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

Alex Tingey, Braxton Booth, Jeremy Cruz, Spencer Cameron

5140 Poster Presentation

Key Ideas

Key ideas for our project include but are not limited to:

- Linear Regression
- Ridge Regression
- Dimensionality Reduction
- Neural Networks
- Polynomial Regression
- Support Vector Regression
- Dataset creation.
- Coefficient of Determination

ABSTRACT

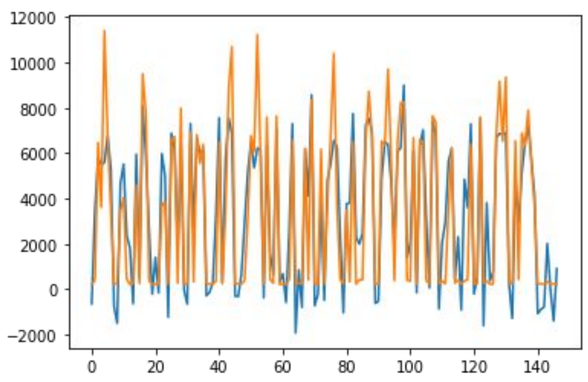
The goal of our project is to analyze the performance of different regression techniques in predicting the price of Bitcoin by using features of posts made on the popular internet forum /R/Bitcoin as input. We aim to use the best selected regression method in an attempt to identify correlation between the posts of users on the site and the percent return of the security during the corresponding period.

Data Processing Pipeline

We have created a Python script which uses the RESTful Pushshift API to collect data from Reddit posts and comments. The collected data includes sentiment analysis of textual bodies, scores (measured as a weighted sum of upvotes and downvotes), and the frequency of the words "BTC" or "Bitcoin" within the posts made to the subreddit /r/bitcoin on a selected day. With this data we then constructed a dataset of X values where each row of the dataset is the average of these features on a given day.

Experiment #1: Linear Regression

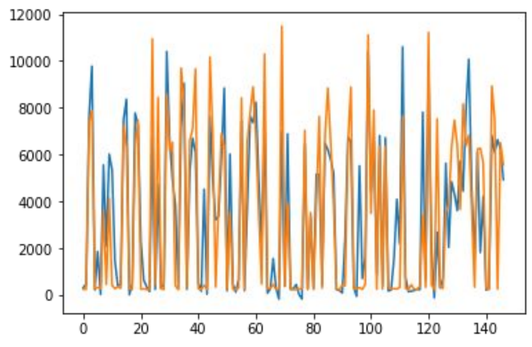
For this experiment, we made use of the sklearn.linear_model library. In the graph on the right, you can see our findings. The orange line represents the actual prices of bitcoin, and the blue our predictions.



In order to obtain the above figure, we created our model using two years of data. We tested the model using an 80-20 split of our data for testing and validation. The reasoning behind using linear regression was that we wanted to gain a baseline for our experiments. 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. This approach achieved an R^2 Score of 0.766.

Experiment #2: Non-Linear Regression

For this experiment, we made use of the sklearn.svm.SVR library. In the graph on the right you can see our findings. Once again, the orange line represents the actual price of bitcoin and the blue represents our predictions.



As before, we created our model with two years of data, utilizing an 80-20 split for testing and validation. There are many reasons to use support vector regression. We first thought about using support vector regression after reading the feedback from our peers in the class. We realized that it could be a good fit for our data. This is in part due to the fact that support vectors allow the user to have more control over the accuracy of the model. The C hyperparameter used in constructing the SVR model is very important to determine the importance of not making errors. The SVR model generalizes well because it allows for some error. In addition to this, SVR enables us to use kernels. The use of a non-linear kernel (RBF kernel) allows us to perform regression in higher dimensions. The tuning of hyperparameters C and epsilon achieved an R^2 Score of 0.77 for this model.

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

Coefficient of Determination R^2

R^2, which ranges from 0 to 1, is a measure of how well a regression model fits the data. It is calculated by subtracting from 1 the sum of squared residuals over the squared difference of the data and its mean (this is proportional to variation). 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 from the input feature vector. R^2 was chosen specifically as the metric to compare each approach as it is possible to compute very efficiently for all regression approaches and the resulting values are very easy to interpret.

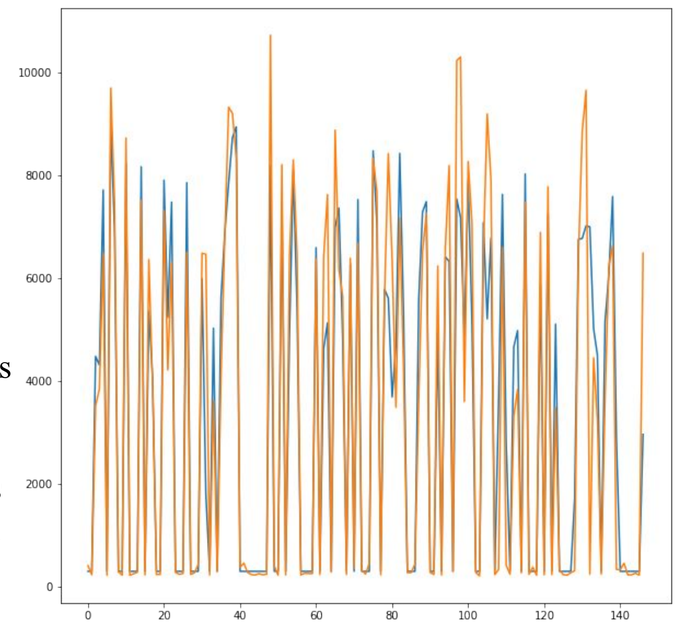
Experiment #3: Lasso Vs. Ridge Regression

For this experiment, we analyzed the performance of two different types of linear regression models. Lasso regression and Ridge Regression. For Lasso we achieved an R^2 Score of 0.666 for this model. For Ridge we achieved an R^2 Score of 0.661 for this model. We had hoped to gain some sort of insight about our data from analyzing their respective accuracies. Although their accuracies are very similar, we can still try to use these findings to formulate some weak assumptions about our data. For one, we can assume that since both models are relatively similar, there aren't many correlated variables. Neither of these solutions offer us any improvement over your standard linear regression.

Experiment #4: Keras Neural Net Regression

For this experiment we created a model using Tensorflow's Keras framework. The model is a generic feedforward ANN which has been tuned by the Kerastuner package to produce a unique shaped network which optimizes the MSE loss on the validation data. This model produced the best results, achieving an R^2 of 0. 862.

Some very interesting behavior arose from the network during hyperparameter 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. The networks that outperformed others often had latent layers with 500 neurons, while the beginning and end layers had less than 100. This could imply that the best network is one that expands the dimensionality of the data to find a better fit.



Results:

There are lots of conclusions we can draw from the four experiments mentioned above.

Firstly, our model works much better when we fit it into high dimensional models. For example, our SVR regression model performs better than our simple linear regression model. That is because the Radial Basis Function kernel used in this method, maps our data into very high dimensional spaces. This allows it to achieve a better approximation than the linear model. This phenomenon is taken a step further if we analyze our neural network regression model. The neural network regression model achieves an even higher accuracy with more dimensions.

Secondly, the addition of more data made our models perform much better compared to our initial findings. Initially we had only 215 days of data. This caused the models we built to not generalize well at all. The highest accuracy we achieved with this data was ~25%.

Thirdly, the accuracy of our model at predicting bitcoin prices shows us that we had a decent process for collecting data. Frequency of the phrase "BTC", score, post length, and sentiment scores seem to correlate well to the rises and falls of the price of Bitcoin.

Finally, we determined a few conclusions from the analysis of the graphs themselves. Our linear models had a tendency to overestimate both the dips and the spikes. Conversely, our non-linear models had a tendency to only overestimate the spikes. However, their overestimates of the spikes is more severe compared to the linear model cases.