Bitcoin Day Trade Price Forecast

# Abstract

Stock Market prediction has been a matter of high interest, from financial sectors to individual traders, for many years. Different approaches, using technical indicators and statistical analysis, have been implemented, as well as the use of Machine Learning Algorithms. The purpose of this project is to develop a model for short-term prediction, for day trade strategies, of the price fluctuation for the cryptocurrency of Bitcoin in the Bitstamp Exchange. The dataset used for this project is available at Kaggle with the historical minute-to-minute update of OHLC (Open, High, Low, Close), Volume in Bitcoin and in USD, as well as the weighted bitcoin price, from December 31, 2011 to August, 2019. The techniques implemented will be of time-regression analysis and recurrent Neural Networks, utilizing R, Python and Tableau for data visualization.

# Introduction

Stock Market forecast has always been a common interest in the financial market. Several methods involving statistical, technical analysis are daily applied towards defining trading strategies for long or short periods. With the develop of Neural Networks algorithms, many advents of trying to use this technology have been studied to increase prediction’s accuracy and stability.

Cryptocurrency, such as Bitcoins, have been the focus of many traders for the past few years. Having a volatile trading history with high price increases and drops in 2013 and 2018. Typical trading strategies can involve buying and selling during medium to long periods, e.g days and weeks. Other type of strategy, is performing operations within each day, i.e intraday or day trading. The later benefits from short-term price fluctuation typically using periods of one, five, fifteen, thirty and sixty minutes (Seagal 2019).

The objective of this project is to verify to what extent can applying Neural Networks help forecast the next minutes of Bitcoin prices within a day fluctuation, using previous data and statistical indicators.

# Literature Review

## In this chapter the history of cryptocurrency and its exchange market, as well as statistical models, technical analysis, and Neural Network applications towards stock price forecasting are reviewed.

## 2.1. Cryptocurrency

## Cryptocurrencies are defined as digital tokens commodity, that are used as digital store of value (Hougan, Kim, & Lerner, 2019). They utilize a cryptography system that works as an anonymized ledger of financial transactions called Blockchains, to allow secure transfer in a distributed and decentralized manner (Sundararajan, 2016). The main idea of the cryptocurrency is to allow parties to transact directly without the need for a trusted party (Nakamoto, 2008). In 2009 cryptocurrency Bitcoin emerges, and since then it has matured significantly into a large network of computers around the world.

## Cryptocurrency exchange market

## Whilst envisioned as a peer-to-peer digital currency, bitcoin and other cryptocurrencies soon demanded a grow in electronic trading venues to facilitate trading. Exchanges can be used to buy, sell and trade cryptocurrencies for other cryptocurrencies or physical currencies, allowing buyers and sellers to negotiate prices offering liquidity and setting a reference (Hileman, Rauchs ,2017). The first exchange was founded in 2010 and since then many other exchange markets have been created, dealing with a variety of cryptocurrencies. The largest crypto exchanges today are sizeable enterprises based in different countries around the world.

## Stock prediction a statistical time series approach

## Stock prediction is defined as predicting the future stock price using the historical data. Since the stock market information is a collection of variables over time it can classified as a time series. As described in Guida (2019) typical statistical models applied for time series analysis include autoregressive (AR) models, moving averages (MA) models, mixed autoregressive moving averages models (ARMA) models, seasonal models, unit-root nonstationarity, regression models with time series errors, and differenced models for long-range dependence. For multivariate time series analysis Vector autoregression (VAR) models are one of the most widely statistical methods applied. All these models attempt to capture the linear relationship between the current time step (t) and the information available prior to time t. Correlation plays an important role in understanding these models. In particular, correlations between the variable of interest and its past values correlations.

## Stock prediction a Technical Analysis approach

## Technical indicators are a standard method for analyzing stock price variation. Technical analysis relays totally on company’s stock price in market, and volume trade on a particular price (Shah, Parth, 2015), and use charts and indicators to show the current trend and trend reversals. Some important technical indicators include: Relative Strength Index (RSI), Moving Average Convergence Divergence (MACD), Stochastic Oscillator (Stoch), Rate of change (RoC). (Technical Indicator and Overlay, 2020)

## Stock prediction a Neural Network approach

# As Bhat & Kamath (2013) elucidates “the development of technology and mathematics various new technological methods have been proposed for stock price prediction, which are applied along with technical methods. Genetic Algorithm (GA), Support Vector Machine (SVM), or Neural Networks (NNs) etc. have been applied to predict Stock prices.” (p 1).

# A common family of Neural Networks used for time series forecasting are called recurrent Neural Networks (RNN). Differing from the normal feed-forward NNs, RNN are capable to capture the temporal context in its feedback connections, which are capable of capturing the time varying dynamics of the underlying system (Bianchi et al. 2017).

# Dataset

The dataset for this project is publicly available at www.kaggle.com/mczielinski/bitcoin-historical-data with a CC BY-SA 4.0 license that allows share and adapt the dataset for any purpose, even commercially. The dataset was first created by extracting the information from the exchange (Bitstamp) utilizing an API. For the purpose of this project the dataset is already in a csv file with minute to minute updates of OHLC (Open, High, Low, Close), Volume in BTC and USD, and weighted bitcoin price. The data also have a Timestamp in Unix time. Timestamps without any trades or activity have their data fields filled with NaNs. If a timestamp is missing, or if there are jumps, this may be because the exchange (or its API) was down, the exchange (or its API) did not exist, or some other unforeseen technical error in data reporting or gathering.

## Data Exploration

Code Available at: https://github.com/kmaciver/Ryerson\_Capstone/tree/master/Data\_Cleaning

Initial Dataset is comprised of 8 variables and 3.997.697 observations. Where the variables present the following data types:

|  |  |  |
| --- | --- | --- |
| **Variable** | **Data Type** | **Description** |
| Timestamp | int | Unix time for every minute |
| Open | num | Open-price of Bitcoin for every span of one minute |
| High | num | High-price of Bitcoin for every span of one minute |
| Low | num | Low-price of Bitcoin for every span of one minute |
| Close | num | Close-price of Bitcoin for every span of one minute |
| Volume\_.BTC. | num | Volume of Bitcoin transacted for every span of one minute |
| Volume\_.Currency. | num | Volume of USD transacted for every span of one minute |
| Weighted\_Price | num | Average Price of Bitcoin for every span of one minute |

Table 1 – Data Structure

After Timestamp conversion, and insertion of a date column, the summary of the data set is the following:

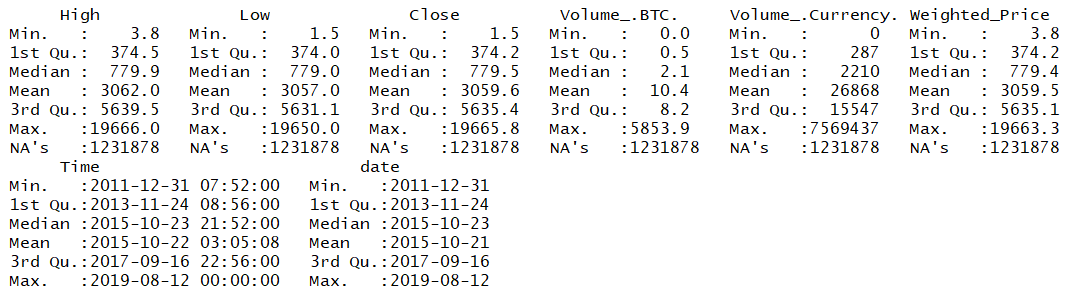


Figure 1 – Data Summary

The figure above illustrates the time range of the data being from December 31, 2011 to August 12, 2019.

Generating initial visualizations:

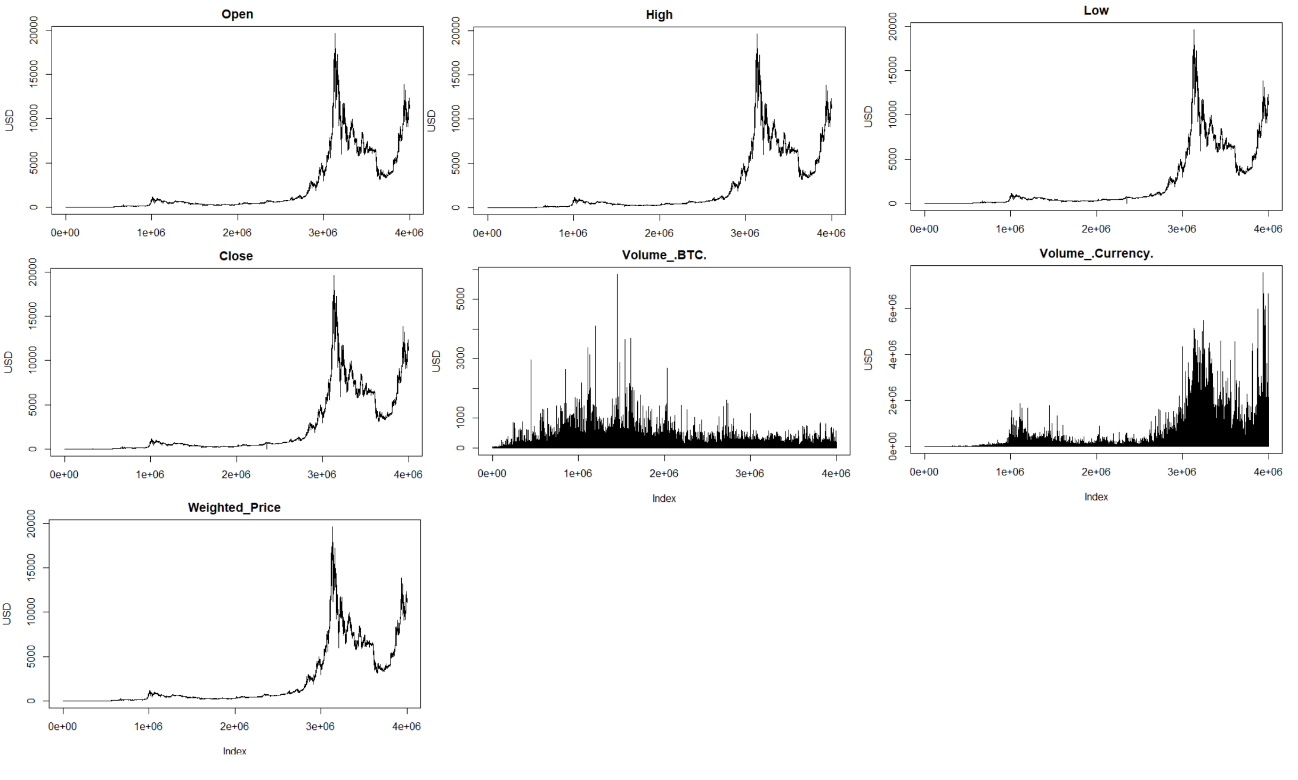


Figure 2 – Variables Plot

Analyzing Variables distribution:

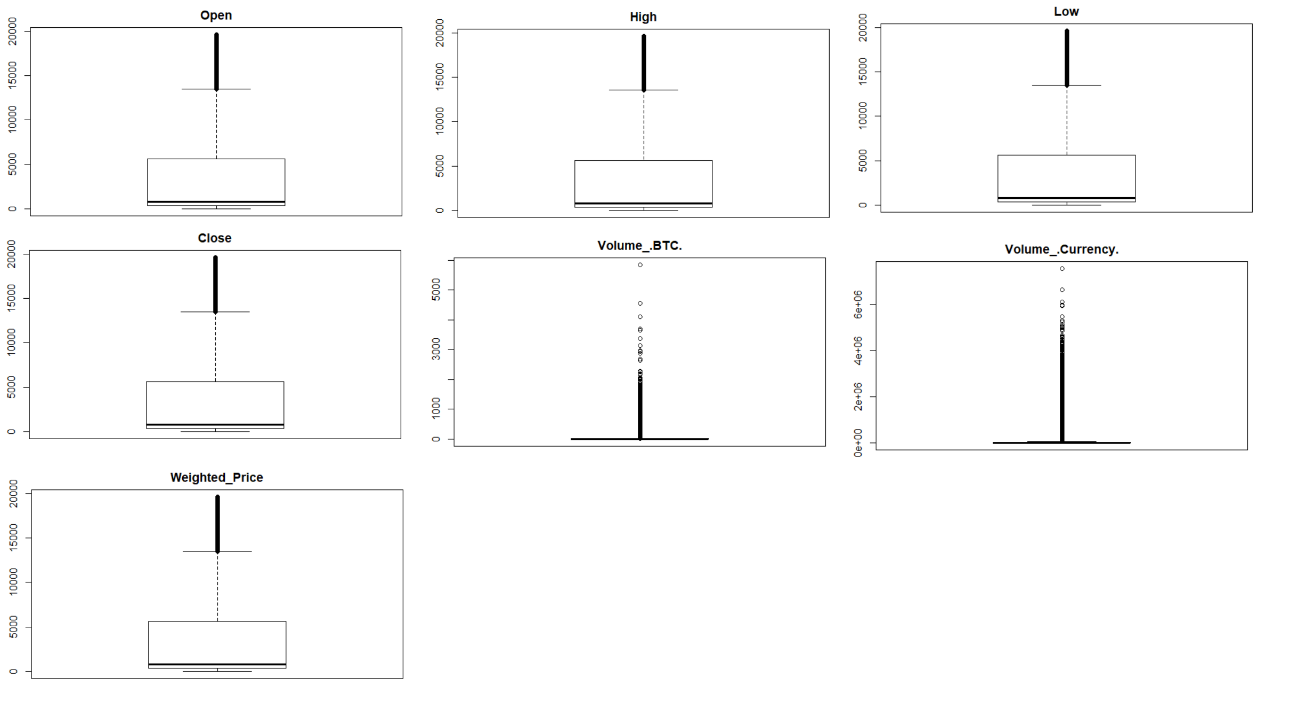


Figure 3 – Boxplots

Although boxplots above describe some considerable number of outliers, the summary of the data shows that there are no negative or other inconsistent number present. Bitcoin has had a volatile history so outliers are most likely to be present, therefore no treatment for outliers was implemented. The table bellow lists the percentage of outliers present on each variable.

|  |  |
| --- | --- |
| Variable | % Outlier |
| Open | 1.28% |
| High | 1.29% |
| Low | 1.27% |
| Close | 1.28% |
| Volume\_.BTC. | 8.31% |
| Volume\_.Currency. | 9.97% |
| Weighted\_Price | 1.28% |

Table 2 – Outliers

Plotting variables histogram distribution.

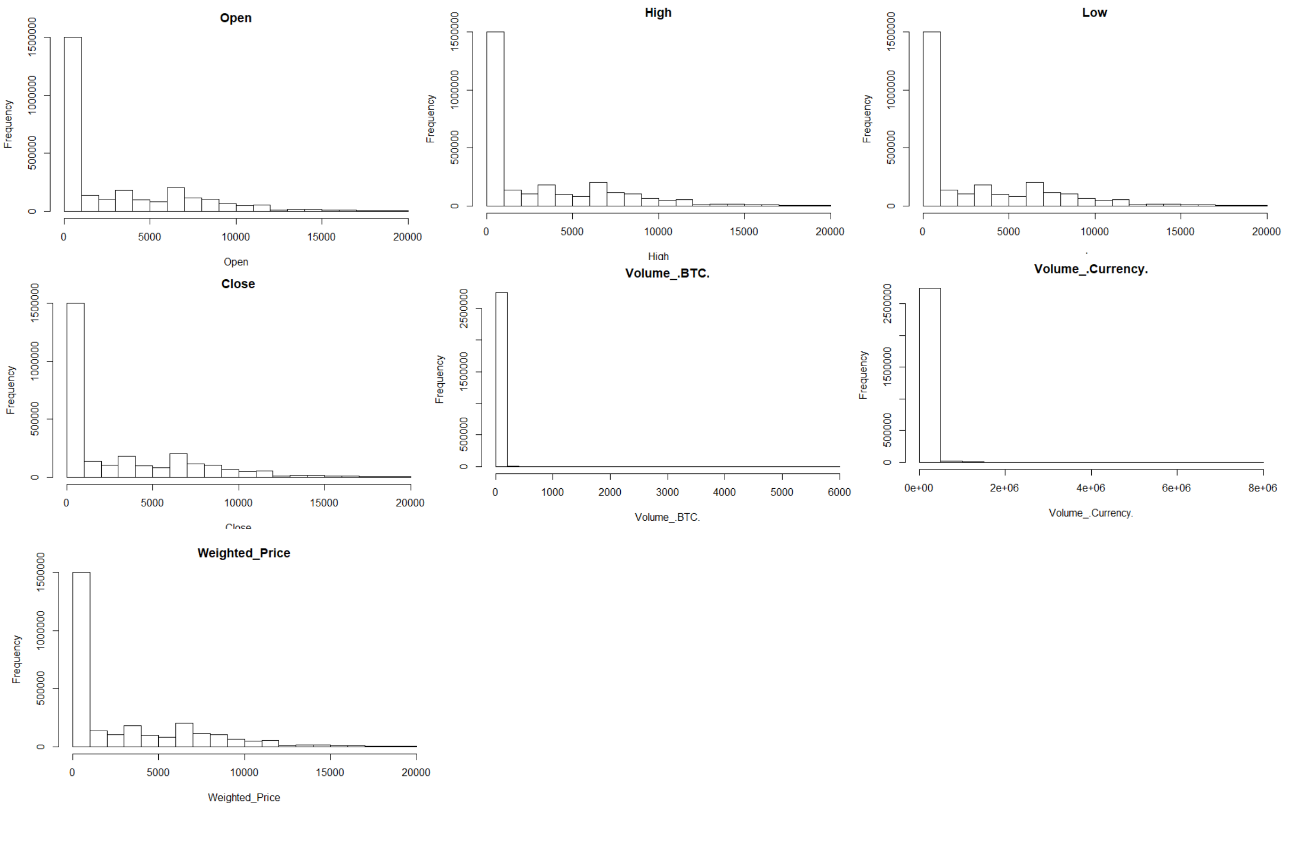


Figure 4 – Histogram

## Data Cleaning

Code Available at: https://github.com/kmaciver/Ryerson\_Capstone/tree/master/Data\_Cleaning

As showed in Figure 1, the dataset has a considerable amount of missing values. The approach for handling the missing information was to first analyze the distribution of NaN values within each day.

|  |  |
| --- | --- |
| Distribution | % Missing Records per Day |
| Min. | 0.00% |
| 1st Qu. | 4.79% |
| Median | 22.08% |
| Mean | 30.80% |
| 3rd Qu. | 41.60% |
| Max. | 99.93% |

Table 3 – Distribution of Missing Records per day

The table above describes that fifty percent of the days have at least 22.08% of missing records, and an average of 30.8% of missing records within each day.

Sparse data within a day won't be useful for the predicting algorithm. Therefore, only the data of the days that have at most the average percentage of the missing records distribution will be used.

For the remaining missing records, the approach implemented was the insertion of the values corresponded to the previous known value, also taking into account that at the end all days should have 1440 observations, considering that’s the of total minutes of a day.

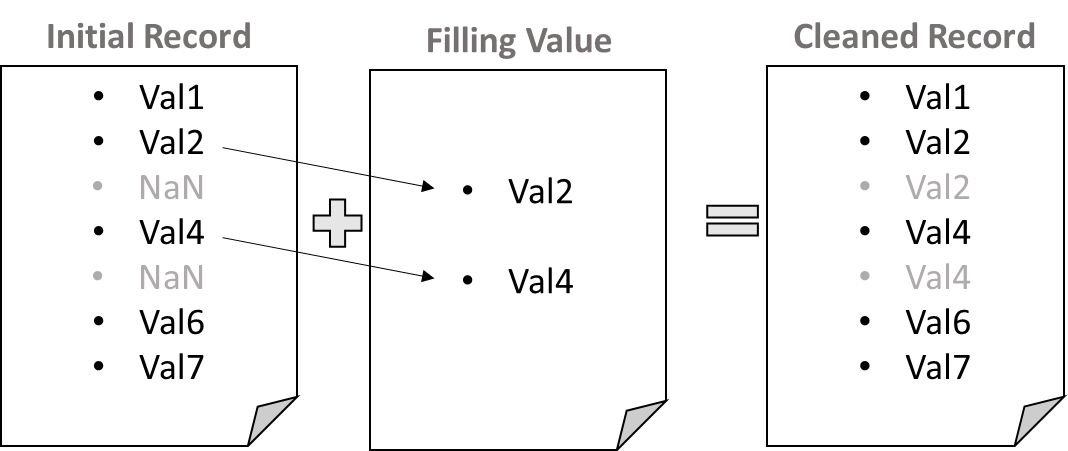


Figure 5 – Missing Record Filling procedure

### Daily Summary Data

The objective of the project is to predict future prices within a particular day. As will be further discussed during the approach paragraph, two data sets will be used: one containing the price fluctuation in minutes for each day; and one containing a daily summary of the price fluctuation. The later will be used as a boundary for the intraday prediction.

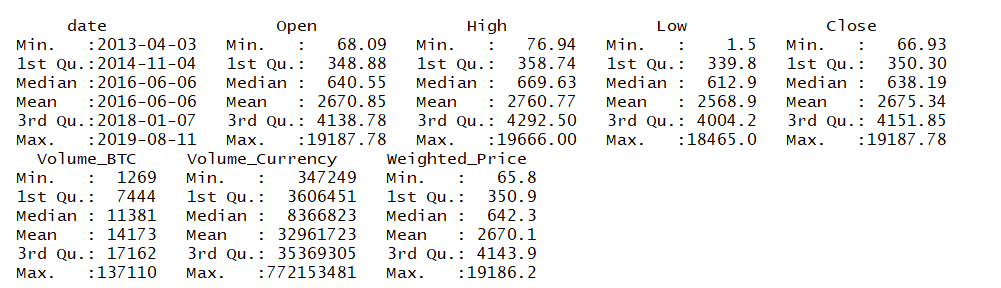


Figure 6 – Daily Summary Data

Although daily summary data didn’t present any missing records, there were still gaps of missing days within the data. Since daily summary will be used for future prediction on variability of data, the data must be continuous throughout the time span. Therefore, missing dates where filled with prior records following the same procedure used previously.

### Technical Indicators

Technical indicators are commonly used in chart technical analysis for stock market forecast. These indicators help identify current trends and trends reversals. Initially, four technical indicators will be added to the dataset.

### RSI

Relative Strength Index (RSI) indicates the overbought and over sold regions and hence the change in momentum. This oscillates between 0 and 100. Above 70 is marked as oversold region and below 30 is marked to be overbought region. RSI can also be used to see the general trend. (Bhat & Kamath, 2013)

Where,

### MACD

Moving average convergence divergence (MACD) has two lines namely MACD line and Signal Line, which give us signals of trend changes with cross overs. These two lines also show the movement of Stock with their coming closer to each other(convergence) and departing from each other(divergence). (Bhat & Kamath, 2013)

Where,

### STOCH

Stochastic Oscillator is a momentum indicator that relates the location of each day's close relative to the high/low range over the past n periods.

Where,

### RoC

Rate of Change indicator finds percentage difference of a series over two observations.

Since all indicators use past values inserting them into the data set generates new missing records.

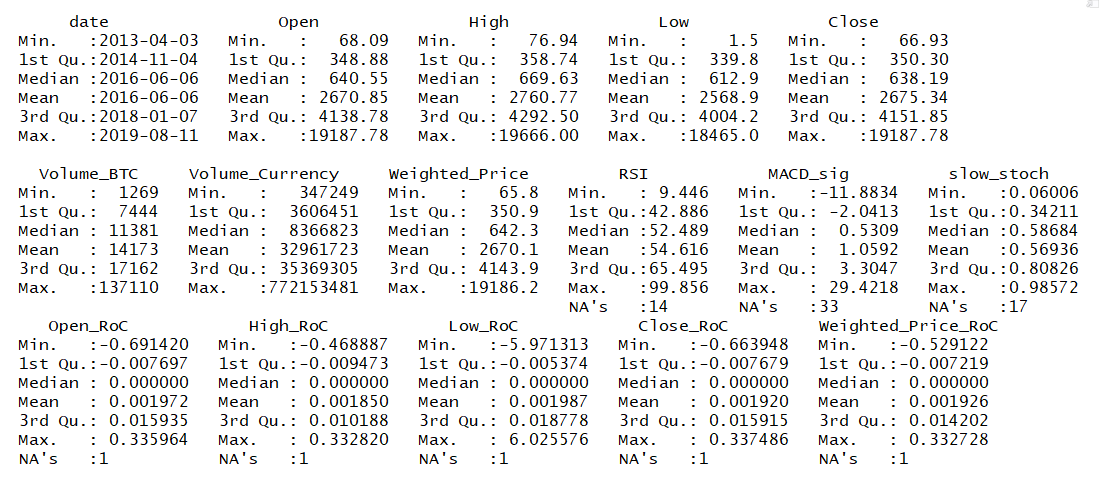


Figure 7 – Daily Summary Data, with Technical Indicators and NaN values

In order to maintain all records, missing records from the RoC technical indicators were filled with 0. For remaining technical indicators, missing value will be filled with the first value known.

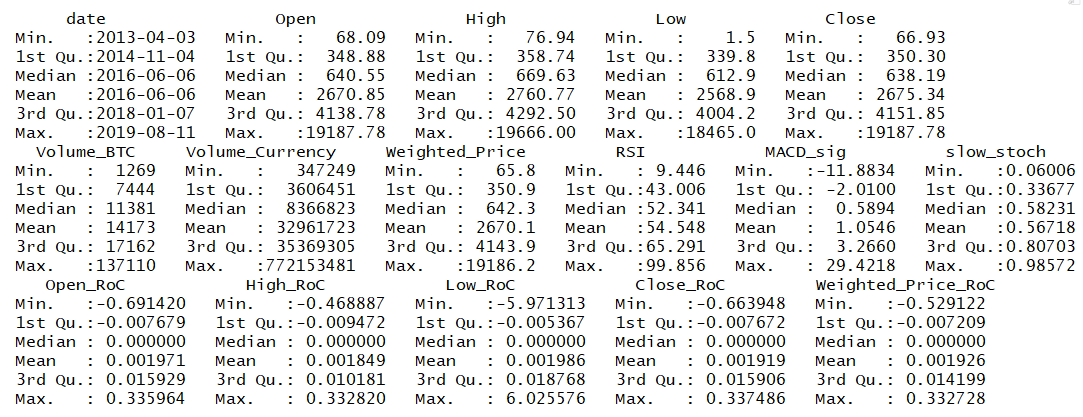


Figure 8 – Cleaned Daily Summary Data, with Technical Indicators

Final size of Daily Summary data is 2322 observations and 16 variables.

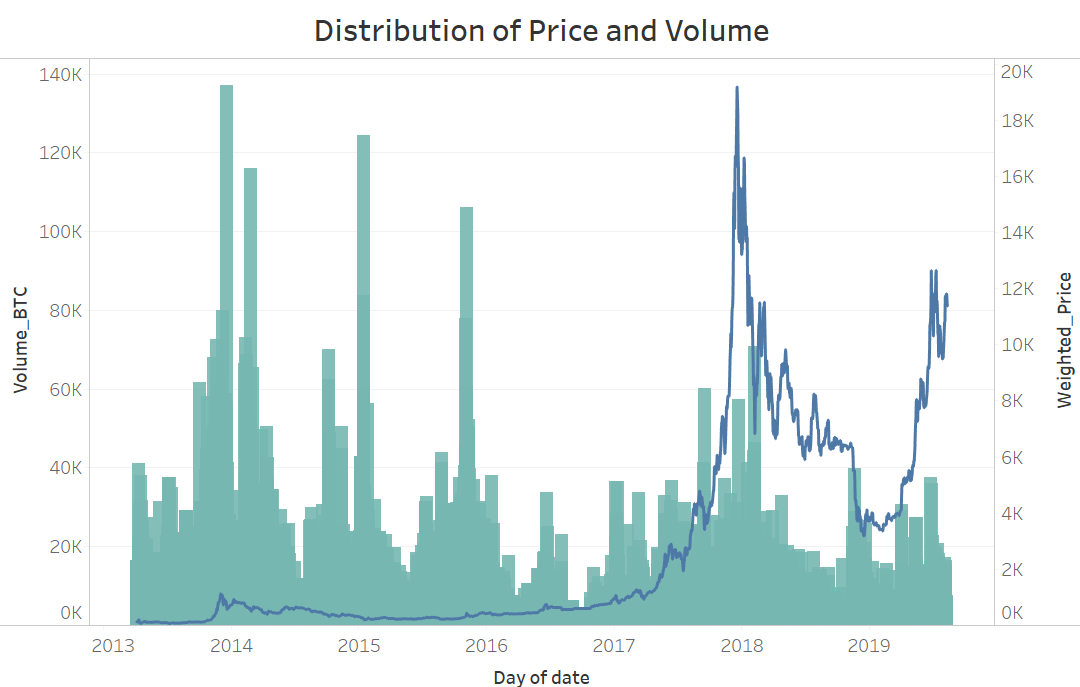


Figure 9 –Daily Summary Data price and Volume

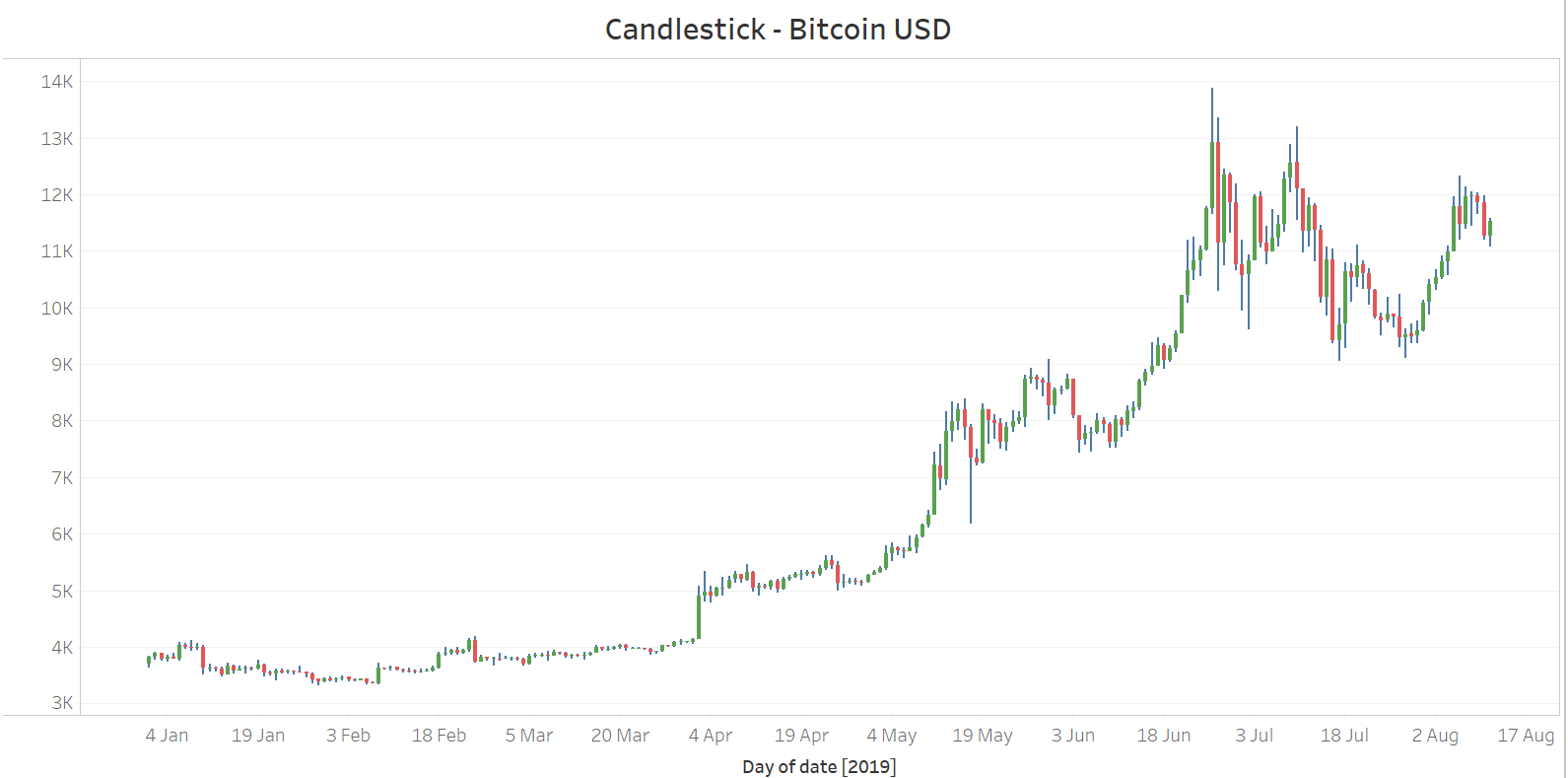


Figure 10 –Daily Summary Data Candlestick chart period of 2019

### Day Trade Data

Following the same steps as before Day Trade Data is created.

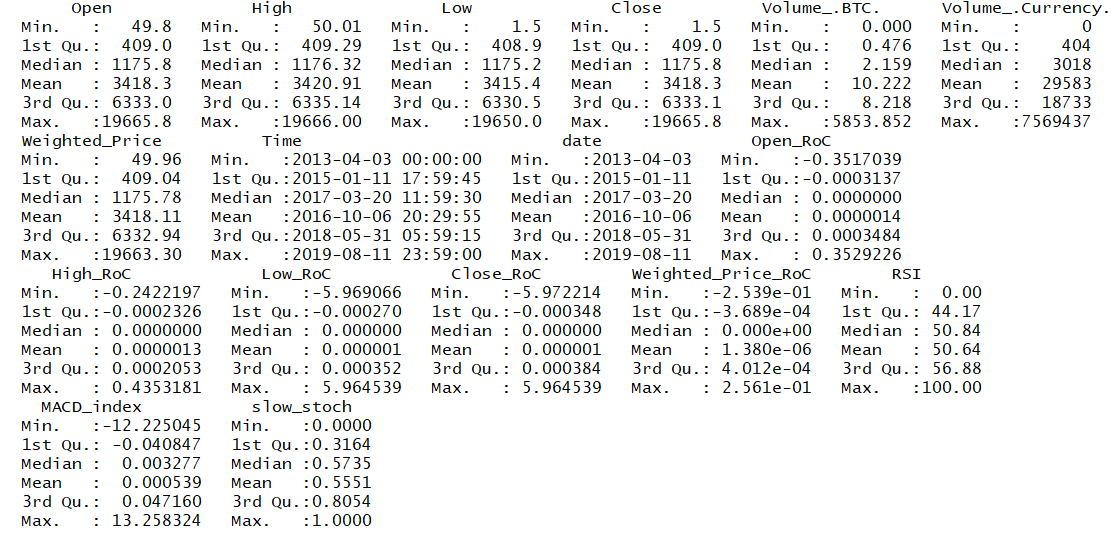


Figure 11 – Cleaned Day Trade Data, with Technical Indicators

Final size of Day Trade data is 2.498.400 observations (1735 days with 1440 observations each) and 17 variables.

# Feature Selection

## Daily Summary Data Correlation

Code Available at: https://github.com/kmaciver/Ryerson\_Capstone/tree/master/Data\_Cleaning

Analyzing variables cross correlation.

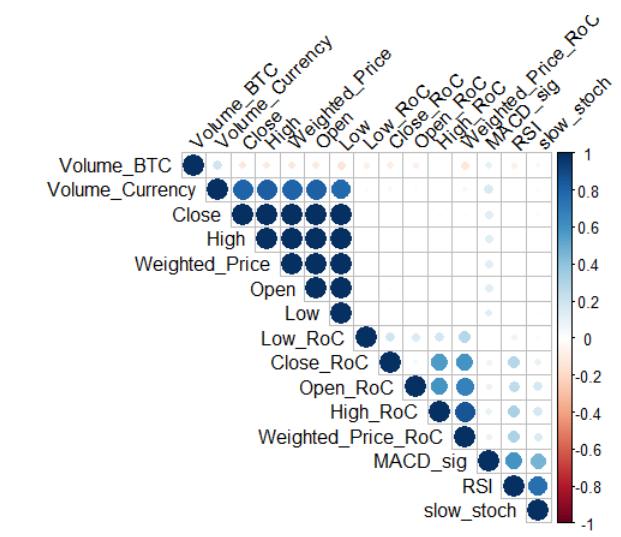


Figure 12 –Day Trade Data correlation plot

The feature Volume Currency is defined as the Volume x Weighted price, which explain the strong correlation with the OHLC and Weighted price attributes.

Slow Stoch, RSI and MACD are strongly correlated since they all have the objective of capturing trend changes.

Although the correlation plot above describes some correlation within features, for time regression analysis the key factor stands on correlation of the past values and the current time period of the label. Since forecasting is the underline objective, feature selection must take into account how the data of the previous times steps affect the prediction of the current time step.

## Daily Summary Data Auto-Correlation

Code Available at: https://github.com/kmaciver/Ryerson\_Capstone/tree/master/Data\_Cleaning

Auto-correlation function calculate how previous steps of a variable influence the current timestep. In order to apply autocorrelation, the function must first be transformed to a stationary function. When a function is stationary it revolves around a mean of zero, with no seasonality present.

For Daily Summary data, two variables will be used for later predictions: High-price and Low-price.

### Auto-Correlation – High-price Variable

Applying log transformation and lagged and iterated differences for the High variable, the data present the following format

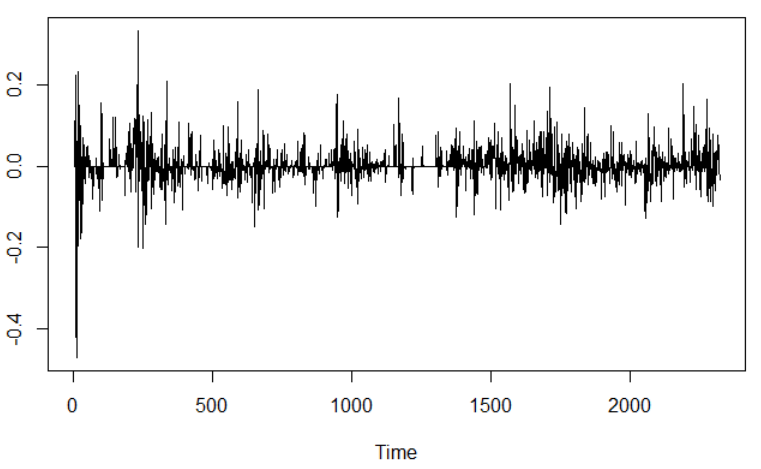


Figure 13 –Stationarity transformation applied to variable High-price

Augmented Dickey-Fuller Test for stationarity:

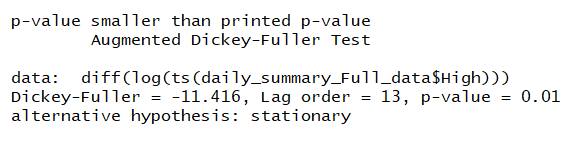


Figure 12 – Augmented Dickey-Fuller Test variable High-price

The results of the Augmented Dickey-Fuller indicates the rejection of the null hypothesis, thus the data can be considered stationary.

Plotting Auto-Correlation function to stationary data:

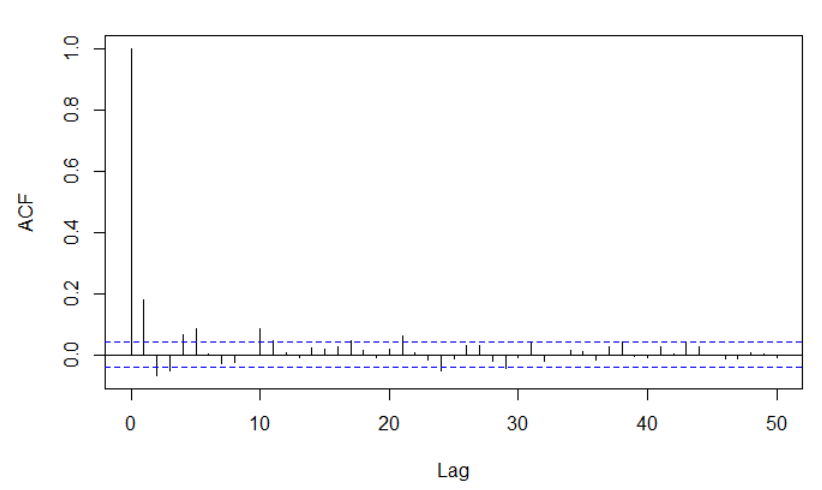


Figure 14 – ACF for Variable High-price

The ACF plot above show significant lag values at 1, 2, 3, 4, 5, 21 and 24 days, for the High-price attribute.

### Auto-Correlation – Low-price Variable

Applying the same function transformation of stationarity for the Low-price variable, and plotting the Auto-Correlation function.

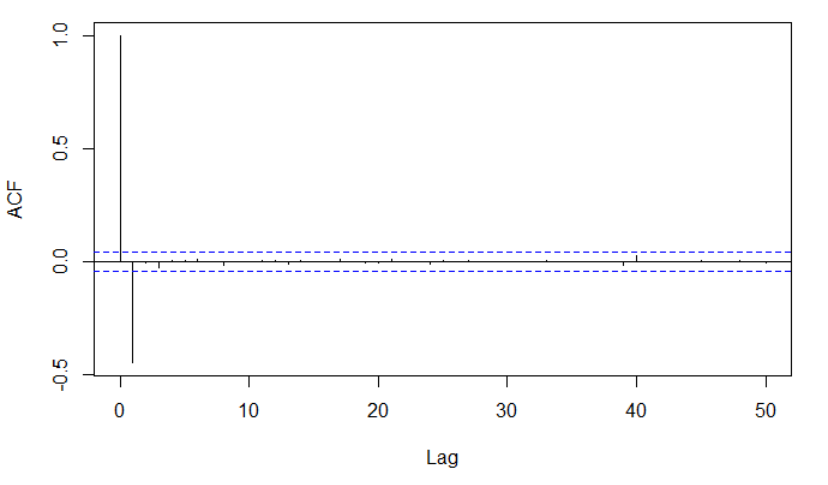


Figure 15 – ACF for Variable Low-price

The ACF plot above show significant lag value at 1 day, for the Low-price attribute.

## Daily Summary Feature Selection

Code Available at: https://github.com/kmaciver/Ryerson\_Capstone/tree/master/Feature\_Selection

Auto-correlation functions help to evaluate how far back previous time-steps are still influencing the current data, however, ACF can’t capture the cross-influence of previous time-steps of other variables.

In order to capture the importance of each feature with respect to the label, the data was transformed into a supervised learning problem, and feature importance was then calculated using gradient boosting regressor and random forest regression algorithms.

### Transforming Data for Feature Selection

First step was to exclude the Volume Currency data, since it has been previously stated that it is a function of two other variables: Volume BTC and Weighted Price.

For the remaining 14 numeric features in the data, the previous 25 steps where considered, therefore expanding the dataset from 14 to 364 columns.

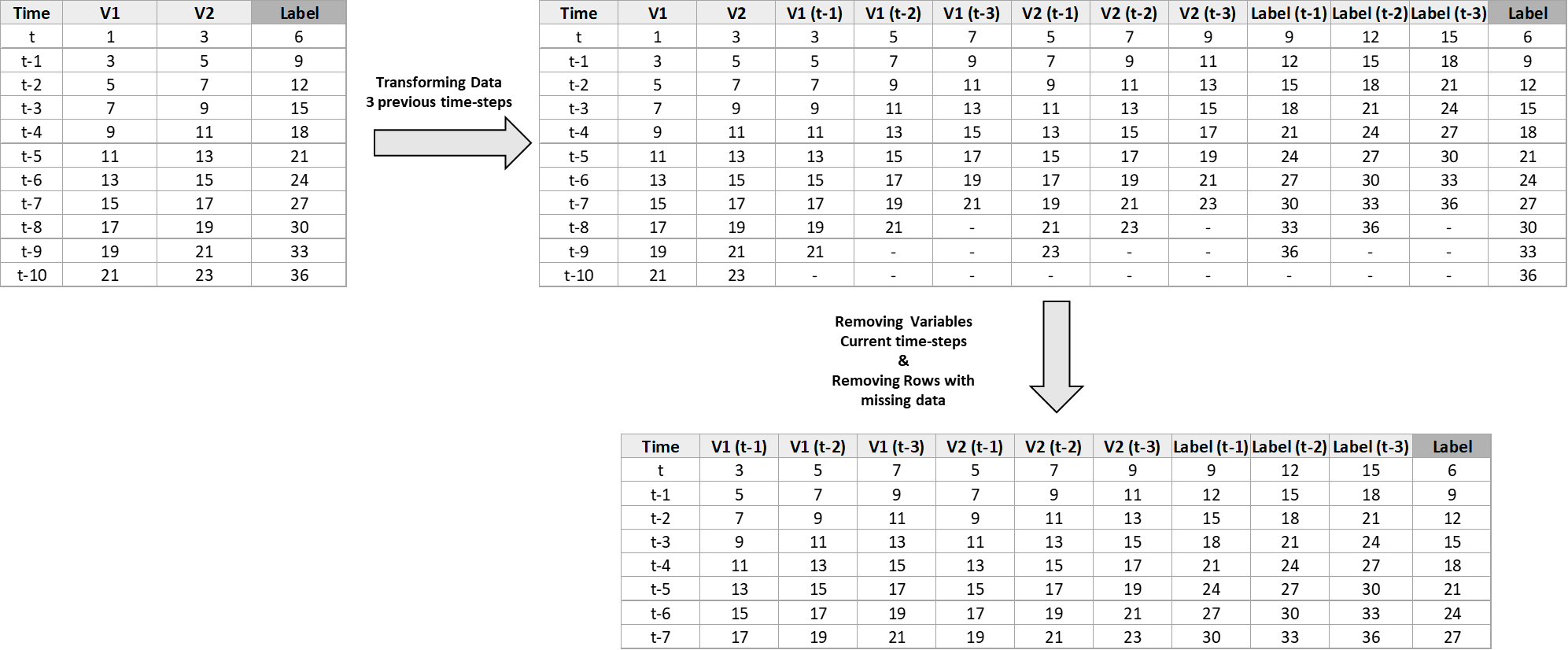


Figure 16 – Illustration of converting data into a supervised learning problem considering three previous time-steps

With the data transformed, stationarity transformations where also applied, followed be applications of random forest regressor and Gradient boosting regressor, where their respective feature importance were calculated. This approach was applied for both High-price as a label and Low-price as a label.

The results of the feature importance calculations are illustrated bellow:

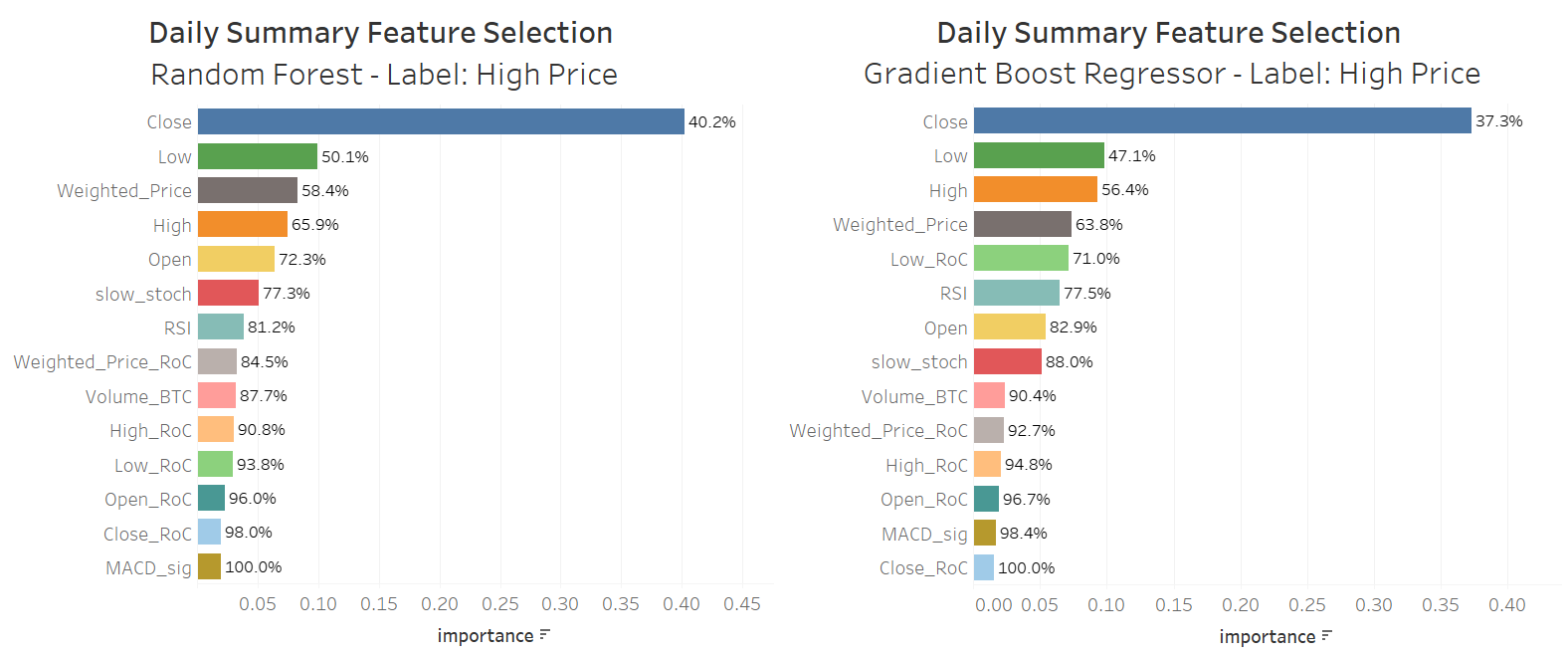


Figure 17 – Feature Importance for Daily Summary Data and High-price as Label Algorithm Comparison

Averaging the feature importance for each algorithm, the final result for High-price as a label is the following:

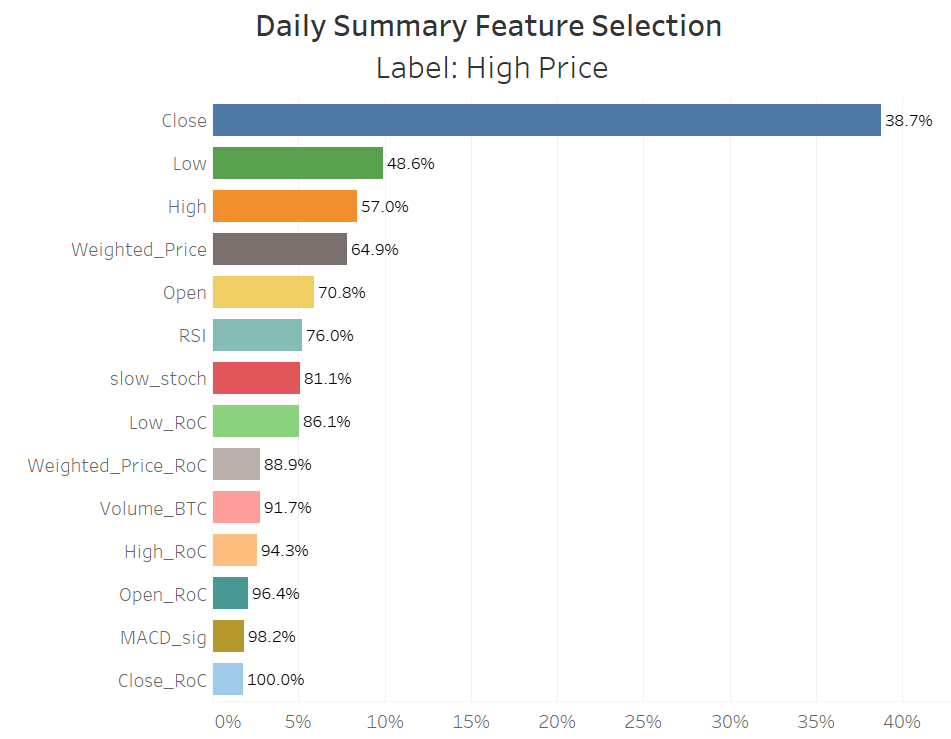


Figure 18 – Feature Importance for Daily Summary Data and High-price as Label

Applying the same approach now to Low-Price as a Label.

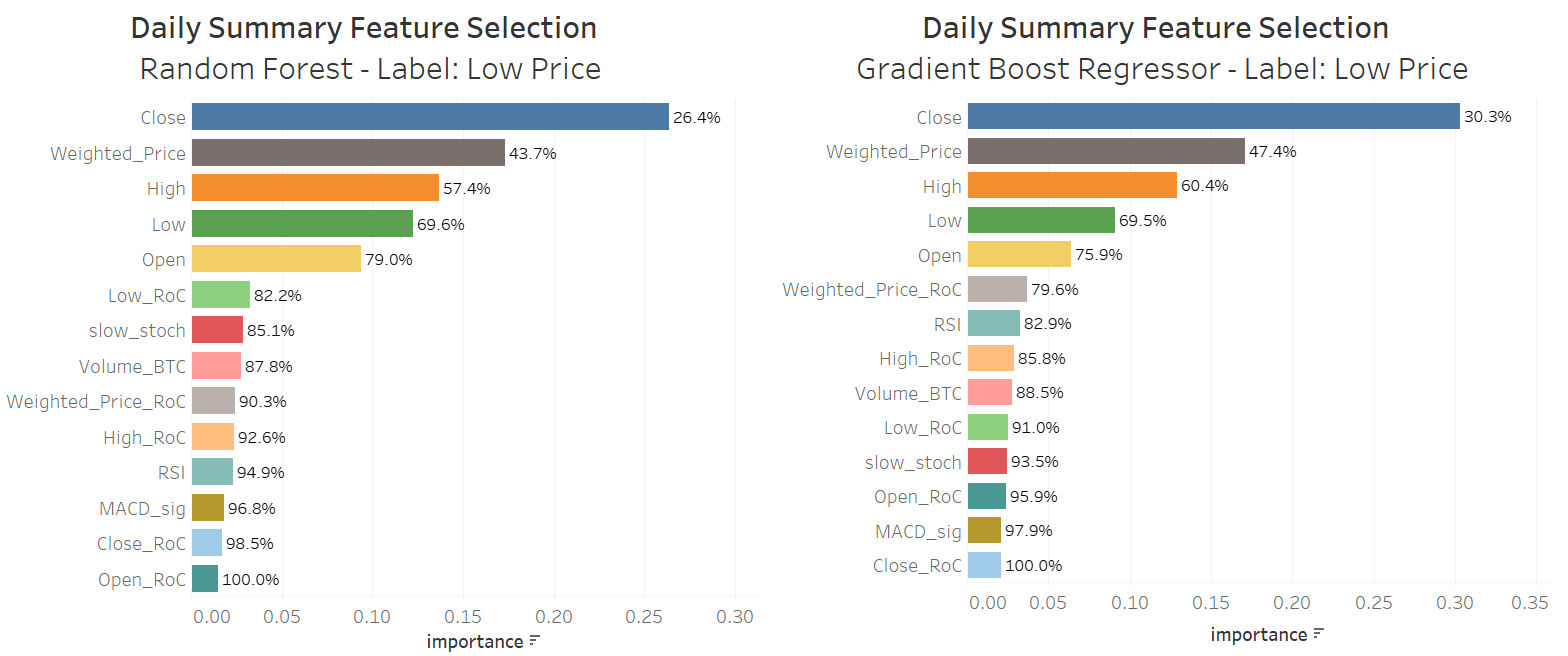


Figure 19 – Feature Importance for Daily Summary Data and Low-price as Label, Algorithm Comparison

Averaging the feature importance for each algorithm, the final result for Low-price as a label is the following:

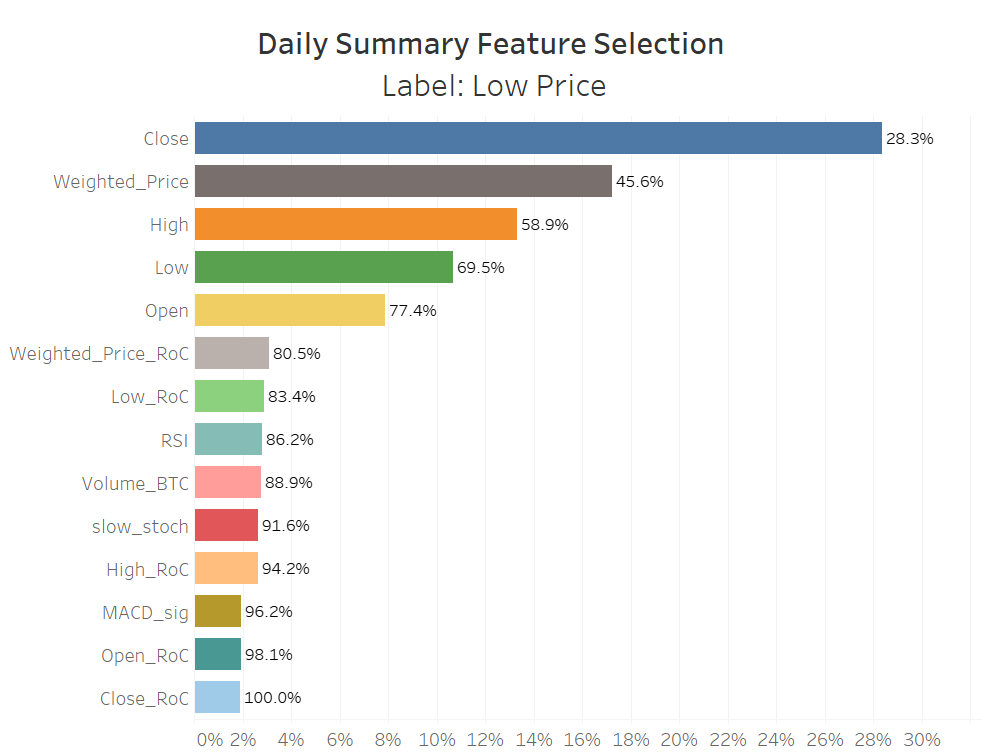


Figure 20 – Feature Importance for Daily Summary Data and Low-price as Label

A backward elimination function was applied in order to train, test and compute the mean squared error (mse) for the data for each of the labels. The function repeated the process while eliminating features one by one, from the lowest importance to highest. In order to match the feature to be eliminated in each loop, a combined rank was calculated, based on the average of the importance for both labels: High Price, and Low Price.

The combined ranking and the results for the feature elimination are presented bellow:

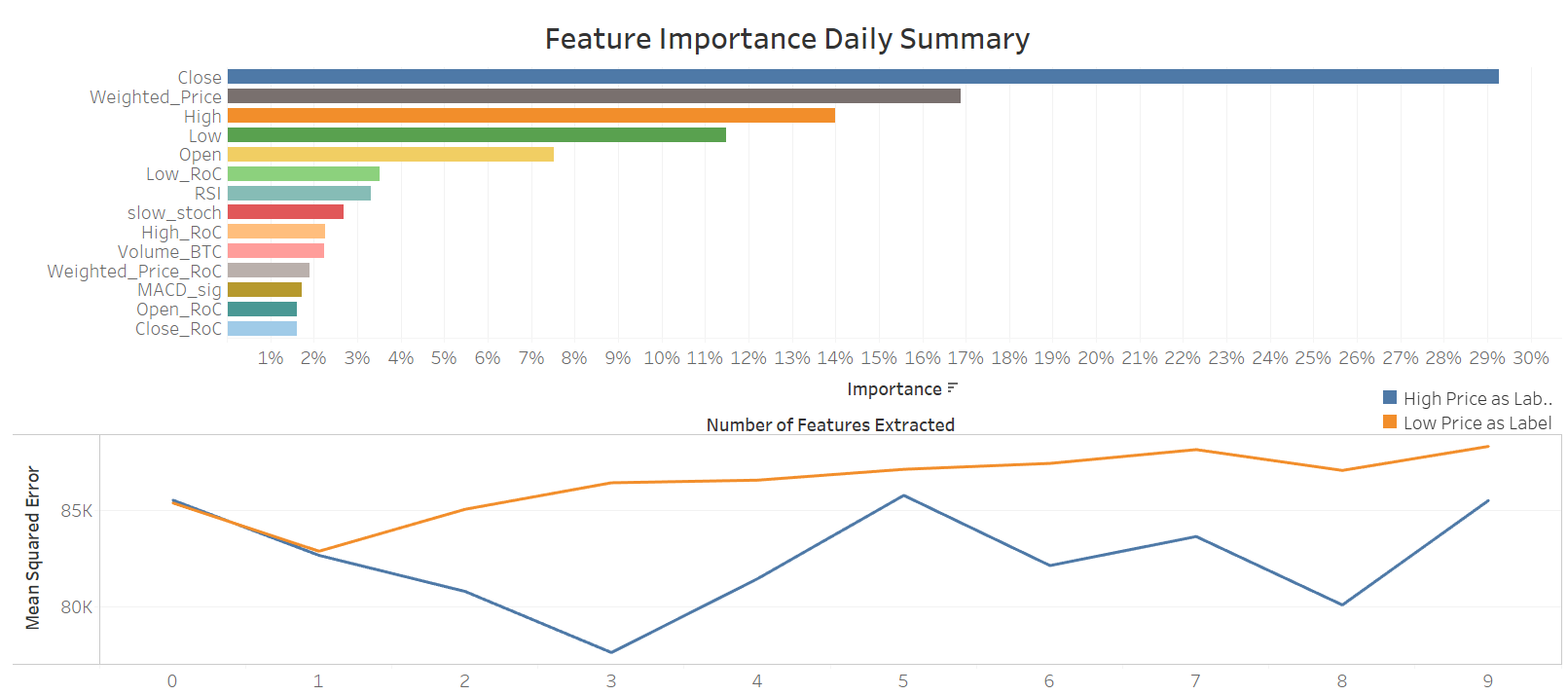


Figure 21 – Number of features to extract – Daily Summary Data

The plots above show that the best result is achieved for both High Price and Low Price labels, when the bottom feature is removed, i.e., Close\_RoC.

When, calculated feature importance with respect to time lags, the results where the following:

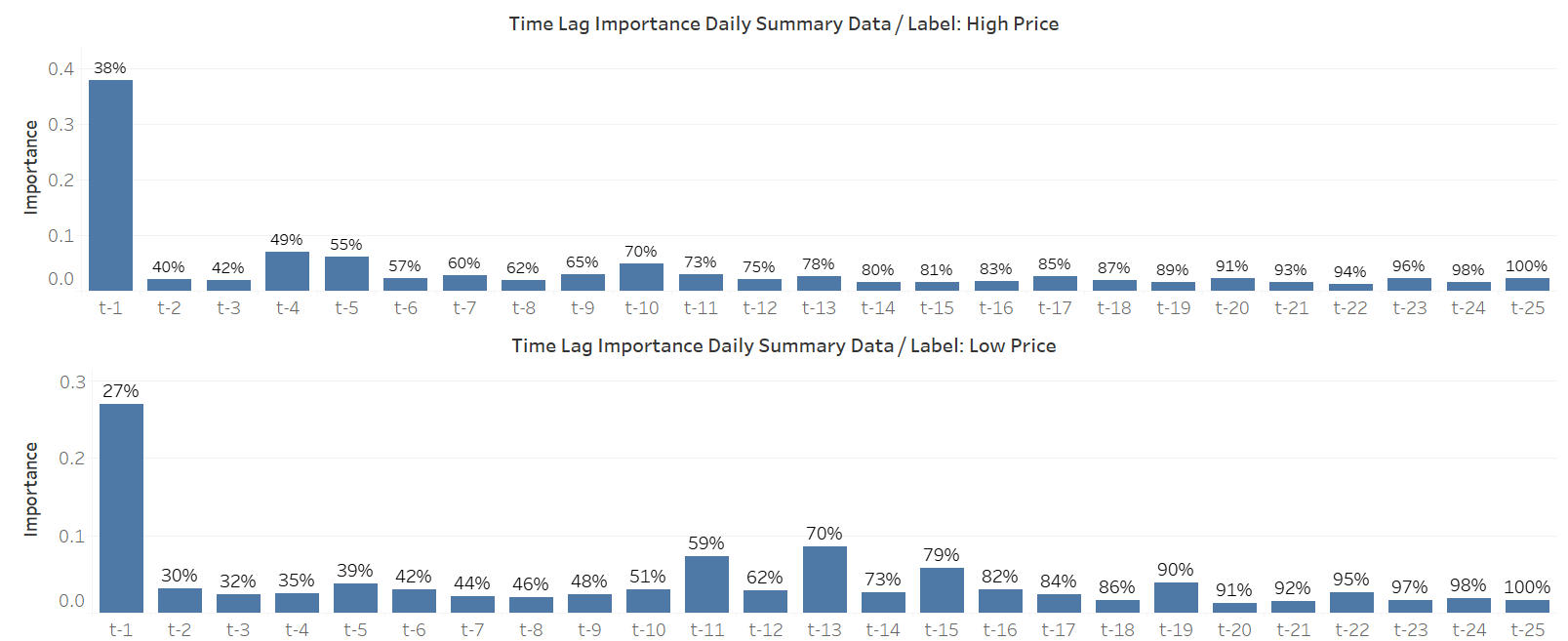


Figure 22 – Time-step Importance for Daily Summary Data for Both Labels

The plots above show that for all labels and algorithms the previous time-step has the higher importance, but is also show that up to the t-25 there is still importance being attributed.

## Day Trade Data Correlation

Code Available at: https://github.com/kmaciver/Ryerson\_Capstone/tree/master/Data\_Cleaning

Analyzing variables cross correlation.

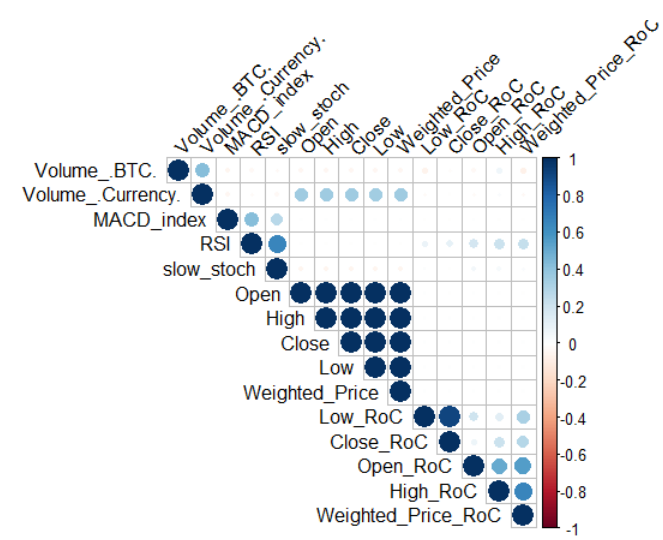


Figure 23 –Day Trade Data correlation plot

## Daily Summary Data Auto-Correlation

Code Available at: https://github.com/kmaciver/Ryerson\_Capstone/tree/master/Data\_Cleaning

The feature of interest in the Day Trade data is the Weighted Price. Differently, then Daily Summary data, the Day Trade data contains the minute to minute data for a total of 1735 days, and, therefore, the auto-correlation should be analyzed on a day-to-day basis.

### Auto-Correlation – Weighted Price Variable

Sampling random ten days of the Day Trade data, applying log transformation and lagged and iterated differences for the Weighted variable, and calculating the auto-correlation functions, the results were the following:

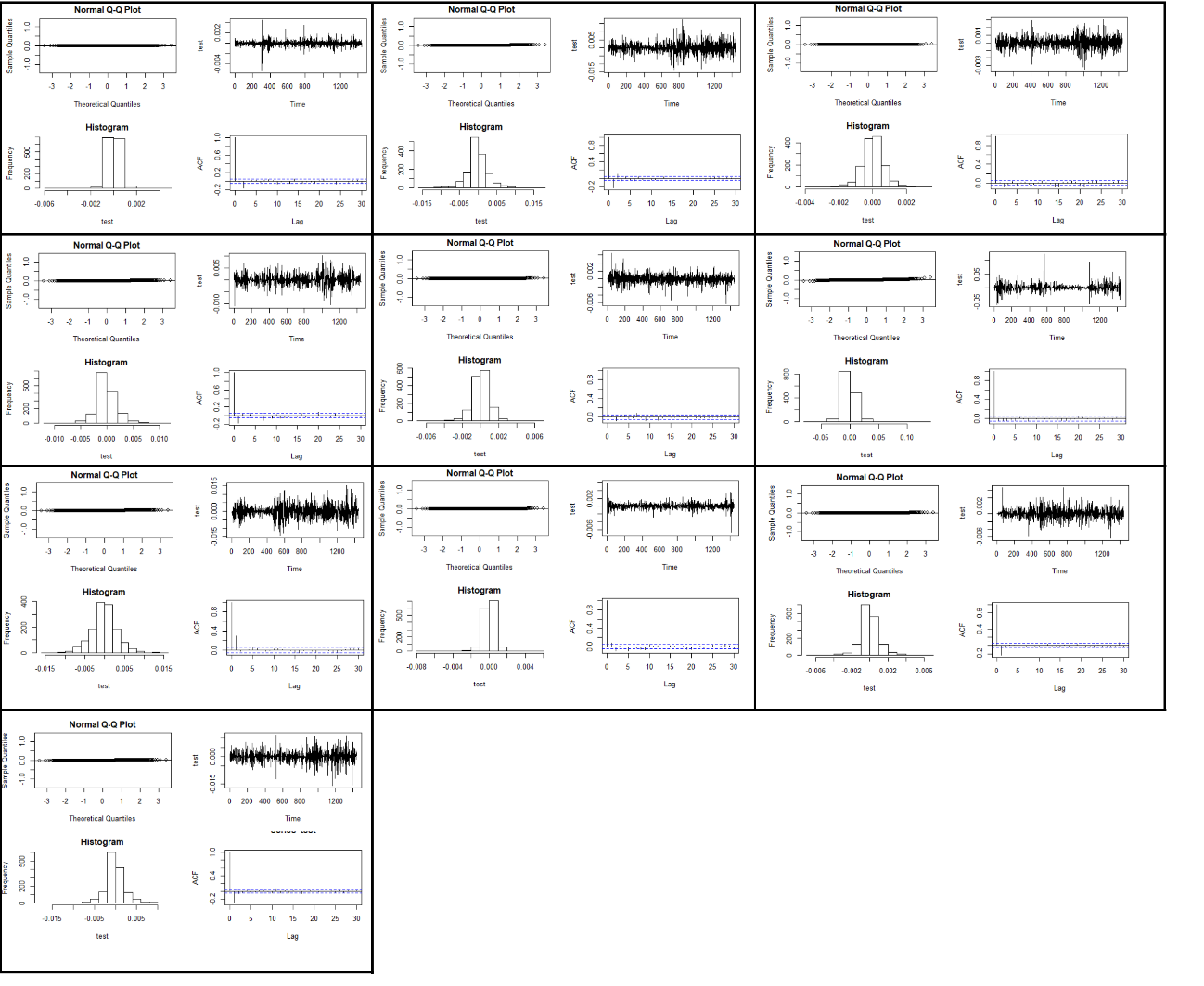


Figure 24 – 10 sampled Days ACF for Weighed Price – Day Trade Data

For all the sampled days the data stationarity was achieved by applying log and diff transformations. Although, the AFC graphs vary in the number of significant lagged time steps, it appears the furthest significant lagged value was t-25.

## Day Trade Feature Selection

Code Available at: https://github.com/kmaciver/Ryerson\_Capstone/tree/master/Feature\_Selection

As described before, Day Trade data must be analyzed in a day-to-day basis. Therefore, a similar approach to the Feature Selection step for Daily Summary was executed, to every date on the data set. The average of the feature importance was the calculated for both of the algorithms.

The results of the feature importance calculations are illustrated bellow:

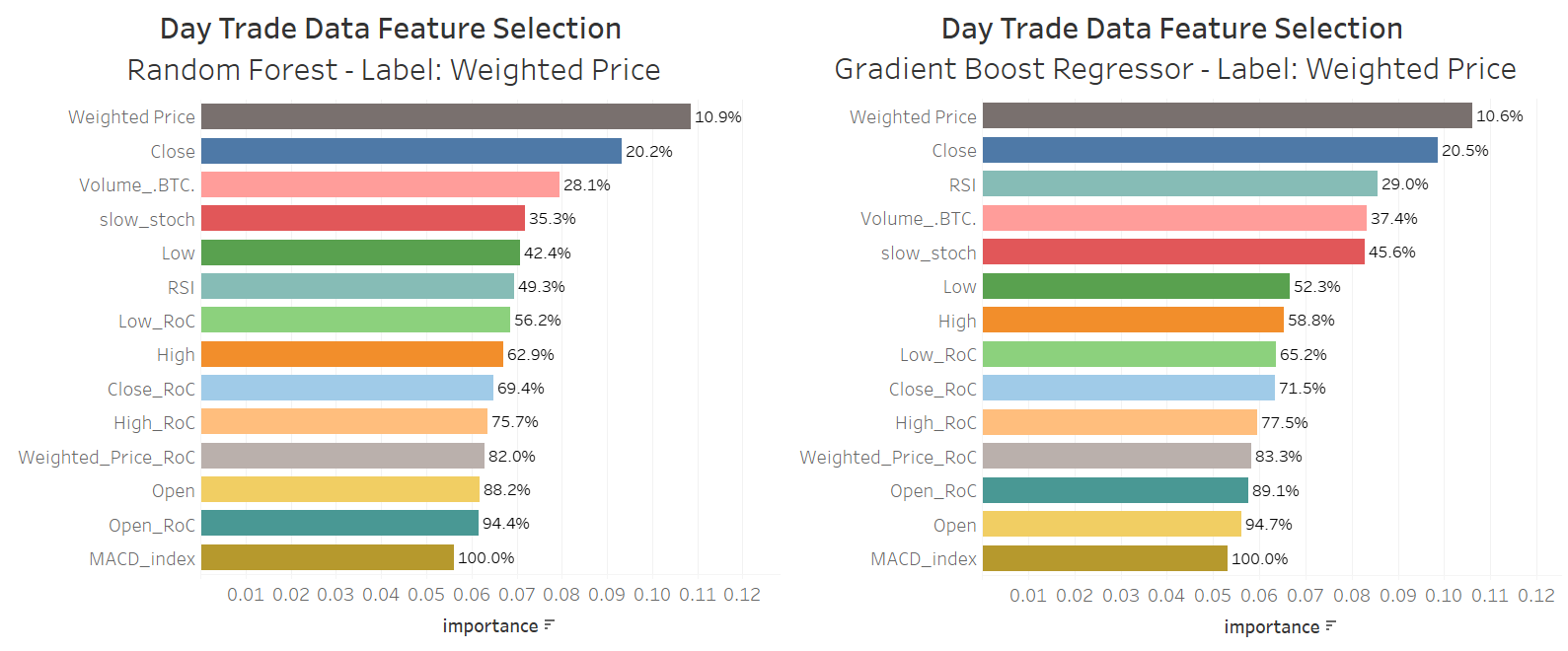


Figure 25 – Feature Importance for Day Trade Data / Label: Weighted Price / Algorithm Comparison

Averaging the feature importance for each algorithm, the final result is the following:

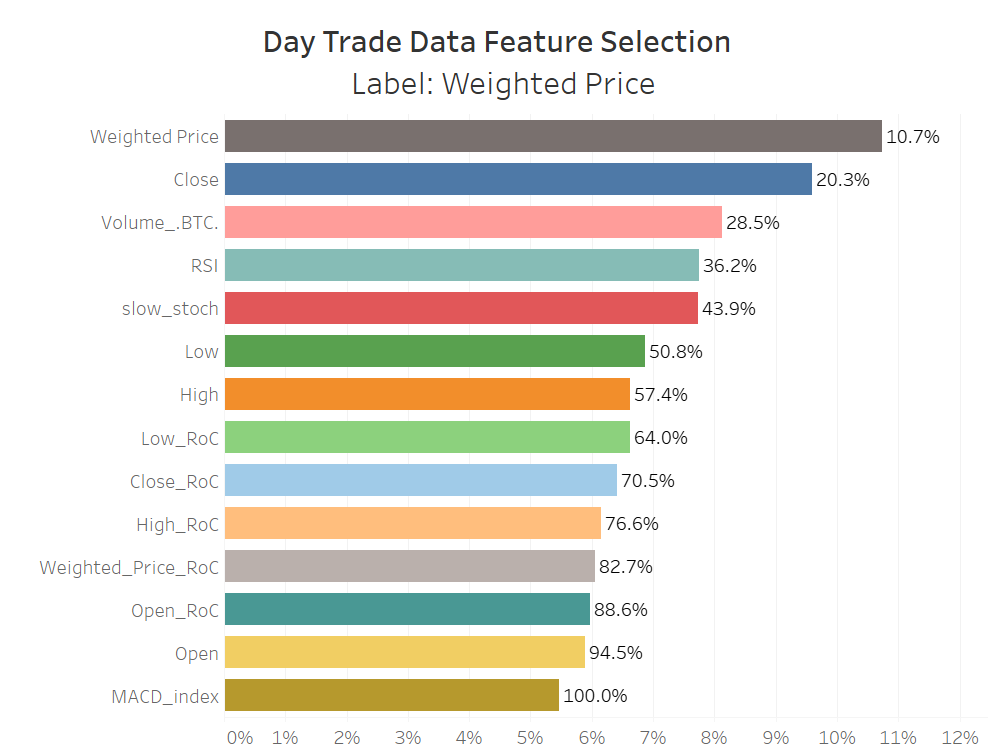


Figure 26 – Feature Importance for Day Trade Data / Label: Weighted Price

The results show that feature importance is much more balanced in the Day Trade data, with Weighted Price as a label.

Applying the same backward elimination function to each day in the data set, and averaging the mse throughout the days, the results were the following:

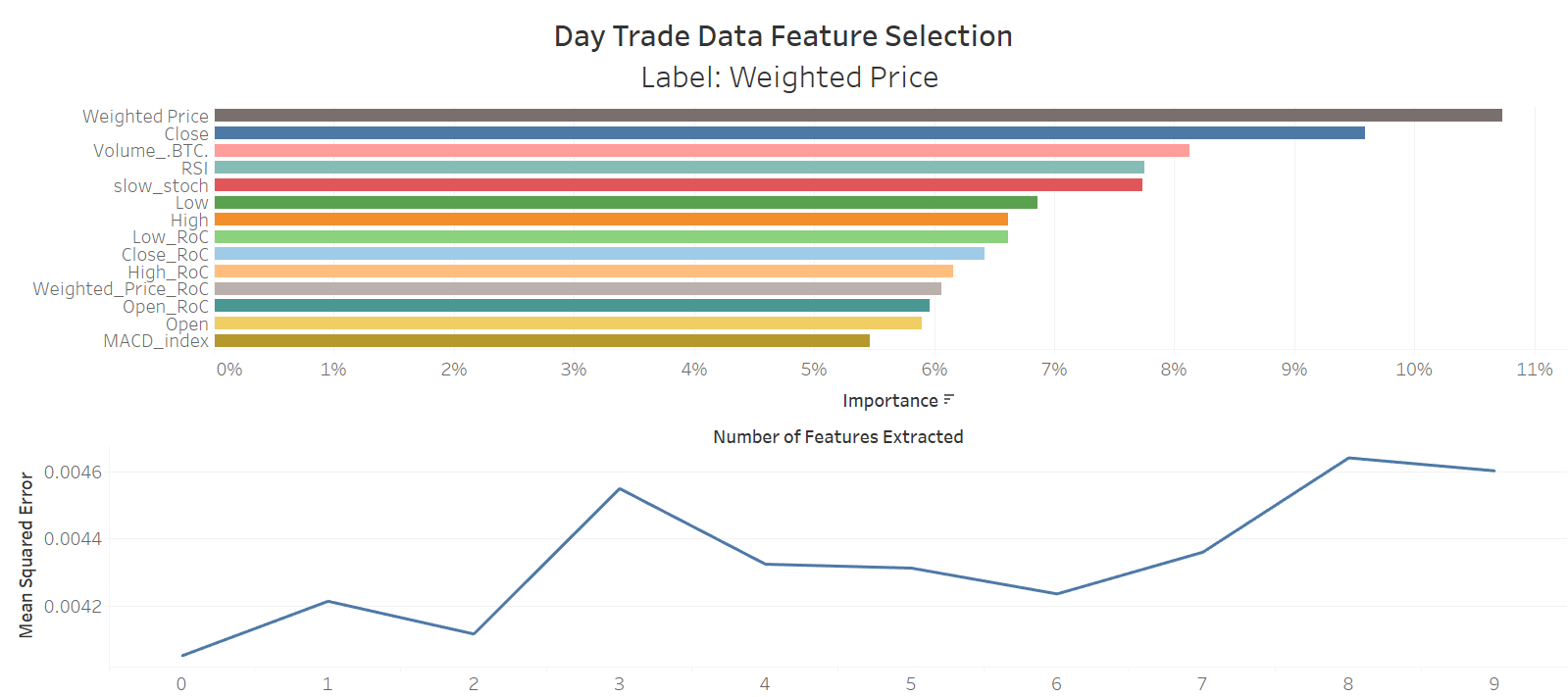


Figure 27 – Number of features to extract – Day Trade Data

The plots above show that the best result is achieved when no feature is extracted.

When, calculated feature importance with respect to time lags, the results where the following:

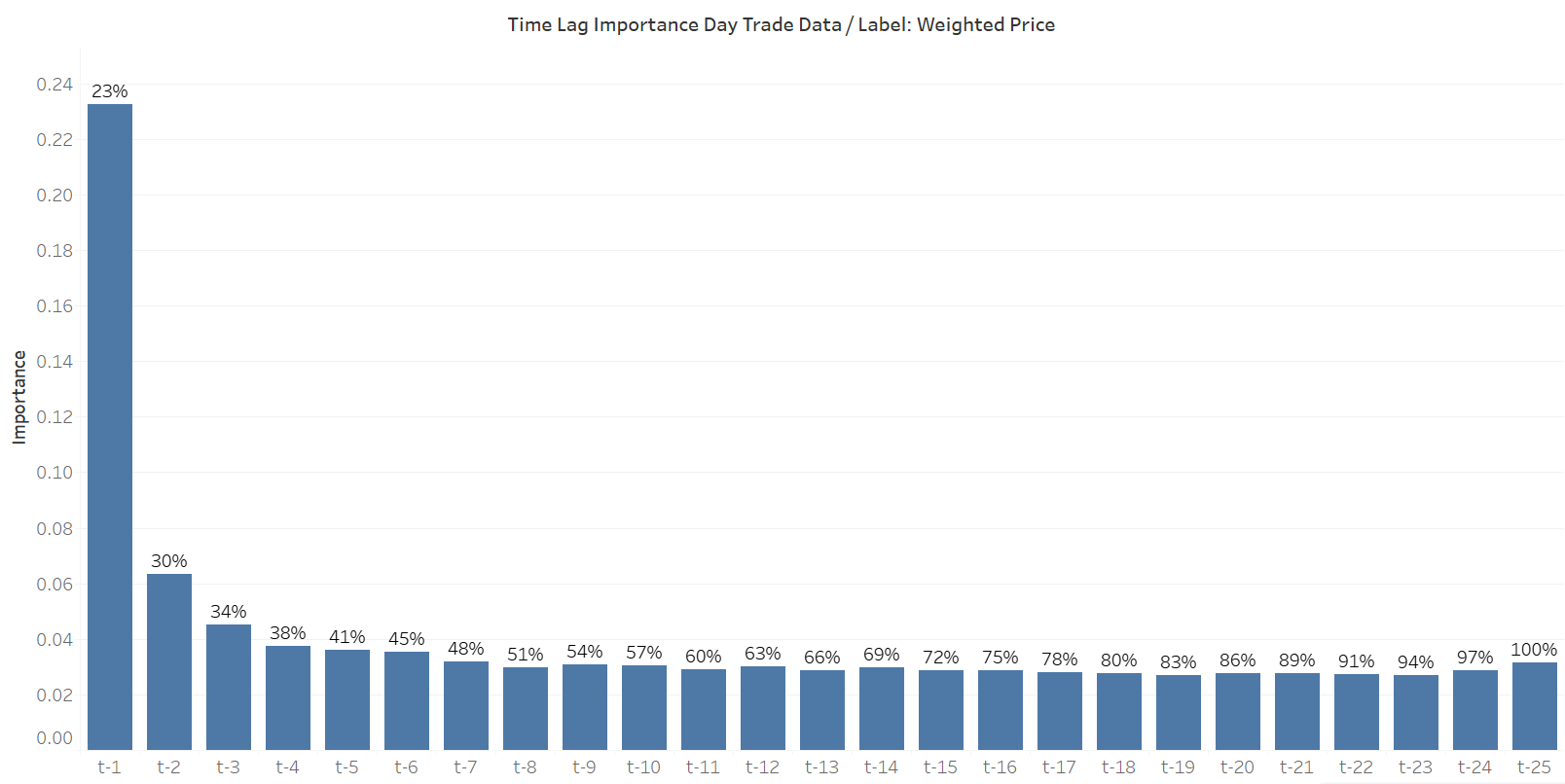


Figure 28 – Time-step Importance for Day Trade Data

The plots above show that for all labels and algorithms the previous time-step has the higher importance, but is also show that up to the t-25 there is still importance being attributed.

The plots above show that for all labels and algorithms the previous time-step has the higher importance, but is also show that up to the t-25 there is still importance being attributed.

# Approach

As mention in the introduction, the objective of this project is to verify to what extent can Neural Networks help forecast the next minutes of Bitcoin prices within a day fluctuation. The approach for this project consists in evaluating three algorithms:

1. Non-Neural Algorithm
2. Neural Network (LSTM) using Day Trade data as input
3. Neural Network (LSTM) using Day Trade data and Daily Summary data as input

The block diagram bellow illustrates the approach planned.

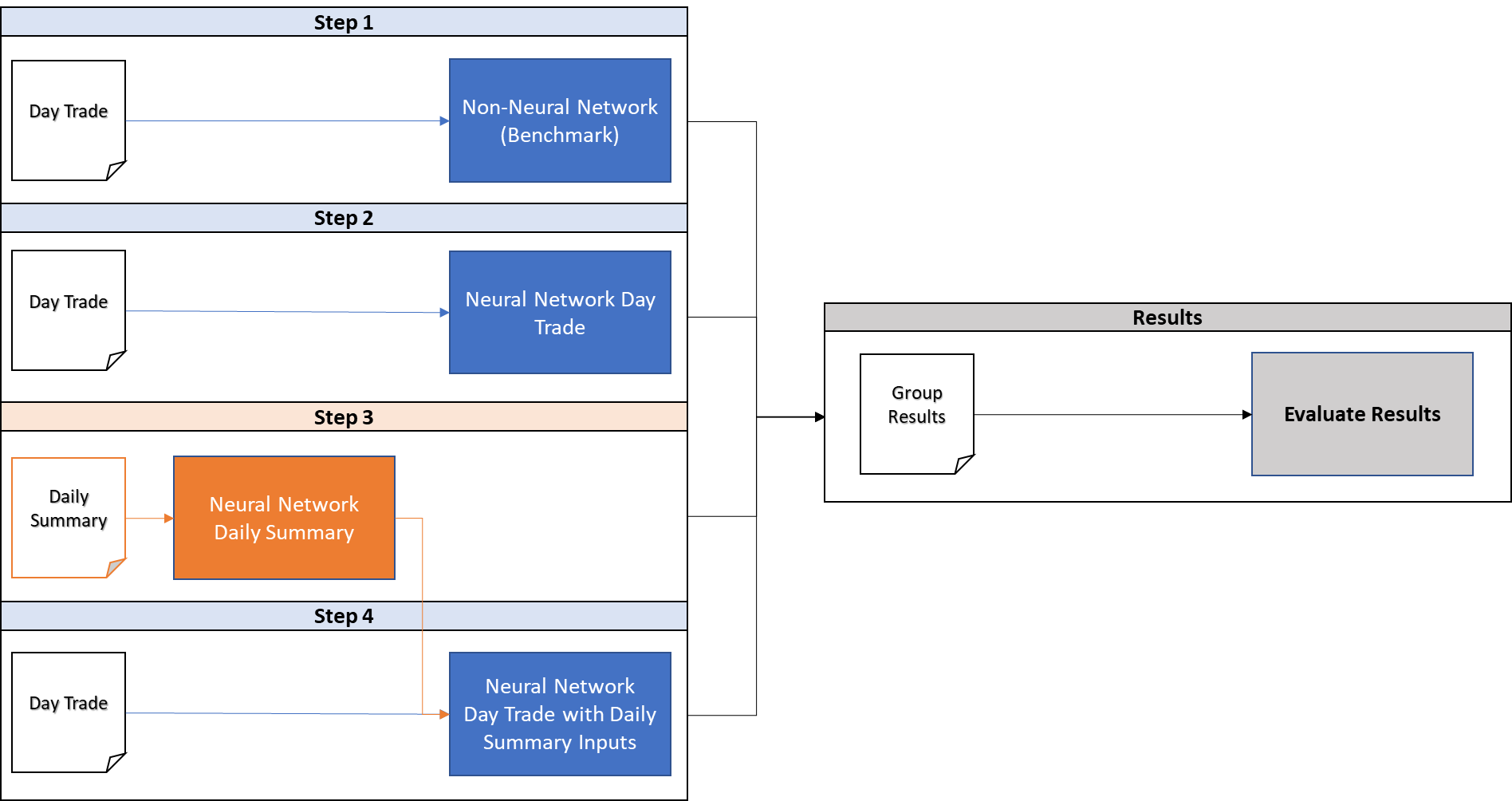


Figure 29 –Approach Block Diagram

## 5.1 Step 1: Non-Neural Network (Benchmark)

Code Available at: https://github.com/kmaciver/Ryerson\_Capstone/tree/master/Approach/Step1-Benchmark

For this step a non-Neural algorithm is chosen to generate predictions of the Day Trade data. The objective is to compare these results with the Neural Networks', in order to justify, or not, the application of Neural Networks for this particular problem.

For this step, any regression type algorithm can be applied. Gradient boosting regressor was the selected algorithm.

The input data consists of 1735 days divided in 1440 minutes, resulting in 2.498.400 rows. The data also contains 16 columns, where 14 will effectively be used for prediction.

The first step into enabling the use of regression algorithm for this problem is to transform the current data in supervised learning type of problem. To achieve this, each row of the data must contain the information of the current time step and previous twenty-five-time steps. By trying to perform this transformation the number of columns drastically increase to 364 columns.

### Scalability of Non-Neural Network Algorithm

While trying to apply the above-mentioned transformation, the memory required to read and modify the data increased to a point that surpassed the available RAM memory of the device. Several cloud services were also tested: Google Colab; Kagle; Databricks Community Edition. Non of them proved to have enough RAM to execute the task.

This RAM problem is one evidence to why regressor algorithms, such as Gradient Boost Regressors, do not scale for large data sets. Since batch learning is not possible, all data must be available for the calculations, which eventually becomes an impediment due to memory un-availability.

## 5.2 Step 2: Neural Network Day Trade data

Code Available at: https://github.com/kmaciver/Ryerson\_Capstone/tree/master/FinalModel/Step2-DayTrade

For this step a Neural Network algorithm is used to try to predict the Weighted Price of bitcoin 10 minutes in the future inside a particular day. The input data is the Day Trade data from the previous approach. The Neural Network to be applied must have the characteristic to retain the information of the previous steps, for that Recurrent Neural Networks (RNN), Gated Recurrent Unit (GRU), and Long-Short Term Memory (LSTM) are possible candidates. For this project LSTMs where selected in order to avoid minimize problems with vanishing gradients.

First, the data has the column Volume Currency removed as a result from the feature selection part of this project. The data is then split with 90% of the days for training (1561 days) and the remaining for testing (174 days).

The data is also transformed in batches of 25 minutes per day and trained to predict the 10th minute ahead.

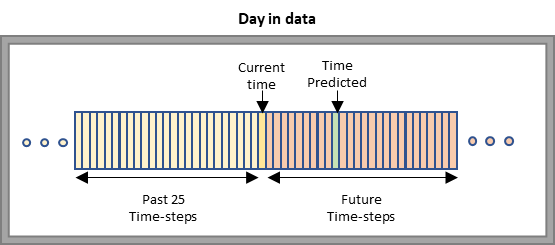


Figure 30 –Illustration of the data inputs and outputs connections

Before training the data must also be transformed into arrays and normalize. Normalization is a data preprocessing technique that transforms data with attributes of different units into a known uniform scale. This is so that none of the attribute values dominate over others, comparison and aggregation of attributes become easier and the data become better conditioned for convergence (Gupta, et al., 2019). For this report the MinMaxScaler was used for normalizing the data.

For generating the batches for training a technique called sliding windows was used. There are several variations to how sliding windows can be applied. In this project two variations were used, one for training and one for testing. The images bellow illustrates how both techniques work.

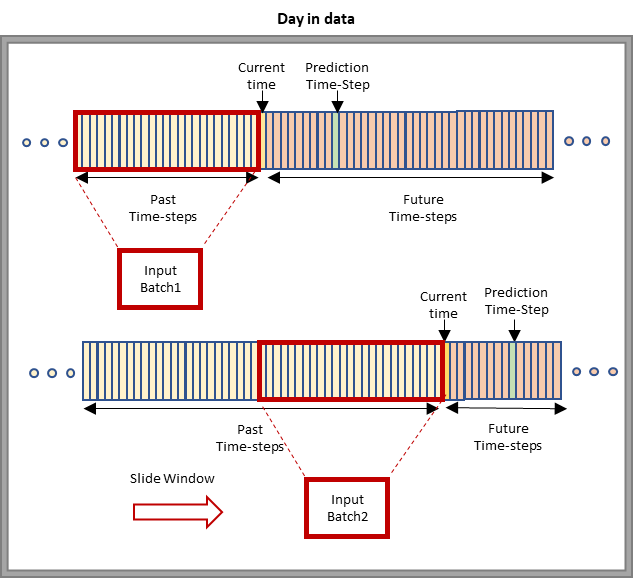


Figure 31 –Sliding Window technique applied for Training set

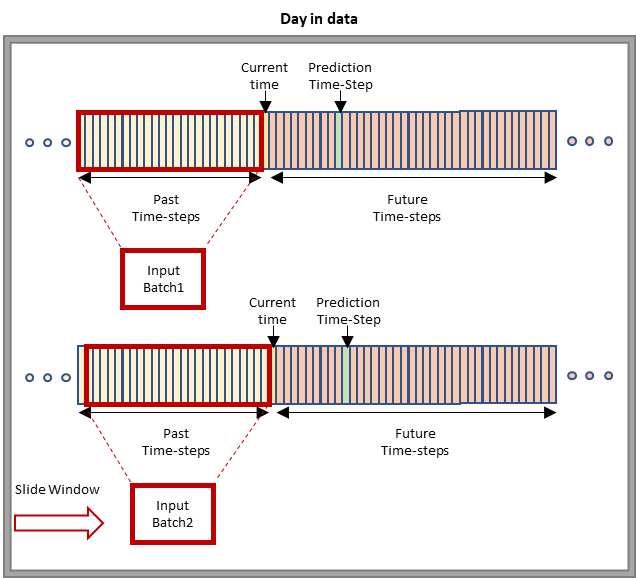


Figure 31 –Sliding Window technique 2 applied for Testing set

As the images above illustrate, the difference between the techniques applied to the training and to testing sets is the number of time-steps the sliding window rolls over for each batch. For the training set the number of time-steps is the same as the batch input, this reduces over-fitting problems.

The model was trained with the following structure:

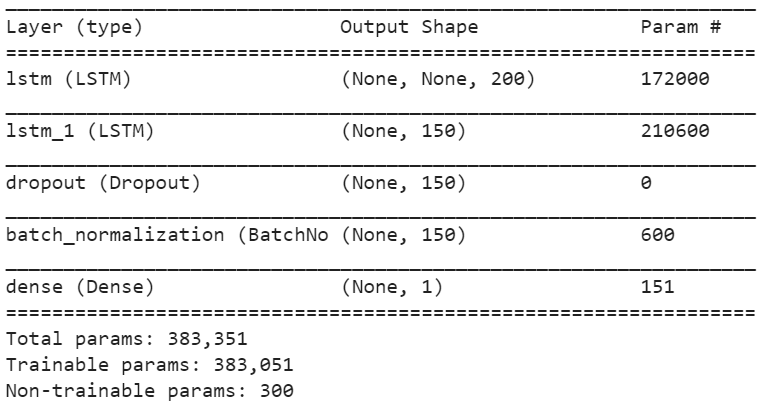


Figure 32 –Neural Network Summary for Day Trade Predictions

The selected activation function for the Dense layer was the Linear function. The selected optimizer was Adam, and the loss function was Logarithm of the hyperbolic cosine of the prediction error (logcosh), which works mostly like the mean squared error, but will not be so strongly affected by the occasional wildly incorrect prediction.

## 5.3 Step 3: Neural Network Daily Summary

Code Available at: https://github.com/kmaciver/Ryerson\_Capstone/tree/master/FinalModel/Step3-DailySummary

For this step a Neural Network algorithm is used to try to predict the next day High and Low-Price values of bitcoin. The input data is the Daily Summary data that consists of 2322 days (rows) and 15 features.

First, the data has the columns Volume Currency and Open RoC removed as a result from the feature selection part of this project. The data is then split with 90% of the days for training (2088 days) and the remaining for testing (233 days).

The data was then transformed into array and scaled using MinMax Scaler. Following the same sliding window technique applied in step 2 for the training data, the data was reshaped into batches of 25 days as inputs.

The model was trained with the following structure:

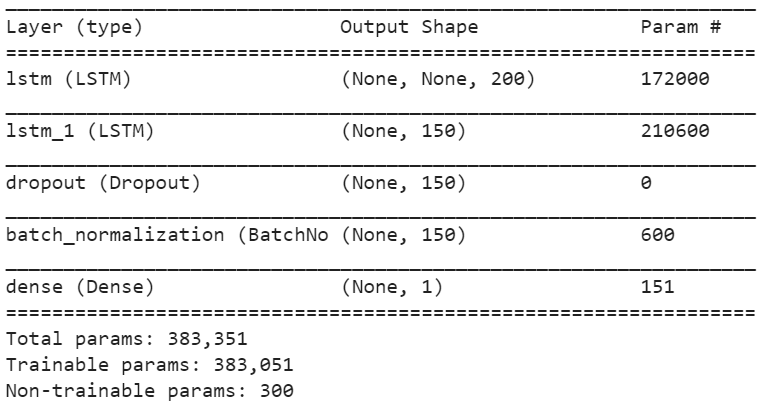


Figure 33 –Neural Network Summary for Daily Summary Predictions

The selected activation function for the Dense layer was the Linear function. The selected optimizer was Adam, and the loss function was Logarithm of the hyperbolic cosine of the prediction error (logcosh), which works mostly like the mean squared error, but will not be so strongly affected by the occasional wildly incorrect prediction.

## 5.4 Step 4: Neural Network Day Trade with Daily Summary Inputs

Code Available at: https://github.com/kmaciver/Ryerson\_Capstone/tree/master/FinalModel/Step4-DayTrade\_DailySummaryInputs

For this step a Neural Network algorithm is used to try to predict the Weighted Price of bitcoin 10 minutes in the future inside a particular day. As in step 2, the input data is the Day Trade data the difference here is that two columns are added corresponding to the High and Low-Price predictions of each day made by the Neural Network described on step 3. The premise is that by providing a prediction of what the highest and lowest values of the current day are going to be, the Neural Network can learn to predict the weighted price within those boundaries.

The image bellow illustrates the concept of how the two Neural Networks interact and the desire result of this step.

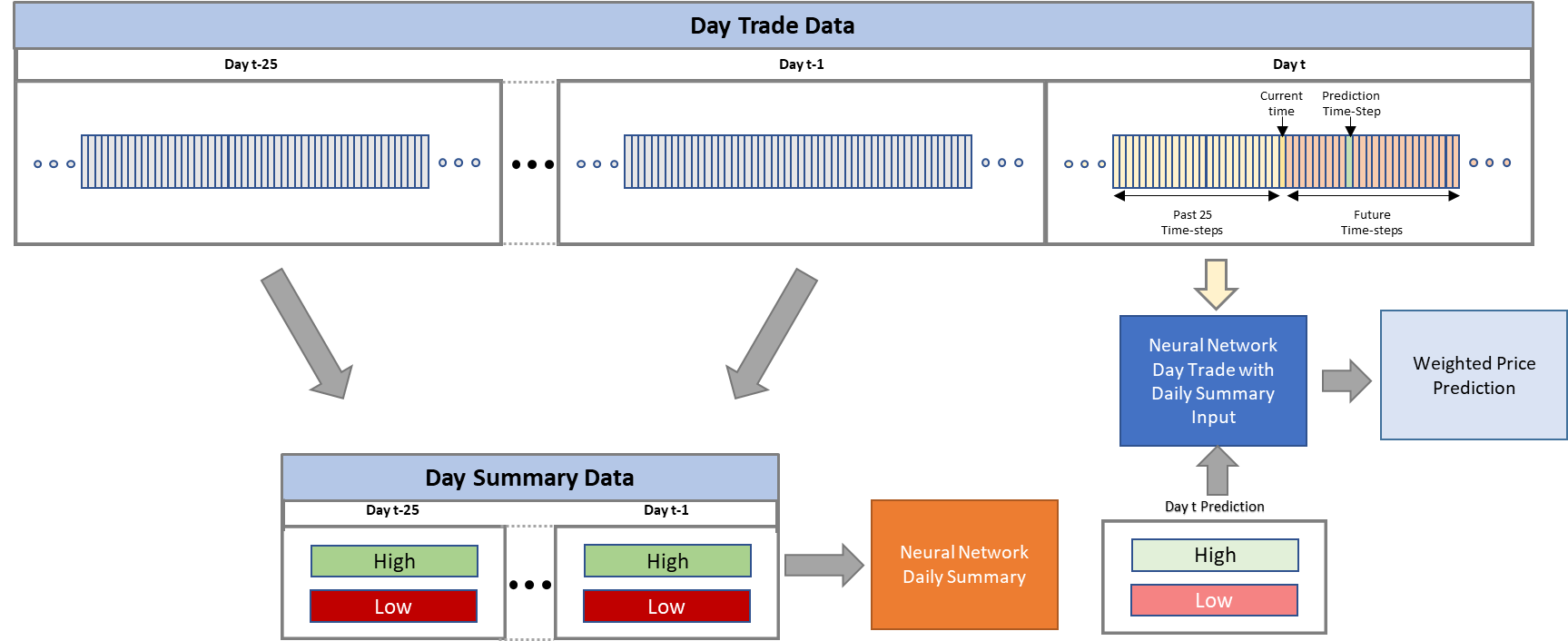


Figure 34 –Interaction between Daily Summary Neural Network and Day Trade with Daily Summary Input Neural Network

The model was trained with the following structure:

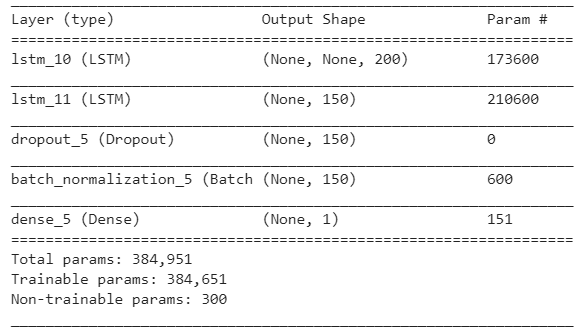


Figure 35 – Neural Network for Day Trade with Daily Summary input

The selected activation function for the Dense layer was the Linear function. The selected optimizer was Adam, and the loss function was Logarithm of the hyperbolic cosine of the prediction error (logcosh).

# Results

After training the models the results were the following:

## 6.1 Neural Network Day Trade data - Results

The Neural Network was trained with the following parameters:

* Batch Size for training = 50
* Dropout = 0.4
* Optimizer – Adam (Lr=1e-3)
* Reduce Learning rate on Plateau (factor=0.8, patience=4, min\_lr=1e-4)
* Number of Epochs = 100
* Steps per Epoch = 50

The image bellow shows the training and Validation loss along the epochs:

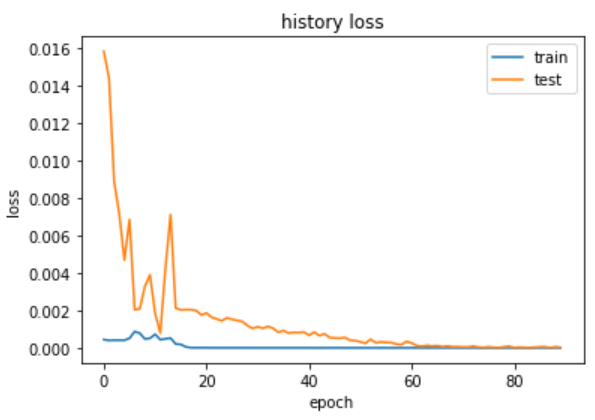


Figure 35 – Training and Validation Loss history

The image above shows a gradual descent of the validation loss.

The predictions based on the train model are the following:

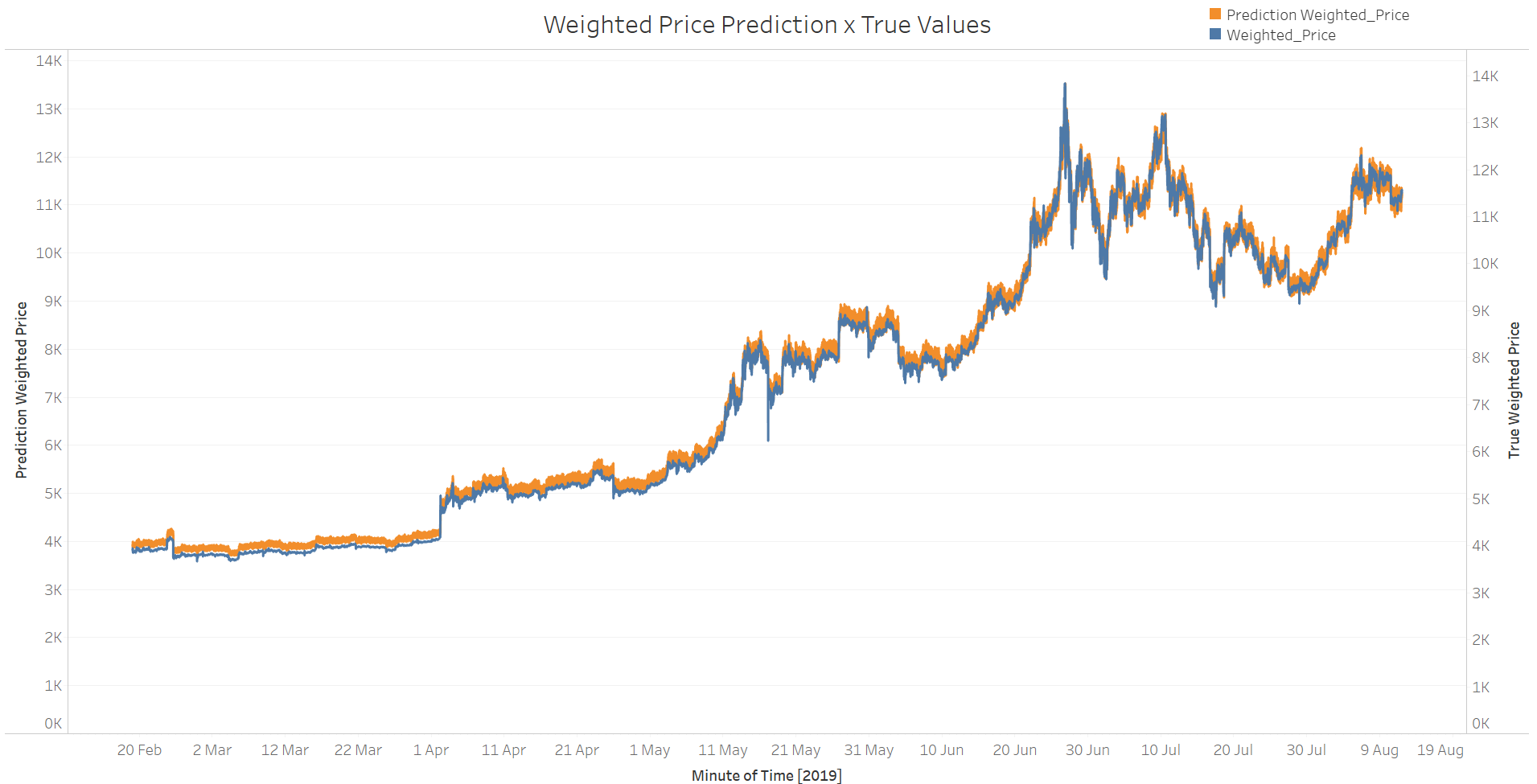


Figure 36 –Overall Weighted Price Prediction x True Values

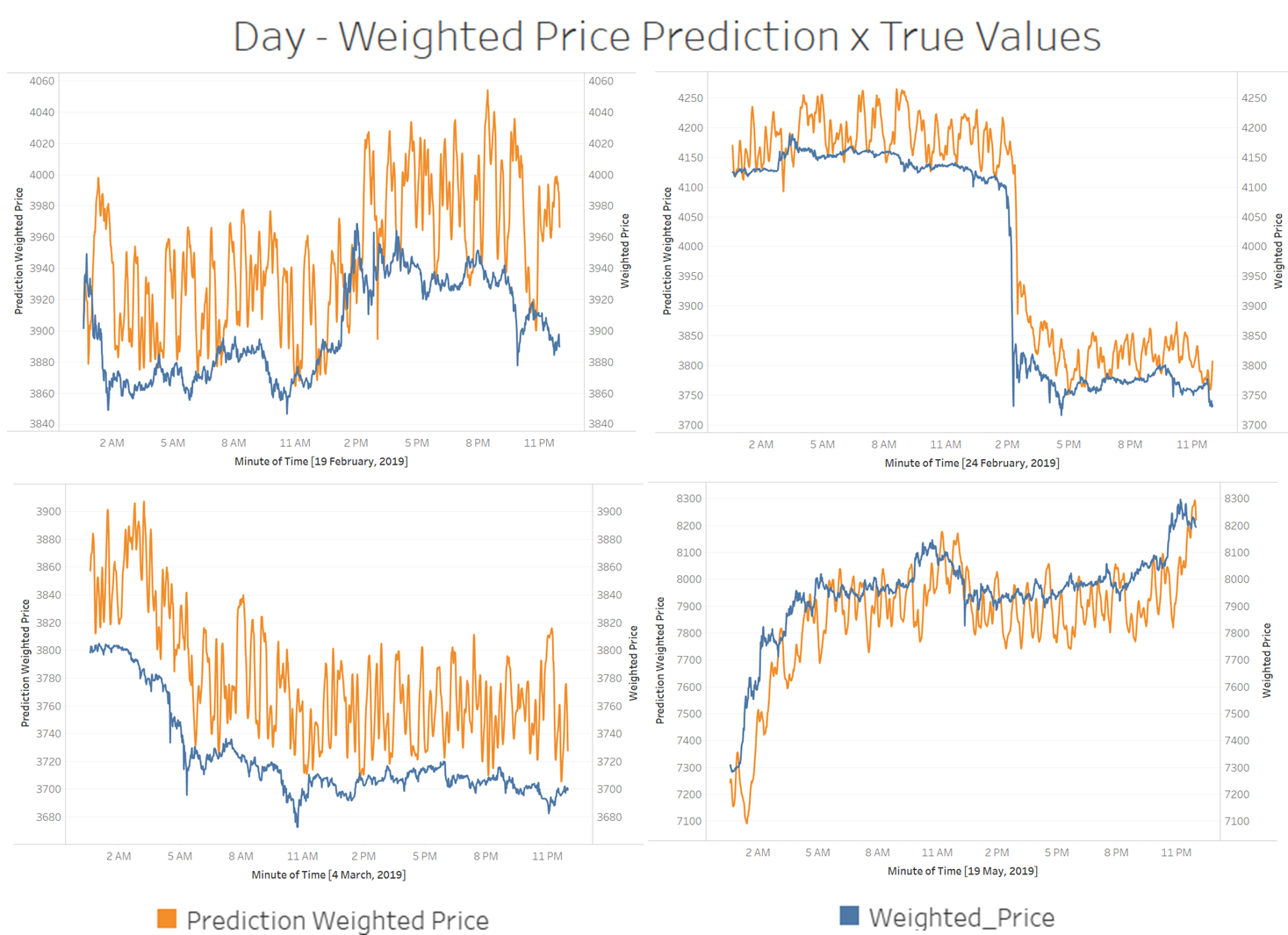


Figure 37 – Sampled days - Weighted Price Prediction x True Values

The images show the model’s results appear to be noise and with high variance. Although it seems to capture high market fluctuation. The table below show the results for Mean Squared Error (MSE) and Mean Absolute Error (MAE) for a total of 10 iterations.



Table 4 – Result Table for MSE and MAE step 2

## 6.2 Neural Network Daily Summary data - Results

The Neural Network was trained with the following parameters:

* Batch Size for training = 60
* Dropout = 0.4
* Optimizer = Adam (Lr=1e-3)
* Reduce Learning rate on Plateau (factor=0.2, patience=4, min\_lr=1e-4)
* Number of Epochs = 100
* Steps per Epoch = 50

The image bellow shows the training and Validation loss along the epochs:

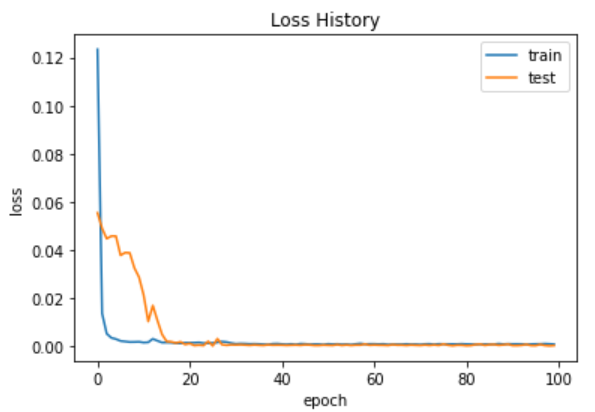


Figure 38– Training and Validation Loss history

The predictions based on the train model are the following:

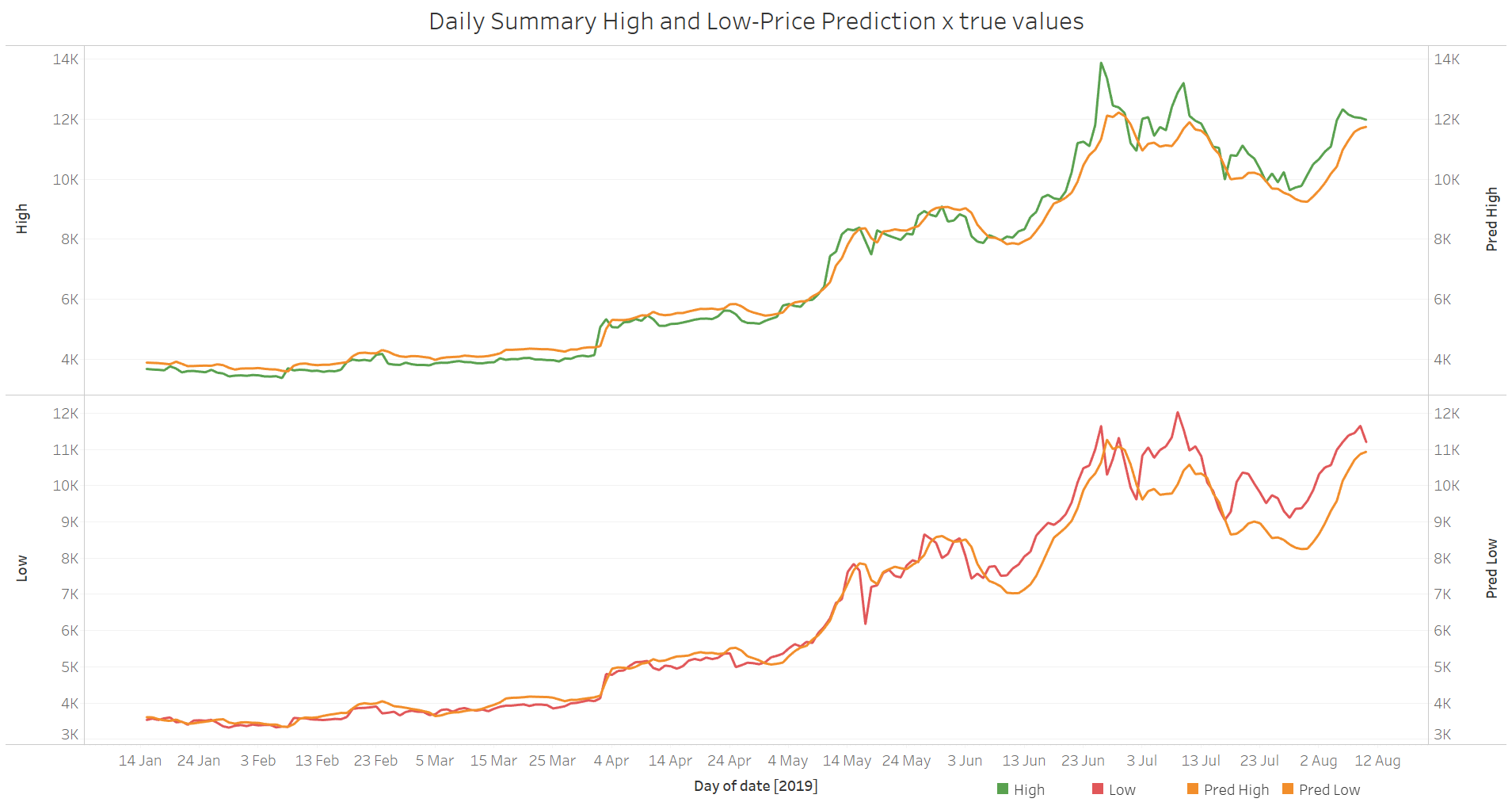


Figure 39 – High and Low-Price prediction x High-Low Price of Validation data

The image shows the model was able, to some extent, predict market fluctuation. The Mean Squared Error for High and Low-Price predictions are 243.646 and 314.755 respectively.

## 6.3 Neural Network Day Trade with Daily Summary Inputs - Results

The Neural Network was trained with the following parameters:

* Batch Size for training = 50
* Dropout = 0.4
* Optimizer = Adam (Lr=1e-3)
* Reduce Learning rate on Plateau (factor=0.8, patience=4, min\_lr=1e-4)
* Number of Epochs = 100
* Steps per Epoch = 50

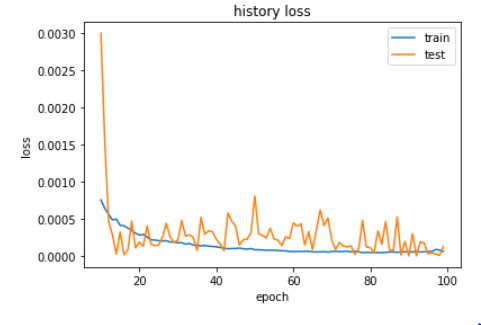


Figure 40– Training and Validation Loss history

The predictions based on the train model are the following:

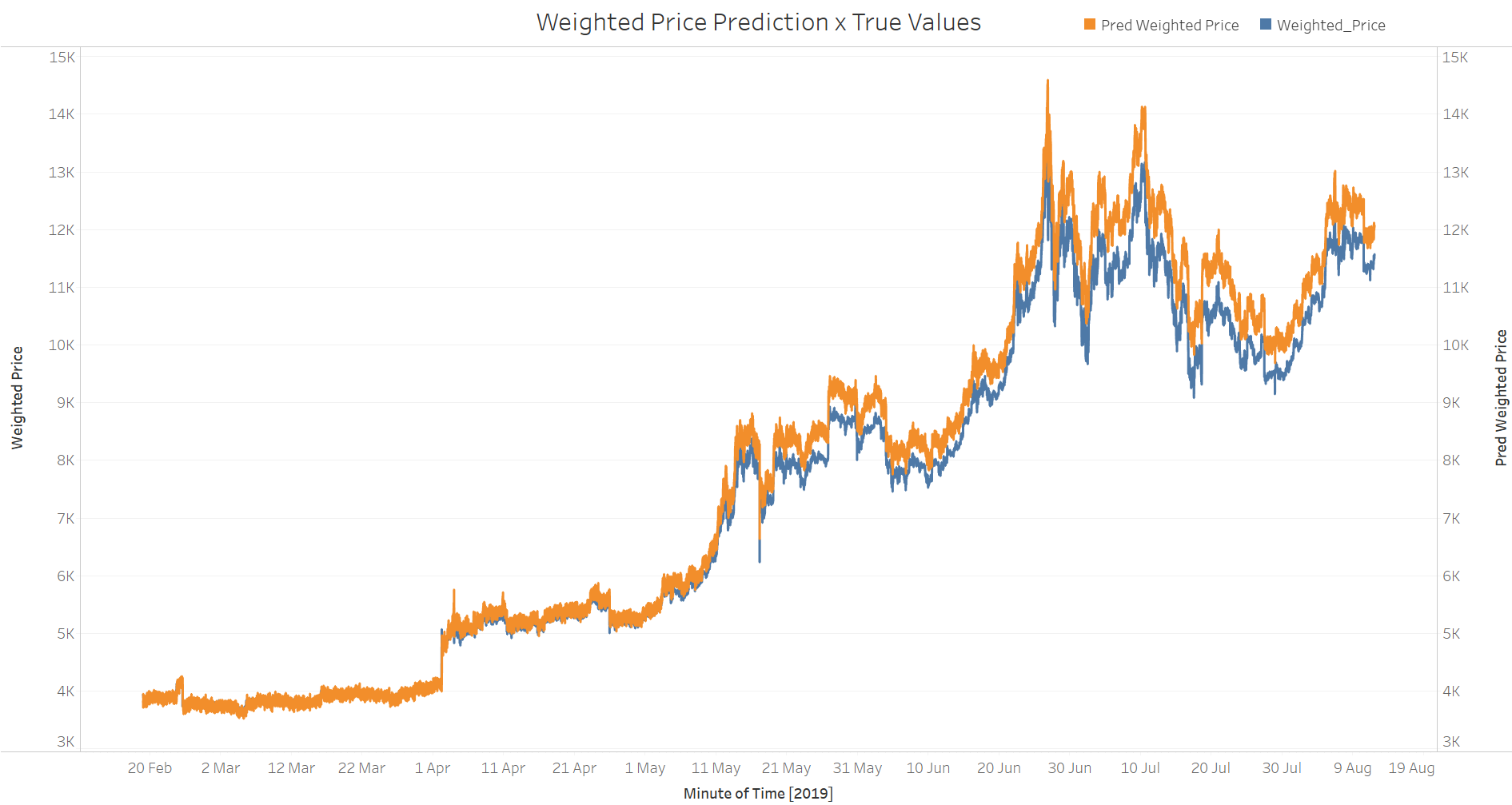


Figure 41 –Overall Weighted Price Prediction x True Values

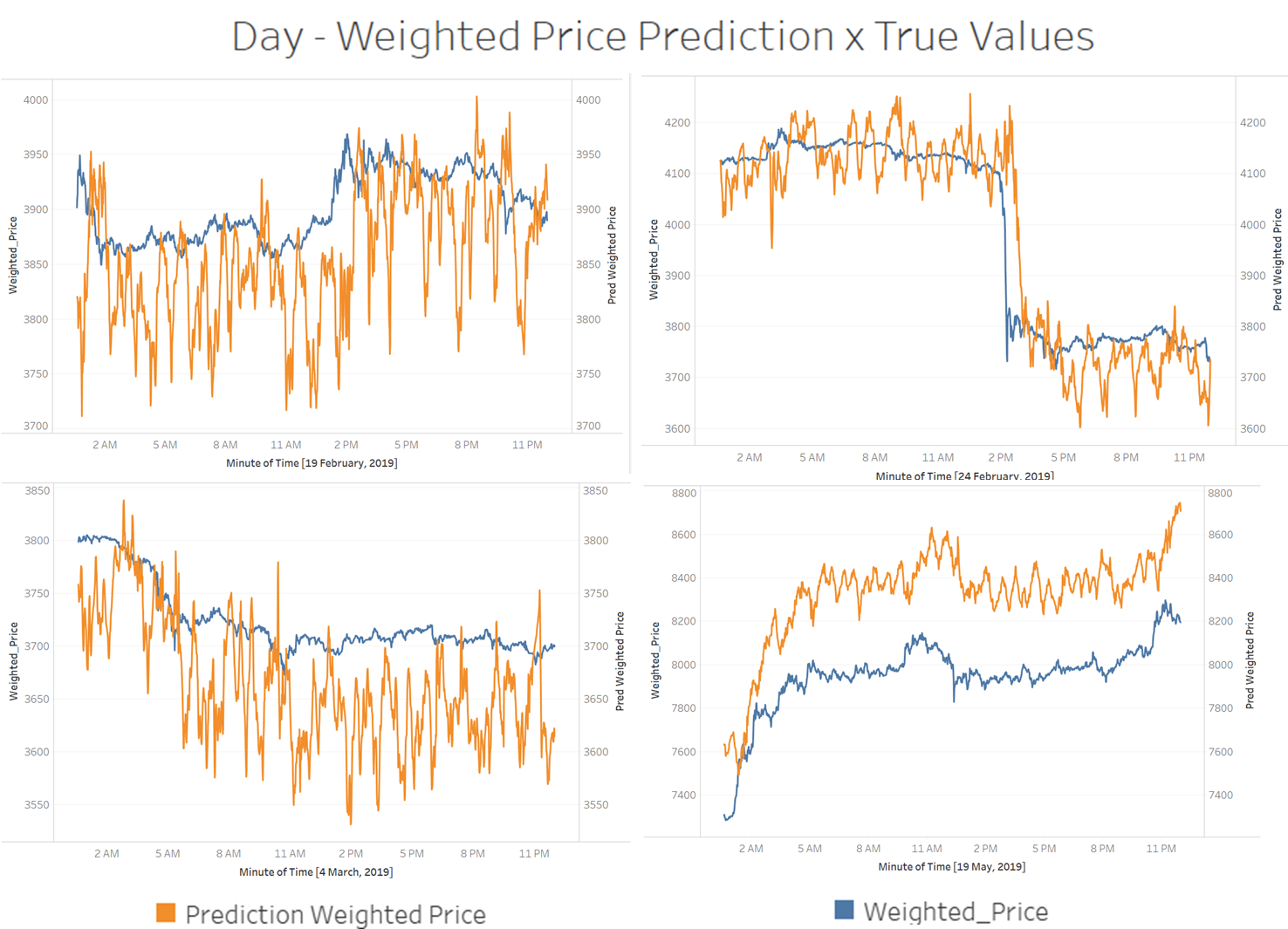


Figure 42 – Sampled days - Weighted Price Prediction x True Values

As oppose to what initially expected, the input from the Daily Summary High and Low-Price prediction appeared to have shifted the overall predictions to be higher than the real values (Figure 39). The images on Figure 40 show the model’s results continued to be noise and with high variance. The table below show the results for Mean Squared Error (MSE) and Mean Absolute Error (MAE) for a total of 10 iterations.



Table 5 – Result Table for MSE and MAE step 4

## 6.4 Evaluating Step 2 and Step 4

Plotting the boxplot distributions from the results from tables 4 and 5 for Mean Squared Error and Mean Absolute Error.

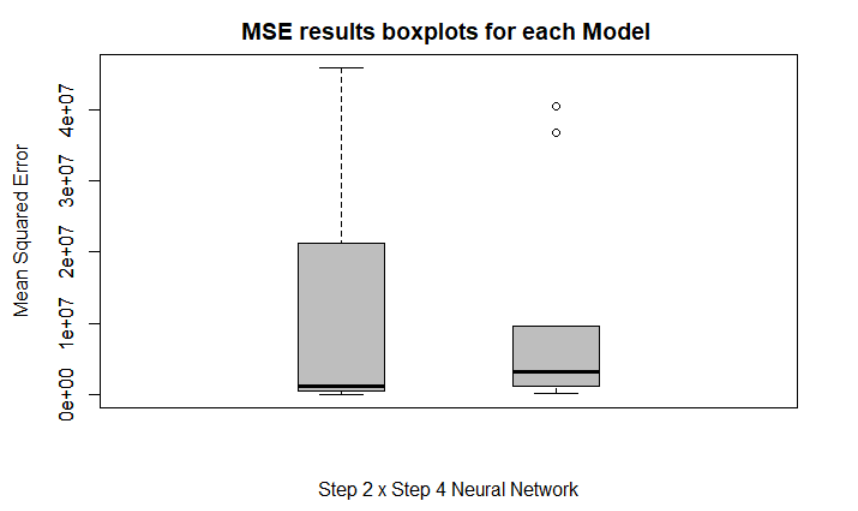


Figure 43 – Boxplot Distribution for MSE results

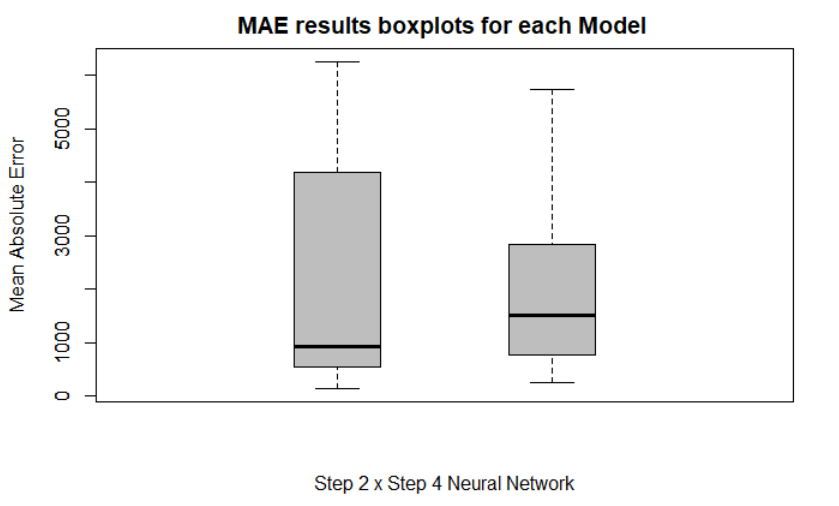


Figure 44 – Boxplot Distribution for MAE results

The plots above show that although the mean error is smaller for the Neural Network Day Trade (step 2), the Neural Network Day Trade with Daily Summary Inputs model (step 4) appears to have a smaller variance and therefore could be classified as a more stable model.

The results for applying the Wilcoxon rank sum test for both MSE and MAE distribution are plotted below:

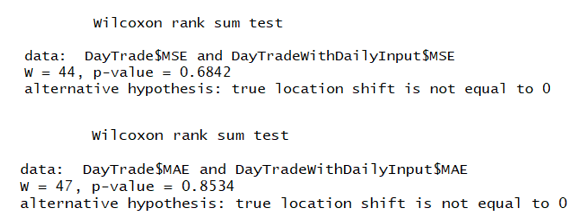


Figure 45 – Results for Wilcoxon rank Sum test for MSE and MAE

From the results above it is not possible to reject the null hypothesis, thus implying that both models aren’t statistically different.

With respect to the results achieved by the LSTM models, the high variance of the predictions makes it inapplicable for algorithmic trading. Some aspects that may have contribute to the predictions’ performance are:

1. **Normalization for non-stationary data**

For this project a normalization scaler was applied (MinMax Scaler) and fitted for all the training data. This approach has two downsides. The first being that it isn’t scalable for larger datasets and the second that, by choosing a normalization window that is too big the final resolution may not be fine grained enough so the final rescaled answers could be distorted.

A possible different technique to overcome this problem is to apply a sliding normalization window in order to normalize the data based on a number of previous data seen. However, due to the non-stationarity of the data, a volatile dataset could be difficult to normalize in the case that future data are out of bound with respect to the historical data used for normalization.

1. **The Persistence Model:**

Recurrent Neural Networks have been demonstrated to be useful for learning long-term dependencies. However, in many cases for non-stationary data, where the desire output feature is also part of the inputs, these models present the tendency to predict the value at time “t+1”, as simply being the value at time “t”. In order to avoid falling into the persistence models one of the possibilities is to transform the data into stationary before feeding it into the model. During this project this technique was also tested, although the model failed to predict results that resemble the daily trends.

New advances in the area of seq2seq enables the possibility to use different algorithms such as transformers and apply them to stock market forecast. Other techniques, such as embedding the past information into a lower dimension could also possibly help in avoiding falling into the persistence model.

1. **Model Complexity:**

The assumption within trying to forecast future sock prices is that there are patterns that repeat over time. Bitcoin’s stock prices have presented to be a very volatile dataset. Therefore, as the data approaches a ‘random-walk’ similarity, the task of trying to extract patterns increases in complexity. Albeit, throughout this project different types of architectures and tuning parameters where used, it is possible that for this dataset increasing the model complexity and performing higher number of steps per epoch could generate better predictions.

# Conclusions

The objective of this project was to verify to what extent applying Neural Networks can help forecast the next minutes of Bitcoin prices within a day fluctuation, using previous data and statistical indicators. When tried to compare a Recurrent Neural Network approach with other non-Neural approaches the project showed the lack of scalability of the later, which corroborates why Neural Networks are required in this case.

Finally, although recurrent Neural Networks have the qualities to learn sequential data, the high variability of the dataset increased the complexity for pattern learning. The results are not precise enough for applications on Day Trading but, despite the difficulties and downsides of the recurrent model, they show the capability of Neural networks.

# References

[1] Bhat, A., Kamath, S. (2013). Automated Stock Price Prediction and Trading Framework for Nifty Intraday Trading. Conference: 2013 Fourth International Conference on Computing, Communications and Networking Technologies (ICCCNT). DOI: 10.1109/ICCCNT.2013.6726675

[2] Bianchi, F. M., Maiorino, E., Kampffmeyer, M. C., Rizzi, A., & Jenssen, R. (2017). Recurrent Neural Networks for Short-Term Load Forecasting: an Overview and Comparative Analysis. Cham: Springer International Publishing.

[3] Guida, T. (2019). Deep Learning in Finance: Prediction of Stocks Returns with Long Short-Term Memory Networks. In A. Wiley (Ed.), Big Data and Machine Learning in Quantitative Investment.

[4] Hileman, Dr. G., Rauchs, M. (2017). Global Cryptocurrency Benchmarking Study. Retrieve fromwww.jbs.cam.ac.uk/fileadmin/user\_upload/research/centres/alternative-finance/downloads/2017-global-cryptocurrency-benchmarking-study.pdf (p. 23)

[5]Hougan, M., Kim, H., Lerner, M. (2019, May). Economic and Non-Economic Trading In Bitcoin: Exploring the Real Spot Market For The World’s First Digital Commodity. Retrieved from www.sec.gov/comments/sr-nysearca-2019-01/srnysearca201901-5574233-185408.pdf (pp. 4-6)

[6] Nakamoto, S., (2018). Bitcoin: A Peer-to-Peer Electronic Cash System (p. 1)

[7] Shah, Parth. (2015). Stock Prediction and Automated Trading System. International Journal of Computer Science & Communication. 6. 104.

[8] Segal, T. (2020, January 29). The Ins and Outs of Intraday Trading. Retrieved February 10, 2020, from http://www.investopedia.com/terms/i/intraday.asp

[9] Sundararajan, A. (2017). Laying the Tracks: Digital and Socioeconomics foundation. In A. MIT Press (Ed.) The Sharing Economy: The End of Employment and the Rise of Crowd-Based Capitalism (p. 59). Retrieved from https://ebookcentral-proquest-com.ezproxy.lib.ryerson.ca/lib/ryerson/reader.action?docID=4527739&query=The Sharing Economy: The End of Employment and the Rise of Crowd-Based Capitalism#

[10] Technical Indicators and Overlays. (2020, February). Retrieved from www.school.stockcharts.com/doku.php?id=technical\_indicators

[11] Gupta, Vibhuti & Hewett, Rattikorn. (2019). Adaptive Normalization in Streaming Data.