Hochschule für Wirtschaft und Umwelt Nürtingen-Geislingen

Decoding Ethereum's Market with Algorithmic Trading: A Path to Outperforming Passive Investment?

To Prof. **Bruhn Pascal**

Amelia Cecchini (4106338)

Giulio Cattoni (4106339)

Bhavin Wadhvana (4100316)

Abhay Gopinatha Kurup (4100309)

Index

Abstract

This study examines a custom-built trading algorithm for Ethereum, a popular digital currency. Basic statistical models are applied to predict future price movements of Ethereum. The main aim is to determine whether the trading algorithm can outperform a simple, long-term investment strategy in Ethereum. The results of this algorithm are compared to the outcomes of a passive investment to assess its effectiveness. Additionally, straightforward statistical tests are used to validate our hypothesis. ARIMA is applied to model the mean in Ethereum data, while GARCH is used to model the volatility in financial data.

This research is intended to shed light on whether algorithmic trading can offer better returns in the volatile digital currency market, focusing on the cryptocurrency Ethereum as a case study.

1. Introduction

The cryptocurrency market has experienced exponential growth over the past decade, with trading increasingly being dominated by sophisticated algorithmic strategies. This shift towards algorithmic trading underscores the need to evaluate the effectiveness and potential for outperformance of these strategies over traditional passive investment approaches. Ethereum, as one of the leading cryptocurrencies, presents a unique opportunity to explore these dynamics. This paper tries to develop and test a trading algorithm tailored to Ethereum's market characteristics, offering insights on forecasting future returns and volatilities and demonstrate the correlation between the active strategy, based on the trading algorithm, and the passive investment, based on the buy and hold strategy.

Overview of Ethereum

Ethereum, launched in 2015, is a decentralized platform that enables smart contracts and decentralized applications (DApps) to be built and operated without any downtime, fraud, control, or interference from a third party. Unlike Bitcoin, which is primarily a digital currency, Ethereum's

Ether (ETH) serves as a fuel for operating the distributed application platform. It provides a flexible and robust framework for developers, contributing to its widespread adoption and positioning it as a crucial player in the cryptocurrency space. The Ethereum network has been at the forefront of the blockchain revolution, facilitating the creation of a new ecosystem of decentralized finance (DeFi) and non-fungible tokens (NFTs), further diversifying its use cases and value proposition. Ethereum's versatility and its central role in the digital currency market provide a compelling backdrop for this study[1].

Market Landscape

The cryptocurrency market has seen significant capital inflows, leading to a substantial increase in market capitalization and trading volumes across various digital currencies. Ethereum has demonstrated remarkable growth in terms of market cap, prices, and trading volumes, reflecting its increasing importance and the widespread investor interest in its underlying technology. Ethereum boasts the second largest market cap, second only to Bitcoin. It had a market cap of $ 146.45 billion and a daily volume of 3.01 billion[2].

The cryptocurrency attained an all-time high price of $ 4,865.57 on 08 November 2021[3]. The dynamic nature of Ethereum's market, with its significant price fluctuations and speculative interest, makes it an ideal candidate for algorithmic trading strategies aiming to capitalize on short-term movements and patterns.

Ethereum data is collected from Coingecko.com and the data is retrieved for the specified time period, from 01.01.2022 to 01.01.2024.

The daily returns of Ethereum used in this study are logarithmic returns, computed as the logarithmic whose argument is the price of t over the price of t-1: $R_t = \ln(P_t / P_{t-1})$.

[1] https://www.kucoin.com/price/ETH
[2]https://coinmarketcap.com/currencies/ethereum/historical-data/
[3] https://www.statista.com/statistics/806453/price-of-ethereum/

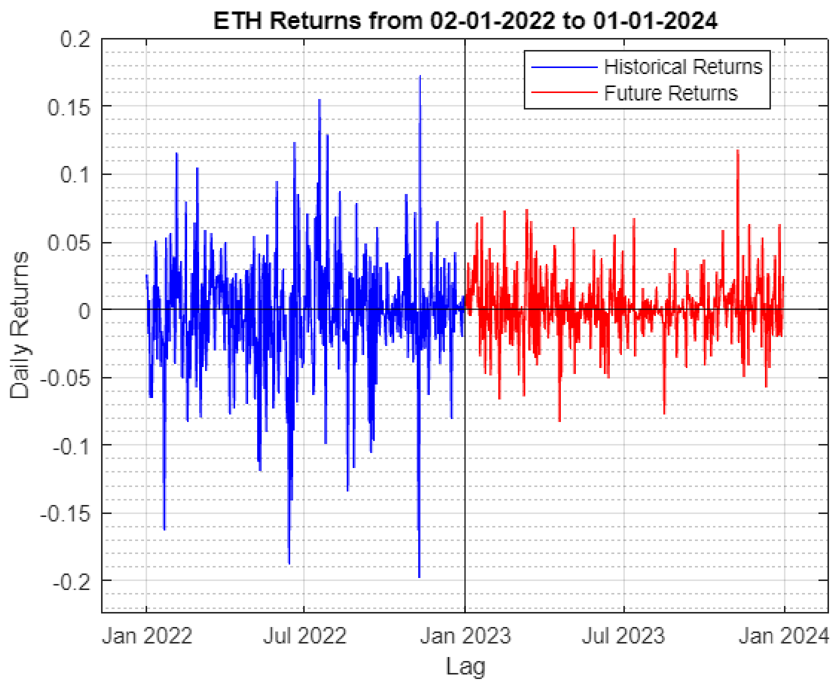
Overall performance of Ethereum is checked by finding Average Daily return, volatility, max gain / drawdown, Value at Risk, Expected Shortfall.

Table 1: Descriptive statistics

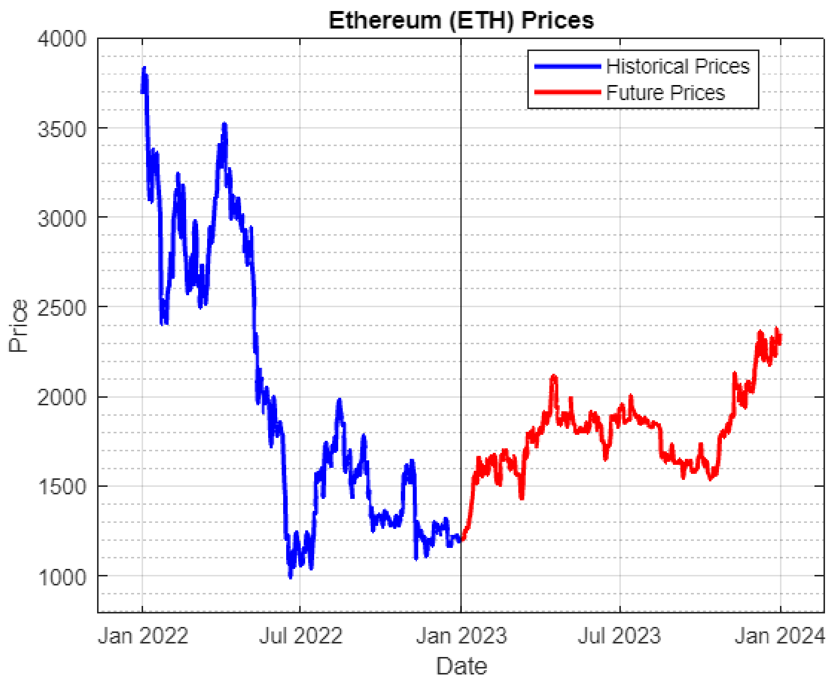
	Variables	ETH
1	Mean	-0.0006
2	Var	0.0013
3	Stan. Dev.	0.0367
4	Kurt	7.7610
5	Skew	-0.4431
6	VaR95	0.0581
7	ES95	0.0942
8	VaR99	0.1172
9	ES99	0.1577
10	Min	-0.1975
11	Max	0.1726
12	Max Gain	0.0396
13	Max Drawdown	0.4337

Table 1 presents the descriptive statistics of the log daily return series of Ethereum. Mean for the mentioned period is negative which is -0.0006 and standard deviation is 0.0367. A skewness value of -0.4431 for stock log daily return data indicates that the distribution of the returns is negatively skewed. A kurtosis value of 7.7610 suggests that the distribution has fatter tails and is more peaked than a normal distribution. A Value at Risk (VaR) at a 95 % confidence level for stock log daily return data means that there is a 5 % probability that the daily loss on the stock will perform worse than - 0.0581. Expected shortfall at 95 % would indicate that the losses that exceed the VaR level at 99% will have an average value of - 0.0942. The value of the investment or portfolio experienced a decline of 43.37% from its peak to its lowest point before recovering, while the max gain return is equal to 3.96 %.

The data of Ethereum are split into two periods: from 01.01.2022 to 31.12.2022 there are historical data, while from 01.01.2023 to 01.01.2024 there are future data.



The above chart shows the historical returns of Ethereum with the future returns over the specified time periods. The blue line represents historical returns, and the red line represents future returns.



This chart shows the historical prices of Ethereum with the future prices over the specified time periods. The blue line represents historical prices, and the red line represents future prices.

2. Methodology

To apply ARMA Model, the historical returns need to be IID and stationary and this is checkable with following tests.

The usage of the logarithmic returns is because they tend to be closer to stationarity compared to stock prices. This does not mean that the stationarity can be justified only by using logarithmic returns. In fact, to check the stationarity the case study uses the Augmented Dickley-Fuller (ADF) Test. The result of this test is 1 indicating a p-value greater than 5% affirming that the historical returns are not stationary.

First, the identification of local peaks and troughs is made. The function gives 1 if a peak or trough is found, otherwise it gives 0. A mean of 0.5344 and a variance of 0.2495 describe the distribution of peaks and troughs of the ETH historical returns. If the distribution was normally distributed, the mean should be around 0.33 and the variance should explain the frequency of how these peaks and troughs differs from the mean. With the values obtained, the distribution has more peaks and troughs than the normal one and they are clustered in periods.

Second, two tests are conducted to check the randomness of the historical returns. The Turning Points Test checks how many times there is a change in the direction of the returns. The left part indicates that the number of turning points should be around two third of the return series' length for a random time series. The right part of the inequation represents the significance level of 95% percentile of the normal distribution.

$$\left|\sum(historical\ returns) - \frac{2}{3}(length(historical\ returns)\right| > 1.96 \times \sqrt{\frac{8 \times (length(historical\ returns))}{45}}$$

If this inequation is true, it means that the returns are not IID and it is confirmed by this case study.

On the other hand, the Runs Test is performed to check another aspect of the randomness of the distribution. Particularly it evaluates the length and the number of positive and negative returns' value than the median. It gives a value of 0, indicating that the numbers and the distribution of these runs are consistent with a normal random run.

Even if these tests are concerned both on the randomness of the returns, the two observed results can coexist because they analyse different aspects of the randomness. In conclusion, the median of the distribution is around 0, but there are different patterns for the turning points.

The following tests are conducted on the residuals of the historical returns. The residuals are calculated as the difference between the historical returns and their mean. These tests check if the residuals of the time series are *white noise*, pointing out the residuals don't show autocorrelation or heteroskedasticity.

The Ljung-Box Q-test analyses if the residuals are significantly autocorrelated. If the null hypothesis is rejected, it means that there are patterns in the data that show autocorrelation between the returns. In this case study the null hypothesis is accepted, and it means that there is no-significant autocorrelation between the residuals.

Another performed test is based on the Ljung-Box Q-test, but it takes the squared residuals, to check if there is an autocorrelation structure between the residuals' distribution. If the null hypothesis is rejected, it can be assumed that the variance of the returns changes during the considered period, which it could influence the accuracy of the model. In this case study, the null hypothesis is rejected, showing a significative change in the variance over time.

The last conducted test is ARCH test. This shows if the residuals have ARCH effects due to conditional heteroskedasticity. If the null hypothesis of the ARCH effects' lack is rejected, it means that the variance of the residuals is not constant over time. In this case study, the null hypothesis is rejected, showing ARCH effects in the residuals.

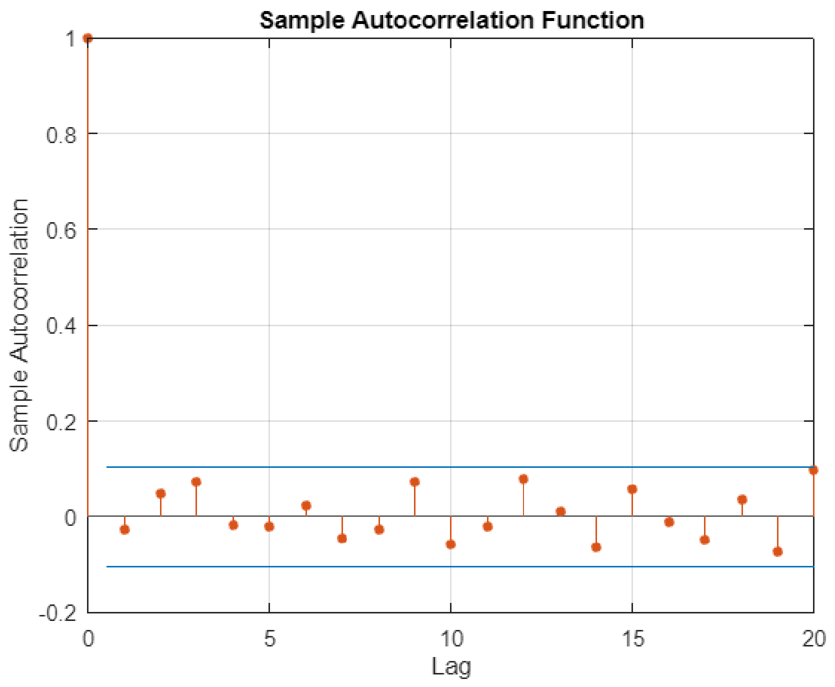
These results show that the ETH historical returns are not IID, so the ARMA model cannot be applied to predict future returns, but it is needed a more sophisticated one.

ARIMA model

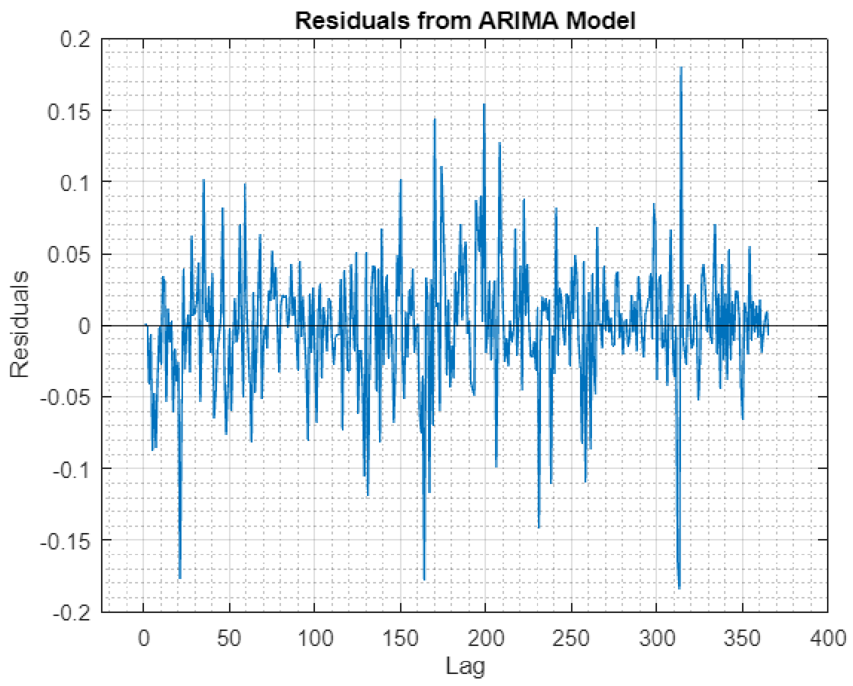
To solve this problem, ARIMA model is used to calculate the conditional mean. The chosen ARIMA model is $ARIMA(1,1,2)$, which uses an autoregressive term of order 1 ($p = 1$), a first-order differentiation ($D = 1$) and two moving average terms ($q = 2$) to predict the future values of a time series.

Once the constants of the model are defined, there is the estimation of the parameters for the historical returns. The AR term is equal to -0.941415, the first MA term is 0.102739 and the second MA term is -0.962824. To check if the model is correct for the data, the epsilons are calculated, and the following tests are conducted:

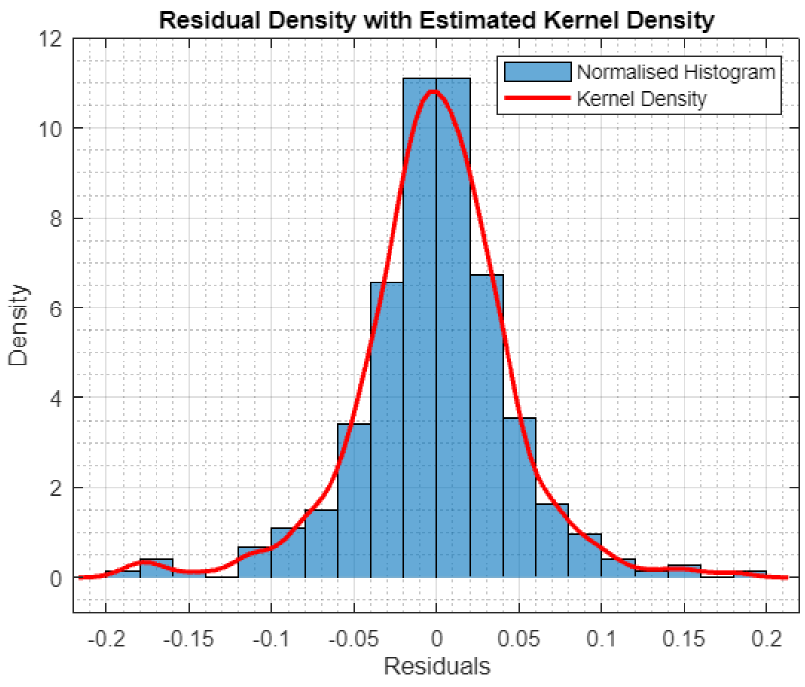
- **Autocorrelation between the residuals** through *autocorr* function and Ljung-Box Q-test. This test gives a true result. So, as can be seen also from the figure below, the residuals don't show significant autocorrelation.



- **If the residuals have mean 0**. And, as the figure below shows, they have.



- Calculate the **density of the residuals**. The histogram shows that the residuals have a normal density and follow the Kernel density.



It is possible to conclude that the $ARIMA(1, 1, 2)$ model is correct and fits our data.

GARCH model

The $GARCH\ (1, 1)$ model is chosen and estimated on the residuals of the $ARIMA(1, 1, 2)$ model. This type of model is used to capture the volatility clustering phenomenon commonly observed in financial time series, where high volatility events tend to cluster together.

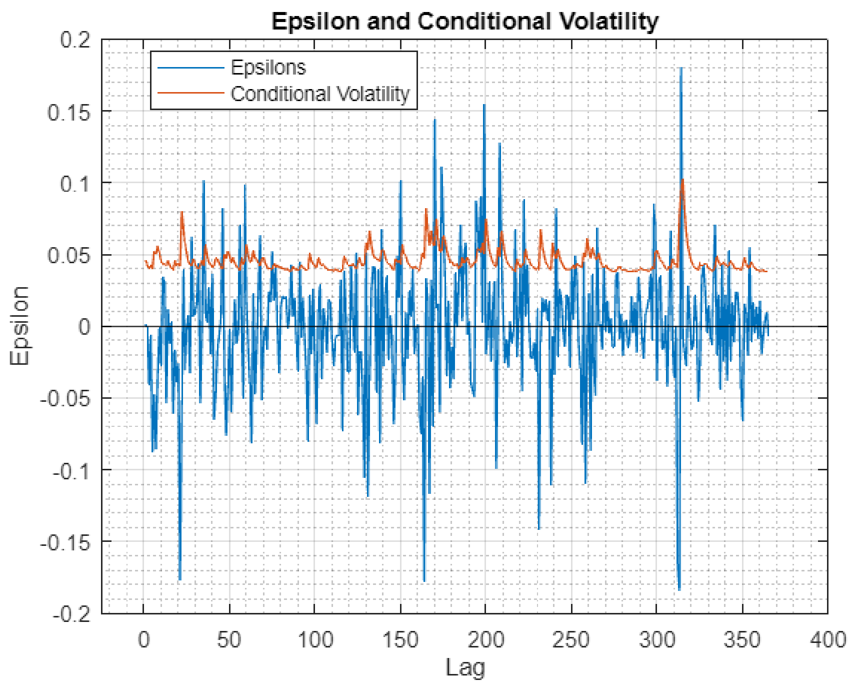
The model parameters (alpha, beta, omega) determine how the past value of the series and the past volatility affect the current volatility.

After estimating the GARCH model parameters, the code derives the conditional variance (and hence volatility) of the time series.

The script simulates a time series based on predefined GARCH parameters (alpha, beta, omega), creating a controlled environment in which is observable the behaviour of the GARCH model. This part involves preallocating space for variables, initializing the series, and then iterating over the time to simulate the series.

The final section of the GARCH model code fits to the simulated data using maximum likelihood estimation (MLE), a common approach for estimating the parameters of statistical models. This optimization uses the *fmincon* function which enforces constraints to keep the parameters within reasonable limits. The estimated parameters are then compared to the actual ones used in the simulation and those estimated by MATLAB.

The figure below provides an effective illustration of the dynamic nature of volatility in financial time series data. The periods where the residuals (Epsilons) have large deviations from zero correspond to periods of increasing conditional volatility, reinforcing the concept that volatility is not constant over time but rather reacts to market movements.



Combined ARIMA and GARCH model

Once the case study estimated the ARIMA and GARCH, to get the best fitting model is necessary mix the parameters given by those two.

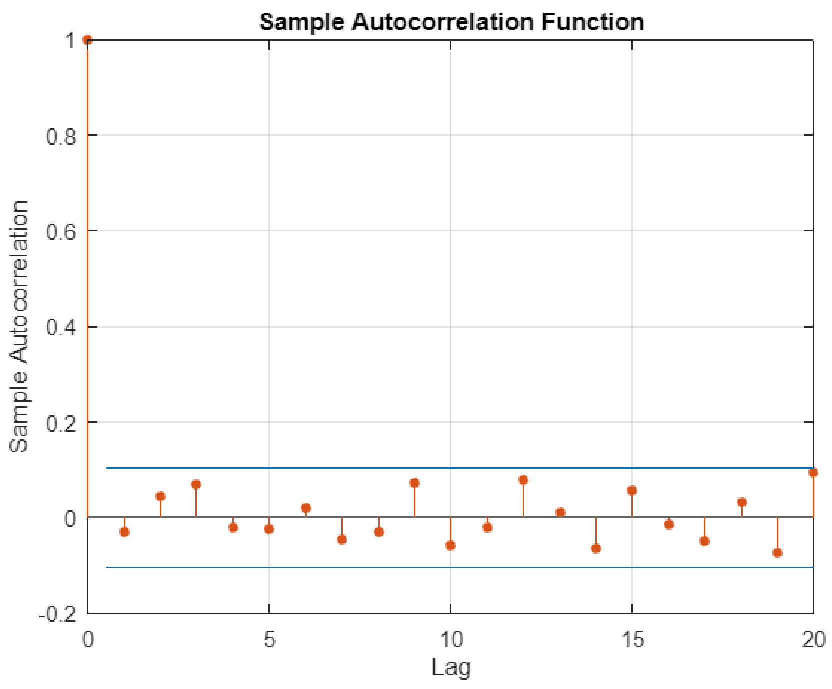
The combination of ARIMA and GARCH models is particularly useful for financial time series analysis for several reasons. GARCH model can shape volatility changes (conditional heteroscedasticity) commonly observed in financial markets. By combining them, the averaging and volatility processes can be modelled simultaneously, allowing for more accurate predictions.

In addition, understanding volatility dynamics (through the GARCH component) and returns trends (through the ARIMA component) provides a more comprehensive picture of market risk.

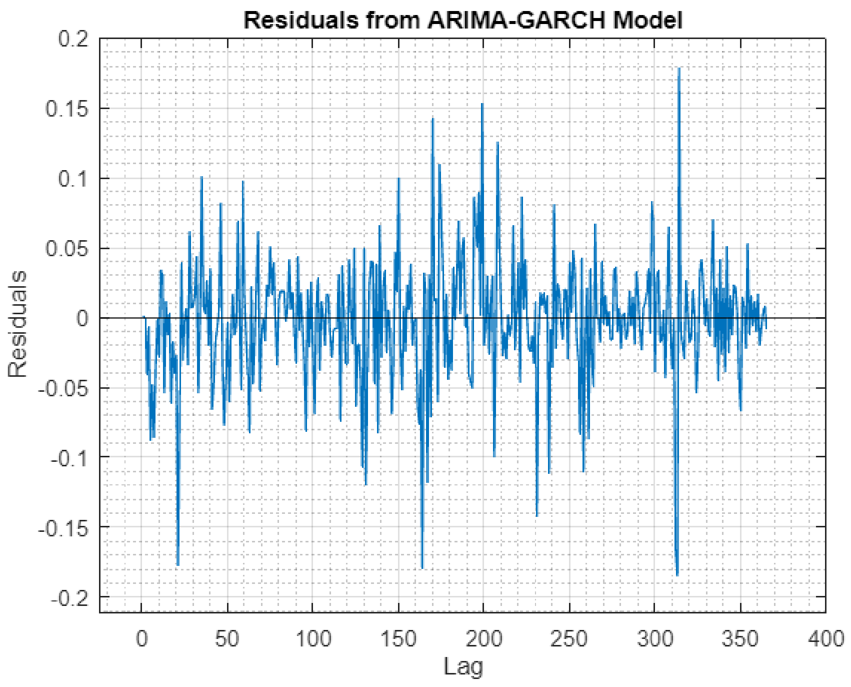
To see if the model is correct for the data, the same tests as for ARIMA are conducted to check the autocorrelation, the mean, and the density of the residuals.

As can be seen below, all hypotheses are respected.

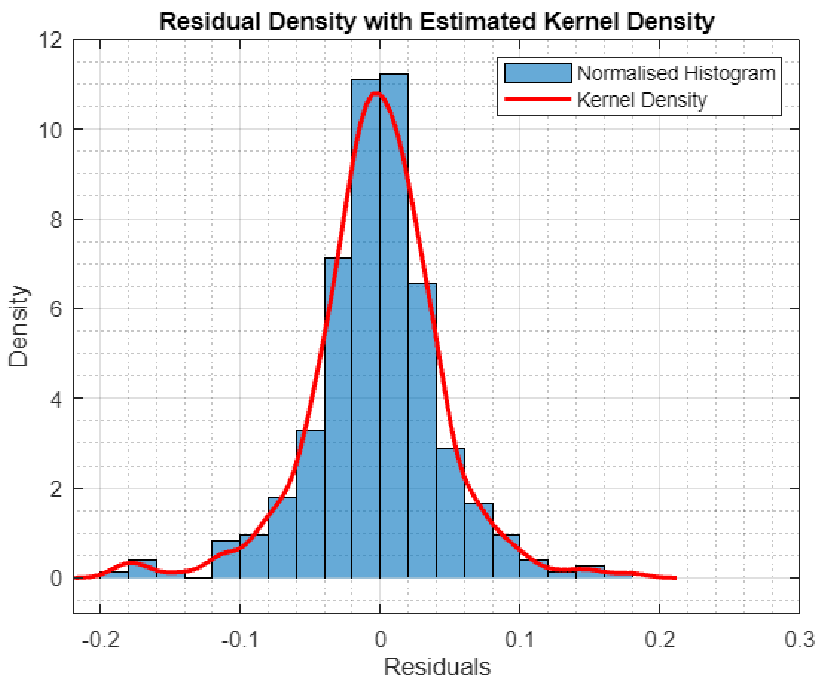
- **Autocorrelation between the residuals** through *autocorr* function and Ljung-Box Q-test. This test gives a true result. So, as can be seen also from the figure below, the residuals don't show significant autocorrelation.



- **If the residuals have mean 0**. And, as the figure below shows, they have.



- Calculate the **density of the residuals**. The histogram shows that the residuals have a normal density and follow the Kernel density.



The formula used to forecast returns and variance based on this model are the following:

$$r_{t+1} = \omega_{ARIMA} + r_t + \phi \times \nabla r_{t-1} + \theta \varepsilon_t + \theta \varepsilon_{t-1}$$

$$\sigma_t^2 = \omega_{GARCH} + \alpha\varepsilon_{t-1}^2 + \beta\sigma_{t-1}^2$$

Linear Regression

A linear regression is used to determine how to variables are related. Independent variables (X) are used to predict the dependent variable (Y). The relation of X and Y is measured by their Beta, as the degree of change in outcome for one unit of change in the independent variable.

Once that beta1 and beta0 are estimated, it is possible to calculate the expected values of the dependent variable.

After that it is necessary to back test the linear regression to check the goodness of fit. When the variance of Y is as small as possible, it means that the linear regression fits well the data. To do that the Explained and Residual variances are calculated to obtain the Total variance as a sum of the two. At this point it is possible to obtain the R-squared: "a measure of the degree to which variations in the dependent variable can be explained in terms of variations in the independent variables. Its value is between 0 and 1. If the regression model exactly explains the data, it is 1 and if it explains none of the data, it is 0"[4].

The Mean Squared Error and Standardized Mean Squared Error are calculated to verify if the two variables are linearly related. Then it's possible to obtain the t-statistical value and the p-value to do the T-test for significancy.

To check if the p-value is correct, it is used the *fitlm* command of MATLAB.

3. Analysis of the code

The analysis and the explanation of the MATLAB code is in Appendix A.

[4] https://www.oxfordreference.com/

4. Result

This study implemented an ARIMA-GARCH modelling approach to forecast financial returns and assess trading strategy efficacy. The ARIMA model was calibrated to predict the next day's return based on historical data, incorporating autoregressive (AR) and moving average (MA) parameters, as well as differencing of the series to achieve stationarity.

```
ARIMA (1,1,2) Model (Gaussian Distribution):


                 Value      StandardError    TStatistic      PValue  
              __________    _____________    __________    __________


Constant      -6.348e-05     0.00013168       -0.48207        0.62976
AR {1}          -0.94142       0.032827        -28.678     7.2421e-181
MA {1}          0.010274       0.021494        0.47799        0.63266
MA {2}          -0.96282       0.018799        -51.216              0
Variance       0.0021214     0.00010387         20.423     1.0466e-92
```

Based on their extremely low p-values and highly significant t-statistics, the AR (1) and the MA (2) variables, and variance terms in this ARIMA (1,1,2) model seem to be highly important overall. On the other hand, there is no statistical significance for the constant, MA (1). These findings shed light on the time series' dynamics and can help determine which ARIMA model to use and how to interpret it. The significant variance indicates that the model has captured a substantial amount of the variability in the data.

The GARCH model was employed to estimate time-varying volatility, leveraging its capability to model conditional variances.

```
    GARCH (1,1) Conditional Variance Model (Gaussian Distribution):

                  Value       StandardError    TStatistic      PValue
               __________    _____________    __________    __________


    Constant   0.00063475      0.00027642         2.2964      0.021655
    GARCH {1}     0.54971         0.16303         3.3718    0.00074683
    ARCH {1}      0.15332        0.055046         2.7854     0.0053463
```

GARCH (1,1) model accounts for both recent squared returns (alpha) and the persistence of past volatility (beta). The constant term (omega) sets the baseline volatility level. These parameters collectively help model time-varying volatility in financial data. These results suggest that all three parameters (constant, GARCH (1), and ARCH (1)) are statistically significant in the GARCH model, as indicated by their t-statistics and p-values. This indicates that both autoregressive and conditional heteroskedasticity components are important in modelling the volatility of the time series data.

Parameters such as alpha, beta, and omega are initialized based on conventional assumptions, allowing the model to capture the persistence in volatility and the impact of past squared shocks on current variance.

	Omega	**Alpha**	**Beta**
1 True	0.0200	0.0500	0.8500
2 Estimated	0.0002	0.0402	0.9331
3 Matlab	0.0002	0.0408	0.9325

Omega represents the baseline or ambient volatility. In other words, it's the minimum level of volatility even in the absence of any shocks or news. The estimated value is significantly smaller than the true value, suggesting that the model predicts very low baseline volatility. Alpha captures the immediate impact of past squared returns (volatility shocks) on the current volatility. A higher alpha indicates that recent shocks have a stronger effect on volatility. The estimated alpha is slightly lower than the true value, implying a moderate responsiveness to past shocks. Beta represents the persistence of volatility. It measures how past volatility affects future volatility. A higher beta indicates stronger persistence. The estimated beta is close to the true value, suggesting that volatility tends to persist over time. It seems that the estimated values are relatively close to the true values, indicating a reasonably accurate estimation process.

Trading Algorithm

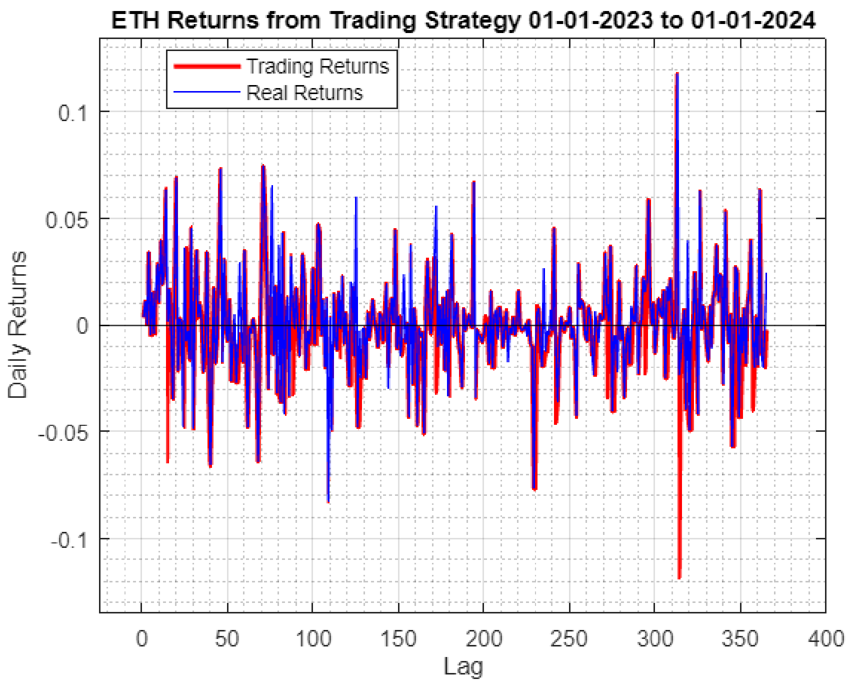
In the trading simulation, a decision-making algorithm is incorporated, inducing buy or sell actions predicted on the forecasted returns. The trading simulation executes with a starting capital base of $ 10,000, engaged in buy and sell decisions over the course of one year, calibrated weekly to reflect the evolving return forecasts. An investor, that uses this trading algorithm, presents a moderate propensity to take risks as evaluated by the Risk Attitude value (0.5114). This value is also increased by the constraint that, every time that the algorithm predicts a sale, all the shares in the portfolio are sold. The trading strategy results in a new cash balance of $ 785.45 and a reported loss of $ - 798.38, with the final portfolio value standing at $ 7,668.48, representing a -23.32% return on the initial investment. The active strategy's performance is marked by moments of insufficient funds to further buy shares, as well as occasions where there are not enough shares to sell, reflecting the inherent constraints and risks of active trading. In contrast, the passive strategy, devoid of such frequent transactional decisions, capitalizes on the market's overall upward trend without incurring the costs and risks associated with active trading. It results in a final portfolio value of $ 18,761.86, a substantial increase from the initial investment, yielding an 87.62% return.

In the comparative analysis of Active and Passive investment strategies, we find distinct differences in performance and risk profiles as summarized in the table 2 below.

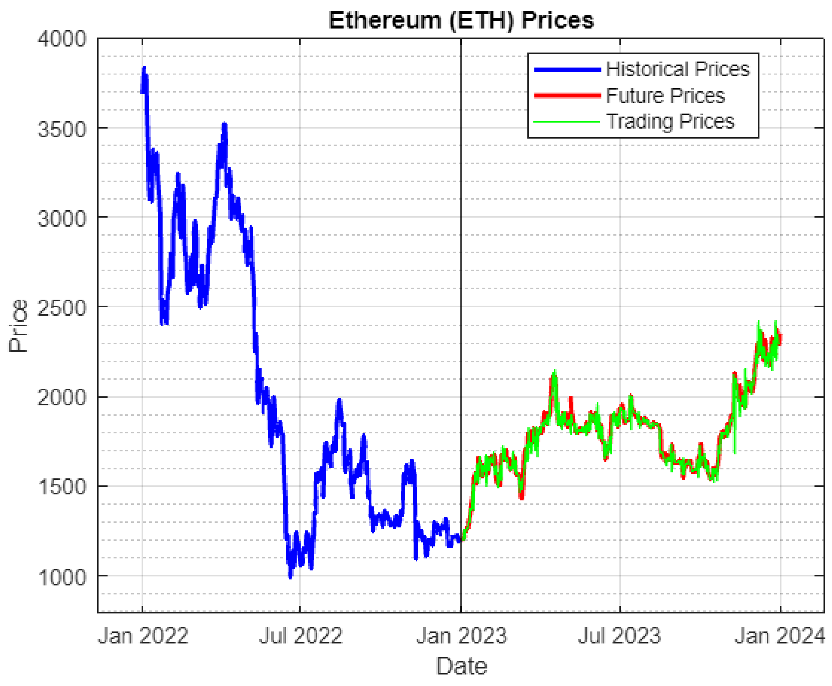
Table 2: Active vs. Passive Strategies

	Variables	Active	Passive
1	Mean	-0.0024	0.0019
2	Var	0.0007	0.0006
3	Stan. Dev.	0.0261	0.0244
4	Kurt	5.5359	5.2958
5	Skew	0.0737	0.3493
6	VaR95	0.0472	0.0406
7	ES95	0.0611	0.0525
8	VaR99	0.0751	0.0629
9	ES99	0.0885	0.0723
10	Min	-0.1181	-0.0827
11	Max	0.1181	0.1181
12	Max Gain	0.2538	0.6862
13	Max Drawdown	0.1990	0.1119

In conclusion, the active strategy does not offer the possibility of higher gains and with significantly higher risk and the potential for losses, as evidenced by negative mean returns and a high maximum drawdown. In contrast, the passive strategy provides more stable returns, as indicated by its positive mean returns and lower risk metrics.



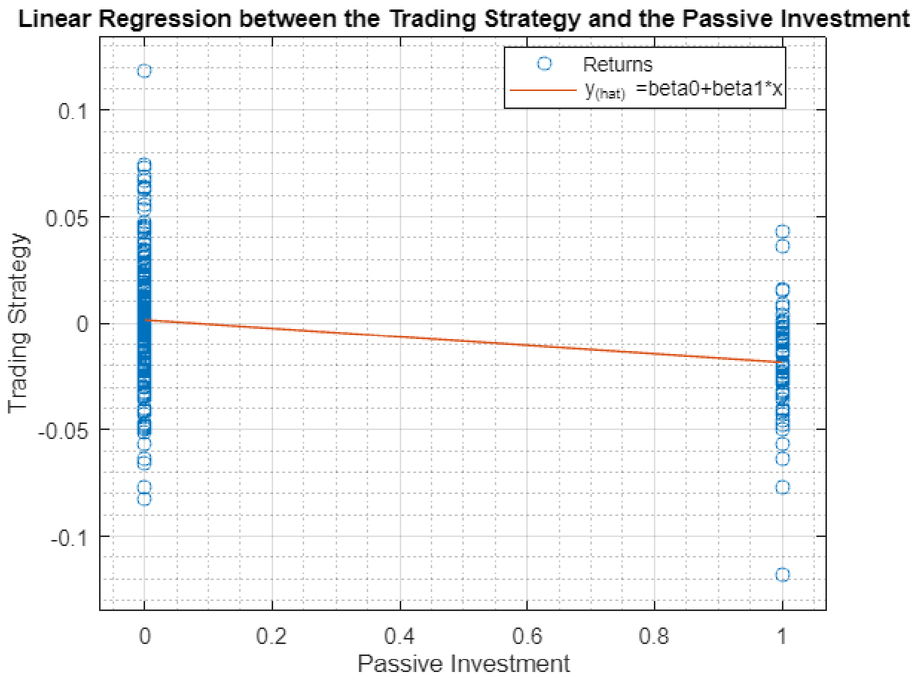
The graphical depiction above offers a compelling visual comparison between forecasted and actual daily returns of the Ethereum (ETH) trading strategy from January 1, 2023, to January 1, 2024. The blue line, indicating real returns, displays the volatility and unpredictability inherent in cryptocurrency markets. The red line, representing trading returns, appears to have periods of close correlation as well as divergence from the actual returns, suggesting that while the model has some predictive power, it's not fully able to capture the market's volatility. The zero line is a baseline indicating whether the predictions over or under-estimate the market performance.



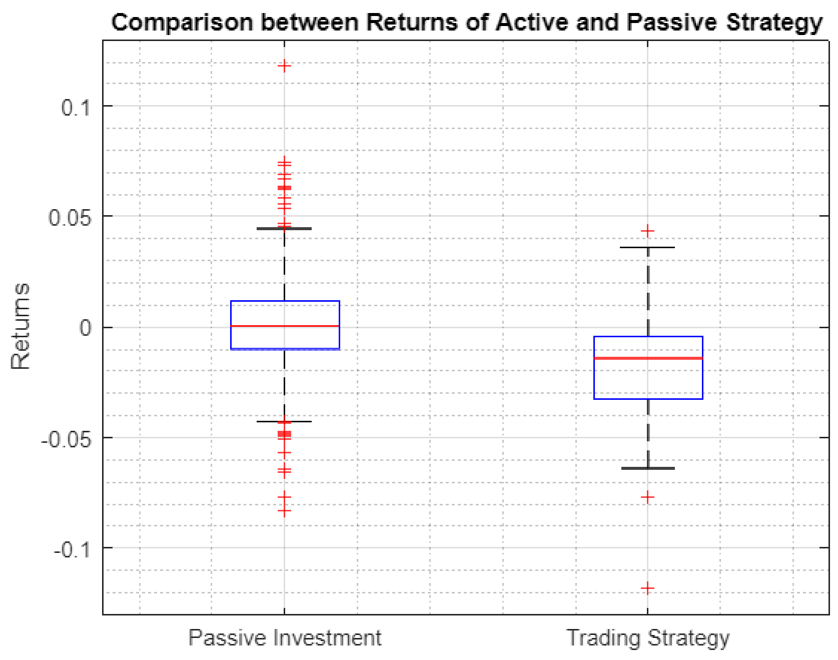
The graphical illustration depicted above shows the prices of Ethereum. The blue line is the same for Active and Passive strategies because it represents the historical prices of the cryptocurrency. The red line figures the ETH real prices in 2023 while the green line displays the path of the prices used for the trading strategy. Despite the final value of the trading strategy is negative, the prices trend follows with high accuracy the trend of the real prices, showing that the model fits well the data.

Linear Regression

The graph depicting linear regression between the trading strategy and the passive investment in the Ethereum market demonstrates a negative linear relationship, with a correlation coefficient of -0.30354. This correlation coefficient represents a mild inverse association between the risk-adjusted performance of the trading strategy and the passive investment approach. Specifically, the negative slope (beta1) of -0.0199 in the regression equation suggests that the active strategy is negatively correlated with higher returns and positively with the lower ones. The distribution of data points around the regression line reflects a varied performance of the trading strategy when benchmarked against the passive investment, indicating that higher passive investment returns do not consistently translate to higher returns in the trading strategy. This suggests that the active strategy overestimates the positive movements of the market compared to a passive approach. The study's findings, emphasize the trading strategy's relative underperformance, suggesting that a passive investment might be a more reliable approach in terms of risk-adjusted returns in the Ethereum market.



The boxplot graphically summarizes the distribution of returns for both the active strategy and passive investment within the context of the Ethereum market. The median of returns for both strategies is approximately zero, but the trading strategy shows a wider interquartile range (IQR), indicating greater variability in returns compared to the passive strategy. Furthermore, the presence of outliers, particularly on the negative side, is more pronounced for the trading strategy. This suggests that while there may be potential for higher returns in active trading, the risk of significant losses is also greater.



The wider spread of returns for the active strategy, as illustrated by the boxplot, supports the findings of higher risk compared to a passive approach. It is also consistent with the linear regression analysis that indicated a mild negative relationship between the performances of the two strategies. In contrast, the passive investment strategy shows a more compact IQR, suggesting a more consistent, even if potentially less lucrative performance which is in line with our results.

Hypothesis Testing

The assumptions for this study are as follows.

H_0: There is no significant difference in average returns between the active and passive strategy.

H_1: There is a significant difference in average returns between the active and passive strategy.

A Goodness of Fit test and a T-test are utilized to determine the statistical significance of the difference in average returns between the two strategies. The regression analysis yields an R-square of 0.092136 and an Adjusted R-square of 0.089635, indicating that only a small proportion of the variability in the trading strategy's returns is explained by these variables, thus suggesting that they are not strong predictors of performance. The T-test on the regression coefficient β1 resulted in a t-statistic of -6.0696 and a p-value of 3.2304e-09, which is significantly lower than the conventional alpha level of 0.05, compelling us to reject the null hypothesis (H_0) that there is no significant difference in average returns between the active and passive strategies. Therefore, the alternative hypothesis (H_1) is accepted that there is a significant difference in average returns. In addition, with the negative value of beta1, it is possible to affirm that, on average, the active trading strategy underperforms the passive strategy.

5. Conclusion

In concluding this research, a multifaceted evaluation of active and passive Ethereum investment strategies provides critical insights into their comparative performances. The active trading strategy demonstrates a negative mean return of -0.24%, a stark contrast to the passive strategy's mean return of 0.19%. This discrepancy in returns is compounded by a higher risk profile for the active strategy, as evidenced by its higher variance and standard deviation, alongside a pronounced kurtosis indicating a greater likelihood of extreme outcomes. Furthermore, risk measures such as the value-at-risk (VaR) and expected shortfall (ES) are considerably higher, with a maximum drawdown of 19.90% compared to the passive strategy's 11.19%, underscoring the potential for more substantial losses.

These quantitative findings are visually supported by the comparison of forecasted and actual daily returns, which reveals the challenges inherent in predicting the volatile cryptocurrency market and highlights the high risk associated with active ETH trading strategies.

The distribution of returns, depicted through a boxplot, further corroborates these results, showing a wider spread for the active strategy indicative of its higher risk, as opposed to the more consistent performance of the passive strategy.

Lastly, the hypothesis testing, utilizing a Goodness of Fit and T-test, provides a robust statistical foundation for rejecting the null hypothesis in favour of the alternative, affirming a significant difference in average returns between the strategies and suggesting the underperformance of the active strategy in comparison to its passive counterpart. The statistical analysis aligns with the narrative presented in the study, where the active trading strategy is assessed against a passive investment benchmark. The regression analysis complements the previous findings of the study, providing a quantitative basis for the conclusion that the active strategy, in this case, may not yield a superior performance in terms of average returns when compared to the passive strategy within the Ethereum market. This could also suggest that other variables, not included in the model, might have a substantial impact on returns, or that the market conditions during the period of the second model were not conducive to the strategy employed by the algorithm.

The implications of these findings are multiple and affect various stakeholders in the cryptocurrency market. Traders are confronted with the delicate balance of risk and return, where the pursuit of higher gains through active strategies may not necessarily outweigh the risks incurred. Investors, particularly those with a long-term horizon, might find solace in the relative stability of passive strategies, which, while potentially less lucrative, present a lower risk profile.

In conclusion, it is not true that the passive investment always performs better than a trading strategy. This depends on the choice of the order of ARIMA and GARCH models and the setting of the trading algorithm.

Appendix A

GROUP 3 CASE STUDY (Etherium)

```
clear
close all;
format default;

rng(123);    % set a seed for reproducibility

Data = readtable("eth2022-2023.csv")    % Loading data
```

Assesment

```
rtotETH = diff(log(Data.Price));             % Calculate daily returns
```

Technical and graphical analysis

```
Len = length(rtotETH);                       % n° returns
Mean = mean(rtotETH);                        % mean return
Variance = var(rtotETH);                     % variance
Stan_dev = std(rtotETH);                     % standard deviation
Kurt = kurtosis(rtotETH);                    % kurtosis
Skew = skewness(rtotETH);                    % skewness
VaR95 = -quantile(sort(rtotETH),0.05);       % VaR 95%
ES95 = -mean(sort(rtotETH(rtotETH<-VaR95))); % Expected-Shortfall 95%
VaR99 = -quantile(sort(rtotETH),0.01);       % VaR 99%
ES99 = -mean(sort(rtotETH(rtotETH<-VaR99))); % Expected-Shortfall 99%
Min = min(rtotETH);                          % Min
Max= max(rtotETH);                           % Max
cumulative_returns_ETH = cumsum(rtotETH);    % cumulative sum
Max_Gain = max(cumulative_returns_ETH);      % Max Gain
Max_Drawdown_ETH = 0;
Local_Min_ETH = cumulative_returns_ETH(1);
for i = 2:length(cumulative_returns_ETH)
    if cumulative_returns_ETH(i) > cumulative_returns_ETH(i-1)
        Local_Min_ETH = cumulative_returns_ETH(i);  % Update Local Min if
current cumulative_return is greater than previous
    else
        drawdown_ETH = Local_Min_ETH - cumulative_returns_ETH(i);
        if drawdown_ETH > Max_Drawdown_ETH
            Max_Drawdown_ETH = drawdown_ETH;        % Update Max Drawdown if
current drawdown is greater
        end
    end
end
Max_Drawdown = Max_Drawdown_ETH;              % Max drawdown
```

```
 names = {'Mean'; 'Var'; 'Stan. Dev.'; 'Kurt'; 'Skew'; 'VaR95'; 'ES95';
'VaR99'; 'ES99'; 'Min'; 'Max'; 'Max Gain'; 'Max Drawdown'};
 ETH_Value = [Mean; Variance; Stan_dev; Kurt; Skew; VaR95; ES95; VaR99; ES99;
Min; Max; Max_Gain; Max_Drawdown];

 Table_ETH = table(names, ETH_Value)
```

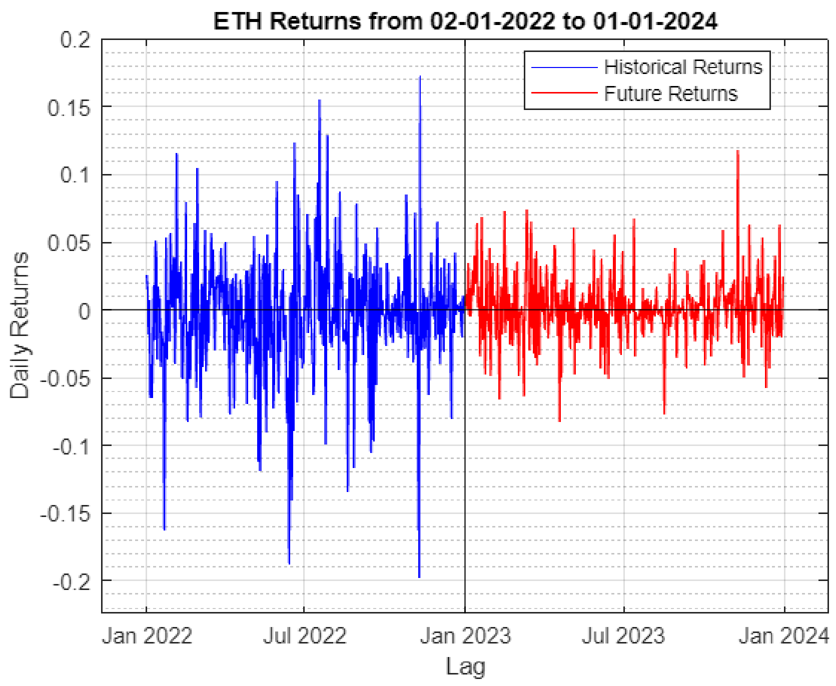
```
Table_ETH = 13×2 table
```

	Variables	ETH
1	Mean	-0.0006
2	Var	0.0013
3	Stan. Dev.	0.0367
4	Kurt	7.7610
5	Skew	-0.4431
6	VaR95	0.0581
7	ES95	0.0942
8	VaR99	0.1172
9	ES99	0.1577
10	Min	-0.1975
11	Max	0.1726
12	Max Gain	0.0396
13	'Max Drawdown	0.4337

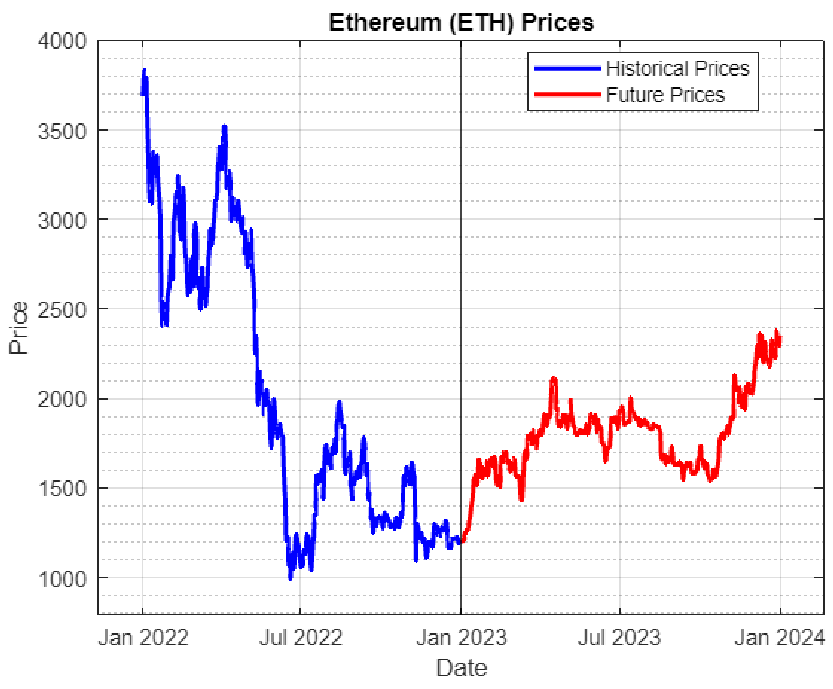
Breaking data

```
 historical_returns = rtotETH(1:365);          % from 02-01-2022 to 31-12-2022
 objective_returns = rtotETH(366:end);          % from 01-01-2023 to 01-01-2024

 % Plot Ethereum Returns
 plot(Data.Date(1:365),rtotETH(1:365), "b");
 hold on
 plot(Data.Date(366:end-1),rtotETH(366:end), "r");
 yline(0, "k");
 xline(365, "k");
 title('ETH Returns from 02-01-2022 to 01-01-2024', 'FontSize', 14,
'FontWeight', 'bold');
```

```
xlabel('Lag', 'FontSize', 12);
ylabel('Daily Returns', 'FontSize', 12);
legend('Historical Returns','Future Returns', Location = 'best');
axis padded;
grid on;
grid minor;
set(gca, 'FontSize', 10);
box on;
hold off
```



```
% Plot Ethereum Prices
plot(Data.Date(1:365), Data.Price(1:365), "b", 'LineWidth', 2);
hold on
plot(Data.Date(366:end), Data.Price(366:end), "r", 'LineWidth', 2);
xline(365, "k");
title('Ethereum (ETH) Prices', 'FontSize', 14, 'FontWeight', 'bold');
xlabel('Date', 'FontSize', 12);
ylabel('Price', 'FontSize', 12);
legend('Historical Prices','Future Prices', Location = 'best');
axis padded;
grid on;
grid minor;
set(gca, 'FontSize', 10);
box on;
hold off
```



IID test and Stationarity test

```
 iid_returns = historical_returns;   % Initialize the historical returns for IID testing

 for i = 2:length(iid_returns)-1
     if ((iid_returns(i+1)<iid_returns(i)) && (iid_returns(i)>iid_returns(i-1))) || ((iid_returns(i+1)>iid_returns(i)) && (iid_returns(i)<iid_returns(i-1)))
         iid_returns(i) = 1;  % Mark as 1 if a local maximum (peak) or minimum (trough) is found
     else
         iid_returns(i) = 0;  % Otherwise, mark as 0
     end
 end

 iid_returns = iid_returns(2:end-1);    % Remove the first and last elements because they cannot be peaks or troughs
 mu = {mean(iid_returns)}               % Calculate the mean
```

```
mu = 1×1 cell array

    {[0.5344]}
```

```
 Var ={var(iid_returns)}                % Calculate the variance
```

```
Var = 1×1 cell array

    {[0.2495]}
```

```
 % Hypothesis testing if returns are IID
 if (abs (sum (historical_returns)-(2/3)*length(historical_returns) ) ) >
(1.96*sqrt( (8*length(historical_returns))/45 ))
     h_TPT = 1            % Reject the null hypothesis, indicating non-IID
returns
 else
     h_TPT = 0            % Accept the null hypothesis, indicating IID returns
 end
```

```
h_TPT = 1
```

```
 H = runstest(historical_returns)                             % Runstest for
randomness
```

```
H = 0
```

```
 residual = historical_returns - mean(historical_returns);   % Calculate
residuals from historical returns
 h = lbqtest(residual)                                        % Ljung-Box Q-
test for residual autocorrelation
```

```
h = logical

  0
```

```
 h2 = lbqtest(residual.^2)                                    % Test the
hypothesis that there are significant ARCH effects
```

```
h2 = logical

  1
```

```
 h3 = archtest(residual)                                      % ARCH-Test for
residual heteroscedasticity
```

```
h3 = logical

  1
```

```
 adftest(historical_returns)                                  % Stationarity
test; if ans = 1 -> non stationarity
```

```
ans = logical

  1
```

Our returns are not IID and not stationary, so we cannot apply the ARMA model. We have to find another model that fits the data (ARIMA Model).

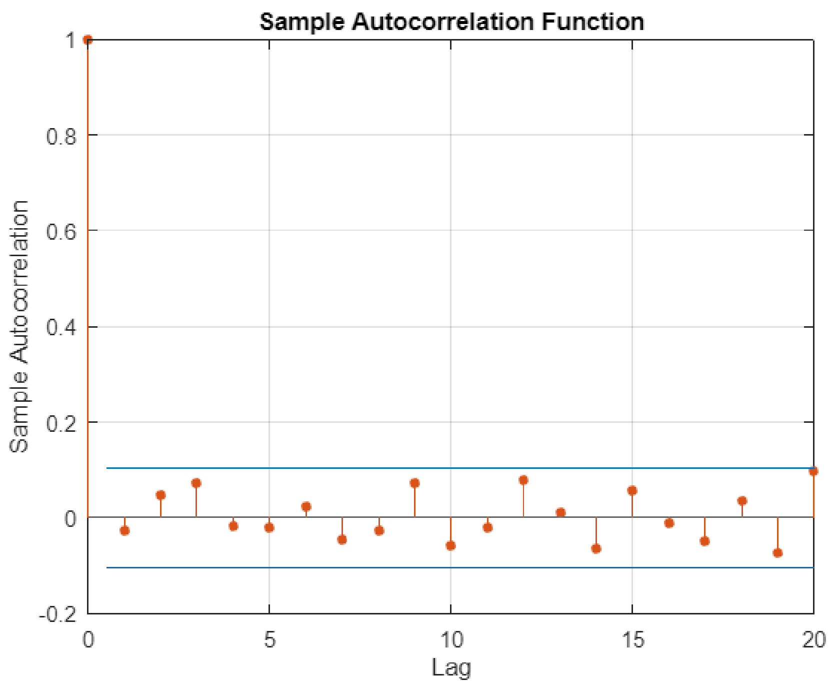
ARIMA Model

```
 ARIMAmodel = arima(1,1,2)                  % Define of ARIMA model; p = 1, D =
1 (the logret are not stationary), q = 2
 EstArimaModel = estimate(ARIMAmodel, historical_returns)   % Estimate the
parameters of ARIMA Model on historical returns
```

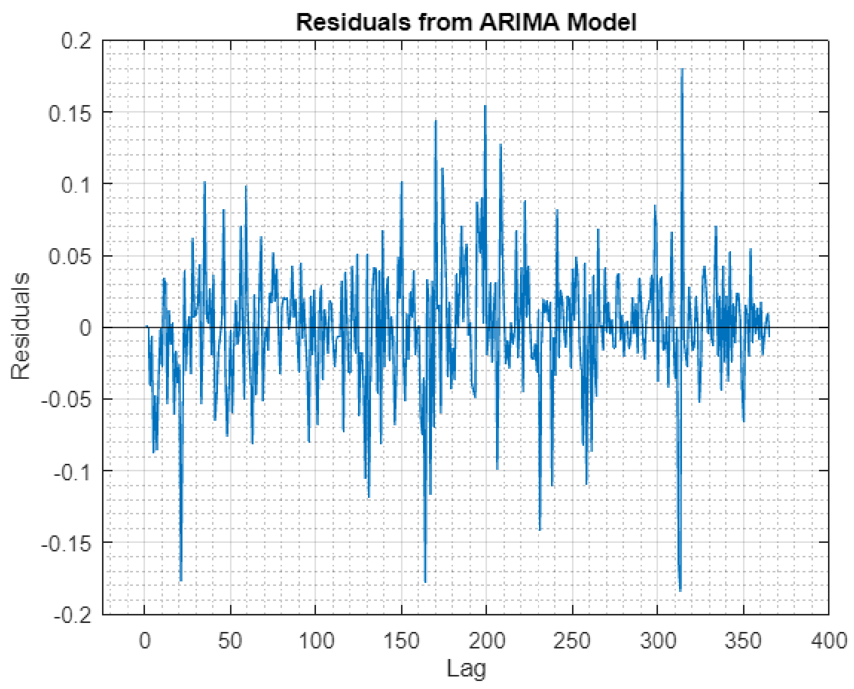
```
    ARIMA (1,1,2) Model (Gaussian Distribution):

                  Value      StandardError    TStatistic      PValue
               __________    _____________    __________    ___________

    Constant   -6.348e-05     0.00013168       -0.48207         0.62976
    AR {1}       -0.94142       0.032827        -28.678     7.2421e-181
    MA {1}       0.010274       0.021494        0.47799         0.63266
    MA {2}       -0.96282       0.018799        -51.216               0
    Variance    0.0021214     0.00010387         20.423      1.0466e-92
EstArimaModel =
  arima with properties:

     Description: "ARIMA (1,1,2) Model (Gaussian Distribution)"
      SeriesName: "Y"
    Distribution: Name = "Gaussian"
               P: 2
               D: 1
               Q: 2
        Constant: -6.34804e-05
              AR: {-0.941415} at lag [1]
             SAR: {}
              MA: {0.0102739 -0.962824} at lags [1 2]
             SMA: {}
     Seasonality: 0
            Beta: [1×0]
        Variance: 0.0021214
```

```
 epsilons = infer(EstArimaModel, historical_returns);       % Calculate
residuals from the ARIMA model
```

```
 autocorr(epsilons)                                                    % Autocorrelation
function between the residuals
```



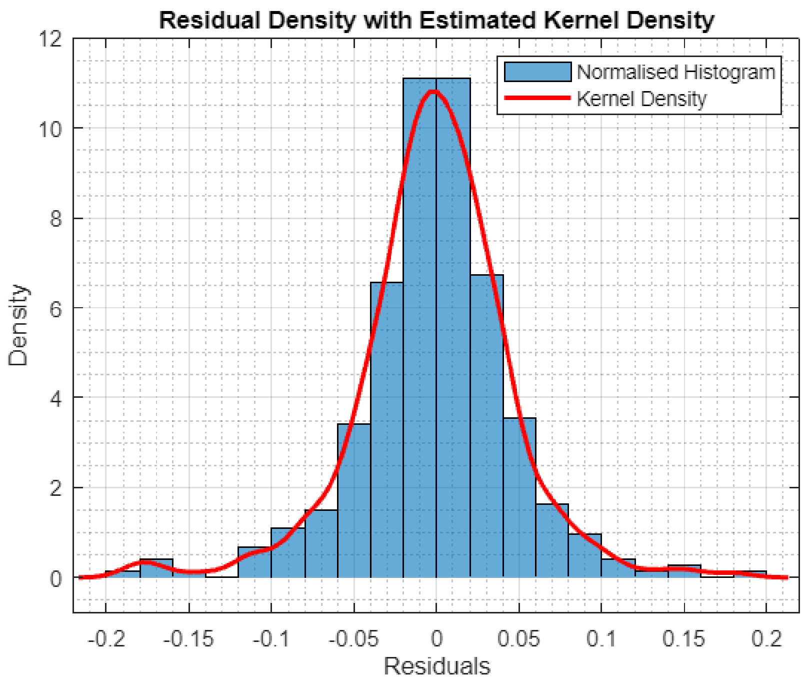
```
 plot(epsilons);                    % check if the residuals have mean 0
 hold on;
 yline(0, "k");
 title('Residuals from ARIMA Model', 'FontSize', 14, 'FontWeight', 'bold');
 xlabel('Lag', 'FontSize', 12);
 ylabel('Residuals', 'FontSize', 12);
 grid on;
 grid minor;
 axis padded;
 set(gca, 'FontSize', 10);
 box on;
 hold off;
```



```
 h1 = lbqtest(epsilons)                              % Ljung-Box Q-test on residuals
for lack of autocorrelation
```

```
h1 = logical

   0
```

```
 figure;
 histogram(epsilons, 'Normalization', 'pdf')                % Plot of the
residuals histogram with normalisation PDF
 hold on

 [f, xi] = ksdensity(epsilons);                             % Calculation of
residual Kernel Density
 plot(xi, f, 'r-', 'LineWidth', 2)                          % Red line for
Kernel Density

 title('Residual Density with Estimated Kernel Density', 'FontSize', 14,
'FontWeight', 'bold');
 xlabel('Residuals', 'FontSize', 12);
 ylabel('Density', 'FontSize', 12);
 legend('Normalised Histogram', 'Kernel Density', Location ='northeast');
 grid on;
 grid minor;
 axis padded;
 set(gca, 'FontSize', 10);
 box on;
 hold off
```



Our ARIMA Model is ***ARIMA (1, 1, 2)***

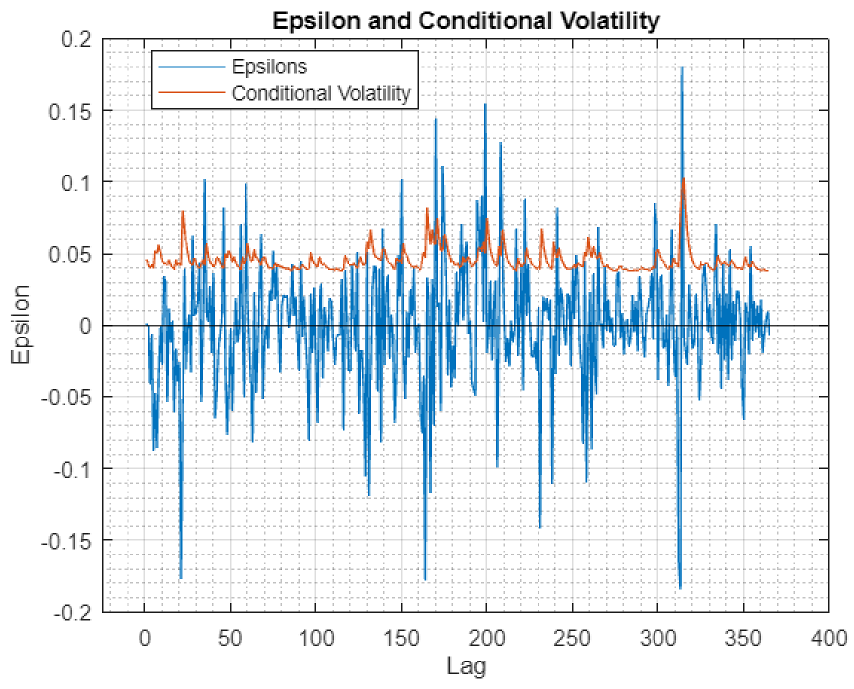
GARCH Model

```
 VarMdl = garch(1,1)   % Define a GARCH(1,1) model for volatility
 EstGarchModel = estimate(VarMdl, epsilons)   % Estimate the parameters of the
GARCH model using ARIMA model residuals
```

```
    GARCH (1,1) Conditional Variance Model (Gaussian Distribution):

                    Value       StandardError    TStatistic      PValue   
                  __________    _____________    __________    __________

    Constant      0.00063475     0.00027642        2.2964        0.021655
    GARCH {1}        0.54971        0.16303        3.3718      0.00074683
    ARCH {1}         0.15332       0.055046        2.7854       0.0053463
EstGarchModel = 
  garch with properties:

     Description: "GARCH (1,1) Conditional Variance Model (Gaussian Distribution)"
      SeriesName: "Y"
    Distribution: Name = "Gaussian"
```

```
             P: 1
             Q: 1
      Constant: 0.000634755
         GARCH: {0.549706} at lag [1]
          ARCH: {0.153325} at lag [1]
        Offset: 0
```

```
 condVar = infer(EstGarchModel, epsilons);      % Calculate the conditional 
variance
 condVol = sqrt(condVar);                       % Calculate the conditional 
volatility

 plot(epsilons);
 hold on
 plot(condVol);
 yline(0, "k");
 title("Epsilon and Conditional Volatility",'FontSize', 14, 'FontWeight', 
'bold');
 xlabel('Lag', 'FontSize', 12);
 ylabel('Epsilon', 'FontSize', 12);
 legend("Epsilons","Conditional Volatility", location="best");
 grid on;
 grid minor;
 axis padded;
 set(gca, 'FontSize', 10);
 box on;
 hold off
```



Our GARCH model is ***GARCH (1, 1).***

Data test via MatLab

Assumptions

```
 alpha = 0.03;                % weight of previous Y, >0
 beta = 0.95;                 % weight of previous sigma, >0
 omega = 0.0001;              % constant, >0
 rng default
 z = normrnd(0,1,[1,365]);    % 1000 norm. innovations with mu=0, var=1
```

Preassigning space for loop

```
 sigma2 = zeros([1,365]);                 % variance
 sigma2(1,1)= omega/(1-(alpha+beta));     % Uncondit. var -> Innitial
value at t=0
 global epsilon
 epsilon = zeros([1,365]);                % error term
 epsilon(1,1) = sqrt(sigma2(1,1)).*z(1,1); % initial value at t=0
 Y = zeros([1,1000]);                     % value of time series
 Y(1,1) = epsilon(1,1)*sqrt(sigma2(1,1)); % initial value at t=0
```

Estimating data

```
 for i = 2:365
     sigma2(i) = omega+alpha*epsilon(i-1)^2+beta*sigma2(i-1);   % variance
     epsilon(i) = sqrt(sigma2(i)).*z(i);                        % error term
     Y(i) = Y(i-1)+epsilon(i)*sqrt(sigma2(i));                  % predicted
values
 end

 % GARCH MLE template

 % epsilon is the vector (Tx1) containing the time serie of the returns of the
simulated GARCH model

 theta0 = [0.02  0.05   0.85];    % starting value for the parameter vector
omega; alpha; beta

 A = [-1   0    0   ;             % contraints A*theta<=bounds
     1    0    0    ;
     0    -1   0    ;
     0     1   0    ;
     0    0    -1   ;
     0    0     1   ];

 bounds= [0;
     0.99999;
     0;
     0.99999;
     0;
     0.99999];
 % fmincon:
 options = optimset('Display','iter','TolFun',1e-10,'TolX',1e-20, 'TolCon',1e-
20,'MaxFunEvals',5000);

 [theta_hat, fval] = fmincon(@garch_loglikelihood, theta0, A, bounds, [] , [],
[], [], [], options)
 data = epsilon.';
 GARCH_X1 =
garch('Offset',0,'GARCHLags',1,'ARCHLags',1,'Distribution','Gaussian');
 GARCH_X1 = estimate(GARCH_X1,data,'Display','off');
 garch_comparison= table([theta0(1); theta_hat(1); GARCH_X1.Constant], ...
     [theta0(2); theta_hat(2); GARCH_X1.ARCH], ...
     [theta0(3); theta_hat(3); GARCH_X1.GARCH], ...
     'VariableNames',{'Omega' 'Alpha' 'Beta'},'RowNames',{'True';
'Estimated';'Matlab'})
```

garch_comparison = 3×3 table

	Omega	Alpha	Beta
1 True	0.0200	0.0500	0.8500
2 Estimated	0.0002	0.0402	0.9331
3 Matlab	0.0002	0.0408	0.9325

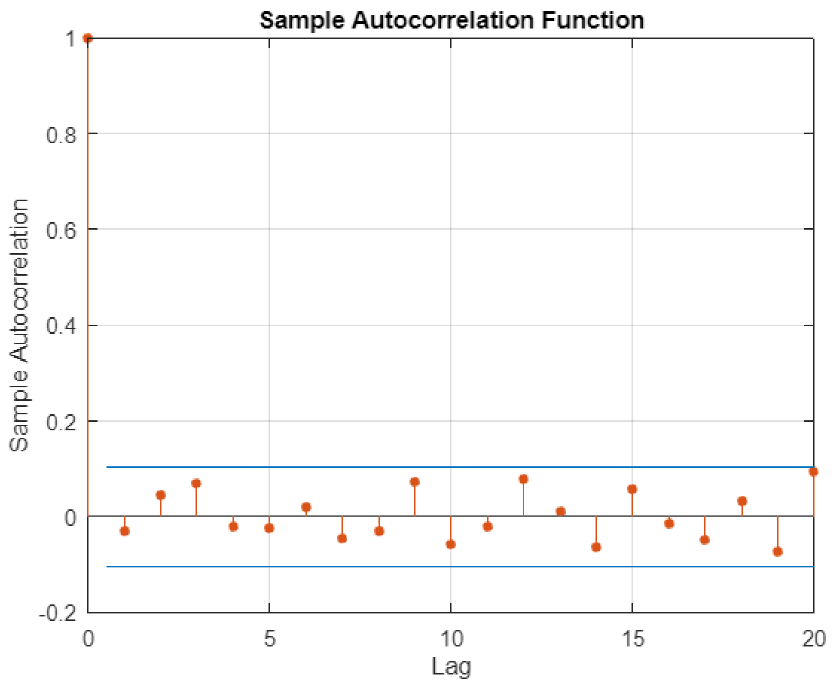
Combined ARIMA and GARCH Model

```
 VarMdl = garch(1,1);                            % Model for Conditional Volatility
 Mdl = arima('ARLags',1,'AR',-0.941415,'D',1,'MALags',1:2,'MA',{0.0102739,-
0.962824},'Variance',EstGarchModel); % Combine ARIMA and GARCH Models using
previous parameters
 Mdl.Distribution = 'Gaussian';                  % Distribution of Residuals
 options = optimoptions(@fmincon, 'Display', 'off', 'Diagnostics', 'off',
'Algorithm', 'sqp', 'TolCon', 1e-7);
 EstMdl = estimate(Mdl,historical_returns,'Display', 'off', 'Options',
options) % Estimate the combined ARIMA-GARCH model parameters using historical
returns
```

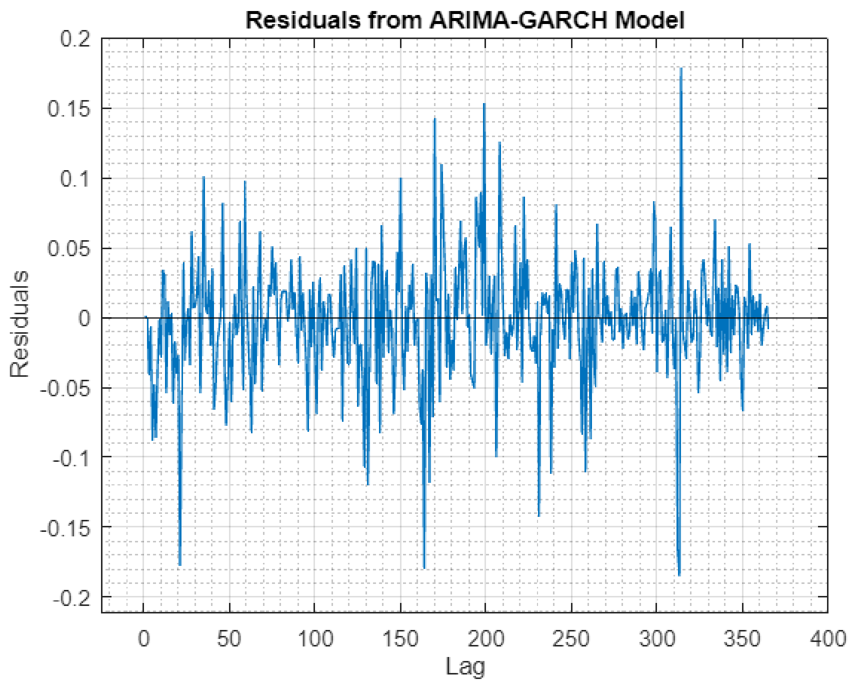
```
EstMdl = 

  arima with properties:


     Description: "ARIMA (1,1,2) Model (Gaussian Distribution)"

      SeriesName: "Y"

    Distribution: Name = "Gaussian"

               P: 2
```

```
                D: 1
                Q: 2
         Constant: -8.83336e-07
               AR: {-0.941415} at lag [1]
              SAR: {}
               MA: {0.0102739 -0.962824} at lags [1 2]
              SMA: {}
      Seasonality: 0
             Beta: [1×0]
         Variance: [GARCH (1,1) Model]
```

```
 OmegaARIMA = EstMdl.Constant;                        % ARIMA model constant
 AR = EstMdl.AR{1};                                   % ARIMA model AR
coefficient
 MA1 = EstMdl.MA{1};                                  % ARIMA model first MA
coefficient
 MA2 = EstMdl.MA{2};                                  % ARIMA model second MA
coefficient
 Omega = EstMdl.Variance.Constant;                    % GARCH model constant
(omega)
 GARCH = EstMdl.Variance.GARCH{1};                    % GARCH model GARCH
coefficient (beta)
 ARCH = EstMdl.Variance.ARCH{1};                      % GARCH model ARCH
coefficient (alpha)
 [res,v,logL] = infer(EstMdl, historical_returns); % Infer residuals, variance
and log-likeli of return series

 autocorr(res)                                        % Autocorrelation function
between the residuals
```



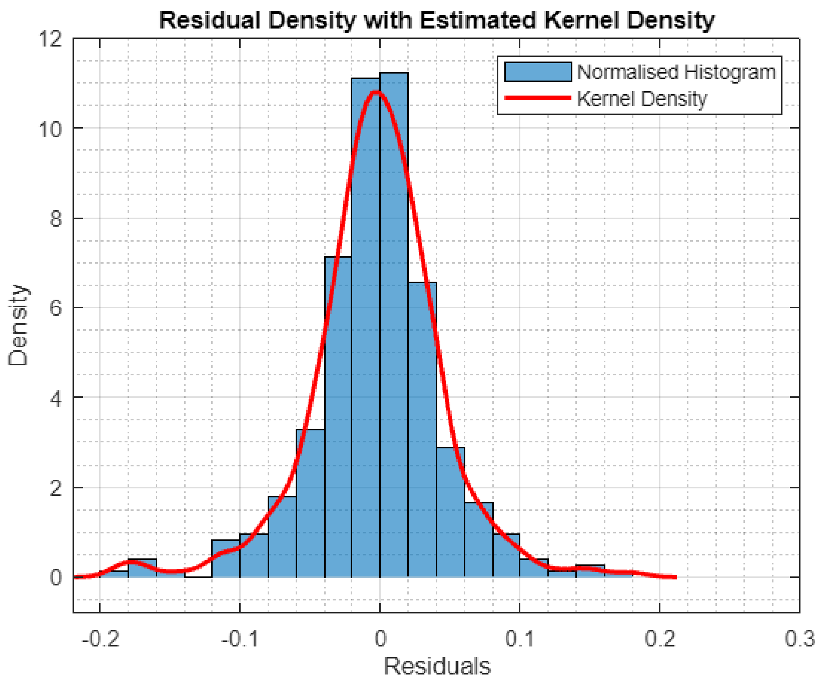
```
plot(res)                    % check if the residuals have mean 0
hold on
yline(0, "k");
title('Residuals from ARIMA-GARCH Model', 'FontSize', 14, 'FontWeight',
'bold');
xlabel('Lag', 'FontSize', 12);
ylabel('Residuals', 'FontSize', 12);
grid on;
grid minor;
axis padded;
set(gca, 'FontSize', 10);
box on;
hold off
```



```
 h1 = lbqtest(res)     % Ljung-Box Q-test for residual autocorrelation
```

```
h1 = logical

   0
```

```
 histogram(res, 'Normalization', 'pdf')              % Plot of the residual
histogram with normalisation PDF
 hold on
 [f, xi] = ksdensity(res);                          % Calculation of residual
kernel density
 plot(xi, f, 'r-', 'LineWidth', 2);                  % Red line for Kernel
Density
 title('Residual Density with Estimated Kernel Density', 'FontSize', 14,
'FontWeight', 'bold');
 xlabel('Residuals', 'FontSize', 12);
 ylabel('Density', 'FontSize', 12);
 legend('Normalised Histogram', 'Kernel Density', Location = 'northeast');
 grid on;
 grid minor;
 axis padded;
 set(gca, 'FontSize', 10);
 box on;
 hold off
```



Forecasting of the first day

In this trading strategy there are two parameters, the Risk Tolerance and a Threshold, to guide the trading decisions. The Risk Tolerance determines how much money a person is willing to risk for a single operation (purchase or sale). The Threshold is a predefined limit used to take the decision of buy, sell or hold a share based on model forecasts.

In this case study the Risk Tolerance is equal to 2% and it is used only for the purchasing transactions because the purpose is to limit losses only during them.

Instead, the Threshold is equal to 3%. So, if the forecasted return is higher than the Threshold, the decision is to buy; if it is lower, the decision is to sell. If the forecasted return is in the middle, the decision is to hold the share in the portfolio.

```
 % Initialize variables
 riskTolerance = 0.02; % 2 % of portfolio for transaction
 decisionThreshold = 0.03; % 3 % of the expected return to decide if buy or
sell

 lastValue1 = historical_returns(end);   % Last observed return
 lastDiff1 = diff(historical_returns);   % Differenced returns series
 lastVariance1 = v(end);                 % Last known conditional variance

 Y_forecast1 = OmegaARIMA + lastValue1 + AR * lastDiff1(end) + MA1 * res(end)
+ MA2 * res(end-1); % 1-day return forecast
 lastDiff1(end+1) = Y_forecast1 - lastValue1;   % Update the differenced
returns series with the forecasted return
 lastValue1 = Y_forecast1;  % Update the last value with the forecasted return
 var_forecast1 = Omega + ARCH * res(end)^2 + GARCH * lastVariance1; % 1-day
variance forecast
 lastVariance1 = var_forecast1;  % Update the last variance with the
forecasted variance
 sigma_forecast1 = sqrt(var_forecast1);  % Calculate the forecasted standard
deviation
```

Trading Strategy for the first day

The portfolio at the starting date is composed only of cash (10.000 USD) and no shares.

```
 % Initialize trading variables
 currentPosition = 0;          % Current position, assuming starting with no
stocks
 currentCash = 10000;          % Starting cash
 profit = 0;                   % Initial profit for the trading strategy

 positionSize = currentCash * riskTolerance / sigma_forecast1; % Calculate the
position size based on risk management
```

The position size is calculated as a fraction of the cash available, proportional to the risk tolerance and inversely proportional to the expected volatility of the financial asset.

```
 % Trading decision based on the forecasted return
 if Y_forecast1 > decisionThreshold   % Predicting a return greater than the
Threshold, decide to buy
      numSharesToBuy = max(1, floor(positionSize / Data{365, 'Price'}));  %
Determine the number of shares to buy
```

To determine the number of shares to buy it is considered the position size to limit the risk exposition of the portfolio. To optimize the algorithm, there is the freedom to also buy one share if the result of floor(positionSize / Data{365, 'Price'}) is equal to zero if there is enough cash (code line below).

```
     if currentCash >= numSharesToBuy * Data{(365), 'Price'}   % Check if it's
possible to buy numShares
         currentPosition = currentPosition + numSharesToBuy; % Increase in the
number of shares of the currentPosition
         currentCash = currentCash - numSharesToBuy * Data{365, 'Price'}; %
Adjustment of liquidity following acquisition
         TradReturns1 = log(Data{(365), 'Price'} / Data{(364),'Price'});    %
First return of the trading strategy
```

If there is the acquisition, the trading return is equal to the logarithmic one with the following formula:

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

```
         disp(['Bought ', num2str(numSharesToBuy), ' shares']);
     else
         TradReturns1 = objective_returns(1);      % First return of the
trading strategy
```

If there is no acquisition, the trading return is equal to the return of the passive investment.

```
         disp('Insufficient funds to buy shares');
     end
```

```
 elseif Y_forecast1 < - decisionThreshold  % Predicting a return lower than
the Threshold, decide to sell
     numSharesToSell = currentPosition; % Determine the number of shares to
sell
```

To determine the number of shares to sell, the algorithm decides to sell every share in the portfolio.

```
     if numSharesToSell > 0  % Check if there are shares to sell
         currentPosition = currentPosition - numSharesToSell; % Decrease in
the number of shares of the position
         currentCash = currentCash + numSharesToSell * Data{365, 'Price'}; %
Adjustment of liquidity following sale
         profit = profit + numSharesToSell * (Data{365, 'Price'} - Data{364,
'Price'}); % Calculate the profit assuming Data{364, 'Price'} is the
acquisition price
```

The profit is calculated adding to the previous one the number of sold shares multiplied by the difference between the price of today and the price of the acquisition date.

```
         disp(['Sold ', num2str(numSharesToSell), ' shares']);
         TradReturns1 = log(Data{(365), 'Price'} / Data{(366), 'Price'});    %
First return of the trading strategy
```

If there is the sale, the trading return is equal to the logarithmic one with the following formula:

$$r(t) = \ln\left(\frac{P_t}{P_{t+1}}\right)$$

```
     else
         TradReturns1 = objective_returns(1);    % First return of the trading
strategy
```

If there is no sale, the trading return is equal to the return of the passive investment.

```
         disp('Not enough shares to sell');
     end
 else
     TradReturns1 = objective_returns(1);    % First return of the trading
strategy
```

If there is no operation, the trading return is equal to the return of the passive investment.

```
     disp('No trading action taken');
 end
```

```
No trading action taken.
```

```
 RiskLevel1 = riskTolerance / sigma_forecast1;  % Risk Level of the trading
strategy for the first day
```

```
 disp(['New position: ', num2str(currentPosition),' shares']);  % Print out 
the new position
```

```
New position: 0 shares
```

```
 disp(['New cash balance: ', num2str(currentCash), ' USD']);    % Print out 
the new cash levels
```

```
New cash balance: 10000 USD
```

```
 disp(['New Profit: ', num2str(profit), ' USD']);               % Print out 
the new profit
```

```
New Profit: 0 USD
```

Forecasting of the future values

```
 % Initialize variables
 ForecastRet = zeros(length(objective_returns),1);      % Creating a vector 
for the forecasted returns
 ForecastVol = zeros(length(objective_returns),1);      % Creating a vector 
for the forecasted volatility
 TradReturns = zeros(length(objective_returns),1);      % Creating a vector 
for the returns of the trading strategy
 RiskLevel = zeros(length(objective_returns),1);        % Creating a vector 
for the risk levels
 ForecastRet(1) = Y_forecast1;                          % First value of the 
forecasted returns
 ForecastVol(1) = sigma_forecast1;                      % First value of the 
forecasted volatility
 TradReturns(1) = TradReturns1;                         % First value of the 
returns of the trading strategy
 RiskLevel(1) = RiskLevel1;                             % First value of the 
risk levels

 numPeriods = 7;                                        % Number of future 
periods to forecast
 k = 0;                                                 % Counter for the n° 
of periods
 lastVariance = lastVariance1;                          % Last updated known 
conditional variance
 lastDiff = diff(rtotETH);                              % Updated differenced 
returns series
```

364 days need to be forecasted. To do so there are two for-loops. The first for-loop takes weekly intervals and at every loop it recalibrates the model on the real returns.

The second for-loop calculates for everyday of the week the forecasted return and variance and decides, based on the trading algorithm, which move it does in the market.

```
 for j = 1:numPeriods:(length(objective_returns) - numPeriods)  % Weekly 
intervals
```

```
     data_tot = rtotETH((j):(365+j));  % data_tot contains the ETH returns for
the 365 days following index j
     EstMdl = estimate(Mdl,data_tot,'Display', 'off', 'Options', options); %
Estimate the combined ARIMA-GARCH model parameters using data_tot
     OmegaARIMA = EstMdl.Constant; % Model parameters
     AR = EstMdl.AR{1};
     MA1 = EstMdl.MA{1};
     MA2 = EstMdl.MA{2};
     Omega = EstMdl.Variance.Constant;
     GARCH = EstMdl.Variance.GARCH{1};
     ARCH = EstMdl.Variance.ARCH{1};
     [res,v,logL] = infer(EstMdl, data_tot);
     lastValue = objective_returns(j);  % Update the lastValue everytime the
model is estimated

     for i = 1:numPeriods
         Y_forecast(i) = OmegaARIMA + lastValue + AR * lastDiff(365+i+j-2) +
MA1 * res(end) + MA2 * res(end-1); % i-day return forecast
         lastValue = Y_forecast(i); % Update the last value with the
forecasted return
         var_forecast(i) = Omega + ARCH * res(end)^2 + GARCH * lastVariance; %
i-day variance forecast
         lastVariance = var_forecast(i); % Update the last value with the
forecasted variance
         positionSize = currentCash * riskTolerance / sqrt(var_forecast(i)); %
Calculate the position size based on risk management
```

The position size is calculated as a fraction of the cash available, proportional to the risk tolerance and inversely proportional to the expected volatility of the financial asset.

```
         RiskLevel(i+1 + k*7) = riskTolerance / sqrt(var_forecast(i)); % i-
day's Risk Level of the trading strategy
         if Y_forecast(i) > decisionThreshold % Predicting a return greater
than the Threshold, decide to buy
             numSharesToBuy = max(1, floor(positionSize / Data{(365+i+j-1),
'Price'})); % Determine the number of shares to buy
```

To determine the number of shares to buy it is considered the position size to limit the risk exposition of the portfolio. To optimize the algorithm, there is the freedom to also buy one share if the result of floor(positionSize / Data{365, 'Price'}) is equal to zero if there is enough cash (code line below).

```
             if currentCash >= numSharesToBuy * Data{(365+i+j-1), 'Price'} %
Check if it's possible to buy numShares
                 currentPosition = currentPosition + numSharesToBuy; %
Increase in the number of shares of the currentPosition
                 currentCash = currentCash - numSharesToBuy * Data{(365+i+j-
1), 'Price'}; % Adjustment of liquidity following acquisition
                 Acqui_Date = 365+i+j-1; % Date of acquisition
                 disp(['Bought ', num2str(numSharesToBuy), ' shares']);
```

```
                TradReturns(i+1 + k*7) = log(Data{(365+i+j), 'Price'} /
Data{(365+i+j-1), 'Price'});    % Updating returns of the trading strategy
```

If there is the acquisition, the trading return is equal to the logarithmic one with the following formula:

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

```
            else
                TradReturns(i+1 + k*7) = objective_returns(i+1 +k*7);   %
Updating returns of the trading strategy
```

If there is no acquisition, the trading return is equal to the return of the passive investment.

```
                disp('Insufficient funds to buy shares');
            end

        elseif Y_forecast(i) < -decisionThreshold % Predicting a return lower
than the Threshold, decide to sell
            numSharesToSell = currentPosition; % Determine the number of
shares to sell
```

To determine the number of shares to sell, the algorithm decides to sell every share in the portfolio.

```
            if currentPosition > 0 % Check if there are shares to sell
                currentPosition = currentPosition - numSharesToSell; %
Decrease in the number of shares of the position
                currentCash = currentCash + numSharesToSell * Data{(365+i+j-
1), 'Price'}; % Adjustment of liquidity following sale
                profit = profit + numSharesToSell * (Data{(365+i+j-1),
'Price'} - Data{(Acqui_Date), 'Price'}); % Calculate the profit assuming
Data{Acqui_Date, 'Price'} is the acquisition price
```

The profit is calculated adding to the previous one the number of sold shares multiplied by the difference between the price of today and the price of the acquisition date.

```
                disp(['Sold ', num2str(numSharesToSell), ' shares']);
                TradReturns(i+1 + k*7) = log(Data{(365+i+j-1), 'Price'} /
Data{(365+i+j), 'Price'});   % Updating returns of the trading strategy
```

If there is the sale, the trading return is equal to the logarithmic one with the following formula:

$$r(t) = \ln\left(\frac{P_t}{P_{t+1}}\right)$$

```
            else
                TradReturns(i+1 + k*7) = objective_returns(i+1 + k*7);   %
Updating returns of the trading strategy
```

If there is no sale, the trading return is equal to the return of the passive investment.

```
                disp('Not enough shares to sell');
            end
        else
            TradReturns(i+1 + k*7) = objective_returns(i+1 + k*7);   %
Updating returns of the trading strategy
```

If there is no operation, the trading return is equal to the return of the passive investment.

```
            disp('No trading action taken');
        end
        ForecastRet((i+1) + k*7) = Y_forecast(i);             % Store the
forecasted return
        ForecastVol((i+1) + k*7) = sqrt(var_forecast(i));     % Store the
forecasted volatility
    end
    k = k+1;   % Update the counter
end

disp(['New position: ', num2str(currentPosition),' shares']);    % Print out
the new position
```

```
New position: 3 shares
```

```
disp(['New cash balance: ', num2str(currentCash), ' USD']);        % Print
out the new cash levels
```

```
New cash balance: 785.4528 USD
```

```
disp(['New Profit: ', num2str(profit), ' USD']);                   % Print
out the new profit
```

```
New Profit: -798.388 USD
```

```
FinalValue_Trad = currentCash + currentPosition*Data{730,"Price"};  % Final
Value of the Portfolio
disp(['Final Value: ', num2str(FinalValue_Trad), ' USD']);         % Print
out the final value of the portfolio for the trading strategy
```

```
Final Value: 7668.4848 USD
```

```
RiskAttitude = mean(RiskLevel)                                    % Risk
Attitude of the Trading Strategy
```

```
RiskAttitude = 0.5114
```

The Risk Attitude value is in the range of 0, that means an extremely risk-averse investor, to 1, that means an extremely risk-taking investor. In this case the result of 0.5114 indicates a moderate risk-taking investor.

Performance of the Passive Strategy (Buy and Hold Strategy)

With the passive strategy at the starting date, it is calculated how many shares it is possible to buy having an initial value of the portfolio equal to 10.000 USD. Then these shares are kept in the portfolio until the last day when they are sold at the final price.

```
 initialCash = 10000;                             % Initial investment
 initialPrice = Data{(365), 'Price'};             % First price in the data
 numShares1 = round(initialCash / initialPrice)    % Number of shares can be
bought initially
```

```
numShares1 = 8
```

```
 remainingCash = initialCash - numShares1*initialPrice;  % Remaining cash
after buying shares

 finalPrice = Data{(730), 'Price'};                     % Last price in the
data
 finalValue = numShares1 * finalPrice + remainingCash;   % Final value of the
passive investment
 buyHoldProfit = finalValue - initialCash;                % Total profit of
the buy and hold strategy
 buyHoldReturn = (finalValue - initialCash) / initialCash; % Total return of
the buy and hold strategy
 disp(['Final Value: ', num2str(finalValue), ' USD']);                       %
Print out the final value of the portfolio for the passive strategy
```

```
Final Value: 18761.8681 USD
```

```
 disp(['New Profit: ', num2str(buyHoldProfit), ' USD']);                     %
Print out the new profit
```

```
New Profit: 8761.8681 USD
```

```
 disp(['Passive Investment return: ', num2str(buyHoldReturn), ' USD']);     %
Print out the return of the passive investment
```

```
Passive Investment return: 0.87619 USD
```

Performance of the Trading Strategy (Active Strategy)

```
 ActiveReturns = (currentCash + currentPosition*Data{730,"Price"} -
initialCash) / initialCash; % Total return of trading strategy
 disp(['Active Strategy return: ', num2str(ActiveReturns)]);
% Print out the return of the trading strategy
```

```
Active Strategy return: -0.23315
```

Analysis of the Trading Strategy and of the Passive Strategy

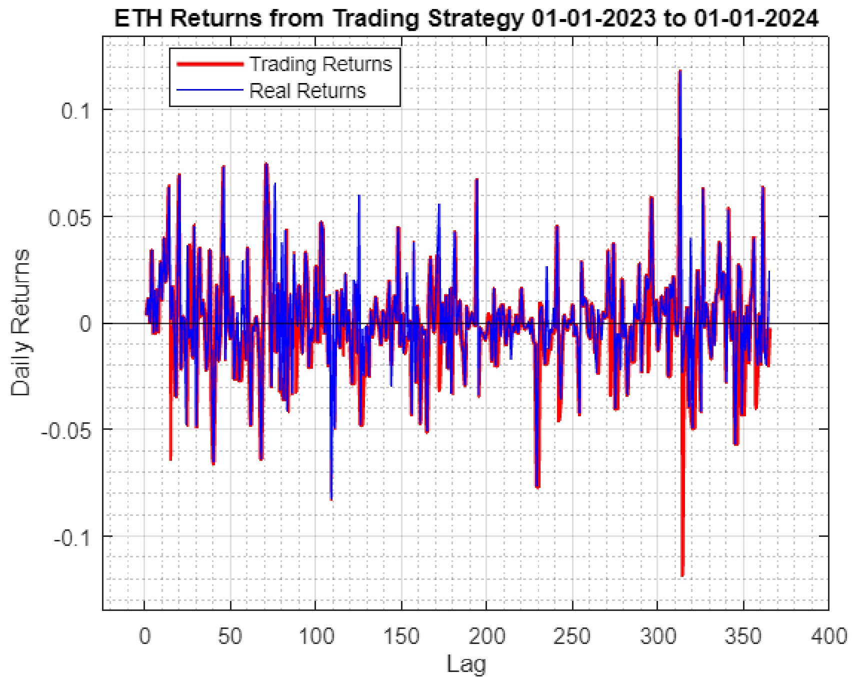
```
 % Trading Strategy
 Mean_Trad = mean(TradReturns);                                      % Average
daily returns
 Var_Trad = var(TradReturns);                                        % Variance
 SD_Trad = std(TradReturns);                                 % Standard
deviation
 Kurt_Trad = kurtosis(TradReturns);                          % Kurtosis
 Skew_Trad = skewness(TradReturns);                          % Skewness
 VaR95_Trad = -quantile(sort(TradReturns),0.05);             % VaR 95%
 ES95_Trad = -mean(sort(TradReturns(TradReturns<-VaR95_Trad)));  % Expected-
Shortfall 95%
 VaR99_Trad = -quantile(sort(TradReturns),0.01);             % VaR 99%
 ES99_Trad = -mean(sort(TradReturns(TradReturns<-VaR99_Trad)));  % Expected-
Shortfall 99%
 Min_Trad = min(TradReturns);                                % Min
 Max_Trad = max(TradReturns);                                % Max
 cumulative_returns_Trad = cumsum(TradReturns);              % Cumulative
sum of daily returns
 Max_Gain_Trad = max(cumulative_returns_Trad);               % Max gain
 Max_Drawdown_for_Trad = 0;
 Local_Min_Trad = cumulative_returns_Trad(1);
 for i = 2:length(cumulative_returns_Trad)
     if cumulative_returns_Trad(i) > cumulative_returns_Trad(i-1)
         Local_Min_Trad = cumulative_returns_Trad(i);
     else
         drawdown_Trad = Local_Min_Trad - cumulative_returns_Trad(i);
         if drawdown_Trad > Max_Drawdown_for_Trad
             Max_Drawdown_for_Trad = drawdown_Trad;
         end
     end
 end
 Max_Drawdown_Trad = Max_Drawdown_for_Trad;                  % Max
drawdown

 % Passive Strategy
 Mean_Pass = mean(objective_returns);                        % Average
daily returns
 Var_Pass = var(objective_returns);                          % Variance
 SD_Pass = std(objective_returns);                           % Standard
deviation
 Kurt_Pass = kurtosis(objective_returns);                    % Kurtosis
 Skew_Pass = skewness(objective_returns);                    % Skewness
 VaR95_Pass = -quantile(sort(objective_returns),0.05);       % VaR 95%
 ES95_Pass = -mean(sort(objective_returns(objective_returns<-VaR95_Pass)));  %
Expected-SHortfall 95%
 VaR99_Pass = -quantile(sort(objective_returns),0.01);       % VaR 99%
 ES99_Pass = -mean(sort(objective_returns(objective_returns<-VaR99_Pass)));  %
Expected-SHortfall 99%
 Min_Pass = min(objective_returns);                          % Min
 Max_Pass = max(objective_returns);                          % Max
 cumulative_returns_Pass = cumsum(objective_returns);        % Cumulative
sum of daily returns
 Max_Gain_Pass = max(cumulative_returns_Pass);               % Max gain
 Max_Drawdown_for_Pass = 0;
 Local_Min_Pass = cumulative_returns_Pass(1);
 for i = 2:length(cumulative_returns_Pass)
     if cumulative_returns_Pass(i) > cumulative_returns_Pass(i-1)
         Local_Min_Pass = cumulative_returns_Pass(i);
     else
         drawdown_Pass = Local_Min_Pass - cumulative_returns_Pass(i);
         if drawdown_Pass > Max_Drawdown_for_Pass
             Max_Drawdown_for_Pass = drawdown_Pass;
         end
     end
 end
 Max_Drawdown_Pass = Max_Drawdown_for_Pass;                       % Max
drawdown

 % Comparison of Trading and Passive Strategies
 names = {'Mean'; 'Var'; 'Stan. Dev.'; 'Kurt'; 'Skew'; 'VaR95'; 'ES95';
'VaR99'; 'ES99'; 'Min'; 'Max'; 'Max Gain'; 'Max Drawdown'};
 Trading = [Mean_Trad; Var_Trad; SD_Trad; Kurt_Trad; Skew_Trad; VaR95_Trad;
ES95_Trad; VaR99_Trad; ES99_Trad; Min_Trad; Max_Trad; Max_Gain_Trad;
Max_Drawdown_Trad];
 Passive = [Mean_Pass; Var_Pass; SD_Pass; Kurt_Pass; Skew_Pass; VaR95_Pass;
ES95_Pass; VaR99_Pass; ES99_Pass; Min_Pass; Max_Pass; Max_Gain_Pass;
Max_Drawdown_Pass];
 Table_Comp = table(names, Trading, Passive)
```

```
Table_Comp = 13×3 table
```

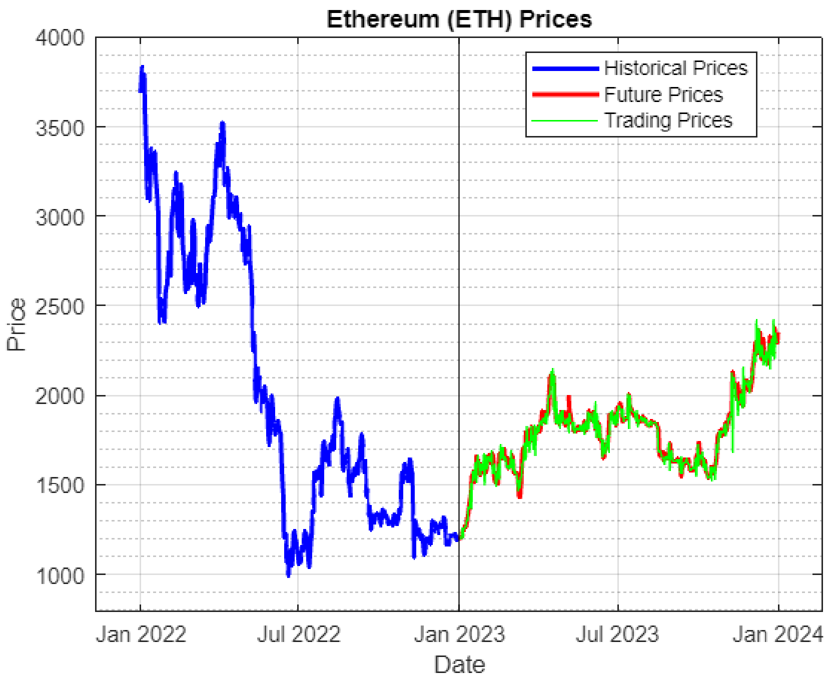
	Variables	Active	Passive
1	Mean	-0.0024	0.0019
2	Var	0.0007	0.0006
3	Stan. Dev.	0.0261	0.0244
4	Kurt	5.5359	5.2958
5	Skew	0.0737	0.3493
6	VaR95	0.0472	0.0406
7	ES95	0.0611	0.0525
8	VaR99	0.0751	0.0629
9	ES99	0.0885	0.0723
10	Min	-0.1181	-0.0827
11	Max	0.1181	0.1181
12	Max Gain	0.2538	0.6862

	Variables	Active	Passive
13	Max Drawdown	0.1990	0.1119

```
 % Plot Future Returns of Trading and Passive Strategy
 plot(TradReturns, "r", 'LineWidth', 2);
 hold on
 plot(objective_returns, col = "blue");
 yline(0, "k");
 title('ETH Returns from Trading Strategy 01-01-2023 to 01-01-2024',
'FontSize', 14, 'FontWeight', 'bold');
 xlabel('Lag', 'FontSize', 12);
 ylabel('Daily Returns', 'FontSize', 12);
 legend('Trading Returns','Real Returns', Location = 'best');
 axis padded;
 grid on;
 grid minor;
 set(gca, 'FontSize', 10);
 box on;
 hold off
```



```
 TradPrice = zeros(length(TradReturns),1);  % Calculate the price from the
trading strategy
 for i = 1:365
     TradPrice(i) = Data{(364+i), 'Price'}*exp(TradReturns(i));
 end
 TradPrice(end+1) = TradPrice(end);

 % Plot Ethereum Prices
 plot(Data.Date(1:365), Data.Price(1:365), 'b', 'LineWidth', 2);
 hold on;
 plot(Data.Date(366:end), Data.Price(366:end), 'r', 'LineWidth', 2);
 plot(Data.Date(366:end),TradPrice, 'g');
 xline(365, "k");
 title('Ethereum (ETH) Prices', 'FontSize', 14, 'FontWeight', 'bold');
 xlabel('Date', 'FontSize', 12);
 ylabel('Price', 'FontSize', 12);
 legend('Historical Prices','Future Prices','Trading Prices', Location =
'best');
 axis padded;
 grid on;
 grid minor;
 set(gca, 'FontSize', 10);
 box on;
 hold off;
```



Linear Regression

The linear regression considers as dependent variable “y”, the vector of the returns of the trading strategy and as independent variable “x” a dummy variable. This one takes the number 0, if the vector takes the value of the return of the passive investment, while takes 1, if the vector takes the value of the return of the trading strategy.

```
 y = TradReturns;    % Returns of the Trading Strategy
 x = zeros(length(TradReturns),1);

 for i = 1:length(TradReturns)
     if y(i) == objective_returns(i)
         x(i) = 0;  % if the return of the trading strategy is equal to the
one of the passive investment
```

```
     else
         x(i) = 1;  % if the return of the trading strategy is equal to the
active strategies returns
     end
 end

 corr_value = corr(x, y);              % correlation coefficient between the
two variables
 disp(['Correlation coefficient between x and y: ', num2str(corr_value)]);
```

```
Correlation coefficient between x and y: -0.30354
```

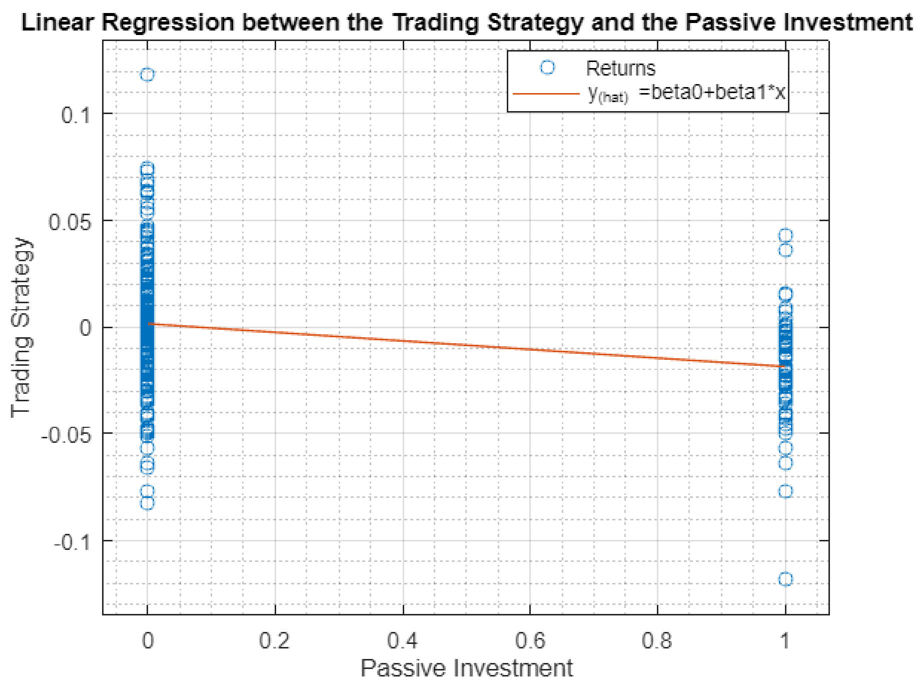
Calculating of regressor coefficients

```
 beta1 = sum((x - mean(x)) .* (y - mean(y))) / sum((x - mean(x)).^2)  %
Estimate β1, slope of regression line
```

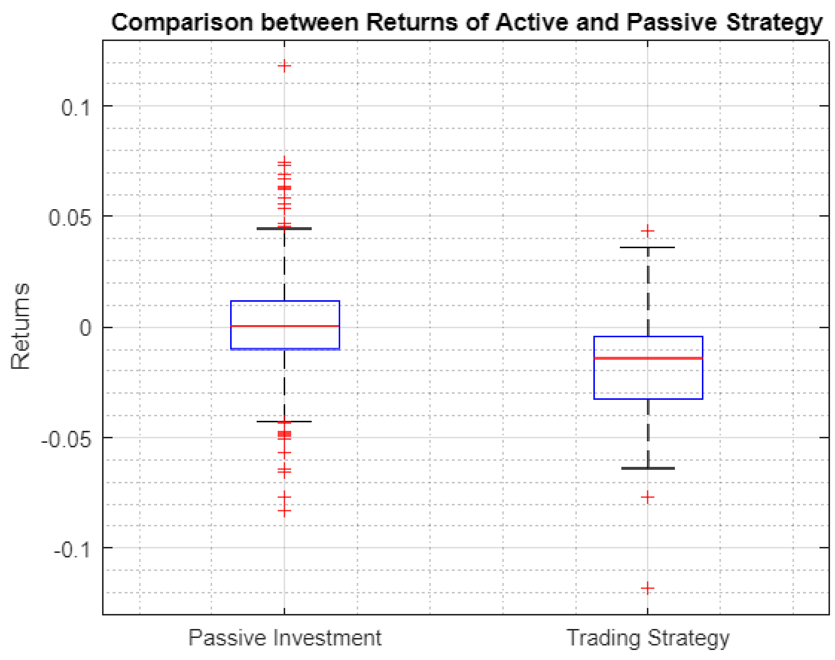
```
beta1 = -0.0199
```

The value of the slope of regression is negative. It means that the active strategy is negatively correlated with higher returns and positively with the lower ones.

```
 beta0 = mean(y) - beta1 * mean(x);                            % Estimate β0
 y_hat = beta0 + beta1 .* x;                                   % Expected
values

 scatter(x, y);    % Linear Regression and Optimal Least Squares
 hold on
 plot(x, y_hat);
 title('Linear Regression between the Trading Strategy and the Passive
Investment', 'FontSize', 14, 'FontWeight', 'bold');
 legend('Returns', 'y_(hat) =beta0+beta1*x', Location = 'best');
 xlabel('Passive Investment', 'FontSize', 12);
 ylabel('Trading Strategy', 'FontSize', 12);
 grid on;
 grid minor;
 axis padded;
 set(gca, 'FontSize', 10);
 box on;
 hold off
```



```
 % Box plot
 allValues = [y(x == 0); y(x == 1)];     % Combining all the values in one
vector to display the boxplot
 groupLabels = [repmat({'Passive Investment'}, length(y(x == 0)), 1);
repmat({'Trading Strategy'}, length(y(x == 1)), 1)];   % Creating tags for
dummies
 boxplot(allValues, groupLabels);
 title('Comparison between Returns of Active and Passive Strategy',
'FontSize', 14, 'FontWeight', 'bold');
 ylabel('Returns', 'FontSize', 12);
 grid on;
 grid minor;
 set(gca, 'FontSize', 10);
 box on;
```



Goodness of Fit

```
 ESS = sum((y_hat - mean(y)).^2); % Explained Variance
 RSS = sum((y - y_hat).^2);       % Residual Variance
 TSS = ESS + RSS;                 % Total Variance
 Rsquare = ESS / TSS;             % R-square
 RsquareAdj = 1 - ((RSS / (length(y) - 2)) / (TSS / (length(y) - 1)));  % R-
square adjusted
 disp(['R-square: ', num2str(Rsquare)]);
```

```
R-square: 0.092136
```

```
 disp(['Adjusted R-square: ', num2str(RsquareAdj)]); % The R square is pretty
low
```

```
Adjusted R-square: 0.089635
```

Significance test for beta1 (T-test)

```
 MSE = RSS / (length(y) - 2);     % Mean-Squared-Error
 SMSE = sqrt(MSE);                % Standardized Mean-Squared-Error
 std_beta1 = SMSE / sqrt(sum((x - mean(x)).^2));  % Standard deviation of
beta1
 t_stat = beta1 / std_beta1;      % T-statistic value for beta1
 p_value = 2 * (1 - tcdf(abs(t_stat), length(y) - 2));  % P-value for beta1
 disp(['t-statistic for beta1: ', num2str(t_stat)]);
```

```
t-statistic for beta1: -6.0696
```

```
 disp(['p-value for beta1: ', num2str(p_value)]);
```

```
p-value for beta1: 3.2304e-09
```

H0: There is no significant difference in average returns between the active and passive strategy.

H1: There is a significant difference in average returns between the active and passive strategy.

```
if p_value < 0.05
    disp("Reject the null hypothesis: There is a significant difference in average returns between the active and passive strategy for ETH.");
else
    disp("Fail to reject the null hypothesis: There is no significant difference in average returns between the active and passive strategy for ETH.");
end
```

```
Reject the null hypothesis: There is a significant difference in average returns between the active and passive strategy for ETH.
```

Since the null hypothesis is rejected and the value of beta1 is negative, there is a significance difference in average returns between the active and passive strategy for ETH. In this context it means that the active strategy has a lower performance than the passive strategy for ETH because it is positively correlated with lower returns.

```
fitlm(x,y,'linear') % Check the result
```

```
ans = 

Linear regression model:

    y ~ 1 + x1


Estimated Coefficients:

                   Estimate       SE         tStat       pValue  

                   _________    _________    _______    __________


    (Intercept)    0.0015467    0.0014573     1.0613       0.28925

    x1             -0.019915    0.0032812    -6.0696    3.2304e-09



Number of observations: 365, Error degrees of freedom: 363

Root Mean Squared Error: 0.0249

R-squared: 0.0921, Adjusted R-Squared: 0.0896

F-statistic vs. constant model: 36.8, p-value = 3.23e-09
```

Loglikelihood function

```
function loglikelihood = garch_loglikelihood(theta)
global epsilon
sigma_hat2(1) = var(epsilon);   % initial value of sigma^2
loglikelihood = log(sigma_hat2(1)) + (epsilon(1)^2 / sigma_hat2(1));  % initial value of log-likelihood
for i = 2:length(epsilon)
```

```
     sigma_hat2(i) = theta(1) + theta(2) * epsilon(i-1)^2 + theta(3) *
sigma_hat2(i-1);    % Conditional variance
     loglikelihood = loglikelihood + log(sigma_hat2(i)) + (epsilon(i)^2 /
sigma_hat2(i));  % Likelihood
 end
 end
```

References:

- https://www.kucoin.com/price/ETH
- https://coinmarketcap.com/currencies/ethereum/historical-data/
- https://www.statista.com/statistics/806453/price-of-ethereum/
- https://www.oxfordreference.com/