Updated Bitcoin Prediction Model Report

**Approach to Model Design and Feature Selection:**

I began with the assumption that independent variables have a simple relationship with the price of Bitcoin rather than a complex relationship. This assumption led me to use linear regression for my model instead of a neural network approach. For feature selection, I first added a 7-day rolling volume weighted average price (VWAP) feature to the data, named vwap\_7d. To do this, first I calculated price\_volume by multiplying each day’s price by the day’s volume. The volume in this case was obtained as the total amount of Satoshis transacted for the day. I then calculated rolling\_price\_volume for the previous 7 days, which is the total price\_volume for the previous 7 days. I then did the same by calculating rolling\_volume as the total volume in Satoshis from the previous 7 days. From these, VWAP for that day is then calculated as rolling\_price\_volume divided by rolling\_volume. Additionally, I added 10 days of lagged price data to the features. Any rows with NAs due to these additions were dropped. I initially trained the model with all available features and assessed feature importance with SHAP values. I decided to take the features contributing the most to the model’s prediction based on the SHAP values. These features were 'lag\_1', 'lag\_7', 'lag\_8', 'lag\_4', 'lag\_2', 'lag\_6', 'vwap\_7d', '1:20Occ', '1:20Sats' and they were the features I used to train the final version of my model.

**Training, Validation, and Testing Strategy Rationale:**

My strategy for training, validation, and testing relied on using as much data as we could gather after removing NAs from the dataset that were introduced by the lagged values, VWAP feature and merging of the 3 provided sets. This resulted in 2394 total days of observations for our prices and features beginning on 2011 day 162. Our test data was limited to just December 2017, which is a small percentage of the total data provided. Due to this, I proceeded with a smaller than usual validation set to provide better approximation of what the testing set results would be. My validation set was approximately 7.5% of the data, with the remaining percentage of data being in the training set. Since the price data provided is temporal in nature, I used the datetime library to help split the test data from the training and validation data based on the requested start date of the test data, 12/01/2017. There are print statements verifying data was kept in temporal order within my code. While training the model and validating, I used a time series split for the training and validation data to ensure no data leakage. This ensured no future prices were used to predict past prices in the training and validation stage. I also scaled my data with StandardScaler. The scalar was fitted to the training data only and then used to transform both train, validation, and test data. This was my approach to avoid any data leakage that could occur by creating a new scalar from my validation and test data separately.

**Model RMSE Value and SHAP Results:**

The Final RMSE value for my model was 837.72. This means that on average, my predicted December 2017 Bitcoin prices were within +/- 837.72 of the actual December 2017 Bitcoin prices. Additionally, the R2 measure of my model for the testing data was approximately 0.86. For the first 5 days of the prediction period in December, my code will output the following print texts: SHAP values for each of the features for the specific day, along with a list of the top 5 most influential features for that day based on the absolute value of the SHAP value. Additionally, the code outputs a SHAP summary plot for each of those days individually. Once those plots have been clicked through, I then output a final SHAP summary plot for the full set of testing data. Based on the results of the SHAP analysis, vwap\_7d seemed to be the biggest contributor to the model’s predictions, providing a much larger contribution than most of the other features.

**Challenges Faced and How They Were Approached:**

The challenges I faced within the model were the selection and fine tuning of arbitrary decisions such as how many features to use, how many days of lagged prices to include, how many splits to use in the time series split of test/validation data, and data pre-processing.

Regarding data preprocessing, I wanted to make sure there were no duplicate columns of data within the 3 files provided. One part of this was addressed above, where I noticed that the ‘TotalTx’ column in the ‘pricedBitcoin2009-2018 file and ‘OccChainletsInTime.txt’file was the same set of data. To remedy this, I removed one of the duplicate columns and kept the other. Another example of data preprocessing challenge was noticing that the ‘AmoChainletsInTime.txt’ file and ‘OccChainletsInTime.txt’file had the same column names for the (i:o) variables; to address this issue, I renamed the columns in both files with suffixes ‘Sats’ for Satoshis and ‘Occ’ for Occurrences, for each file respectively. I also used datetime library to help process the training, validation, and test data based on dates. This helped keep the data in temporal order.

As mentioned above, I addressed the feature selection challenge by initially training the model with all available features. From there, I ran a SHAP analysis to see the most influential features in the model’s predictions. I decided to take the features contributing the most to the model’s prediction based on the SHAP values. These features were 'lag\_1', 'lag\_7', 'lag\_8', 'lag\_4', 'lag\_2', 'lag\_6', 'vwap\_7d', '1:20Occ', '1:20Sats. I originally tried adding in all the chainlet data from 1:1 to 5:5 for both the occurrence and Satoshi data files, but they did not help increase model performance and thus were not included in the final implementation. I also attempted to use the extreme chainlets mentioned in class, 20:20Sats and 20:20Occ, but these also did not result in better test performance. I found the best model test performance to occur with 1:20Sats and 1:20Occ as the chainlet variables included.

Regarding the arbitrary selection of 10 lagged days of prices and n = 12 splits for the time series split, these were approached by adjusting my code to run loops using different numbers for each of the variables for lags and time series splits. From there, I chose the number for each that provided the best performance based on RMSE. In the end, these were 10 for the number of lagged days and 12 for the n of time series splits.