Predicting Bitcoin Returns Using Bitcoin Metrics and Macroeconomic Indicators

Justin Su **Nick Longueira** **Max Nemecek** **Abhimanyu Agashe** **Wally Lutz**

Abstract

This project investigates the potential to predict Bitcoin (BTC) returns using a combination of BTC-specific metrics and macroeconomic indicators. With recent U.S. policy proposals—including the creation of a national Bitcoin Reserve and increased regulation—the relevance of such predictive models is growing. As Bitcoin adoption expands, accurate return forecasting tools could benefit investors, financial institutions, and policymakers by enabling more informed decisions and contributing to market stability. This study aims to develop a model that captures key drivers of Bitcoin performance across both crypto-native and traditional economic dimensions.

1 Motivation

1.1 Initial Question

Can we predict Bitcoin (BTC) returns using BTC-specific metrics and macroeconomic indicators?

1.2 Relevance

The relevance of this question has become even more significant with recent developments in U.S. policy. The U.S. government has recently proposed initiatives to create a national Bitcoin Reserve and enforce new cryptocurrency regulations. This signals a shift toward a greater overall acceptance of Bitcoin. As adoption grows, so will the number of prospective investors. With this being said, models capable of accurately forecasting Bitcoin returns could become extremely useful tools for investors, financial institutions, and governments. Reliable prediction models would not only help individuals and institutions make informed investment decisions, but they could also contribute to stabilizing Bitcoin markets by reducing uncertainty and better modeling volatility. By incorporating both BTC metrics and macroeconomic indicators, this approach aims to provide an advantageous view of the factors influencing Bitcoin's performance.

1.3 Related Works

There are many publications on predicting the trends of Crypto using Machine learning. There is a 2018 study using a dataset of Bitcoin prices starting from August 2013 to predict the future prices of Bitcoin (1). There is a similar 2023 study that uses random forest regression and LSTM methodology to predict future prices of Bitcoin (2). These studies concluded that bitcoin prices follow an autoregressive process that responds to the FAMA-French momentum factor.

There have also been several studies attempting to link cryptocurrency trends to financial market indicators. One 2024 analysis established granger causality between the fear and greed index (FGI), which measures stock market volatility, and the returns of Bitcoin and Ethereum (3). These findings supported those of a 2020 study, which concluded that various financial market indicators, such as stock and gold prices had an impact on cryptocurrency returns (4).

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

2 Data

2.1 BTC Dataset

Our Bitcoin dataset contained observations of Bitcoin prices from 7/17/2010 to 1/1/2025 on a daily basis. The features of this dataset included: high prices (high), low prices (low), date (date), open prices (open), and close prices (close). For our purposes, we decided to use 1/1/2015 - 1/1/2025 as our range because it more accurately reflects the current fluctuation of BTC. This dataset was found on Kaggle:

https://www.kaggle.com/datasets/svaningelgem/crypto-currencies-daily-prices

2.2 Macroeconomic Metrics Dataset

Our macroeconomic metrics datasets consisted of the following indicators: DFF (risk free effective rate), DAAA (Moody's triple A bond yields), UNRATE (unemployment rate), WRESBAL (federal reserves), and VIX (implied S&P 500 volatility). These indicator datasets were found on: Federal Reserve Economic Data | FRED | St. Louis Fed

2.3 Time Series Validity

Stationary In order to model time series data using lagged values of the dependent variable, it must have stationary behavior. If a series has a unit root, where the coefficient on the lagged variable in an autoregressive process is greater than or equal to one, past values are amplified and estimation will not converge properly. As a result, we are unable to model price directly, as it does not have a constant mean. Instead, we attempt to model log returns, which are generally stationary in financial analysis. To confirm, an augmented Dickey-Fuller test was performed, which gives evidence at the 1% level to reject the null hypothesis of a unit root.

Transformations In our actual implementation, we decided to transform the close column from our BTC dataset and turn it into our log return column, which was the target variable. We used this equation to compute log returns:

$$log\_returns = log(close_i/close_{i-1})$$

3 Files and Code

All of the data files and code used in this project can be found in this Github Repository

4 Models

Our question aimed to answer whether we could accurately predict Bitcoin returns given the predictors we have mentioned thus far. Our approach to answering this question involved trying to fit several different models, using feature selection methods for these models, training the models and the first 80% of the data sequentially, testing the models in the remaining 20%, and then comparing the evaluation metrics among the models (MAE, RMSE, R^2).

4.1 Vanilla RNN

We decided to use the standard RNN from Pytorch to create our Vanilla RNN model as a baseline model from which we will compare the rest of our models. We chose to start with RNN models because they naturally handle the sequential nature of the data. After validation, we decided on using a lag size of ten steps and a learning rate of 0.001. Using the model, we found that the model yielded moderate results, with a RMSE of 0.027737, a MAE of 0.020139, and a R^2 of -0.2766.

4.2 LSTM

As an extension of our RNN model, we also decided to use the LSTM (Long Short Term Memory) model also from the Pytorch package. LSTM adds additional features from the basic RNN, including

a memory cell that allows the nodes in the neural network to retain information from previous nodes, preventing gradient loss. After validation, we found that LSTM was more sensitive to overfitting due to the additional features that the model includes. In response, we chose to lower the learning rate to 0.0005 and set the momentum to 0.0001. Using the model, we found the model yielded better results compared to Vanilla RNN, with RMSE of 0.026962, MAE of 0.019231, and R^2 of -0.2062.

4.3 Random Forest

We investigated whether Bitcoin (BTC) returns could be predicted using a combination of BTC metrics and macroeconomic indicators using Tree models. Given Bitcoin's extreme volatility, frequent outliers, and nonlinear nature, we selected tree-based models, specifically Random Forests, because they can naturally handle these challenges without the need for additional scaling or transformation.

Our models were fit to predict the log return of Bitcoin from one day to the next day. For feature selection, we initially chose: Bitcoin's daily open, high, low, and close prices, the federal funds rate (DFF) and the corporate bond yield (DAAA). These features were selected based on economic intuition: Bitcoin price movements are often influenced by risk sentiment in the economy, which DFF and DAAA can capture.

In our first model, we used a basic approach with no hyperparameter tuning, with a one-day lag for each column. We split the dataset into a sequential 80% training and 20% testing split. This first model yielded poor results, with a MAE of 0.061281 and a RMSE of 0.069117. The R^2 value was -3.971885, indicating that the model performed worse than using just the mean. This suggested that more historical information and better model tuning would be needed.

Based on the first model, we improved this model by introducing more feature engineering. We changed the lagged features, using information from the previous five days for each predictor. Also, We computed rolling statistics such as a 5-day moving average of log returns, a 5-day moving average of prices, and a 5-day rolling volatility measure to capture short-term momentum. We then performed hyperparameter tuning using a grid search over the number of estimators and tree depth. We used TimeSeriesSplit for cross-validation so sequentiality was upheld. This second model showed improvement, with an MAE of 0.021453, an RMSE of 0.029691, and an R^2 of 0.180672. This positive R^2 indicated that the model could now explain a meaningful portion of the variation in Bitcoin's returns, although there was still considerable unpredictability.

See figure 1 to compare actual vs. predicted log returns over time for the more successful model.

4.4 Prophet

In addition to the previous models, we decided to use a Prophet model to investigate Log BTC returns as well. Prophet is a forecasting model developed by Meta that performs best on time series data formatted as daily observations. Since our data is formatted in this way, we decided it would be worthwhile to attempt to use this model on our data. The parameters for the model were the date and the highest and lowest price of BTC on each date, and it predicted the daily log return of BTC. We achieved an MAE of 0.0208, an RMSE of 0.0291, and an R^2 of -0.08654. The model performed similarly to a number of other models that we attempted to fit on this dataset in that it seems to inadequately capture the high variance of the BTC dataset.

4.5 XGBoost

We attempted to investigate the behavior of returns and log returns via the XGBoost model.

Motivation XGBoost is a universalized framework for Gradient Boosted Trees, a method of Tree-based regression where older more inaccurate trees are consistently replaced with more accurate trees with iterations of gradient boosting slowly increasing accuracy. Gradient Boosting sacrifices the interpretability of tree-based regression but is able to add a higher level accuracy. XGBRegressor

performing excellently on close data provided hope that it would perform well on returns and log returns data, so using the first 80 percent of the data as train and last 80 percent as test, we moved forward with the XGB implementation.

Inference and Conclusion for XGBoost Purely looking at the R^2 does not provide great results, which are negative for both Returns and Log Returns, but when we consider our lack of knowledge about the mean of future returns, we can rely on metrics other than the R^2. We have relatively low MAE's and MSE's in both cases, but we can definitely recognize some degree of overfitting by the iterative nature of XGBoost, considering we achieve worse results than some base models XGBoost is an upgrade on. Visual inference of log returns data is positive, however, with predicted peaks and dips regularly matching actual peaks and dips, respectively.

4.6 Variance Model

Motivation Under CAPM or a similar financial market theory, rational investors maximize the Sharpe ratio of their risky assets, equivalent to the ratio of that asset's excess return over the standard deviation of returns. Therefore, investors are equally interested in predicting the volatility of BTC returns. Extremely risky assets like bitcoin usually display volatility clustering. Additionally, since White's test for heteroskedasticity gives evidence at the 1% level to reject the null of constant variance, we can attempt to model potential changes in volatility.

Models For interpretability, the variance models use standard returns scaled by 100, rather than log returns. We use the absolute value of returns as a proxy for true observed variation. First, we estimate bitcoin as an autoregressive process of order one, with an additional exogenous VIX variable, which has been shown to correlate with cryptocurrency performance. Then, we estimate a GARCH(1,1) variance model assuming t-distributed errors on the conditional variance from the AR(1) process. Past analyses have found that GARCH(1,1) consistently performs better than other conditional variance models on asset returns data. This includes a constant, the lagged squared error, and the lagged conditional variance. Finally, to forecast this model to the test set, we estimate the model recursively for each day t in order to generate the predicted volatility given all of the information up to $t-1$.

Results As a baseline, fitting the mean absolute return of the training set on the test set gives an MAE of 2.3021. The GARCH predictions give an MAE of 2.1914, which is a slight improvement, but not enough to conclude that it is viable. Despite this weak performance, the graph shows that the model appears to capture overall patterns well, but cannot capture the full range of variation in the highly noisy data. Extremely high outliers also distort the predictions, resulting in overprediction overall. Censoring these outliers may improve the performance significantly, though could result in bias.

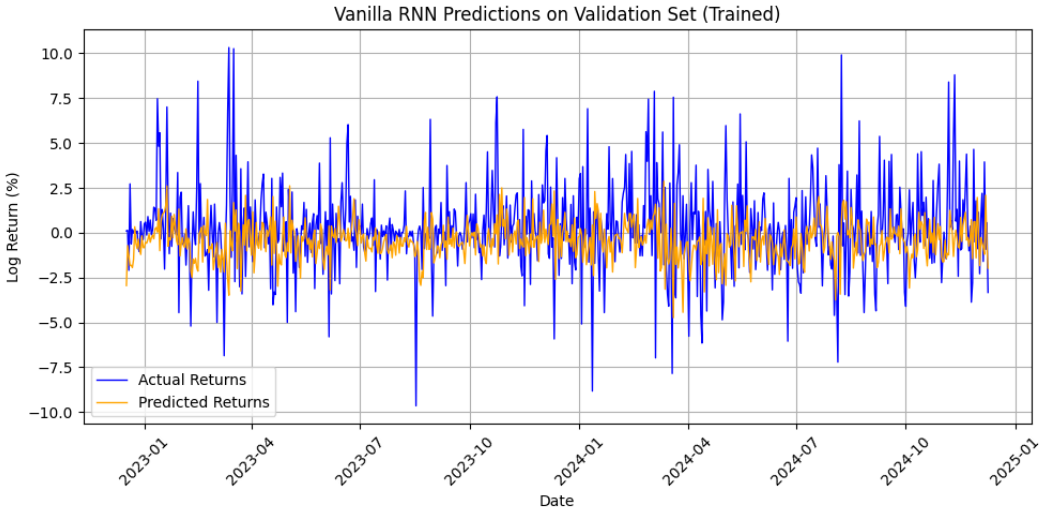

Figure 1: Vanilla RNN

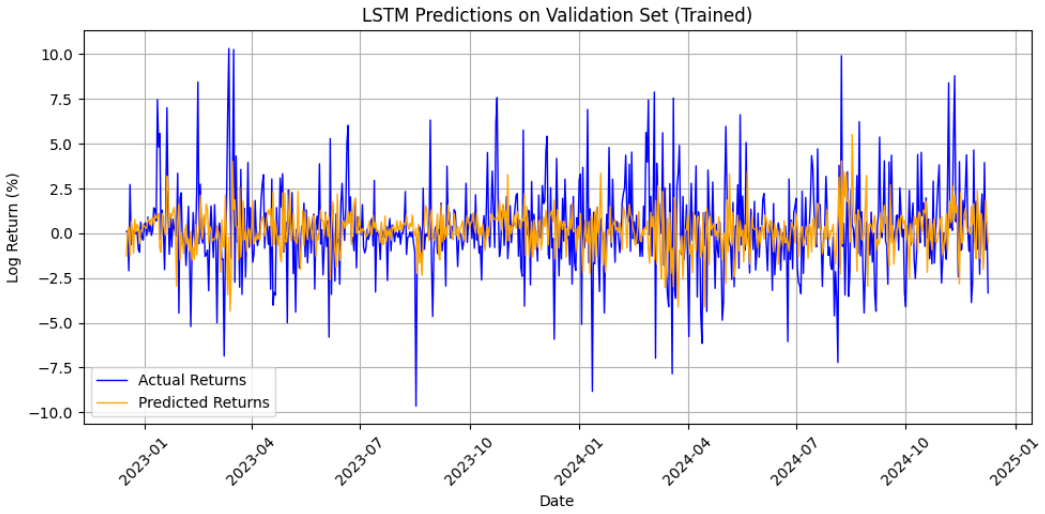

Figure 2: LSTM

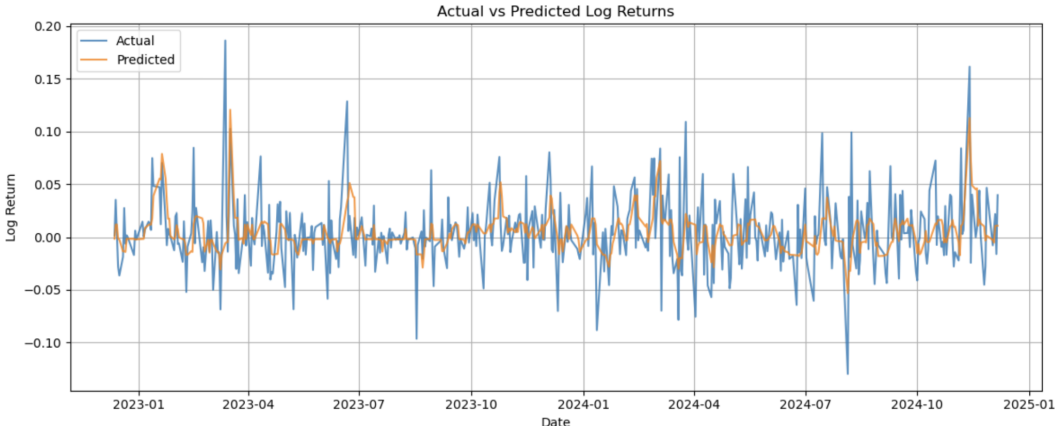

Figure 3: Random Forest

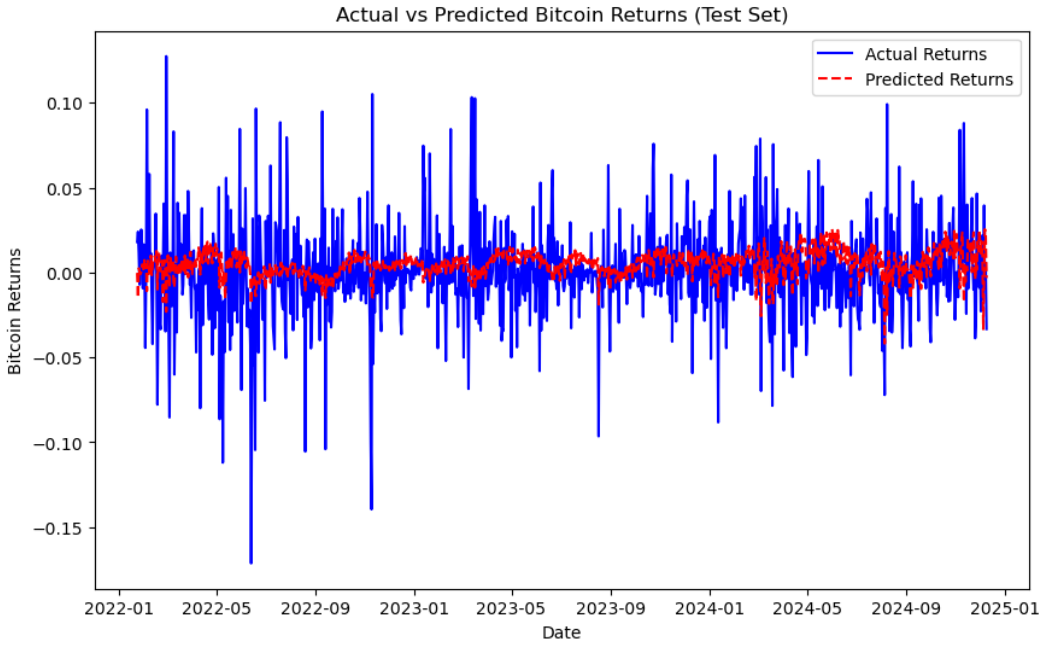

Figure 4: Prophet

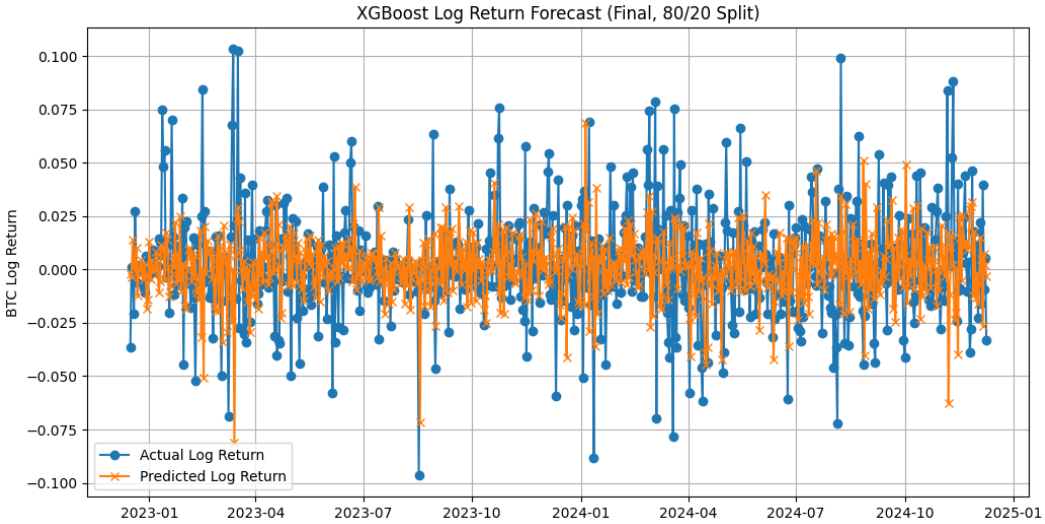

Figure 5: XGBoost

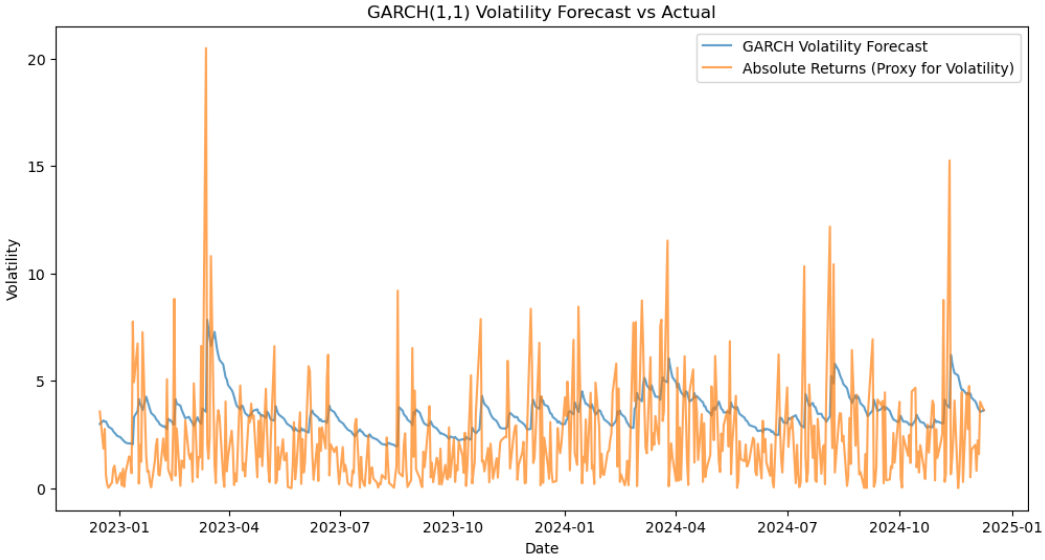

Figure 6: Garch variance model

5 Result Comparison

Table 1: Model Performance Metrics

Model	MAE	RMSE	R^2
Vanilla RNN	0.02139	0.027737	-0.2766
LSTM	0.019231	0.026962	-0.2062
Random Forest	0.021453	0.029691	0.180672
Prophet	0.0208	0.0292	-0.08654
XGBoost	0.020956	0.028244	-0.3210

Looking at each model, the MAE and RMSE values for each are all relatively similar. There is no clear winner based on these two metrics; however, our random forest model was the only one to yield a positive R^2 value. While its value of 0.180672 is not traditionally impressive for models fit on cross-sectional data, since we were working with time series data it is not as insignificant as appears at first glance. It implies that the random forest model does capture a meaningful portion of the variation of log BTC returns. Considering how intense this variation is day-to-day, the fact that this model captures any of this variation constitutes an important feat.

In addition, the GARCH model predicting BTC volatility instead of log returns managed to capture some of the general trends of BTC volatility. It achieved an MAE of 2.1914 that, combined with the graph of the actual volatility of the test set compared to the predicted volatility, reveals that the model captures some of the general volatility trends within the data. It does not perform extraordinarily well, but it is clear that at least some of the general trends are modeled.

6 Next Steps

As we can see, most of our models, while showing promising results in the sense that they could model some of the volatility and directions of the Bitcoin returns, resulted in quite poor evaluation metrics like MAE, RMSE, and R squared. Given these results, we see that we may need to implement additional tweaks to our models that allow them to capture some of the inherent characteristics of Bitcoin.

The first attribute of Bitcoin's trends is the heavy amount of outliers. Quantile regression does not assume that our returns will be normally distributed which allows for the model to better accommodate a heavily skewed set like Bitcoin returns. It does this by assigning a larger penalty term to extremely high prices and to extremely low prices. This will result in outliers not influencing the models fit as significantly. Another characteristic inherent to Bitcoin is its non-stationarity and its tendency to go through cycles based on varying market conditions. Regime switching models allow for users to fit numerous different models dependent on the state of the surrounding market. An example of this would be taking into consideration the various different states of the surrounding market such as Bull Markets. This could be more accurate than relying on the basic macroeconomic indicators that we considered as we started this project. So while we could expand our list of macroeconomic indicators, it seems as if other studies have implemented ways for the models themselves to function based specifically off of different market states. In addition to these possibilities, we understand that our tree-based models seem to lend themselves well to these characteristics of Bitcoin, so it is likely that we could implement these additional changes within the tree-models. Finally, we noticed that we did not implement interaction terms in any of our models and realized that this could potentially improve our models by capturing some of the relationships that our parameters have with one another.

After considering our model performances and our overall question, we understand that while it is impressive to reach a positive R squared value and solid MAE and RMSE metrics for our tree model, Bitcoin itself is becoming institutionalized in a way that could introduce numerous new factors that are unseen to us at this moment. With this being said, as Bitcoin returns continue to be modeled,

these new external factors must be considered.

Introduction To The Quantile Regression Model - Statistical Modeling and Forecasting
Regime-Switching Models: Capturing Structural Changes in Time Series

References

(1) S. McNally, J. Roche and S. Caton, "Predicting the Price of Bitcoin Using Machine Learning," 2018 26th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP), Cambridge, UK, 2018, pp. 339-343, doi: 10.1109/PDP2018.2018.00060.

(2) Chen, Junwei. 2023. "Analysis of Bitcoin Price Prediction Using Machine Learning" Journal of Risk and Financial Management 16, no. 1: 51. https://doi.org/10.3390/jrfm16010051

(3) Everton, A. C., Kelmara, M. V., and Thue, P. S. (2024). The impact of investor greed and fear on cryptocurrency returns: a Granger causality analysis of Bitcoin and Ethereum. Review of Behavioral Finance, 16(5), 819-835. https://doi.org/10.1108/RBF-08-2023-0224

(4) Xiao, H., and Sun, Y. (2020). Forecasting the Returns of Cryptocurrency: A Model Averaging Approach. Journal of Risk and Financial Management, 13(11), 278. https://doi.org/10.3390/jrfm13110278