Comparing the viability of LSTMs & Attention Mechanisms for forecasting financial time series

Baylee Devers
pl18146@bristol.ac.uk

Nims Hills
il18809@bristol.ac.uk

Thomas Howes
ji18486@bristol.ac.uk

April 2021
https://github.com/EMAT31530/ai-group-project-ml-for-finance

Abstract

Time series forecasting is a common task in machine learning and has particular importance in the financial world, since data for asset prices is collected as a financial time series. In this paper we restrict our focus to the time series corresponding to the price of Bitcoin, in the hope that due to the large influence of amateur investors, the sentiment of Bitcoin on social media will be directly relevant to its price movements. We hypothesize that an LSTM used together with an attention mechanism will outperform LSTMs alone, and lead to to better predictions due to their ability to attend to long-term dependencies in the time series data. Further, we analyse each model in comparison to an uninformed baseline that acts without knowledge of any data features and detail its performance in comparison to our models.

Contents

1 Introduction **2**
2 Literature review **2**
3 Experimentation & Analysis **4**
4 Method & Implementations **5**
4.1 Data Collection 5
4.2 Preprocessing 5
4.3 Machine learning algorithms 6
4.3.1 Dense (classification) 6
4.3.2 LSTM (classification) 6
4.3.3 LSTM (regression) 7
4.3.4 Attention mechanisms 7
5 Discussion **8**
6 Conclusion **9**
7 Future work **10**

1 Introduction

Algorithms such as Long short-term memory [1] (LSTM) and Attention Mechanisms have gained widespread interest in recent years for the task of language translation. This is because both methods can deal with sequence prediction problems where the input and output lengths vary.

Our focus however will be to see how these methods can be applied to forecast future values of multivariate time series data, such as financial time series, using information about values from a previous time. These methods have not only been used to forecast the price of traditional financial products like the S&P 500 or FTSE 100 stock market indices [2, 3], but also alternative investments and commodities, such as Crude oil [4], Limestone [5], and cryptocurrencies.

Cryptocurrencies are a set of decentralised digital currencies used in peer-to-peer transactions that have mostly seen a surge in value in 2020 and are chosen for proposed benefits such as privacy, transaction speed, and trust-less nature such as their independence of the monetary policy of a given central bank. The COVID-19 pandemic is thought to have driven a lot of recent market activity as governments brace for the economic implications of their non-pharmaceutical interventions. The cryptocurrency market is categorised by high volatility; no closing trading periods, as would be seen on a stock exchange, and a high level of data availability [6] which make them a particularly favourable case study. Factors motivating the market activity have ranged from the difficulty of mining each currency, to the spot price of assets such as gold [7], considered to be a substitute for cryptocurrencies for their shared characteristics.

In this paper, we will use some of the methods at the forefront of Artificial Intelligence research, to forecast the price of the largest cryptocurrency by market cap, Bitcoin [8], using its historic time series data. These will include machine learning models for regression and classification, as well as, Natural Language Processing of social media data.

We believe that the cryptocurrency market could be a good example of a market heavily driven by public textual data since it attracts many more amateur active investors than traditional financial instruments due to its accessibility.

2 Literature review

Our work was originally motivated by the paper [9] which used a deep learning framework, and Natural Language Processing, to predict movements in the US stock market from data embedded within the question and answer component of a company's earnings call transcript, which is a public report of the recent financial performance by senior company executives that involves spontaneous interactions between the executives, investors, and equity analysts.

As well as using textual features from the transcript, there is a industry categorization feature, to distinguish the unique market dynamics of each sector. The objective is to learn a predictive function that can classify binary movement of a stock at closing price, on the day immediately after the earnings call. The model is trained on the constituent companies of the S&P 500 index, and the industry categorisation is shown to be of particular significance, since the model performs the best in the information technology sector, and performs most poorly in the energy sector.

This corresponds to traditional interpretations of the markets driving factors, the technology sector is mostly thought to be driven by high frequency information, whilst the value of companies in the energy sector is strongly correlated to the price of particular commodities and macroeconomic trends. Thus, it was important for us to appropriately reflect some of the key features of the cryptocurrency market in our work to ensure that model will yield relevant results. Each section of the earnings call is represented as a vector by applying a bidirectional LSTM with an attention mechanism on the tokens,[10] however the word embedding layer of the model was not trainable, to reduce computational complexity.

A particular shortcoming of this paper is that the feature vectors do not account for any technical analysis of the historic time series data which is a significant element of many investment decisions, we extend our model to include some features representing the technical analysis.

Extensive work has been done to also learn the volatility of a market based on the textual data from earnings call transcripts [11]. The paper also incorporates a set of common financial features observed to drive volatility, such as past volatility, overall market volatility, and the book-to-market ratio, which is the ratio between the value of a companies assets and its current market trading price.

It aims to predict a continuous feature (regression) as prediction share price volatility in the context of a classification problem would lack interpretability.

The model is a Feedforward Neural Network trained on a very large database of 4.3k UK companies, and is tested against sophisticated benchmarks, such as Ridge Regression [12], and Huber Regression [13], with multiple evaluation metrics, so it provides more rigorous results than paper [9].

Performance is influenced by a range of hyperparameters, which are tuned with a Bayesian optimisation algorithm [14]. When we visualise the Attention, we see that high-attention is allocated towards tokens associated with the Brexit vote, this result is quite intuitive, given that the uncertainties of Brexit are well documented [15], so we can see that this mechanism is effective.

The results of this paper are then directly compared to the current state of the art model [16] which uses a semiparametric Gaussian Copula Regression model, originating from Copula theory. This model seeks to separate the uniform martingales and their complex multivariate stochastic dependencies, to find the dependencies of random variables without prior assumptions for the covariate or independent variable. By performing a probability integral transform, they move beyond the count-based bag-of-words feature space to marginal cumulative density functions space. By using a parametric copula, in this case, the Gaussian copula, the computational cost from fully nonparametric methods is significantly reduced.

The results of Profet are competitive with the Gaussian Copula which motivated further research into Attention mechanisms. This method was initially developed for the task of language translation [17], which requires a sequence to sequence translation because words in a phrase of a language, are not translated to the target language one-to-one. Traditionally, this task is performed with a recurrent neural network, that factor along the symbol positions of the input and output sequences, and align the positions to steps in communication time, they then generate a sequence of hidden states h_t, as a function of the previous hidden state and the input for position t, this approach prevents parallelisation within the training examples which makes computation extremely intensive at long sequence lengths.

Attention methods are offered as a solution to this problem, by allowing dependencies to be modelled without regard for their distance in the input or output sequences. Here, they propose a model architecture that relies entirely on an attention mechanism to draw global dependencies. Attention functions map a query, and a set of key-value pairs to an output, which is computed as a weighted sum of the values, where the weights are computed with a compatibility function of the query and the corresponding key. Here they use $Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V$, attention is computed simultaneously with the sets of queries, keys, and values, packed into matrices Q, K, and V respectively. Sinusoidal positional encodings are used to inject information about the position of a token, since the model has no recurrence or convolution, maximum path length between long range dependencies is reduced significantly with respect to a standard RNN using this architecture. Attention mechanisms have been implemented in multivariate time series problems [18] to forecast traffic congestion, solar power and electricity consumption, and the foreign exchange rates of various currencies.

The paper [19] addresses a very similar task to ours, textual data from twitter posts is used to determine the directional movement of four of the largest cryptocurrencies by market cap. The sentiment of each tweet is analyzed as positive, negative, or neutral, with a package specifically attuned to twitter data, and information from the tweets is aggregated into a vector and the model is trained using Multi-Layer perceptron neural network, Support Vector Machines, and Random Forests. Multilayer perceptrons (MLPs) are a class of feedforward artificial neural networks that can use back propagation and supervised learning for training. Neurons from the hidden layers apply a linear summation function

$$w_1x_1 + w_2x_2 + ... + w_mx_m,$$

followed by a non-linear activation function, in this case a hyperbolic tangent activation function, to the values of the previous layers. The output layer transforms the values received from the last hidden layer into outputs. Support Vector Machines is a supervised learning algorithm that uses a kernel function to construct a set of hyperplanes in high dimensional space and seeks to maximise the distance between the hyperplane and the nearest training examples. This is done by obtaining the training examples that are the closest to the maximum margin hyper-plane which are denominated support vectors. SVMs are used for regression and classification problems, and in [20] it is shown that SVMs are resistant to overfitting. This paper uses a Gaussian radial basis kernel function:

$$K(x; y) = exp(-1/\sigma^2(x-y)^2)$$

in the algorithm. Random Forests are an ensemble learning method that operate by constructing a multitude of decision trees and fitting them to sub-samples of the data. They control predictive accuracy and over-fitting by averaging the predictions of each decision tree. Confidence intervals for this analysis are obtained with K-fold cross-validation and the 95% confidence interval for accuracy reveals that MLPs are the best performing model for 3/4 currencies. This paper, along with [21] show that learned textual features in a classification model may be enough to infer meaningful market outcomes. Further research has included the use of sentiment analysis to predict cryptocurrency market bubbles [22], and textual data used to infer financial time series has included news headlines [23], and 10-K fillings [24], it is likely that with a different choice of textual data, that reflects the economic consensus surrounding the markets studied, that we could infer financial time series more accurately.

3 Experimentation & Analysis

Initially, we explored scraping data from the YouTube API [25] with the intention of applying NLP techniques such as word embeddings [26] and attention mechanisms to the titles and comments on videos. However, due to a low request quota imposed by Google, we decided against using YouTube as a source of textual data. The strict quota would have been particularly limiting since working with textual data necessarily requires a large quantity due to its sparse nature.

One of the first problems we encountered was exploding loss values, which was resolved by scaling the data and ensuring no NaN values were still present in the time-series.

As our baseline, we took the prediction of price for the next time step to be identical to price for the latest time-step in the window. This is justified as asset prices are often modelled as martingales [27]. i.e. for some random asset price P_t, it is often assumed to be the case that:

$$\mathbb{E}(P_{t+1}) = P_t$$

We sometimes refer to this baseline as 'the naive predictor' and use it as a comparison and also a sanity check to verify that our models perform suitably.

We found our initial LSTM model, trained only on the opening price feature, performed poorly with validation loss greater than the baseline. In order to add more features, we first generalised the preprocessing function `stack_windows`. Once this was done, we included the volume traded and also features from the public ledger such as block size and transaction fees.

With the baseline in mind, it became apparent that a more fitting approach would be to make inferences about the percentage change in prices rather than the raw price; the idea being that the function the model would have to learn would have lower degree and thus, be learned faster. The reasoning is that instead of learning that the price, P_{t+1}, at the next time-step, is a Gaussian with $\mu = P_t$ the model would instead only need to learn that the next price *change* is a Gaussian with $\mu = 0$.

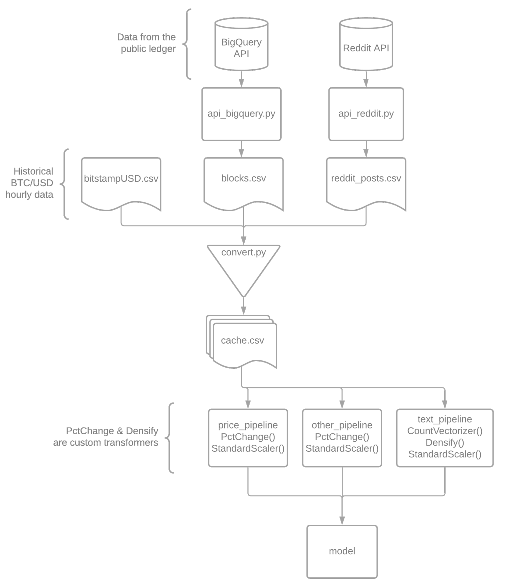
After framing the problem as that of trying to predict the percentage change in price the trained LSTM model managed to outperform the baseline. Moreover, the model converged much faster and passed the baseline on the first epoch. We implemented an early stopping callback so as to achieve the model with the minimum validation loss.

Until this point, we had only a few features. Although the YouTube API had not proven to be useful for our purposes we still wanted to include textual data and so wrote a scraper to collect data from the Bitcoin Reddit community. We used the pushshift.io API [28] to gather data about posts including each post's title, author and number of comments.

Finally, with the feature preprocessing and working model in place we experimented with attention, most notable by using a regulariser given by $\left\|(AA^T - I)^2\right\|_F$ as seen in [29], but since many of these are mechanisms are built for different tasks, we struggled to make many of the initial implementations fit.

4 Method & Implementations

In the following subsections, we detail the data flow through our final model and the different model architectures that we compared.



Data from the public ledger
BigQuery API
Reddit API
api_bigquery.py
api_reddit.py
Historical BTC/USD hourly data
bitstampUSD.csv
blocks.csv
reddit_posts.csv
convert.py
cache.csv
PctChange & Densify are custom transformers
price_pipeline PctChange() StandardScaler()
other_pipeline PctChange() StandardScaler()
text_pipeline CountVectorizer() Densify() StandardScaler()
model

Figure 1: Scraping & preprocessing steps

4.1 Data Collection

Three sources of data were used as inputs to our model:

- Price & Market Data
  For the target data and main features, the clear choice was a large historical dataset on Kaggle containing data at a 1-minute resolution for the BTC/USD pair. This includes features such as the opening and closing price, the highest and lowest price in the interval, volume traded, and the volume weighted average price. Figure 2 shows the BTC/USD price distribution in 1-minute intervals.

- Blockchain Data
  One reason why we decided to use time series data for Bitcoin's price was the availability of transaction data recorded on a public ledger. Due to this ledger's large size, we used the database provided by Google's BigQuery API and have written SQL queries to obtain only the data necessary to build our features.

- Social Media Data
  Among the oldest and most active forums for discussing Bitcoin is the r/Bitcoin subreddit on reddit.com. As such, it was a natural choice as a source of textual data that would likely be related to price movements. In particular, we used a third party API, the pushshift.io API to gather title data along with additional metadata from close to 1 million posts.

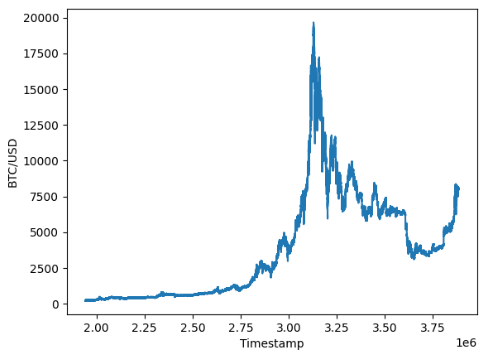


Figure 2: BTC/USD 1-minute interval opening prices

4.2 Preprocessing

All of the aforementioned sources of data needed to be prepared and have meaningful features extracted. Preparing temporal data presents more challenges than a typical data set and careful consideration must be taken to ensure that testing is not done on seen data and that the features do not contain the target data.

The first step was to clean the data and merge the different csvs gathered from the above sources. This is done in convert.py where time-steps containing null data are filled with the data of the preceding time-step and data is aggregated into hourly intervals.

In order to attain the hour on hour change in opening price we made use of pandas' DataFrame method `.pct_change` and wrapped this within an sklearn custom transformer so it could be used as part of a pipeline. This transformer was then applied to all of the numerical feature columns. For the textual data, the post titles were concatenated into one string. A bag of words model `CountVectorizer` that extracted only unigrams and has a hard-coded vocabulary, `sentiment_words`, was applied to the string. The vocabulary chosen consists of words whose frequency we hope would reflect the community sentiment and thus price movement.

As outlined in section 3, we used the `MinMax` and later `StandardScaler` sklearn transformer to scale all features of the data to have 0 mean and unit variance. When working with RNNs scaling data is absolutely necessary due to the issue of exploding gradients.

In contrast to data made up of a set of class instances, which should usually be split randomly into training and testing sets, temporal data should be partitioned into training and testing sets systematically. The split should be such that all training instances come before the testing instances, this ensures that no future information is 'seen' by the model which may allow it to overfit.

The data passed into the model is also not as simple as n hours by m features. The third dimension of the hours in the past must be considered. In the code it is referred to as the time 'window' and has size `window_len`. The 3 dimensional tensor is constructed from the feature matrix by the `stack_windows` function.

4.3 Machine learning algorithms

In this section, we will explore our use of different machine learning techniques to approach the problem. We started by applying a long short-term memory (LSTM) network to our problem in the context of both linear regression and classification. For comparison, we also used a very simple dense neural network. Finally, we tried using an attention mechanism to reduce computational complexity for global dependencies.

4.3.1 Dense (classification)

When considering the problem as a classification task, we decided to implement a fully connected neural network. We did not expect this to perform well, but thought it could perhaps serve as a baseline to compare the other models to. Due to the target data (opening price) taking continuous values, we needed to form discrete classes in order to approach this as a classification problem. We proceeded by partitioning the data into up and downwards price movements by comparing the opening price at each hourly interval. Class 0 represented a decrease in price and class 1 represented an increase. Originally, the data was split into three classes; increase, decrease and constant. However, the last class was promptly removed since the change in price did not take discrete values, there would not be instances where the price would remain exactly constant.

In the preprocessing of the data, we applied one-hot encoding to our class labels. We do this because we are using a loss function, `binary_crossentropy` to measure the performance of our predictions. Comparing the output of our model to the true values gives us a loss value, and using backpropagation, we can update the weights of the neural network, over a certain number of iterations, to minimise the loss. Finally, it is important to plot the loss, particularly the validation loss, and accuracy in order to see the performance of our model. Visual representations comparing the model to a baseline can then be interpreted and improvements can be made.

4.3.2 LSTM (classification)

After reviewing a variety of papers in section 2, it became apparent that machine learning methods that take into account data from previous iterations were more effective when carrying out price predictions. This led to the decision to apply an LSTM, which uses data from previous iterations to update weights in the current iteration. In Figure 3, it is shown that an LSTM cell involves the use of a memory cell to store information and this information is then manipulate by being passed through three gates. These gates are called the forget gate, input gate and output gate. The forget gate decides which information from previous steps is relevant and which to discard, the input gate determines which information from the current step to add, and output gate controls what the next hidden state should be; what is passed through the next step.

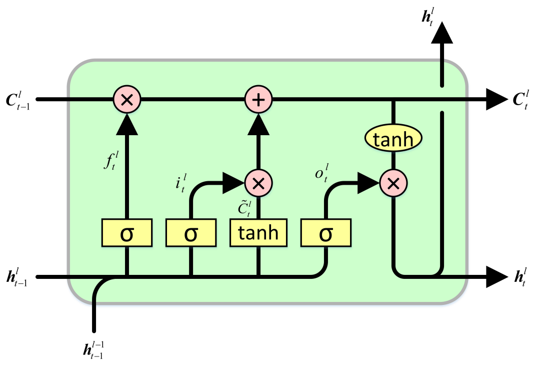


Figure 3: LSTM cell [30]

Our hypothesis was that using a model with an LSTM architecture would perform better than a simple dense neural network as it would have knowledge of the order from previous hours, which is crucial due to the decay in importance of time-steps the further in the past they are. Creating this model did not require much deviation from our initial Dense model; We were able to re-use the functionality that handles the classes. The main difference was the use of the function `stack_windows` which collected the data into an array where each row of the ndarray corresponds to `window_len` consecutive hours immediately before the hour used as a label.

4.3.3 LSTM (regression)

It seems appropriate that this problem should be framed as a regression task, given that we are interested not only in the direction of price movement, but also the magnitude of such a movement. In our final regression model, we decided on an LSTM layer with 128 units followed by a Dropout layer to tackle gradient explosion and act as a regularization method. We used a mean squared error loss function and optimized this by stochastic gradient

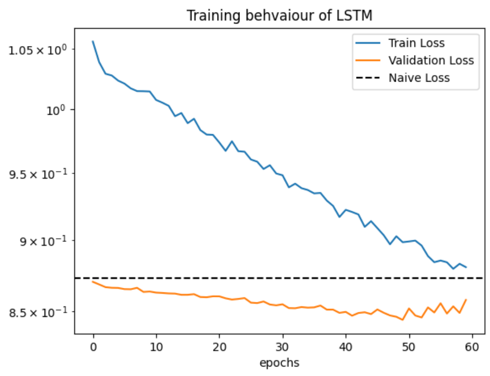


Figure 4: Loss convergence for LSTM model

descent with a learning rate of 0.01. For reasons unknown to us, the Adam optimizer seemed to perform worse. One surprisingly important parameter we changed was the `batch_size`. Although they decreased training time significantly, batch sizes of larger than 500 resulted in poor convergence behaviour and worse overall validation loss, presumably due to the non-stochastic descent becoming stuck in a local minima and failing to reach the global minimum. For this reason we decided on a batch size of 100 Figure 3 shows the predictions made on the test set by the trained LSTM. *Note, the horizontal axis has been cropped*

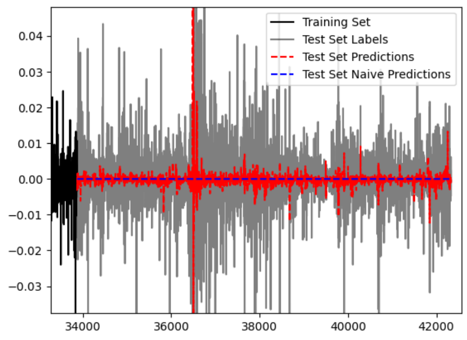


Figure 5: Price predictions made by the LSTM

4.3.4 Attention mechanisms

To implement our Attention we used a custom keras layer `Attention` from keras-attention-mechanism

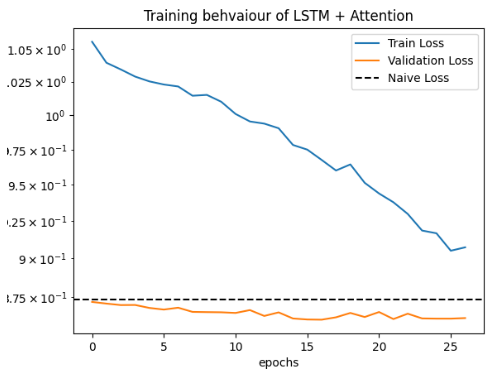


Figure 6: Loss convergence for Attention model

[31] with 64 units, together with the `LSTM` layer. This uses a scoring function given by Luong's multiplicative [32] and Bahdanau's additive, is used to evaluate attention weights and the attention vector is given by $\alpha_t = f(c_t, h_t) = tanh(W_c[c_t : h_t])$ where c_t are elements of the context vector. In 10 we see that while both models outperform the naive predictor, but surprisingly, the LSTM without an Attention layer has the lower validation loss. One reason for this may be that long-range dependencies are not significant, and given that our application is novel, the literature on this was insufficient.

5 Discussion

The primary benefit of training our model on Bitcoin related data is that much of the dialogue driving its value is public, and given the very nature of Bitcoin as a digital peer-to-peer platform, the leading traders of Bitcoin have a large presence online. In comparison, the London Stock Exchange, is primarily driven by large institutional investors, and a lot of the valuable information driving these markets is often protected [33] given the various conflicts of interest in the financial community, thus, we would expect Bitcoin to be much more accurately modelled using data scraped from the web, than large corporate entities like those listed on the London Stock Exchange.

The reason we decided not to include more cryptocurrencies, is due to the high Bitcoin dominance, the percentage of the entire cryptocurrency market accounted for by Bitcoin and there is also a strong historic correlation between rises in Bitcoin's market cap, and rises in the market cap of other major cryptocurrencies [34], and so the unique value of information from each new currency quickly begins to diminish. However we acknowledge that this may change in the future.

```
[[2408  70]
 [2402  84]]
Accuracy:  0.5020145044319098
Precision:  0.5454545454545454
Recall:  0.03378921962992759
```

Figure 7: Metrics for the densely connected model

The reason we decided to try to predict Bitcoin's trading price one hour into the future, is that a period too short, such as one minute, might be influenced too much by stochastic noise to obtain decisive results, and a period too long, such as one month, leaves enough time for the sentiment to change significantly. Thus we hope time-steps of one hour will be most suitable.

The dense model achieved an accuracy of around 50% (as seen in Figure 7), which is what we would expect to see, given this problem was modelled as a binary classification algorithm and due to its simplicity. That is, the model is correct 50% of the time. We can further observe that the recall was extremely low. After observation of the LSTM using linear regression, it was found that the models were, generally, predicting prices to be lower than they actually were. This could explain why the large number of false predictions of class 0 (a decrease in Bitcoin price), and consequently, the reason for such a low recall score. After plotting a graph of the loss, we could see that the loss continued to decrease, however, the graph of the accuracy clearly showed that the model could not achieve an accuracy greater than 50%; no better than randomly guessing which class. In addition, this model did not perform better than the `most_frequent` and `prior` dummy classifier. It became clear that another approach would be needed to obtain any significant results.

Initially, the LSTM model was created using a `window_len` of 10 in the `stack_windows` function, gathering information from the 10 previous hours, and using the three features used previously on the

Dense model; volume of BTC transacted, volume of corresponding currency transacted in the given window, and VWAP- Volume Weighted Average Price. This performed quite poorly and did not have an accuracy higher than the Dense model or the 'most frequent' and 'prior' dummy classifiers. After completing some research into why this might be, it was found that we needed to split the temporal data into training and testing sets systematically, as mentioned in section 4.2. One adjustment we made to the LSTM (Classification) model was adding more features. These included block size, transaction count, transaction fees and number of keywords found in Reddit post titles. In the early stages, the algorithm only took into account three features, but after ensuring the data was preprocessed correctly, and with the addition of the numerical features and textual data from Reddit, we found this improved the performance greatly, with an accuracy of around 54.4%, as shown in Figure 8. Furthermore, we can see in Figure 9 that both the training and validation loss were lower than the naive loss. Also, we can observe that the loss is continually decreasing and this could suggest a convergence of the loss after a larger number of epochs.

```
[[2939  1294]
 [2563  1662]]
Accuracy:  0.54389820288484275
Precision:  0.5622462787550744
Recall:  0.3933727810650888
```

Figure 8: Metrics for LSTM classification model

6 Conclusion

We found that both an LSTM & LSTM + Attention model surpassed the baseline loss for the test set. Moreover, to our surprise, the LSTM model outperformed the LSTM + Attention model and made better predictions on the test set. It is possible that the addition of the Attention layer would start to result in tangible performance gains if we were to increase the size of `window_len` beyond 15. Attempting such, we found that the gradient explosion occurred and the loss diverged. The results for a window length of 15 can be seen in Figure 3.

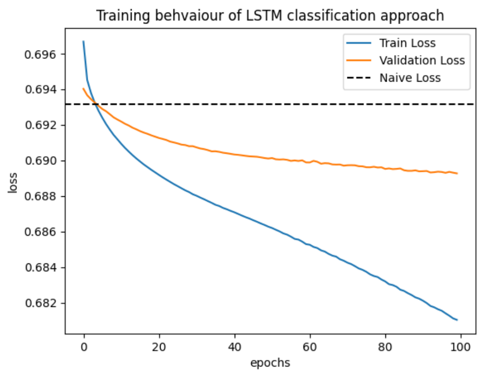


Figure 9: Loss convergence for LSTM classification model

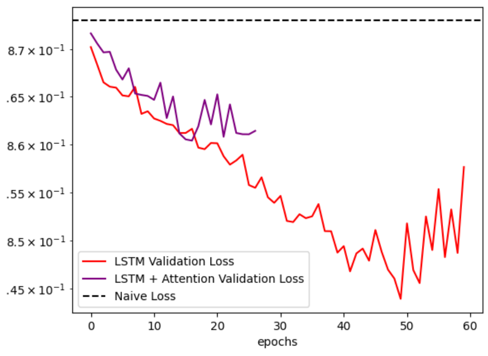


Figure 10: Comparison of loss convergence between models

To demonstrate how these predictions may be used, we designed trading strategies that update their portfolios based on the predictions made by the respective models. Each strategy is to hold Bitcoin only during the hours when the model predicts an upwards price movement and hold cash otherwise. To illustrate this; Let V_i be the value of the portfolio on time-step i and U_i be the indicator function which is 1 if the model predicts an an upwards movement and 0 otherwise. Let B_i be percentage change in the price of Bitcoin. Then we have:

$$V_{i+1} = \begin{cases} 1 & \text{if } i = 0 \\ V_i \cdot (B_i \cdot U_i + 1) & \text{if } i > 0 \end{cases}$$

Indeed, it makes sense to compare these with a naive strategy which simply holds Bitcoin during all time intervals. We refer to this as the 'hold' strategy and in some sense it can be viewed as the strategy that makes use of the naive model's predictions where U_i is constant.

In Figure 11 the superiority of the LSTM + Attention model highlights that it is not only important to predict price movements correctly, but it is also of great importance that we make correct predictions for large price movements, in the following section we suggest an improvement with this in mind. One should note that this should not be taken as rigorous testing; We do not expect to be able to replicate these results under real conditions due to exchange fees and other factors.

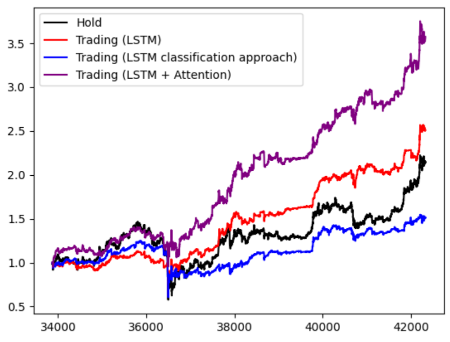


Figure 11: A comparison of trading strategies

7 Future work

One straightforward way in which to improve our models would be to consider alternative choices of the loss function, as alluded to in the conclusion. It may be the case that there are loss functions that more closely resemble what we want to optimize for when pursuing financial gain or mitigating risk. For the classification scenario specifically, we could weight the loss at each time-step by the magnitude of its price movement to make it behave more like a regression model. In doing so, we allow the classification model to understand that it is of greater importance to predict larger price movements correctly.

Another limitation of our research is the lack of fine-tuning on the hyperparameters of our models; particularly the LSTM (Classification) model and Dense model. Performing a hyperparameter sweep on both models would ensure the optimal parameters and could, in theory, improve the models' performance and accuracy significantly. This could be carried out by implementing Bayesian hyperparameter optimisation [9] on each model, instead of random or grid search, it builds a probabilistic model that maps hyperparameter values to a validation set objective and iteratively evaluates certain hyperparameters based on the model's current configuration. After updating the structure and repeating the process, the method aims to find the optimal hyperparameter configuration, and in contrast to random and grid search optimisation, it keeps track of findings from past evaluations. This results in better hyperparameters in fewer iterations and would be the direction we would take in the future.

Aside from changes to the model architecture, the use of additional features may improve the models we already have. Further features that could be engineered could include indicators of sentiment and visibility such as upvotes on reddit, comment embeddings weighted by upvotes or number of new subscribers to r/Bitcoin. Furthermore, given the growing influence of social media, it may prove beneficial to weight textual data by how important and influential their author is. This would involve a larger collection of data, and possibly using data from other social media platforms, such as Twitter or Facebook. The comment embeddings could also be fed into the Attention layer directly, and this way the attention mechanism would learn which tokens are associated with large changes in the value of Bitcoin.

We were not able to evaluate several other machine learning algorithms that might have been relevant to the problem. Our intention is to examine other models, such as ARIMA (autoregressive integrated moving average). This is another statistical method for time series forecasting and these models, like LSTM, take into account the information from previous calculations to predict future values. The three most important parameters in ARIMA are, p (past values used for forecasting the next value), q (past forecast errors used to predict the future values), d (order of differencing)While ARIMA parameter tuning can be very time-consuming, auto ARIMA can be implemented to select the ideal combination of p, q and d that results in the least error. Comparing more machine learning algorithms will give us a better idea as to how best to model this problem, and is how we plan to continue our research.

A final avenue of research could be to apply our models to other cryptocurrencies, and our model may perform better for some of these. Additionally, we could use the movements of correlated cryptocurrencies and markets to better forecast the BTC/USD market. As mentioned in section 5, this went beyond our interest for this particular project, but could be explored in the future, particularly if the Bitcoin dominance decreases significantly.

References

[1] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[2] David MQ Nelson, Adriano CM Pereira, and Renato A de Oliveira. Stock market's price movement prediction with lstm neural networks. In *2017 International joint conference on neural networks (IJCNN)*, pages 1419–1426. IEEE, 2017.

[3] Li-Chen Cheng, Yu-Hsiang Huang, and Mu-En Wu. Applied attention-based lstm neural networks in stock prediction. In *2018 IEEE International Conference on Big Data (Big Data)*, pages 4716–4718. IEEE, 2018.

[4] Varun Gupta and Ankit Pandey. Crude oil price prediction using lstm networks. *International Journal of Computer and Information Engineering*, 12(3):226–230, 2018.

[5] Tawum Juvert Mbah, Haiwang Ye, Jianhua Zhang, and Mei Long. Using lstm and arima to simulate and predict limestone price variations. *Mining, Metallurgy & Exploration*, 38(2):913–926, 2021.

[6] Micael Ferreira, Sven Rodrigues, Catarina I Reis, and Marisa Maximiano. Blockchain: A tale of two applications. *Applied Sciences*, 8(9):1506, 2018.

[7] Luisanna Cocco, Roberto Tonelli, and Michele Marchesi. An agent based model to analyze the bitcoin mining activity and a comparison with the gold mining industry. *Future Internet*, 11(1):8, 2019.

[8] Satoshi Nakamoto and A Bitcoin. A peer-to-peer electronic cash system. *Bitcoin.–URL: https://bitcoin. org/bitcoin. pdf*, 4, 2008.

[9] Zhiqiang Ma, Grace Bang, Chong Wang, and Xiaomo Liu. Towards earnings call and stock price movement. *arXiv preprint arXiv:2009.01317*, 2020.

[10] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[11] Christoph Kilian Theil, Samuel Broscheit, and Heiner Stuckenschmidt. Profet: Predicting the risk of firms from event transcripts. In *IJCAI*, pages 5211–5217, 2019.

[12] Arthur E Hoerl and Robert W Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970.

[13] Peter J Huber. Robust estimation of a location parameter. In *Breakthroughs in statistics*, pages 492–518. Springer, 1992.

[14] Will Koehrsen. A conceptual explanation of bayesian hyperparameter optimization for machine learning, 2018.

[15] Lee A Smales. "brexit": A case study in the relationship between political and financial market uncertainty. *International Review of Finance*, 17(3):451–459, 2017.

[16] William Yang Wang and Zhenhao Hua. A semiparametric gaussian copula regression model for predicting financial risks from earnings calls. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1155–1165, 2014.

[17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *arXiv preprint arXiv:1706.03762*, 2017.

[18] Guokun Lai, Wei-Cheng Chang, Yiming Yang, and Hanxiao Liu. Modeling long-and short-term temporal patterns with deep neural networks. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 95–104, 2018.

[19] Franco Valencia, Alfonso Gómez-Espinosa, and Benjamín Valdés-Aguirre. Price movement prediction of cryptocurrencies using sentiment analysis and machine learning. *Entropy*, 21(6):589, 2019.

[20] Francis EH Tay and Lijuan Cao. Application of support vector machines in financial time series forecasting. *omega*, 29(4):309–317, 2001.

[21] Qing Li, TieJun Wang, Ping Li, Ling Liu, Qixu Gong, and Yuanzhu Chen. The effect of news and public mood on stock movements. *Information Sciences*, 278:826–840, 2014.

[22] Cathy Yi-Hsuan Chen and Christian M Hafner. Sentiment-induced bubbles in the cryptocurrency market. *Journal of Risk and Financial Management*, 12(2):53, 2019.

[23] Ziniu Hu, Weiqing Liu, Jiang Bian, Xuanzhe Liu, and Tie-Yan Liu. Listening to chaotic whispers: A deep learning framework for news-oriented stock trend prediction. In *Proceedings of the eleventh ACM international conference on web search and data mining*, pages 261–269, 2018.

[24] Chuan-Ju Wang, Ming-Feng Tsai, Tse Liu, and Chin-Ting Chang. Financial sentiment analysis for risk prediction. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pages 802–808, 2013.

[25] Alphabet Inc. Youtube api. https://developers.google.com/youtube, 2018.

[26] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. *arXiv preprint arXiv:1310.4546*, 2013.

[27] Jin-Chuan Duan and Jean-Guy Simonato. Empirical martingale simulation for asset prices. *Management Science*, 44(9):1218–1233, 1998.

[28] Jason Baumgartner. pushshift.io. https://pushshift.io/, 2018.

[29] Lin Zehui, Pengfei Liu, Luyao Huang, Junkun Chen, Xipeng Qiu, and Xuanjing Huang. Dropattention: A regularization method for fully-connected self-attention networks. *arXiv preprint arXiv:1907.11065*, 2019.

[30] Stack overflow. What is the architecture behind the keras lstm cell? https://i.stack.imgur.com/RHNrZ.jpg, 2019.

[31] philpperemy. keras-self-attention. https://github.com/philipperemy/keras-attention-mechanism, 2020.

[32] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*, 2015.

[33] Trang Hoang, Emma Neuhauser, and Hossein Varamini. The linkage between insider trading activities, market efficiency, and stock information content. *The Business & Economics Digest*, page 22, 2015.

[34] Darko Stosic, Dusan Stosic, Teresa B Ludermir, and Tatijana Stosic. Collective behavior of cryptocurrency price changes. *Physica A: Statistical Mechanics and its Applications*, 507:499–509, 2018.