UNIVERSITY OF SUSSEX

A Variance Inflation Factor (VIF) Feature Selection Approach to Bitcoin Price Forecasting

Daniel Peacock

Finance and Technology

May 2022

Abstract

Innovative technology has created new ways for individuals to make investments. Deep learning techniques are mathematical models that can be used to forecast timeseries data. These models have been applied in forecasting literature to predict the price of assets with mixed results. This research applies deep learning technique Long Short-Term Memory (LSTM) to predict the price of bitcoin using the error metric root mean squared error (RMSE) to evaluate models. A simple buy/sell strategy was implemented to compare profit/loss results with RMSE. Variance inflation factor (VIF) was implemented, and 3 different feature sets were tested. Each model was built using Hyperbolic Tangent (TanH) and Rectified Linear Units (ReLU) activation functions, both prior to and after hyperparameter tuning. The features used for this task include cryptocurrency market data, technical indicators data, blockchain data, macroeconomic data, traditional market data, and search engine data. The results oppose current literature and suggest that one feature forecasting is the best predictor of bitcoin price in terms of RMSE on unseen test data, irrespective of activation function and hyperparameter tuning. VIF had the effect of reducing overfitting but did not prove an effective strategy to minimise RMSE loss or maximise profit/loss.

Table of Contents

1. Introduction

Blockchain is an emergent technology defined broadly as a type of distributed ledger technology (DLT) holding permanent records of all transactions (Lafourcade & Lombard-Palet, 2020). Bitcoin (Nakamoto, 2008) was the first blockchain, intended for use in finance as an alternative payment system to traditional banking. It is decentralized meaning every computer participating runs the network, and no individual participant can alter the network state alone. Nodes are computers that validate transactions and group them into blocks. These blocks are broadcast to the rest of the network and must be confirmed as correct by minimum 51% of the systems participants for the whole systems state to be updated. This majority consensus is what provides Bitcoin its security, as no one individual should possess the computing power to pose as 51% of the network's participants.

Transactions are conducted using cryptocurrencies, a medium of exchange functioning as an electronic form of currency native to its blockchain (Chohan, 2017). Digital coins differ from physical currencies as they are not associated with central banks, national borders, or sovereigns (Maese, Avery, Naftalis, Wink, & Valdez, 2016).

Markets exist for individuals to buy and sell assets like stocks, commodities, and currencies. Cryptocurrencies are a new type of digital asset born from blockchain technology that now have their own markets. Like forex markets, crypto markets operate 24hrs. Bitcoin is the highest market capitalization cryptocurrency and now the 14th largest currency based on market capitalization (CoinMarketCap, 2021), compared with fiat currencies like the United States Dollar ($). Blockchain and DLT has provided a new means for sending payments, cryptocurrencies are the economic incentive to use these networks.

Cryptocurrencies are more volatile than assets in other markets meaning their price rises and falls more dramatically (Pichl & Kaizoji, 2017). Predicting cryptocurrency prices has therefore become an attractive task for high-risk investors. Since 2017, over 170 hedge funds have emerged specialising in cryptocurrencies (Mokhtarian & Lindgren, 2017). More than 1,500 cryptocurrencies are actively traded worldwide across different exchanges. The cryptocurrency market is now a financial system worth over $800+ billion (Mokhtarian & Lindgren, 2017).

Towards the end of 2020 during the coronavirus pandemic bitcoins price rose substantially. Since, large volumes of bitcoin have been bought and sold creating recent timeseries data that contains frenetic volatility, shown in Figure 1.

Figure 1. Bitcoin Close Price

Note: Graph was created in Python. Bitcoin values are $ values. See section 3.2.1 for data source.

Long Short-Term Memory (LSTM) (Hochreiter & Schmidhuber, 1997) is a prediction model that can be used to forecast timeseries data such as bitcoin price. Forecasting research has looked extensively at assessing the effectiveness of LSTMs for predicting asset prices like stock price or fiat currency price (Rana, Uddin, & Hoque, 2019) (Mehtab & Sen, 2020) (Sen & Chaudhuri, 2018). Many have found it to be superior to traditional forecasting techniques for predicting bitcoin when evaluated using common forecasting error metrics (Jay et al, 2020) (McNally et al, 2018) (Li & Liao, 2017). Due to LSTMs cited superiority, in this research I implement LSTM models to forecast bitcoin price.

2. Literature Review

Fama (1970), introduced the efficient market hypothesis stating that historical price data cannot be used to accurately predict future price changes. This seminal work has been challenged by investors and researchers, some of whom have made above market profits. *Palamalai et al.* (2021), tested the top 10 cryptocurrencies based on market capitalization against the efficient market hypothesis and found that markets are generally not efficient. They note that previous literature had mixed results but generally supports the notion that cryptocurrency markets are inefficient. Inefficient markets by nature do not have all information factored into their prices. This is relevant to our study as it indicates the possibility of additional information that could be acquired and used for price prediction.

Mallqui & Fernandes (2019), conducted research into the effects of macroeconomic and blockchain data for predicting the price of Bitcoin, using both Support-Vector Machine (SVM) and ANN with TanH activation. Their feature selection method was Principal Component Analysis (PCA). They concluded that blockchain information such as difficulty, gas fees, mining profitability and macroeconomic information such as SP500, crude oil futures, gold futures prices are important factors for bitcoin price prediction. They also note that the use of technical indicators such as Williams %R and MACD may improve model results and suggest these as data features to be used in future research. Sovbetov (2018), examined the factors that influence the five most common cryptocurrencies using the Autoregressive Distributed Lag cointegration technique. He noted that crypto-market related factors like market beta, trading volume, and volatility are significant factors that influence prices. He also noted how other market data such as SP500 index returns showed some influence over the long-term price of bitcoin. Saad et al. (2019), used machine learning techniques to predict bitcoin prices and concluded that blockchain information such as difficulty, gas fees, mining profitability are important factors for bitcoin price prediction. Research by Yelowitz and Wilson (2015) found that internet search data, in particular Google Trends data, are useful predictors of bitcoin price changes. This finding is further corroborated from research conducted by Wolk (2019), whose empirical forecasting findings suggests that social media sentiment such as Twitter Sentiments and Google Trends data are useful predictors of cryptocurrency prices. Other forecasting studies (Yildirim & Toroslu 2021) (Yong et al. 2015), conducted on forex markets have used a range of technical indicator data to predict exchange rates with improved results. These findings somewhat corroborate the findings from Palamalai et al. (2021), that cryptocurrency markets are inefficient, and opposes Fama's (1970) efficient market hypothesis. Research by Chen et al (2020) showed that by including additional information, LSTMs are superior forecasting techniques to other more traditional models like Support Vector Machine.

Jay et al. (2020), tested deterministic models against stochastic models for cryptocurrency price prediction. Deterministic models (Multi-Layer Perceptron (MLP), and LSTM) were found to perform better than random walk. Both Deterministic models used Rectified Linear Units (ReLU) and were trained using Adam optimizer. McNally et al. (2018) showed that RNN and LSTM are better at price prediction when compared with more traditional neural network models like Multi-Layer Perceptron (MLP). It was suggested that this is due to RNNs and LSTMs ability to retain information about price behaviour prior to the forecasting window. They also used a range of activation functions and found that TanH was superior, although the differences were not significant. Other research (Li & Liao, 2017) has also shown that RNN architectures are superior to more traditional forecasting techniques (SVM, ARIMA) for other types of asset price prediction. The forecasting literature referenced here suggests that LSTMs are a superior forecasting technique compared to traditional econometric models.

Rana, Uddin, and Hoque (2019), conducted research to analyse the effects of activation functions and optimizers on stock price prediction using LSTMs. They tested this for a range of machine learning techniques and found two activation/optimiser pairs that produced the best results. These were linear activation function with adamax optimizer, and TanH activation with adam optimizer. TanH was found to be significantly better in terms of RMSE compared with ReLU when using adam optimizer.

Dutta et al (2020), conducted research into predicting the price of bitcoin using LSTM & GRU architectures. Their research used a set of exogenous and endogenous features including but not limited to price of bitcoin, bitcoin daily lag returns, price volatility, miners' revenue, transaction volume, transaction fees, hash rate, money supply, block size, SP500 returns, Google Trends data, and Metcalfe-UTXO. They used the activation function hyperbolic tangent (TanH) as it produced better results when evaluated by RMSE. They note that TanH suffers from vanishing/exploding gradient but speculate that RNN model prediction was improved due to the second derivative being sustained for a longer time before converging to zero when compared with ReLU. In their study they used Variance Inflation Factor (VIF) to select the features used to predict prices. They also implemented a simple buy/sell and long/short strategy using signals that evaluated forecasted price against actual price, with their results showing that profits were gained over a 7-day test period. Their results were validated using the Diebold-Mariano hypothesis test. They mention that hyperparameter tuning was implemented but do not specify by what methods. Their project write-up is approximately 5,000 words. I felt as though much of their methodology lacked sufficient explanation.

Other forecasting research conducted on stock prices (Mehtab & Sen, 2020) (Sen & Chaudhuri, 2018) has also implemented VIF as a feature reduction technique although to the best of my knowledge, no study exists which critically evaluates different feature selection methods for asset predictions.

Research above suggests that by including additional data features from data categories like technical indicators, blockchain data, social sentiment, macroeconomic indicators, our LSTM models' performances should improve. To test this, data from all the mentioned categories was collected. Three different feature sets were created: 1 feature, all features, VIF selection features; and provided to LSTM models to forecast bitcoin price.

Literature pertaining to whether TanH activation is superior to ReLU in terms of common error metrics is mixed. The two main papers I have drawn my experimental design from (Rana, Uddin, and Hoque, 2019) (Dutta et al. 2020) both suggest TanH is superior. Although other forecasting literature shown above suggests these differences are not significant (McNally et al, 2018), or used ReLU anyway (Jay et al, 2020). To test which is superior, I built LSTM models using both TanH and ReLU activation functions to compare the results.

This research contributes to the field of bitcoin asset forecasting by critically evaluating the effects of applying VIF to a set of features cited above as being useful for LSTM bitcoin forecasting. Much of the methodologies used in Dutta et al's research (2020), such as VIF and Diebold-Mariano hypothesis testing, are also employed here and explained in more detail than they go into. Additionally, both TanH and ReLU activations were tested to clarify mixed opinions in literature about which is superior.

3. Methodology

The experiment I conducted in this research has two parts. In Part 1, I build a base model LSTM and explore the impact of forecasting with different sets of features. Three feature sets were used. The first feature set contains only close price. The second feature set contains all 58 features. The third feature set had feature selection applied to it and contained 26 features. Feature selection for this third part was conducted using Variance Inflation Factor (VIF) (James, Witten, & Hastie, 2013). All three feature sets were tested using my base model for both Hyperbolic Tangent (TanH) (Saury, 1774), and Retified Linear Units (ReLU) (Nair & Hinton, 2010), activation functions. The results from base models using different feature sets / activation functions are compared to analyse the effects of including additional features and using alternative activation functions on loss scores of unoptimized models.

In Part 2, I perform grid search hyperparameter tuning for each feature set. Hyperparameter tuning for our LSTM models in this section was also applied for both TanH and ReLU activation functions. Results from hyperparameter tuned models are compared with base model results so that the effects of tuning can be analysed. The results from tuned models using different feature sets / activation functions are compared to analyse the effects of including additional features and using alternative activations on loss scores of optimized models.

I also implemented a simple buy/sell trading strategy on model forecasts. The results of this allowed for a direct comparison between forecasting loss scores and profit/loss calculations. The implications of this relate to the real-world application of the task we are conducting, which is to use forecasting results to place cryptocurrency trades.

Models were implemented on Python (Guido van Rossum, 1991) using Keras (Chollet, 2015) deep learning library built over TensorFlow (Google, 2015). Keras was created to make implementations of deep learning models fast and easy as it has inbuilt functions for build models. Using this library expedited my research process. Due to the pre-built functions of Keras, the ability to alter the mathematical properties of deep learning models is limited. When instantiating Keras models without specifying specific hyperparameter values, default values are used (Chollet, 2015).

The following pages provide an exposition of the methodologies that were employed during this research, they are important for understanding the experimental results and discussion.

3.1 Model: Long Short-Term Memory

Artificial Neural Networks (Rosenblatt, 1958) (ANN) are prediction models that loosely base their structure on biological neural networks (McCulloch & Pitts, 1943). They can be described as weighted directed graphs, where inputs are assigned importance values useful for predicting some output variable. A neural network is a black box forecasting method in the sense that while it can approximate any function, studying its structure doesn't give insight into the structure of the function being approximated (Koh & Liang, 2017). Simple ANNs themselves fail to include long-term dependencies as their previous state is lost once the forecasting window has passed (Medsker & Jain, 1999). Recurrent neural networks (Medsker & Jain, 1999) (RNN) are adaptations of ANNs that use different architecture, attempting to remember temporal patterns present within the information they are provided. RNNs are universal approximators in the sense that given a network with 1 hidden layer they can approximate any continuous function for inputs within a specific range (Schäfer & Zimmerman, 2006). Long Short-Term Memory (Hochreiter & Schmidhuber, 1997) (LSTM) uses an RNN architecture to retain information prior to the forecasting window using its cell state. LSTM is known as a deep learning model as it allows multiple hidden layers to be stacked, giving it depth.

LSTMs (Hochreiter & Schmidhuber, 1997) take as input a 3-D tensor containing [#samples, #time steps, #features]. In our case, the model is provided data which contains a matrix of values that represent time, and features related to bitcoin, like daily market index prices or blockchain data; and are asked to forecast the next day's value. The structure of LSTM involves a cell state, and 3 gates: input, output, forget. The cell state transports information through the sequence chain acting as the memory of the network. The forget gate decides what information to keep by considering the previous hidden state values and the current input X, which is a tensor of features. By transforming the data using a sigmoid function, values are expressed between 0 and 1, with values closer to 0 unable to pass on as much information as they are perceived as less important. The Hadamard product of this sigmoid output and the cell state is taken to effectively 'forget' some of the data. The input gate updates the cell state by taking the Hadamard product of our hidden state's values - run through separate sigmoid and target activation function (usually TanH or ReLU). This provides our cell state with information about which features are important at our current timestep. The cell state is updated by performing the pointwise 'plus' operation on the cell state values that have been transformed by our forget gate and our pointwise i and `s output. output gate decides the next hidden state. The output gate transforms our hidden state values including current input using a sigmoid function to determine how much of the current hidden state's information should be passed onto the next hidden state. The next hidden state is determined by taking the values from the output gate and performing the Hadamard product of this and our recently calculated current cell state. This is run through our activation function (TanH, ReLU) prior to taking the product. Mathematically, LSTM can be defined using the following the equations:

1. $f_t = \sigma_g(W_f x_t + U_f h_{t-1} + b_f)$
2. $i_t = \sigma_g(W_i x_i + U_i h_{t-1} + b_i)$
3. $o_t = \sigma_g(W_o x_t + U_o h_{t-1} + b_o)$
4. $\grave{s}_t = \sigma_g(W_s x_t + U_s h_{t-1} + b_s)$
5. $s_t = f_t \odot s_{t-1} + i_t \odot \grave{s}_t$
6. $h_t = o_t \odot \mathrm{TanH}(s_t)$

where f, i, o: forget, input, output gates. Wx: weight for the respective gate(x) neurons. h_{t-1}: output of the previous LSTM timestamped block. Xt is the input tensor at current timestamp. b is the bias for the respective gates. $\grave{s}_t$: candidate for cell state or memory at timestamp t, s_t: cell state at timestamp t. h_t: hidden state. $\odot$: pointwise multiplication. σ: sigmoid activation function

Or represented diagrammatically:

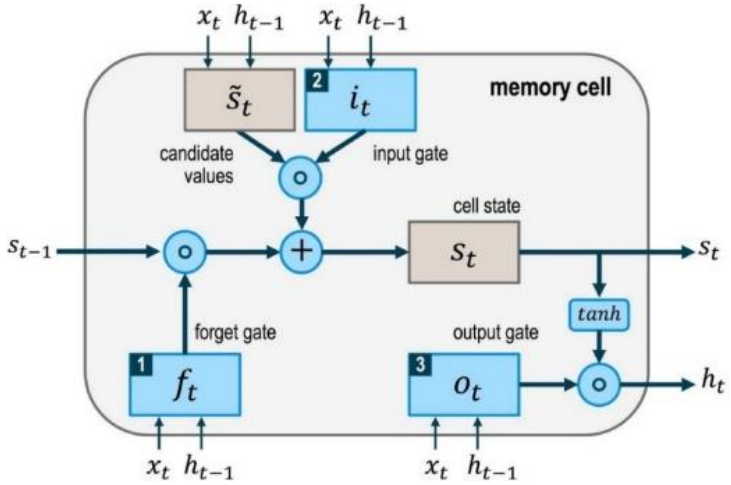


Figure 2. LSTM Memory Cell

Image taken from: Chen, Xu, Jia, & Gao (2021)

Hyperparameters

LSTMs have certain parameters, known as hyperparameters, whose values control its learning process. These include but are not limited to batch size, number of epochs, learning rate, decay rate, network weight initialization, number of neurons in a hidden layer (width), the number of hidden layers (depth), dropout rate, weight constraint, loss function, choice of optimization algorithm, and activation function. Testing different hyperparameter values and comparing the results against models which used Keras default values also allows for analysis into the efficacy of hyperparameter tuning.

3.1.1 Batch Size

Batch size refers to the number of 3-D tensor [#samples, #time steps, #features] samples that are shown to our model before updating its weights. At the end of each batch a forecast of the target variable is made and compared with the actual value resulting in an error value. This error value is passed to our optimization algorithm which attempts to improve our models' forecasts by minimizing the gradient of its function associated with this loss. Small batch sizes make smaller gradient updates and suggests that our data's complexity is captured sufficiently in small quantities. For instance, here I use timeseries price data, so improved model results using a smaller batch size may suggest that shorter trends in price changes are sufficient to capture next day forecasting. During hyperparameter tuning a range of batch sizes was tested (8, 16, 32, 64). The default batch size for Keras is 32 which is used for my base model.

3.1.2 Epochs

Number of epochs refers to how many complete cycles of the training data (split into batches) our model should be shown before it stops learning. The number of epochs used varies based on the task being performed. Letting our models train on training data for too many epochs can lead to overfitting. This is where the model is unable to accurately forecast on unseen test data as it has spent too much time optimising feature weights on training data. During hyperparameter tuning a range of epochs were tested (40, 60, 80). This was explored so that subsequent hyperparameter tuning could benefit from using the optimal number of epochs.

During model testing on training and testing data I specified all models to run for 100 epochs but implement early stopping if no improvement in loss is seen in over 20 epochs.

3.1.3 Learning Rate

Learning rate or step size, refers to value that regulates how much the weights of our model can change. For our model to forecast accurately it must learn the optimal weights for the number of neurons specified. It does this by updating the weights assigned to each neuron depending on the loss that was found from using those weights. For instance, if the loss was high, then the weights must be changed to reduce the loss. Learning rate specifies how much the weights are allowed to update after each batch. Smaller learning rates should theoretically require more epochs to produce the lowest possible loss as their weight updates are smaller. During hyperparameter tuning a range of learning rate values were tested (0.0001, 0.0005, 0.001, 0.005, 0.01). The default learning rate in Keras is 0.001 which is used for my base model.

3.1.4 Learning Rate Decay

During model training it can sometimes be important to decrease the learning rate as training progresses. Decay rate specifies how much the maximum learning rate value should be reduced after each epoch. Decay is important in this regard as it can prevent models from overfitting to training data when they are allowed to train for many epochs. During hyperparameter tuning a range of Decay rates were tested (0.0, 0.01, 0.05). Keras default Learning Rate Decay is 0.0 which was kept for base model testing.

3.1.5 Network Weight Initialization

Network weight initialization refers to the procedure of setting the initial weights for our features to a random distribution, then to be optimised as training takes place. Weight initialization technique can be an important way to ensure our models converge to the lowest loss. Many weight initialization techniques exist (de Sousa, 2016), the default in Keras is Glorot (Xavier) Uniform, which is used for my base model. During hyperparameter tuning a range of Network Weight Initialization techniques were tested (‘glorot uniform’, 'uniform', 'lecun uniform', 'normal', 'zero', 'glorot normal', 'he normal', 'he uniform').

3.1.6 Network Width

Width refers to the number of neurons present in an LSTM layer. Increasing the number of neurons in our hidden layers also increases the number of parameters present within the model, as more neurons being introduced means more weights are introduced. In theory, additional parameters should help our model to represent much more complex data. For stochastically noisy data such as bitcoin price, a wider network may help our models converge to better loss values. A network that is too wide can lead to overfitting. The default value in Keras is 32 neurons which is used for my base model. During hyperparameter tuning a range of values was tested for number of neurons in hidden layer (10, 20, 30, 40, 50, 60, 75, 100).

3.1.7 Network Depth

Depth refers to the number of LSTM hidden layers present within our model. Increasing the number of hidden layers within our model also increases the number of parameters present within the model. Increasing the depth of an LSTM model also increases the complexity required to produce optimal results. In other words, hyperparameter tuning becomes more difficult for deeper networks. Both base models and tuned models contain one hidden layer, allowing for easier analysis into the effects of tuning other hyperparameters.

3.1.8 Dropout Rate

Dropout regularization (Hinton et al., 2014) refers to the number of neurons that are randomly dropped before being fed to the next layer. Randomly dropping neurons ensures that our model sees different neuron configurations for each epoch it trains. The effects of dropout are well known to help prevent models from overfitting to training data. Keras does not have a default dropout value. My base model employs a dropout rate of 0.1, kept in consonance with the study conducted by Dutta et al. (2020). During hyperparameter tuning a range of dropout rates was tested (0.0, 0.1, 0.2, 0.3, 0.4)

3.1.9 Weight Constraint

Weight constraint refers to a function applied to neurons in a hidden layer to limit the values of the weights they are associated with. Maximum norm weight constraint refers to the procedure of forcing the weights to have a magnitude at or below a given limit. The idea of weight constraints is to help prevent overfitting by ensuring weights that were seen as extremely valuable to training data (i.e., high magnitude values) but not necessarily to testing data, are not overemphasized. Keras does not specify a default weight constraint value. Our base model does not use weight constraint. During hyperparameter tuning a range of weight constraint values were tested (1, 2, 3, 4, 5).

3.1.10 Loss Function

Loss refers to the formula used to penalise model forecasts compared with the actual price (Bickel & Doksum, 2015). When we specify a loss function for our model, we are telling it to learn the weights associated with the lowest possible loss. Mean Squared Error (MSE) is the default value set for Keras and is also used for both base model and tuned models.

3.1.11 Activation Function

Activation functions are used in ANN's to interpret the data and pass on information about how important each feature is (Wood, 2021). In this sense, activation functions allow our models to produce nonlinear responses to the data they are provided, something traditional forecasting methods are unable to achieve. LSTM was initially designed with TanH (Saury, 1774) as its activation function. Use of TanH activation function creates problem of vanishing or exploding gradient (Bohra, 2021), where the gradients used to update the weights explode or vanish. A solution is to use the alternative activation function ReLU (Nair & Hinton, 2014). The difference between TanH, Sigmoid and ReLU can be shown graphically, mapping inputs to different values. Mathematically, ReLU limits the vanishing and exploding gradient problem compared to tanh as shown by the graph of their derivative functions. This is shown where ReLU line moves statically from 0 to 1 upon the input data transferring from negative to positive. One problem with ReLU is where neurons have the potential to become zeroed out never to be activated again. This happens when the input is 0, forcing a neuron into passive state. Variants of ReLU such as leaky ReLU (He, Zhang, Ren, & Sun, 2015) have been designed to address this issue where negative values are not assigned 0 gradient values but rather some minute negative value, allowing the model's neurons to continue to update. In this research I build models using ReLU and Tanh activation functions for both base and tuned models.

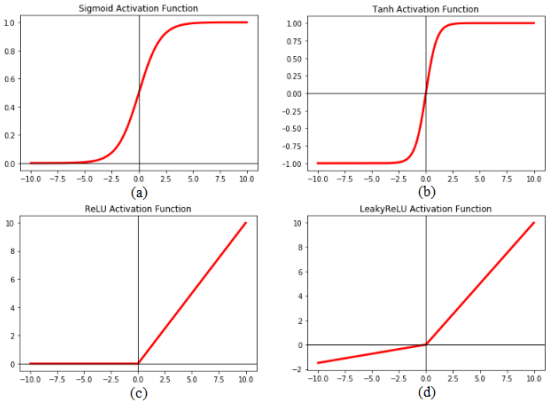


Figure 3. Activation functions
See Appendix A

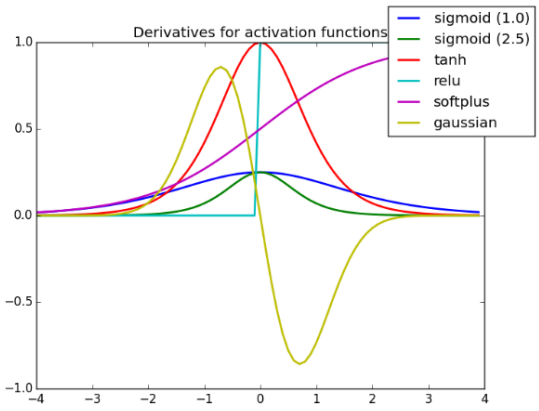


FIgure 4. Derivates for Activation Functions
See Appendix B

3.1.12 Optimizers

Many gradient decent optimizers exist for deep learning tasks. The most popular for financial forecasting seen in literature is adaptive moment estimation (Adam) (Kingma & Ba, 2014). It uses an exponentially decaying average of past gradients and exponentially decaying average of past squared gradients. This gives it an adaptive learning rate. The update equations defining Adam optimizer are shown below:

1. $m=\beta_1 m-(1-\beta_1)\nabla\theta J(\theta)$
2. $s=\beta_2 s+(1-\beta_2)\nabla\theta J(\theta)\odot\nabla\theta J(\theta)$
3. $\grave{}m=m\oslash 1-\beta_1^t$
4. $\grave{}s=s\oslash 1-\beta_1^t$
5. $\theta=\theta+\eta m\hat{}\oslash\sqrt{\grave{}s+\epsilon}$

m denotes the first moment, resembling momentum that records the past normalised gradient. s denotes the estimates of the second moment of the gradients. β beta denotes the initial decay rate used when estimating respective first and second moments of the gradient. θ theta denotes parameter e.g., biases, activations, and weights. J is the objective function. ∇ denotes the gradient of J. $\nabla\theta J(\theta)$ is therefore the rate of change. `m and `s are bias corrected first and second moment estimates that are used to update the parameters as shown by the update rule 5. η denotes the learning rate (eta). ⊘ denotes element-wise division. ⊙ denotes element-wise multiplication.

Adam updates every parameter with an individual learning rate, meaning every parameter in the network has a specific learning rate associated. The single learning rate for each parameter is computed using lambda (the initial learning rate) as an upper limit. This means that every single learning rate can vary from 0 (no update) to lambda (maximum update). The learning rates adapt themselves during training steps. In this research I use adam optimizer with both base and tuned models.

3.2 Data

To implement my LSTM data was needed for it to learn from. The literature cited in Section 2. (Mallqui & Fernandes, 2019) (Sovbetov, 2018) (Yelowitz & Wilson, 2015) (Wolk 2019) (Yildirim & Toroslu, 2021) Suggests that including Cryptocurrency specific data (3.2.1, 3.2.2), Blockchain data (3.2.3), Social sentiment data (3.2.4), Macroeconomic & Traditional Market data (3.2.5), and technical indicator data (3.2.6) will improve the loss scores for my model. To test this, many features from the mentioned categories were collected for this task. Data was collected for period 2014-02-16 / 2022-02-28. This was then truncated to 2017-05-19 / 2022-02-28 for model experiments. The reason for this, was that cryptocurrency data was collected from the Kraken exchange. 2017-05-19 was the day that XRP started trading on Kraken, so all values prior to 2017 were zero.

The data was split into train, test, and validation sets to ensure the models did not over/underfit to the train set. Initially train and test were split into, train (90%), test (10%). Further to this during model training, the training set was further split into, train (90%), validation (10%) which resulted in a total of: train (81%), validation (9%), test (10%).

The following pages provide information for each feature.

Variable	**Description**
3.2.1 Kraken Data	Cryptocurrency Specific Data https://support.kraken.com/hc/en-us/articles/360047124832-Downloadable-historical-OHLCVT-Open-High-Low-Close-Volume-Trades-data [Accessed 09 February 2022]
BTC High	Highest price for bitcoin over a 24-hour period
BTC Low	Lowest price for bitcoin over a 24-hour period
BTC Open	Open price for bitcoin over a 24-hour period
BTC Close	Close price for bitcoin over a 24-hour period
BTC Volume	Volume traded for bitcoin in USD over a 24-hour period
ETH Close	Close price for ether over a 24-hour period
ETH Volume	Volume traded in USD over a 24-hour period
XRP Close	Close price for ripple over a 24-hour period
XRP Volume	Volume traded in USD over a 24-hour period
3.2.2 Kraken Data, Bitcoinity	Cryptocurrency Specific Data https://bitcoinity.org/ [Accessed 09 February 2022]
BTC Trading Volume Kraken Euro	Volume traded from Euro over a 24-hour period

Kraken BTC Volatility	Standard deviation from all market trades
Kraken Bid Ask Spread	Difference of prices quoted for sale and purchase of bitcoin

3.2.3 Blockchain Data

Blockchain Specific Data

https://www.blockchain.com/api/charts_api [Accessed 09 February 2022]

Hash Rate	Estimated number of tera hashes performed per second on the Bitcoin network over the last 24 hours
Difficulty	Measure of the fluctuating difficulty set by the Bitcoin network; Difficulty relating to required computational power to mine blocks
Cost Per Transaction	Miners Revenue divided by number of transactions; Miners Revenue is USD block rewards and transaction fees paid to miners
Output Value Per Day	Total value of all transaction outputs per day, including coins returned to sender as change
Net Value /Transaction	Market Cap divided by total transactions volume in USD over past 24 hours

3.2.4 Google Trends Data

Social Sentiment Data

https://trends.google.co.uk/trends/?geo=GB [Accessed 09 February 2022]

Gtrend bitcoin	Google search interest over time for key word 'bitcoin'
Gtrend bitcoin ban	Google search interest over time for key words 'bitcoin ban'
Grend buy bitcoin	Google search interest over time for key words 'buy bitcoin'

3.2.5 Bloomberg Data

Macroeconomic & Traditional Market Data

https://bba.bloomberg.net/ [Accessed 09 February 2022]

Gold	US Dollars per Troy Ounce: Ticker XAUUSD
Silver	US Dollars per Troy Ounce. Ticker XAGUSD
Crude Oil Brent	Futures contract in USD. Ticker: COK2 Comdty
US Bond Price 2 yr	US Dollars Price on a 2-year Treasury (BCLASS) bond. Ticker: CT2 Govt
US Bond Price 5 yr	US Dollars Price on a 5-year Treasury (BCLASS) bond. Ticker: CT5 Govt

US Bond Price 10 yr	US Dollars Price on a 10-year Treasury (BCLASS) bond. Ticker: CT10 Govt
SP500	Index that includes 500 leading US companies and captures approximately 80% coverage of available market capitalization. Ticker: SPX Index
VIX	Chicago Board Options Exchange Volatility index – Up-to-the-minute market estimate of the expected volatility of the S&P 500 Index. It is the midpoint of real-time S&P 500 Index (SPX) option bid/ask quotes. Ticker: VIX Index
NASDAQ	NASDAQ Composite Index is a broad-based capitalization-weighted index of stocks in all three NASDAQ tiers: Global Select, Global Market and Capital Market. Ticker: CCMP Index
Dow Jones	Dow jones Industrial Average is a price-weighted average of 30 blue-chip stocks that are generally the leaders in their industry. Ticker: INDU Index
FTSE 100	Capitalization-weighted index of the 100 most highly capitalized companies traded on the London Stock Exchange. Ticker: UKX Index
Euronext	Euronext 100 index is market-cap weighted index of the 100 largest and most liquid stocks traded on Euronext. Ticker: N100 Index
USD EURO XRATE	The Spot Exchange Rate between USD and Euro – Price of 1 USD in EUR. Ticker: USDEUR BGN Curncy
US Dollar Index	Average of exchange rates between USD and major world currencies, rates supplied by some 500 banks. Ticker: DXY Curncy

---

3.2.6 Technical Indicator Data	Technical Indicator Data https://www.tradingtechnologies.com/xtrader-help/x-study/technical-indicator-definitions/list-of-technical-indicators/ [Accessed 09 February 2022]
daily_return	Percentage return of BTC Close Kraken Data
cum_daily_return	Cumulative returns of BTC Close Kraken Data
rsi	Current price normalized as a percentage between 0 and 100: 100-(100/(1+RSI))
Williams %R	Similar to stochastic oscillator, normalizes price as a percentage between 0 and 100: -100*((High-Close)/(High-Low))
ma7	Mean price over a rolling window of 7 days: (Sum(Price,n))/n
ma21	Mean price over a rolling window of 21 days: (Sum(Price,n))/n

ema_26	Exponential moving average over 26 days, represents an average of prices but places more weight on recent prices: (P-EMAp)*K+EMAp P=Price, EMAp=Exponential Moving Average for previous period, K=Smoothing constant, equal to 2/(n+1), n=number of periods
ema_12	Exponential moving average over 12 days.
MACD	Moving Average Convergence Divergence is the difference between our ema_26 and ema_12.
ema	Exponential moving average with decay = 0.5. Decay alpha = 1/(1+com). Decay is raised to the power n where n increases by 1 for each time period looking backwards. In this sense decay penalises older information
Momentum	Simple 3 day momentum: Price – Price of n periods ago
bb_high	Bollinger Band high, window=21, std=2: (rolling mean+(rolling std*number of std))
bb_low	Bollinger Band low, window=21, std=2: (rolling mean-(rolling std* number of std))
stok	Stochastic Oscillator K, n=14 days: SMA(100SMA(((Close-Low)/(High-Low)),n)),Kma). Kma=Period of Moving Average used to smooth
stod	Stochastic Oscillator D, n=14 days: SMA(stok,Dma). Dma=Period of Moving Average used to smooth stok
ROC	Price Rate of Change 12 days, compares current price with the previous price from selected number of periods ago: (Current Price/Price of n bars ago)-1.0)*100. n=Time period
CCI	Commodity Channel Index compares current mean price with the average mean price over a typical window of 20 periods. (M-A)/(0.015*D). M=(H+L+C)/3, H=High, L=Low, C=Close, A=n period moving average of M, D=mean deviation of absolute value of difference between mean price and moving average of mean prices, M-A
TEMA	Triple Exponential Moving Average offers moving average with less lag than traditional exponential moving average: EMA1=EMA(t,period), EMA2=EMA(EMA1,period), EMA3=(EMA(EMA2,period)), TEMA=3*EMA1-3*EMA2+EMA3
turning_line	Part of Ichimoku, the turning line is: (High+Low)/2, for past 9 days
standard_line	Part of Ichimoku, the standard line is: (High+Low)/2, for past 26 days
ichimoku_span1	(Standard Line + Turning Line)/2, plotted 26 days ahead of today
ichimoku_span2	(High+Low)/2, for past 52 days, plotted 26 days ahead of today

3.3 Experimental Design

Design involved a choice of pre-processing, feature selection technique, model evaluation method, and choice of hypothesis test. **Data pre-processing** was first applied to ensure that the data being fed to the model was of the appropriate form.

3.3.1 Data Pre-processing

Data was normalized by first checking for missing values. Project_Set_MarketData.csv contained blank cells for weekends which was backfilled in Python i.e., filled using the previous days values. No other data was missing. Data was then scaled using a minmax scaler between (0,1) as this was the required values for our LSTM model. Mathematically, Min Max Scaler can be described as:

$$X_{sc} = (X - X_{min}) \,/\, (X_{max} - X_{min})$$

Where X_{min} and X_{max} are the smallest and largest values in the data set. As such X_{min} will be scaled to value 0 and X_{max} will be scaled to value 1.

3.3.2 VIF

The feature reduction technique used in this research was Variance Inflation Factor (VIF). VIF is a measure used to detect multicollinearity in regression analysis (James, Witten, & Hastie, 2013). High multicollinearity between two predictor variables suggests that the variance in our dependent (prediction) feature that they capture, may be sufficiently explained by omitting one of the features. VIF is more often used in traditional linear regression analysis where high multicollinearity can lead to spurious forecasting results. The application VIF feature reduction for LSTMs should be questioned as LSTMs do not produce linear forecasting responses to data.

VIF uses R^2 goodness of fit as its measure of feature correlation. R^2 is calculated by regressing a target variable against all other variables and represents the percentage of variance in each feature that the set of features explains. As R^2 is a denominator variable that is subtracted from 1, any increase in its value represents increased multicollinearity, and results in an increase in VIF score. R^2 gets plugged into the VIF formula such that mathematically, VIF can be written as:

$$VIF = 1 \,/\, (1 - R^2_i)$$

Research suggests that any VIF score over 10 should be omitted from use in regression models (Craney & Surles, 2002). My process of implementing VIF feature reduction experimentally was to first split our data into groups that appear similar and collect VIF scores, eliminating any scores over 10, one by one, largest each time. The categories identified were (3.2.4) = Social Sentiment Data, (3.2.6) = Technical Indicator Data, (3.2.5) = Macroeconomic & Traditional Market Data, (3.2.3) = Blockchain Specific Data, (3.2.1) (3.2.2) = Cryptocurrency Specific Data. The reason I separated data into these categories was that they are groups whose features are known to influence each other's values. Once this was completed data was wrapped back into one data frame. Wrapping the data together caused VIF scores for certain features to increase above 10 again where they had previously been below 10 in their prior respective 'grouped' sets. The wrapped group also had any scores above 10 eliminated one by one until a set of 26 features was returned. It should be noted that VIF was conducted for features prior to scaling and over the entire time-period, not split into train and test. The final VIF features scores are shown in Table 1, where VIF score denotes the R^2 value of that feature from regressing all features against bitcoin close price, input to the equation above:

Table 1. VIF scores for selected 26 features

Variable	VIF Score	Variable	VIF Score
Intercept	0.0000	XRP Volume	1.4854
Gtrend buy bitcoin	2.8491	Roc	3.9255
Gtrend bitcoin ban	2.1334	Momentum	1.6522
Daily return	1.4243	Tema	4.5457
Rsi	7.2251	Crude Oil Brent	7.3318
Williams % R	3.7281	VIX	1.8750
MACD	2.4292	US 2-year bond price	7.2674
Stod	4.1069	FTSE100	9.1586
Output Value Per Day	1.3649	USD EURO XRATE	1.4536
Net Value to Transactions	1.3464	Hash Rate	5.7458
BTC Trading Volume Kraken Euro	4.4658	Kraken Bid Ask Spread	4.1622
Kraken BTC Volatility	5.6074	BTC Volume	5.3386
ETH Volume	2.2520	XRP Close	2.4006
BTC Close	Target		

3.3.3 Model evaluation:

Models were evaluated using the error metric - Root Mean Squared Error (RMSE), which can be described mathematically as:

$$\sqrt{((1/n) * \Sigma (\text{actual} - \text{forecast})^2)}$$

RMSE is a quadratic error metric that measures the average magnitude of the loss. The difference between our forecast and observed price are squared and then averaged over the sample, the square root of this is then taken. RMSE gives a high weight to large errors since the errors are squared before they are averaged. As we are forecasting on volatile test data, using RMSE is my preferred choice of error metric as it heavily penalises large errors.

A simple buy / sell strategy was implemented to compare the results of RMSE with profit / loss calculations, the strategy performs:

1. Buy: when predicted price tomorrow is greater than predicted price was today
2. Don't Buy: when predicted price tomorrow is less than predicted price was today

The trading strategy that was utilised here is a heuristic. The logic of the strategy involves buy/sell trades and assumes a selling of all assets at the end of the day with subsequent re-evaluation of buying/selling positions once the next prediction is returned. Ultimately, the

summed value of all the single daily trades profit/loss indicates whether the strategy was profitable or not over the testing timeframe. The returns that are described are simple returns and do not account for compound profits i.e., the holding of an asset for more than one day if two consecutive positive predictions are returned. It also fails to account for any transaction fees. Close price 'bought' is taken to be the strike price that the exchange offers while this may not actually be the case.

The reason for making trades using the previous forecasted value rather than the previous actual price is that larger feature sets appear to follow the price less accurately. By using the forecasted price of yesterday we are seeing if the model's sequential forecasts are predicting changes in the behaviour (direction) of the actual price, irrespective of whether it correctly forecasted the price to be higher or lower than the actual price. In other words, using this strategy somewhat illustrates whether model predictions are just lagged predictions of the actual price. As we are looking at a testing period where bitcoin price decreases substantially, we could set a benchmark profit/loss to be the decrease in price from start to finish as this would represent holding the asset over the time frame. There is a downward trend in bitcoin price for our test data. A buy/sell strategy can only make money when there is both an actual increase in price and a predicted increase in price (1/4 of cases). Therefore, I do not expect models to return positive profit/loss. This strategy has not been implemented with the intention as being the most profit reaping. It is meant to illustrate the lagged behaviour of predictions using profit/loss as its criterion for measurement. Most asset forecasting literature does not employ profit/loss calculations, so although I wrote code for a long/short strategy, I have only included buy/sell, to allow me to cross-examine the efficacy of RMSE model evaluation from a practical standpoint.

3.3.4 Diebold Mariano Statistical Test

The Diebold Mariano Statistical Test is a hypothesis test used to evaluate the similarity of two sets of forecasts against a predictor variable (Diebold & Mariano, 2002). It is a measure of the autocovariance of two arrays of loss values. Two sets of predictions are generated which creates two sets of errors. The difference between these two errors is taken. If the difference in losses follows a normal distribution, they are said to be equally accurate on average. The error metric used to perform the statistical tests was Mean Squared Error (MSE) which was used in consonance with our test error metric and with our model's loss criterion. The test follows a student-T distribution with degrees of freedom (T−1). The null hypothesis of the test H0, is that there is no difference between the expected loss of the two models' forecasts. The alternative hypothesis H1, is there is a difference expected between the losses that the models have forecasted. As we are performing a 2 tailed test the 1% critical Z-score value is 2.58 (Sjsu.edu. 2007). This value is weighted against our DM test statistic. If the absolute value of the test statistic is greater than our Z-score then H0 is rejected, and we assume that the loss distribution of the two models' forecasts is expected as different with 99% confidence. It should be noted that although such testing statistically proves a difference in the expected forecasting loss, generalising these results to model performance is not suggested. This is because the models we are running initialise weights randomly and therefore model performance can change each time they are run. For instance, a model whose results are better and is proven to be expected as different from another model's loss whose scores were worse does not imply its scores will be better every time they are tested.

4. Model Experiments

Part 1. Base Model

Base Model experiments used an LSTM architecture with one LSTM layer (input/hidden) and one dense (output) layer. It had 32 neurons in its hidden layer, and used 30 one day previous timesteps, also known as window size, to forecast the next bitcoin close price. 30 days was chosen as being congruous with a trading calendar month. It used adam optimisation and had dropout rate of 0.1, which was implemented in consonance with the value used in Dutta et al's research (2020), intended to avoid overfitting. The loss criterion specified was Mean Squared Error (MSE). Batch size was chosen to be 32, the models ran for 100 epochs but employed early stopping which halted the training if no improvement in loss was found for over 20 epochs.

Models were trained on 1570 days of data that covered a period from 2017-05-19 / 2021-09-07. With a 30 day look back period this gave 1540 forecasting predictions. During training, the validation loss was recorded, and models were set to use the weights that minimised the loss of the validation set. The validation set encompassed the final 10% of the training data which was 157 days period 2021-04-03 / 2021-09-07. As the validation period also encompassed a period of high volatility, setting models to optimise their weights based on this validation period was implemented to produce the best results on our unseen test data. The weights that were used to produce the lowest validation loss for all epochs were saved to file and loaded for our test data. Test data encompassed a 173-day period from 2021-09-08 / 2022-02-28, with a forecasting window of 30 this gave 143 forecasts. Two base model architectures were run for each feature set – base model with 'ReLU' activation function, and base model with 'TanH' activation function. The results for train, test, and generalisation gap RMSE scores on aggregated scale inversed bitcoin forecasting is found in Table 2 below:

Table 2. Aggregated base model RMSE scores

LSTM Base Model	Train RMSE	Test RMSE	Generalisation Gap
1 feature TanH	986.5972	1776.6675	790.0703
1 feature ReLU	1051.1418	1988.3953	937.2535
26 features TanH	1331.8290	2912.5302	1580.7012
26 features ReLU	1387.6342	2705.5946	1317.9604
58 features TanH	1216.3092	8141.6649	6925.3557
58 features ReLU	989.1622	3549.8430	2560.6808

Note: RMSE values are scaled inversed values from respective train / test sets. Train RMSE covers a 1540-day period from 2017-06-19 / 2021-09-06. Test RMSE covers a 143-day test period 2021-10-08 / 2022-02-28. Generalisation Gap refers to the difference between train and test RMSE. Large generalisation gap values imply a model's inability to generalise the weights learned from training to unseen test data.

Referring to train RMSE scores, all models performed similarly. Train RMSE scores were overall impressive. Models using 26 features were the worst performing on train RMSE irrespective of activation function. 1 feature TanH and 58 features ReLU performed the best on train RMSE. Referring to test RMSE scores, models using 1 feature were the best performing shown where they had significantly lower test RMSE scores that models that included more features. Although models using 26 features were the worst on train RMSE they outperformed models using the larger 58 features sets on test RMSE. Models using 1 feature performed the best overall as they had the lowest scores on train and test sets which translated to lower generalisation gap scores. This suggests that models using 1 feature are better able to generalise the weights learned on training data to unseen volatile test data. Models using 58 features struggled to generalise the weights learned in training to unseen test data. This suggests that including too many features has an adverse effect on forecasting unseen test data. The huge generalisation gap of models using 58 features was somewhat reduced by performing VIF feature selection. This may suggest

that reducing multicollinearity in our features helps models to generalise the weights learned in training data to unseen test data thus preventing overfitting.

The lowest test RMSE forecasting loss found in both Part 1. and Part 2. was validated using Diebold-Mariano (DM) (MSE) hypothesis testing (Diebold & Mariano, 2002). By testing the lowest test RMSE against every other model forecast from that part we accept/reject that its values were expected as different. Each feature set in Part 1. And Part 2. also had DM testing applied for TanH vs. ReLU pairs to check for differences in the forecasting distributions of different activation function. Further to this, Tuned Model vs. Base Model DM testing was applied to check for differences prior to and after conducting hyperparameter tuning. The DM test statistic here is our best model (1 feature TanH) weighted against each other respective model. The result for each model is shown in the Table 3, below.

Table 3. Base Model Results

Base LSTM: Feature set with activation	RMSE (scale inversed test data) (4 d.p)	Diebold Mariano test statistic	Profit / Loss ($)
1 feature TanH	1776.6675	Target	-13,560.51
1 feature ReLU	1988.3953	-3.0141311	-13,701.31
58 features TanH	8141.6649	-26.71882023	-8,892.46
58 features ReLU	3549.8430	-7.57908073	-3,963.89
26 features TanH	2912.5302	-6.62927876	-8,774.97
26 features ReLU	2705.5946	-6.43204932	-4,387.18

Note: RMSE scores are scaled inversed forecasting values from trained model on unseen test data. Scores are aggregated over a 143-day test period 2021-10-08 / 2022-02-28

The best model in terms of test RMSE forecasting was 1 feature TanH. This result does not conform to the literature referenced above which suggested that including additional features should improve model RMSE scores. VIF feature selection improved model loss scores compared to 58 features implying a positive effect on model forecasting. As RMSE is my evaluation method of choice, I concluded that 1 feature TanH was the best model and performed DM tests for every other base model forecast against its forecasts. Referring to DM test statistic results, we reject the null hypothesis H0 on every occasion and thereby assume that the distribution of our forecasts for our 1 feature TanH model were expected as different to all other model forecasts. The best model in terms of profit/loss was 58 features ReLU, however all models made a loss over the testing period. This model also had the second worst RMSE. For both 58 features and 26 features, ReLU outperformed TanH in terms of both RMSE and profit/loss. This was the opposite for 1 feature base model testing where TanH outperformed ReLU for both RMSE and profit/loss. As these results were run on a base model it is hard to draw insight from their results. It appears including additional features improved the model's ability to predict behavioural changes in price which resulted in better buy/sell decisions. However, the inclusion of these features caused our models to incorrectly follow the actual price shown by their inferior test RMSE scores.

The diagram below visualises the base model results for our 143-day testing period:

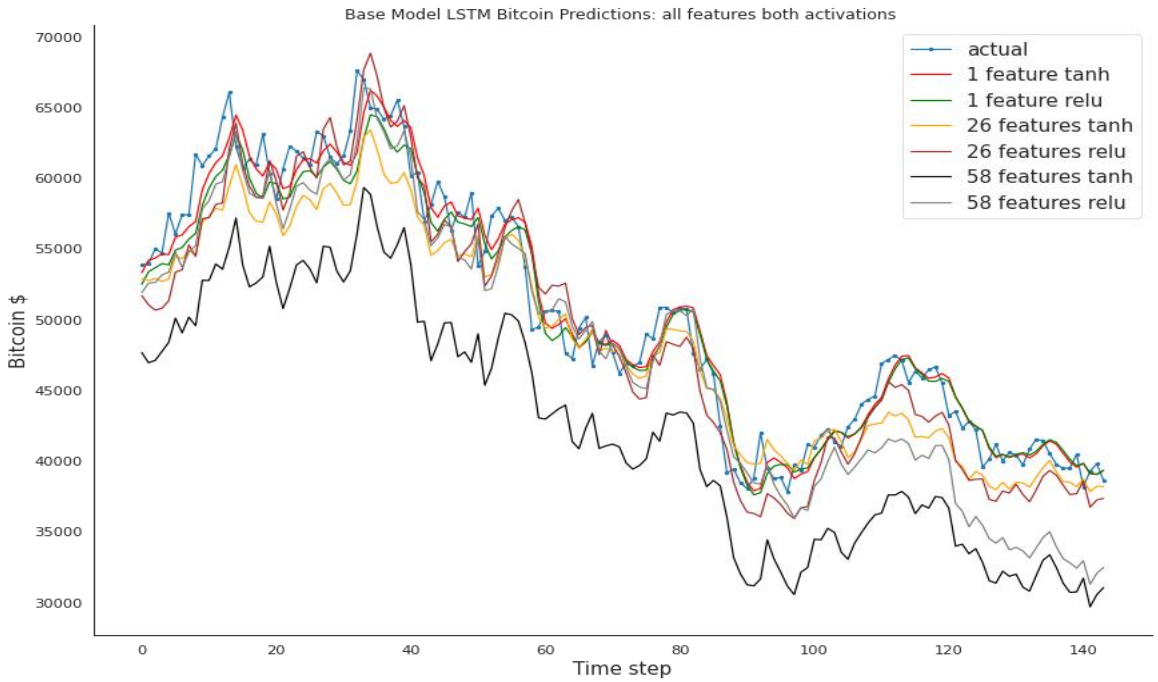


Figure 5. Base Model Forecasting

Note: Graph shows daily price of bitcoin + forecasting values for trained hyperparameter tuned models on unseen test data and covers a 143-day test period 2021-10-08 / 2022-02-28, Bitcoin values are $ values.

The superior RMSE scores of 1 feature models can be seen, where they manage to follow the price of bitcoin closely, even on volatile unseen data, for the whole period. 58 features performed well on training data but overfit to the weights learned from training and clearly failed to follow the price of bitcoin. 58 features ReLU appears to follow the price for a while but towards the end of the testing period dramatically tails off. 26 feature models performed comparably with 1 feature forecasts but starts to underrepresent the values after the large dip in price following timestep 80.

To further investigate the effects of TanH vs ReLU I conducted hypothesis testing on TanH vs. ReLU Base Models for each feature set, the results are shown in Table 4, below:

Table 4. Base Model results for activation pairs

Base LSTM: Feature set with activation	RMSE (scale inversed test data) (4 d.p)	Diebold Mariano test statistic	Profit / Loss ($)
1 feature TanH	1776.6675	Target	-13,560.51
1 feature ReLU	1988.3953	-3.0141311	-13,701.31
26 features TanH	2912.5302	Target	-8,774.97
26 features ReLU	2705.5946	-1.43686854	-4,387.18
58 features TanH	8141.6649	Target	-8,892.46
58 features ReLU	3549.8430	-26.02781731	-3,963.89

Note: RMSE scores are scaled inversed forecasting values from trained model on unseen test data. Scores are aggregated over a 143-day test period 2021-10-08 / 2022-02-28

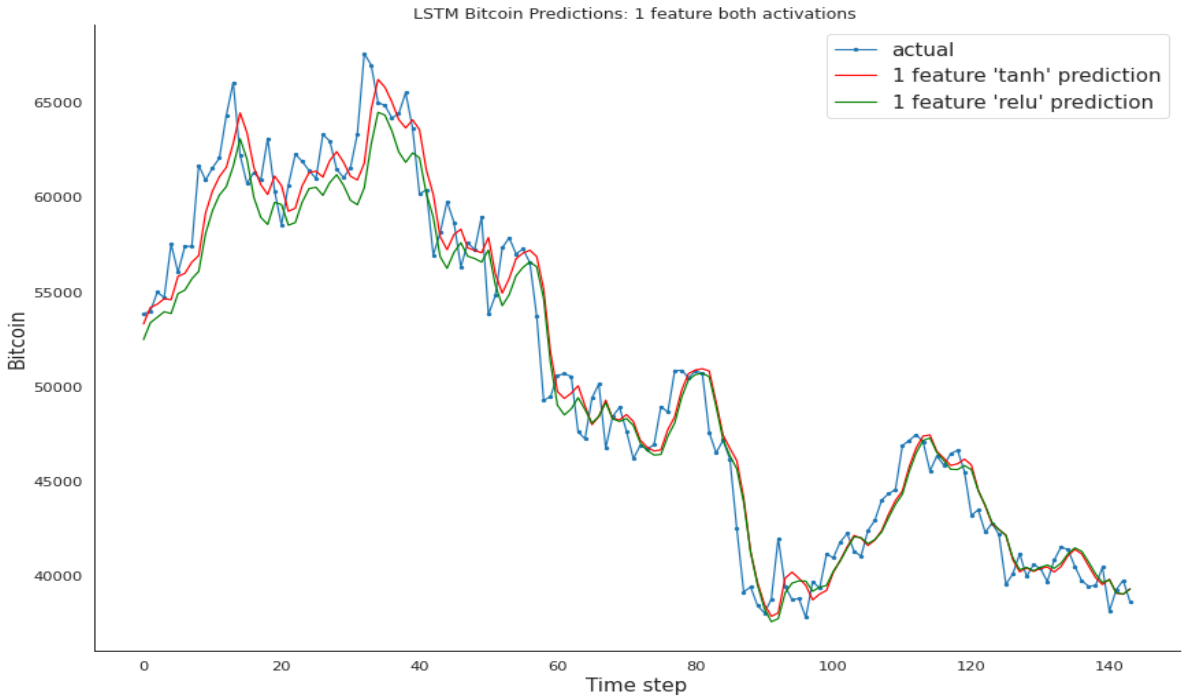


Figure 6. Base Model 1 feature TanH vs. ReLU

Note: Graph shows daily price of bitcoin + forecasting values for trained 1 feature base models on unseen test data covering a 143-day test period 2021-10-08 / 2022-02-28, Bitcoin values are $ values.

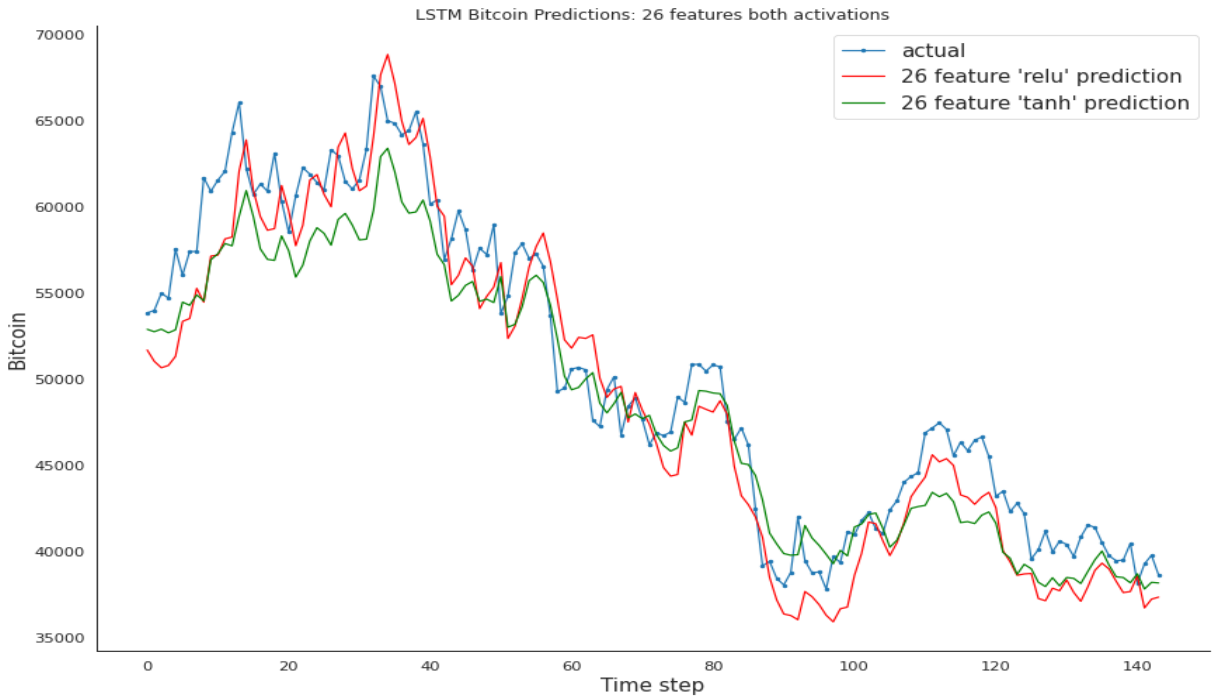


Figure 7. Base Model 26 features TanH vs. ReLU

Note: Graph shows daily price of bitcoin + forecasting values for trained 26 features base models on unseen test data covering a 143-day test period 2021-10-08 / 2022-02-28, Bitcoin values are $ values

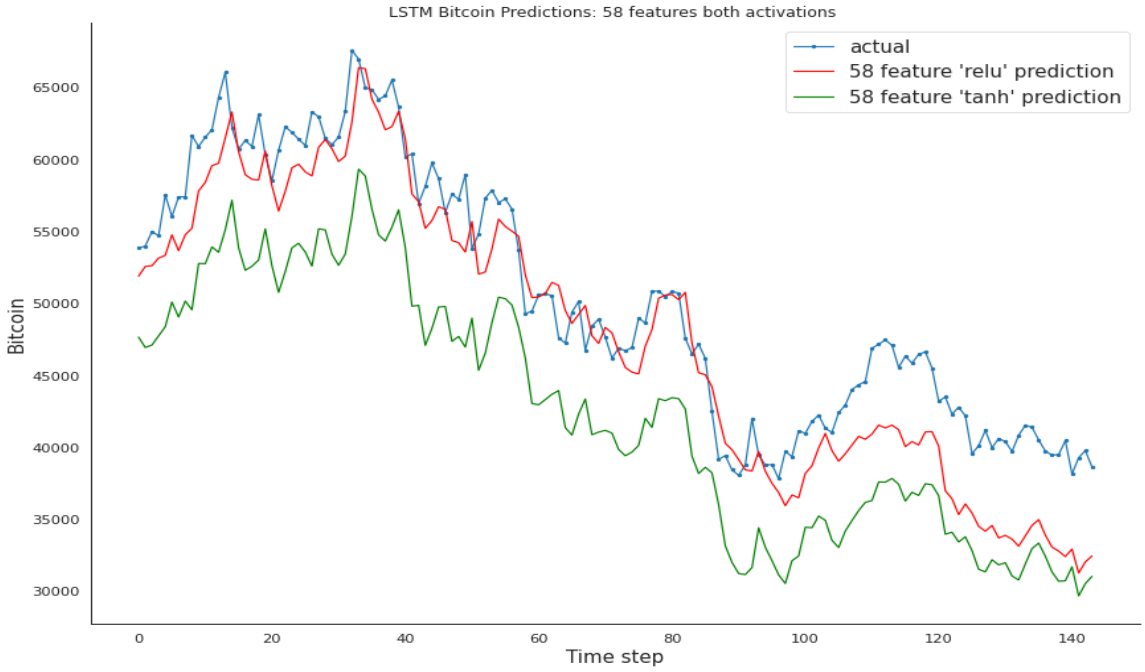


Figure 8. Base Model 58 features TanH vs. ReLU

Note: Graph shows daily price of bitcoin + forecasting values for trained 58 features base models on unseen test data covering a 143-day test period 2021-10-08 / 2022-02-28, Bitcoin values are $ values.

At first glance ReLU appears to be a better activation function for both 58 features and 26 features in terms of both RMSE and profit/loss. 1 feature forecasting has the best RMSE scores with TanH being superior. ReLU was significantly better in terms of profit/loss for both 26 and 58 features. Larger feature sets did better in terms of anticipating price change signals. Both 1 feature TanH vs. ReLU and 58 features TanH vs. ReLU have the absolute value of their DM test statistics above 2.58. Therefore, we fail to accept the null hypothesis and assume the distribution of their forecasting losses is expected to be different. 26 features TanH vs. ReLU had a DM test statistic of -1.43 (2.d.p), which meant that the null hypothesis here was accepted thereby assuming that the difference between their losses follows a normal distribution. This suggests their forecasting accuracies were similar in terms of MSE although they had significantly different profit/loss scores. One potential reason for this could be due to VIF reducing multicollinearity between features. Our 58 features TanH vs. ReLU test showed a DM test statistic of -26.03 (2.d.p) which meant the null was rejected. The models have very visible differences in forecasting. 58 TanH consistently underrepresented the price, whereas 58 ReLU followed the price but diverged after the large dip in price at around timestep 80. Base model results should not be generalised as there may be hyperparameters associated that are ineffective for forecasting that feature set.

Part 2. Tuned Model

For Tuned Model experiments to take place hyperparameter tuning was first applied. The hyperparameter tuning technique implemented was GridSearchCV (Liashchynskyi & Liashchynskyi, 2019). It works by exhaustively trying all the combinations of values passed in from a Python dictionary to the model. It is named Grid Search as it involves systematically testing more than one value of each parameter and often more than one parameter, hence creating a grid of values to evaluate. CV stands for (K fold) Cross-Validation (Refaeilzadeh et al. 2009) and works by splitting the training data into several sub-groups K and testing the performance of each. For instance, if we specify K = 5 we split the data into 5 groups and use 4 of those groups to evaluate the remaining unseen group. The mean result of the 5 periods we

test is returned and the best score out of our grid of parameters and parameter values is taken to be the optimal hyperparameter configuration. Cross-validation visualised diagrammatically:

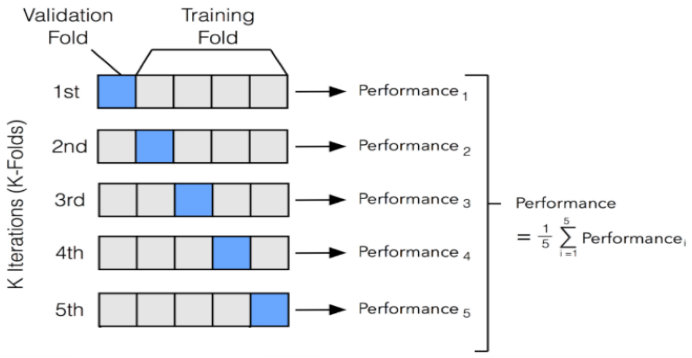


Figure 9. Cross Validation

Image taken from http://ethen8181.github.io/machine-learning/model_selection/model_selection.html [Accessed 27 April 2021]

Weight initialization and number of neurons in the hidden layer were tuned individually. All other hyperparameters were tuned in pairs: Batch size & number of epochs, decay & learning rate, dropout rate & weight constraint. Batch size was tested with number of epochs as often smaller batch sizes need more epochs to converge to optimal loss values. Decay has a direct effect on learning rate values and so were trained together. Dropout rate and weight constraint were trained together as they can both be methods used to prevent overfitting. The order in which the hyperparameters were tuned is the order that they appear in the table below. Each subsequent test utilised the values found in the previous test. The values tested were: Batch size=[8, 16, 32, 64], Epochs=[40, 60, 80], Learning rate=[0.0001, 0.0005, 0.001, 0.005, 0.01], decay=[0.0, 0.01, 0.05], Weight Initialization=[‘glorot uniform’, 'uniform', 'lecun uniform', 'normal', 'zero', 'glorot normal', 'he normal', 'he uniform'], Dropout rate=[0.0, 0.1, 0.2, 0.3, 0.4], Weight constraint=[1, 2, 3, 4, 5], Neurons=[10, 20, 30, 40, 50, 60, 75, 100]. Batch size 64 was omitted for ReLU testing as it meant performing 3 extra cross validations (we tested 3 epoch values) which were not supported by cuDNN (NVIDIA, 2022), and therefore were too time consuming. None of the TanH testing conducted prior had found any feature set size to perform optimally using batch size 64. Decay=0.005, Weight constraint=5, Dropout rate=0.4 were all also omitted from ReLU testing due to time constraints. Further to this, 26 features ReLU, which was found to perform optimally with batch size 8 and epochs 80, had additional features removed as it performed extremely slowly. The values further removed were Dropout rate=0.2 & 0.3. Table 5, below shows each models configuration after hyperparameter tuning:

Table 5. LSTM Hyperparameter Tuning Results

Hyperparameter	1 feature TanH	1 feature ReLU	58 features TanH	58 features ReLU	26 features TanH	26 features ReLU
Batch Size	8	32	16	16	8	8
Epochs	80	40	80	60	80	80
Learning rate	0.0005	0.01	0.001	0.005	0.005	0.001
Decay	0.0	0.01	0.0	0.0	0.01	0.0
Weight Initialization	Glorot Normal	Glorot Uniform	Lecun Uniform	Normal	Glorot Uniform	Glorot Uniform
Dropout rate	0.0	0.0	0.0	0.0	0.0	0.0
Weight constraint	1	2	2	1	2	3
Number of Neurons	100	30	20	20	40	60

After obtaining hyperparameters for each our 6 models they were re-run. Like our base model testing each model was allowed to run for 100 epochs with early stopping employed which halted the training if no improvement in loss was found after 20 epochs. The weights associated with the epoch that had the lowest validation loss were saved to file and used for our test period. The same training, validation and testing period were used for both base model and hyperparameter tuned models to ensure a fair comparison of results. Table 6, below shows the train, test, generalisation gap results for each Tuned model:

Table 6. Aggregated Tuned model RMSE scores

LSTM Tuned Model	Train RMSE	Test RMSE	Generalisation Gap
1 feature TanH	887.3095	1733.1608	845.8513
1 feature ReLU	941.9307	1830.0265	888.0958
26 features TanH	1012.5387	2172.5266	1159.9879
26 features ReLU	993.0350	2310.7409	1317.7059
58 features TanH	278.1077	3228.2565	2950.1488
58 features ReLU	230.1149	1797.8072	1567.6923

Note: RMSE values are scaled inversed values from respective train / test sets. Train RMSE covers a 1540-day period from 2017-06-19 / 2021-09-06. Test RMSE covers a 143-day test period 2021-10-08 / 2022-02-28. Generalisation Gap refers to the difference between train and test RMSE. Large generalisation gap values imply a model's inability to generalise the weights learned from training to unseen test data.

Referring to train RMSE, models using 58 features significantly outperformed other models, suggesting that hyperparameter tuning had a positive effect on models that include surplus features for train RMSE scores. Models using 26 features again had the worst train RMSE scores. It appears that VIF selection was not effective at getting models to perform optimally on training data. Referring to the test RMSE, models using 1 feature performed the best again after hyperparameter tuning. Models using 58 features again had the worst generalisation gap scores, although these scores significantly decreased from our base model results. This reiterates the findings from base models that 58 features sets are overfitting, even with hyperparameter tuning implemented. But does suggest that hyperparameter tuning had the effect of reducing overfitting, just not eliminating it. Tuned model 58 features ReLU had comparable RMSE scores to tuned 1 feature models on unseen test data. This was not the case with base model testing,

which suggests that hyperparameter tuning was an effective way to improve forecasting loss on large feature sets. As model 58 features ReLU had significantly lower train RMSE scores compared to models using 1 feature it suggests this model overfit to the training data and may be able to further improve its test scores. Potentially, a more exhaustive list of hyperparameters being tested could have produced better results for model 58 features ReLU. If this were the case, literature which espouses that including additional features has an improved effect on RMSE scores would be correct.

Table 7, below shows test RMSE scores, DM test statistics, and profit/loss for tuned models:

Table 7. Tuned Model Results

Tuned LSTM: Feature set with activation	RMSE (scale inversed test data) (4 d.p)	Diebold Mariano test statistic	Profit / Loss ($)
1 feature TanH	1733.1608	Target	-493.58
1 feature ReLU	1830.0265	-1.72412567	-11,915.61
58 features TanH	3228.2565	-10.6029162	4,551.21
58 features ReLU	1797.8072	-4.5193411	-2,455.19
26 features TanH	2172.5266	-4.46808154	-3,035.18
26 features ReLU	2310.7409	-3.95612066	-11,104.68

Note: RMSE scores are scaled inversed forecasting values from trained model on unseen test data. Scores are aggregated over a 143-day test period 2021-10-08 / 2022-02-28

The best model in terms of RMSE was again, 1 feature TanH. This model was also the second best in terms of profit/loss with a loss of $-493.58. The second-best model in terms of RMSE was 58 features ReLU. The best model in terms of profit/loss was 58 features TanH which conversely had the worst RMSE. This was the only model which made a profit, amounting to $4,551.21. The findings here oppose the opinions of current forecasting literature which claim that including additional features improved test forecasting loss. However, the results do suggest that including additional features does have an improved effect on a buy/sell strategy. It may be that the limited number of values used to tune hyperparameters or the hyperparameter tuning method was ineffective and that further tuning would improve our larger feature sets RMSE beyond 1 feature forecasts. Results were again validated using DM hypothesis testing. The only case where the null hypothesis was accepted was 1 feature TanH vs. 1 feature ReLU, suggesting that the distribution of forecasts between 1 feature TanH / ReLU was not expected as different.

The diagram below visualises the tuned model results for our 143-day testing period:

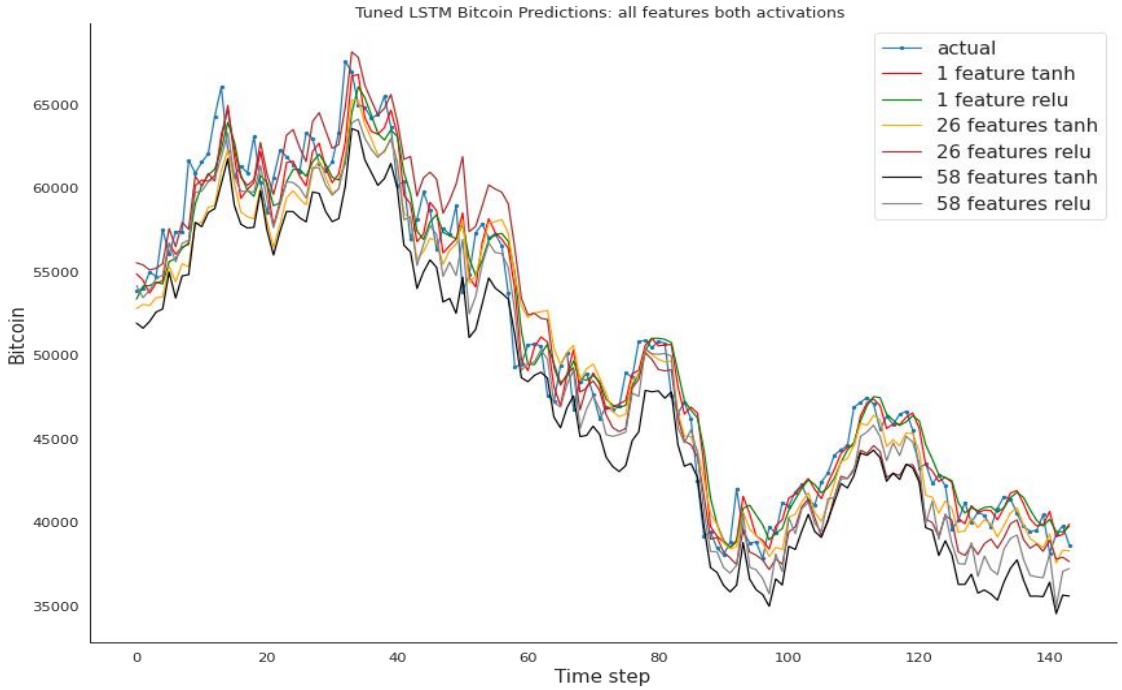


Figure 10. Tuned Model Forecasting

Note: Graph shows daily price of bitcoin + forecasting values for trained hyperparameter tuned models on unseen test data and covers a 143-day test period 2021-10-08 / 2022-02-28, Bitcoin values are $.

The superior RMSE scores of 1 feature models can once again be seen, where they manage to follow the price of bitcoin closely, even on volatile unseen data, for the whole period. 58 features performed extremely well on training data but overfit the weights learned from less volatile data again and clearly failed to follow the price of bitcoin, systematically underrepresenting its value throughout the testing period. 58 features ReLU again appears to follow the price more closely for a while but towards the end of the testing period diverges away from the actual price. 26 features ReLU was the only model that appeared to overestimate the actual price which can be seen where forecast price was significantly higher between timestep 30 – 70.

To further investigate the effects of TanH vs ReLU I conducted hypothesis testing on both TanH and ReLU Tuned Models for each feature set, the results are shown in Table 8 below:

Table 8. Tuned Model Results TanH vs. ReLU

Tuned LSTM: Feature set with activation	RMSE (scale inversed test data) (4 d.p)	Diebold Mariano test statistic	Profit / Loss ($)
1 feature TanH	1733.1608	Target	-493.58
1 feature ReLU	1830.0265	-1.72412567	-11,915.61
26 features TanH	2172.5266	Target	-3,035.18
26 features ReLU	2310.7409	-0.90636125	-11,104.68
58 features TanH	3228.2565	Target	4,551.21
58 features ReLU	1797.8072	13.93744403	-2,455.19

Note: RMSE scores are scaled inversed forecasting values from trained model on unseen test data. Scores are aggregated over a 143-day test period 2021-10-08 / 2022-02-28.

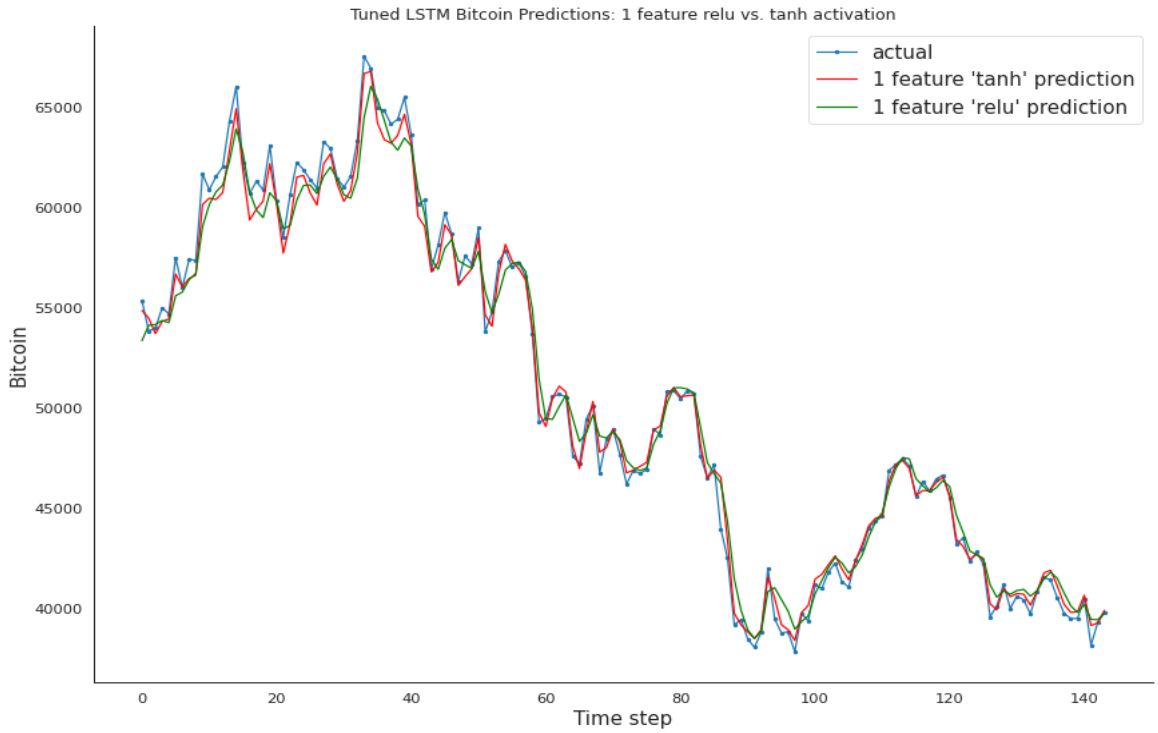


Figure 11. Tuned Model 1 feature TanH vs. ReLU

Note: Graph shows daily price of bitcoin + forecasting values for trained 1 feature hyperparameter tuned models on unseen data and covers a 143-day test period 2021-10-08 / 2022-02-28, Bitcoin values are $ values.

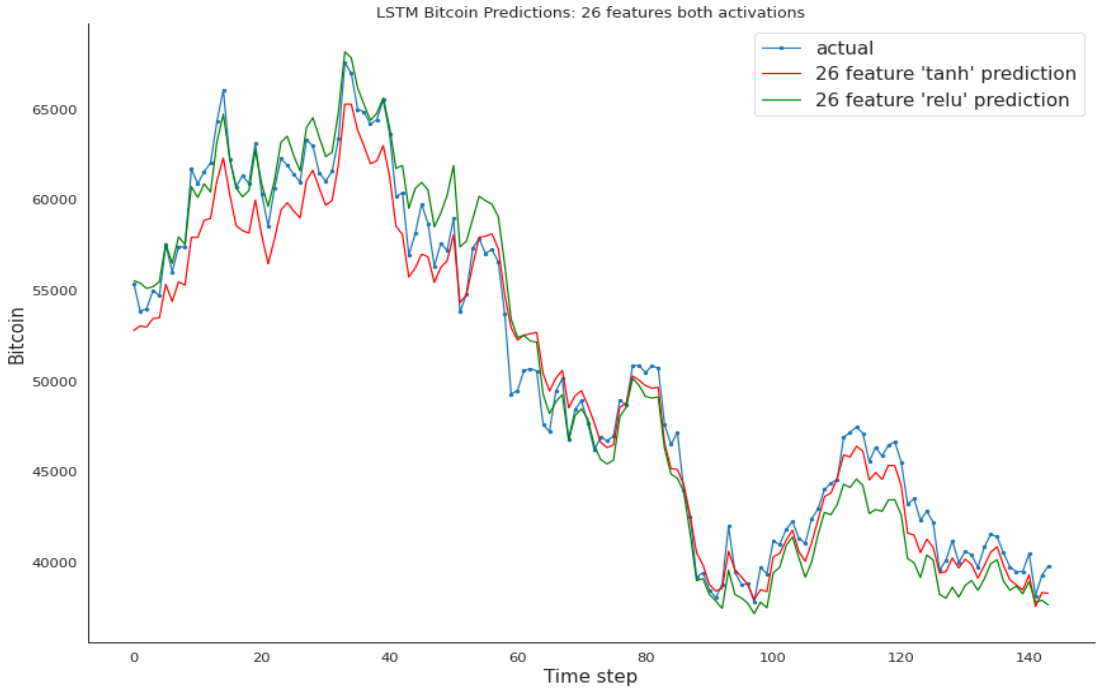


Figure 12. Tuned Model 26 features TanH vs. ReLU

Note: Graph shows daily price of bitcoin + forecasting values for trained 26 features hyperparameter tuned models on unseen test data and covers a 143-day test period 2021-10-08 / 2022-02-28, Bitcoin values are $ values.

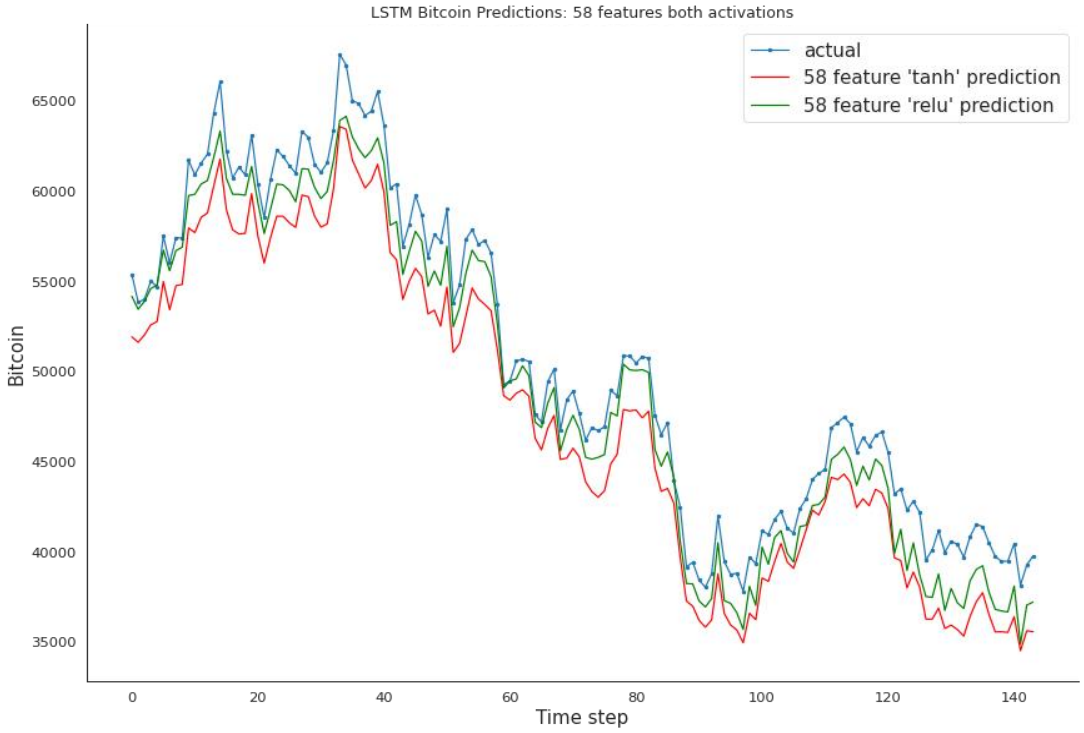


Figure 13. Tuned Model 58 features TanH vs. ReLU

Note: Graph shows daily price of bitcoin + forecasting values for trained 58 features hyperparameter tuned models on unseen test data and covers a 143-day test period 2021-10-08 / 2022-02-28, Bitcoin values are $ values.

The null hypothesis H0 was accepted for both 1 feature TanH vs. ReLU and 26 features TanH vs. ReLU. Except from dropout rate, both 1 feature TanH and 1 feature ReLU had different hyperparameters configurations after gridsearch was conducted. Previously, using uniform base model architectures 1 TanH vs. 1 ReLU, the null was rejected. After these models were allowed to find optimal hyperparameters for their respective activation functions, the autocovariance of their forecasting losses grew more similar. Despite this, their profit/loss was drastically different. For 58 features TanH vs. ReLU the null hypothesis was rejected in favour of our alternative hypothesis H1. For both base models and tuned models TanH vs. ReLU, including all 58 features showed a large difference in their expected loss distributions. The DM test statistics for TanH vs. ReLU forecasting reduced after hyperparameter tuning, suggesting hyperparameter tuning brings their loss distributions closer together. TanH vs. ReLU hypothesis testing for VIF selection set once again accepted the null hypothesis that the models are equally accurate on average.

To further investigate the effects of hyperparameter tuning I conducted hypothesis testing for base model vs. tuned model forecasts using the same feature set with the same activation function. The results for these tests can be found in the following Table 9:

Table 9. Tuned vs. Base Model Results

LSTM: Feature set with activation	RMSE (scale inversed test data)	Diebold Mariano test statistic	Profit / Loss ($)
1. Tuned 1 feature TanH	1733.1608	Target	-493.58
Base 1 feature TanH	1776.6675	-0.72480211	-13,560.51
2. Tuned 1 feature ReLU	1830.0265	Target	-11,915.61
Base 1 feature ReLU	1988.3953	2.44450166	-13,701.31
3. Tuned 26 features TanH	2172.5266	Target	-3,035.18
Base 26 features TanH	2912.5302	-6.29301975	-8,774.97
4. Tuned 26 features ReLU	2310.7409	Target	-11,104.68
Base 26 features ReLU	2705.5946	-2.61108372	-4,387.18
5. Tuned 58 features TanH	3228.2565	Target	4,551.21
Base 58 features TanH	8141.6649	-32.50315354	-8,892.46
6. Tuned 58 features ReLU	1797.8072	Target	-2,455.19
Base 58 features ReLU	3549.8430	-7.6490748	-3,963.89

Note: RMSE scores are scaled inversed forecasting values from trained model on unseen test data. Scores are aggregated over a 143-day test period 2021-10-08 / 2022-02-28, See Appendix (C, D, E, F, G, H)

The null hypothesis of our DM hypothesis test, tuned model vs. base model, was accepted for both 1 feature TanH and 1 feature ReLU. This suggests that the improved RMSE from tuning hyperparameters was insignificant as our models' forecasts did not follow a different average distribution. It is surprising therefore that profit / loss improved so much for our 1 feature TanH base model vs. tuned model. The null hypothesis of our DM hypothesis test, tuned model vs. base model, was rejected for all other tests. This suggests that including additional features and then performing hyperparameter tuning cause the difference between respective models forecasts to follow a differen distribution. RMSE improved for each model following hyperparameter tuning. The only occasion where this improvement in RMSE did not translate to an improvement in profit/loss was 26 features ReLU. For all graphs of tuned model vs. base model predictions see appendix (C, D, E, F, G, H).

5. Conclusion

This research has looked at the effects of feature selection, activation functions and hyperparameter tuning for LSTM bitcoin price forecasting. Including additional features to LSTM models was not shown to produce better RMSE scores on test data, opposing literature. The effectiveness of VIF as a feature selection method is unclear. It produced better RMSE on test data for our base model using both TanH and ReLU activations. However, out of all 26/58 feature models, 58 features hyperparameter tuned ReLU produced the best test RMSE. The best model in terms of profit/loss was 58 features TanH which was also the worst model in terms of RMSE on test data, suggesting that RMSE scores are not an effective way to measure lagged predictions. For both base model and tuned model TanH vs. ReLU, hypothesis testing showed that VIF 26 feature models had no difference in their expected loss distributions. This suggests VIF selection caused the loss distributions of LSTM forecasting to appear similar, irrespective of model configuration. Base model 1 feature TanH vs. ReLU hypothesis testing showed that there was an expected difference in their forecasting distributions. This changed once hyperparameter tuning was applied. Both base model and tuned model TanH vs. ReLU testing showed that 58 feature models had different expected accuracies, suggesting that including additional features that possess multicollinearity causes significant differences in how models using either TanH or ReLU forecast. For TanH vs. ReLU hypothesis testing, the absolute value of the DM statistic reduced for all models after hyperparameter tuning which suggests that allowing models to choose their configuration leads to more similar forecasting accuracies. Hyperparameter tuning caused RMSE to improve for all models. It also the improved profit/loss of all models apart from 26 features ReLU. TanH had better profit/loss for tuned models, but worse RMSE compared to ReLU for larger feature sets. The expected forecasting accuracies of 1 feature models prior to and after hyperparameter tuning was no different. The expected forecasting accuracies of 26 features and 58 features models prior to and after hyperparameter tuning was different, suggesting hyperparameter tuning has significant impact on forecasting when including additional features. As the best model in terms of RMSE did not have the best performance in terms of profit/loss it somewhat suggests this method of evaluating models does not account for changes in price behaviour. As our data is volatile, models must be able to predict volatile changes in price. 1 feature predictions followed the actual price of bitcoin most closely but exhibited lagged behaviour which caused their forecasts to have some of the worst profit/loss.

5.1 Limitations

The buy/sell strategy implemented here was used to illustrate the lagged behaviour of unseen test predictions. The strategy was not implemented with the intention of being the most profit reaping and therefore I admonish its use as a practical investment strategy. Graphs of train RMSE loss could not be obtained due to time constraints. Re-running the code would have meant obtaining novel results which would take considerable time to re-analyse. This is an omission on my part as it meant making analytical assumptions based on aggregated train RMSE scores, which provides limited scope. Due to time constraints a limited number of parameter values were tested during hyperparameter tuning. It is very possible that more values being tested would have led to further improvements in RMSE or profit/loss. The results found were from one run and not aggregated over multiple runs. A better way to get more generalisable results is to take the average result from many runs. VIF was conducted prior to splitting data into train/test which may have introduced bias into the data. The mathematical assumptions of DM hypothesis testing were not scrutinized during this research, and it may be that such testing is ineffective for the number of testing data points used. To ensure the reproducibility of my results I set a random seed. This had the effect of ensuring model results

were the same whilst the kernel was active. As I was conducting my research in Google Colab each time the runtime disconnected model results changed. I was unable to tune hyperparameters during a single runtime session and so it is probable that the values found were not optimal for the final running of the notebook. The experimental results detailed in this paper were run using the same kernel so a comparison between their results is acceptable, although I do not advise generalising these results as they change each time a new kernel is instantiated. Models using larger feature sets did appear to overfit to training data, potentially re-visiting hyperparameter tuning with the intention to reduce this overfitting could provide better results.

5.2 Further Research

This research investigated the effects of including additional features, TanH vs. ReLU activation, and hyperparameter tuning. Each of these areas could warrant further research. Other feature selection methods such as PCA (Abdi & Williams, 2010) or LSTM Autoencoders (Malhotra et al., 2016) could be implemented, and results could be cross-examined to analyse the effects that these feature selection methods have on model forecasts. Employing a methodology more like Rana, Uddin, and Hoque (2019), more activation function / optimizers pairs such as Leaky ReLU or RMSprop (Hinton, Srivastava, & Swersky, 2012) could be tested, and their results cross-examined to analyse the effects that function / optimizer pairs have on bitcoin price compared with stock price. Different hyperparameter tuning methods such as randomised search (Bergstra & Bengio, 2012) or Bayesian optimization (Snoek, Larochelle, & Adams, 2012) could be implemented to cross-examine the effects that different hyperparameter tuning methods have on model forecasts. Further to this, different hypothesis tests such as ANOVA (Scheffe, 1999) could be implemented to cross-examine the results of DM hypothesis testing. Some research (Singh et al., 2021) (Hamayel & Owda, 2021), appears to suggest that alternative RNN architectures like GRU, bi-LSTM, bi-GRU may be superior to LSTM for asset price prediction evaluated using model loss. In this research I applied forecasting using only LSTM as it was suggested as a superior model compared with more traditional forecasting methods. To improve RMSE scores, other RNN architectures could be implemented to test the comparative efficacy of different model architectures; keras makes this very easy. Different trading strategies could be employed. Placing trades based on the forecasted price tomorrow being greater than the actual price today is possibly more intuitive and may perform better particularly for 1 feature forecasting that followed bitcoin price more closely. More sophisticated trading strategies such as long/short which allow for profit during downward price trends could also be investigated to analyse the effect on profit/loss calculations. In this research I only used search engine data as a pseudo feature for social media data. Many papers have investigated the effects of using social media sentiment in western countries as a predictor of bitcoin price (Wolk, 2018) (Karalevicius, Degrande, & Weerdt, 2018), and found it to have a significant positive impact on forecasting. If further improvements in forecasting loss were desired, this is an area I would suggest researching. As Bitcoin is a decentralized system used worldwide, obtaining social media sentiment data from other countries could also provide interesting research (Huang et al., 2021). Search engine data used in this research is not representative of the community driven opinions of social media and news. Therefore, using this data in further research is advised. Social media data is easy to obtain and Natural Language Processing sentiment analysis models such as Vader (Hutto & Gilbert, 2014) or BERT (Devlin, Chang, Lee, Toutanova, 2018) can be used to infer sentiment scores from the social media data obtained.

Profesional Considerations

Public interest

In alignment with the Chartered Institute for IT's code of conduct for BSc members (Bsc.org 2021), consideration of public interest is necessary. To (a) safeguard the wellbeing of readers it is here explicated that neither the application of pattern recognition libraries implemented on financial time series data, or the written fundamental analysis of cryptocurrency's utility are intended as investment advice. Readers who wish to invest in cryptocurrencies should do their own research before taking risks on highly speculative digital currencies. Similarly, the results and use of forecasting models in this research are not intended as investment advice, their architectures are complex and can be ineffective if used by individuals without proper understanding. Cryptocurrency data collected is not indicative of the exact prices that exchanges offer to their users. Individuals wishing to invest in currencies using such means should only ever invest what they can afford to lose as losses are common for retail traders. All data (b) procured for price prediction was anonymous, ensuring that third parties' privacy was protected. To promote equal access to the benefits of this research finalised code has been made available along with the completed write-up and evaluation. Full disclosure of completed research promotes the continued development of academic study (d).

Professional Competence and Integrity

Ensuring the probity of this research is required for the recognition of scientific rigour (Bsc.org 2021). This relates to professional competence and integrity. The research I have conducted was solely my own (a). No external help was sought out other than that of my supervisor and free online resources. Any related work, be it academic papers or software code that is used has been referenced appropriately to ensure I am awarded no recognition indicative of a level of competence I do not possess (b). The cryptocurrency and forecasting literature grow at a fast pace. New studies are published that make older studies redundant. I have (c) continuously tried to check for relevant updates in the field and to grow my skills based on research already conducted. The legislation surrounding cryptocurrencies is complex and differs depending on municipality (d). Cryptocurrency markets are a recent phenomenon and in many countries' laws are non-existent. Readers should conduct themselves in accordance with the laws of their domicile and never attempt to break the law for fear of incurring serious penalty. To uphold the integrity of this research I proclaim that no acceptance or offering of bribery or unethical inducement of any form has taken place (e).

Duty to Relevant Authority

Duty to relevant authority means carrying out my professional responsibilities by conducting research with due diligence and in accordance with the Chartered Institute for IT's code of conduct (a). This meant accepting professional responsibility for my own work and not putting my supervisor in a position where conflicts of interest could arise (b, c). I have at best effort ensured the research conducted here was lawful (d). The findings of my research were in no way misrepresented or doctored (e).

Duty to the Profession

To ensure the duty of both finance and IT professions are not violated I have at best effort upheld the reputation of both by not performing any actions that could have brought either into disrepute (a). This meant disclosing any failures of the research and submitting honest self-written work to uphold the scientific reputation of the Chartered Institute for IT (b, d). It is the hope of myself that any reader of my research should find encouragement through the process I have laid out. Inspiring more people to involve themselves in IT will improve the quality of life for everyone (e).

I would also like to thank both of my supervisors for allowing me to conduct this research:

Prof. Luc Berthouze, Complex Systems
University of Sussex, Falmer, East Sussex, L.Berthouze@sussex.ac.uk

Dr. Malgorzata Sulimierska, Finance and Technology
University of Sussex, Falmer, East Sussex, ms70@sussex.ac.uk

without their help this research would not have been possible.

Reference List:

Abdi, H. and Williams, L.J., 2010. Principal component analysis. *Wiley interdisciplinary reviews: computational statistics*, *2*(4), pp.433-459.

Bcs.org. 2021. *BCS, THE CHARTERED INSTITUTE FOR IT CODE OF CONDUCT FOR BCS MEMBERS*. [online] Available at: <https://www.bcs.org/media/2211/bcs-code-of-conduct.pdf> [Accessed 3 December 2021].

Bergstra, J. and Bengio, Y., 2012. Random search for hyper-parameter optimization. *Journal of machine learning research*, *13*(2).

Bickel, P.J. and Doksum, K.A., 2015. *Mathematical statistics: basic ideas and selected topics, volumes I-II package*. Chapman and Hall/CRC.

Bohra, Y., 2021. *Vanishing and Exploding Gradients in Deep Neural Networks*. [online] Analytics Vidhya. Available at: <https://www.analyticsvidhya.com/blog/2021/06/the-challenge-of-vanishing-exploding-gradients-in-deep-neural-networks/> [Accessed 7 December 2021].

Chen, W., Xu, H., Jia, L. and Gao, Y., 2021. Machine learning model for Bitcoin exchange rate prediction using economic and technology determinants. *International Journal of Forecasting*, *37*(1), pp.28-43.

Chen, Z., Li, C. and Sun, W., 2020. Bitcoin price prediction using machine learning: An approach to sample dimension engineering. *Journal of Computational and Applied Mathematics*, *365*, p.112395.

Chohan, U.W., 2017. Cryptocurrencies: A brief thematic review. *Available at SSRN 3024330*.

Chollet, F., 2022. *Keras documentation: LSTM layer*. [online] Keras.io. Available at: <https://keras.io/api/layers/recurrent_layers/lstm/> [Accessed 8 May 2022].

Chollet, F., 2022. *Keras: the Python deep learning API*. [online] Keras.io. Available at: <https://keras.io/> [Accessed 8 May 2022].

CoinMarketCap. 2021. *Bitcoin Compared To The Largest Fiat Currencies In The World by Market Cap | CoinMarketCap*. [online] Available at: <https://coinmarketcap.com/fiat-currencies/> [Accessed 6 December 2021].

Craney, T.A. and Surles, J.G., 2002. Model-dependent variance inflation factor cutoff values. *Quality engineering*, *14*(3), pp.391-403.

de Sousa, C.A., 2016, July. An overview on weight initialization methods for feedforward neural networks. In *2016 International Joint Conference on Neural Networks (IJCNN)* (pp. 52-59). IEEE.

Devlin, J., Chang, M.W., Lee, K. and Toutanova, K., 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Diebold, F.X. and Mariano, R.S., 2002. Comparing predictive accuracy. *Journal of Business & economic statistics*, *20*(1), pp.134-144.

Dutta, A., Kumar, S. and Basu, M., 2020. A gated recurrent unit approach to bitcoin price prediction. *Journal of Risk and Financial Management*, *13*(2), p.23.

Fama, E., 1970. Efficient Capital Markets: A Review of Theory and Empirical Work. *The Journal of Finance*, 25(2), p.383.

He, K., Zhang, X., Ren, S. and Sun, J., 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision* (pp. 1026-1034).

Hamayel, M.J. and Owda, A.Y., 2021. A Novel Cryptocurrency Price Prediction Model Using GRU, LSTM and bi-LSTM Machine Learning Algorithms. *AI*, *2*(4), pp.477-496.

Hinton, G., Srivastava, N. and Swersky, K., 2012. Neural networks for machine learning lecture 6a overview of mini-batch gradient descent. *Cited on*, *14*(8), p.2.

Hochreiter, S. and Schmidhuber, J., 1997. Long short-term memory. *Neural computation*, *9*(8), pp.1735-1780.

Huang, X., Zhang, W., Tang, X., Zhang, M., Surbiryala, J., Iosifidis, V., Liu, Z. and Zhang, J., 2021, April. Lstm based sentiment analysis for cryptocurrency prediction. In *International Conference on Database Systems for Advanced Applications* (pp. 617-621). Springer, Cham.

Hutto, C. and Gilbert, E., 2014, May. Vader: A parsimonious rule-based model for sentiment analysis of social media text. In *Proceedings of the international AAAI conference on web and social media* (Vol. 8, No. 1, pp. 216-225).

James, G., Witten, D., Hastie, T. and Tibshirani, R., 2013. *An introduction to statistical learning* (Vol. 112, p. 18). New York: springer.

Jay, P., Kalariya, V., Parmar, P., Tanwar, S., Kumar, N. and Alazab, M., 2020. Stochastic Neural Networks for Cryptocurrency Price Prediction. *IEEE Access*, 8, pp.82804-82818.

Karalevicius, V., Degrande, N. and De Weerdt, J., 2018. Using sentiment analysis to predict interday Bitcoin price movements. *The Journal of Risk Finance*.

Kingma, D.P. and Ba, J., 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Koh, P.W. and Liang, P., 2017, July. Understanding black-box predictions via influence functions. In *International conference on machine learning* (pp. 1885-1894). PMLR.

Lafourcade, P. and Lombard-Platet, M., 2020. About blockchain interoperability. *Information Processing Letters*, *161*, p.105976.

Liashchynskyi, P. and Liashchynskyi, P., 2019. Grid search, random search, genetic algorithm: A big comparison for NAS. *arXiv preprint arXiv:1912.06059*.

Li, W. and Liao, J., 2017, October. A comparative study on trend forecasting approach for stock price time series. In *2017 11th IEEE International Conference on Anti-counterfeiting, Security, and Identification (ASID)* (pp. 74-78). IEEE.

Maese, V., Avery, A., Naftalis, B., Wink, S. and Valdez, Y., 2016. Cryptocurrency: A Primer. *Banking Law journal*, [online] 133(1), pp.468-471

Malhotra, P., Ramakrishnan, A., Anand, G., Vig, L., Agarwal, P. and Shroff, G., 2016. LSTM-based encoder-decoder for multi-sensor anomaly detection. *arXiv preprint arXiv:1607.00148*.

Mallqui, D.C. and Fernandes, R.A., 2019. Predicting the direction, maximum, minimum, and closing prices of daily Bitcoin exchange rate using machine learning techniques. *Applied Soft Computing*, *75*, pp.596-606.

McCulloch, W.S. and Pitts, W., 1943. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, *5*(4), pp.115-133.

McNally, S., Roche, J. and Caton, S., 2018, March. Predicting the price of bitcoin using machine learning. In *2018 26th euromicro international conference on parallel, distributed and network-based processing (PDP)* (pp. 339-343). IEEE.

Medsker, L. and Jain, L.C. eds., 1999. *Recurrent neural networks: design and applications*. CRC press.

Mehtab, S. and Sen, J., 2020. A time series analysis-based stock price prediction using machine learning and deep learning models. *International Journal of Business Forecasting and Marketing Intelligence*, *6*(4), pp.272-335.

Mokhtarian, E. and Lindgren, A., 2017. Rise of the Crypto Hedge Fund: Operational Issues and Best Practices for Emergent Investment Industry. *Stanford Journal of Law, Business, and Finance, Forthcoming*.

Nair, V. and Hinton, G.E., 2010, January. Rectified linear units improve restricted boltzmann machines. In *Icml*.

Nakamoto, S., 2008. *Bitcoin: A Peer-to-Peer Electronic Cash System*. [ebook] [Accessed 11 November 2021].

NVIDIA Developer. 2022. *NVIDIA cuDNN*. [online] Available at: <https://developer.nvidia.com/cudnn> [Accessed 8 May 2022].

Palamalai, S., Kumar, K. and Maity, B., 2021. Testing the random walk hypothesis for leading cryptocurrencies. *Borsa Istanbul Review*, 21(3), pp.256-268.

Pichl, L. and Kaizoji, T., 2017. Volatility analysis of bitcoin. *Quantitative Finance and Economics*, *1*(4), pp.474-485.

Rana, M., Uddin, M.M. and Hoque, M.M., 2019, December. Effects of activation functions and optimizers on stock price prediction using LSTM recurrent networks. In *Proceedings of the 2019 3rd International Conference on Computer Science and Artificial Intelligence* (pp. 354-358).

Refaeilzadeh, P., Tang, L. and Liu, H., 2009. Cross-validation. *Encyclopedia of database systems*, *5*, pp.532-538.

Rosenblatt, F., 1958. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, *65*(6), p.386.

Ruder, S., 2016. *An overview of gradient descent optimization algorithms*. [ebook]

Saad, M., Choi, J., Nyang, D., Kim, J. and Mohaisen, A., 2019. Toward characterizing blockchain-based cryptocurrencies for highly accurate predictions. *IEEE Systems Journal*, *14*(1), pp.321-332.

Saury, J., 1774. *Cours complet de mathématiques*. aux dépens de Ruault.

Schäfer, A.M. and Zimmermann, H.G., 2006, September. Recurrent neural networks are universal approximators. In *International Conference on Artificial Neural Networks* (pp. 632-640). Springer, Berlin, Heidelberg.

Scheffe, H., 1999. *The analysis of variance* (Vol. 72). John Wiley & Sons.

Sen, J. and Chaudhuri, T.D., 2018, December. Stock price prediction using machine learning and deep learning frameworks. In *Proceedings of the 6th International Conference on Business Analytics and Intelligence, Bangalore, India* (pp. 20-22).

Singh, A., Kumar, A. and Akhtar, Z., 2021. *Bitcoin Price Prediction: A Deep Learning Approach*. [online] Ieeexplore.ieee.org.

Snoek, J., Larochelle, H. and Adams, R.P., 2012. Practical bayesian optimization of machine learning algorithms. *Advances in neural information processing systems*, *25*.

Sovbetov, Y., 2018. Factors influencing cryptocurrency prices: Evidence from bitcoin, ethereum, dash, litcoin, and monero. *Journal of Economics and Financial Analysis*, *2*(2), pp.1-27.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. and Salakhutdinov, R., 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, *15*(1), pp.1929-1958.

Sjsu.edu. 2007. [online] Available at: <https://www.sjsu.edu/faculty/gerstman/StatPrimer/z-two-tails.pdf> [Accessed 13 April 2022].

TensorFlow. 2022. *TensorFlow*. [online] Available at: <https://www.tensorflow.org/> [Accessed 8 May 2022].

van Rossum, G., 2022. *Welcome to Python.org*. [online] Python.org. Available at: <https://www.python.org/> [Accessed 8 May 2022].

Wood, T., 2021. *Activation Function*. [online] DeepAI. Available at: <https://deepai.org/machine-learning-glossary-and-terms/activation-function> [Accessed 7 December 2021].

Wołk, K., 2020. Advanced social media sentiment analysis for short-term cryptocurrency price prediction. *Expert Systems*, *37*(2), p.e12493.

Yelowitz, A. and Wilson, M., 2015. Characteristics of Bitcoin users: an analysis of Google search data. *Applied Economics Letters*, *22*(13), pp.1030-1036.

Yıldırım, D.C., Toroslu, I.H. and Fiore, U., 2021. Forecasting directional movement of Forex data using LSTM with technical and macroeconomic indicators. *Financial Innovation*, *7*(1), pp.1-36.

Yong, Y.L., Ngo, D.C. and Lee, Y., 2015, June. Technical indicators for forex forecasting: a preliminary study. In *International Conference in Swarm Intelligence* (pp. 87-97). Springer, Cham.

Appendix

Appendix A

Graph of activation functions tanh, sigmoid, relu

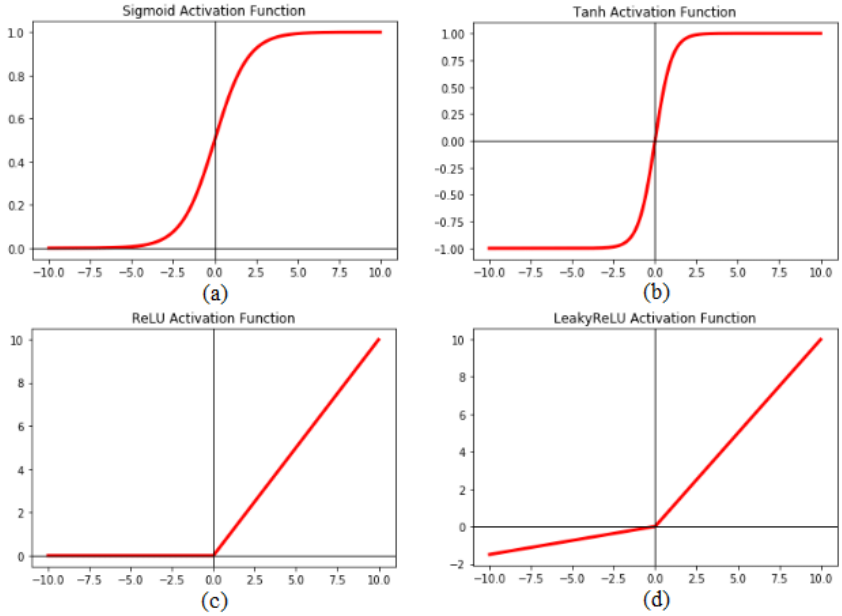


Image taken from - https://www.researchgate.net/figure/Plot-of-different-activation-functions-a-Sigmoid-activation-function-b-Tanh_fig4_339991922. [Accessed 2022-05-02].

Appendix B

Graph of derivatives for activation functions

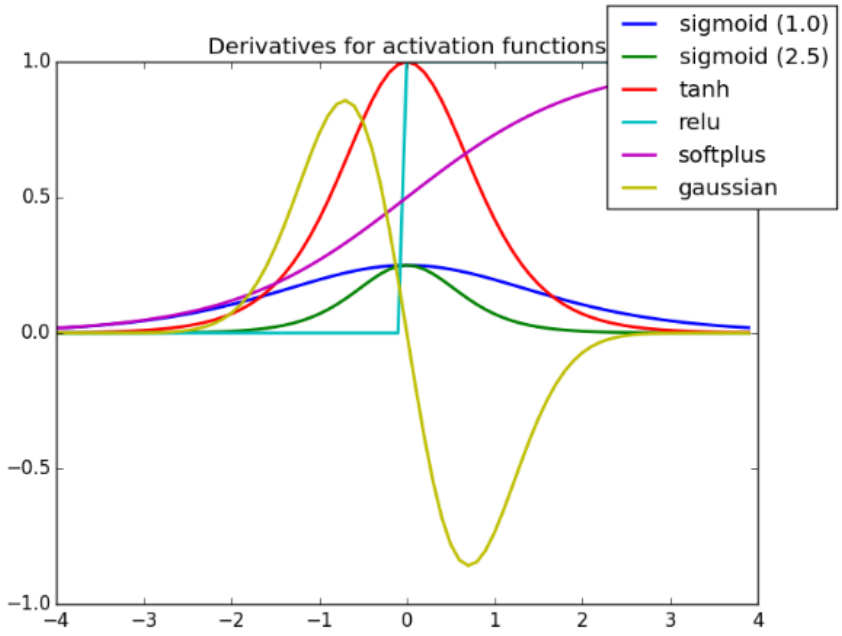
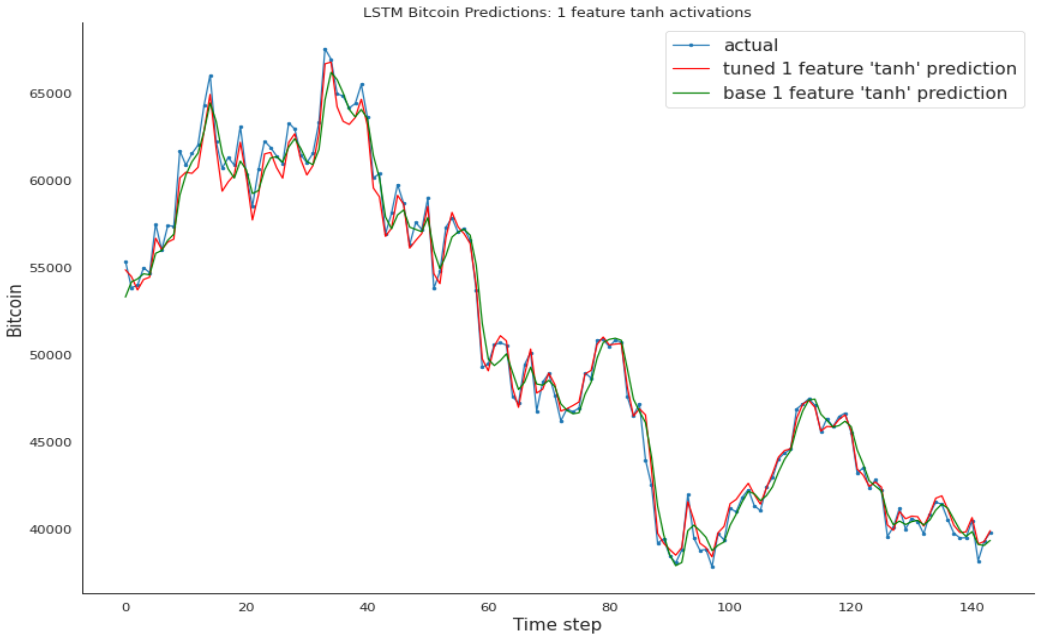


Image taken from - https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6. [Accessed 2022-05-02].

Appendix C

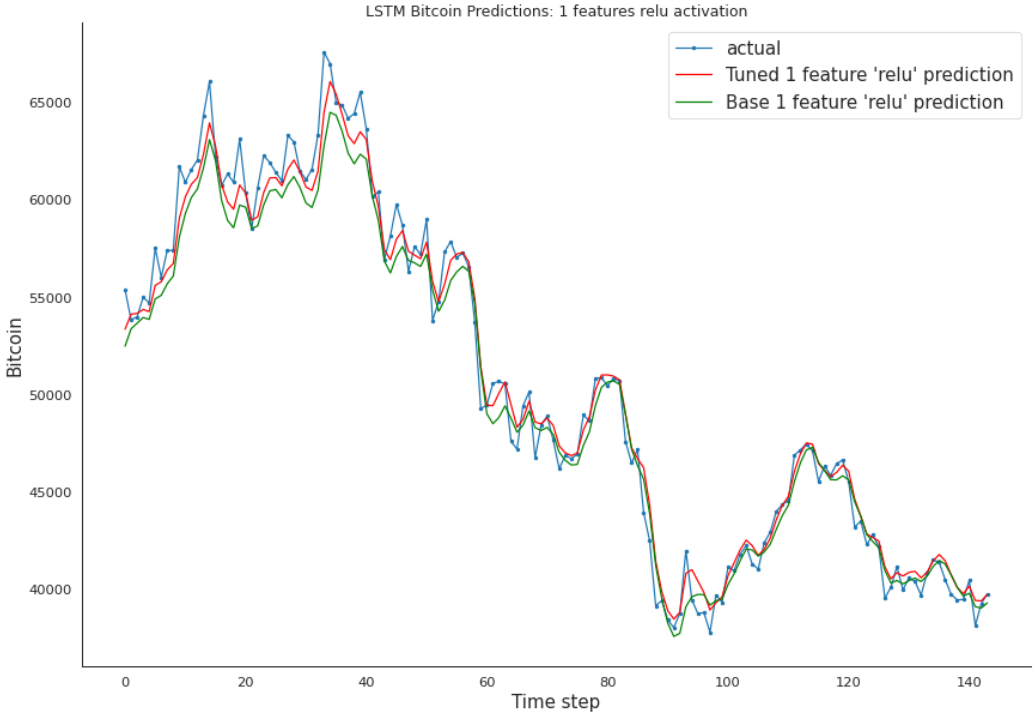
Graph of Tuned Model vs. Base Model 1 feature TanH



Note: Graph shows daily price of bitcoin + forecasting values for pre-trained tuned vs base 1 feature TanH models on unseen test data and covers a 143-day test period 2021-10-08 / 2022-02-28, Bitcoin values are $ values.

Appendix D

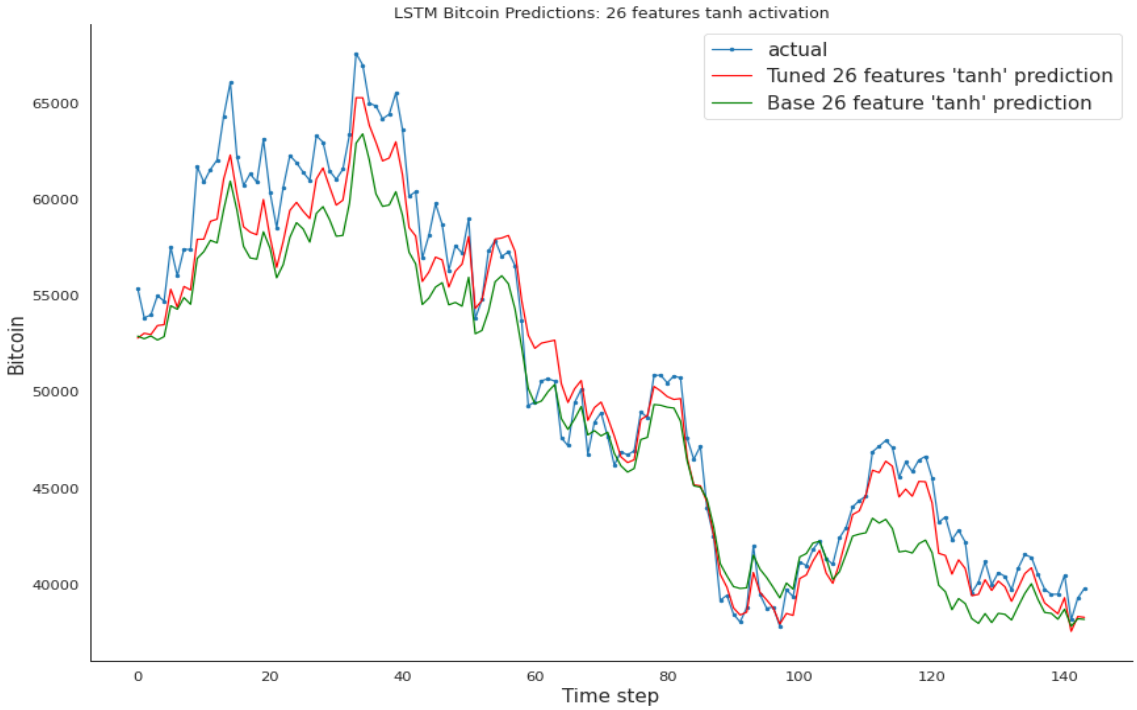
Graph of Tuned Model vs. Base Model 1 feature ReLU



Note: Graph shows daily price of bitcoin + forecasting values for pre-trained tuned vs base 1 feature ReLU models on unseen test data and covers a 143-day test period 2021-10-08 / 2022-02-28, Bitcoin values are $ values.

Appendix E

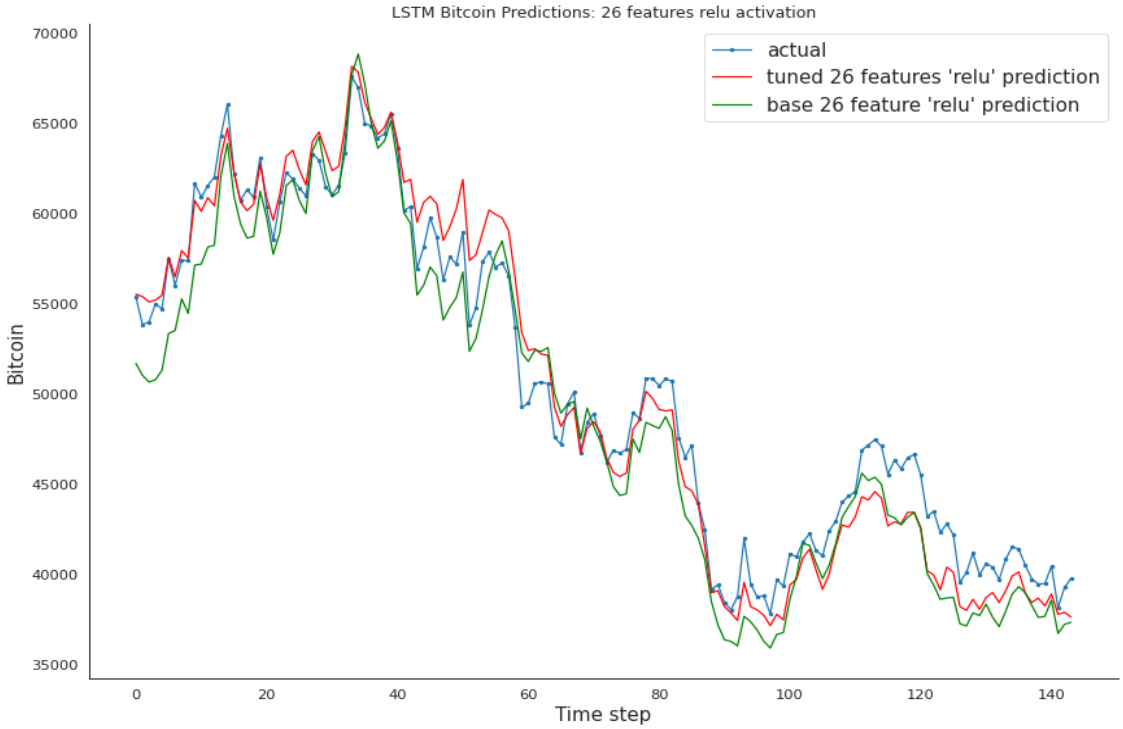
Graph of Tuned Model vs. Base Model 26 features TanH



Note: Graph shows daily price of bitcoin + forecasting values for pre-trained tuned vs base 26 features TanH models on unseen test data and covers a 143-day test period 2021-10-08 / 2022-02-28, Bitcoin values are $ values.

Appendix F

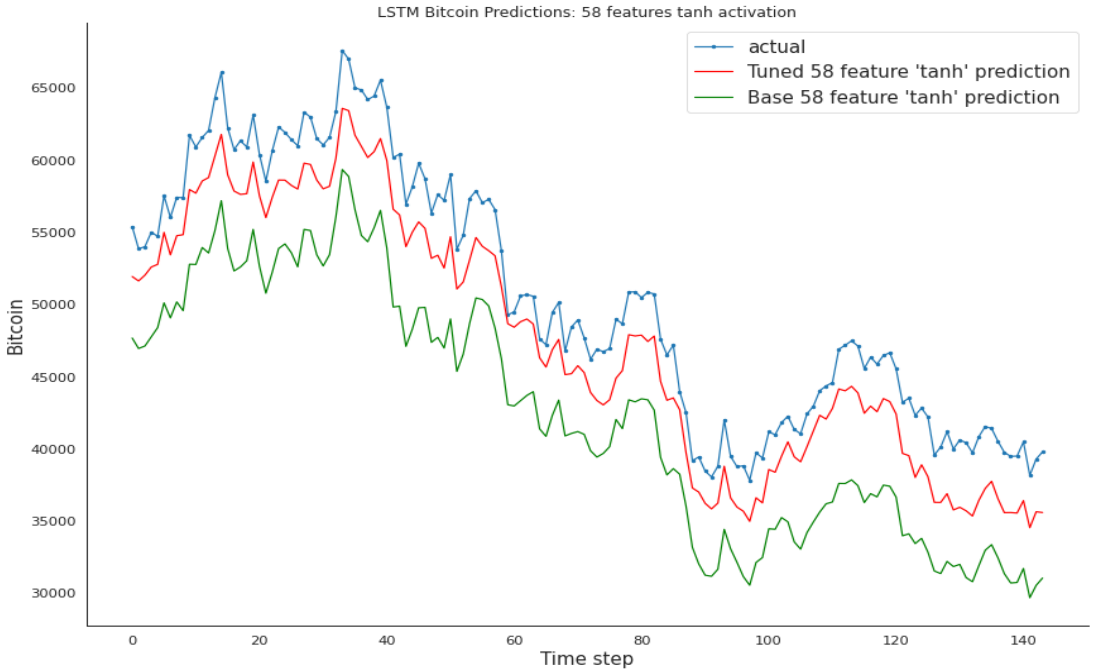
Graph of Tuned Model vs. Base Model 26 features ReLU



Note: Graph shows daily price of bitcoin + forecasting values for pre-trained tuned vs base 26 features ReLU models on unseen test data and covers a 143-day test period 2021-10-08 / 2022-02-28, Bitcoin values are $ values.

Appendix G

Graph of Tuned Model vs. Base Model 58 features TanH



Note: Graph shows daily price of bitcoin + forecasting values for pre-trained tuned vs base 58 features TanH models on unseen test data and covers a 143-day test period 2021-10-08 / 2022-02-28, Bitcoin values are $ values.

Appendix H

Graph of Tuned Model vs. Base Model 58 features ReLU

Note: Graph shows daily price of bitcoin + forecasting values for pre-trained tuned vs base 58 features ReLU models on unseen test data and covers a 143-day test period 2021-10-08 / 2022-02-28, Bitcoin values are $ values.