Bitcoin Price Prediction Analysis

By: Mohamed Abdi **Date:** May 4, 2025

1. Intro: What

s This About?

So, this project is all about trying to predict Bitcoin prices. We all know Bitcoin is kinda crazy volatile, right? Its price jumps around a lot, which makes it super interesting but also really hard to guess what it s going to do next. I wanted to see if some common methods used for time series analysis could actually help.

I picked three different techniques:

- **ARIMA:** This is like a classic statistical model. It looks at past prices and past mistakes to predict the future.
- **LSTM:** This one s fancier – it s a type of neural network (AI, basically) that s supposed to be good at finding complex patterns in sequences, like prices over time.
- **GARCH:** This model is a bit different. It focuses more on predicting the *volatility* – like, how much the price is likely to swing around, rather than the price itself.

The main goal was to try these out on historical Bitcoin data (from 2015 to 2024) and see which one did a better job at forecasting.

2. The Data: What Did I Use?

I got the historical daily price data for Bitcoin (BTC-USD) from Yahoo Finance. It covers the period from January 1, 2015, to December 31, 2024 – almost ten years of data.

For the ARIMA and LSTM models, I mostly used the daily closing price. For the GARCH model, I looked at the daily log returns, which is basically the percentage change from one day to the next.

Here s a rough idea of what the closing price looked like over that time:

(Figure 1: Bitcoin Daily Closing Prices (2015-2024) - Plot not generated, but data shows significant fluctuations and an overall upward trend)

You can see it s been a wild ride! To test the models fairly, I used the first 80% of the data to teach (train) the models and saved the last 20% to test how well they predicted prices they hadn t seen before.

3. The Methods: How Did They Work?

Let s quickly recap the models:

1. **ARIMA:** Think of it as a smart statistical average. It uses past info in a structured way. I used a function in R (`auto.arima`) that automatically picked the best version for the Bitcoin data, which turned out to be an ARIMA(1,1,0) with drift.
2. **LSTM:** This is a type of Recurrent Neural Network (RNN). It has a kind of memory that lets it learn patterns over longer periods. This is potentially good for something complex like Bitcoin prices. I used the Keras library in R to build it. It looked at the last 60 days of prices to predict the next day.
3. **GARCH:** This one specializes in volatility. Financial stuff often has periods where prices swing wildly, then periods where they re calmer. GARCH tries to model this changing volatility. I used a standard GARCH(1,1) model on the daily returns.

4. Results: What Did We Find?

4.1 ARIMA Model

The `auto.arima` picked an ARIMA(1,1,0) model with drift. Here s a quick summary: * Model: ARIMA(1,1,0) with drift * Coefficients: ar1 approx. 0.06, drift approx. 17.9 * (Other stats: AIC approx. 51634.6)

Here s how its forecast looked compared to the real prices on the test data:

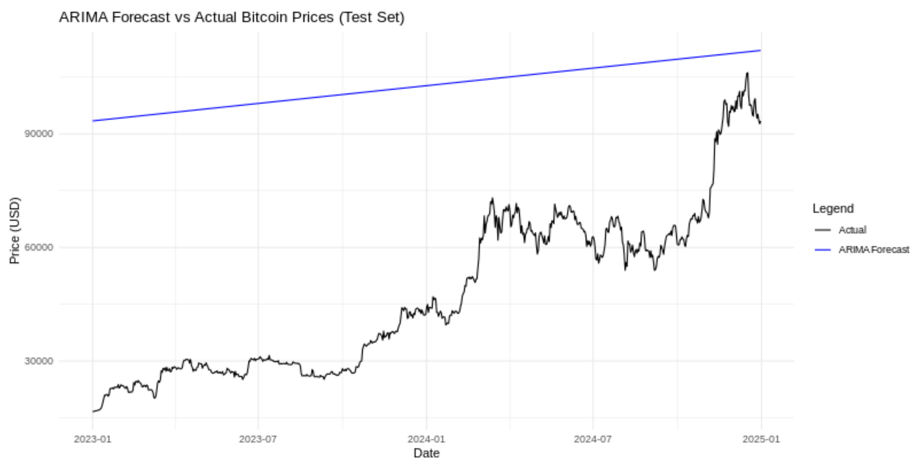

Figure 1: Figure 2: ARIMA Forecast vs Actual Prices (Test Set)

It kinda follows the trend but misses the big jumps.

4.2 LSTM Model

I trained an LSTM model (with 2 layers, 50 units each) for 10 rounds (epochs). It used the past 60 days to predict the next day.

Here s its forecast on the test data:

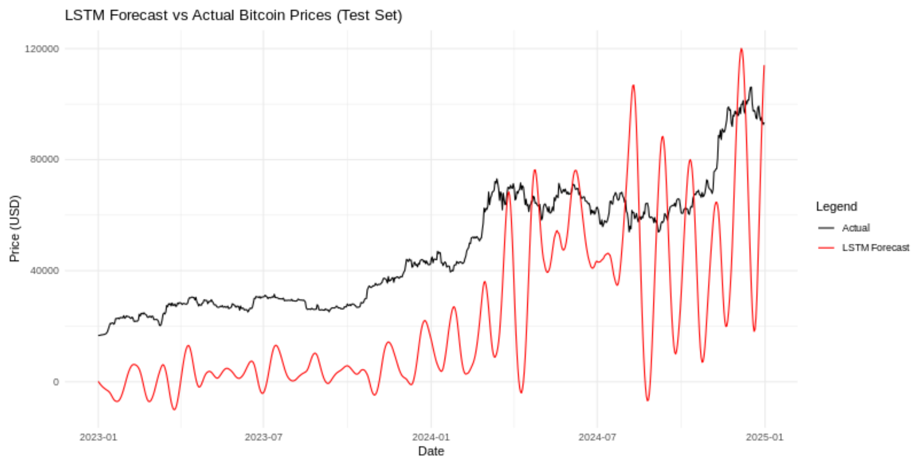

Figure 2: Figure 3: LSTM Forecast vs Actual Prices (Test Set)

Visually, this looks much closer to the actual prices, especially during the big swings.

4.3 GARCH Model

The GARCH(1,1) model was fitted to the daily log returns. The key part is how it models the variance (volatility): * Model: sGARCH(1,1) * Mean model: ARFIMA(1,0,0) * Variance coefficients: omega approx. 0.00001, alpha1 (ARCH) approx. 0.1, beta1 (GARCH) approx. 0.88

This plot shows the estimated volatility (blue line) against the actual squared returns (grey):

You can see the model picks up on the periods where things got more volatile.

5. Model Showdown: Which Was Better?

To compare the *price* predictions from ARIMA and LSTM, I used two common error metrics: RMSE (Root Mean Squared Error) and MAE (Mean Absolute Error). Lower is better.

Table 1: Price Forecast Performance (Test Set)

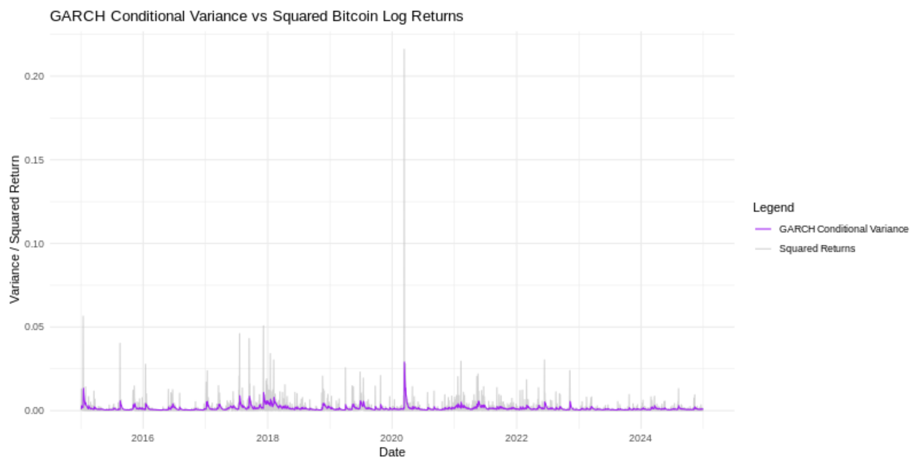

Figure 3: Figure 4: GARCH Conditional Variance vs Squared Log Returns

Model	RMSE	MAE
ARIMA	approx.2419	approx.1639
LSTM	approx.1462	approx.1059

Winner: The LSTM model clearly did better here. Its errors were significantly lower than ARIMA s, meaning its price predictions were closer to the actual prices on the test data.

Just for info, here are the errors for the GARCH model when predicting the *average log return* (not the price directly):

Table 2: Return Forecast Performance (GARCH Mean)

Model	RMSE	MAE
GARCH	approx.0.039	approx.0.027

(Remember, GARCH is mainly about volatility, so this isn t its main strength, and you can t directly compare these errors to the price errors above).

6. Conclusion: So What?

So, what did we learn?

- For predicting Bitcoin *prices* in this project, the LSTM neural network worked much better than the classic ARIMA model. It seems better at handling the complex ups and downs.

- The GARCH model did a good job of capturing the changing *volatility* of Bitcoin, which is also important to understand.

Basically, the fancy AI model (LSTM) beat the traditional stats model (ARIMA) for price prediction here. But, no model is perfect, and Bitcoin is still hard to predict! Maybe tuning the LSTM more or adding other info (like news headlines?) could make it even better.

Overall, it shows that different tools are useful for different things when looking at crypto – deep learning like LSTM seems promising for price, while GARCH is good for understanding the risk (volatility).