Using Regression to analyze the relationship between historical Bitcoin prices and posts on /R/Bitcoin

Spencer Cameron, Braxton Booth, Jeremy Cruz, Alex Tingey

ABSTRACT

The goal of our project is to analyze the performance of utilizing different regression techniques in predicting the price of Bitcoin by using features of posts made on the popular internet forums /R/Bitcoin, in addition to historic pricing data, as input. We aim to use our best selected regression method in an attempt to predict the movements of the security before they occur, enabling a predictive trading strategy.

KEYWORDS

Linear Regression, Ridge Regression, Neural Networks, Support Vector Regression, Data-set creation, Coefficient of Determination, Bitcoin

ACM Reference Format:
Spencer Cameron, Braxton Booth, Jeremy Cruz, Alex Tingey. 2021. Using Regression to analyze the relationship between historical Bitcoin prices and posts on /R/Bitcoin. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

We explore the techniques of linear regression, ridge regression, lasso regression, SVR regression, and regression using a neural network. We ran these methods of regression on a data-set suitable for cross validation, utilizing the sum of residuals squared between our label value and predicted value to determine the best model.

2 MOTIVATION

The past few years have seen a major surge in retail investing. This is largely because millions of people throughout the world now have access to the market with trading platforms such as Robinhood or TD Ameritrade becoming mainstream. As trading rose so too did activity on trading forums, where users discuss the performance of securities, their trading strategies, and current trends in markets. One extremely popular website for such forums is Reddit where millions of people get their trading information. /R/Bitcoin on Reddit is one such forum with nearly 3 million members currently. Recent volatility in Bitcoin as well as many stocks can be largely attributed to discussions taking place on these online forums.

3 COEFFICIENT OF DETERMINATION

R^2, which ranges from 0 to 1, is a measure of how well a regression model fits its target y data. It can be inferred that the closer a model's R^2 is to 1 the better it performs, with 1 representing a model that is able to perfectly predict our target y values. Throughout all of our experiments, our target y was selected as Yahoo Finance's listed closing price for BTC on the given day. R^2 was chosen specifically as the metric to compare each approach as it is very easy to interpret and compute.

$$R^2 = 1 - \frac{SS_{\text{res}}}{SS_{\text{tot}}} \quad [1]$$

4 DATA COLLECTION PIPELINE

We have created a Python script which uses the RESTful Pushshift API to collect our data from Reddit posts and comments. This entailed setting up proxies and using multithreading to optimize our API requests. We then had to process the JSON request results for the meaningful information. This data was then written to multiple CSV files which were processed and cleaned to form a single CSV file containing the entirety of our collected data thus far. This CSV is then paired with historical Bitcoin data, retrieved by utilizing the Yahoo finance API. This Bitcoin data takes the typical shape of (Open, Close, Low, High, Volume) for a given date.

4.1 Sentiment Analysis

To perform sentiment analysis on the Reddit data we used the FinBERT transformer model, which outputs a probability distribution over whether the text data should be classified as positive, negative, or neutral.[2] Typically the maximum probability is used to classify the text, however we chose to keep all three values and use them within the dataset to preserve as much of the sentiment information as possible in the underlying text.

4.2 Normalization

Before training the models X is normalized using SKlearn's MinMaxScaler, which scales the input data to be between zero and 1 [3]. This is done because the features provided by the FinBERT are probabilities, and thus always between 0 and 1. When these probabilities are paired with our other metrics, which are often very large real numbers, some of the models had a hard time achieving good results.
By normalizing the data we map all of the features between 0 and 1, making all of the features have the same possible max and min values. This helps the models fit to the data better while preserving the relationships of our feature vectors on different days.

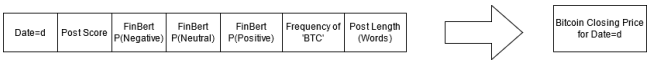
4.3 Number of Samples

To increase the accuracy of our features we sampled 100 posts, and 100 comments per post, from the API for a total of 1,596 days worth of data. This means $O(15,950,000)$ posts and comments were sampled and processed (as not all posts have 100 comments). These samples were then averaged for the day, as this corrects for measurement error (Gaussian Noise) present in any data collection task.

Conference'17, July 2017, Washington, DC, USA
2021. ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

4.4 Regression Data Experiments

4.4.1 Initial Approach. In our initial approach we attempted to build a regression model from our Reddit collected features for a target day onto BTC's corresponding closing price. The motivation behind this strategy was that it may be possible to sample the sentiment of posts made early in the day to determine what price Bitcoin would be at closing, making it possible to tell if Bitcoin is going to increase in price (and therefore make a profitable trade).



4.4.2 Look-Ahead Regression. Our initial approach was flawed in two crucial ways. First, the resolution (measured as number of samples per day) of the data is too low to implement a trading strategy involving tracking hourly prices of bitcoin. Second, we were asking the models to perform an incredibly difficult task by providing them with solely with information likely to only be partially correlated to the exact price of Bitcoin.
To address this we implemented two changes:

(1) We set our target price for the regression (y) to a future day's BTC closing price. This was accomplished by simply removing t (referred to as the look-ahead) days from the first portion of y and t days from the final portion of X. Since X and y are sorted by date in ascending order this has the effect of mapping observed X values for day d onto the y (BTC close) of day $d + t$.
(2) We added BTC price information to X for day d. This proved to be very valuable, as the average R^2 of the models trained on this data decreased only slightly when t increased from 1 to 10 (from 0.99 to 0.95). This suggested the model would generalize well to larger, and therefore more useful, values of t.

5 REGRESSION EXPERIMENTS

5.1 Train-Test-Tune Split

For each regression model we used a train/test split of 0.3, meaning 30% of our data was not used to train the model, but instead to evaluate how the model performs on data it has not observed before. This means roughly 478 days of the total 1,596 are used exclusively to test the given model.
This split was performed using SKlearn's train_test_split functionality, which also randomizes the order of (X, y) pairs such that it is unlikely two data-sets created using this method are unlikely to be exactly the same. This allows to evaluate the models, specifically obtain an R^2, using data that they have not been trained on.

5.1.1 Hyper-Parameter Tuning. Many of the regression techniques we used have hyper-parameters which can impact the performance of the model. For Lasso and Ridge regression this parameter is *alpha*. For SVR regression there are two parameters, C and ϵ. And finally for the Keras Neural Network the Hyper-parameters take the form of depth and width of layers within the network, as it is a simple feed-forward ANN.
Each of these models was tuned using a separate train-test split dataset to avoid tuning the model in such a way that it cannot generalize to new, yet unseen data[1]. Tuning was performed on the SKLearn models using Sklearn.model_selection.GridSearchCV, and on the Neural Network using the Kerastuner package. After tuning was completed the hyper-parameters of the top 3 models with the best R^2 scores were averaged and used to create the tuned model which was then evaluated on a different train-test split of the data.

5.2 Final Regression Dataset Used

For each of the following models we used the look-ahead regression technique outlined in section 4.4.2 to predict the next day's $(t + 1)$ closing Bitcoin price based on BTC price information and our collected natural language features from Reddit.

5.3 Notes on Interpreting Graphical Results

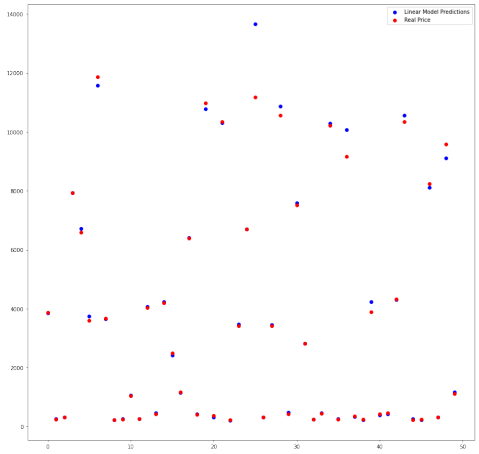
To create a graph representing how well the model fits to the data we used a scatter plot showing the real closing price of BTC for day $d + t$ (y) and the model's predicted price $(\hat{y})$. This graph is composed of 50 points from the validation set to make it easier to interpret. Each model was evaluated against the entire validation set, but we chose to plot only 50 y and $\hat{y}$ pairs to make the graph easier to interpret.

5.4 Basic Linear Regression

We decided to use Linear Regression as a starting point. We felt that since it is the most basic regression technique, it will provide us a good baseline to gauge our ability to build an accurate model.

5.4.1 The Experiment. For this experiment, we made use of the sklearn.linear_model library. We had inferred that linear regression would most likely be outperformed by other regression models as our data might not follow a direct linear correlation to the actual Bitcoin prices. Throughout all experiments the R^2 produced by linear regression was always the lowest of all of our models. However, particularly after adding BTC price data to X, the obtained R^2 often differed by less than 1% between all models; implying that the linear regression model is not necessarily worse than others.

5.4.2 Linear Regression Results.

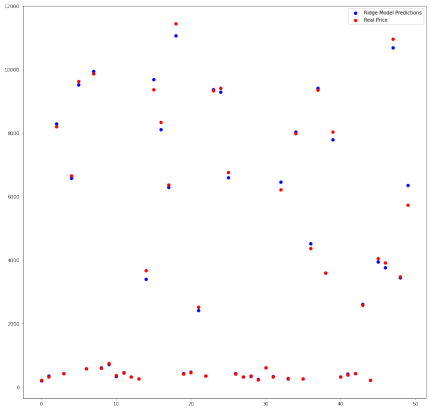


The graph above is a plot of the actual Bitcoin prices (red) vs. our predicted Bitcoin prices (blue). This model obtained an R^2 of 0.9792046394668421, signalling a very good fit.

5.5 Lasso And Ridge Regression

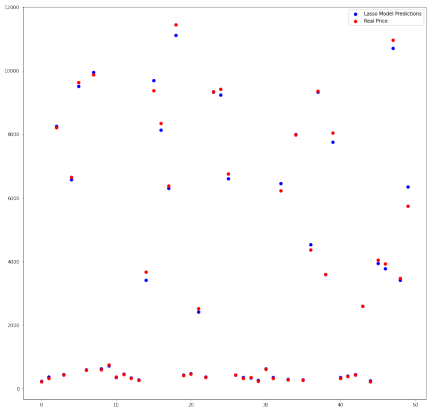
We then analyzed the performance of Lasso and Ridge Regression, as implemented by the sklearn.linear_model package. We inferred that, because the two algorithms share the aim of achieving a more sparse set of regression coefficients, differences within observed R^2 could tell us something about our data. For Ridge regression $alpha$ represents regularization strength[4]. For Lasso regression $alpha$ represents the $L1$ term used to solve the objective equation for Lasso Regression [5].

5.5.1 *Ridge Regression Results.*



This model achieved an R^2 of 0.9810381750577002. The hyper-parameter $alpha$ = 0.1001001 was selected, as it was the average of the 3 models with the highest achieved R^2 during tuning.

5.5.2 *Lasso Regression Results.*



This model achieved an R^2 of 0.9804051783639447. The hyper-parameter $alpha$ = 2.102102102102102167 was selected as it is the average of the 3 models with the highest achieved R^2 during tuning.

5.6 SVR Regression

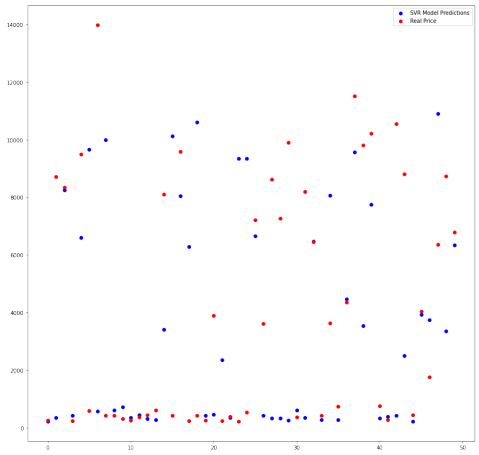
SVR Regression was selected as it is a non-linear regression technique which tends to generalize well by allowing for some error. Additionally, SVR enables us to use non-linear kernels (RBF kernel) which allows us to perform non-linear regression.

5.6.1 *Experiment Details.*

For this experiment, we made use of the sklearn.svm.SVR library. The C hyper-parameter used in constructing the SVR model is used as a penalty parameter which helps to minimize error present within the model [6]. The hyper-parameter ϵ represents the level of error acceptable to the SVR [6]. As part of our experiment, we spent a good amount of time tuning both hyper-parameters.

5.6.2 *SVR Results.*

The tuned hyper-parameters C = 100012.32 and ϵ = 0.457 achieved an R^2 score of 0.97664 for this model.



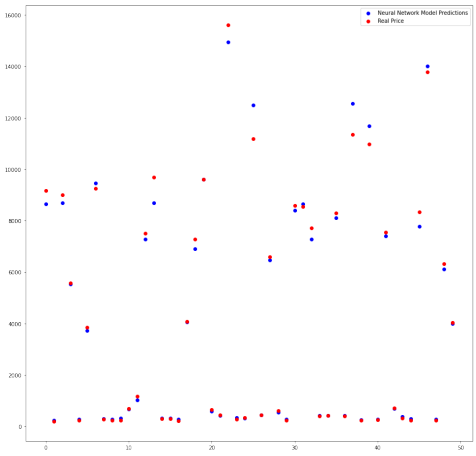
6 NEURAL NETWORK REGRESSION

6.1 The Experiment:

For this experiment we created a model using the Keras framework. The model is a generic feed-forward ANN which has been tuned by the Kerastuner package to produce a unique shaped network which optimizes the MSE (Mean Squared Error of Residuals) loss function used to train the network on the regression task.

6.2 Results:

While tuning, the Hyperband tuner has the capability to determine the number of neurons on a given layer and the number of layers above the output layer, which is a single Neuron with the ReLu activation function. The networks that outperformed others often had latent layers with more than 1,500 neurons inter-weaved with layers containing as little as 22 neurons. This suggests a combination of dimensionality expansion (as X is made of only 10 features) and dimensionality reduction is used by the ANN to produce its results.



This model achieved an R^2 of 0.9942782229043742. The final hyperparameters are as follows:

Layer 1	Layer 2	Layer 3	Layer 4	Layer 5	Layer 6	Layer 7	Layer 8	Layer 9	Layer 10
568	1661	22	203	1566	39	1054	799	1172	1

Each layer utilizes the ReLu activation function. The selected learning rate for a maximum of 150 epochs of training was 0.2.

7 BACKTESTING

7.1 Motivation

While R^2 is useful when building regression models, our true goal was to determine if we could predict the future price of BTC accurately enough to guide a profitable trading strategy. Backtesting assesses the performance of a trading strategy by simulating it on historical data.[7] By creating a trading strategy that utilizes our bitcoin predictions for future days and backtesting it, we can gain an understanding of how useful our models are in making profitable trades.

7.2 Methodology

Using the Python Backtrader library we created a trading strategy with two tuned SVR models, chosen for its combination of accuracy and speed in training, trained on the prior 730 days of data, using a five and ten day lookahead ($t = 10$ and $t = 20$). These two models are then used to create an SMA crossover strategy[8] based on the $\hat{y}$ of both models. The trading strategy is as follows:

(1) Backtrader simulates Bitcoin prices over a target span of days, each time requesting the two SVR models for the future price. The simulation gives the model $10,000 at the very beginning to allow the model to enter the market.
(2) If the $t = 10$ lookahead crosses the $t = 20$ lookahead value with a positive slope, we issue a buy order for as much BTC as can be purchased with the un-invested cash on hand.
(3) If the $t = 10$ lookahead crosses the $t = 20$ lookahead value with a negative slope we issue an order to sell at least one of the currently owned BTC.

A cross is defined as an observed intersection of the lines defined by two points: the price of BTC on day d, and the price on day $d + t_i$. In our strategy we have two values for t, $t_1 = 10$ and $t_2 = 20$.

7.3 Backtesting Results

We back-tested the above strategy on historical data from one-hundred times to determine if the trading strategy can reliably generate a profit. We measure the percent increase in portfolio value as:

$$\frac{EndingPortfolioValue - 10,000}{10,000}$$

The average percent change over one-hundred trials was 0.4877274. This means that, on average, our trading strategy produces a fairly decent profit.

8 MODEL COEFFICIENTS EXPERIMENT

It is common for regression models predicting future prices based on previous days prices to converge to a trivial solution which only slightly manipulates the input price of the previous day to make a prediction. These models often fail to generalize for larger look-ahead values, making them less useful when attempting to build a profitable trading strategy. We can analyze if our model is converging to this trivial solution by examining the final weights for *alpha* obtained by linear regression.

8.1 One Day Lookahead Model

The weights in *alpha* and their associated features are as follows:

Post Length	Post Score	'BTC'	P(Pos)	P(Neut)	P(Neg)	Open	High	Low	Volume
-1189.10486	621.2882	309.401	-102.7697	-351.15801	2816.4284	-8357.7946	14203.8543	10813.5077	151.25867

8.2 Ten Day Lookahead Model

The weights in *alpha* and their associated features are as follows:

Post Length	Post Score	'BTC'	P(Pos)	P(Neut)	P(Neg)	Open	High	Low	Volume
1879.478	543.147	-1057.5147	-1081.1725	2200.945	10096.2127	-9294.7487	6748.666	17252.7180	1626.0173

8.3 Interpreting Results

The model does not appear to converge to a trivial result on the one day look-ahead as it has substantial weights for the post length and FinBert sentiment probabilities. While the price data does have the highest cumulative weights, this is not too concerning as the other variables are weighted heavily. Additionally, it can be seen that the weight of the non-price data increases significantly between the two look-ahead values, suggesting that these features matter more when attempting to make a prediction further into the future. The $t = 10$ model achieved a R^2 above 0.95, meaning that our approach generalizes well for larger amounts of t.

9 CONCLUSION

We learned a great deal from this project. First, we learned that predicting future prices can be very important in developing efficient trading strategies. By implementing our predictions on top of an existing rudimentary trading strategy, we could get very impressive returns. We also learned not to discount the opinion of large internet forums. Oftentimes, online trading speculation can be a question of correlation or causation with respect to its effect on market value. Regardless of this fact, the online sentiment on Reddit still very nearly followed the behavior of Bitcoin's price volatility.

REFERENCES

[1] Jeff Philips. Mathematical foundations for data analysis. https://mathfordata.github.io/versions/M4D-v0.6.pdf.

[2] Dogu Araci. Finbert: Financial sentiment analysis with pre-trained language models, 2019.

[3] Scikitlearn documentation: Minmax scaler. https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.

[4] Scikitlearn documentation: Ridge regression. https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Ridge.

[5] Scikitlearn documentation, lasso regression. https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Lasso.

[6] Scikitlearn documentation: Svr. https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVR.

[7] James Chen. Investopedia: Backtesting, 2021. https://www.investopedia.com/terms/b/backtesting.asp.

[8] Adam Hayes. Investopedia: Simple moving average (sma), 2021. https://www.investopedia.com/terms/s/sma.asp.