Bitcoin Price Prediction With Deep Learning

Carson Aziz
180800140
aziz0140@mylaurier.ca

Mohamed Qadeer Assan
160257370
assa7370@mylaurier.ca

Apr 6, 2023

Abstract:

Crypto price prediction is a challenging task due to the complex, dynamic, and noisy nature of financial markets. Accurate predictions of stock prices can provide valuable insights for investors and facilitate informed decision-making. The primary objective of this project is to develop a deep learning model for sequences that predicts daily stock prices using a comprehensive set of features, including price-based, volume-based, volatility-based, technical indicators, fundamental analysis, market-based, macroeconomic indicators, and time-based sequence shaped features.

To achieve this goal, we will first extract and preprocess historical financial data from multiple sources including but not limited to public financial websites, APIs, and alternative data providers. We will then perform feature engineering to draw relevant information and create a wide range of features that capture different aspects of financial instrument price movements. We will also apply feature selection and dimensionality reduction techniques to identify the most significant features and avoid overfitting.

For the methodology, we will focus on deep learning models for sequences, such as Long Short-Term Memory (LSTM) networks. These models are well suited for handling time series data and are capable of capturing the complex patterns dwelling in stock prices. We will train these models on a dataset split into training, validation, and testing sets. Model performance will be evaluated using multiple metrics including Mean Absolute Error (MAE), Mean Squared Error (MSE), and Root Mean Squared Error (RMSE). These will be used to compare the performance of different deep learning architectures to select the best one.

By the end of this project, we aim to create a reliable and robust stock prediction model that leverages the power of deep learning for sequences. With this we intend to provide valuable insights for investors and develop our understanding of quantitative finance and machine learning applications in the financial industry.

1 Introduction to Project

Cryptocurrency prices tend to be very volatile. Bitcoin is no exception. The price of Bitcoin fluctuates frequently with some fluctuations being significant. Because of the volatility of Bitcoin's price, it is desirable to be able to make accurate predictions on it's future price.

A Long Short-Term Memory (LSTM) model is a type of neural network, specifically a recurrent neural network (RNN), that takes sequences of data to produce a prediction. LSTM models are useful over run-of-the-mill RNNs as they can learn long-term dependencies of the sequence and use that to make predictions. They also have the ability to selectively remember or forget pieces of data. The LSTM uses both long term and short term data from the given sequence to make it's prediction. LSTMs have been used in finance with positive results to make predictions on stock prices. We believe an

LSTM model is an appropriate strategy for making predictions on Bitcoins price and is what we have chosen to explore in this project.

In order to make the most accurate model for Bitcoin price prediction, various models, and configurations of those models, must be tested and measured in order to compare which model configuration is giving the best results. Models will be selected based on these metrics for the final Bitcoin price prediction model.

This report will explore the results of using LSTM models to make Bitcoin price predictions. We will explore the data pre-processing requirements, feature selection, and model tuning that went into crafting our model.

2 Pre-Processing and Exploratory Data Analysis

2.1 Data Collection

The data used to train the Long Short-Term Memory (LSTM) models includes Bitcoin, Etherium, Dogecoin and Ripple (XRP). The data was collected using the Bitfinex Python library. Bitfinex is a popular cryptocurrency exchange, with a public API that allows us to get real-time and historical Bitcoin price data from Bitfinex. The data comprises the high, low, open and close prices of Bitcoin in USD, as well as the high prices of Bitcoin in EUR and JPY. Additionally it includes the high prices of Ethereum, Dogecoin and Ripple (XRP) in USD.

The collected data spans one year. When the program is run, it retrieves data from the corresponding date a year ago until the present day at 4-hour intervals. There are roughly 13 000 pieces of data for Bitcoin, and roughly 2000 pieces of data each for Ethereum, Dogecoin, and Ripple (XRP).

The use of the high, low, open and close Bitcoin prices provides a broad view of the price fluctuation over time. The high prices of Ethereum, Dogecoin and Ripple (XRP) are included, as cryptocurrency prices tend to follow similar trends over longer periods of time, and the high data of these cryptocurrencies could have a correlation with the high price of Bitcoin on a given day. The Bitcoin prices in JPY and EUR are included as additional data points.

Bitfinex is one of the largest and most commonly used cryptocurrency exchanges at the moment, and they provide a public API for access to real-time crypto prices, as well as historical data for various cryptocurrencies. We have used the Bitfinex Python library, which uses the Bitfinex API, to gather the data used to train our model.

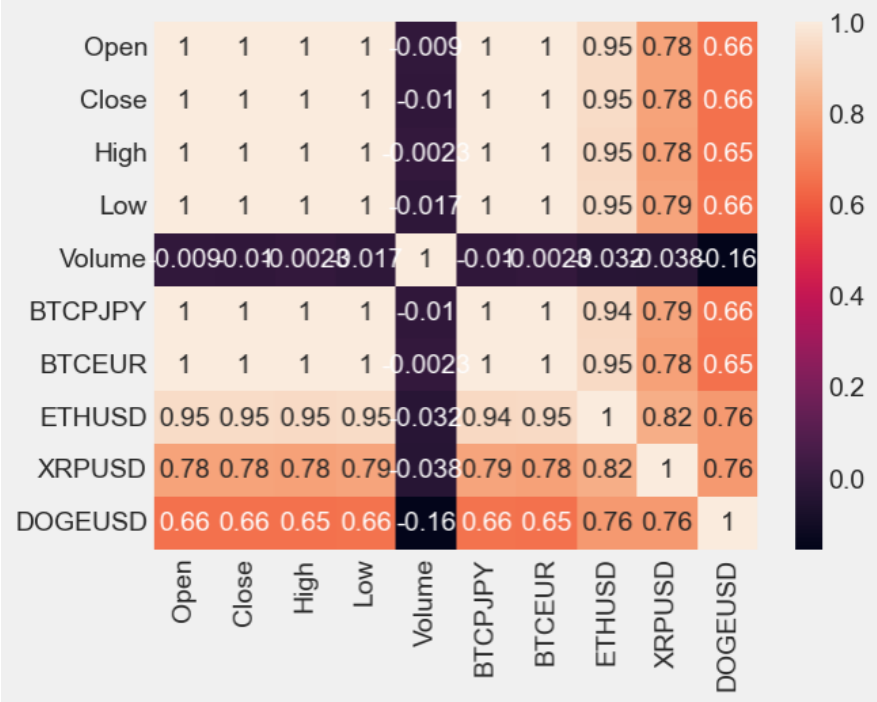
2.2 Data pre-processing

1. Collect data for each ticker (BTCUSD, BTCEUR, ETHUSD, etc…)
2. Order ticker data by date
3. Extract “High” columns from each ticker data set except BTCUSD
4. Concatenate “High” columns onto BTCUSD dataset.
5. Normalize the data
6. Generate 60 sample sequences of data
7. Convert data to numpy array of the appropriate shape to feed to the neural network model

2.3 Exploratory Data Analysis and Visualizations

When measuring the correlation between the different features, the results we found were mostly expected. The high, low, open and close prices for BTCUSD were all found to be correlated with each-other. Additionally, the high prices of the other cryptocurrencies were found to have a varying significance of correlation with the high,

low, open and close prices, with Ethereum having the highest correlation then Ripple, then Dogecoin. Surprisingly the volume of BTCUSD was not found to have any statistically significant correlation with the high, low, open or close prices.



For our model we carefully selected the features we deemed important for predicting the high price of Bitcoin. From the data we collected for BTCUSD, we used the high, low, close, and open prices to train the model, as well as the high prices Ethereum and Ripple (XRP) and Bitcoin in EUR and JPY. We selected these features based on the assumption that the high prices of the other cryptocurrencies would be an indicator of Bitcoins performance. Because cryptocurrency is so volatile, and cryptocurrencies tend to influence other cryptocurrencies, we included their high price data in our training data, in order to make better predictions. We left out the volume and DOGEUSD as features of our model as their correlation was not significant enough to warrant their inclusion in the model. The volume feature in particular had near zero correlation.

3 Methodology

3.1 Platform and Machine Configurations Used

Environment: Visual Studio Code with Conda environment
CPU: Apple Silicon M1 Pro
RAM: 16 GB

3.2 Data Split

The data was split into training and testing sets using a 90-10 split ratio respectively. The training set was split again using a 90-10 split ratio for training and validation sets respectively. This training set was used to train the model. And the testing set was used to evaluate the models performance.

3.3 Model Planning

We used two different models to make predictions on Bitcoins price. The models had the same structure, and differed by their feature sets. We used Keras sequential models to build a neural network to make our predictions. Using keras sequential model allows us to shape the neural network by creating it layer by layer. Our neural network consisted of multiple LSTM layers, multiple dropout layers, and multiple dense layers. LSTM (Long Short-Term Memory) layers are a type of recurrent neural network that can capture dependencies in sequential data, which makes them good choices for time series prediction. Dropout layers can be useful in preventing overfitting by “dropping out” some of the input values during training to force the neural network to learn more generalized features that are not dependant on specific inputs. Dense layers can learn complex non-linear relationships between it’s input and it’s output.

The dropout layer randomly sets a percentage of its input to zero during each training epoch, effectively dropping those inputs. This can make the model less sensitive to noise, and can help produce more accurate predictions on new data.

The dense layer is a fully connected layer that performs a linear operation on its input data, followed by a non-linear activation function. Dense layers are used to learn mappings between the input data and the output layers.

Our model's structure processes the data in stages. Each stage begins with an LSTM layer. This is done in order to learn long-term dependencies on data after they pass

through the various stages of the network, the next layer in the stage is then either a dropout layer or a dense layer. Dropout layers help desensitize that stage from noise output by the previous stage. While the dense layers help create mappings to the prediction output nodes.

Our models differ in their feature sets. We have selected feature sets that differ in minimum correlation requirements in order to have a more diverse set of features that we use to make predictions on our input data.

Model Optimization

In order to optimize the models parameters, we experimented with different configurations of the models structure, and various hyper parameters.

The two models differ in correlation requirements, our first model has a minimum 0.90 correlation allowance for features chosen in and our second model has a minimum 0.70 correlation allowance. With this we aim to research how correlations between features affect prediction quality.

We begin with the neural network setup pictured below as our baseline to improve upon based on tuning. Comparisons are made by analyzing mean squared error between train and validation sets. When comparing MSE between train and validation sets we aim to see both values decreasing and also converging together.

```
model = Sequential()
model.add(LSTM(units = 50, activation = 'linear', return_sequences = True, input_shape
= (X_train.shape[1], X_train.shape[2])))
model.add(LSTM(units = 60, activation = 'linear', return_sequences = True))
model.add(Dropout(0.3))
model.add(LSTM(units = 80, activation = 'linear', return_sequences = True))
model.add(Dropout(0.4))
model.add(LSTM(units = 120, activation = 'linear'))
model.add(Dense(units =1))
model.summary()
```

The different parameters we decided to tune in order to optimize our model were the activation function of the layers, the number of layers and the nodes within those layers,

and the optimizer parameter.. The performance metrics of these different configurations allowed us to select the most optimal model.

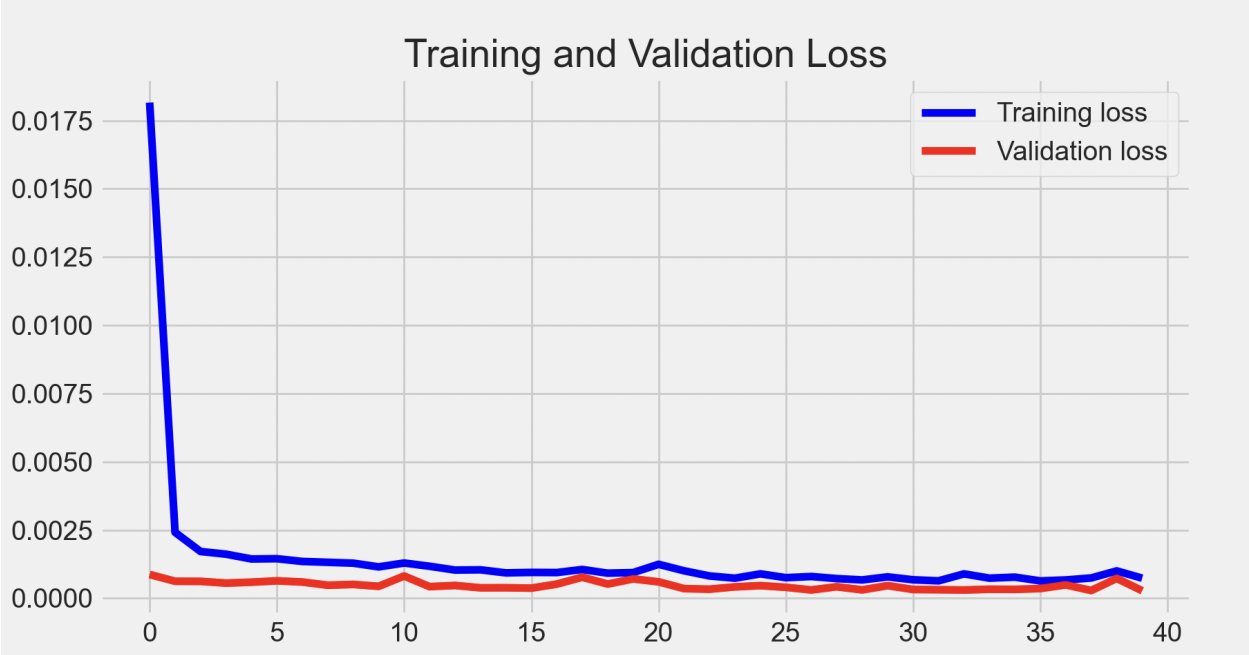
4 Results

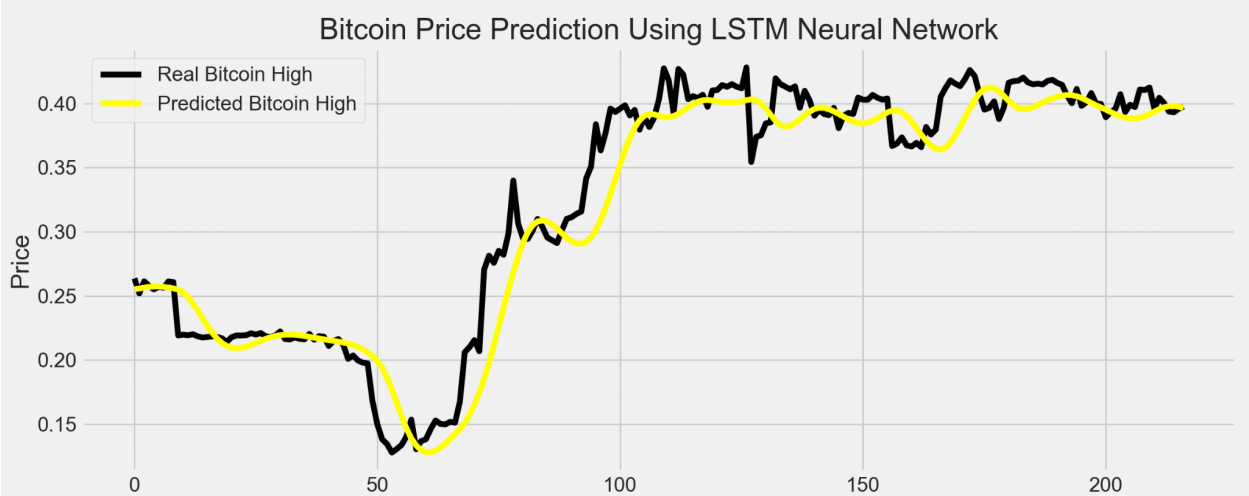
The baseline model mentioned above is iteratively improved upon using the metrics below to achieve an optimized version of our model.

4.1 Linear vs ReLu

Model 1 (0.90+ feature correlation with BTCUSD High)

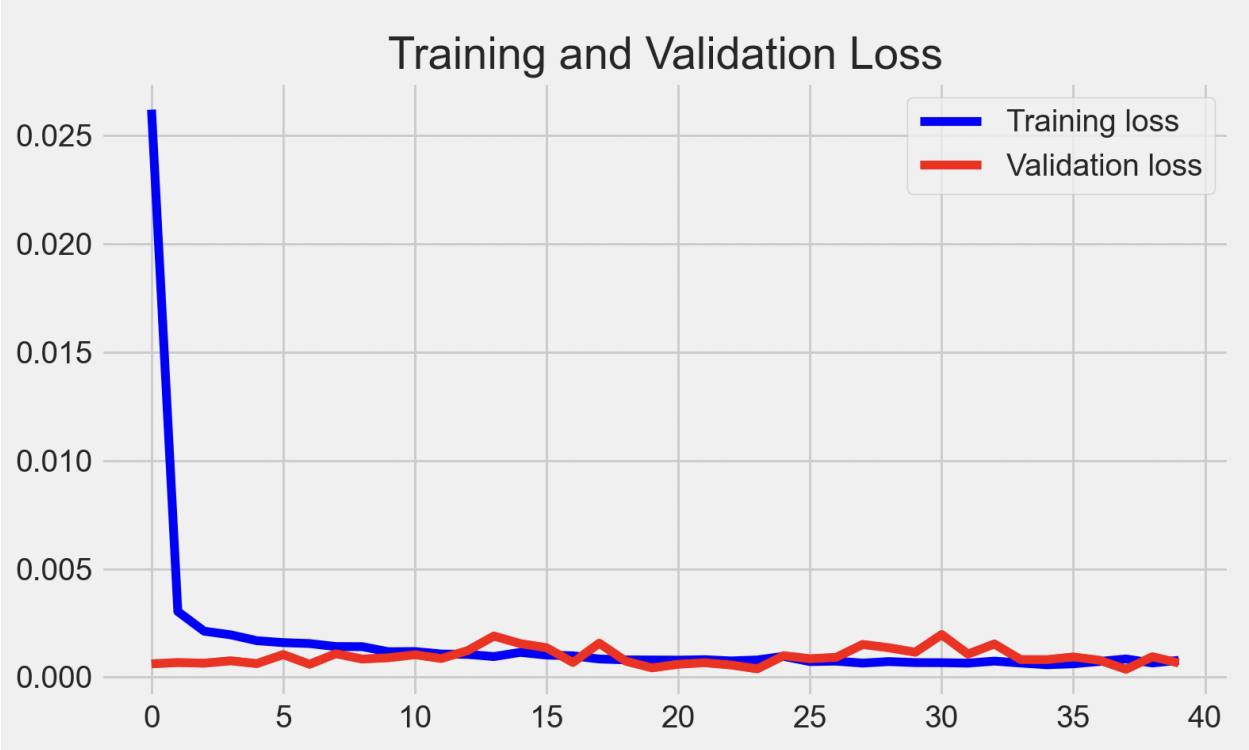
Linear

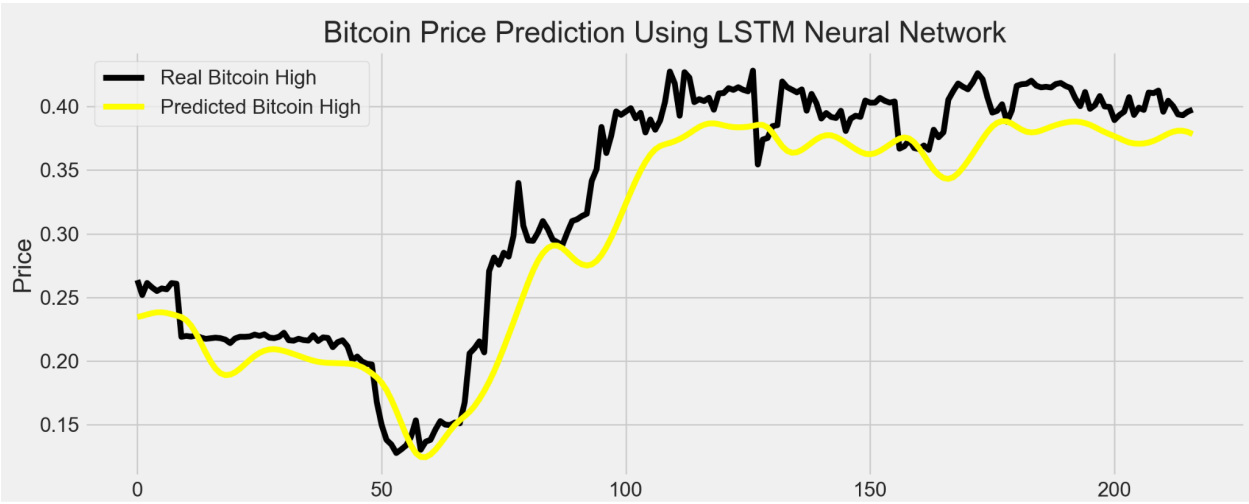




MSE: 0.0004933533558310701
RMSE: 0.02221155905899156
MAE: 0.017265330707986068
R-squared: 0.944503386620869

ReLu

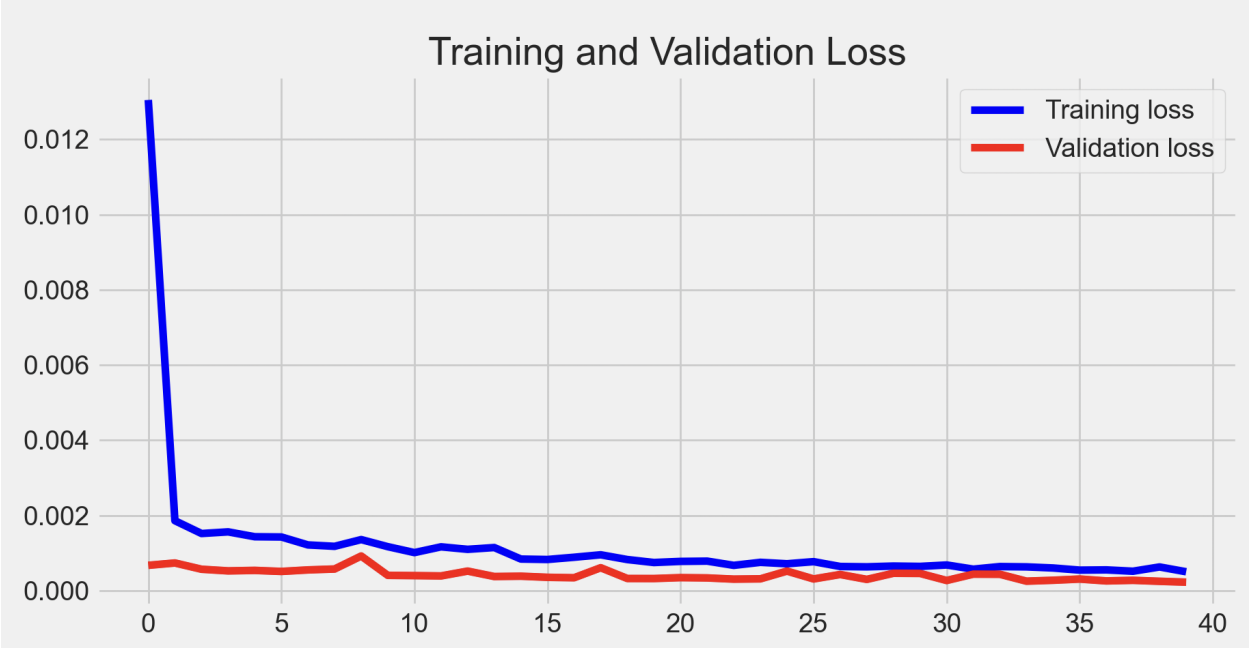


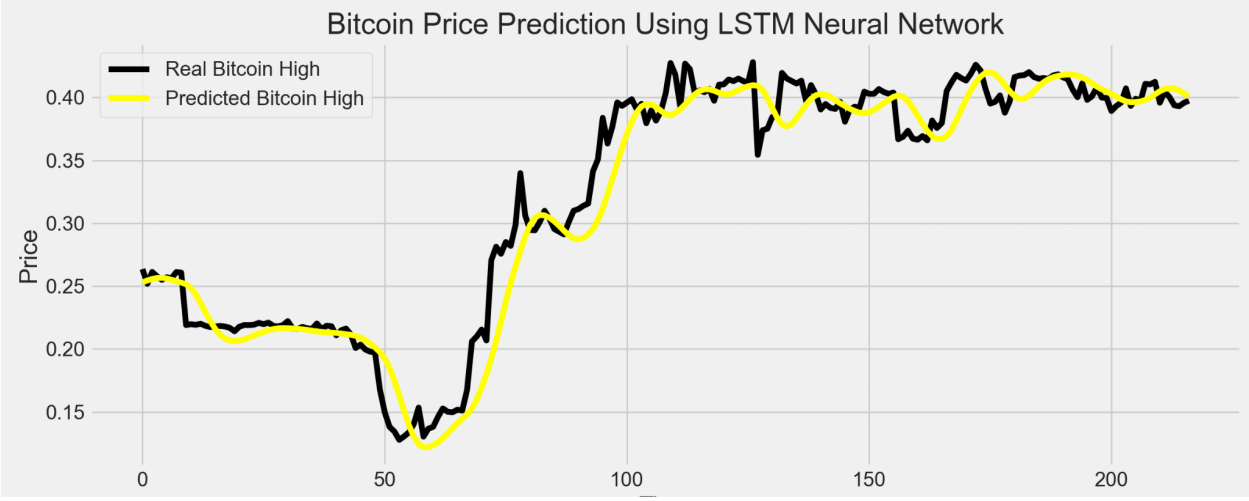


MSE: 0.0011939077383123962
RMSE: 0.03455297003605329
MAE: 0.028067887342849776
R-squared: 0.86569902610298

Model 2 (0.70+ feature correlation with BTCUSD High)

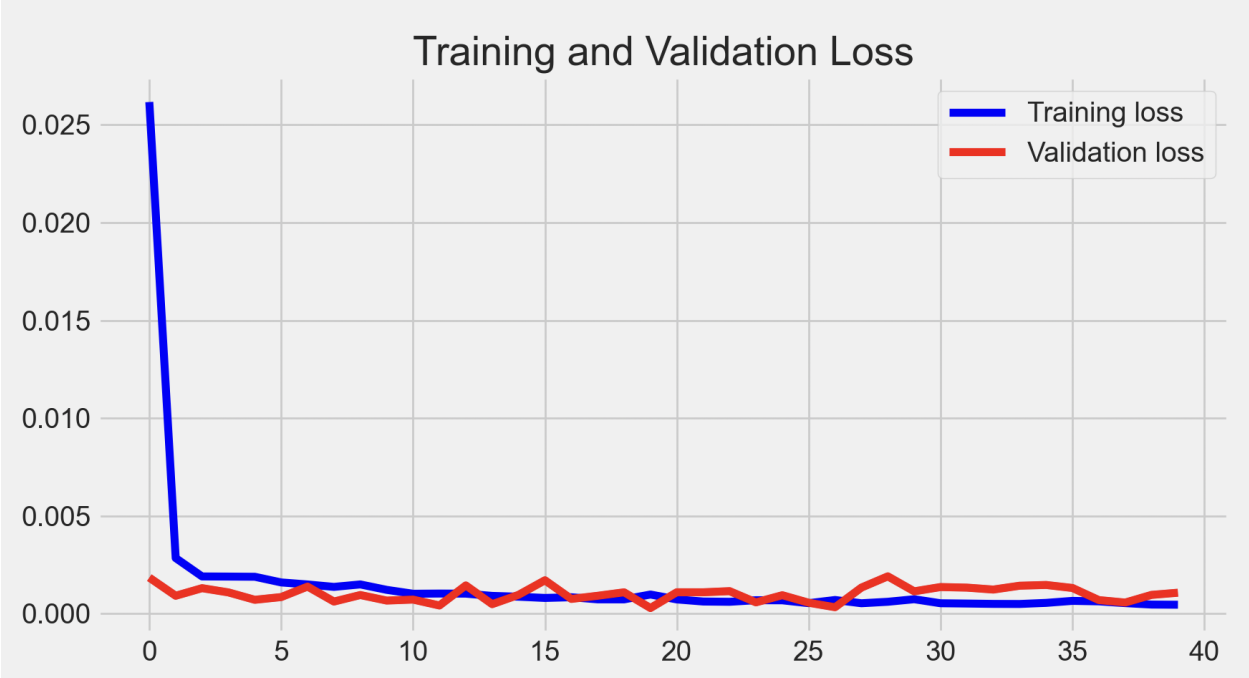
Linear

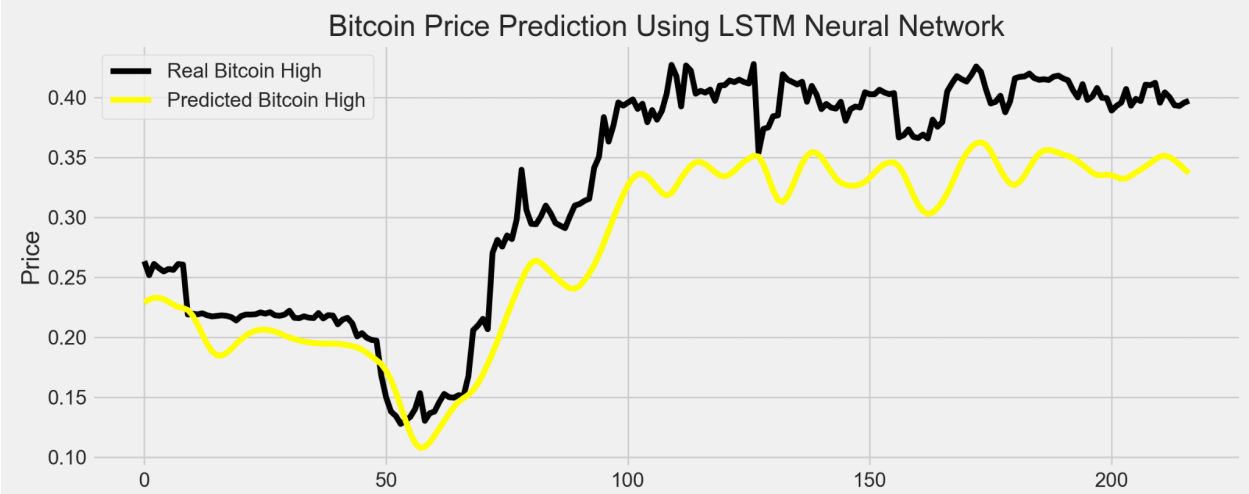




MSE: 0.0005838693082270527
RMSE: 0.024163387763868144
MAE: 0.017333702245582606
R-squared: 0.934321376596225

ReLu





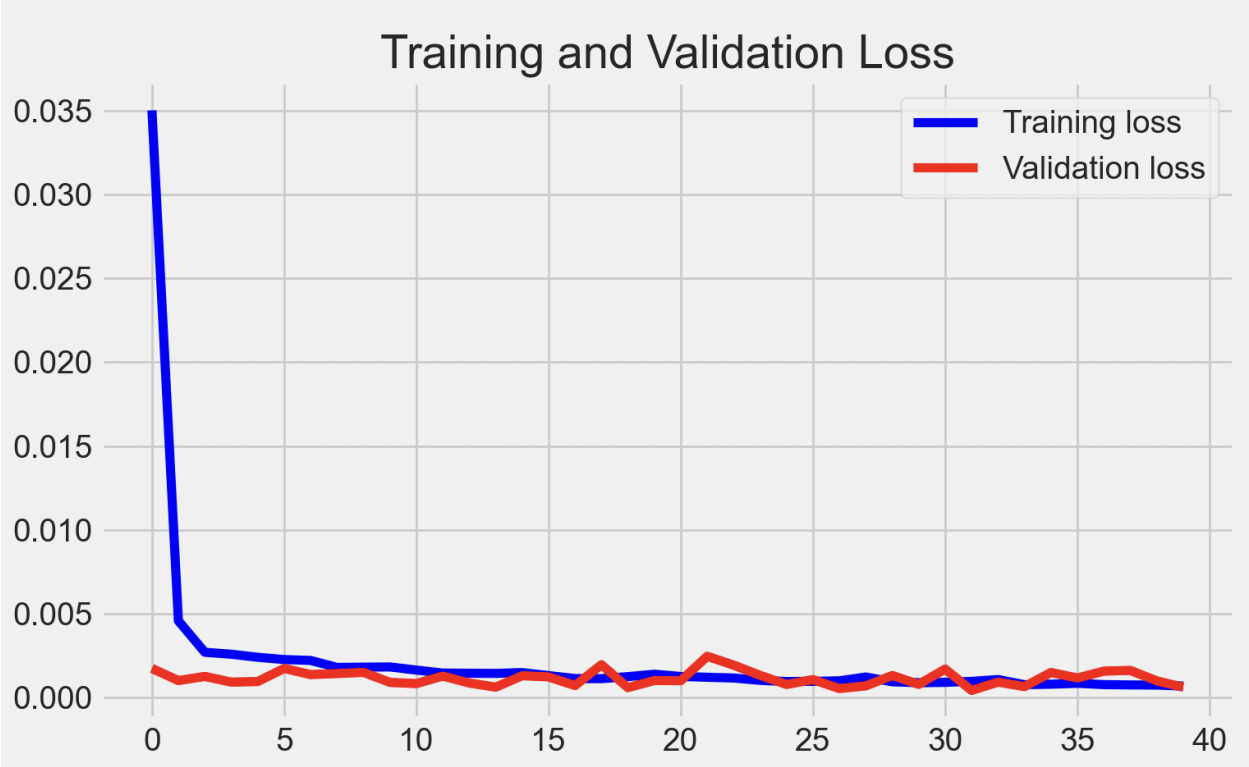
MSE: 0.0030210789172693776
RMSE: 0.05496434223448306
MAE: 0.049265687322890044
R-squared: 0.6601631534924615

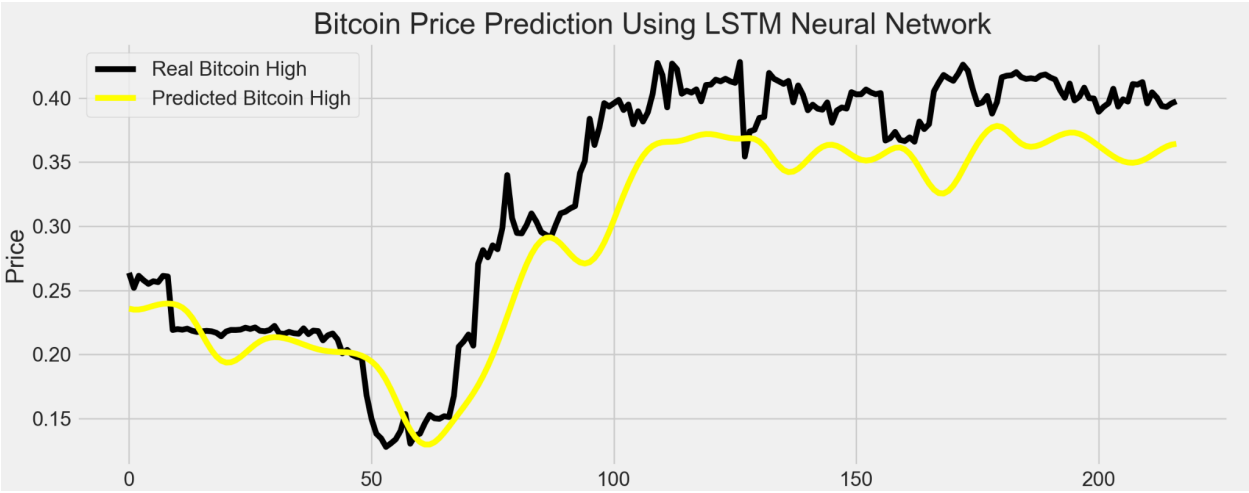
We maintain linear activation due to smoother convergence of train and validation MSE and better fitted prediction line..

4.2 Adding More Layering & Nodes Per Layer

Model 1 (0.90+ feature correlation with BTCUSD High)

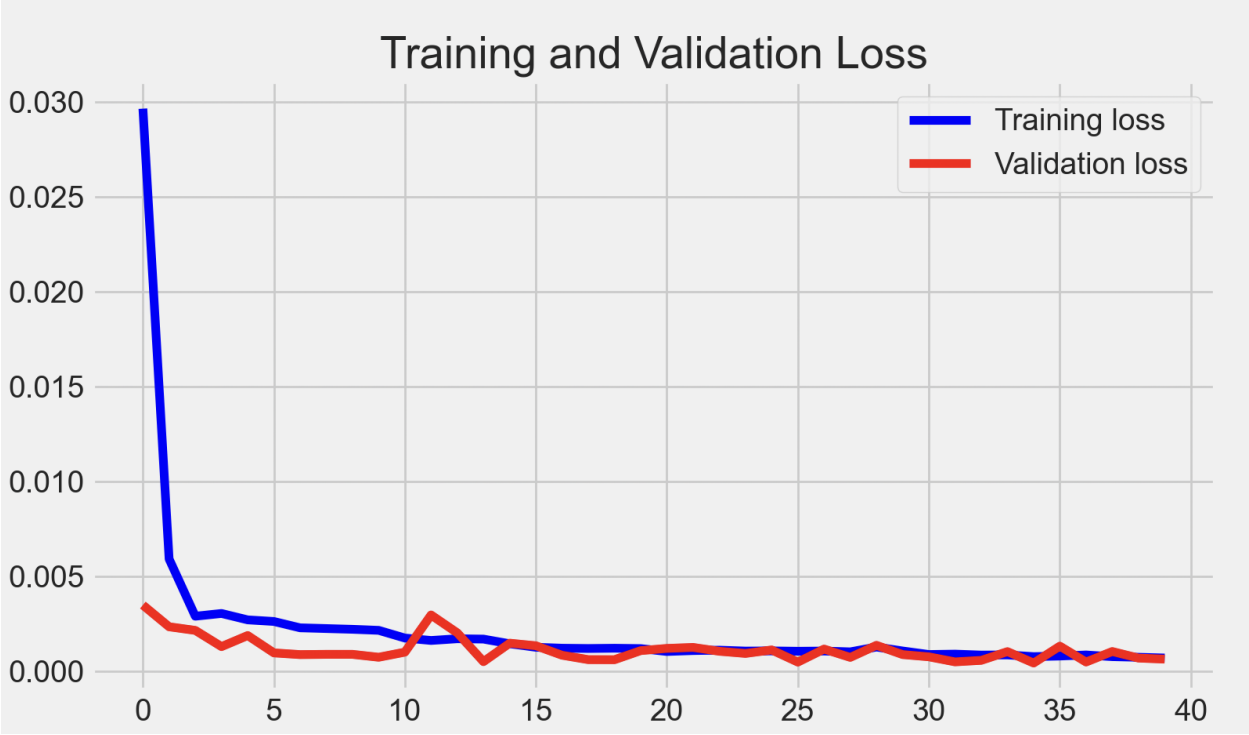
Adding two more LSTM layers with a total of 140 more nodes

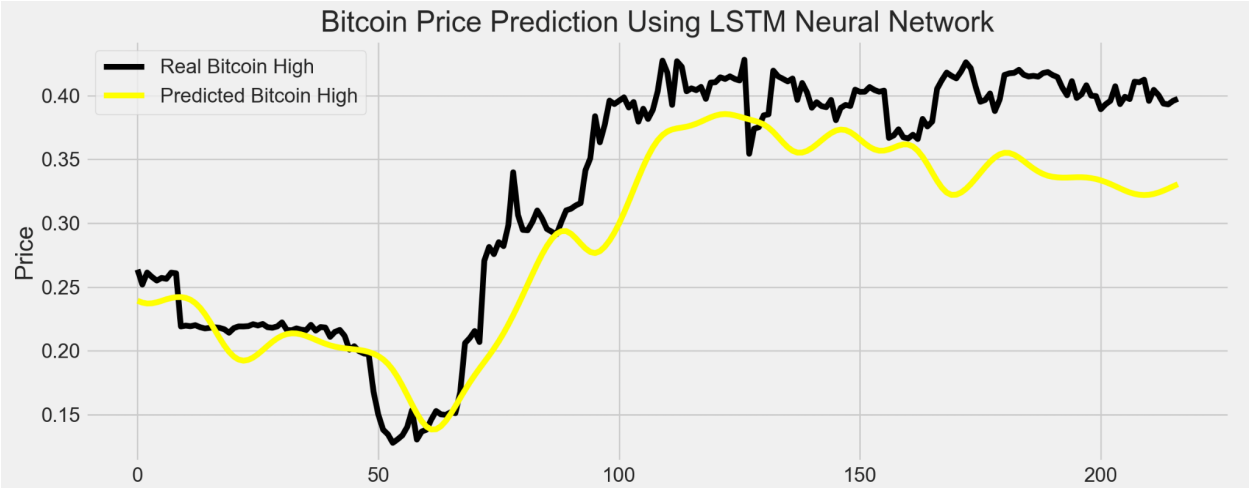




MSE: 0.0019989447121235735
RMSE: 0.04470955951609872
MAE: 0.03691427760574574
R-squared: 0.7751415683225521

Model 2 (0.70+ feature correlation with BTCUSD High)





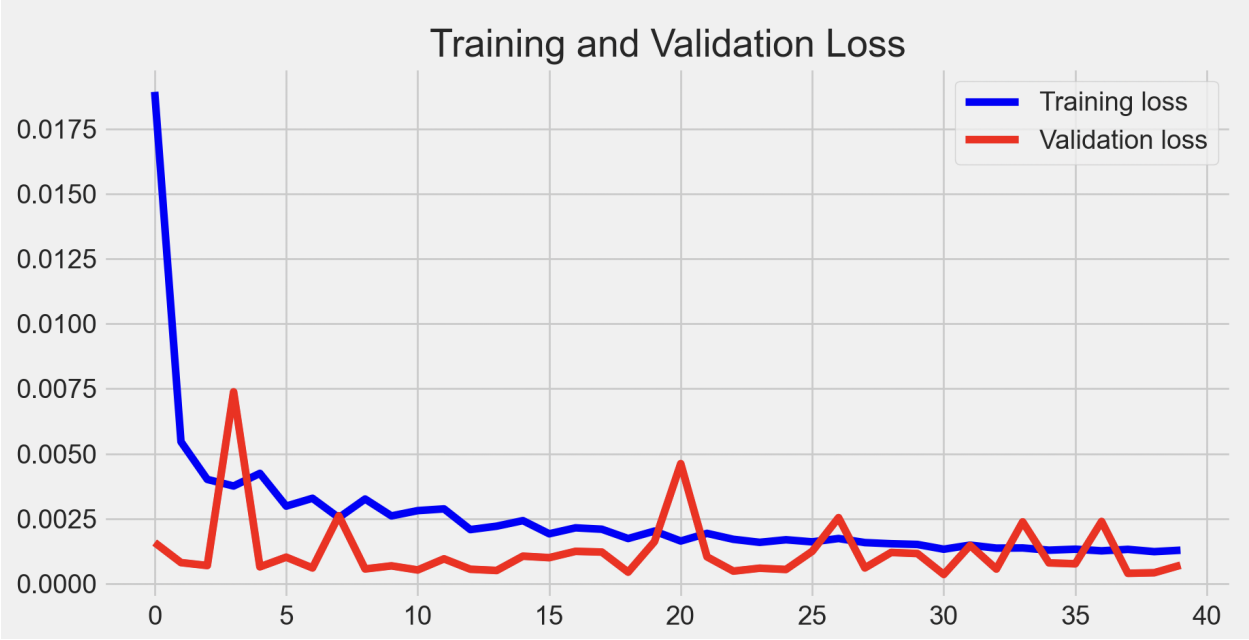
MSE: 0.002439719045789978
RMSE: 0.04939351218318027
MAE: 0.040141736099284786
R-squared: 0.7255594939456131

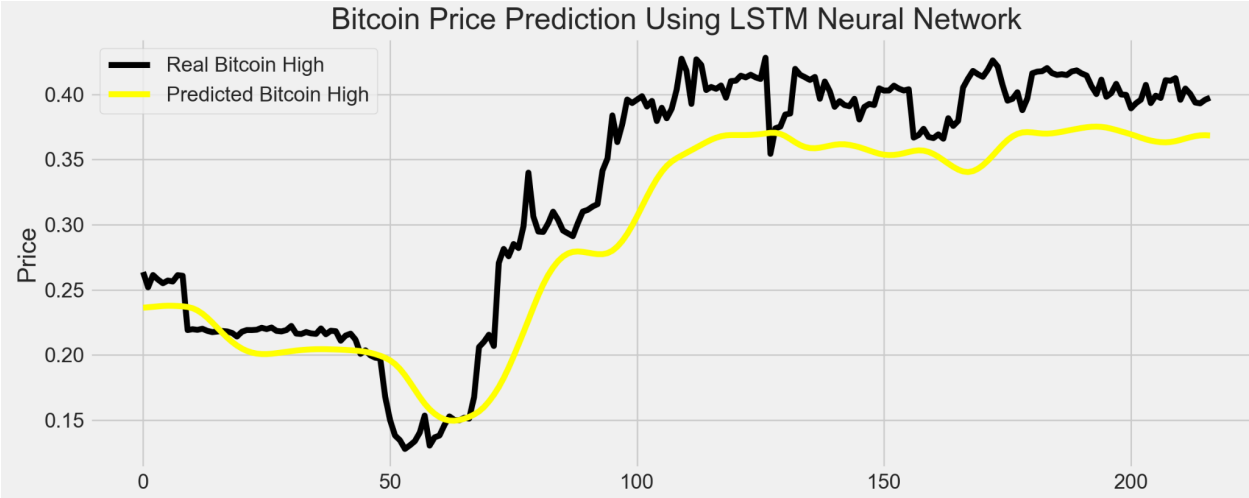
Adding further layering results in underfitting, so we remove the extra layering for further optimization.

4.3 Optimizer Parameter Comparisons

Model 1 (0.90+ feature correlation with BTCUSD High)

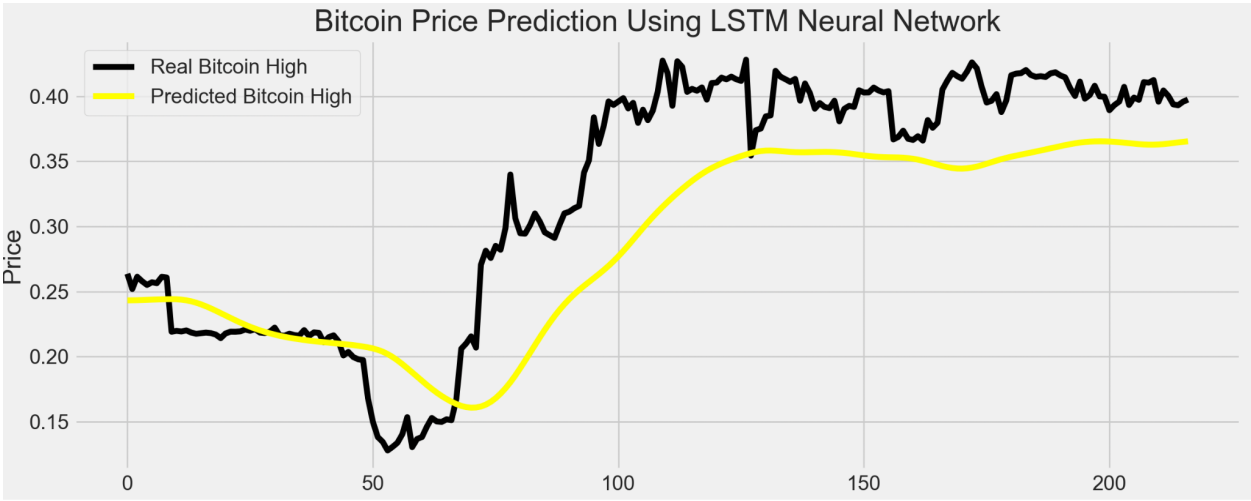
RMSProp





MSE: 0.0018291731516093487
RMSE: 0.04276883388180403
MAE: 0.035810583601385684
R-squared: 0.7942389283491368

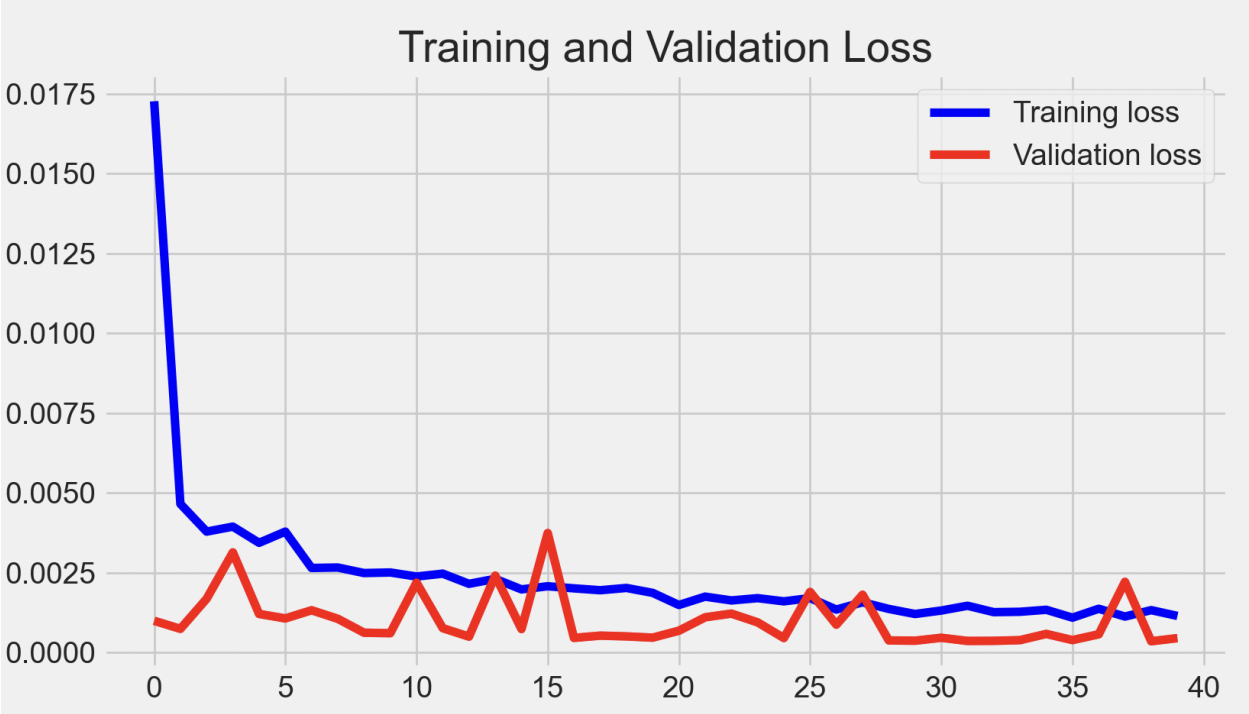
Stochastic Gradient Descent

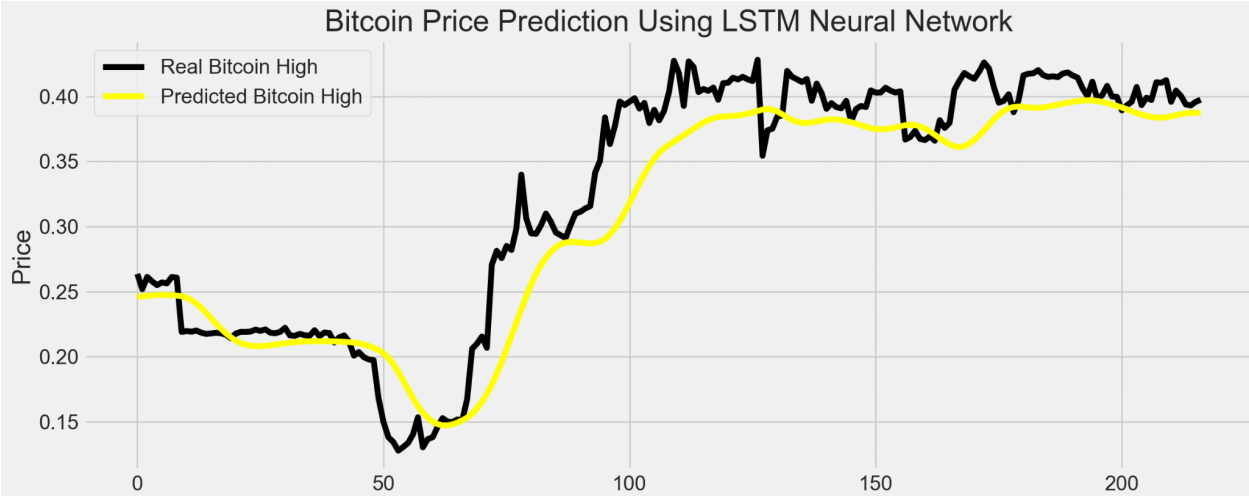


MSE: 0.003339448222421179
RMSE: 0.05778795914739661
MAE: 0.04754990364812505
R-squared: 0.6243502457034203

Model 2 (0.70+ feature correlation with BTCUSD High)

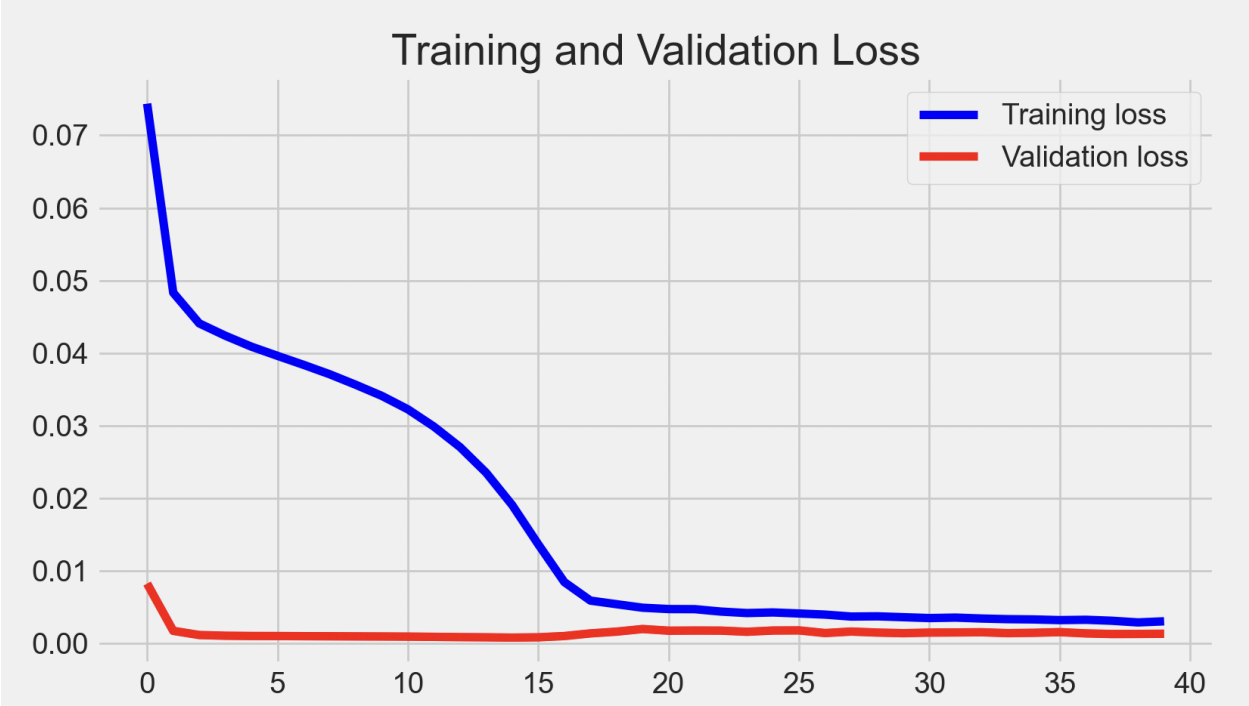
RMSProp

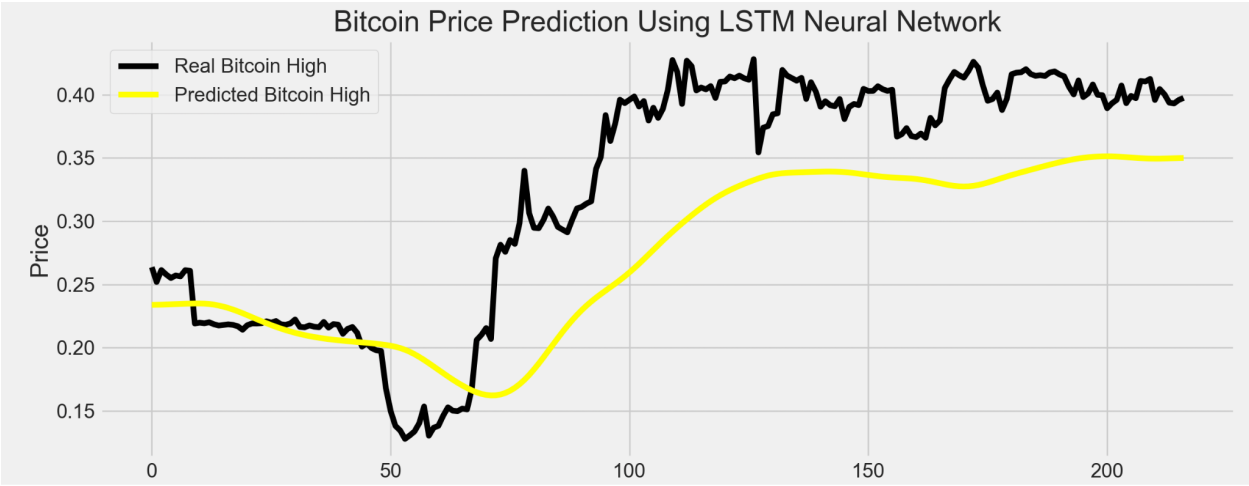




MSE: 0.0010777118233994932
RMSE: 0.03282852149274306
MAE: 0.02459338124097045
R-squared: 0.8787697383824031

Stochastic Gradient Descent





MSE: 0.004829678475610005
RMSE: 0.06949588819210821
MAE: 0.05878434772526976
R-squared: 0.4567163759230287

Conclusion

The application of LSTM models to forecast the high price of Bitcoin has yielded promising results, with our models demonstrating reasonable accuracy and directional symmetry. Our top-performing model achieved a mean squared error of 265.094 and a directional symmetry of 0.640, suggesting that our model can successfully predict the direction of high price fluctuations around 64% of the time. These outcomes showcase the potential of using LSTM models for predicting cryptocurrency prices, and future research could involve additional refinement of the model and feature selection. Possible improvements could involve adding more features with between 0.70 and 0.80 correlation as slightly lower correlation allowed for better generalized fitting of the model. We ultimately came to the conclusion that our model 2 with 0.70+ feature correlation was the best prediction model.