Crypto Factor Modeling Data Challenge

Multi-Factor Risk Models for Cryptocurrency Price Variance:

Knock Knock, Are You There Aladdin?[1]

Firuze Simay Sezgin
Department of International Relations, Bilkent University, Ankara, Turkey
Corresponding author. firuze.sezgin@bilkent.edu.tr
ORCID: 0000-0002-4319-8315

Yunus Gümüşsoy
yunusgumussoy@icloud.com
Independent Researcher, Ankara, Turkey

2024

[1] Aladdin® by BlackRock - software for portfolio management.

Crypto Factor Modeling Data Challenge

Abstract

To predict cryptocurrency price variance, we constructed a custom dataset encompassing Bitcoin and Ethereum prices along with on-chain metrics, technical indicators, macroeconomic indexes, global economic volatility metrics. We employed a range of multi-factor risk models, including Linear Regression variants with Ridge, Lasso, ElasticNet models, as well as advanced models such as Long Short-Term Memory (LSTM) and eXtreme Gradient Boosting (XGBoost). To enhance parameter optimization and model discovery, we utilized Automated Machine Learning (AutoML) tools such as T-POT, MLJAR, and H2O.ai. Based on AutoML tools' recommendations, we developed and tested hybrid models such as RidgeCV combined with DecisionTreeRegressor and RidgeCV with ElasticNet. Further, we constructed a stacked ensemble model including Decision Tree, Gradient Boosting (GBM), Support Vector Regression (SVR), and RidgeCV. We also separately evaluated GBM and LightGBM models to assess their predictive accuracy. Moreover, we conducted stress testing using the LightGBM model under a simulated market crash scenario, assuming a 30% price drop in a single day. To further assess our models' performance under different economic conditions, such as interest rate fluctuations, we ran Monte Carlo simulations by repeatedly sampling from the distributions of our input variables. Due to the meticulous curation of our dataset – comprising variables meticulously selected through an extensive literature review and informed by domain expertise – all tested models demonstrated satisfactory performance. Notably, models recommended by AutoML tools have outperformed others in predictive accuracy. Specifically, the LSTM and XGBoost models have demonstrated exceptional capabilities in forecasting real-world prices. We observed the critical importance of hyperparameter tuning across all tested models, primarily due to its enhancing effect. Our analysis explores the impact, usability, and real-world applicability of these models. We also discuss the limitations of our study and provide recommendations for further research. In alignment with principles of transparency and accountability, we share our data and scripts via GitHub repository.

1. Introduction

To develop multi-factor risk models that explain cryptocurrency price variance, we constructed a comprehensive dataset by sourcing data from numerous sources on cryptocurrency exchanges, on-chain metrics, global markets, macroeconomic indices, and even international relationships trends. This extensive dataset, comprising over 206 variables, enables a detailed and multifaceted analysis of cryptocurrency price variance.[2] To ensure our dataset accurately reflects real-world market conditions and maximizes the explanatory power of our models, we adhered to a rigorous process for data collection, selection, and design. Our goal is to create a robust and representative dataset that captures the evolving dynamics of today's financial markets. Leveraging a range of tools including Python with libraries such as Pandas, NumPy, seaborn and scikit-learn, our model focuses on thorough data processing, model training, evaluation, and simulations. Rather than reinventing the wheel, our focus is on discovering, enhancing, and fine-tuning existing models to meet the complex demands of today's financial markets.

In the following sections, we detail our methodology, the rationale behind our data collection and processing strategies, algorithm selection, and the practical aspects of implementation. Our aim is to develop sophisticated multi-factor risk models to effectively explain and predict cryptocurrency price variance.

2. Data Collection, Quality Assurance, and Exploration

2.1 Data Collection

We sourced a diverse range of data from platforms including Yahoo Finance, CryptoQuant, coinglass, bitcoinvisuals, CoinMarketCap, CoinGecko, TradingView, Economic Policy Uncertainty, and alternative.me. This allowed us to gather comprehensive information on cryptocurrency prices, trading volumes, blockchain statistics, network activities and nodes, transaction data and hash rates, currency data, market capitalization, dominance, exchange activity, global indices, and market pairs. To manage and analyze this extensive dataset, we employed data science tools such as Stata, Excel, Spyder, Jupyter Notebook and Visual Code Studio, primarily using Python.

Our on-chain metrics cover network activities such as transaction fees and speed, miner revenues, volume, hash rate, and node metrics. For macroeconomic indicators, we included United States (US) interest rates and inflation rates, as well as global indices such as CAC40, Nikkei225, DAX, the Emerging Markets Index, and the Volatility Index (VIX). Additionally, we utilized world uncertainty measures, including the World Uncertainty Index (WUI), the World Trade Uncertainty Index (WTUI), the World Pandemic Uncertainty Index (WPUI), and the US Equity Market Volatility Index, spanning about 45 specific categories to capture diverse sources of risk.

In the subsequent section, we discuss our feature engineering efforts, where we created variables such as lagged prices and volumes, simple and exponential moving averages, and various technical indicators, complementing the collected data.

[2] https://github.com/yunusgumussoy/Crypto-Factor-Modeling-Data-Challenge

2.1.1 Feature Engineering

To build a high-quality, accurate, and comprehensive dataset, we not only explored unique data sources and novel data collection methods, but also engineered features that add significant predictive value.

First, we began by recognizing the critical role that lagged and shifted prices in predicting current prices, especially in financial time series data like ours. This approach is a commonly used practice in time series forecasting and modeling, where past values of a variable are used to predict its future values.[3][4][5] Financial time series, like cryptocurrency prices, often exhibit *autocorrelation*, meaning past prices are correlated with future prices. By creating and including lagged prices into our dataset, we aimed to capture and utilize these temporal relationships effectively.

Second, we incorporated Moving Averages as a fundamental component of our dataset. We included both short-term moving averages (e.g., 5-day and 10-day) and long-term moving averages (e.g., 200-day) to smooth out price data and highlight trends over different time horizons. Moreover, we engineered lagged volatility measures, such as standard deviation and Average True Range (ATR), and integrated them into our dataset. These measures are essential for understanding market stability and the degree of price fluctuations, helping us capture the inherent volatility of cryptocurrency markets.

To further enrich our dataset, we utilized the following variety of momentum indicators. Rate of Change (ROC) measures the percentage change between the current price and a previous price, helping to identify the speed and strength of price movements. The Relative Strength Index (RSI) is used to assess overbought or oversold conditions in the market by comparing the magnitude of recent gains to recent losses. Moving Average Convergence Divergence (MACD) shows the relationship between two moving averages of prices, offering insights into potential buy or sell signals. Bollinger Bands are plotted with two standard deviations away from a simple moving average, indicating volatility and potential price breakouts. We also incorporated trend indicators, such as the Average Directional Index (ADX), which measures the strength of a trend regardless of its direction. These indicators are crucial for identifying market momentum and trends, providing a more comprehensive analysis of cryptocurrency price movements.

The selection of appropriate window sizes was an integral part of our feature engineering process. We chose specific window sizes to capture different market behaviors. The 20-Day and 30-Day Window Sizes correspond roughly to one trading month (assuming approximately 20 trading days per month), making them ideal for analyzing and comparing monthly performance that captures medium-trends and volatility. A 30-day window strikes a balance between being responsive to recent changes and smoothing out short-term fluctuations. It is long enough to filter out noise but short enough to react to significant changes in return patterns. Based on our literature review, experience, and industry practices, many financial

[3] Abdollahi, H., & Ebrahimi, S. B. (2020). A new hybrid model for forecasting Brent crude oil price. Energy, 200, 117520.
[4] Suaza-Medina, M. E., Zarazaga-Soria, F. J., Pinilla-Lopez, J., Lopez-Pellicer, F. J., & Lacasta, J. (2023). Effects of data time lag in a decision-making system using machine learning for pork price prediction. Neural Computing and Applications, 35(26), 19221-19233.
[5] Abdollahi, H. (2020). A novel hybrid model for forecasting crude oil price based on time series decomposition. Applied energy, 267, 115035.

analysts and researchers prefer 30-day (or 1-month) rolling windows for various metrics, including volatility and Sharpe Ratios, as these align well with common investment and reporting periods.[6]

We also used 7-day window for very short-term analysis and high-frequency trading strategies, capturing immediate market reactions. 14-days window was employed to identify momentum and mean-reversion strategies, offering a balance between short-term responsiveness and reliability.[7] The 50-day window provides a longer-term perspective, smoothing out more noise and capturing more extended trends. The 100-day and 200-day windows were used for quarterly and semi-annual analysis, providing a more stable view of performance over longer periods, which is crucial for understanding long-term market trends.[8]

Finally, recognizing the highly emotional nature of the cryptocurrency market – where investors often act irrationally due to fear or greed (greed when the market is rising which results in FOMO (Fear of missing out) or fear of losing by seeing red numbers and sell coins) – we decided to incorporate the Fear & Greed Index from alternative.me into our dataset. The inclusion of this index was driven by several factors. First, the index offers a simple and intuitive measure of Bitcoin market sentiment, ranging from 0 ("Extreme Fear") to 100 ("Extreme Greed"). This straightforward metric allows us to quickly gauge the overall mood of the Bitcoin market. Second, the index is composed of several elements, including volatility, market momentum/volume, social media sentiment (particularly Twitter analysis), Bitcoin dominance, and market cap. Finally, the index incorporates Google Trends data, analyzing Bitcoin related search queries, especially the change of search volumes as well as recommended other currently popular searches. By integrating the Fear & Greed Index, we aimed to account for the psychological factors that often drive market movements, adding an additional layer of insight to our predictive models.

To ensure transparency and reproducibility, we have documented our entire feature engineering process. The full code for this process is available in our GitHub repository,[9] allowing for reviewing the feature engineering codes and replication. Additionally, Appendix A includes a comprehensive data codebook that provides detailed explanations of each variable and its relevance to cryptocurrency price movements. We also provided clear justifications for the selection of each feature, explaining their importance in capturing the dynamics of the cryptocurrency market. Through this detailed and methodical feature engineering process, we have constructed a robust dataset designed to support the development of accurate and insightful multi-factor risk models for predicting cryptocurrency price variance.

2.2 Data Quality

Ensuring the accuracy, completeness, and timeliness of our data was a critical aspect of our process. We validated our data by cross-referencing it with alternative sources to verify its reliability. Given the high-dimensional nature of our dataset, we prioritized preserving its historical depth by capturing the earliest possible data points for each feature. This approach

[6] Marchesi, F. (2024). Redefining Financial Forecasting: A CAPE-Based Approach to S&P 500 Analysis and Portfolio Construction.
[7] Baltas, A. N., & Kosowski, R. (2012). Improving time-series momentum strategies: The role of trading signals and volatility estimators. SSRN eLibrary.
[8] Kaourma, T. C. (2020). Essays on retail investors trading behavior in FX markets.
[9] https://github.com/yunusgumussoy/Crypto-Factor-Modeling-Data-Challenge

was aimed at mitigating the risk of *underfitting* and *overfitting* in our models, ensuring they are well-calibrated and generalizable across different time periods and market conditions.

2.3 Exploratory Data Analysis (EDA)

We conducted a thorough Exploratory Data Analysis (EDA) to gain insights into the structure and quality of our dataset. Our EDA process involved implementing robust data exploration techniques to detect and handle missing values, remove duplicates, and correct any inconsistencies.[10] We visualized missing data using heatmaps, which allowed us to quantify the extent and distribution of missing values for each feature, providing a clear understanding of potential data quality issues.

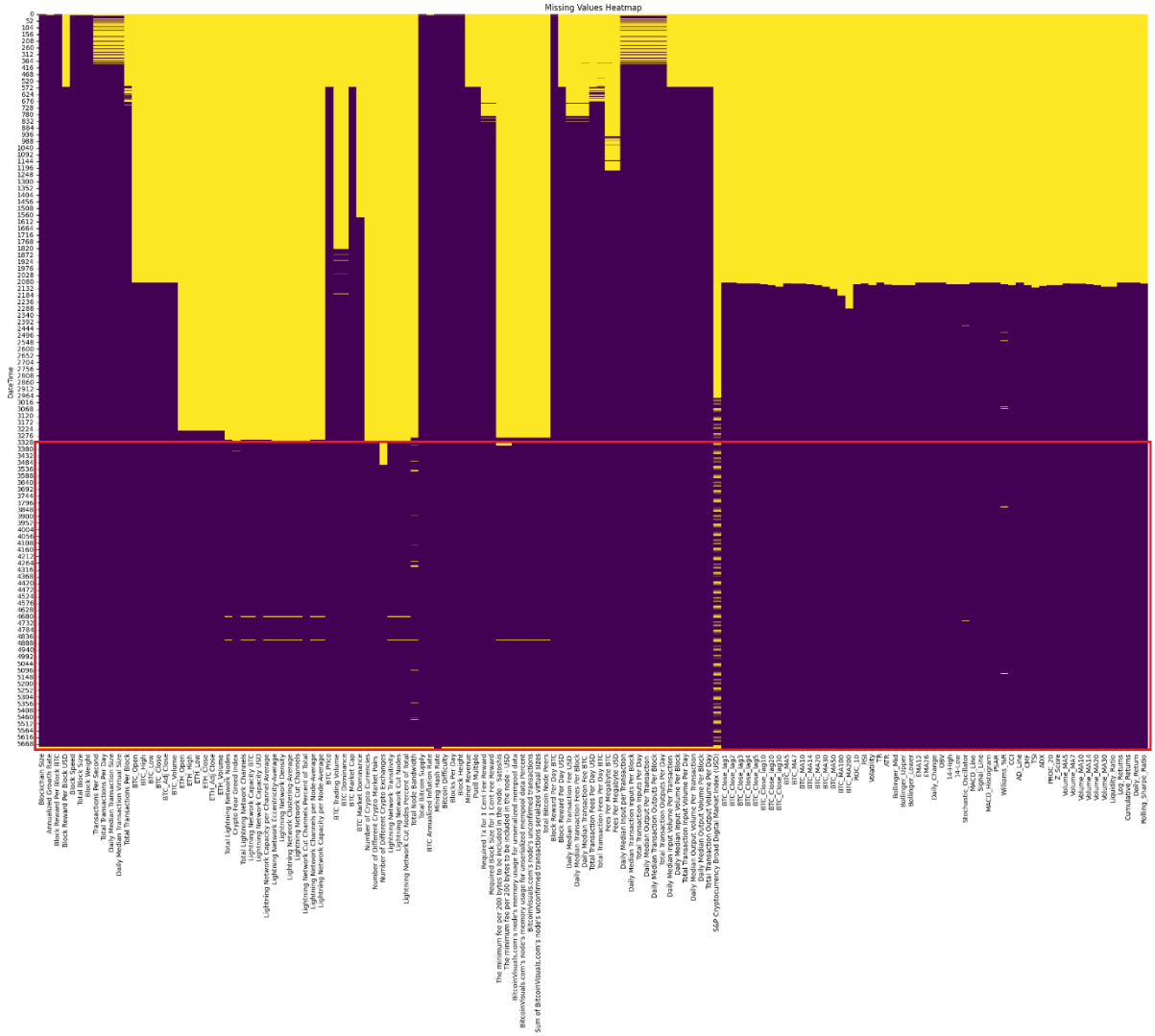
For engineered features such as lagged prices and various calculated indexes, we encountered instances where some data points resulted in zero values, representing missing data. To address this issue, we treated zero values as missing data. Additionally, NaN values generated by operations like shifting and applying rolling windows were also classified as missing values.

This project has been a learning experience, particularly in handling missing data within our dataset. For instance, we discovered that when replacing zeroes with Pandas Library (with pd.NA code), the data type of columns was inadvertently changed from float to object. This change complicated the process of plotting numerical variables and performing further analysis. To resolve this, we opted to replace zeroes using NumPy library (with np.nan code), as np.nan is more commonly used with numerical types and does not convert the column's data type.

However, for other features including indexes and certain observations, we retained the zeroes. This distinction was crucial, especially for time series exploratory analysis, where data types and missing values can cause failures in stationarity tests, visualizations, autocorrelation plots, and partial autocorrelation plots. By carefully managing these data quality issues, we ensured that our dataset was robust and suitable for the advanced analyses required in this project.

[10] https://github.com/yunusgumussoy/Crypto-Factor-Modeling-Data-Challenge/blob/main/1-Data-Exploration.ipynb

Figure 1. *Missing Values Heatmap.*



Additionally, we identified and addressed outliers for every feature that could potentially distort our models. Recognizing the impact of outliers on the accuracy and reliability of our results, we ensured that each was carefully examined and managed. We also checked for constant columns in our dataset, identifying any columns with a standard deviation of zero, which could indicate that they contain the same value across all rows. Such columns can be problematic as they do not contribute any variance to the model, and their presence can skew the results.

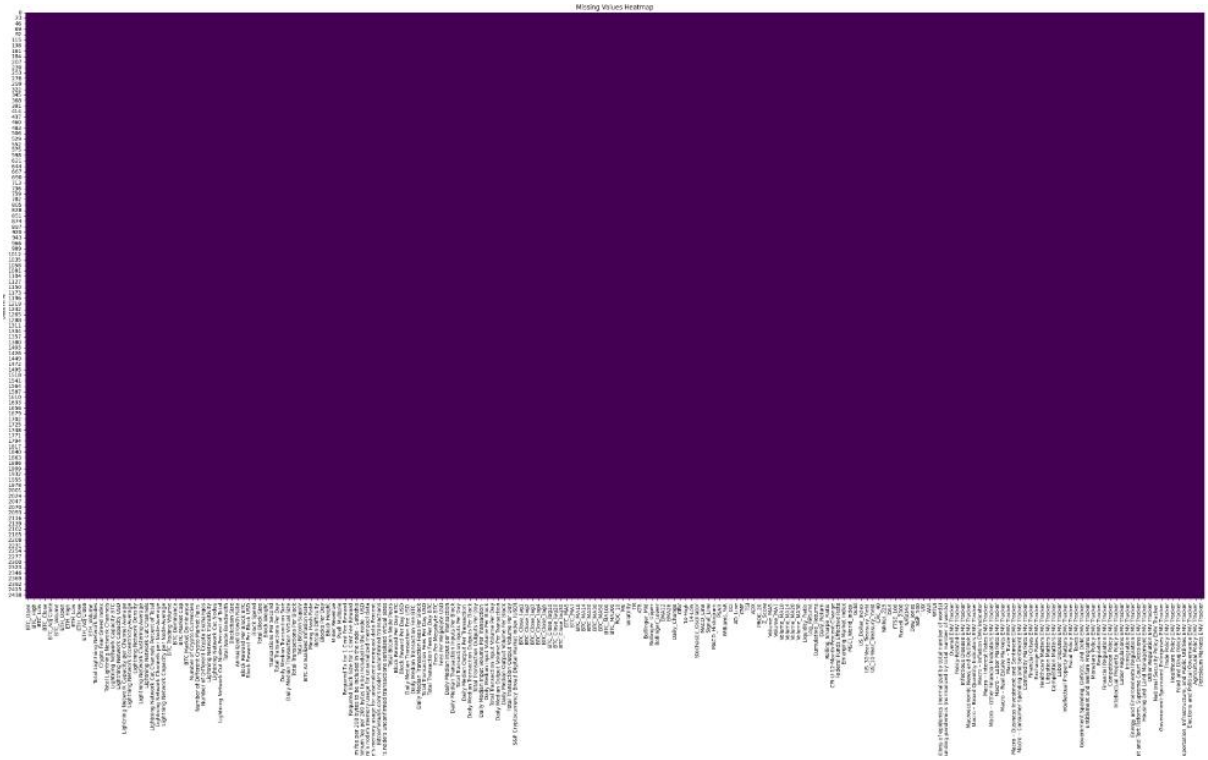
Missing values and outliers are common in financial data, often resulting from global or local economic shocks, health crises, armed conflicts, or natural disasters. During our review of missing values, we detected that many gaps in the data were due to holidays or non-trading days, which is typical in financial datasets.

Given that most exploratory tests – including distribution checks, skewness and kurtosis calculations, outlier checks, correlation analysis, stationarity tests, autocorrelation plots, and cointegration tests – require complete data, we prioritized handling missing values before

conducting the analyses. As seen in Figure 1, we selected a specific segment of our dataset (highlighted in red) to proceed with for our analyses.

To address missing values, we decided to use the *forward fill* method, a widely accepted approach in financial data analysis that assumes continuity during non-trading days. This method involves replacing missing entries with the last available value, under the assumption that the market did not experience significant changes during the missing period. However, as there were still some missing values after applying forward fill, we supplemented this approach with *backward fill*, which fills in the gaps using the next valid observation. As illustrated in Figure 2, the cover of the dataset significantly improved after applying the filling methods. By carefully managing outliers and missing values, we aimed to maintain the integrity of our dataset, ensuring that it was robust enough to support the rigorous analyses and modeling required for this project.

Figure 2. *Missing Values Heatmap after Handling.*



To guide our subsequent processes, such as *standardization* and *normalization* during model selection, we examined the distributions of our features. For instance, our approach was informed by prior knowledge and experience: when features are approximately normally distributed, standardization is typically more effective. Conversely, if features are not normally distributed or have different ranges, normalization might be more appropriate. Therefore, we assessed the distribution of each variable independently to make well-informed decisions throughout the project.

To this end, we utilized histograms and boxplots to analyze distributions and identify outliers in Figure 3 and 5. This was crucial because significant outliers can distort normalization by compressing the range based on extreme values. In contrast, standardization, which centers

data around the mean and scales it by the standard deviation, is generally more robust to outliers. To further refine our understanding, we calculated and visualized the *skewness* and *kurtosis* for each feature, providing insights into the symmetry of the distributions in Figure 4.

Additionally, scatterplots in Figure 6 were employed to explore the relationships between pairs of variables. This visual examination helped us identify potential correlations and interactions, further informing our choice of whether to apply standardization or normalization. By rigorously analyzing these aspects, we ensured that our data preprocessing choices were tailored to the specific characteristics of our dataset, setting a solid foundation for accurate and reliable model performance.

Figure 3. *Histograms of Features.*

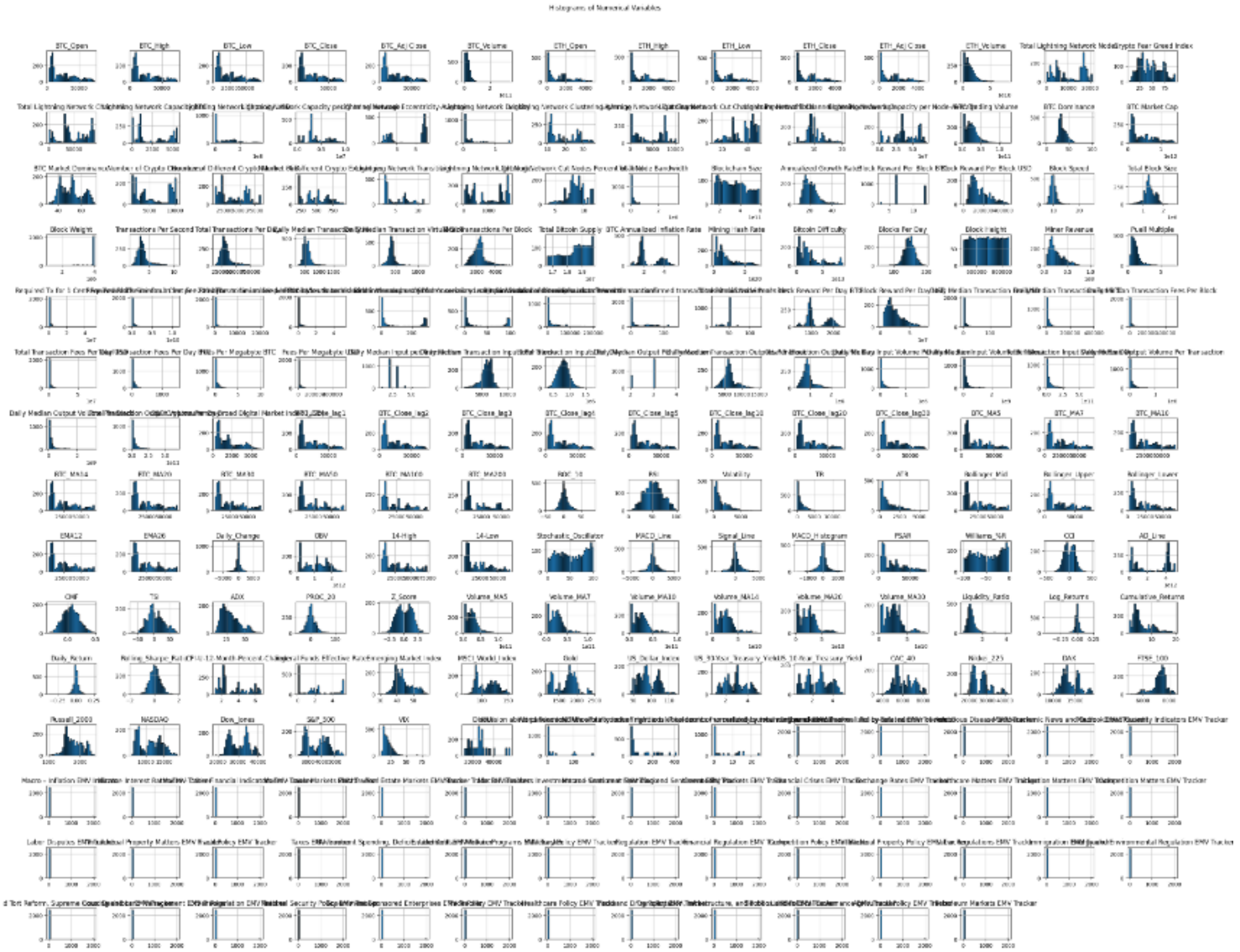


Figure 4. Skewness And Kurtosis for Each Feature.

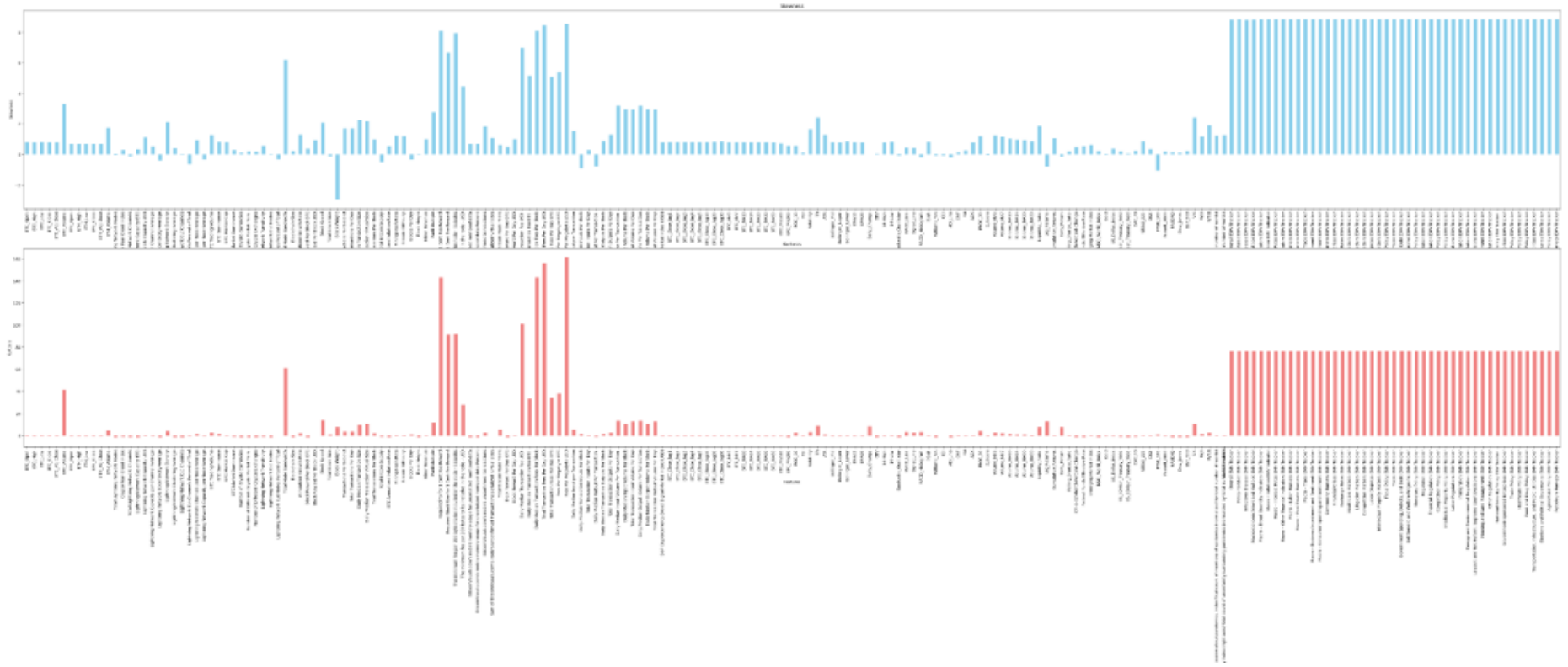


Figure 5. *Identify Potential Outliers in Numerical Features.*

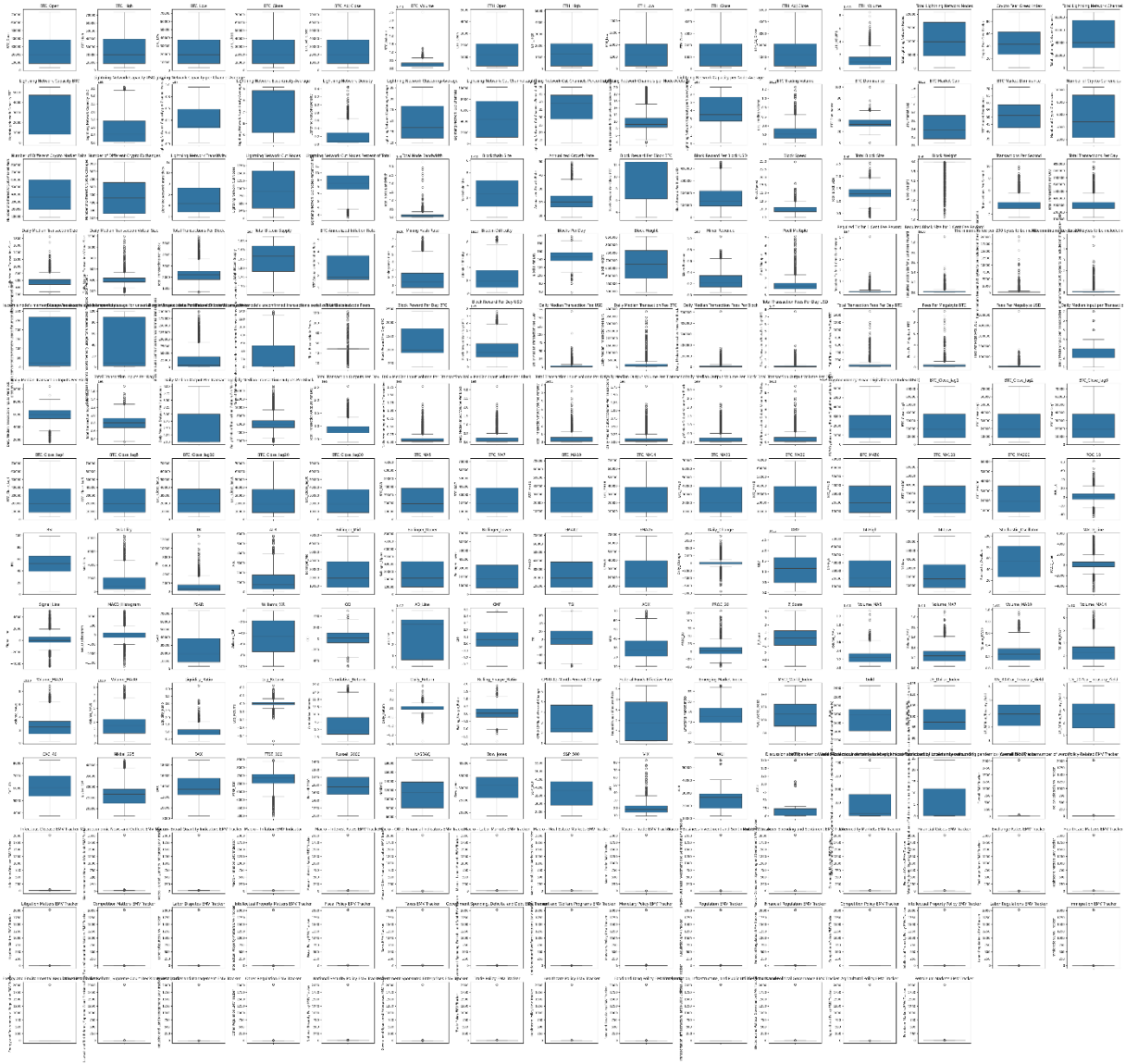
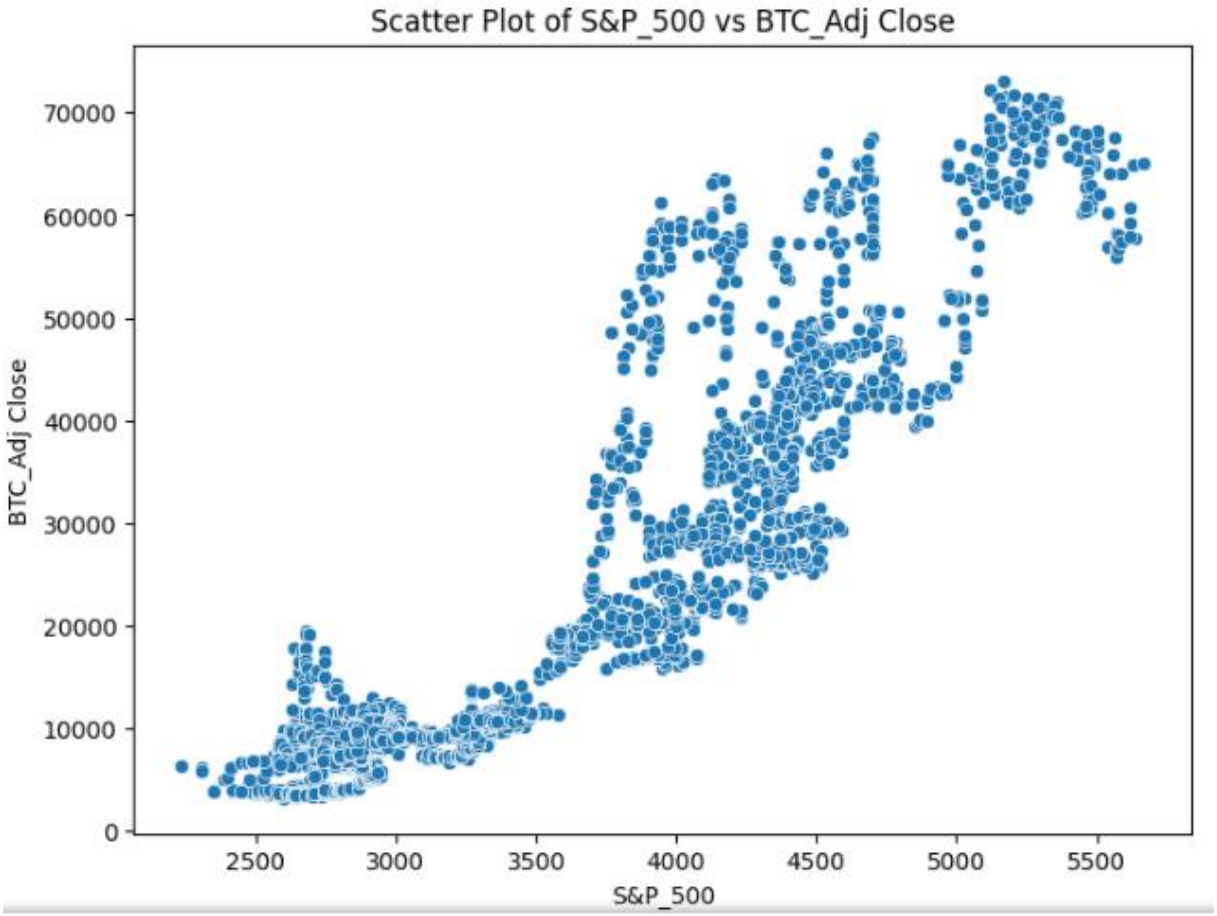


Figure 6. *Scatter Plots for Bivariate Analysis (S&P500 vs. BTC_Adj Close).*



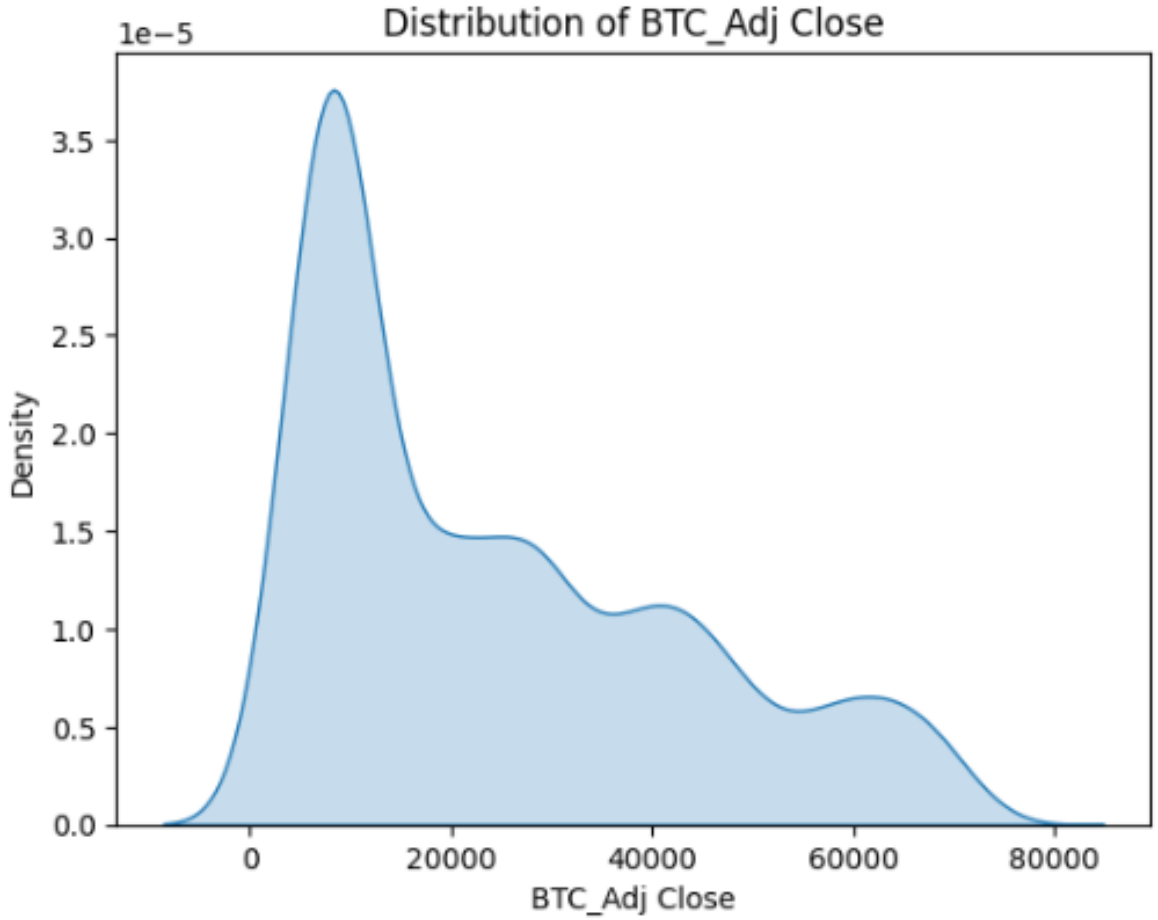
Please note that only one variable's plot is presented here for the sake of brevity and clarity. For additional plots of other variables, please refer to the data and code snippet available in the footnote.[11]

To detect multimodality (multiple peaks) in the variables, we plotted Kernel Density Estimate (KDE) plots for each numerical variable in our dataset (Figure 7). Multimodal distributions can reveal the presence of distinct subpopulations within the data, which can be critical for two reasons: (1) guiding further segmentation or clustering tasks, and (2) informing feature engineering by either creating features that capture these peaks or splitting the variable into different categories.

Only one variable's plot is shown here to maintain brevity and enhance readability. For further exploration of additional variables' plots, please refer to the data and code snippet provided in the footnote.[12]

[11] https://github.com/yunusgumussoy/Crypto-Factor-Modeling-Data-Challenge/blob/main/1-Data-Exploration.ipynb
https://github.com/yunusgumussoy/Crypto-Factor-Modeling-Data-Challenge/blob/main/Dataset.xlsx
[12] https://github.com/yunusgumussoy/Crypto-Factor-Modeling-Data-Challenge/blob/main/1-Data-Exploration.ipynb
https://github.com/yunusgumussoy/Crypto-Factor-Modeling-Data-Challenge/blob/main/Dataset.xlsx

Figure 7. *Kernel Density Estimate (KDE) plot for BTC_Adj Close.*



To explore the interactions between multiple variables, we visualized the *correlation matrix* to identify highly correlated pairs in Figure 8. By analyzing this matrix and the corresponding *correlation coefficients,* we considered dropping features that exhibited high correlation with others (e.g., a correlation above 0.9) to mitigate the risk of *multicollinearity*. Multicollinearity can distort the significance of individual predictors in a model, leading to unreliable estimates and reduced interpretability.

In Figure 9, we focused on extracting the correlation values of the target variable, *BTC_Adj Close*, to create a heatmap that highlights its relationship with other features. This targeted visualization allowed us to identify which variables had the strongest associations with our target, guiding our feature selection and model refinement processes. During our data exploration, we discovered that some variables were redundant – specifically, BTC_Close was identical to BTC_Adj Close, and ETH_Close was identical to ETH_Adj Close. As a result, we excluded BTC_Close and ETH_Close from the final dataset to streamline our analysis.

Figure 8. *Correlation Matrix*.

When working with time series data, especially in a multi-factor risk model such as ours, it is crucial to assess *stationarity* not only for the target variable but also for key explanatory variables. Stationarity means that the statistical properties of a time series, such as mean, variance, and autocorrelation, remain constant over time. Achieving stationarity can be challenging, especially with real-time finance data like cryptocurrencies, which are characterized by high volatility.

To address this, we employed both the Augmented Dickey-Fuller (ADF) test and the Kwiatkowski-Phillips-Schmidt-Shin (KPSS) test to evaluate stationarity. The KPSS test is often used in conjunction with the ADF test. The ADF test helps determine the presence of a unit root, indicating non-stationarity, while the KPSS test assesses whether the series is stationary around a deterministic trend. Using these tests in tandem provides a comprehensive view of the data's stationarity, allowing us to identify variables that require differencing or transformation.

Our analysis revealed that most of our variables are non-stationary, necessitating further differencing or transformation to meet model requirements (Figures 10 and 11). By applying these rigorous tests, we ensured that our time series data was appropriately preprocessed, enhancing the reliability and accuracy of our multi-factor risk model.

Figure 9. *Correlation Matrix with the Target Variable.*

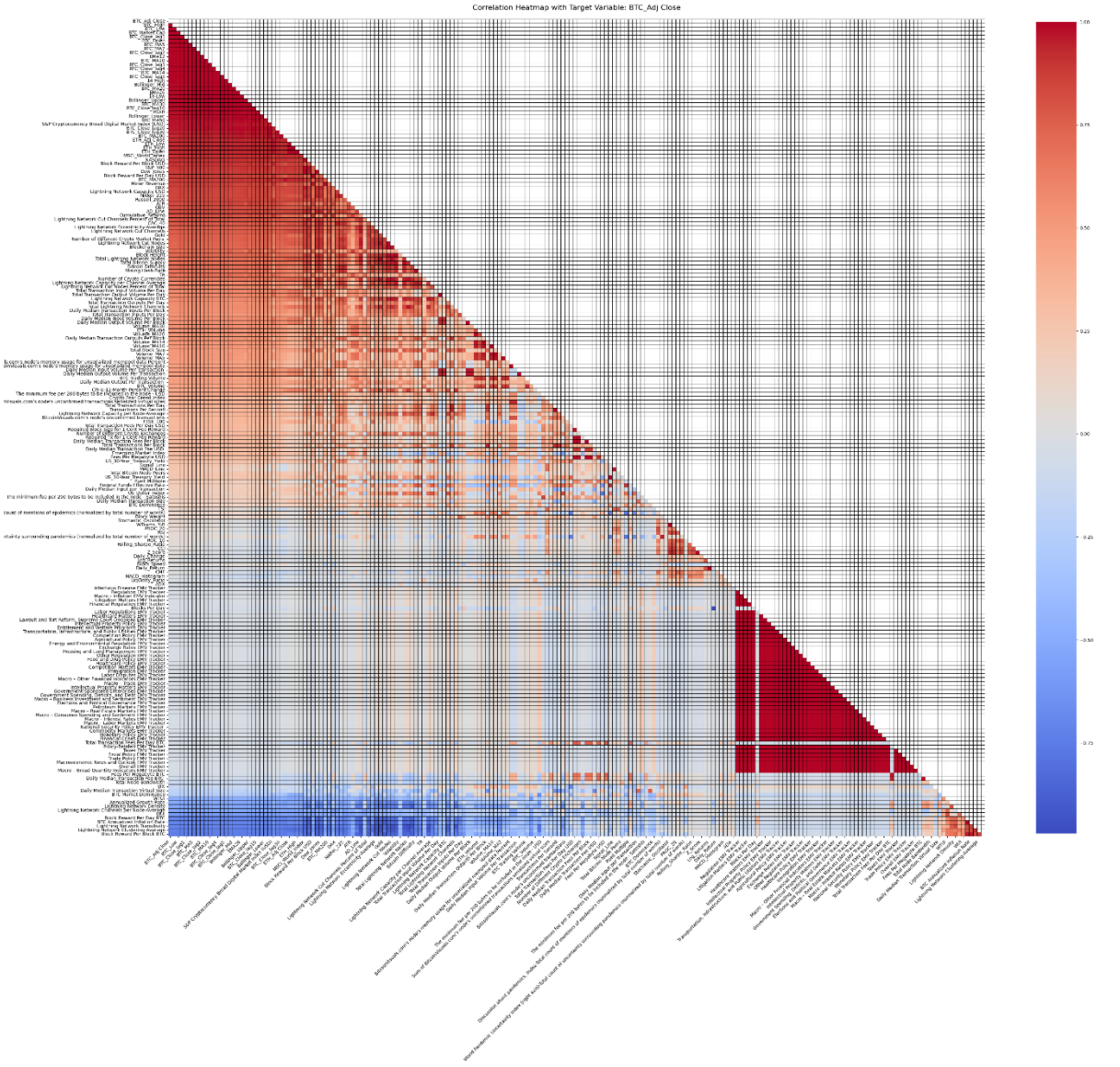


Figure 10. *Results of Augmented Dickey-Fuller Test for BTC_Adj Close and ETH_Adj Close.*

```
Results of Augmented Dickey-Fuller Test for BTC_Adj Close:
ADF Test Statistic        -0.722533
p-value                    0.840826
# Lags Used               14.000000
# Observations Used     2439.000000
Critical Value (1%)       -3.433034
Critical Value (5%)       -2.862726
Critical Value (10%)      -2.567401
dtype: float64
Non-Stationary
------------------------------------------------------------
Results of Augmented Dickey-Fuller Test for ETH_Adj Close:
ADF Test Statistic        -1.206240
p-value                    0.670906
# Lags Used               17.000000
# Observations Used     2436.000000
Critical Value (1%)       -3.433037
Critical Value (5%)       -2.862727
Critical Value (10%)      -2.567402
dtype: float64
Non-Stationary
------------------------------------------------------------
```

Figure 11. *Results of KPSS Test for BTC_Adj Close and ETH_Adj Close.*

```
Results of KPSS Test for BTC_Adj Close:
KPSS Test Statistic       4.350204
p-value                   0.010000
# Lags Used              30.000000
Critical Value (10%)      0.347000
Critical Value (5%)       0.463000
Critical Value (2.5%)     0.574000
Critical Value (1%)       0.739000
dtype: float64
Non-Stationary
------------------------------------------------------------
Results of KPSS Test for ETH_Adj Close:
KPSS Test Statistic       4.493554
p-value                   0.010000
# Lags Used              30.000000
Critical Value (10%)      0.347000
Critical Value (5%)       0.463000
Critical Value (2.5%)     0.574000
Critical Value (1%)       0.739000
dtype: float64
Non-Stationary
------------------------------------------------------------
```

We only presented the test results for two variables for brevity and clarity. For further exploration of other variables' tests results, please refer to the data and code snippet provided in the footnote.[13]

[13] https://github.com/yunusgumussoy/Crypto-Factor-Modeling-Data-Challenge/blob/main/1-Data-Exploration.ipynb
https://github.com/yunusgumussoy/Crypto-Factor-Modeling-Data-Challenge/blob/main/Dataset.xlsx

Differencing is a technique used to remove trends or seasonal components from time series data to achieve stationarity. While this process can be effective in stabilizing the statistical properties of the series, it also has the potential to eliminate significant signals, leading to a loss of crucial information. This can make the data noisier and may compromise the model's ability to make accurate predictions. Similarly, transformations are often applied to stabilize variance, normalize the distribution of data, or address non-linear relationships. However, these transformations can also distort the underlying relationships between variables and the target, potentially diminishing the predictive power of the model.

We prioritized maintaining the integrity of our data, carefully considering the implications of non-stationarity while selecting our models. To address these challenges, we opted for models that are less sensitive to non-stationarity.

We visualized the stationarity of the data by decomposing it into its trend, seasonal, and residual components. This visualization provided a clear view of the time series components, helping us to determine the appropriate handling of stationarity.

In our plots displayed in Figure 12, the Original Series represents the raw time series data. The Trend Component highlights the long-term movement in the data – whether upward, downward, or flat. A strong and clear trend indicates that the series is non-stationary and may require differencing to achieve stationarity.

The Seasonal Component displays the repeating patterns at fixed periods (e.g., daily, monthly, yearly). A pronounced seasonal pattern suggests that seasonal differencing might be necessary. Finally, the Residual Component, which remains after removing the trend and seasonal components, should ideally be stationary with constant mean and variance. However, if the residuals are not stationary, additional transformations might be required. We deferred specific actions to address non-stationarity, such as differencing, log transformations, or detrending, to the model selection phase.

We also employed Autocorrelation Plot (ACF) and Partial Autocorrelation Plot (PACF) to further analyze the data. The ACF shows the correlation of the time series with its past values (lags), helping to identify repeating patterns, measure the extent of autocorrelation and how past values influence future values (Figure 13). The PACF displays the correlation between the time series and its past values after removing the effects of intermediate lags (Figure 14). This plot is instrumental in determining the order of autoregressive models, as it summarizes the direct relationship between observations at different lags while excluding the influence of intervening observations.

Figure 12. *Stationarity Visualization of BTC_Adj Close.*

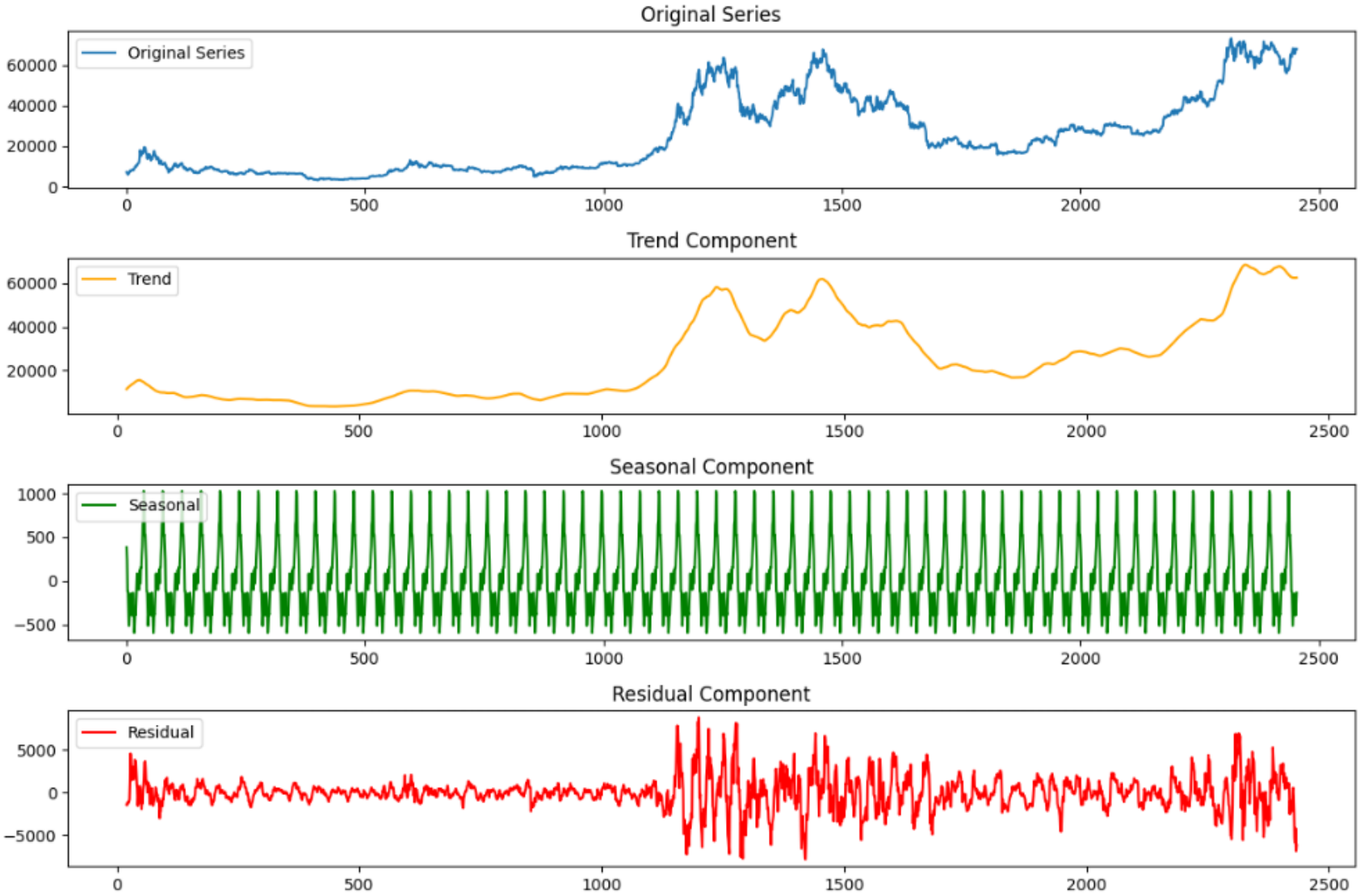


Figure 13. *Autocorrelation plot of BTC_Adj Close.*

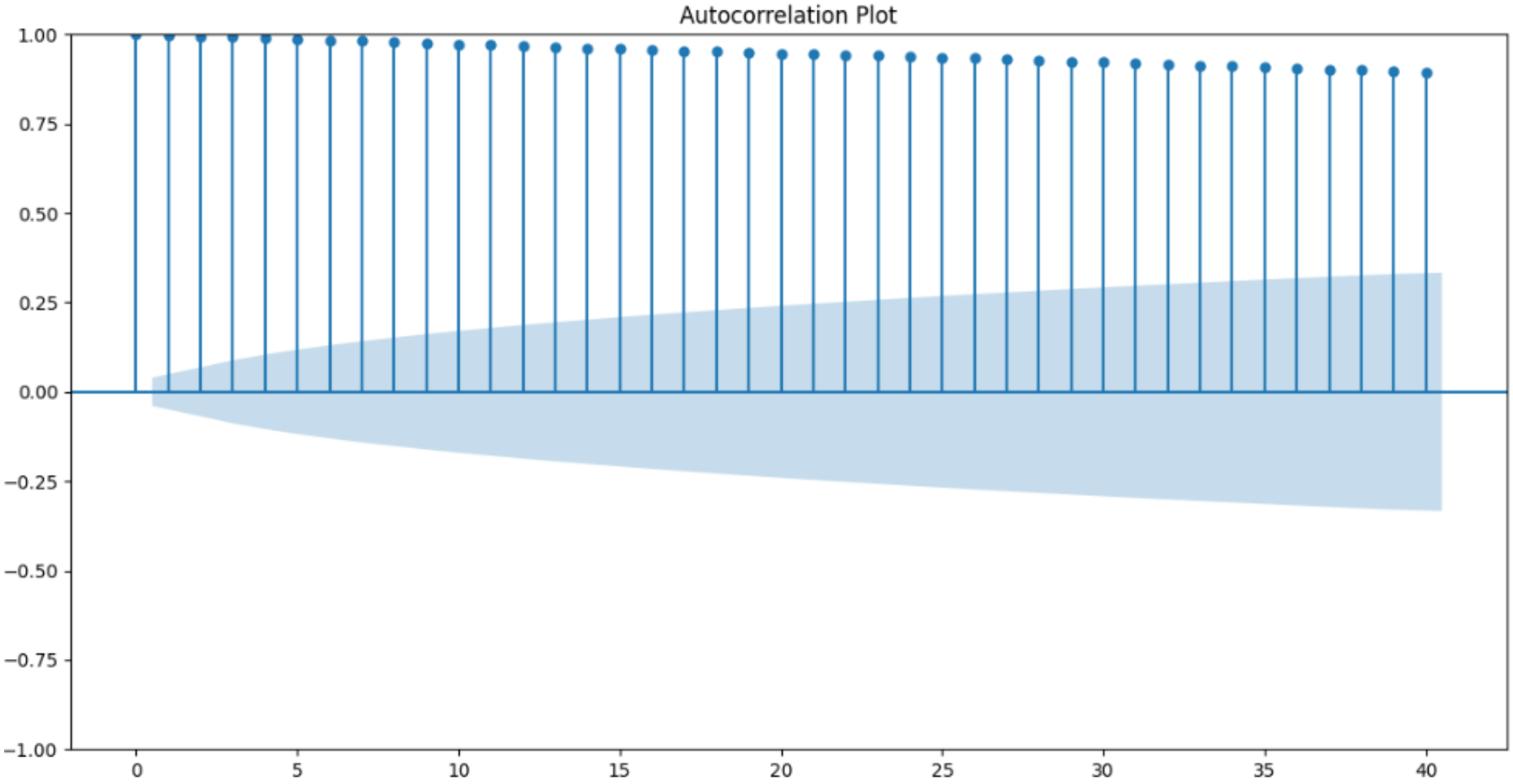
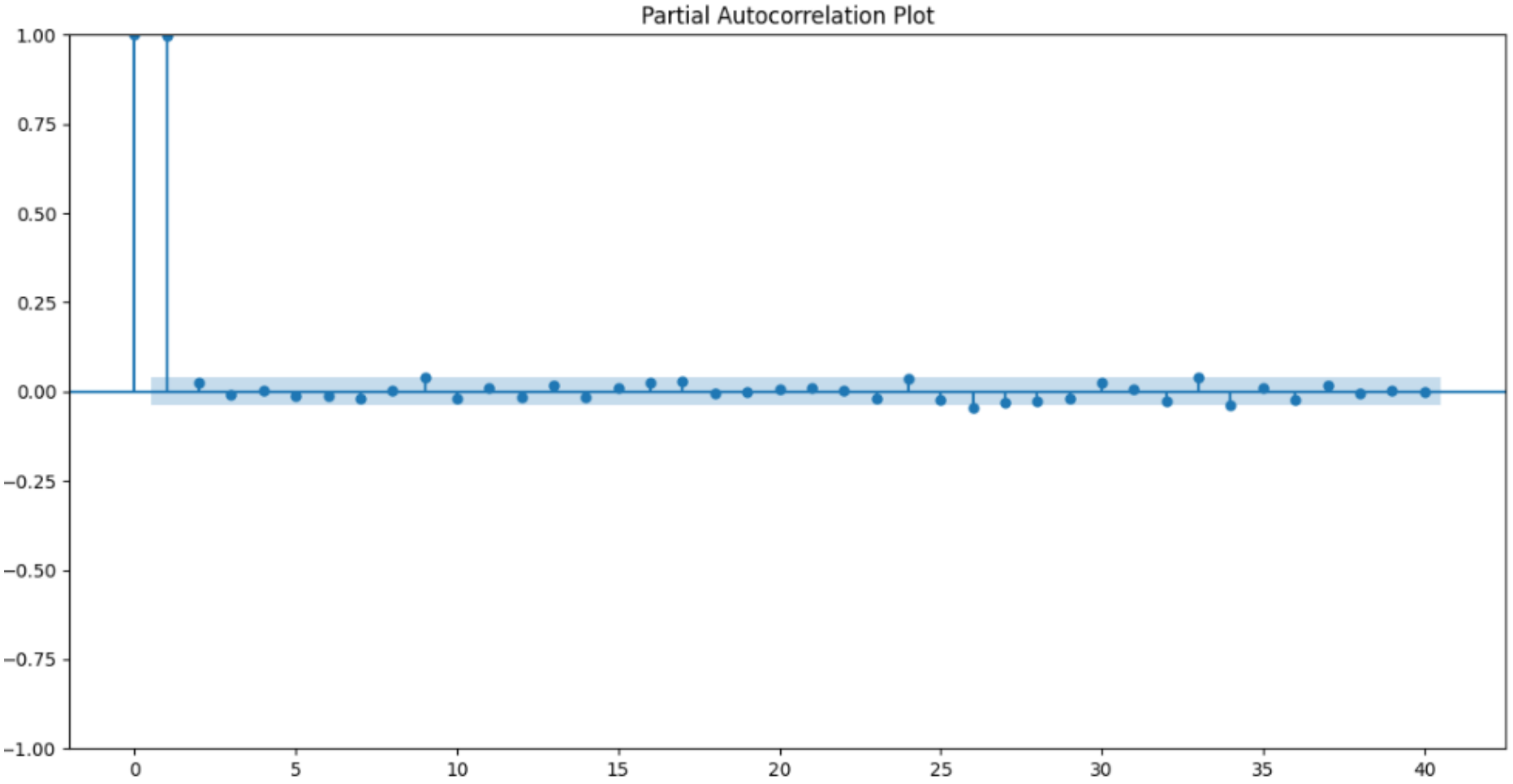


Figure 14. *Partial Autocorrelation plot of BTC_Adj Close.*



With the ACF displaying a total of 40 lags (Figure 13), we observe that the correlation scale ranges from -1 to 1, representing zero correlation to full correlation. Notably, there are more than 40 lags with a positive correlation above 0.5, approaching 1, indicating that this time series is likely to require substantial differencing to achieve stationarity. The consistently high coefficient values and positive correlations across lags suggest that the correlation persists significantly, with points remaining outside the blue region of the plot, denoting strong statistical significance. It is crucial to ensure the absence of missing values in the data, as their presence could invalidate the ACF results.

In the PACF (Figure 14), we observe significant correlations at the first and second lags, with subsequent lags showing no significant correlation. So, we can see as the indirect correlations are not calculated, the correlation "legs" dwindle completely to zero.

Moreover, the ACF and PACF are instrumental in determining the appropriate non-seasonal and seasonal terms for our modeling. For example, the PACF reveals that, since there are two positive partial autocorrelations, the higher-order autocorrelations are explained by the lag-2 autocorrelation. Simply put, by examining the PACF, we can determine the appropriate number of Autoregressive (AR) terms needed to capture the autocorrelation structure of the time series. If the PACF is significant only at these two lags, with other lags dropping abruptly to zero, it suggests incorporating an AR term of 2 into our model. Overall, the ACF and PACF guides our approach to modeling, analyzing and forecasting the time series data.

Additionally, we assessed *cointegration* to model long-term equilibrium relationships among multiple time series. Cointegration indicates that, despite individual non-stationarity, the time series move together over time. Prior to testing for cointegration, we verified the non-stationarity of individual time series through both ADF and KPSS tests and visualizations.

Given that we have more than two time series, in Figure 15, we employed the *Johansen test* to detect multiple cointegration relationships among the series. This test is suitable for analyzing up to 12 time series. We selected time series based on the correlation matrix to fit within this limit.

To interpret the Johansen cointegration test results, we focused on two key statistics in Figure 16: *the Trace Statistic* and *the Max-Eigen Statistic*. These statistics help identify the number of cointegrating relationships (long-term equilibrium relationships) among the tested variables. Our results indicate the presence of likely 2 or 4 cointegrating relationships among the 12 variables. This suggests that, despite short-term deviations, these variables share a long-term equilibrium relationship, providing valuable insights into their joint behavior over time, particularly in financial and economic contexts.

Figure 15. *Johansen Cointegration Test.*

```
# Cointegration Test
from statsmodels.tsa.vector_ar.vecm import coint_johansen

# Johansen test has limits, 12 variables
# Selected 12 variables according to correlation matrix
selected_columns = ["BTC_Adj Close", "OBV", "TR", "Volatility",
                    "MSCI_World_Index", "Gold", "Federal Funds Effective Rate", "US_Dollar_Index", "S&P_500",
                    "Discussion about pandemics, Index-Total count of mentions of epidemics (normalized by total number of words)",
                    "Petroleum Markets EMV Tracker", "Financial Regulation EMV Tracker"]
subset_df = df[selected_columns].dropna()  # Drop rows with NaN values

# Perform the Johansen test (assuming df contains multiple time series)
johansen_test = coint_johansen(subset_df, det_order=0, k_ar_diff=1)

# Extract the test statistics and critical values
trace_stat = johansen_test.lr1
max_eigen_stat = johansen_test.lr2
critical_values = johansen_test.cvt

print(f'Trace Statistics: {trace_stat}')
print(f'Max-Eigen Statistics: {max_eigen_stat}')
print(f'Critical Values (90%, 95%, 99%): \n{critical_values}')

# Check for cointegration based on the Trace statistic and critical values
print("\nCointegration Check based on Trace Statistic:")
for i in range(len(trace_stat)):
    if trace_stat[i] > critical_values[i, 1]:  # 95% level
        print(f"Series are cointegrated at rank {i+1}")
    else:
        print(f"No cointegration at rank {i+1}")
        break

# Check for cointegration based on the Max-Eigen statistic and critical values
print("\nCointegration Check based on Max-Eigen Statistic:")
for i in range(len(max_eigen_stat)):
    if max_eigen_stat[i] > critical_values[i, 1]:  # 95% level
        print(f"Series are cointegrated at rank {i+1}")
    else:
        print(f"No cointegration at rank {i+1}")
        break
```

Figure 16. *Results based on the Trace and Max-Eigen statistics.*

```
Trace Statistics: [1.25621249e+03 4.49861238e+02 2.89440440e+02 2.13875317e+02
 1.58893628e+02 1.21010663e+02 8.91546408e+01 6.18030146e+01
 3.67320460e+01 1.62190398e+01 4.17804982e+00 5.03298454e-01]
Max-Eigen Statistics: [8.06351250e+02 1.60420798e+02 7.55651231e+01 5.49816889e+01
 3.78829648e+01 3.18560226e+01 2.73516263e+01 2.50709685e+01
 2.05130063e+01 1.20409899e+01 3.67475136e+00 5.03298454e-01]
Critical Values (90%, 95%, 99%):
[[326.5354 334.9795 351.215 ]
 [277.374  285.1402 300.2821]
 [232.103  239.2468 253.2526]
 [190.8714 197.3772 210.0366]
 [153.6341 159.529  171.0905]
 [120.3673 125.6185 135.9825]
 [ 91.109   95.7542 104.9637]
 [ 65.8202  69.8189  77.8202]
 [ 44.4929  47.8545  54.6815]
 [ 27.0669  29.7961  35.4628]
 [ 13.4294  15.4943  19.9349]
 [  2.7055   3.8415   6.6349]]

Cointegration Check based on Trace Statistic:
Series are cointegrated at rank 1
Series are cointegrated at rank 2
Series are cointegrated at rank 3
Series are cointegrated at rank 4
No cointegration at rank 5

Cointegration Check based on Max-Eigen Statistic:
Series are cointegrated at rank 1
No cointegration at rank 2
```

3. Model Development

The evolution of multi-factor risk models in cryptocurrency pricing, as well as for traditional assets, demonstrates a growing sophistication in capturing various dimensions of asset characteristics and market behavior. Historically, these models have progressed from simpler frameworks to more advanced multi-factor approaches, reflecting the financial industry's ongoing effort to enhance market prediction and understanding.[14] Modern multi-factor models are designed to incorporate additional elements of asset pricing that earlier models often overlooked, offering a more comprehensive view of market dynamics.[15]

Integrating machine learning with factor modeling represents a significant advancement in predictive accuracy. Machine learning algorithms excel at uncovering complex patterns within data and can adapt dynamically to new information.[16] This adaptability is particularly

[14] Zhou, F., Huang, Z., & Zhang, C. (2022). Carbon price forecasting based on CEEMDAN and LSTM. Applied energy, 311, 118601.
[15] Jin, Z., Yang, Y., & Liu, Y. (2020). Stock closing price prediction based on sentiment analysis and LSTM. Neural Computing and Applications, 32, 9713-9729.
[16] Liu, Y., Yang, C., Huang, K., & Gui, W. (2020). Non-ferrous metals price forecasting based on variational mode decomposition and LSTM network. Knowledge-Based Systems, 188, 105006.

valuable in the volatile and ever-changing landscape of financial markets, ensuring that models remain effective as market conditions shift.[17]

Our approach seeks to harness the strengths of both traditional models and machine learning techniques. By combining the foundational insights of conventional factor analysis with the sophisticated capabilities of machine learning, we aim to create models that are better equipped to handle the complexities of today's financial markets.

3.1 Baseline Model

In the initial phase of our modeling efforts, we established a baseline model. This fundamental model serves as a reference point against which more complex models can be evaluated. Its primary purpose is to provide a benchmark, allowing us to assess whether more sophisticated models offer substantial improvements over this simple approach. Evaluating against the baseline model helps determine if the additional time and computational resources invested in advanced models yield significant benefits.

For our baseline model, we have selected the Bitcoin Adjusted Close price (BTC_Adj Close) as the target variable for predicting cryptocurrency price variance. This choice is grounded in three key reasons. First, Bitcoin, being the largest and most established cryptocurrency, serves as a benchmark for the entire crypto market. Its price movements significantly influence other cryptocurrencies, making it a representative proxy for broader market trends. Second, the Adjusted Close price reflects Bitcoin's true value at the end of each trading day. This value accounts for all trading activity, offering a reliable measure of Bitcoin's market value and ensuring accuracy in our predictions. Third, Bitcoin boasts the most extensive historical data among cryptocurrencies, providing a robust dataset for modeling and prediction. This rich dataset is essential for effectively training machine learning models and developing predictive insights.

3.1.1. Model Selection 1: Linear Regression

We began our model trials with Linear Regression due to its simplicity and ease of interpretation. Linear Regression assumes a linear relationship between the independent variables (features) and the dependent variable (target), making it an ideal starting point for many regression tasks. However, rather than stopping at a basic implementation, we delved deeply into both the data and the model to extract insights and pave the way for more advanced modeling techniques.[18]

First, we determined our target variable and relevant features. We then split the data into training and testing sets, using 20% of the data for testing. This allocation allows us to retain a larger portion of the data for training, which is particularly beneficial when dealing with complex datasets where more data is needed for the model to learn effectively. We set the *random state* to 42 to ensure reproducibility; using a consistent random split ensures that the data split remains the same across different runs of the model, facilitating consistent comparison

[17] Wu, J. M. T., Li, Z., Herencsar, N., Vo, B., & Lin, J. C. W. (2023). A graph-based CNN-LSTM stock price prediction algorithm with leading indicators. Multimedia Systems, 29(3), 1751-1770.

[18] https://github.com/yunusgumussoy/Crypto-Factor-Modeling-Data-Challenge/blob/main/2-Model-Selection1-LinearRegression.ipynb

of results. The choice of 42 is conventional in the data science community, though any integer could serve this purpose.

Next, we calculated the Variance Inflation Factor (VIF) to detect multicollinearity among the features (Figure 17). High VIF scores indicate that a variable is highly correlated with other variables, which can lead to unstable coefficient estimates and inflated standard errors in regression models. Although we considered removing variables with high VIF scores, we decided to retain them for this baseline model to better understand how they might influence more advanced models in subsequent steps.

Figure 17. *Variance Inflation Factor (VIF) to detect multicollinearity on unscaled data.*

```
# Compute Variance Inflation Factor (VIF) for Multicollinearity on unscaled data
vif_data = pd.DataFrame()
vif_data["feature"] = X.columns
vif_data["VIF"] = [variance_inflation_factor(X.values, i) for i in range(X.shape[1])]

# Sort the DataFrame by VIF scores in descending order
vif_data = vif_data.sort_values(by="VIF", ascending=False)

print(vif_data)
```

```
                                  feature           VIF
83   Total Transaction Output Volume Per Day  1.473691e+11
80    Total Transaction Input Volume Per Day  1.473546e+11
82      Daily Median Output Volume Per Block  1.040056e+09
79       Daily Median Input Volume Per Block  1.018263e+09
51                             Block Height  2.600972e+03
..                                      ...           ...
183                     Taxes EMV Tracker  1.081490e+00
199                Trade Policy EMV Tracker  1.080541e+00
112                             Daily_Change  1.055386e+00
136                              Log_Returns  1.035018e+00
138                             Daily_Return  1.032476e+00

[206 rows x 2 columns]
```

To address potential multicollinearity issues, we incorporated regularization techniques, including Ridge, Lasso and Elastic Net to the Linear Regression Model. These techniques help mitigate multicollinearity by penalizing large coefficients, thus stabilizing the model.

We then scaled the features (i.e., the independent variables, X_train and X_test) while leaving the target variable (i.e., the dependent variable, y_train and y_test) unscaled. Scaling the target variable is typically avoided to maintain the interpretability of the model's predictions and ensure that the predicted values are directly comparable to the actual values in their original units. After fitting the model, we conducted Recursive Feature Elimination (RFE) to assess the contribution of each feature to the model (Figure 18).

Figure 18. *Recursive Feature Elimination (RFE).*

```
Selected Features: Index(['BTC_Close_lag1', 'BTC_Close_lag2', 'BTC_Close_lag4', 'BTC_MA5',
       'Daily_Change'],
      dtype='object')
                                     Feature  Ranking
93                                    BTC_MA5        1
85                              BTC_Close_lag1        1
88                              BTC_Close_lag4        1
112                               Daily_Change        1
86                              BTC_Close_lag2        1
..                                        ...      ...
18       Lightning Network Clustering-Average      198
103                                       RSI      199
40                              Block Weight      200
141              Federal Funds Effective Rate      201
73   Daily Median Transaction Inputs Per Block      202

[206 rows x 2 columns]
```

While VIF is usually computed on the original (unscaled) data to assess multicollinearity, we also calculated VIF after scaling (X_train_scaled) to observe how normalization affected both the data and the model. An infinite VIF value indicates perfect multicollinearity, meaning that the feature is perfectly correlated with one or more other features, leading to an infinitely large variance inflation (Figure 19).

Figure 19. *Variance Inflation Factor (VIF) to detect multicollinearity on scaled data.*

```
                                          feature       VIF
52                                  Miner Revenue       inf
104                                    Volatility       inf
108                                Bollinger_Upper       inf
59   BitcoinVisuals.com's node's memory usage for u...       inf
97                                       BTC_MA20       inf
..                                            ...       ...
24                                 BTC Dominance  5.739001
62                         Total Bitcoin Node Peers  5.540730
126                                           ADX  4.988204
75            Daily Median Output Per Transaction   4.775431
33                           Total Node Bandwidth  2.610044

[206 rows x 2 columns]
```

We utilized multiple evaluation metrics, including R-squared, Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and Mean Absolute Error (MAE) to provide a well-rounded assessment of the model's performance. Finally, we compared the performance of our baseline Linear Regression model against its regularized counterparts: Ridge, Lasso, and ElasticNet (Table 1).

Table 1. *Comparison of Linear Regression, Ridge, Lasso, and ElasticNet models with default parameters.*

	Linear Regression	**Ridge Regression**	**Lasso Regression**	**ElasticNet**
R-Squared Score	1.0	0.9998196144084206	**0.9998960412919523**	0.9990215941078564
Mean Squared Error (MSE)	3.8027649156586244e-21	61754.50669624028	**35589.97520841178**	334954.5420395507
Root Mean Squared Error (RMSE)	0.00	248.50454059481544	**188.6530551261012**	578.7525741105 18
Mean Absolute Error (MAE)	5.133365920617752e-11	177.9308257691205	**136.46975872230777**	415.9925639162611

We conducted hyperparameter tuning using grid search to optimize the model's performance. The results of this tuning process, along with a comparison of the models, are summarized in Table 2.

Table 2. *Comparison of Linear Regression, Ridge, Lasso, and ElasticNet models after Hyperparameter Tuning.*

	Linear Regression	**Ridge Regression**	**Lasso Regression**	**ElasticNet**
R-Squared Score	1.0	0.9999692656090376	**0.9999907960719198**	**0.9999907960719198**
Mean Squared Error (MSE)	3.8027649156586244e-21	10521.833456177264	**3150.9392368105096**	**3150.9392368105096**
Root Mean Squared Error (RMSE)	0.00	102.57598869217524	**56.133227564522286**	**56.133227564522286**
Mean Absolute Error (MAE)	5.133365920617752e-11	71.820399394 43183	**37.232221546879536**	**37.232221546879536**

The ElasticNet's l1_ratio parameter controls the balance between L1 and L2 regularization. After performing hyperparameter tuning with grid search, we found the best parameters to be: {'alpha': 1, 'l1_ratio': 1}.

With an l1_ratio of 1, ElasticNet essentially reduces to Lasso regression, relying solely on L1 regularization.

By examining the model coefficients, we gained insights into the influence of each feature on the target variable. Larger absolute coefficients signify a stronger impact, while the sign (positive or negative) indicates the direction of the relationship (Figure 20).

Figure 20. *Feature Coefficients.*

	Coefficient
BTC_Close_lag1	6.061873e+04
BTC_MA5	2.250722e+04
Daily_Change	1.265999e+04
Daily Median Input Volume Per Block	4.189991e-08
Daily Median Output Volume Per Transaction	1.872280e-08
...	...
Agricultural Policy EMV Tracker	-2.335051e-08
Daily Median Output Volume Per Block	-4.194499e-08
BTC_Close_lag4	-4.611521e+03
BTC_Close_lag2	-4.611521e+03
BTC_Close_lag3	-4.611521e+03

[206 rows x 1 columns]

To ensure the robustness of our Linear Regression model, we included both standard and time-series-specific cross-validation techniques. To further validate our model, we conducted k-fold cross-validation (with k=5), allowing us to assess its performance across different subsets of the data. Finally, we implemented cross-validation specifically tailored for time-series data to account for the sequential nature of the data. This approach provided a more accurate evaluation of the model's predictive capabilities in a temporal context.

Table 3. *Cross Validations Scores.*

Cross-validation scores	0.99997754	1	1	1	1
Average cross-validation score	0.999995507687179				
TimeSeriesSplit cross-validation scores	5.05411275e-20	4.03329978e-21	1.52440533e-21	7.91192944e-21	1.58918799e-20
Average TimeSeriesSplit cross-validation score	1.598052840 0538126e-20				

High cross-validation scores, particularly R-squared values close to 1 or very low MSE values, indicate that the model fits the data well. These metrics suggest that the model effectively captures the underlying patterns in the data, resulting in accurate predictions and minimal error.

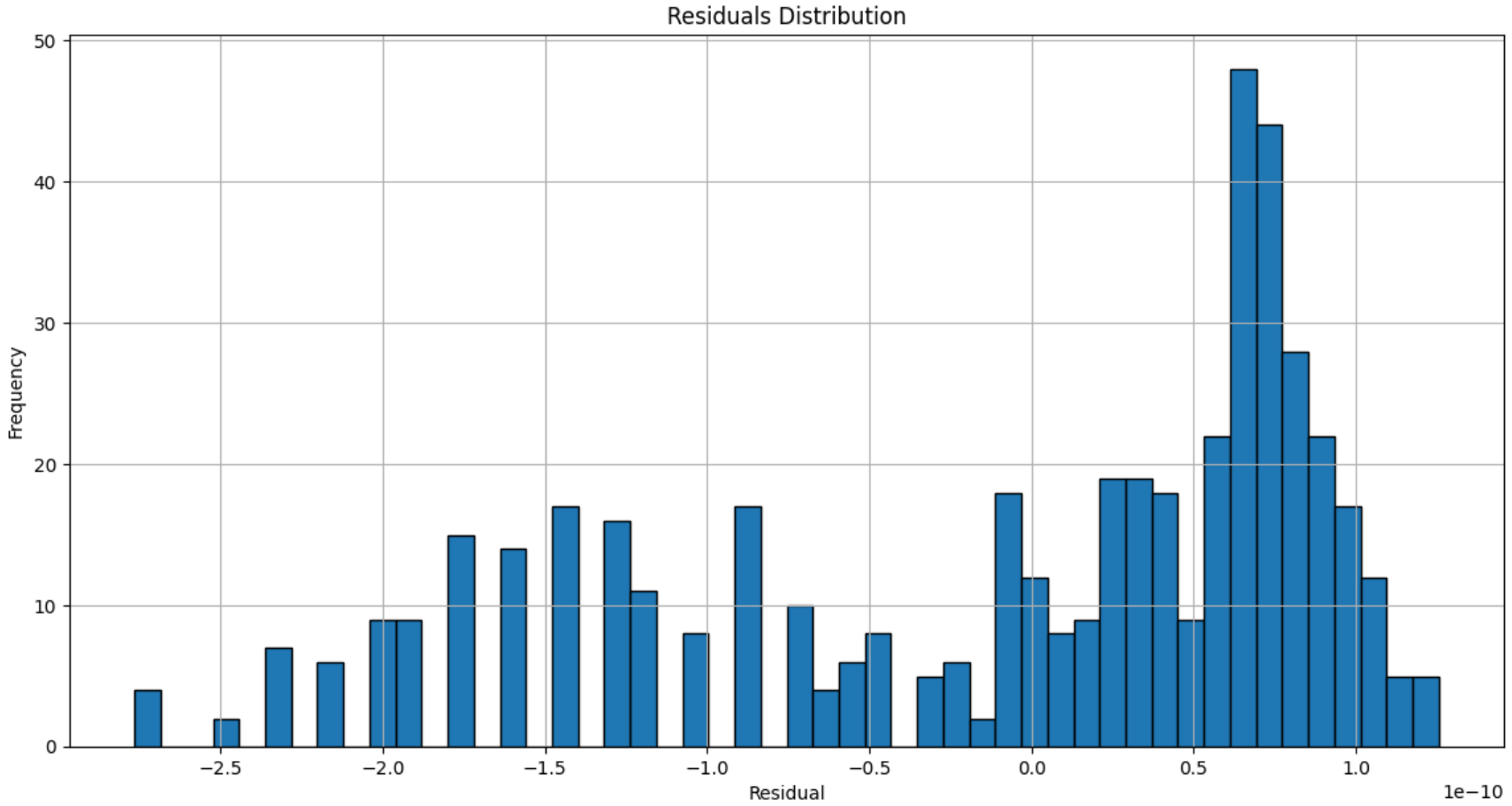
The TimeSeriesSplit cross-validation scores indicate suboptimal performance when considering the temporal structure of the data. This discrepancy indicates that although the model may perform well on randomly partitioned data, it struggles to effectively manage temporal dependencies or trends inherent in time series data.

Figure 21 shows the distribution of residuals, which represent the differences between actual and predicted values. Ideally, these residuals should be normally distributed and centered around zero, indicating that the model's predictions are unbiased. However, the observed skewness in the distribution may signal potential model inadequacies, suggesting that the model may not fully capture the underlying patterns in the data.

To further assess model performance, we also employed scatter plots, Q-Q plots, and box plots to compare the actual values with the predicted ones. These visualizations provided additional insights into the model's accuracy and the distribution of errors.

Following this analysis, we proceeded with more advanced models, including Long Short-Term Memory (LSTM) networks and XGBoost. Additionally, we leveraged AutoML tools like T-POT, MLJAR, and H2O.ai for automated model selection and hyperparameter tuning. These tools helped streamline the process of identifying the best-performing models and optimizing their parameters, enhancing the overall robustness of our approach.

Figure 21. *Residuals Distribution.*



3.1.2. Model Selection 2: Long Short-Term Memory

LSTM models, a specialized type of recurrent neural network (RNN), are well-suited for handling sequences and time series data, making them ideal for our application. Unlike the Linear Regression model, which operates on static data, LSTMs are designed to capture temporal dependencies and patterns within sequential data, and they can process multiple variables (features) simultaneously. To address potential biases and scale differences across features, we implemented cross-sectional normalization using MinMaxScaler from this point forward. This technique adjusts each feature individually, ensuring that all variables are scaled to a common range. By doing so, it prevents features with larger magnitudes from dominating the model, allowing each feature to contribute equally. This normalization is particularly crucial in datasets with diverse feature types, as it standardizes the value ranges and enhances the model's ability to detect subtle patterns and relationships across different assets and indicators.

After extensive experimentation, we identified the optimal model parameters that yielded the best results:[19]

```
model = Sequential()

model.add(LSTM(units=256, activation='relu', return_sequences=True,
input_shape=(X_train.shape[1], X_train.shape[2])))

model.add(Dropout(0.2))

model.add(LSTM(units=128, activation='tanh', return_sequences=True))

model.add(Dropout(0.2))

model.add(LSTM(units=64, activation='linear'))

model.add(Dropout(0.2))

model.add(Dense(1, activation='sigmoid'))

opt = Adam(learning_rate=0.0001)

model.compile(optimizer=opt, loss='mean_squared_error')

model.fit(X_train, y_train, epochs=250, batch_size=16, validation_data=(X_test, y_test),
verbose=1)
```

Table 4. *LSTM Model Evaluation.*

Mean Squared Error (MSE)	171969.7957921721
R-Squared Score	0.9995063730197185
Mean Absolute Error (MAE)	304.83469990247585

We normalized the data across all features with varying scales. This scaling process typically adjusts the data range to either 0 and 1 or centers it around a mean of 0 with a standard deviation of 1. Thus, even after applying inverse scaling to the predictions, the resulting plots

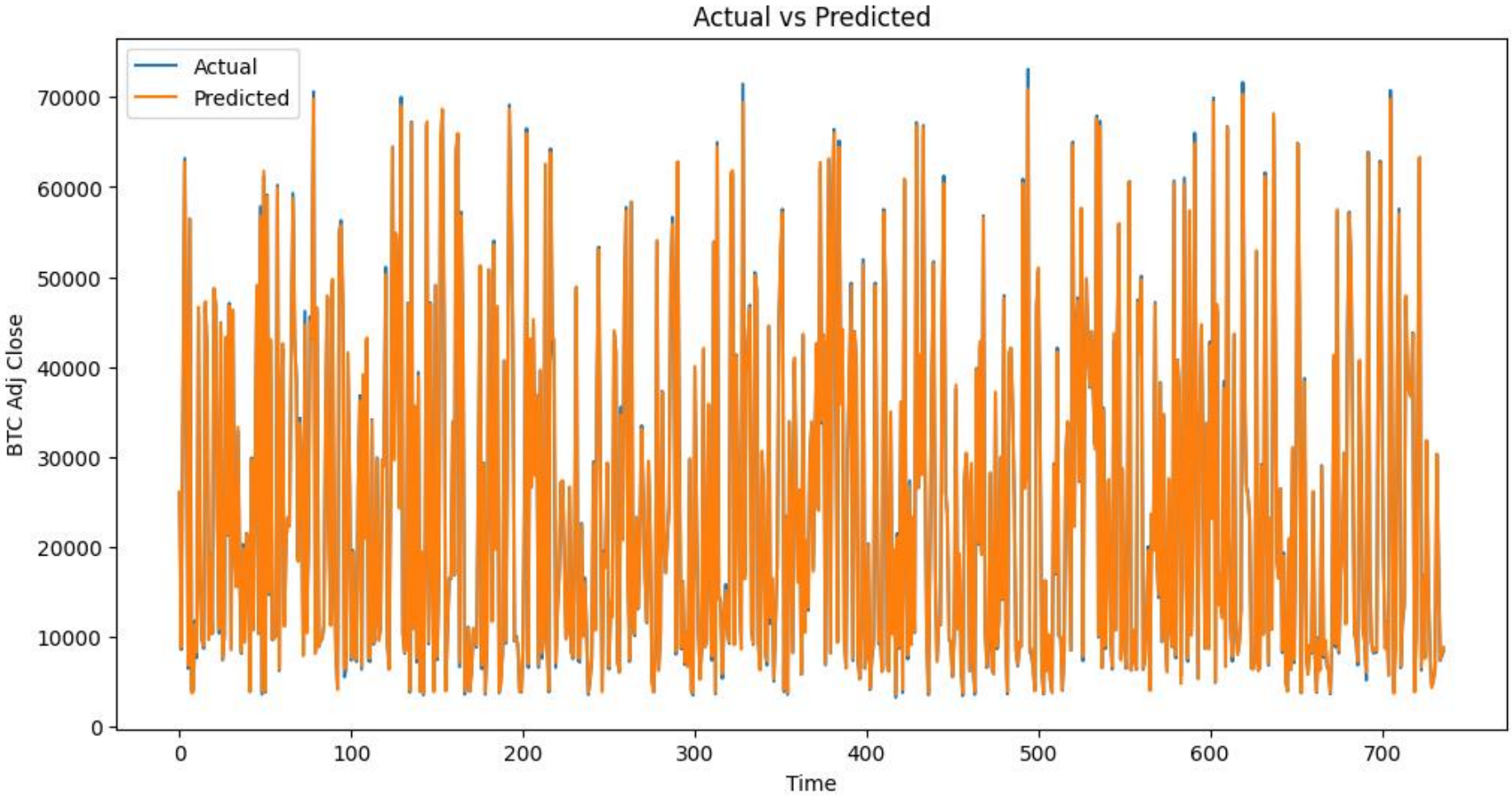
[19] https://github.com/yunusgumussoy/Crypto-Factor-Modeling-Data-Challenge/blob/main/2-Model-Selection2-LSTM.ipynb

do not display the true values. Instead, they reflect the scaled version of the data, which is essential for maintaining consistency during model training and evaluation.

To explore the predictive power of an LSTM model[20] focused on a single feature, specifically the Bitcoin Adjusted Close price and its lagged variations (17-days lag based on our trials), we developed a specialized model. This approach allows us to assess how well the LSTM can predict and forecast actual prices. The model was trained on Bitcoin data spanning from January 1, 2014, to July 27, 2024 (start="2014-01-01", end="2024-07-27"), and was tasked with forecasting the subsequent 30-day period. The decision to select this period was driven by the intention to evaluate the LSTM model's performance in a real-time context, offering insights into its practical applicability in live market conditions. These figures are from 22 to 25.

[20] https://github.com/yunusgumussoy/Crypto-Factor-Modeling-Data-Challenge/blob/main/LSTM-BTC.py

Figure 22. *LSTM Actual vs. Predicted.*



.

Figure 23. *LSTM Prediction.*

Figure 24. *LSTM Forecasting.*

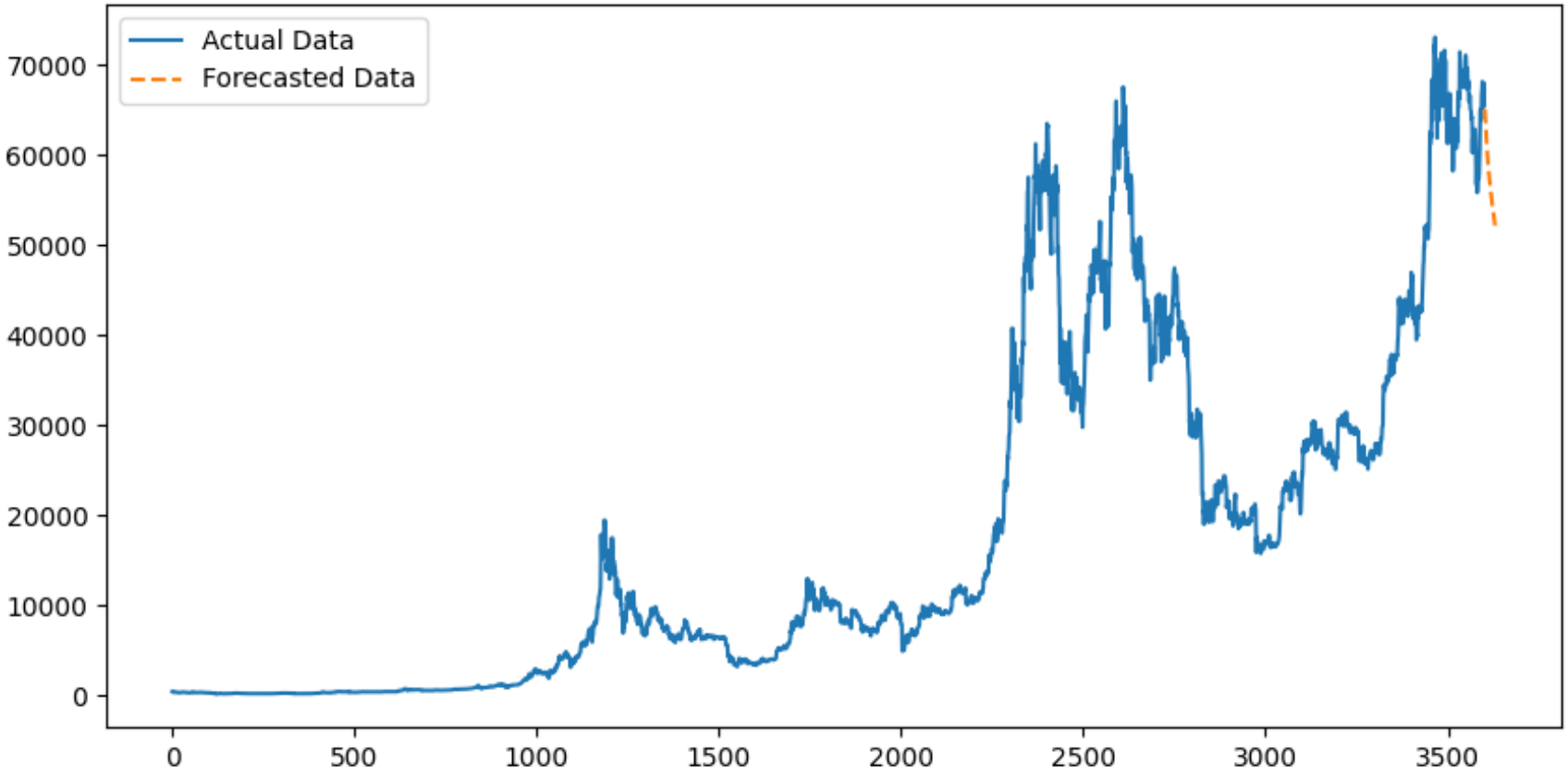
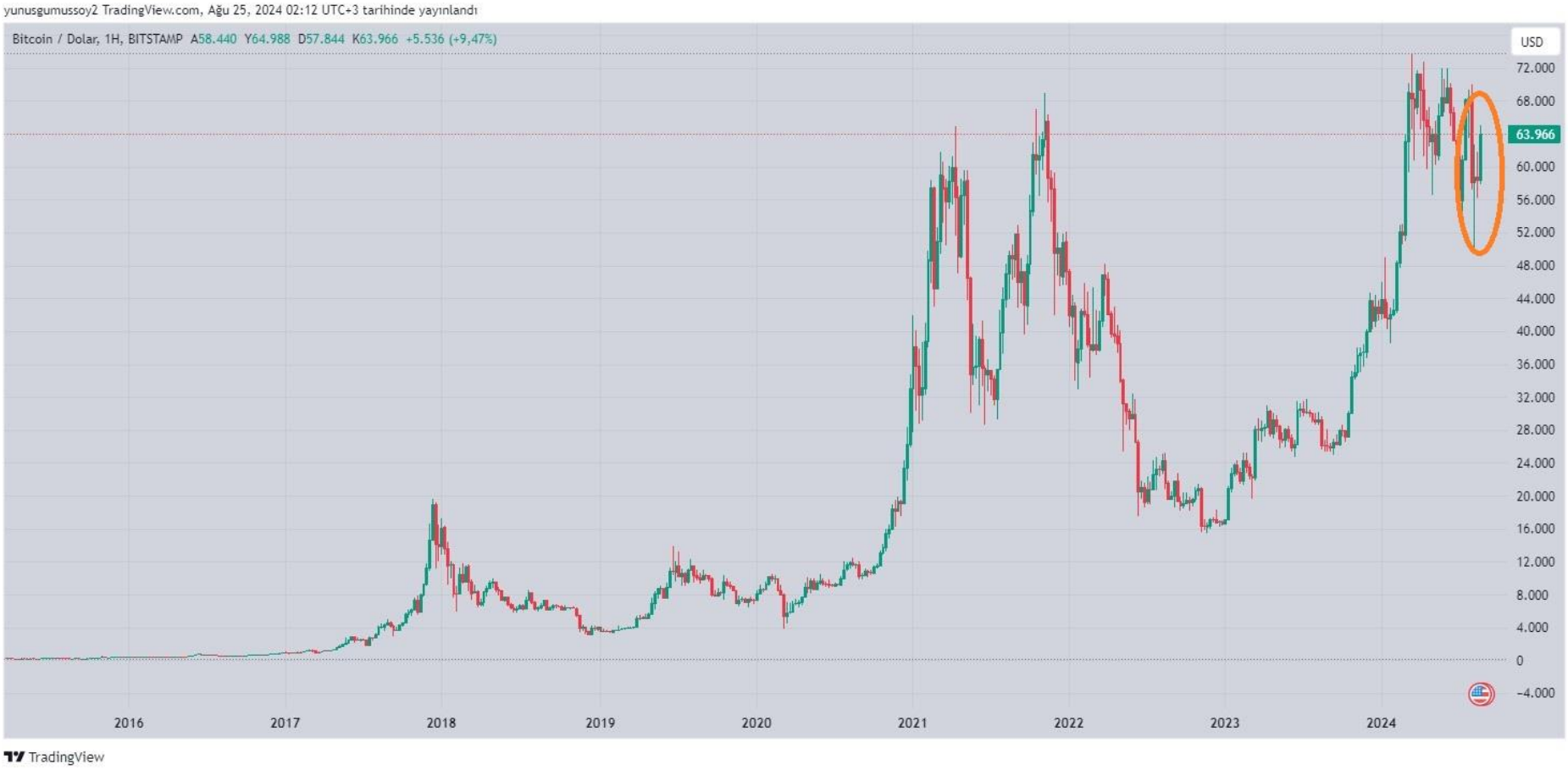


Figure 25. *Real-time Bitcoin Price.*



It is important to highlight that Bitcoin price reached approximately $69,000 on July 27. However, following that peak, the price dropped to around $49,000 by August 5, and as of August 24, it has not returned to the $69,000 level. This pattern underscores the potential of the LSTM model's forecasting capabilities, which could be leveraged as part of a trading strategy. By accurately predicting such significant price movements, the LSTM model could provide valuable insights for making informed trading decisions in the highly volatile cryptocurrency market.

3.1.3. Model Selection 3: Extreme Gradient Boosting (XGBoost)

We also chose to use XGBoost for its robust performance with datasets containing multiple features. XGBoost is designed to handle large datasets efficiently, both in terms of computation and memory usage.

For our XGBoost model, we configured it with the following parameters:[21]

```
model = xgb.XGBRegressor(
    objective='reg:squarederror',
    n_estimators=1000,
    learning_rate=0.01,
    max_depth=6,
    subsample=0.8,
    colsample_bytree=0.8,
    random_state=42
)
```

Table 5. *XGBoost Model Evaluation.*

Mean Squared Error	30732.217417469816
R-Squared Score	0.9995588120326391
Mean Absolute Error	215.50110575496092

Similar to the approach with LSTM, we also developed an XGBoost model[22] that uses a single feature: Bitcoin Close price and its lagged variations (8-days-lagged, based on our trials). We trained this model on Bitcoin data from January 1, 2014, to July 27, 2024 (start=”2014-01-01”, end=”2024-07-27”) and forecasted the subsequent 30-days period (Figure 26). This real-time experiment allows us to evaluate the model’s forecasting performance under current market conditions.

[21] https://github.com/yunusgumussoy/Crypto-Factor-Modeling-Data-Challenge/blob/main/2-Model-Selection3-XGBoost.py

[22] https://github.com/yunusgumussoy/Crypto-Factor-Modeling-Data-Challenge/blob/main/XGBoost-BTC.py

Figure 26. *XGBoost Bitcoin Price Forecasting.*

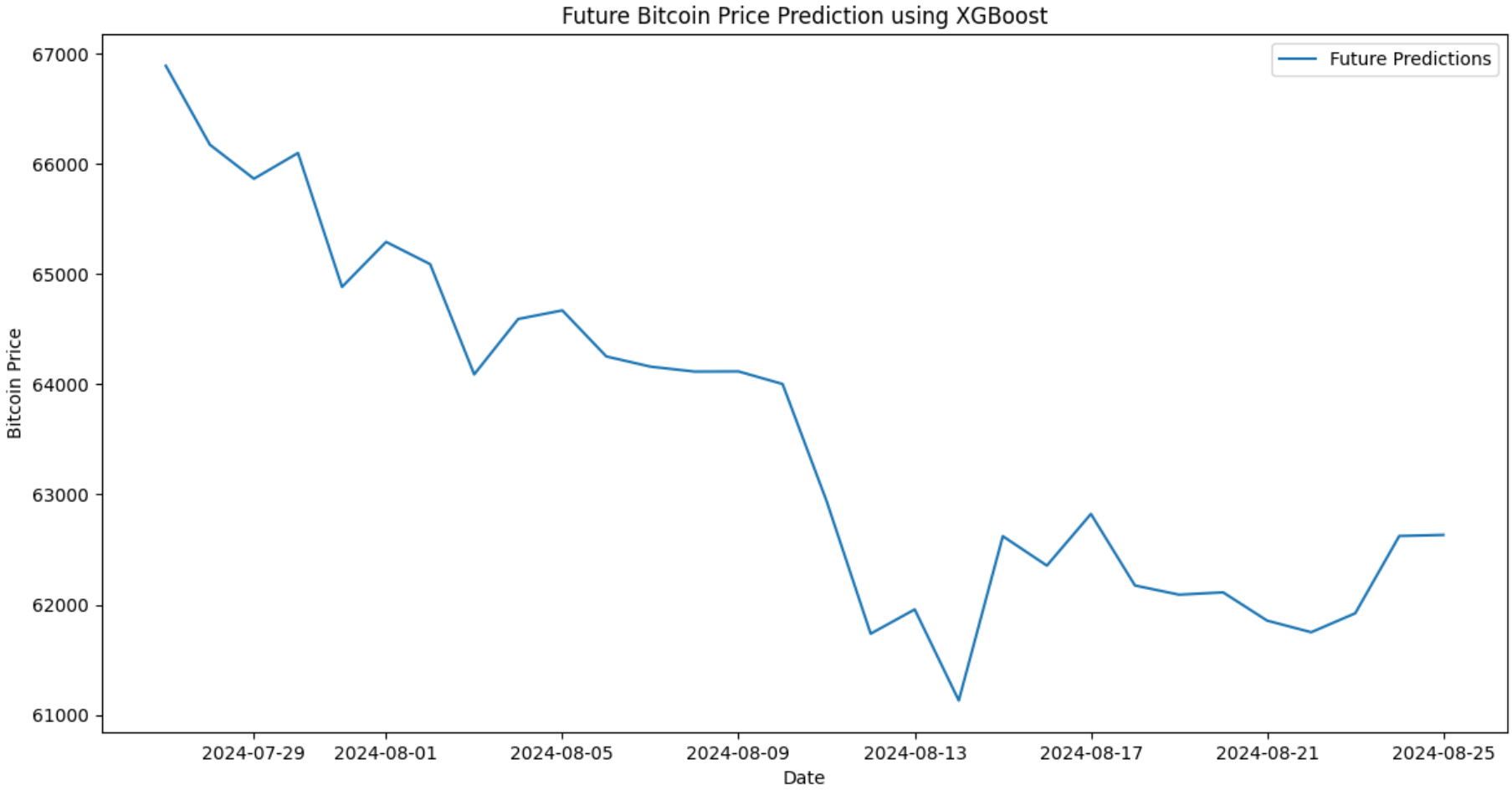


Figure 27. *Real-time Bitcoin Price.*

The forecasting results from the XGBoost model closely align with the actual Bitcoin price movements observed during the forecast period (Figures 26 and 27). This similarity suggests that XGBoost, like LSTM, holds potential for application in trading strategies. Both LSTM and XGBoost models have demonstrated strong performance, providing valuable insights for predicting and forecasting Bitcoin prices.

3.2 Automated Machine Learning

To enhance our model search and optimize hyperparameters efficiently, we turned to Automated Machine Learning (AutoML) tools. These tools have introduced us to robust models and configurations, significantly accelerating the process compared to manual methods. This approach allowed us to focus more on strategic decisions and interpretations. We utilized Python libraries such as mljar-supervised,[23] H2O.ai,[24] and T-Pot[25] for this purpose.

3.2.1. Model Selection 4: RidgeCV with DecisionTreeRegressor & RidgeCV with ElasticNet

We ran TPOT on our dataset at two different times, each run yielding distinct optimal pipelines:

First Run: RidgeCV(DecisionTreeRegressor(StandardScaler(input_matrix), max_depth=8, min_samples_leaf=10, min_samples_split=17))[26]

Second Run: RidgeCV(ElasticNetCV(StandardScaler(input_matrix), l1_ratio=1.0, tol=0.001))[27]

Although we initially used MinMaxScaler for normalization, TPOT recommended models using StandardScaler. We tested all models with both normalization techniques, observing that the performance varied depending on the model.

Among the tested four models, RidgeCV with ElasticNet (using StandardScaler) demonstrated the best performance, outperforming the other configurations.

[23] https://github.com/yunusgumussoy/Crypto-Factor-Modeling-Data-Challenge/blob/main/9-AutoML%20Scripts/3-mljar.py
[24] https://github.com/yunusgumussoy/Crypto-Factor-Modeling-Data-Challenge/blob/main/9-AutoML%20Scripts/5-h2oai.py
[25] https://github.com/yunusgumussoy/Crypto-Factor-Modeling-Data-Challenge/blob/main/9-AutoML%20Scripts/0-t-pot.py
[26] https://github.com/yunusgumussoy/Crypto-Factor-Modeling-Data-Challenge/blob/main/9-AutoML%20Scripts/1-t-pot-suggested-model1.py
[27] https://github.com/yunusgumussoy/Crypto-Factor-Modeling-Data-Challenge/blob/main/9-AutoML%20Scripts/2-t-pot-suggested-model2.py

Table 6. *Model Selection 4.*

Model Evaluation	RidgeCV with DecisionTreeRegressor (StandardScaler)	RidgeCV with DecisionTreeRegressor (MinMaxScaler)	RidgeCV with ElasticNet (StandardScaler)	RidgeCV with ElasticNet (MinMaxScaler)
Mean Squared Error	538973.4759252893	541395.6501358565	**9343.249358697387**	63631.05225673375
R-Squared Score	0.9984529152451027	0.9984459625675365	**0.9999731808719915**	0.9998173516225164
Mean Absolute Error	421.0244913258848	424.8939286442145	**67.12845791286209**	180.06453856572 37

3.2.2. Model Selection 5: Ensemble Stacked Model (tree – gbm – svr & RidgeCV)

In our run with MLJAR, we obtained a stacking ensemble model that integrates various base learners with a meta-learner. The specific configuration of this model includes:[28]

('tree', DecisionTreeRegressor(max_depth=8, min_samples_leaf=10, min_samples_split=17)),

('gbm', GradientBoostingRegressor(n_estimators=100, learning_rate=0.1, max_depth=3)),

('svr', SVR(kernel='linear', C=1.0)

RidgeCV(alphas=[0.1, 1.0, 10.0])

This stacking ensemble uses StandardScaler for normalization. We also manually tested this model using MinMaxScaler to compare the impact of different scaling techniques on the model's performance. The results of these evaluations are summarized in Table 7.

Table 7. *Model Selection 5.*

Model Evaluation	Ensemble Stacked Model (StandardScaler)	Ensemble Stacked Model (MinMaxScaler)
Mean Squared Error	**186321.59670654748**	194129.81147394236
R-Squared Score	**0.9994651772032419**	0.9994427642820701
Mean Absolute Error	**263.9091894341168**	270.89454175280696

[28] https://github.com/yunusgumussoy/Crypto-Factor-Modeling-Data-Challenge/blob/main/9-AutoML%20Scripts/4-mjlar-suggested-model.py

3.2.3. Model Selection 6: GBM & LightGBM

In our analysis with H2O.ai, we identified the Gradient Boosting Machine (GBM) as a suitable algorithm for our model. We configured our GBM model with the following parameters:[29]

n_estimators=100 — Number of boosting stages to be run

learning_rate=0.1 — Step size shrinkage used in update to prevent overfitting

max_depth=3 — Maximum depth of the individual estimators

min_samples_split=2 — Minimum number of samples required to split an internal node

min_samples_leaf=1 — Minimum number of samples required to be at a leaf node

subsample=1.0 — Fraction of samples used for fitting the individual base learners

random_state=42 — Ensures reproducibility

Also, we tested LightGBM,[30] an advanced gradient boosting framework known for its efficiency in handling large datasets and high-dimensional data. LightGBM demonstrated comparable performance to GBM, but its faster processing speed and ability to manage extensive datasets make it a preferable option in scenarios with large-scale data.

Table 8. *Model Selection 6.*

Model Evaluation	GBM	LightGBM
Mean Squared Error	195539.68804658568	**140468.68624961565**
R-Squared Score	0.9994387173323606	**0.999596794698173**
Mean Absolute Error	264.74949178005704	**217.91506471137438**

In our comparison of GBM and LightGBM, we found the following results:

- MSE: LightGBM had an MSE of 140468.68624961565, which is lower than GBM's MSE of 195539.68804658568. This indicates that LightGBM's predictions have smaller squared errors on average, making LightGBM slightly better at minimizing large errors.
- R-squared (R^2) Scores: Both GBM and LightGBM showed similar R^2 scores, suggesting that they are equally effective at capturing the variance in the target variable.
- Mean Absolute Error (MAE): LightGBM achieved a lower MAE compared to GBM, indicating that LightGBM's predictions are more accurate on average, with smaller absolute errors.

Overall, while both models perform well, LightGBM exhibits a slight advantage in average prediction accuracy (as reflected by the MAE).

[29] https://github.com/yunusgumussoy/Crypto-Factor-Modeling-Data-Challenge/blob/main/9-AutoML%20Scripts/6-h2oai-suggested-model-GBM.py
[30] https://github.com/yunusgumussoy/Crypto-Factor-Modeling-Data-Challenge/blob/main/9-AutoML%20Scripts/7-h2oai-suggested-model-LightGBM.py

3.3 Stress Testing

Stress testing is a vital step in validating financial models, particularly in volatile markets like cryptocurrencies. This process involves simulating extreme conditions to assess how well the model performs under such scenarios, providing insights into its robustness and reliability. For this purpose, we chose the LightGBM model due to its user-friendly nature and its ability to handle data without requiring normalization or standardization in many cases.[31]

3.3.1 Stress Scenario

Our stress scenario simulates a market crash, specifically a sudden 30% drop in cryptocurrency prices within a single day. This scenario is representative of severe economic disruptions, such as a global or regional economic downturn, which could occur during events like a recession. Given the current economic forecasts, including the potential for a recession in the US, this scenario is both relevant and timely for evaluating the model's performance under extreme market conditions.[32]

Table 9. *Stress Scenario Model Evaluation.*

Model Evaluation	LightGBM (Baseline)	LightGBM (Under Crash)
Mean Squared Error	**140468.68624961565**	66930152.82748969
R-Squared Score	**0.999596794698173**	0.8078817906491814
Mean Absolute Error	**217.91506471137438**	6298.308943006476

Under the crash scenario, the MSE increases dramatically to 66930152.82748969. This substantial rise indicates that the model's predictions become significantly less accurate during extreme market conditions, highlighting its struggle to handle such crisis scenarios effectively. The R-Squared score also drops to 0.8079 during the crash scenario. While this score still reflects a reasonable level of explanatory power, the significant decline demonstrates that the model's ability to account for the variance in the target variable is notably diminished under extreme conditions. This decrease suggests that the model may not generalize well to extreme or unforeseen events, revealing potential limitations in its robustness and reliability during high-stress situations.

3.4 Simulations

To evaluate the potential outcomes of our model across various scenarios, we employed Monte Carlo simulation.[33] This method allows us to repeatedly sample from the distributions of our input variables to assess a wide range of possible outcomes and quantify the uncertainty associated with our predictions.

Our goal was to understand how our model performs under different conditions and to simulate extreme scenarios to gauge its robustness. Specifically, we chose to vary interest rates, represented by the Federal Funds Effective Rate in our data, to observe how changes in this key economic indicator impact model performance and predictions. The rationale behind the choice

[31] https://github.com/yunusgumussoy/Crypto-Factor-Modeling-Data-Challenge/blob/main/9-StressTest-LGBM.py
[32] JPMorgan (2024). *The probability of a recession now stands at 35%.* https://www.jpmorgan.com/insights/global-research/economy/recession-probability
[33] https://github.com/yunusgumussoy/Crypto-Factor-Modeling-Data-Challenge/blob/main/9-Simulation-MonteCarlo.py

of interest rate change as a simulation scenario is based on recent insights provided by Federal Reserve Chairman Jerome Powell, who indicated that "the time has come" for the U.S. Federal Reserve to consider cutting interest rates.[34] This statement reflects a broader economic context where interest rate policies significantly impact financial markets, including the cryptocurrency sector.

Table 10. *Simulation.*

Model Evaluation	LightGBM (Baseline)	LightGBM (Simulation)
Mean Squared Error	**140468.68624961565**	
R-Squared Score	**0.999596794698173**	
Mean Absolute Error	**217.91506471137438**	
Simulated Mean MSE		140471.49787159773
Simulated Standard Deviation of MSE		0.0
Simulated Value at Risk (5th percentile)		140471.49787159773
Simulated Mean R-Squared		0.9995967866276133
Simulated Standard Deviation of R-Squared		1.1102230246251565e-16
Simulated Mean MAE		217.94527078026155
Simulated Standard Deviation of MAE		8.526512829121202e-14

The low standard deviation of the MSE across simulations indicates that the model is robust to changes in the Federal Funds Effective Rate within the range of scenarios tested. Although the mean MSE remains relatively high, consistent with initial model evaluations, it confirms that the model's performance remains stable under stress conditions. The Value at Risk (VaR) analysis further suggests that while the model's performance may degrade under extreme scenarios, this degradation is not drastic. The lack of significant variation indicates that the model's performance is less sensitive to interest rate fluctuations in the simulation.

4. Impact and Usability: Real-World Applicability

Our study underscores the critical role of high-quality, accurate data in model testing, as evidenced by the robust performance of all tested models. This research also highlights the significance of a thorough literature review, incorporating insights from academic papers, industry reports, and case studies to identify variables that have proven effective in similar predictive tasks. Additionally, the creation of new variables through transformations and combinations (e.g., lagged prices, moving averages, and ratios) has been pivotal in enhancing model implementation success.

[34] Jones, C., & Partington, R. (2024). 'Time has come' for US Federal Reserve to cut interest rates, says Powell. https://www.theguardian.com/business/article/2024/aug/23/us-feds-cut-interest-rates-powell

The rigorous model selection and evaluation process we employed, which includes both predicting observed data and forecasting unobserved real-world prices, has demonstrated that all tested models are not only theoretically sound but also practically valuable for predicting cryptocurrency prices and informing trading strategies. Our meticulous documentation of every decision, step, and result ensures transparency and reinforces the credibility of our findings, providing a strong foundation for future research.

Furthermore, by making our data and model scripts accessible via GitHub, we have prioritized both user accessibility and broad usability. The forecasting demonstrations show that our models can be seamlessly integrated into existing trading platforms, enabling financial analysts and investors to utilize advanced predictive analytics with minimal technical expertise. Our stress testing and simulation approaches enhance decision-making capabilities in financial markets, allowing stakeholders to anticipate market movements and manage risks more effectively. This commitment to openness and collaboration not only supports ongoing research but also contributes to the broader understanding and practical application of predictive analytics in the rapidly evolving field of cryptocurrency markets.

5. Limitations

While multi-factor models offer a comprehensive approach to understanding cryptocurrency pricing, they come with some limitations. First, many models assume stationary data, but financial data often exhibit trends, volatility clustering, and other non-stationary behaviors. Although we identified non-stationarity in our data, we opted not to make it stationary, preferring models that are less sensitive to non-stationary data. Similarly, we did not address multicollinearity, which can inflate standard errors and obscure the effects of individual factors. High correlations among factors can complicate interpretation and lead to less reliable estimates.

Cryptocurrencies, being relatively new compared to traditional financial assets, present challenges with limited historical data. This limited sample may not capture all market regimes and introduces risks of biased estimates and inaccurate predictions. Data quality issues, such as missing values and noise, can further impact model accuracy. Additionally, some macroeconomic variables and sentiment indicators have sparse or lower-frequency data compared to traditional asset price data.

Lastly, including numerous features in the model can increase the risk of overfitting, where the model performs well on historical data but poorly on unseen data. Financial markets, especially cryptocurrencies, are prone to sudden market shocks, which can cause significant deviations from historical patterns. As observed in our simulations and stress tests, models may struggle to adapt quickly to drastic changes in market behavior.

6. Recommendations for Further Research

This project provides a solid foundation for predicting cryptocurrency prices, but there are several ways it could be expanded and refined. First, dataset can be broadened to incorporate a wider range of cryptocurrencies beyond Bitcoin and Ethereum. The dataset can include assets across different market capitalizations – Large, Mid and Small caps – to enhance the comprehensiveness of the analysis. Further analysis can be performed based on cryptocurrency categories to uncover hidden patterns.

Moreover, trends can be added to both dataset and models. For example, the data can be divided to popular trends as Artificial Intelligence (AI) & Big Data, Real World Assets (RWA), and Meme coins. For these categories, specific cryptocurrencies can be selected. For instance, for AI & Big Data category, FET (FET, AGIX, OCEAN will be merged under ASI), RENDER and TAO can be reviewed. For RWA, assets like Ondo, Mantra and Pendle can be examined. For Meme coins, popular coins like Doge, Shib and Pepe can be examined.

Second, hybrid models can be developed that leverage the strengths of various algorithms. For instance, stacking LightGBM with XGBoost could enhance predictive performance by capturing diverse data patterns. Also, hyperparameters can be further refined for models such as LSTM, GBM, LightGBM, and XGBoost to optimize their performance.

Third, stress testing can be extended to include a broader range of scenarios, encompassing different economic conditions and market shocks. This will help assess model robustness under varied and potentially extreme conditions. Additional simulation techniques can be advanced, such as Markov Chain Monte Carlo (MCMC) or Geometric Brownian Motion (GBM), to evaluate model stability and performance under diverse conditions. By addressing these areas, future research can provide a deeper and more nuanced understanding of cryptocurrency price dynamics, enhance model accuracy, and improve resilience to market fluctuations.

7. Conclusion

This study has explored the use of multi-factor models in financial forecasting through a newly constructed dataset, with a particular focus on cryptocurrency price prediction. The literature highlights the evolution, benefits, and limitations of these models, demonstrating their adaptability and significance in understanding global market price and risk patterns.

Our work began with a basic linear regression model, which was enhanced through the application of regularization techniques such as Ridge, Lasso, and ElasticNet. We employed both traditional and time-series specific cross-validation to assess model performance, and utilized various diagnostic plots (e.g., residuals distribution, actual vs. predicted values, Q-Q plots, box plots) and correlation analyses to validate predictions and residuals behavior.

We encountered issues of multicollinearity and cointegration during data exploration, leading us to select models less sensitive to these problems. Among the regularization techniques, Lasso proved particularly effective due to its feature selection capability. Although linear regression showed a perfect R-squared value and low error metrics, these results indicated potential overfitting, suggesting that while the models fit the training data well, their generalizability to new, unseen data may be limited.

Our exploration extended to advanced models, including LSTM, XGBoost, GBM, and LightGBM. With the help of our meticulously gathered dataset with well-chosen variables based on literature review and our domain knowledge and experience, all tested models perform satisfactorily. We also leveraged AutoML tools such as T-POT, H2O.ai, and MLJAR to identify hybrid models and optimize hyperparameters. Notably, models recommended by AutoML tools—ones we might not have initially considered—have outperformed others in predictive accuracy. Stress testing with LightGBM and Monte Carlo simulations provided insights into model performance under extreme market conditions, such as sudden market crashes and changes in interest rates. Specifically, the LSTM and XGBoost models have demonstrated

exceptional capabilities in forecasting real-world prices. We observed the critical importance of hyperparameter tuning across all tested models, primarily due to its enhancing effect.

In conclusion, while our multi-factor risk models have established a robust foundation for cryptocurrency price prediction, there remains considerable room for improvement. Future research should focus on refining models, exploring additional features and scenarios, and addressing limitations to enhance prediction accuracy and reliability in the ever-evolving landscape of cryptocurrency markets.

References

Abdollahi, H. (2020). A novel hybrid model for forecasting crude oil price based on time series decomposition. *Applied energy*, *267*, 115035.

Abdollahi, H., & Ebrahimi, S. B. (2020). A new hybrid model for forecasting Brent crude oil price. *Energy*, *200*, 117520.

Baltas, A. N., & Kosowski, R. (2012). Improving time-series momentum strategies: The role of trading signals and volatility estimators. *SSRN eLibrary*.

Dash, S. K., & Dash, P. K. (2019). Short-term mixed electricity demand and price forecasting using adaptive autoregressive moving average and functional link neural network. *Journal of Modern Power Systems and Clean Energy*, *7*(5), 1241-1255.

Jin, Z., Yang, Y., & Liu, Y. (2020). Stock closing price prediction based on sentiment analysis and LSTM. *Neural Computing and Applications*, *32*, 9713-9729.

Jones, C., & Partington, R. (2024). 'Time has come' for US Federal Reserve to cut interest rates, says Powell. https://www.theguardian.com/business/article/2024/aug/23/us-feds-cut-interest-rates-powell

JPMorgan (2024). *The probability of a recession now stands at 35%.* https://www.jpmorgan.com/insights/global-research/economy/recession-probability

Kaourma, T. C. (2020). Essays on retail investors trading behavior in FX markets.

Liu, Y., Yang, C., Huang, K., & Gui, W. (2020). Non-ferrous metals price forecasting based on variational mode decomposition and LSTM network. *Knowledge-Based Systems*, *188*, 105006.

Marchesi, F. (2024). Redefining Financial Forecasting: A CAPE-Based Approach to S&P 500 Analysis and Portfolio Construction.

Suaza-Medina, M. E., Zarazaga-Soria, F. J., Pinilla-Lopez, J., Lopez-Pellicer, F. J., & Lacasta, J. (2023). Effects of data time lag in a decision-making system using machine learning for pork price prediction. *Neural Computing and Applications*, *35*(26), 19221-19233.

Wu, J. M. T., Li, Z., Herencsar, N., Vo, B., & Lin, J. C. W. (2023). A graph-based CNN-LSTM stock price prediction algorithm with leading indicators. *Multimedia Systems*, *29*(3), 1751-1770.

Zhou, F., Huang, Z., & Zhang, C. (2022). Carbon price forecasting based on CEEMDAN and LSTM. *Applied energy*, *311*, 118601.

Appendices

Appendix A

Data Codebook

(in order of appearance)

Bitcoin (BTC) Prices

Bitcoin Price Data from Yahoo Finance including Date, BTC_Open, BTC_High, BTC_Low, BTC_Close (Deleted as it is too close to BTC_Adj Close), BTC_Adj Close, BTC_Volume.

Ethereum (ETH) Prices

Ethereum Price Data from Yahoo Finance including Date, ETH_Open, ETH_High, ETH_Low, ETH_Close (Deleted as it is too close to ETH_Adj Close), ETH_Adj Close, ETH_Volume.

Total Lightning Network Nodes

The number of nodes with, without channels and the total. Lightning nodes open payment channels with each other that are funded with bitcoin. When transactions are made across those channels, the channel balance is reflected without having to broadcast a transaction on chain. This creates a second layer on top of the bitcoin network that expands it capabilities. BitcoinVisuals.com

Crypto Fear & Greed Index

Crypto Fear & Greed Index from alternative.me.

The crypto market behavior is very emotional. People tend to get greedy when the market is rising which results in FOMO (Fear of missing out). Also, people often sell their coins in irrational reaction of seeing red numbers. With Fear and Greed Index, it is aimed to save investors from their own emotional overreactions. There are two simple assumptions:

Extreme fear can be a sign that investors are too worried. That could be a buying opportunity.

When Investors are getting too greedy, that means the market is due for a correction.

The index analyzes the current sentiment of the Bitcoin market and crunch the numbers into a simple meter from 0 to 100. Zero means "Extreme Fear", while 100 means "Extreme Greed".

Lightning Network Channels

Unique = channels connecting nodes directly for the first time. Duplicate = channels between nodes that are already connected. Total Lightning Network Channels. BitcoinVisuals.com

Lightning Network Capacity (BTC & USD)

The cumulative bitcoin capacity across all channels. Lightning nodes open payment channels with each other that are funded with bitcoin. When transactions are made across those channels, the channel balance is reflected without having to broadcast a transaction on chain. This creates a second layer on top of the bitcoin network that expands it capabilities. BitcoinVisuals.com

Lightning Network Capacity per Channel

Daily median capacity per channel statistics. BitcoinVisuals.com

Lightning Network Eccentricity

Eccentricity is a distance measure of the network. It's the maximum number of hops required to reach another node (among shortest paths). BitcoinVisuals.com

Lightning Network Density

Density is a measure of completeness of the network. It's a ratio of actual / potential channels. BitcoinVisuals.com

Lightning Network Clustering

Clustering coefficient is the ratio of interconnections between a node's peers. A value of 0 means the node is a hub, and none of its peers are connected. A value of 1 means the node forms a clique with its peers. BitcoinVisuals.com

Lightning Network Cut Channels

A cut channel (aka cut edge, or bridge) is a channel between two nodes that connects different components of the network. This channel's removal would prevent other nodes from having a path. BitcoinVisuals.com

Lightning Network Channels per Node

Daily median channels per node statistics. BitcoinVisuals.com

Lightning Network Capacity per Node

Daily median capacity per node statistics. BitcoinVisuals.com

BTC Trading Volume

Daily bitcoin trading volume and market dominance. Volume reflects a 24-hour period of time. Dominance is a measure of bitcoin volume versus the entire cryptocurrency market. Historical volume goes back to 2013. BitcoinVisuals.com

BTC Dominance & Market Capitalization

Bitcoin market capitalization and market dominance. Capitalization = total supply * price.

Number of Crypto Currencies

Number of different crypto currencies / coins in the market.

Number of Crypto Market Pairs

Number of different crypto market pairs that exist. For example BTC/ETH (Bitcoin/Ethereum) would count as a market pair.

Number of Crypto Exchanges

Number of different crypto exchanges that exist.

Lightning Network Transitivity

Transitivity is a measure of clustering. It's the ratio of potential triangles present. A value of 1 means every path of length 2 loops back into a triangle.

Lightning Network Cut Nodes & Percent of Total

A cut node (aka cut vertex) is a node that connects different components of the network. This node's removal would prevent other nodes from having a path.

Total Node Bandwidth

Data usage of BitcoinVisuals.com's Bitcoin Core node. The node is unpruned and transaction-indexing. All other settings are default.

Blockchain Size and Annualized Growth Rate

Cumulative size of bitcoin blocks (serialized block headers and transactions).

Block Reward Per Block (BTC & USD)

The reward miners get for mining a block (excluding transaction fees). Started at 50 BTC and halves every 210,000 blocks. The block reward is how new bitcoin is "minted" or brought into the economy.

Block Speed

Daily median time between bitcoin blocks. Excludes first day.

Total Block Size

Daily median block size (serialized block headers and transactions). "Stripped" refers to data sent to legacy nodes that has witness data removed and stays under the 1MB cap. "Witness" is the size of witness data sent to segwit-compliant nodes, which combined with "stripped" data can be over 1MB.

Block Weight

Daily median block weight. Weight is a measure of the size of bitcoin blocks including the segwit discount. Weight = (tx size with witness data stripped) * 3 + (tx size). Each block has a limit of 4M weight units or 1M vbytes (1 vbyte = 4 weight units).

Transactions Per Second

Calculation = transactions per day / 86400.

Transactions Per Day

Daily transaction totals, excluding coinbase transaction (miner reward). Split between segwit and non-segwit.

Daily Median Transaction Size

Daily median transaction size statistics per block, excluding coinbase transaction (miner reward).

Daily Median Transaction Virtual Size

Daily median transaction virtual size statistics per block, excluding coinbase transaction (miner reward). Virtual size includes the segwit discount.

Total Transactions Per Block

Daily median transactions per block, excluding coinbase transaction (miner reward). Split between segwit and non-segwit.

Total Bitcoin Supply

Total bitcoin supply issued through block rewards which halve every 210,000 blocks.

BTC Annualized Inflation Rate

Inflation rate is annualized.

Mining Hash Rate

The data is an estimate of how many hashes per second bitcoin miners are performing on the network. Estimate = difficulty * 232 / time. The bitcoin network has a global block difficulty that adjusts every 2016 blocks (~2 weeks) based on a target time of 10 minutes per block. As difficulty increases, more hashpower must be added to have the same statistical chance of finding a block. The time between bitcoin blocks can vary dramatically if there is a large increase or decrease in hashpower within this 2 week period.

Bitcoin Difficulty & Blocks Per Day

The bitcoin network has a global block difficulty that adjusts every 2016 blocks (~2 weeks) based on a target time of 10 minutes per block. Valid blocks must have a hash below this target, therefore difficulty is a measure of how difficult it is to find this hash. As difficulty increases, more hashpower must be added to have the same statistical chance of finding a block. The time between bitcoin blocks can vary dramatically if there is a large increase or decrease in hashpower within this 2 week period.

Block Height

The total number of blocks in the chain (aka height), and the daily rate of growth.

Miner Revenue & Puell Multiple

The Puell multiple is a way to gauge market cycles from a mining profitability / compulsory sellers' perspective. It takes total miner revenue and adjusts by its yearly moving average. Calculation = mining revenue / 365-day simple moving average of mining revenue.

Required Transaction & Block Size for 1 Cent Fee Reward

Assume the transaction fee is fixed at $.01. How many transactions and what block size would someone need to match bitcoin's fee reward per block? Median tx size is used (not including segwit discount).

Minimum Tx Fee (Satoshis & USD)

The minimum fee per 200 bytes to be included in BitcoinVisuals.com's node's mempool (maximum of minrelaytxfee and minimum mempool fee). They use default Bitcoin Core settings.

Memory Usage

BitcoinVisuals.com's node's memory usage for unserialized mempool data. They run default Bitcoin Core settings.

Unconfirmed Transactions - Mempool

The memory pool (aka mempool) is a node's view of unconfirmed transactions that have been broadcast to the network. Not all nodes will have the same mempool due to the amount of memory they've allocated, and specific rules that may censor transactions.

This data is BitcoinVisuals.com's node's unconfirmed transactions and sum of their serialized virtual sizes (i.e. includes segwit discount). They run default Bitcoin Core settings.

Total Bitcoin Node Peers

Peers connected to BitcoinVisuals.com's Bitcoin Core node. The node is unpruned and transaction-indexing. All other settings are default.

Block Reward Per Day (BTC & USD)

The reward miners get for mining a block (excluding transaction fees). Started at 50 BTC and halves every 210,000 blocks. The block reward is how new bitcoin is "minted" or brought into the economy.

Daily Median Transaction Fees Per Tx (USD)

Daily median transaction fee statistics per transaction and block, excluding coinbase transaction (miner reward).

Daily Median Transaction Fees Per Tx (BTC)

Daily median transaction fee statistics per transaction and block, excluding coinbase transaction (miner reward). A bitcoin fee is added by the user to a transaction to incentivize miners to include that transaction in their block. Block space is limited, so bitcoin fees will change based on demand.

Total Transaction Fees Per Day (USD & BTC)

Total transaction fees per day, excluding coinbase transaction (miner reward).

Total Transaction Fees Per MB (BTC & USD)

How much does blockchain usage cost per megabyte? To estimate that, we first calculate the median number of transactions per MB and then multiply that by the median transaction fee. Virtual sizes are used (includes segwit discount). Calculation = (1,000,000 / median tx vsize) * median tx fee.

Daily Median Inputs Per Transaction and Block

Daily median input statistics per transaction and block, excluding coinbase transaction (miner reward).

Total Transaction Inputs Per Day

Total transaction inputs per day, excluding coinbase transaction (miner reward).

Daily Median Outputs Per Transaction and Block

Daily median output statistics per transaction and block, excluding coinbase transaction (miner reward).

Total Transaction Outputs Per Day

Total transaction outputs per day, excluding coinbase transaction (miner reward).

Daily Median Input Vol Per Transaction and Block

Daily median input volume statistics per transaction and block, excluding coinbase transaction (miner reward).

Total Transaction Input Vol Per Day

Total transaction input volume per day, excluding coinbase transaction (miner reward).

Daily Median Output Vol Per Transaction and Block

Daily median output volume statistics per transaction and block, excluding coinbase transaction (miner reward). Outputs = inputs - fees.

Total Transaction Output Vol Per Day

Total transaction output volume per day, excluding coinbase transaction (miner reward). Outputs = inputs - fees.

S&P Cryptocurrency Broad Digital Market Index

The index measures the performance of digital assets satisfying the additional liquidity and market capitalization requirements

BTC Lagged Prices

Financial time series, like cryptocurrency prices, often exhibit autocorrelation, meaning that past prices are correlated with future prices. By creating including lagged prices, we have aimed to capture this relationship.

BTC Moving Averages

Short-term Trading Models: Tend to use shorter lags (1-day, 2-day) and short-term moving averages (MA5, MA10) to capture immediate price changes.

Medium-term Models: Use 10-day, 20-day lags and MA20, MA50 to identify trends over weeks or months.

Long-term Investment Models: Focus on longer lags (50-day, 100-day) and long-term moving averages (MA100, MA200) to capture broader market cycles and trends.

Rate of Change (ROC)

Measures the percentage change in price over a specified period. Useful for identifying the speed and magnitude of price movements.

Relative Strength Index (RSI)

Measures the magnitude of recent price changes to evaluate overbought or oversold conditions. Typically calculated over a 14-day period. Values range from 0 to 100, with levels above 70 indicating overbought and below 30 indicating oversold conditions.

Standard Deviation (Volatility)

Measures the dispersion of price values over a specific period, in our case window=20. Higher values indicate higher volatility.

Average True Range (ATR)

Measures market volatility by considering the entire range of an asset's price for a given period, in our case window=14. Helps identify periods of high and low volatility.

Bollinger Bands

Consists of a moving average and two standard deviations (bands) above and below it. Used to identify overbought or oversold conditions and volatility (window=20).

Exponential Moving Average (EMA)

Similar to the simple moving average (SMA) but gives more weight to recent prices. Faster to react to recent price changes than the SMA (in our case, 12-day and 26-day).

On-Balance Volume (OBV)

Measures buying and selling pressure by adding volume on up days and subtracting it on down days. Used to confirm price trends and predict potential reversals.

Stochastic Oscillator

Compares a particular closing price to a range of prices over a certain period, in our case window=14. Indicates overbought or oversold conditions (similar to RSI).

Moving Average Convergence Divergence (MACD)

Combines two EMAs (in our case, 12-day and 26-day) and subtracts the longer EMA from the shorter EMA. Includes a signal line (9-day EMA of the MACD) to identify buy/sell signals. Often used to identify momentum and potential reversals.

Parabolic SAR

Indicates potential price direction and helps identify potential entry and exit points.

Williams %R

A momentum indicator that shows the level of the closing price relative to the highest high over a look-back period, in our case 14 days. Values range from -100 (most oversold) to 0 (most overbought).

Commodity Channel Index (CCI)

Measures the difference between the current price and the historical average price. Useful for identifying cyclical trends in the market (window=20).

Accumulation/Distribution Line (A/D Line)

Combines price and volume to assess whether an asset is being accumulated or distributed.

Chaikin Money Flow (CMF)

Combines the A/D line with volume, indicating the buying and selling pressure over a specified period (usually 20 days).

True Strength Index (TSI)

Combines a momentum indicator and a smoothing factor to identify the strength of a trend.

Average Directional Index (ADX)

Measures the strength of a trend, regardless of its direction. Often used with +DI and -DI (Directional Indicators) to indicate the trend's direction (window=14).

Price Rate of Change (PROC)

Similar to ROC, but can be calculated over different periods to capture various time horizons (periods=20).

Z-Score

Indicates how many standard deviations a value is from the mean, often used to normalize the price data (window=20).

Volume-Based Moving Averages

Similar to price moving averages, but calculated using volume data to capture volume trends (window=30).

Log Returns

Provides normalized return values.

Cumulative Returns

Tracks the total return over time.

Sharpe Ratio

Measures return adjusted for risk.

Consumer Price Index for All Urban Consumers (CPI-U)

The Consumer Price Index for All Urban Consumers: All Items (CPIAUCSL) is a price index of a basket of goods and services paid by urban consumers. Percent changes in the price index measure the inflation rate between any two time periods.

Federal Funds Effective Rate

The federal funds rate is the interest rate at which depository institutions trade federal funds (balances held at Federal Reserve Banks) with each other overnight. When a depository institution has surplus balances in its reserve account, it lends to other banks in need of larger balances. In simpler terms, a bank with excess cash, which is often referred to as liquidity, will lend to another bank that needs to quickly raise liquidity.

Emerging Markets Index

An index that tracks the performance of stocks in emerging markets, often used to gauge investment opportunities in developing economies.

MSCI World Index

An index that tracks the performance of large and mid-cap stocks across 23 developed countries, providing a broad view of global equity markets.

Gold

The price of gold futures, a widely followed commodity often considered a safe-haven asset during economic uncertainty.

US Dollar Index

An index that measures the value of the U.S. dollar relative to a basket of six major world currencies, often used to gauge the strength of the U.S. dollar.

US 10-Year Treasury Yield

The yield on the U.S. government's 10-year Treasury bond, a key indicator of long-term interest rates and economic expectations.

US 30-Year Treasury Yield

The yield on the U.S. government's 30-year Treasury bond, often used as a benchmark for mortgage rates and long-term investments.

CAC 40 (France)

An index that tracks 40 of the largest publicly traded companies in France, providing insight into the health of the French economy.

Nikkei 225 (Japan)

A price-weighted index that tracks 225 of the largest publicly traded companies in Japan, widely considered the benchmark for the Japanese stock market.

DAX (Germany)

An index that tracks the 30 largest and most liquid German companies traded on the Frankfurt Stock Exchange, representing the performance of the German economy.

FTSE 100 (UK)

An index that tracks the 100 largest companies listed on the London Stock Exchange. It is a key indicator of the UK stock market.

Russell 2000

An index that tracks the performance of 2,000 small-cap companies in the U.S., often used as a benchmark for the small-cap market.

NASDAQ Composite

An index that includes more than 3,000 stocks listed on the NASDAQ stock exchange, heavily weighted towards technology and biotech companies.

Dow Jones Industrial Average

A price-weighted index that tracks 30 large, publicly-owned companies in the U.S. This index is one of the oldest and most followed indicators of the U.S. stock market.

S&P 500

A stock market index that tracks the 500 largest publicly traded companies in the U.S. It is widely regarded as one of the best indicators of the U.S. stock market's performance.

VIX (CBOE Volatility Index)

Often referred to as the "fear gauge," this index measures market expectations of near-term volatility in the S&P 500 index, derived from options prices.

World Uncertainty Index (WUI)

This index measures overall uncertainty across the globe. The index is unbalanced GDP weighted average for 142 countries.

World Trade Uncertainty Index (WTUI)

This index measures trade uncertainty across the globe. The index is an equally weighted average for 143 countries.

Relevance: Captures global economic and political uncertainty, providing insight into how global events impact financial markets, including cryptocurrencies. It’s a broad measure that can capture various sources of risk.

World Pandemic Uncertainty Index (WPUI)

This dataset includes the World Pandemic Uncertainty Index (WPUI) at the global and country level. It also includes an index that measures discussions about pandemics at the global and country level.

Relevance: Specifically measures uncertainty related to pandemics, which has become increasingly relevant since COVID-19. Pandemics can disrupt economies and drive investors to hedge with assets like cryptocurrencies.

US Equity Market Volatility Index

https://www.policyuncertainty.com/EMV_monthly.html

Relevance: Often referred to as the "fear gauge," this index reflects market expectations of volatility. Since cryptocurrencies are influenced by broader market sentiment, this index can be a key indicator of market conditions.