Date of publication: April 25th, 2022

Using Real-time Data to Analyze General Sentiment about Crypto-currency

PETAR GRIGOROV[1], MARIAH MARIN[2], MARC MEIER-DORNBERG [2], KHILNA RAWAL[2], GABRIELLE STONEY[3]
[1]Department of Physical Sciences, College of Arts and Sciences, Embry-Riddle Aeronautical University, Daytona Beach, FL
[2]Department of Mathematics, College of Arts and Sciences, Embry-Riddle Aeronautical University, Daytona Beach, FL
[3]Department of Electrical, Computer, Software, and Systems Engineering, Embry-Riddle Aeronautical University, Daytona Beach, FL

ABSTRACT Crypto-currency is a digital currency designed to work as means of exchange through a computer network. Unlike the stock market, crypto is decentralized, meaning that it does not rely on a central authority (such as a government or bank) to uphold or maintain it. In contrast with actual stocks, crypto-currency is a highly volatile and sentiment-driven form of investment. While stock market indices depend on political and socioeconomic factors such as interest rates, current events, government upheaval, exchange rate fluctuations, natural calamities and so on, the demand for a particular crypto depends mostly on the overall attitude of investors towards it. Social media platforms such as Twitter are among the main sources where information about these sentiments can be extracted. The scope of the project is to use the free Twitter API to collect tweets over a given period of time and use this data to analyze the general sentiment about crypto-currency. In addition to that, the price movements of a crypto-currency (e.g. bitcoin) will be collected over that same period of time. The goal is to predict the price movement for the hours that follow using the sentiment scores mentioned previously. Data will be integrated from two different real-time sources, namely the Twitter API and a crypto-currency data provider. The features will then be extracted and prepared for analysis. This project will leverage cloud capabilities to reliably collect real-time data over a longer time span.

INDEX TERMS Crypto-currency, Stocks, Investment, Social media, Database, Real-time data, Time series prediction

I. INTRODUCTION

THE Twitter API can be used to retrieve and analyze Twitter data using computer programming, as well as build for the conversation on Twitter. Over the years, the Twitter API has grown by adding additional levels of access for developers and academic researchers to be able to scale their access to enhance and research the public conversation. The recently released Twitter API v2 includes a modern foundation, new and advanced features, and quick on-boarding to *Essential* access that allows to retrieve up to 500k tweets per month or 25 requests per 15 minutes. There are also several other APIs for stocks and crypto currency such as CoinAPI, which offers a free plan with up to 100 requests per day.

A. OVERVIEW OF RELEVANT LITERATURE

There have been numerous studies that used Machine Learning models for time series forecasting with the purpose of predicting stock prices or values of crypto currencies, and to conduct sentiment analysis either using Twitter APIs or other media sources. One paper by Ikhlaas Gurrib and Firuz Kamalov, both faculty of the Canadian University of Dubai, proposed a new method to predict values of cryptocurrencies such as bitcoin using sentiment analysis and linear discriminant analysis (LDA). [1] The model essentially trains an LDA classifier that uses information from real-time bitcoin prices and sentimental news/media headlines to forecast bitcoin prices for the following day.

A thesis study done by Roderick Karlemstrand and Ebba Leckstrom from the KTH Royal Institute of Technology attempted to use a Machine Learning time-series model to predict stock prices, this time using Twitter's API to collect data. The model is based on a neural network that is trained with historical stock values and attributes that have been extracted from posts on Twitter. These attributes represent sentiment scores, retweets, followers, etc. [2] The analysis was conducted using a rule-based sentiment analysis tool called *Valence Aware Dictionary and sEntiment Reasoner* (VADER).

A third paper by Chamrajnagar et al. also used Twitter sentiments to predict price fluctuations of *ZClassic*, an alternative cryptocurrency. Each tweet is extracted and classified as either positive, neutral, or negative, and later compiled into two time-series sentiment indices (one weighted and one unweighted). The two indices were trained on an Extreme Gradient Boosting Regression Tree Model. The significance of this paper is that it is the first academic proof of the concept of how powerful social media is in influencing price movements of the highly volatile cryptocurrency market. [3] The final source reviewed for this study is a paper by Loginova et al., which uses similar sentiment-based analysis as the other papers, except the dataset consists of data from Reddit, Bitcointalk and CryptoCompare instead of social media. [4]

B. EXTENDED REVIEW OF LITERATURE

The database project by Chamrajnagar et al. has particularly similar objectives and methods to what our study seeks to accomplish, being the development of a sentiment-based model that predicts cryptocurrency price fluctuations. Although our research focuses on Bitcoin, Chamrajnagar et al. settled on an alternative form of crypto called *ZClassic* (ZCL), due to its high level of predictability via analysis of Tweets. [3] The data (in this case tweets) was collected using the programming language *R*, which features free Twitter analysis packages that are based on statistical computing. One package named *rtweet* was useful in retrieving tweets that contained the terms "ZClassic" or "ZCL" from the previous 7 days. This collection process was repeated 3 times so that there is enough data for analysis. For data preprocessing, all data sets were merged and duplicates were removed, leaving the final data set with 130,000 unique tweets.

A Python algorithm using the *Textblob* package was developed by Chamrajnagar et al. with the purpose of classifying the nature of each tweet, whether positive, negative or neutral. The algorithm is essentially a dictionary, which assigns scores ranging from -1 to +1 to key sentimental words inside a tweet. For example, the word "top" earns the tweet a value of 0.5, while "not great" will result in a -0.4. The combination of words inside a tweet will determine the overall "polarity" of that tweet. Fully positive and negative tweets will receive sentimental values of +1 and -1 respectively, while neutral tweets (with an even combination of positive and negative polarities) will be assigned a 0 sentimental value. Retweets were treated differently by Chamrajnagar et al. Noting their ability to "cause a chain effect, thereby increasing the dispersion of the initial tweet" [3], Chamrajnagar et al. scaled retweets by a factor of 2, where the sentimental scores for them now range from -2 to +2.

For choosing an optimal model, Chamrajnagar et al. used a 10-fold cross validation on their data set to determine a model that features the smallest data loss and highest accuracy. The model selected for their study is a tree ensemble model called *Extreme Gradient Boosting Regression* (XGBoost), which "outputs a weighted sum of the predictions of multiple regression trees by weighing mislabeled examples more heavily" [3]. The advantage of this model, according to Chamrajnagar et al, is that XGBoost is a rule-based learning method. It has a higher potential than traditional regression models to uncover relationships between the features and extracting important information from them. In addition to that, their lack of sensitivity to the range of the data and features eliminates the need for normalization, which would often results in data loss. The key concept behind XGBoost can be defined using the equation below:

$$\hat{y_i} = \phi(x_i) = \sum_{k=1}^{K} f_k(x_i), \quad f_k \in F \tag{1}$$

The prediction value $\hat{y_i}$ from the model at a given observation *i* is equivalent to the predicting function $\phi(x_i)$, which is the sum of each tree *f* in the regression tree forest *F*. This leads to a minimization problem as shown below:

$$\mathscr{L}(\phi) = \sum_{i} l(\hat{y_i}, y_i) + \sum_{k} \Omega(f_k), \tag{2}$$

where

$$\Omega(f) = \gamma\tau + \frac{1}{2}\lambda \parallel \omega \parallel \tag{3}$$

The function $l(\hat{y_i}, y_i)$ represents the mean-squared error, and the $\Omega(f_k)$ term is a "regularization parameter that penalizes each tree for having too many leaves" [3]. To minimize the equation, Chamrajnagar et al. used a greedy algorithm to create the regression tree forest *F*. In constructing the model, they assigned about 30 percent of the data to a testing set, while the remaining data was used for training the model.

Another project by Roderick Karlemstrand and Ebba Leckstrom, graduate students from the KTH Royal Institute of Technology, also developed a sentiment-based model using Twitter API, except this model predicts the prices of stocks instead of cryptocurrencies. Tweets were collected using a library called *Twitter4J* and later pre-processed and cleaned in a Python program using the machine learning library Scikit-learn. The sentiment scores were calculated using *VADER* (Valence Aware Dictionary and sEntiment Reasoner) - a rule-based model for sentiment analysis. Being available as a Python package, a class called *SentimentIntensityAnalyzer* was used by Karlemstrand and Leckstrom, which takes text as input and outputs a "score" object. The weighted compound scores for each tweet were extracted and stored in a CSV file, and later added to the data set for the training phase of the project.

The model chosen for the machine learning aspect of that project is a neural networks type. According to Karlemstrand and Leckstrom, a neural networks model was selected because of its ability to "compute and combine both statistical patterns, time-oriented patterns and random occurrence in data". A model like this has a major advantage in accuracy, where there would be minimal underfitting or overfitting. In addition to that, two types of input training data can be combined. [2] In the case for Karlemstrand and Leckstrom, historical stock data and twitter data were merged. Historical data was necessary for taking past trends into consideration. After creating a complete dataset with the sentiment scores now calculated using *VADER*, the data set was split into a training, verification, and a testing set, which were then used in the neural networks machine learning algorithm.

For performance analysis of the machine learning model, Karlemstrand and Leckstrom computed the mean squared error (MSE) of the predicted price and real price of the stocks.

$$MSE = \frac{1}{n}\sum_{t=1}^{n} {e_t}^2 \tag{4}$$

In this equation e represents the error between the real close price and predicted price, and n is the number of stocks analyzed. The MSE is the average of accumulated error across the whole validation and testing data set to measure the actual performance of the neural network model. [2]

II. MATERIALS AND METHODS

CLOUD platforms provide numerous advantages over a traditional on-premise architecture. Services can be activated easily and often without the need to install them. Cloud native-services are maintained by cloud service providers in a robust manner so that efforts for technical operation and maintenance can be reduced. The most important advantage of cloud computing for this project is the possibility to collect data reliably over the course of several weeks.

The project team has decided to use the Google Cloud Platform (CGP) for retrieving and storing data, training the model, and computing the prediction. The system architecture that takes advantage of GCP's cloud-native services is presented in the following.

Figure 1 shows a high-level system overview. Data on cryptocurrency price movements is collected from an API provided by Coin API and tweets from Twitter are collected using the Twitter API. A sentiment score is computed for every incoming tweet. Both data feeds are stored in BigQuery, the data warehouse solution in GCP [5]. A predictive model is trained on the collected data to forecast the price movements for the respective succeeding time period. A prediction is updated periodically so that a system user can make investment decisions in real-time. The results are ultimately displayed on a dashboard.

FIGURE 1. System Overview

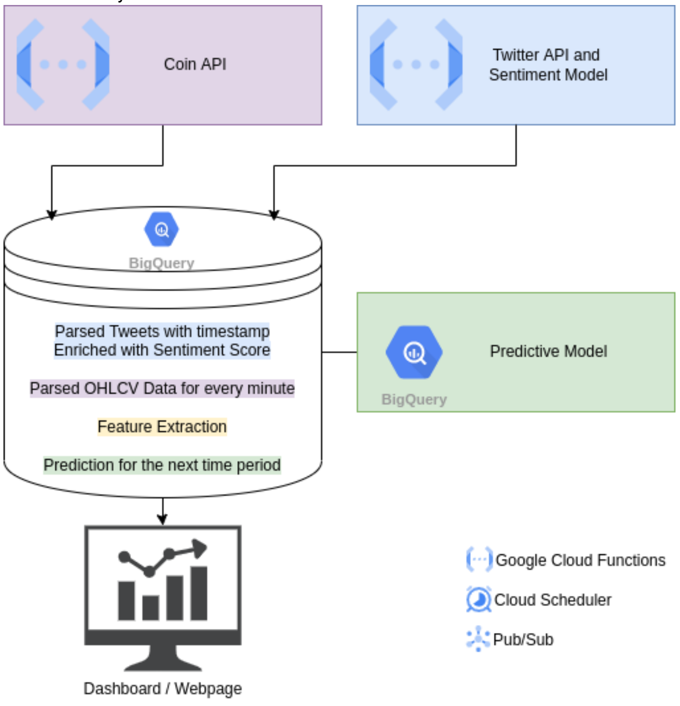
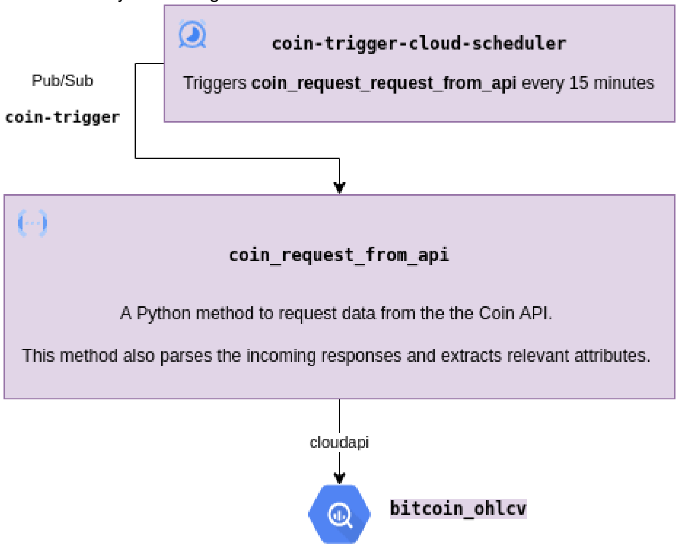

Coin API
Twitter API and Sentiment Model
BigQuery
Parsed Tweets with timestamp Enriched with Sentiment Score
Parsed OHLCV Data for every minute
Feature Extraction
Prediction for the next time period
Predictive Model
BigQuery
Dashboard / Webpage
Google Cloud Functions
Cloud Scheduler
Pub/Sub

FIGURE 2. System Design Coin API Connection


coin-trigger-cloud-scheduler
Triggers coin_request_request_from_api every 15 minutes
Pub/Sub
coin-trigger
coin_request_from_api
A Python method to request data from the the Coin API.
This method also parses the incoming responses and extracts relevant attributes.
cloudapi
bitcoin_ohlcv

A. API CONNECTIONS

Both API connections, the Coin API (Figure 2) as well as the connection to the Twitter API (Figure 3), follow the same architectural pattern. Both connections use Google Cloud Functions, a lightweight solution in GCP, to deploy server-less functions [6]. The Cloud Functions contain Python code that produces the desired outcome. The initial target architecture foresaw a pattern in which the data retrieval, data parsing, and data preparation are each done in individual Cloud Functions. These Cloud Functions would have been connected via an asynchronous messaging queue (Pub/Sub) [7] to allow for a decoupled processing. However, performance tests on local machines showed that the processing time for these steps is minimal so that decoupling is not necessary. For this reason, this project's data ingestion pipeline uses only two Cloud Functions. One Cloud Function is used for the retrieval and ingestion of Coin API data, the other Cloud Function for retrieval and pre-processing

FIGURE 3. System Design Twitter API Connection

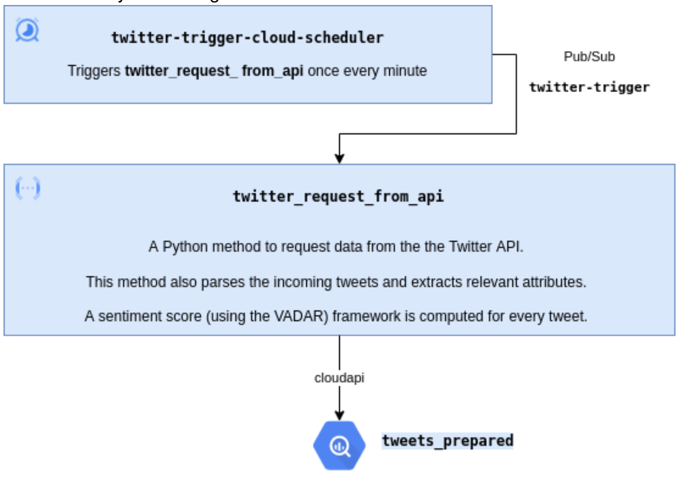

twitter-trigger-cloud-scheduler
Triggers twitter_request_ from_api once every minute
Pub/Sub
twitter-trigger
twitter_request_from_api
A Python method to request data from the the Twitter API.
This method also parses the incoming tweets and extracts relevant attributes.
A sentiment score (using the VADAR) framework is computed for every tweet.
cloudapi
tweets_prepared

FIGURE 4. bitcoin_ohlcv (left) and tweets_filtered (right) Tables in Big Query

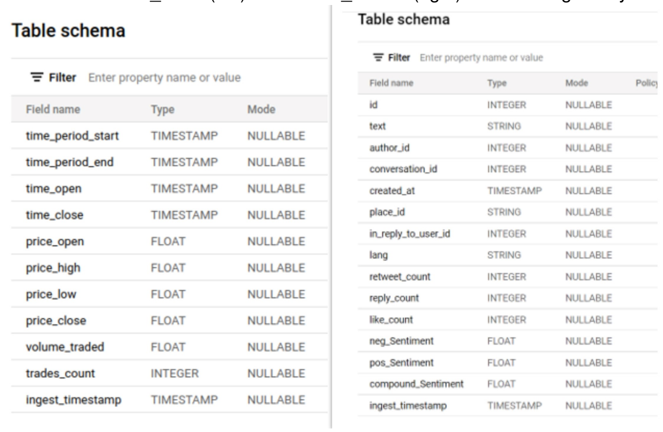


Table schema

Filter Enter property name or value

Field name	Type	Mode
time_period_start	TIMESTAMP	NULLABLE
time_period_end	TIMESTAMP	NULLABLE
time_open	TIMESTAMP	NULLABLE
time_close	TIMESTAMP	NULLABLE
price_open	FLOAT	NULLABLE
price_high	FLOAT	NULLABLE
price_low	FLOAT	NULLABLE
price_close	FLOAT	NULLABLE
volume_traded	FLOAT	NULLABLE
trades_count	INTEGER	NULLABLE
ingest_timestamp	TIMESTAMP	NULLABLE

Table schema

Filter Enter property name or value

Field name	Type	Mode	Policy
id	INTEGER	NULLABLE	
text	STRING	NULLABLE	
author_id	INTEGER	NULLABLE	
conversation_id	INTEGER	NULLABLE	
created_at	TIMESTAMP	NULLABLE	
place_id	STRING	NULLABLE	
in_reply_to_user_id	INTEGER	NULLABLE	
lang	STRING	NULLABLE	
retweet_count	INTEGER	NULLABLE	
reply_count	INTEGER	NULLABLE	
like_count	INTEGER	NULLABLE	
neg_Sentiment	FLOAT	NULLABLE	
pos_Sentiment	FLOAT	NULLABLE	
compound_Sentiment	FLOAT	NULLABLE	
ingest_timestamp	TIMESTAMP	NULLABLE	

of Tweets from the Twitter API. Both Cloud Functions use Python libraries (cloudapi) to write data directly into the BigQuery tables.

Google Cloud's implementation of a cron service, Cloud Scheduler [9], is used to schedule the API calls. Cloud Scheduler triggers the Cloud Functions periodically via messages on a dedicated Pub/Sub queue. The API calls are triggered, in a way that makes use of the upper limit of allowed API calls per time period (e.g., one API call every 15 minutes to CoinAPI).

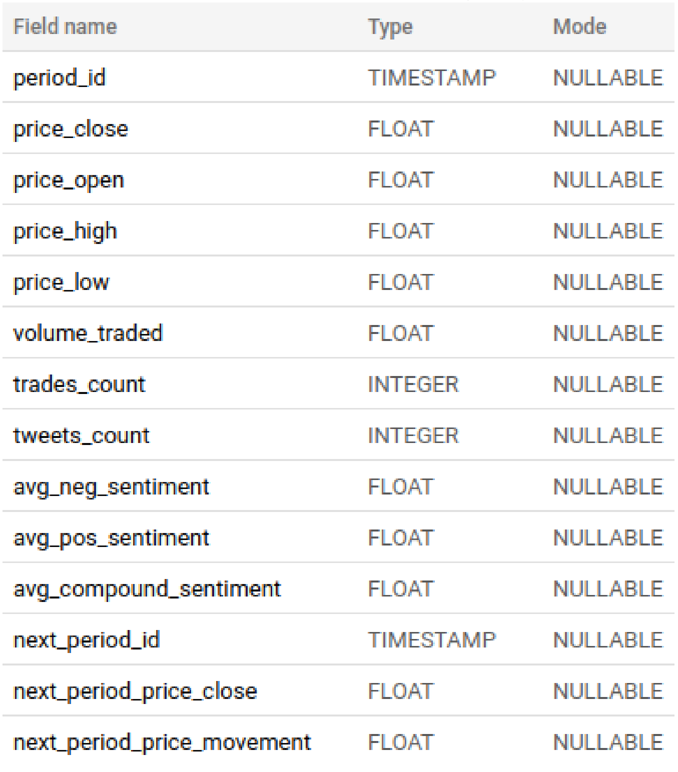
B. DATA STORAGE AND DATA PREPARATION

As mentioned previously, data collected by the Cloud Functions is stored in BigQuery. BigQuery has a landing zone for the data ingested by the Coin API stream and for the data ingested by the Twitter API stream. One table in the landing zone contains one record with cryptocurrency price data for the respective period (bitcoin_ohlcv). Another table contains one tweet and an attached sentiment score per record (tweets_prepared). These two tables are combined and consolidated before the data can be used for model training. Since the table tweets_prepared contains tweets for which no sentiment score could be computed, a view is created. The view (tweets_filtered) is a one-to-one representation of its base table tweets_prepared but it does not contain tweets for which no sentiment score could be calculated.

There are two possible approaches to prepare the input tables for model training. Tables can be written periodically (e.g., after every time the CoinAPI stream added a record to the landing zone) or database views can be used. The second option was preferred since development efforts can be kept low by using database views. However, this approach was found to cause higher costs during model development. For this reason, a scheduled query was developed to update the base table for model development and prediction. This table (bitcoin_sentiments_consolidated) is updated every hour so that new data can be used to predict the bitcoin price for the next hour.

Figure 5 shows the schema description of the BigQuery table bitcoin_sentiments_consolidated.

FIGURE 5. bitcoin_sentiments_consolidated Table in Big Query



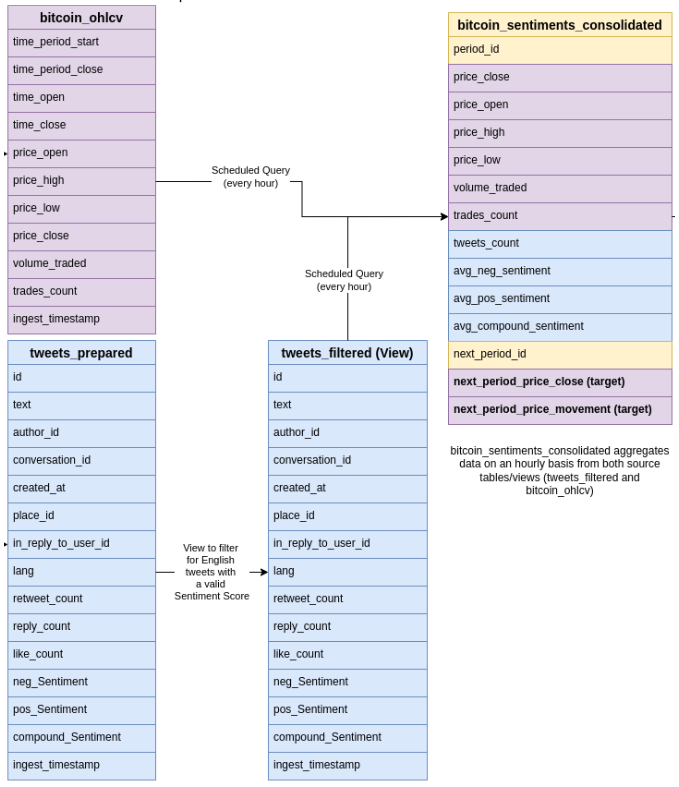
Field name	Type	Mode
period_id	TIMESTAMP	NULLABLE
price_close	FLOAT	NULLABLE
price_open	FLOAT	NULLABLE
price_high	FLOAT	NULLABLE
price_low	FLOAT	NULLABLE
volume_traded	FLOAT	NULLABLE
trades_count	INTEGER	NULLABLE
tweets_count	INTEGER	NULLABLE
avg_neg_sentiment	FLOAT	NULLABLE
avg_pos_sentiment	FLOAT	NULLABLE
avg_compound_sentiment	FLOAT	NULLABLE
next_period_id	TIMESTAMP	NULLABLE
next_period_price_close	FLOAT	NULLABLE
next_period_price_movement	FLOAT	NULLABLE

Figure 6 provides an overview of the dataflow from the landing tables to the table with consolidated data.

C. MODEL

BigQuery ML allows developers to create Machine Learning models with SQL syntax [10] and also generate tables using the data, as seen in Figure 4. Developers can create, train, and store the model in BigQuery itself. This is the

FIGURE 6. Data Preparation Overview



preferred way of creating a model for this project as it allows the team to focus on the model itself rather than additional interfaces to BigQuery. The model is used within BigQuery to generate predictions that can be read from a consuming application such as a dashboard. The model is using linear regression to predict the price of Bitcoin, creating an output table with a prediction every hour. This means that new data is used for a prediction in near-real-time. The new prediction can be read from the output table with Tableau.

The initial linear regression model consisted of 3 parts: model creation, evaluation, and implementation. While creating the model, a preprocessing step was included to ignore data from the last hour for which there are no values for closing prices of Bitcoin yet. Five attributes from the datasets were considered by the model: average Bitcoin compound sentiment score for each time period, opening/closing prices of Bitcoin, average positive and average negative sentiments of Bitcoin. The model's accuracy was then evaluated on a consolidated table that merged the deterministic features from both the Twitter dataset and Bitcoin price dataset. The model was lastly implemented to predict price of Bitcoin.

Unfavorable accuracy performance and prediction results of the initial model called for some modifications to it. Firstly, the target variable was changed from Bitcoin 'closing price' to Bitcoin 'price movement' because it was decided that the change in price at the end of a time period would lead to better prediction accuracy than just the final price. Supporting attributes for the model's input were introduced and then fine-tuned. Those additional attributes include the volume and count of Bitcoin trades, Bitcoin tweet counts, and the net price movements of Bitcoin at the end of each time period. The dataset was also split into a training and testing set, where the model would be trained on 80% of the dataset and its accuracy would be evaluated on the remaining 20% of the dataset.

III. RESULTS AND DISCUSSION

As previously mentioned, models within BigQuery can be opened through user-friendly software that can help visualize the data and draw conclusions from it. *Tableau* was used in this case to generate line graphs and scatter plots with the goal of discovering meaningful relationships within the data, in addition to creating a dashboard of all the findings.

FIGURE 7. Bitcoin Price Fluctuations over a Period of One Month

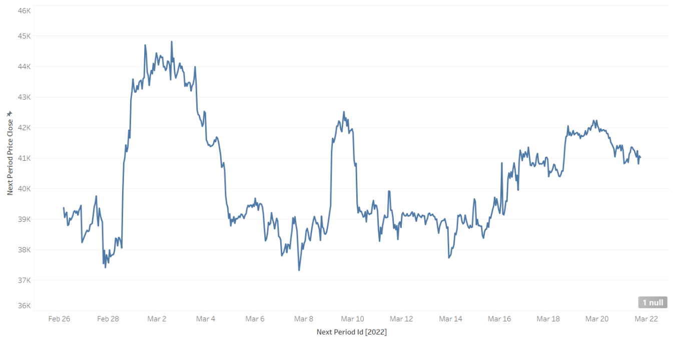


Figure 7 shows how the price of Bitcoin has changed over the time frame in which the tweets were collected. There is a broad peak at the beginning of March, as well as a more narrow peak around March 10th, where Bitcoin costed 5K USD more than usual. Since cryptocurrency is sentiment-driven investment, it was hypothesized that a large number of positive tweets and retweets about Bitcoin would lead to more interest in investors purchasing Bitcoin. As a result, the price of Bitcoin would increase with a larger number of tweets with a positive sentiment. Plotting the hourly average compounded sentimental value of each tweet over that same period of time, as seen in Figure 10 would therefore hypothetically show similar fluctuations as Figure 7. Unfortunately, this was not the case. It was noted that for any given day, there was an even distribution of tweets with both positive, negative and neutral sentimental values.

Another attempt was made to capture the correlation between sentimental tweets and the price of Bitcoin by using the price movement, which is the net change of Bitcoin price at the end of each hour. Figure 11 is a scatter plot showing how much the price of Bitcoin has changed at any hour, given the average compound sentiment score of all tweets posted in that hour. It was expected that a large compound sentiment score would correlate with a positive price movement. Unfortunately that was not the case in this scatter plot, which showed no correlation between the two attributes.

Evidently something went wrong either in the data collection process, or the data was adulterated with a high number of outlier tweets that have not been accurately interpreted by the *VADER* sentiment analysis tool. To spot the discrepancy, a bar chart (see Figure 12) that displays the average number of likes, replies, and retweets for tweets corresponding to a specific compound sentiment score was created. It was noted that positive tweets are more likely to get liked or retweeted. This is expected, since humans are more likely to respond to positive content and more likely to ignore negative content. However, an anomaly can be spotted near a compound score of '0' on both the 'like' and 'reply' bar charts. Upon further investigation of the data, it was found that a large number of collected Tweets were in languages other than English. Since the *VADER* sentiment analysis tool is meant to analyze texts in English, all foreign language tweets that featured the Bitcoin hashtag were assigned a default sentiment score of '0' (neutral). To correct for this issue and hopefully improve the model's performance, a decision was made to remove all non-English tweets and run the model again.

Figures 13 and 14 at first glance demonstrate promising results. Figure 13 is a scatter plot of the relationship between the actual Bitcoin closing prices and the closing prices predicted by the linear model. Evidently the two variables are highly correlated, suggesting that the linear model has accomplished the task. Figure 14 also shows how similar the predicted closing prices (in blue) are to the actual closing prices (in orange) using a time series line graph. It must be noted that the time series graphs were filtered to show only 2 days of data collected over 1-hour intervals. The reason for that is to better visualize the prediction accuracy of the model, as it will be difficult to spot the differences when a larger time frame is used. The wildly fluctuating nature of cryptocurrency prices makes for graphs with an abundance of peaks and troughs. A large time series of such will result in an extremely compressed graph where similarities and differences between prices will be difficult to visualize and detect. Figure 15 is another time series line graph, but this time for the predicted (in yellow) and actual (in purple) price changes of Bitcoin at the end of each 1-hour period. This line graph shows that the linear model is not successful in predicting the price movements of Bitcoin. The model is highly underfitting, as it struggles to fully capture the fluctuations of price movements. The model is flexible enough to predict the trend or overall direction of Bitcoin prices, but it is not flexible enough to do so at a particular instance. It appears that this model attempts to 'play it safe' by trying hard not to fit directly to the highly fluctuating data, suggesting that a linear model is not suitable for such data.

Running performance metrics on the model yielded unsatisfactory results. The MSE (mean-squared error) is extremely large, and the coefficient of determination metric (R-squared) is barely 1%. Figure 8 reports these performance metrics. On a more positive note, it appears that most of the prediction errors lie closer to zero. As seen in Figure 9 which is a histogram showing the distribution of differences between the predicted and actual price movements, the majority of observations have standard errors between -200 and +200. Considering that Bitcoin prices are in the order of tens of thousands, it can be assumed that the model is fairly accurate. However, the underfitting nature of the model shows that it lacks precision. In other words the model gets the job done, but it is not reliable or flexible enough for precise predictions at a specific time due to high simplification bias.

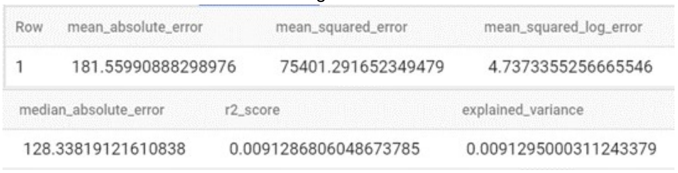


FIGURE 8. Performance of Linear Regression Model

Row	mean_absolute_error	mean_squared_error	mean_squared_log_error
1	181.55990888298976	75401.291652349479	4.7373355256665546

median_absolute_error	r2_score	explained_variance
128.33819121610838	0.0091286806048673785	0.009129500031243379

There are a few possible reasons why the model does not produce good results. The computed sentiment scores from the *VADER* library could be incorrect or incomplete. A way to adjust for the possible inaccuracies of the sentiment scores would be to sanitize the tweets by adding more recognition parameters. In addition to that, broadening the hashtag search is another step to be taken which will hopefully provide the model with more data to work with. Moving forward if these actions do not improve the model, there will most likely be non-linear correlation (or correlation of a combination of attributes) that are yet to be discovered. No matter the case, these correlations would need to be captured with a more complex model. A Neural Network model will most likely perform better for this task. Research procedures for sentiment analysis from relevant projects will be investigated to see the computational differences between them and the model used in this project. This will provide insight on why the current model does not perform well.

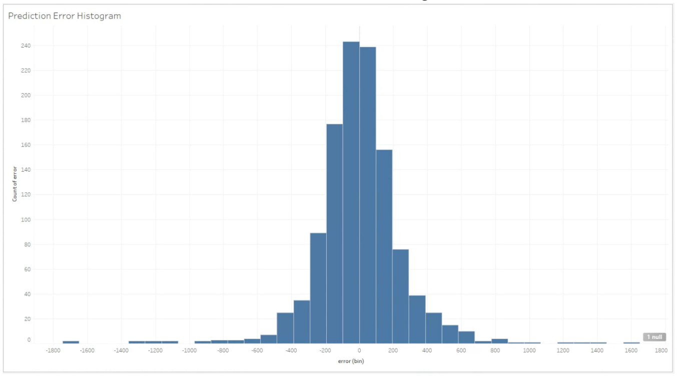


FIGURE 9. Price Movement Prediction Error Histogram

FIGURE 10. Hourly Average Twitter Sentiment Scores over a Period of One Month

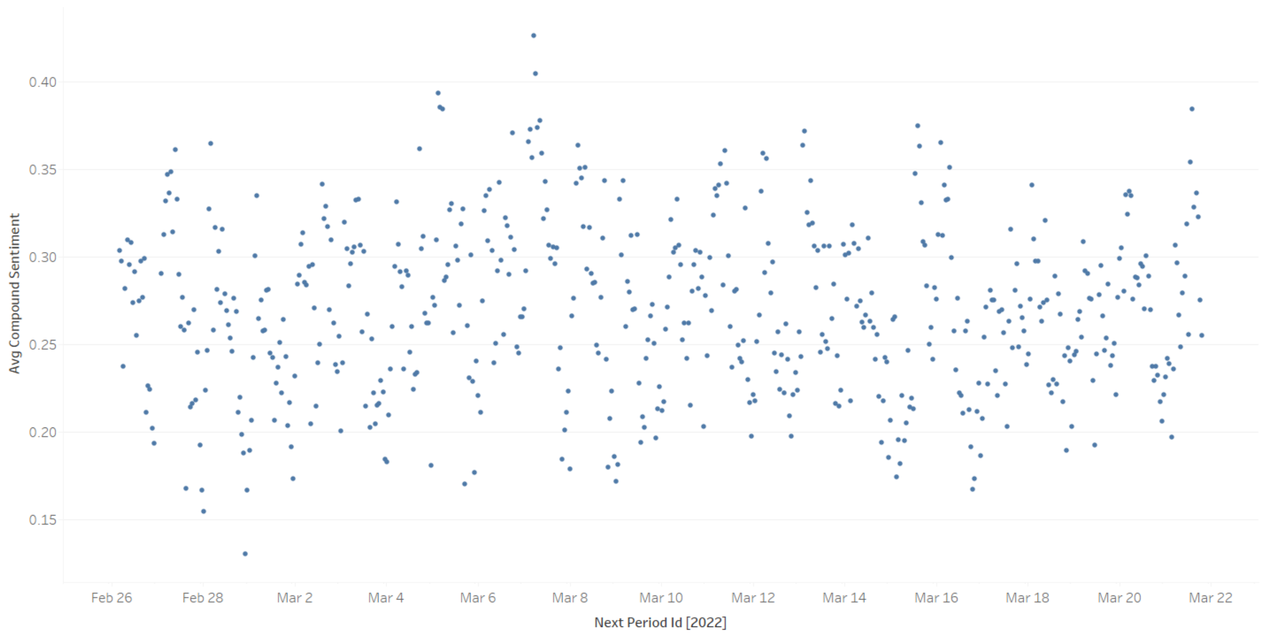


FIGURE 11. Relationship of Bitcoin Price Net Changes and Twitter Sentiment Scores

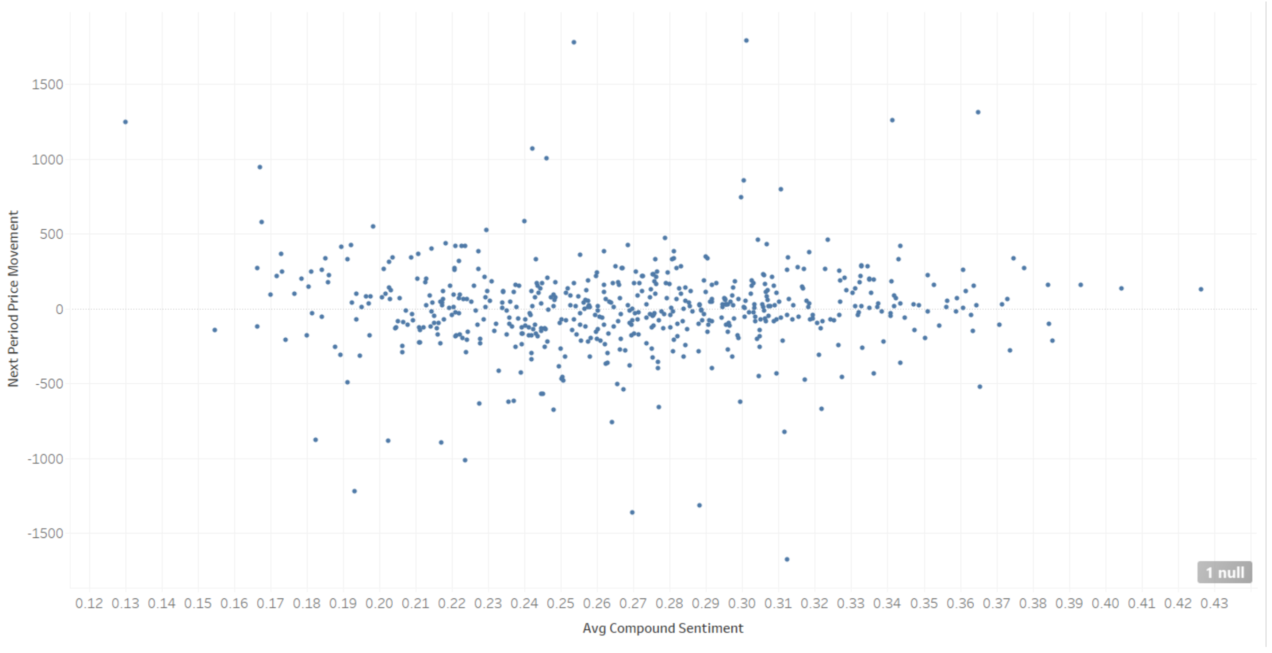


FIGURE 12. Average Number of Likes, Replies, and Retweets of Tweets for each Sentiment Score

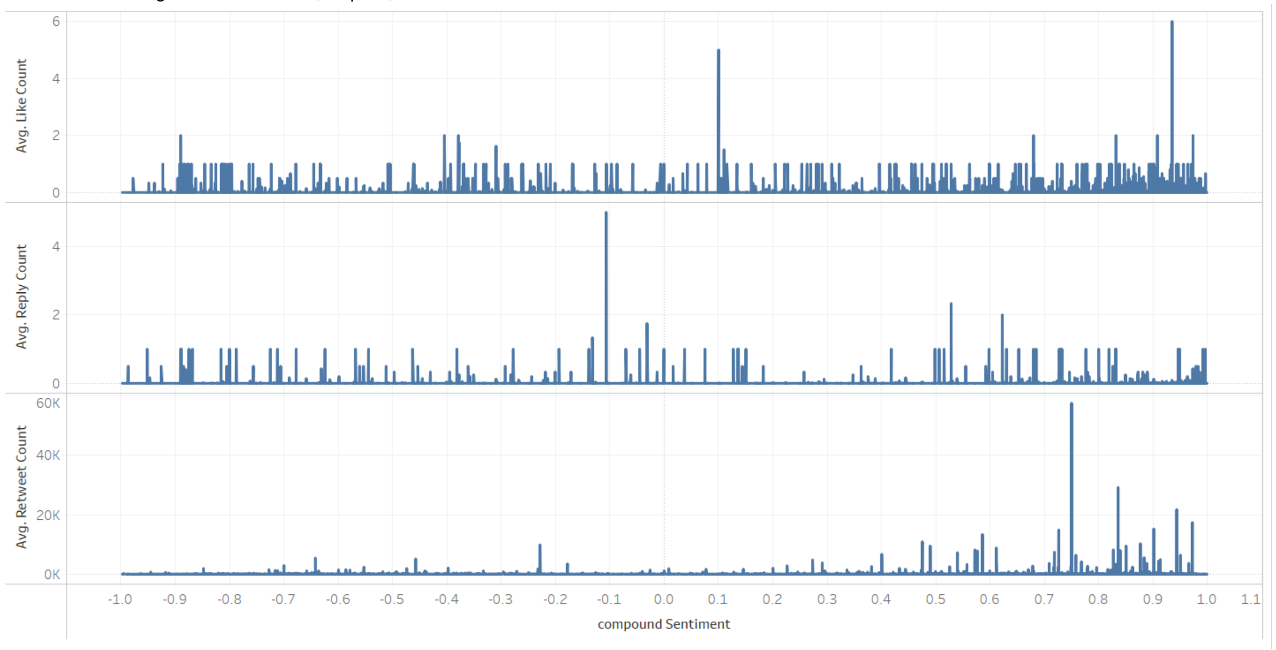
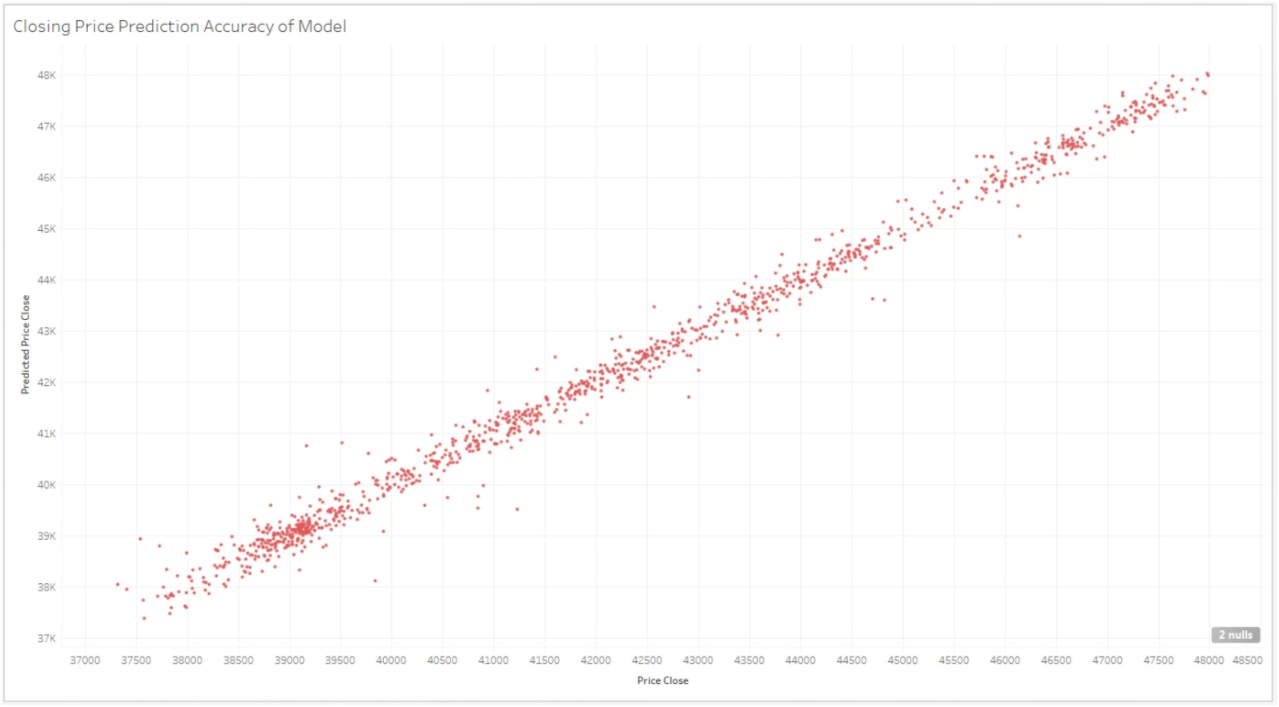


FIGURE 13. Bitcoin Closing Price Prediction Accuracy of Linear Model



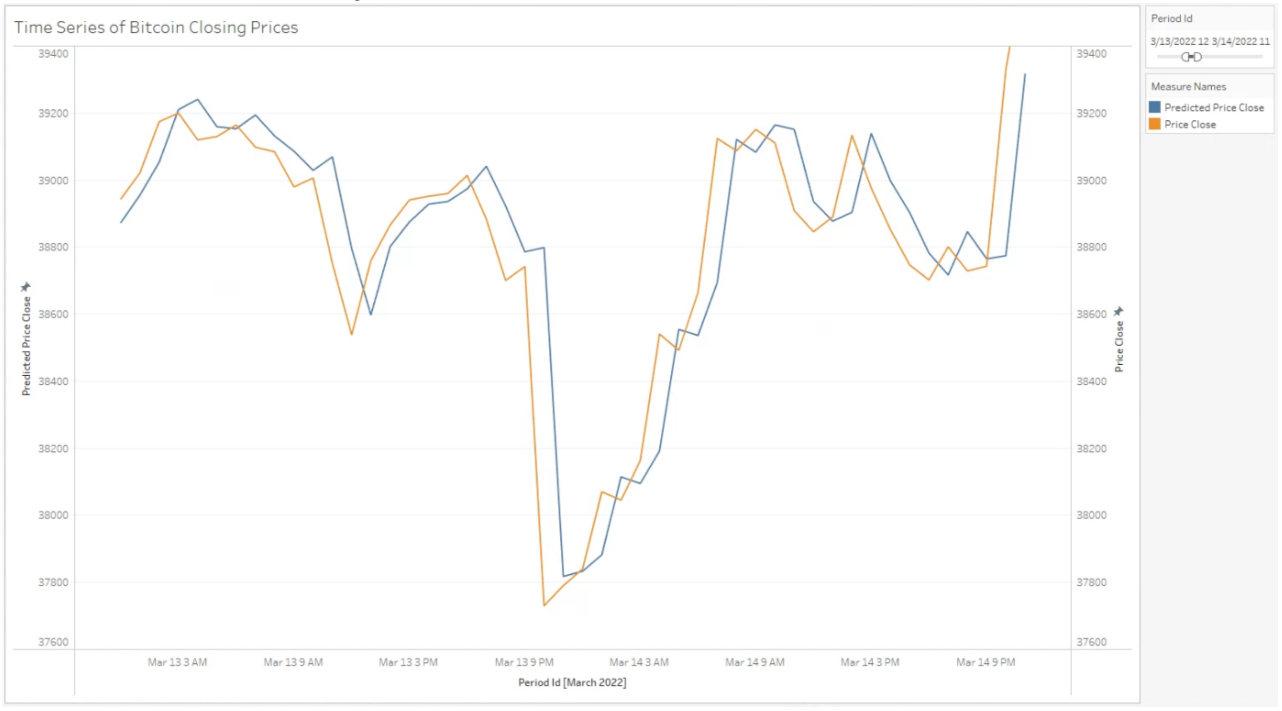


FIGURE 14. Time Series of Bitcoin Closing Prices

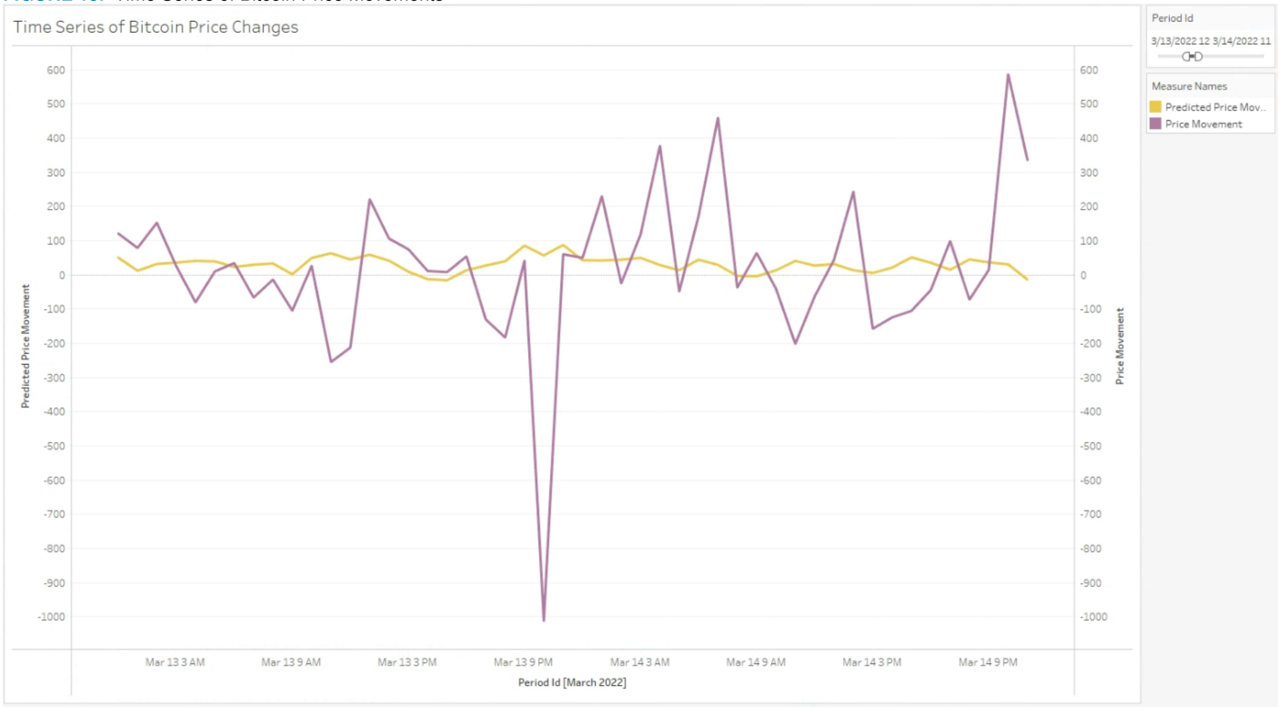
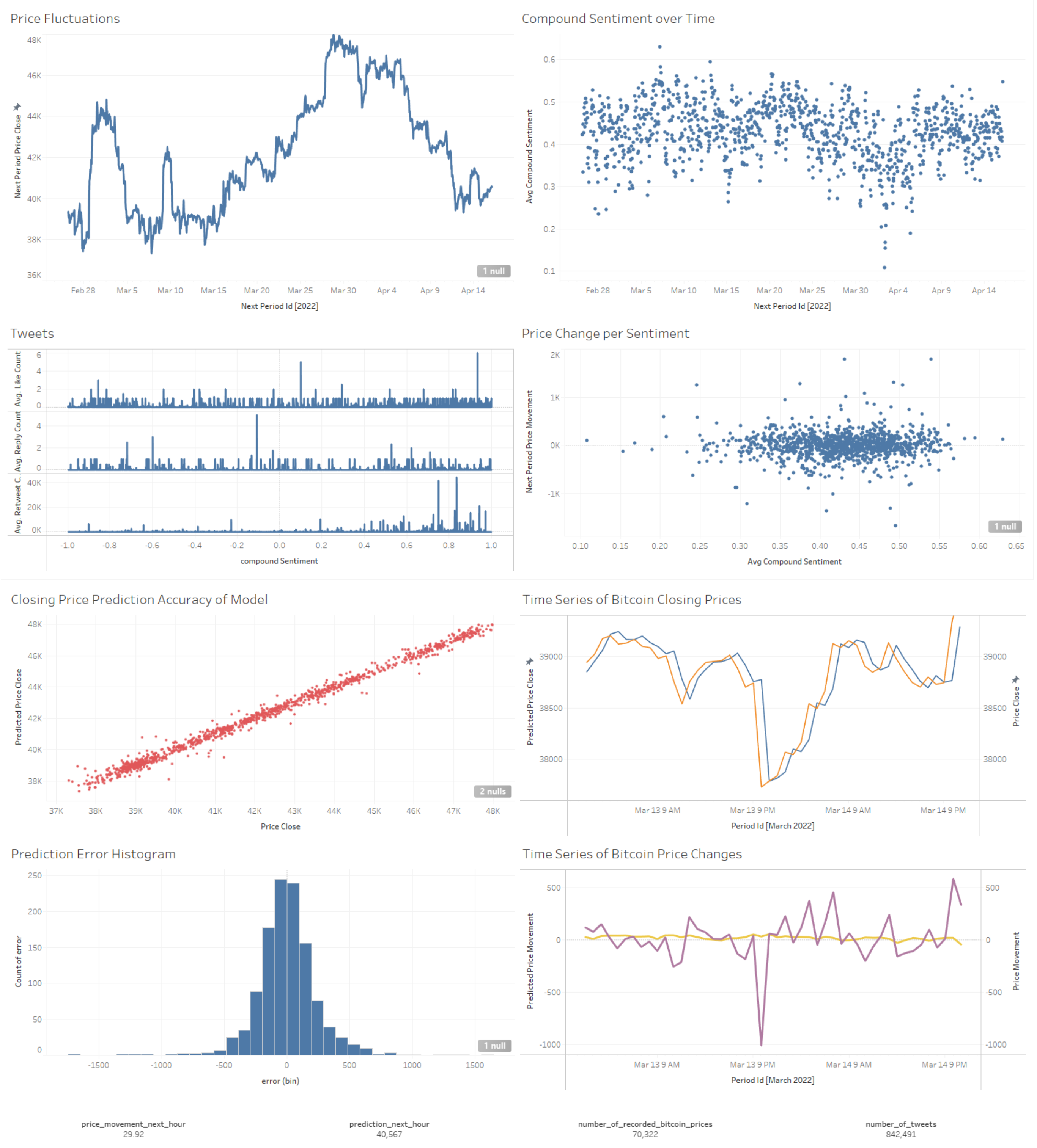


FIGURE 15. Time Series of Bitcoin Price Movements

IV. DASHBOARD



price_movement_next_hour	prediction_next_hour	number_of_recorded_bitcoin_prices	number_of_tweets
29.92	40,567	70,322	842,491

REFERENCES

[1] Gurrib, I., amp; Kamalov, F. (2021, December 15). Predicting bitcoin price movements using sentiment analysis: A machine learning approach. Studies in Economics and Finance. Retrieved February 1, 2022, from https://www.emerald.com/insight/content/doi/10.1108/SEF-07-2021-0293/full/html?skipTracking=true

[2] Karlemstrand, R., amp; Leckstrom, E. (2021, May 4). Using Twitter attribute information to predict Stock Prices. Using Twitter Attribute Information to Predict Stock Prices. Retrieved February 1, 2022, from https://arxiv.org/pdf/2105.01402v1.pdf

[3] Li, T. R., Chamrajnagar, A. S., Fong, X. R., Rizik, N. R., amp; Fu, F. (2019, July 10). Sentiment-based prediction of alternative cryptocurrency price fluctuations using gradient boosting tree model. Frontiers. Retrieved February 1, 2022, from https://www.frontiersin.org/articles/10.3389/fphy.2019.00098/full

[4] Loginova, E., Tsang, W. K., van Heijningen, G., Kerkhove, L.-P., amp; Benoit, D. F. (2021, November 18). Forecasting directional bitcoin price returns using aspect-based sentiment analysis on online text data - machine learning. SpringerLink. Retrieved February 1, 2022, from https://link.springer.com/article/10.1007/s10994-021-06095-3

[5] Google Cloud. BigQuery. Retrieved February 9, 2022, from https://cloud.google.com/bigquery

[6] Google Cloud documentation. Google Cloud Functions. Retrieved February 9, 2022, from https://cloud.google.com/functions/docs#docs

[7] Google Cloud documentation. What is Pub/Sub? Retrieved February 9, 2022, from https://cloud.google.com/pubsub/docs/overview

[8] Google Cloud documentation. Google-provided streaming templates. Retrieved February 9, 2022, from https://cloud.google.com/dataflow/docs/guides/templates/provided-streaming

[9] Google Cloud documentation. Cloud Scheduler. Retrieved February 9, 2022, from https://cloud.google.com/scheduler

[10] Google Cloud documentation. What is BigQuery ML? Retrieved February 9, 2022, from https://cloud.google.com/bigquery-ml/docs/introduction

[11] Dunn, James George. "Using Python and Pandas to Analyze Cryptocurrencies with CoinAPI." James George Dunn, WordPress, 5 Oct. 2020, from https://jamesgeorgedunn.com/2020/10/05/using-python-and-pandas-to-analyse-cryptocurrencies-with-coinapi/.

[12] Diagrams created with *draw.io*