Non-Linear Time Series Methods for Understanding Bitcoin Pricing

Andrew Tweedell[1]

[1]Khoury College of Computer Science
Northeastern University, Boston MA

Abstract

The purpose of this project is to develop models to understand and forecast Bitcoin daily closing prices. Cryptocurrencies, like Bitcoin, present a particular challenge due to their high volatility and complex non-linear characteristics. Models such as ARMAX-GARCH, LSTM Neural Networks, and Generalized Additive Models were researched and evaluated as potential techniques for producing a 7 day forecast for Bitcoin pricing. The root mean square errors for the log returns on pricing for the ARMAX-GARCH, LSTM, and GAM were 0.014, 0.018, and 0.018, respectively. Interestingly, the ARMAX-GARCH, a simplistic autoregressive linear model, proved to be the best option even compared to the other highly non-linear options. There are plenty of alternate techniques for data partitioning, feature transformations, and seasonal decomposition for forecasting financial stocks, which could improve our Bitcoin estimation but further research is needed.

This paper is divided into 5 sections; 1) Introduction, 2) Methods, 3) Results, and 5) Discussion.

Introduction

Bitcoin The technological and economic capabilities of cryptocurrencies are growing rapidly. Bitcoin is one of the most dominant cryptocurrencies actively traded, with a market cap of \$528B! With the earning potential inherent in Bitcoin right now, it is no wonder why many are developing models to predict and exploit its pricing. However, a slightly more holistic understanding of Bitcoin price in context is also beneficial. Understanding Bitcoin price and its relationships with exogenous variables could help cryptocurrency portfolio optimization, inform policies based on economic versus environmental impact, and inform cryptocurrency marketing strategies. Thankfully, the Monash Time Series Repository (Godahewa, et al. 2021) has a curated dataset which includes synchronized exogenous variables which may be related to cryptocurrency prices, such as the number of Google trends and the number of active addresses exchanging Bitcoin.

Economic time series analysis and forecasting as a field is well established; however, different methods have different strengths and weaknesses. Linear methods (including autoregressive (AR) models) are simple and explainable, but often times are not flexible enough. To the contrary, neural networks, like the Long-Short Term Memory (LSTM) are extremely flexible and make few assumptions about data distribution parameters; however, they are very opaque. There are non-linear methods, such as Generalized Additive Models (GAM), that offer a middle ground with better flexibility while maintaining explain-ability. However, cryptocurrencies behave differently than traditional economic indicators and research on non-linear model performance for cryptocurrencies is limited.

Purpose Thus, the purpose of this research is to 1) examine trends and relationships in Bitcoin pricing using linear (AR) and non-linear (LSTM and GAM) methods, 2) compare these methods for Bitcoin price estimation using autoregressive and exogenous variables, and 3) forecast and simulate Bitcoin pricing using optimized models.

ARMAX-GARCH Autoregressive Moving Average with Exogenous variables (ARMAX) models are linear models that can be used to estimate the conditional mean of the process based on past data points, which unfortunately, assumes the conditional variance in the past is constant. ARMAX models alone do not account for clustered volatility behaviors in time series. Fortunately, Generalized Autoregressive with Conditional Heteroscedasticity (GARCH) models allow us to correct the heavy tails and model the randomly varying volatility in our log return. The full ARMAX-GARCH model to estimate the current valuation y_i and concurrent variance σ_i^2 is below.

$$y_i = \mu + \Sigma_{j=1}^{m}\phi_i y_{i-j} + \Sigma_{k=1}^{n}\theta_k\epsilon_{i-k} + \Sigma_{d=1}^{x}\eta_d X_{i-d}$$

$$\sigma_i^2 = \omega + \Sigma_{j=1}^{p}\beta_i\sigma_{i-j}^2 + \Sigma_{k=1}^{q}\alpha_k\epsilon_{i-k}^2$$

For the ARMAX above, μ is the series mean, ϕ is the autoregressive coefficients, θ is the moving average coefficients, and η are the linear fit coefficients of the exogenous variables. For the variance equation, ω is the mean of the variance, β is the autoregressive coefficient of the variance, and α is the moving average coefficients. These ARMAX and GARCH models have been applied to stocks and cryptocurrencies with varying success.

Long Short Term Memory Neural Networks Pure time series forecasting techniques, like the ARMAX-GARCH above, rely on stationarity in univariate time series data. Machine learning and deep learning methods, such as LSTM, allow for higher level feature abstraction to occur regardless of underlying data generating process or distribution, which makes it attractive for financial market forecasting. The LSTM is a type of recurrent neural network, whose LSTM layer architecture allows for networks to retain 'memory' about previous data points to different spans of time (see Figure 1 from Shi Yan for LSTM cell breakdown).

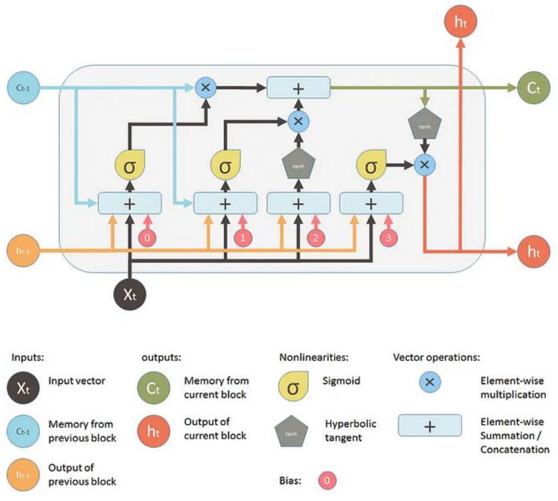


Figure 1: LSTM Cell Architecture. Shi Yan https://blog.mlreview.com/understanding-lstm-and-its-diagrams-37e2f46f1714

Previous researchers (Mudassir, et al. 2020 and Uras, et al. 2020) have attempted LSTMs for modeling Bitcoin; however, these analysis are either univariate or use statistical properties from the time series itself. The effect of exogenous variables is still unknown.

Generalized Additive Models An autoregressive GAM uses the previous time steps features to regress a series of smoothing functions (denoted f below) onto the dependent variable (see Figure 2). In the autoregressive case, one of those features is the previous instance of that dependent variable.

$$g(E[y_i]) = \beta_0 + f_1(x_{1,i-1}) + ... + f_p(x_{p,i-1}) + \epsilon$$

Here g is a link function that allows us to fit any distribution to the response variable, not just Gaussian. β is the typical intercept or mean of the series, while x represents the features for regression. This includes the autoregressive component of the response variable in our case. These smoothing functions f are fit in a model using maximum likelihood estimation based on the number of 'splines' we want to use and the regularization strength (L2 regularization). This method is less explored in financial forecasting but is slowly gaining traction due to it's ability to uncover possibly non-linear relationships between variables.

Methods

This section will describe the data wrangling, feature engineering and modeling process for evaluating the ARMAX-GARCH, LSTM, and GAM techniques for estimating future Bitcoin prices. All data wrangling and analysis was performed in Python 3.10, except for the ARMAX-GARCH modeling, which was performed in R. The proposed methodology is a 1-step ahead prediction using both autoregressive (previous day's log-return price) and exogenous variables (described below).

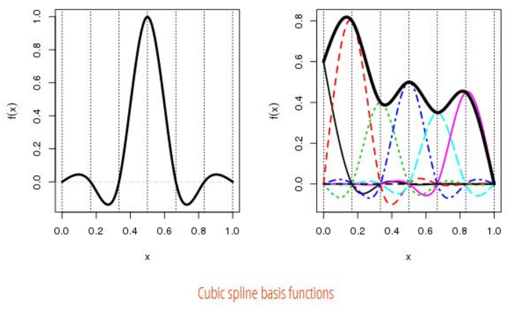


Figure 2: Graphic explaining cubic spline fitting for time series characterization. Gavin Simpson https://fromthebottomoftheheap.net

Data

The data set that will be used for this project comes from the Monash Time Series Data Repository found on Kaggle. The data set consists of 4,581 daily closing prices of Bitcoin from July 2010 to July 2021, as well as 17 various other variables describing other market aspects of Bitcoin. The Bitcoin closing price time series can be seen Figure 3. along with the accompanying histogram. As can be seen there is

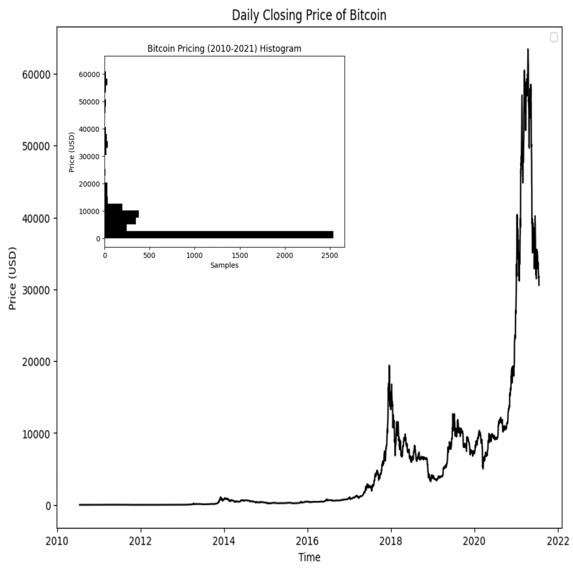


Figure 3: Daily Closing Price for Bitcoin. INSET: Histogram of daily closing prices (horizontal)

an extending period of low volatility up until 2017 when there is a large spike in price along with increased volatility. The price exhibits, and the Augmented Dickey Fuller test confirmed, non-stationarity in trend and variance. The distribution also follows an exponential distribution. Thus, prior to modeling we will have to transform the data. Taking the log return (equation below) is a common transformation in financial time series analysis to achieve a zero-mean normal distribution with interpret-able results (Ding 2018, and Sang-Ha Sung, et al 2022). See Figure 4 for transformation results.

$$logReturn_{y_i} = log(\frac{y_i}{y_{i-1}})$$

Prior to feature transformation or training, the last 7 days

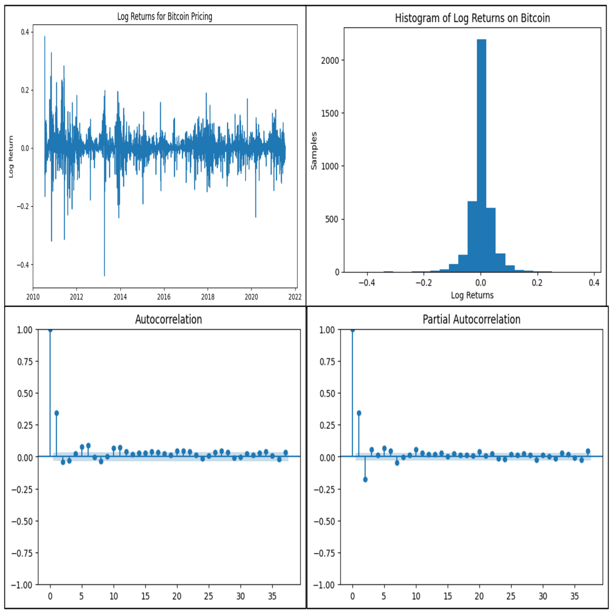


Figure 4: Bitcoin price log return transformation. UPPER LEFT: Time Series, UPPER RIGHT: Histogram, LOWER LEFT: ACF plot, LOWER RIGHT: PACF plot.

of the dataset were held out for evaluation of the forecasting technique. The root-mean-square-error (RMSE) was used as the performance metric.

Exploratory Data Analysis and Feature Engineering

The original dataset contained 17 exogenous variables and the corresponding correlation matrix can be seen in Figure 5. Multiple co-linearity and redundant features can be seen in the correlation matrix. To reduce model complexity, feature reduction was performed using Mutual Information Scoring between each exogenous variable and log-return price. The top 3 features, Google Trends, Difficulty, and Active Addresses, were chosen for further model fitting as they represent different social media sentiment, economical incentive, and environmental metrics, respectively. These 3 features have large differences in ranges between them, which makes training models with gradient descent, like neural networks, very inefficient. Thus, we min-max transformed each

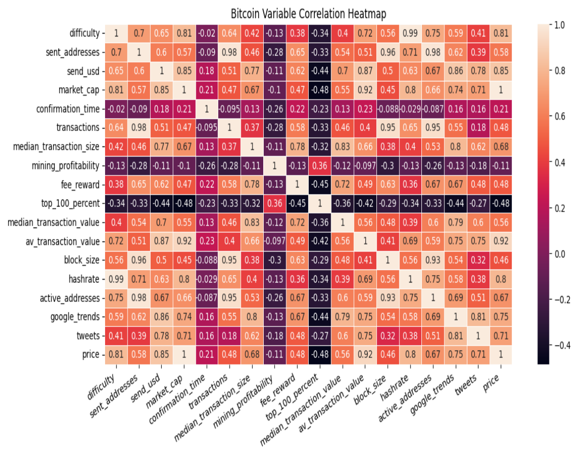
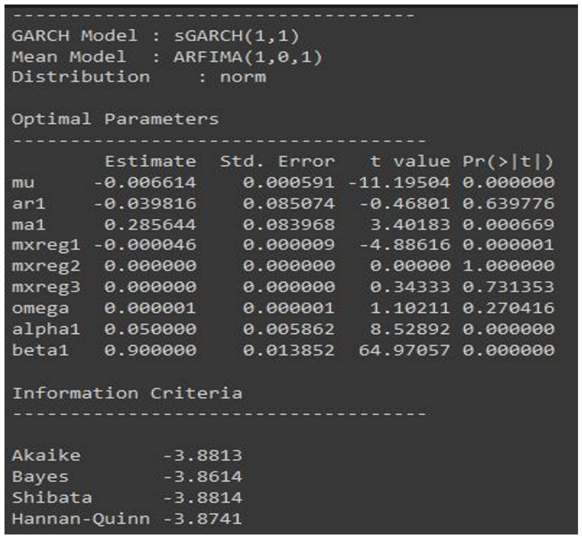


Figure 5: Correlation matrix and heatmap between Bitcoin price and 17 exogenous variables

features to scale from 0 to 1, which preserves the distribution of the feature but allows for faster training, similar to Mudassir, et al. (2020). Ultimately, due to missing values for one or more of these variables, our dataset was reduced to 2,675 samples.

ARMAX-GARCH

The 'rugarch' R package (Galanos 2023) was used to simultaneously fit the ARMAX and GARCH models to the data. Models with varying orders for ARMAX and GARCH were iteratively fit and the model with the lowest AIC used for testing Bitcoin forecasting accuracy. In line with expectations from the ACF and PACF earlier, the ARMAX(1, 1) GARCH(1, 1) model best described the data (model output in Figure 6). The resultant conditional volatility graph is shown in Figure 7. As can be seen in Figure 6, the AR(1),



```
-----------------------------------
GARCH Model  : sGARCH(1,1)
Mean Model   : ARFIMA(1,0,1)
Distribution    : norm

Optimal Parameters
------------------------------------
        Estimate  Std. Error   t value Pr(>|t|)
mu     -0.006614    0.000591 -11.19504 0.000000
ar1    -0.039816    0.085074  -0.46801 0.639776
ma1     0.285644    0.083968   3.40183 0.000669
mxreg1 -0.000046    0.000009  -4.88616 0.000001
mxreg2  0.000000    0.000000   0.00000 1.000000
mxreg3  0.000000    0.000000   0.34333 0.731353
omega   0.000001    0.000001   1.10211 0.270416
alpha1  0.050000    0.005862   8.52892 0.000000
beta1   0.900000    0.013852  64.97057 0.000000

Information Criteria
------------------------------------

Akaike       -3.8813
Bayes        -3.8614
Shibata      -3.8814
Hannan-Quinn -3.8741
```

Figure 6: Output of ARMAX-GARCH fitting from rugarch package. Note the significance of each parameter.

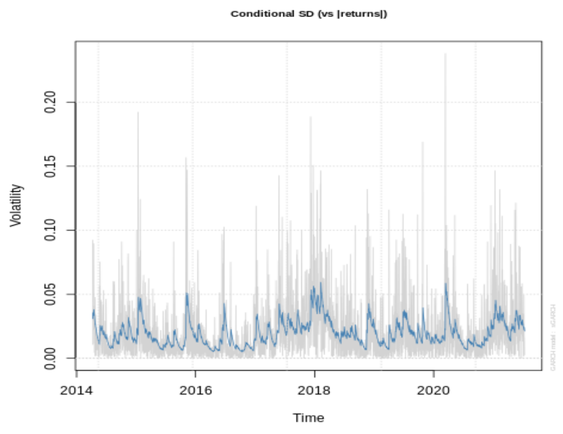


Figure 7: ARMAX-GARCH Conditional Variance for Bitcoin Log Return

Difficulty, Active Addresses, and ω were found to be not significant to log return predictions. β and α were found to be significant, which indicates that the previous days' variances and noise do effect price prediction to some degree. This may lend credence to the theory that Bitcoin pricing is a Random Walk process, where each instance is just the previous day's price plus high variance error. Test set RMSE for the ARMAX-GARCH is reported in the results section in Table 1.

Long-Short Term Memory Neural Network

Long-Short Term Memory neural networks were trained using Pytorch with varying hyperparameters to determine the best architecture for forecasting BTC. The training test set was further broken down into training and validation sets. The architecture and hyperparameters (listed below) with the lowest validation set error was further analyzed with the testing set. The ADAM optimizer with a RMSE loss function was used with a learning rate of 0.01.

- Number of LSTM layers: [1, 2]
- Embedding Features: [8, 16, 32]
- Window Size: [1, 7, 30]
- Activation Function: [relu, tanh]

A batch size of 64 was used over 200 epochs. Early stopping was implemented if there was no improvement or an increase in error in the validation set. Interestingly, the best performing LSTM was 1-layer with a 1-day window size and 8 embedded features and tanh activation. It seems that this is also the network architecture with the lowest number of trainable parameters. Thus, it is hypothesized that there is just not enough data to support training a deep network without highly unstable results. The training curve in Figure 8, shows that both the training and validation curves are highly variable and there is no trend, indicating the network isn't learning. Test set RMSE for the LSTM is reported in the results section in Table 1. Of note, a 1-layer LSTM with a window size of 1 can also be view as a single layer perceptron. As such, a single layer perceptron with 8 hidden

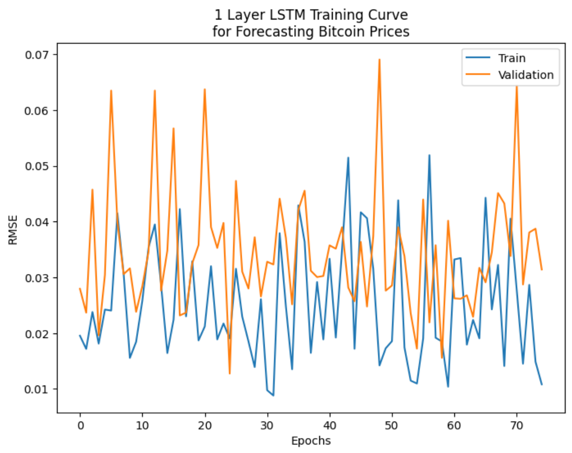


Figure 8: Training and Validation Curve for LSTM

nodes and a tanh activation function was run as well to see if we could get comparable results. Unfortunately, it yielded worse RSME on the testing set, as also seen in Table 1.

Generalized Additive Model

The Python package pyGAM (Servén and Brummitt 2018) was used to fit and tune hyperparameters for the GAM. The two primary hyperparameters to tune is the number of spline functions to fit for each variable and the L2 regularization strength for each feature. This regularization controls the 'wiggliness' of the fitted splines. Similar to the ARMAX-GARCH, the fit with the lowest AIC was chosen as the candidate model for further analysis. After finding the best

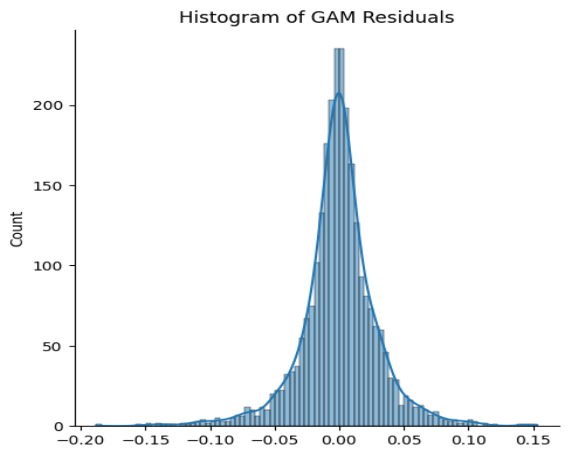


Figure 9: Histogram and PDF of Residuals from GAM fit

model, the residuals were examined to determine the robustness of the fit. Similar to ordinary linear regressions, you want your residuals to be uncorrelated and following a normal distribution. As can be see in Figure 9, our residuals do follow a normal distribution. Not shown is the ACF and PACF of the residuals; however, no autocorrelations were found. Our model is robust. In Figure 10, we visualize the

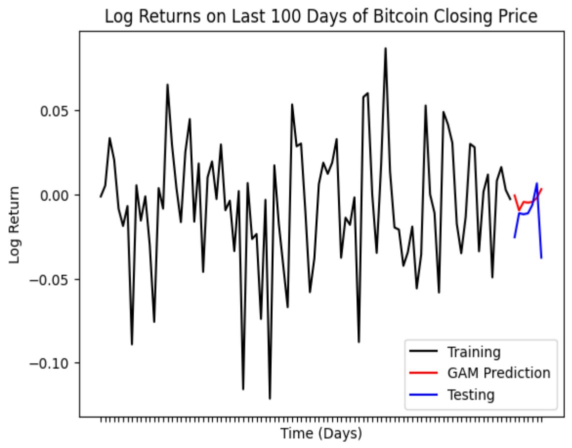


Figure 10: Bitcoin price forecasting using GAM model. Only last 100 days is shown for clarity.

prediction accuracy of the GAM with the training and testing data sets using just the last 100 days of Bitcoin closing price. However, one of the primary benefits of fitting GAMs is the ability to investigate non-linear relationships between the features and the dependent variable. As such, Partial Dependency Plots were made for each feature (Figure 11). The X axis denotes the value of the feature (after the min-max normalization). What is important to note, is how the Y axis changes as the values of the features increase and decrease. For instance, from the far left panel showing Google Trends, we see that the number of Google searches has a constant, and little, effect on Bitcoin pricing. That is until we start getting above the 50% percentile and we see Google trends start to have a large positive effect. The opposite can be seen in the Difficulty feature in the panel next to it. This signifies that Bitcoin price goes down when mining becomes too difficult (and profitability goes down). Active Addresses (3rd panel) shows an almost linear relationship; while the Log Return panel is highly chaotic. Remember that this Log Return feature is the Log Return from the previous day. Test set RMSE for the GAM is reported in the results section in Table 1.

Results

All performance metrics for all models are included in Table 1.

Discussion

The purpose of this analysis was to understand and model the non-linear effects of different Bitcoin characteristics for forecasting future closing prices on the asset. Consequently, the secondary objective was to compare different linear and non-linear methods for forecasting. As can be seen from previous literature and our results here, price volatility continues to preclude accuracy and robust forecasting. Three different techniques or models along the spectrum of linear to non-linear showed similar success (or non-success). The ARMAX-GARCH is a linear estimator using both autoregressive trend and variance to make predictions. This model achieved the lowest RMSE despite its linearity constraints. Our model suggests that the previous days' variances and noise do effect price prediction but that from an ARMAX perspective Bitcoin pricing is just a Random Walk model, where each instance is just the previous day's price plus high variance error. In 2018, Roy, et al. reported a 90.31% RMSE accuracy when fitting an ARIMA; however, it should be noted they only had data up until August 2017, which is before the explosive growth and volatility started occurring in Bitcoin. While Ding (2018) used data up until March 2018 and modeled conditional variance in a GARCH similar to us, the volatility presented an issue. They achieved a Mean Squared Prediction Error (MSPE) of 0.004. ARMAX-GARCH models seem to fail at capturing highly irregular processes, like recent Bitcoin prices.

On the other end of the spectrum, Mudassir et al. tested a stacked neural networks, a support vector machine (SVM), and an LSTM with Bitcoin prices along with other technical indicators, similar to our project. However, these authors took the approach of segmenting the entire time series into 3 different 'segments' to assess model performance during different periods of Bitcoin's life. Consistently, it seemed like their LSTM performed at level or worse than the other methods, especially for the most recent 'regime' when volatility was at its highest. While the authors do not explicitly state the architecture and hyperparameters of their LSTM, the results are in line with the current project. Other machine learning techniques, like the MLP or SVM, may prove to be more beneficial. Uras, et al. (2020) also split Bitcoin price time series data into 'regimes' and applied statistical and machine learning regression for forecasting. Their architecture used one LSTM layer with 64 embedded features and trained with similar batch sizes (32). However, they also saw LSTM be outperformed by traditional linear models, with an AR(1) structure. This might indicate that the most challenging (and interesting) time span for Bitcoin is just a random walk process.

Generalized additive models are a much more niche and complex tool relative to other financial market forecasting tools. Because of how they fit their splines, GAM may not be good for extrapolation on unseen data, which is exactly what forecasting it. Recent tools have been developed to mitigate large errors with Dynamic GAMs (Clark and Wells 2023), but the research is in its infancy. It is positive to see our GAM results were not too far off from the ARMAX-GARCH and LSTM methods. Our results did, however, provide us unique insights into the relationship between Bitcoin pricing and other related variables, as discussed in the methods section over our partial dependency plots.

There are still an almost unlimited number of different ways to approach the same objective outlined in the paper. For instance, taking averages over weeks or months (or any kind of smoothing) could provide improvement; however, this would reduce your available sample size even further. Although the volatility issue might remain. Normality assumptions associated with linear models make things complicated but different Bitcoin price transformations (or non-transformations) could be attempted. Additionally, there are

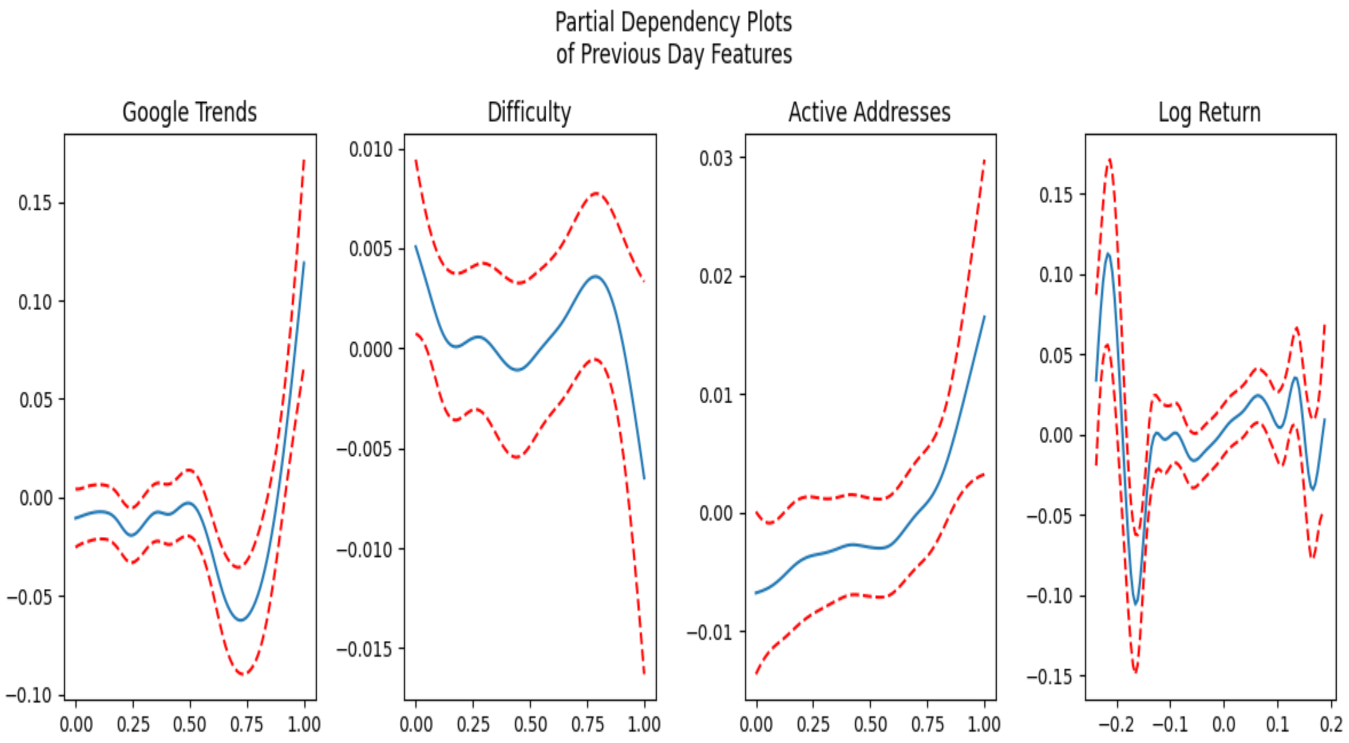


Figure 11: Partial Dependency Plots for Features in GAM

Model	Test RMSE	Real Term Error
ARMA-GARCH	0.014	1.4%
LSTM	0.018	1.8%
MLP	0.022	2.2%
GAM	0.018	1.8%

Table 1: Model evaluation results for testing set RMSE

a number of different normalization techniques for the exogenous variables. As Uras, et al. and Mudassir, et al tried, intelligent partitioning of the data into time spans of similar distributions could help develop threshold based models. A multivariate Gaussian Mixture Model could help clusters of similar time points based on the underlying data generating process. This is an interesting area of research.

Overall, forecasting highly volatile assets, such as Bitcoin, still remains an open challenge for statisticians and machine learning engineers alike. The elucidated facts are that Bitcoin price is a non-stationary time series with periods of irregular variance. Data and tools are available to attempt future price prediction but most still only produce modest accuracies.

Code Repository

The repository can be found at https://github.com/tweedell/CIVE7100_Project which contains the jupyter notebook where all code and data needed for training and analysis resides.

References

Clark, N. J.; and Wells, K. 2023. Dynamic generalised additive models (DGAMs) for forecasting discrete ecological time series. *Methods in Ecology and Evolution*, 14(3): 771–784.

Ding, J. 2018. Time Series Predictive Analysis of Bitcoin with ARMA-GARCH model in Python and R. *University of North Carolina at Chapel Hill Masters Thesis*.

Galanos, A. 2023. *rugarch: Univariate GARCH models.* R package version 1.5-1.

Godahewa, R.; Bergmeir, C.; Webb, G. I.; Hyndman, R. J.; and Montero-Manso, P. 2021. Monash Time Series Forecasting Archive. In *Neural Information Processing Systems Track on Datasets and Benchmarks*.

Mudassir, M.; Bennbaia, S.; Unal, D.; and Hammoudeh, M. 2020. Time-series forecasting of Bitcoin prices using high-dimensional features: a machine learning approach. *Neural Computing and Applications*.

Roy, S.; Nanjiba, S.; and Chakrabarty, A. 2018. Bitcoin

Price Forecasting Using Time Series Analysis. In *2018 21st International Conference of Computer and Information Technology (ICCIT)*, 1–5.

Servén, D.; and Brummitt, C. 2018. pyGAM: Generalized Additive Models in Python.

Sung, S.-H.; Kim, J.-M.; Park, B.-K.; and Kim, S. 2022. A Study on Cryptocurrency Log-Return Price Prediction Using Multivariate Time-Series Model. *Axioms*, 11(9).

Uras, N.; Marchesi, L.; Marchesi, M.; and Tonelli, R. 2020. Forecasting Bitcoin closing price series using linear regression and neural networks models. *PeerJ Comput Sci.*