a greater persistence of historical volatility in BIST 100 return volatility. This suggests that the BIST 100 index has a high level of volatility clustering, when periods of high volatility frequently follow other periods of high volatility.

The GARCH(1,1) model coefficients for the NASDAQ dataset are $alpha[1] = 0.0339$ and $beta[1] = 0.9021$. The alpha coefficient (0.0339) indicates that, when compared to the BIST 100 dataset, the conditional variance of the NASDAQ returns was substantially less affected by past squared residuals. This suggests that recent market shocks have a smaller impact on the volatility of the NASDAQ index.

The GARCH(1,1) model coefficients for the BTCUSD dataset are $alpha[1] = 0.104$ and $beta[1] = 0.8760$. In contrast to the BIST 100 and NASDAQ datasets, the greater alpha coefficient (0.104) indicates that the volatility of the BTCUSD returns is more influenced by recent market shocks or extreme occurrences.

This suggests that the bitcoin market is more susceptible to unexpected shifts or outliers, which can greatly affect its volatility.

The risk profiles of each dataset can be understood through the comparison of the GARCH model coefficients. A higher level of sensitivity to recent market shocks and a comparatively high level of volatility persistence are both seen in the BIST 100 dataset. The NASDAQ dataset shows that recent shocks have a less pronounced effect on volatility and only little persistence. The BTCUSD dataset shows greater reactivity to market shocks and a moderate amount of volatility persistence because it is a cryptocurrency pair with ongoing trading. These results illustrate the distinct risk dynamics and characteristics of each dataset, which are relevant for risk management and forecasting.

It became clear from the residual analysis that our models were unable to identify certain patterns in the datasets. Significant autocorrelation was seen in the residuals for the BIST100 dataset at lags 2 and 5, pointing to the existence of underlying patterns that our ARIMA model was unable to detect. Similar to this, the residuals from the NASDAQ dataset showed high autocorrelation at lags 5 and 8, indicating dependencies that the model did not sufficiently account for. Additionally, the residuals from the BTCUSD dataset showed strong autocorrelation at lags 5, 6, and 10, indicating enduring patterns or trends that our model was unable to fully capture. The limits of our ARIMA models in identifying and capturing all the intricacies included in the datasets are highlighted by these findings.

QQ-plot analysis of each dataset showed that the tails of residuals did not follow the normal distribution. In particular, the QQ-plots for the BIST100, NASDAQ, and BTCUSD datasets showed notable outliers in the ends, pointing to thicker tails in the residual distribution than was anticipated. These deviations imply the presence of extreme values or outliers in the residuals, which may have an impact on the performance and assumptions of the model.

6. CONCLUSION AND FUTURE WORK

In this thesis we examined the volatility dynamics of the BTCUSD cryptocurrency, the NASDAQ index, and the BIST 100 stock exchange index. Instead of the plain price data, we used the log-returns of each time series data. We used standard volatility analysis such as ARCH and GARCH, as well as suitable machine learning methods on the point cloud data we generated from the delay embedding of the log return data. Our main hypothesis in this thesis was that if a model sufficiently explained the volatility of the time series, the residual data (real values minus the model prediction) should be as close to white noise as much as possible. However, the statistical analysis of the residuals showed considerable autocorrelations and departures from predicted distributions. This indicates that even though our models captured some of the underlying patterns, there are still some phenomena remained unexplained in the signal that need to be captured.

By offering insights into volatility dynamics and risk profiles, this thesis makes a contribution to the field of financial market analysis. In our case, the GARCH(1,1) model consistently captures the volatility dynamics in all datasets, albeit with varying magnitudes of coefficients, indicating unique risk dynamics for each market or asset. The BIST 100 index displayed notable and persistent volatility and was extremely responsive to recent market shocks. The NASDAQ index, in comparison, showed a slightly lower vulnerability to similar shocks. In contrast, the BTCUSD dataset showed substantially higher volatility compared to all the assets we looked at when it comes to abrupt movements.

The volatility analyses and conclusions we made in this thesis have applications for risk management and decision-making processes in the financial and cryptocurrency markets, and may serve as a solid foundation for further statistical studies. One may expand our study by using more sophisticated modeling strategies to better capture the underlying dependencies and patterns in the datasets such as neural networks

or ensemble models. Further research might be conducted to analyze the stochastic structure of the series we considered using both endogeneous and exogeneous variables to better understand the volatility dynamics and risk profiles across various markets. One can also broaden our work by including additional financial and cryptocurrency datasets.

REFERENCES

1. Box, G. E., G. Jenkins, G. Reinsel and G. Ljung, *Time series analysis. forecasting and control. 2.Ed.*, Holden-Day, San Francisco, 1976.

2. Engle, R. F., "Autoregressive conditional heteroscedasticity with estimates of the variance of United Kingdom inflation", *Econometrica*, Vol. 50, No. 4, p. 987, 1982.

3. Bollerslev, T., "Generalized autoregressive conditional heteroskedasticity", *Journal of Econometrics*, Vol. 31, No. 3, p. 307–327, 1986.

4. Glosten, L. R., R. Jagannathan and D. E. Runkle, "On the relation between the expected value and the volatility of the nominal excess return on stocks", *The Journal of Finance*, Vol. 48, No. 5, p. 1779–1801, 1993.

5. Chen, T. and C. Guestrin, "XGBoost: A Scalable Tree Boosting System", Vol. abs/1603.02754, 2016, `http://arxiv.org/abs/1603.02754`.

6. Breiman, L., J. H. Friedman, R. A. Olshen and C. J. Stone, *Classification and Regression Trees*, Chapman & Hall, New York, 1984.

7. Zhang, Y., "Stock price prediction method based on XGboost algorithm", *Atlantis Highlights in Intelligent Systems*, p. 595–603, 2022.

8. Wold, H., *A study in the analysis of stationary time series*, Almqvist & Wiksell, Stockholm, 1954.

9. Dickey, D. A. and W. A. Fuller, "Distribution of the estimators for autoregressive time series with a unit root", *Journal of the American Statistical Association*, Vol. 74, No. 366, p. 427, 1979.

10. Van Rossum, G. and F. L. Drake, *Python 3 Reference Manual*, CreateSpace, Scotts

Valley, CA, 2009.

11. PyPI, "pmdarima", `https://pypi.org/project/pmdarima/`, 2023, accessed on June 10,2023.

12. Bashtage, "Arch Models in python", `https://github.com/bashtage/arch/`, 2023, accessed on June 10,2023.

13. xgboost, "XGBoost Docs", `https://xgboost.readthedocs.io/en/stable/`, 2023, accessed on June 10,2023.

14. Anon, D. and H. Singh, "Estimating and forecasting volatility using ARIMA MODEL: A study on NSE, India", *Indian Journal of Finance*, Vol. 13, No. 5, p. 37, 2019.

15. Nakamoto, S., "Bitcoin: A Peer-to-Peer Electronic Cash System", Retrieved from `https://bitcoin.org/bitcoin.pdf`, 2008.

16. Timofeev, R., *Classification and Regression Trees (CART): Theory and Applications*, Master's Thesis, Humboldt University, 2004.

17. Tsay, R. S., "Analysis of Financial Time Series", *Wiley Series in Probability and Statistics*, 2005.

18. Brockwell, P. J. and R. A. Davis, *Introduction to time series and forecasting*, Springer, New York, 2016.

19. Hamilton, J. D., *Time series analysis*, Princeton University Press, Princeton, NJ, 1994.

20. Enders, W., "Applied econometric time series.", *Journal of the American Statistical Association*, Vol. 90, No. 431, p. 1135, 1995.