Jordan Bouknight
ISTA 421

Forecasting Bitcoin Prices Using a Bayesian Approach

Introduction

The basis of the project was to forecast the price of Bitcoin in order to make reasonable inferences about whether or not it would be a good investment for any particular person. The model implementation was purely based on historical Bitcoin prices sourced from MarketWatch. Additional economic/market factors were not used as parameters in order simplify testing and to keep the model focused on the Bitcoin data.

The advantage in taking a Bayesian approach to time series forecasting is that it is particularly useful in handling issues of overfitting, especially with large data sets. Using a Bayesian method allows a way of combining prior information with observed data within a reasonable theoretical framework. When it comes to market data, Bayesian methods can be especially useful since there is always some degree of uncertainty as to how prices may behave. Bayesian inference sets reasonable bounds on this uncertainty and can better model the "beliefs" investors have.

Bayesian Theory first starts with a simple linear regression. From the raw data, a linear model of the form:

$$y_i = \beta x_i + \alpha$$

Where y_i represents the observations, β is the slope, x_i is the independent variable (in this case time) and α is the intercept. In order to make sure this linear model is truly of best fit to the data, the least-squares method is performed such that:

$$\min S(\alpha, \beta), \text{for } S(\alpha, \beta) = \sum_{i=1}^{n} (y_i - \alpha - \beta x_i)^2$$

In order to minimize this sum, the partial derivatives of α and β are set to zero. The result of taking these derivatives and solving for their values is as follows:

$$\hat{\beta} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

$$\hat{\alpha} = \bar{y} - \hat{\beta}\bar{x}$$

Source: https://towardsdatascience.com/the-first-step-in-bayesian-time-series-linear-regression-89a64b826a7e

Here, $\bar{x}$ and $\bar{y}$ are the mean values for time and price, respectively. Having the values for $\hat{\beta}$ and $\hat{\alpha}$ will result in the least squared mean, thus the line of best fit can be plotted onto the existing data.

The next step is implementing Bayesian statistics in order to actually forecast likely data points beyond the observed data. Bayes theorem is as follows:

$$(P|A) = \frac{P(B|A)P(A)}{P(B)}$$

We can describe Bayes Theorem where $(P|A)$ is the posterior probability, $P(B|A)$ is the likelihood, $P(A)$ is the prior belief, and $P(B)$ is the evidence (sometimes called marginalization). This establishes the foundation of the Bayesian analysis. More specific details as to how these ideas were implemented can be found in the procedure.

Procedure

Data was obtained from MarketWatch and downloaded as a .csv file. The dates were converted to number of days, and the closing price was extracted. This simplified the data so that the closing price could be plotted cleanly against time. Figure 1 shows the raw data plotted in Python.

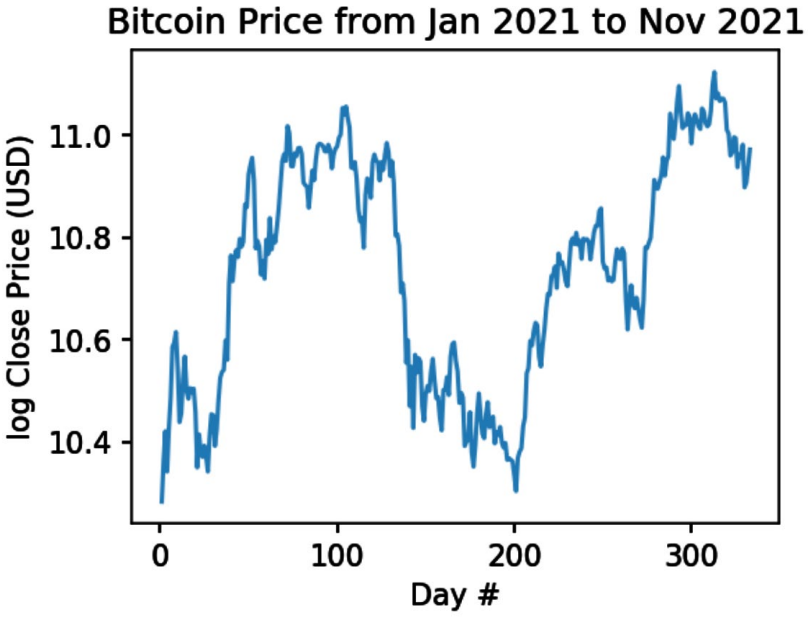

Figure 1. Raw BTC Data

Upon reading in the data, the first step was to perform linear regression on the data set. This was implemented by using the same steps as described in the

introduction. Next, the Bayesian model was implemented using the PyMC3 library. The model would generate a set number of data points given the true values of α and β with added random noise defined as ε. The prior distributions of each of the parameters was set. In this case, the parameters were α, β, and ε. These were set as flat distributions where σ was large, essentially implying that there was a good amount of uncertainty in regard to how the price of Bitcoin would move. The likelihood was then set as such:

$$Y \sim N(\alpha + \beta x, \varepsilon)$$

Each of the parameters were then assigned their own distributions. These distributions were kept generic as to imply that there was not much knowledge about them. The assigned distributions are as follows:

$$\alpha \sim N(0,20)$$
$$\beta \sim N(0,20)$$
$$\varepsilon \sim N(0,5)$$

From this point, the PyMC3 library was utilized in order to determine the priors for each parameter, as well as the likelihood. This was a necessary step in order to determine the predicted values of BTC prices based off of the parameters by sampling. Each of these prior probabilities would be used to visualize the posterior distributions, showing that the Bayesian method did indeed converge on the true values of α, β, and ε. The uncertainty estimation would be plotted against the original data set using the trace values calculated by PyMC3. From this point, the values of α, β, and ε were calculated both by linear regression and Bayesian inference.

Given the values of α, β, and ε the forecasted prices were relatively easy to calculate. The mean values for the prior distributions of each parameter were calculated. The forecasting was modeled by the following:

$$price(t) = (\beta_{mean} * t) + \alpha_{mean} + \varepsilon$$

The model could be extended for any particular length (days). The generated prices were directly a result of the posterior likelihood from previous calculations. The forecast was finally visualized with its own plot.

Results

The values for α, β, and ε were first calculated using linear regression. Table 1 summarizes the outcomes for each of the parameters.

Table 1. Parameter Value Results via Linear Regression

α	10.61645062862417
β	0.0007241982401042338
ε	0.045030936872328134

These values are lower than what would be expected when looking at the raw data, because these are based off of a logarithmic scaling of the data. All the calculations were performed with the same scaling in order to make it easier for certain operations to handle what would have been relatively large numbers. Nonetheless, with the parameter values calculated, the least-squares was plotted against the BTC price data as shown in Figure 2.

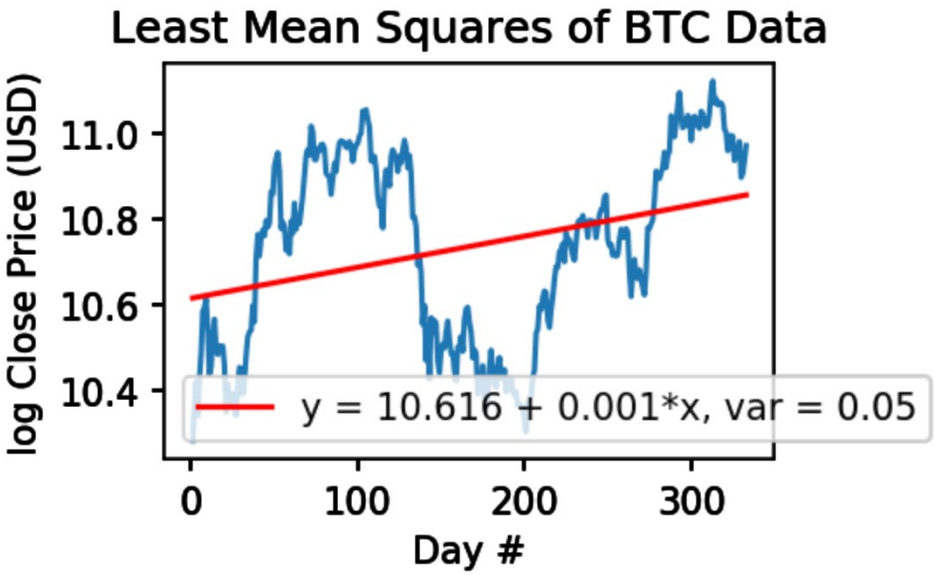

Figure 2. Linear Regression for BTC Data

From this point, the Bayesian model could be implemented. A brief model check was performed using randomly generated data. For the sake of the report, those results are not included, however it can be found commented out in the code. Once the model was validated, it could be applied to the actual BTC data. The priors were calculated using the PyMC3 library. Essentially, each parameter was set to a particular distribution (shown in previous section). Next, the likelihood was calculated using the priors. Figure 3 shows the code for these calculations.

```
with pm.Model() as model:
    # priors
    alpha = pm.Normal('alpha',mu = 0, sigma = 20)
    beta = pm.Normal('beta',mu=0,sigma=20)
    epsilon = pm.Normal('epsilon',5)

    # likelihood
    y_pred = pm.Normal('y_pred', mu = alpha + beta*t,
                       sd = epsilon, observed = y)
    trace_1 = pm.sample(2000, cores = 8)

    idata = az.from_pymc3(trace_1)

    az.plot_trace(idata,var_names = ['alpha','beta','epsilon'],
                  lines=[('alpha',{},alpha_hat),('beta',{},beta_hat),
                         ('epsilon',{},err_var)])
```

Credit: https://towardsdatascience.com/the-first-step-in-bayesian-time-series-linear-regression-89a64b826a7e

Figure 3. Prior and Likelihood Distributions using PyMC3

The results were visualized, showing convergence for the values of α and β; however, ε did not show the same behavior. This will be discussed in the Evaluation section. Figure 4 shows the posterior distributions for each parameter on the left, and the individual sample values for each time step on the right (these are the traces).

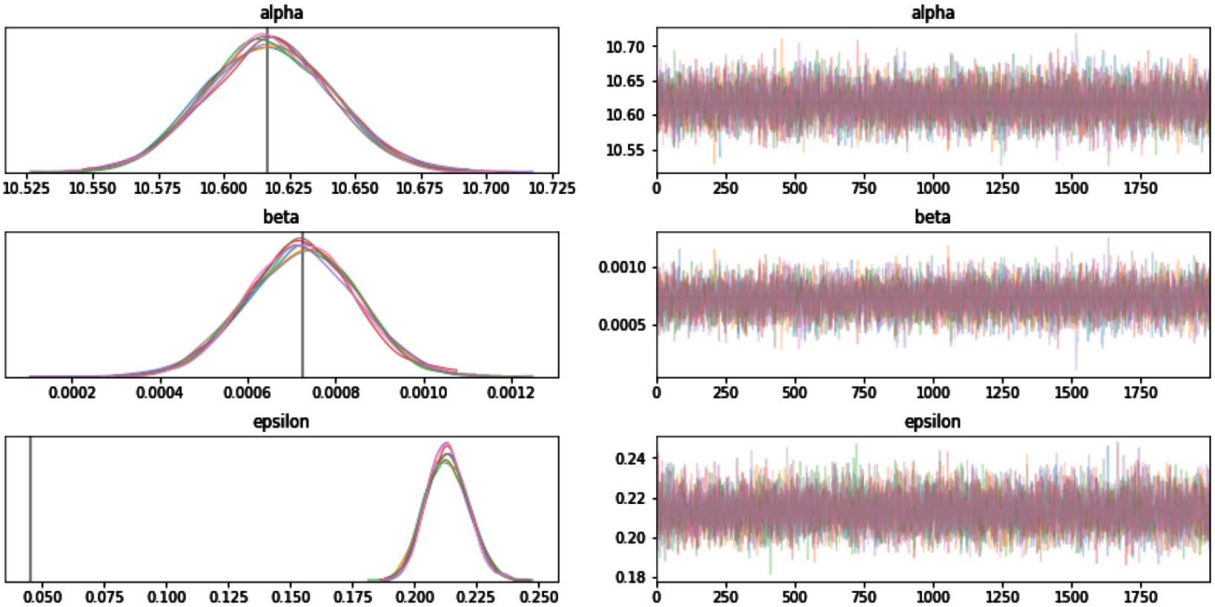

Figure 4. Traces for Parameters

This operation led to the determination of the parameter values via the Bayesian method. The uncertainty in each estimate was plotted against the best fit line found by convergence to show the bounds of the Bayesian model in Figure 5.

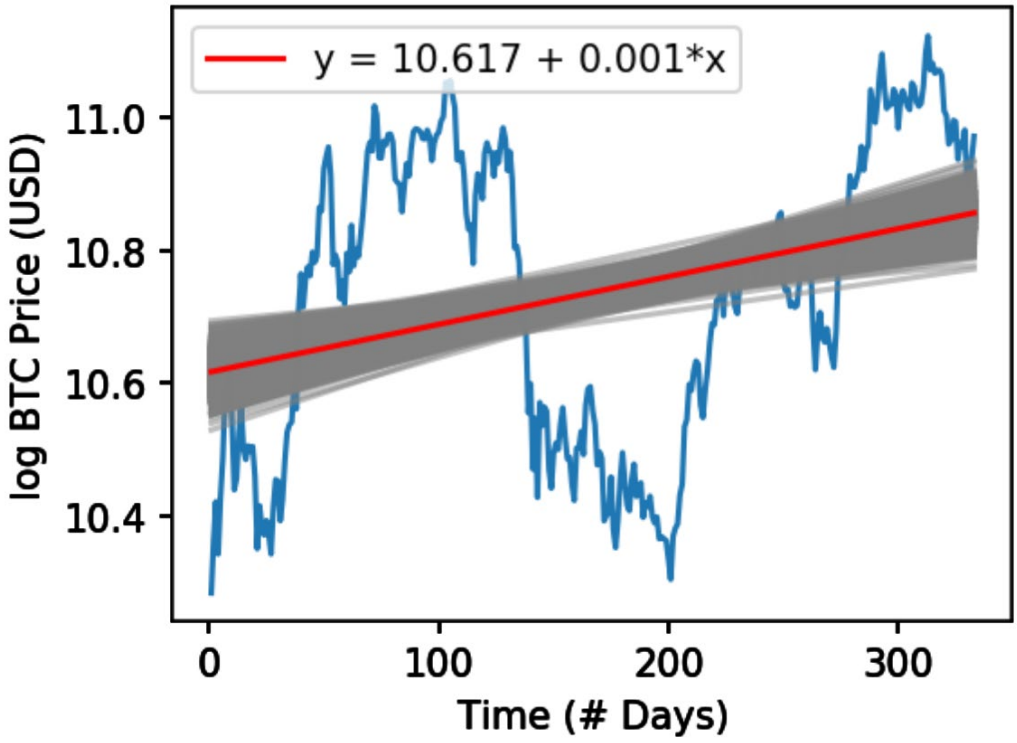

Figure 5. Bayesian Model Visualized

The numerical results are summarized in Table 2, showing the comparison between the two methods.

Table 2. Summary of Results for Each Method

	Bayesian	**Linear Regression**
α	10.616908263718475	10.61645062862417
β	0.0007213420035448258	0.0007241982401042338
ε	0.2132003428908667	0.045030936872328134

From this point, all the information needed to forecast future prices was obtained. The mean trace was calculated by taking the mean values for α and β based off of the Bayesian model. The length of the forecast is a set value, for this case 30 days. Figure 6 shows the 30-day forecast with the error bounds included.

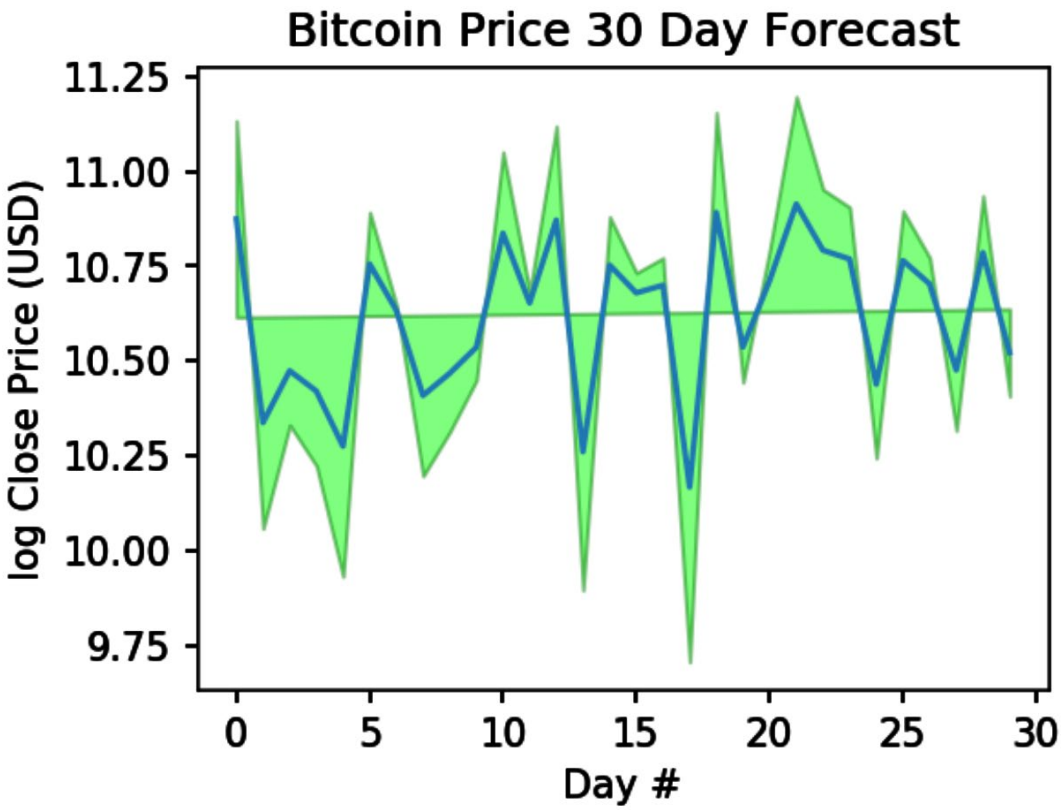

Figure 6. Forecasted BTC Price via Bayesian Inference

Evaluation

The first method implemented was linear regression. From looking at the values of the calculated parameters, these values were relatively easy to calculate. There were not any points of concern and from the plot, the line of best fit seems reasonable with the data. The logarithmic scaling obviously lowers the calculated values, but it was useful for later operations.

There were some issues initially when using the PyMC3 library for the Bayesian inference portion. The first issue was that the kernel in JupyterNotebooks would disconnect upon running the prior and likelihood calculations. The first step to fix this problem was updating to the most recent version of Anaconda. This kept the kernel from disconnecting; however, the next issue was a runtime error when executing the third chain. It was at this point that the logarithmic scale became useful. The scaling of the numbers allowed the operation to run without failure and resulted in the values for α, β, and ε. The function ran 2000 samples for each parameter with 8 running cores. The sample number could easily be increased to 20,000, but this was not measured for the interest of time. A 2000 sample run would take about 2-3 minutes in all.

When referencing Table 2, there is one clear discrepancy. The values for α and β are nearly exactly the same, but the ε value between the two methods is significantly different. This is also shown in Figure 4, where none of the posterior distributions converge on the value that was expected. This could be an issue with the parameterization of ε, or a simple mistake somewhere rooted in the code. In the end, it did not seem to greatly effect the model, but it is definitely still worth mentioning for the purpose of being thorough, as the estimates still be used in the forecast later. The visualization of the bounds of the estimates in Figure 5 was reasonable, and obviously the line of best fit was nearly the same as in the linear regression model.

The forecasting implementation did not pose any problems, but it must be kept in mind that the Bayesian ε estimates were carried over when calculating the mean ε. The forecast values are within reason and can be compared to the actual BTC prices recorded after November 2021. As of Dec 11, 2021, BTC is sitting at a price of $48,847 (12 days outside of the data set used for modeling). Looking at the 30-day forecast, the model shows a log price of about 10.7 ($44,355) on the 12^{th} day. The absolute error between the forecasted $44,355 and true $48,847 was 9.2%. Depending on your appetite risk, this error may or may not be alarming, but for the means of the model, 9.2% is reasonable. The forecast may not exactly match the actual chart for BTC, but this is not unexpected as BTC has high volatility and makes big price swings. The forecast still models these price swings relatively well (this translates to high noise/ large variance).

As mentioned earlier, there were not any additional market indicators incorporated into the model in order to maintain a level of simplicity. However, that does not mean that the model cannot be extended to incorporate these factors as new parameters. Trading patterns and technical analysis definitely can influence the pricing models of any given security whether it be cryptocurrency or stocks. The next steps would be to extend the Bayesian model to accept multivariate parameters in order to improve the forecast. Outside of the data science aspect of the model, it would be useful to automate trades based off of the forecast in order to execute winning strategies. For the time being, this is beyond the scope of the project (but not impossible).

References

[1] Bayesian Time Series - Linear Regression | Towards Data Science

[2] Probabilistic Programming and Bayesian Inference for Time Series Analysis and Forecasting in Python | by Yuefeng Zhang, PhD | Towards Data Science

[3] A Bayesian Approach to Time Series Forecasting | by Daniel Foley | Towards Data Science

[4] Bayesian Linear Regression in Python: Using Machine Learning to Predict Student Grades Part 2 | by Will Koehrsen | Towards Data Science

[5] PyMC3 Documentation — PyMC3 3.11.4 documentation

[6] Johansson, Robert. “Chapter 16 - Bayesian Statistics.” Numerical Python: A Practical Techniques Approach for Industry, Apress, New York (N.Y.), 2015.

[7] Rogers, Simon, and Mark Girolami. A First Course in Machine Learning. Chapman & Hall/CRC, 2020.