Financial Econometrics
Spring Semester 2023

Sentiment analysis of Bitcoin-related tweets and predictive power on Bitcoin value (BTCUSD)

Nicole Shukry (295798)
Zakarya Souid (287759)
Group 8

Professor Elise Marie Gourier

Contents

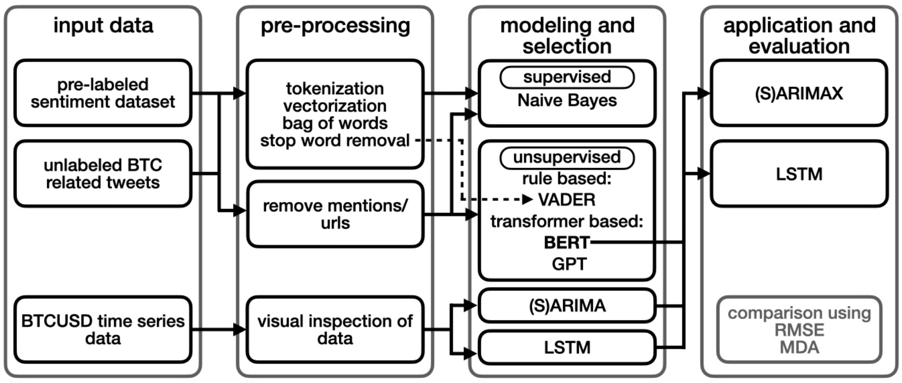
1 Abstract

This project explores the predictive power of Bitcoin-related tweets on the value of Bitcoin (BTCUSD) using sentiment analysis and various machine learning models. We first review the existing literature on the subject, highlighting the growing interest in the predictive power of social media sentiment on cryptocurrency prices. We then detail our data extraction, cleaning, and exploration process, which involves gathering sentiment analysis data and forecasting data. Our sentiment analysis data consists of both pre-labeled sentiment data and unlabeled tweets, while our forecasting data focuses on historical Bitcoin prices.

In our feature engineering phase, we prepare our data for analysis and model training. We then apply Natural Language Processing (NLP) techniques for sentiment analysis, testing several models including VADER, Multinomial Naive Bayes, CryptoBERT, custom fine-tuned FinBERT, and ChatGPT. Each model's principle, data pre-processing steps, implementation, and results are discussed in detail.

Following sentiment analysis, we turn our attention to price forecasting. We begin with a baseline model of Time Series Forecasting, providing a brief background and discussing our implementation and results. We then explore the use of Long-Short Term Memory (LSTM) for price prediction, explaining its logic and detailing our implementation process.

Our study concludes with the implementation of our chosen models and a discussion of their predictive power on Bitcoin value. Our findings try to contribute to the growing body of literature on the influence of social media sentiment on cryptocurrency prices, offering valuable insights for investors and researchers alike.



Simplified Project Framework

2 Introduction

Among financial assets, cryptocurrencies, most notably Bitcoin (BTC) (Nakamoto, 2009), have piqued the interest of many due to their volatile nature. The drivers of these fluctuations remain a subject of ongoing exploration. By its decentralized nature, Bitcoin does not rely on a central authority, but rather a peer-to-peer system, making its value considerably more influenced by general investor sentiment and speculation, as Central Banks' monetary policy cannot have a direct effect on its price. Consequently, understanding the sentiments expressed about Bitcoin may aid in better predicting its value.

The rise of social media platforms has brought about new methods for monitoring and predicting market behaviors (Bollen et al., 2011), and with the advancement of research, particularly in machine learning, exploring what could possibly be return-generating information from the available data is becoming an increasingly tangible reality (Baker and Wurgler, 2007). Machine learning techniques now enable us to extract, analyze, and interpret vast amounts of unstructured data, transforming it into actionable insights that can guide decision-making, market prediction, and strategy development in the financial sector.

A great amount of research has already been done utilizing financial news sentiment to predict traditional markets price directions (Liapis et al., 2023; Fazlija and Harder, 2022; Rao and Srivastava, 2012; Mittal, 2011), and in a similar fashion, more recent studies have correlated BTC's market behavior with public sentiment. Promising results show that price direction prediction be made more accurately when including sentiment information (Critien et al., 2022; Sattarov et al., 2020), as well as effectively boosting the performance of certain BTC trading models (Kim et al., 2023; Nazemi et al., 2020).

In this project, our aim is to add to the existent research by comparing different models' performance, both in the sentiment analysis area as well as the price forecasting area, as well as introducing previously untested methods which could possibly be further improved on. We furthermore investigate the predictive power of Twitter sentiment on Bitcoin value.

3 Previous Literature

Bitcoin price forecasting has previously been attempted without the inclusion of sentiment analysis, using both traditional time series methods such as ARIMA (Wen et al., 2022) as well as more complex machine learning and deep learning methods such as Long Short Term Memory Cells (LSTM), Convolutional and Reccurent Neural Network (CNN, RNN) and hybrid models. Generally, previous work has shown that deep learning models outperform the classical time series models in forecasting long time intervals (Mcnally et al., 2018; Chen et al., 2021; Hua, 2020). Moreover, Sezer et al. (2020), by reviewing a large set of deep learning for financial time series forecasting articles, found that LSTM and its variations along with some hybrid models dominate the financial time series forecasting domain. However, Hua (2020) also found that for short time interval forecasting, i.e 1 day, ARIMA performed better than LSTM. In addition, several studies showed that ARIMA may be more accurate for crypto price forecasting: Pagariya et al. (2022) found ARIMA to be twice as accurate than LSTM for cryptocurrency forecasting on 4 different cryptos including BTC. Serafini et al. (2020) also found that ARIMAX (ARIMA with eXogenous variables) performed better than LSTM in predicting the daily weighted price of Bitcoin by incorporating daily tweets sentiment to their models.

The inclusion of sentiment analysis of Twitter data for price prediction has, itself, been implemented many times, due to the difficulty of understanding the drivers of Bitcoin value:
Preisler et al. (2019) used multiple sentiment analysis models on a human labeled dataset: VADER, Random Forest and Support Vector Machines (SVM) and 2 deep learnings models (Hierarchical Deep Learning for Text Classification (HDLTex) and Convolutional Neural Network for Sentence Classification (CNNSC)) and they found that VADER and CNNSC have an overall equivalent accuracy (they claim it's due to the small size of the dataset). They then showed using time series that the development of Bitcoin prices and investor sentiment are closely related.
Nazemi et al. (2020) used a 2-stage forecasting system based on LSTM and showed the superior performance of incorporating sentiment input relative to machine learning models without input from financial news, and obtain a higher out-of-time rate of return with the proposed forecasting system than with a buy-and-hold strategy.
Kim et al. (2023) pre-trained crypto-specific Language Models in Korean and showed that including sentiment scores along with BTC chart data boosts the performance of BTC trading models and outperformed buy and hold by a large margin in normal markets.
Jagini et al. (2023) calculated the sentiment associated with tweets using VADER to which they also added a slang dictionary, they then used a machine learning algorithm, linear regression, to predict Bitcoin prices by incorporating, in addition to a daily "score sentiment", the volume of tweets related to Bitcoin in that day. They find that the sentiment of tweets does correlate with the shift in Bitcoin price.

4 Data extraction, cleaning and exploration

4.1 Sentiment Analysis Data

4.1.1 Pre-labeled sentiment data

Due to the requirement of training certain models, a labeled dataset was necessary. We downloaded a dataset consisting of 1000 Reddit comments about crypto, categorized according to Positive or Negative sentiment from Surge AI. This dataset was human labeled by Surge labelers.

4.1.2 Unlabeled tweets

The unlabeled data was downloaded from Kaggle. We initially wanted to create our own dataset but Twitter's API and Reddit's API (another potential source of interesting information) changed their rules and prohibited us from using an already existing scrapper like Tweepy. This forced us to create our scrapper through browser automations but we found ourselves also limited by the request rate of Twitter.

The downloaded Dataset was 2.1GB of raw text, amouting to 90M tweets. This was obviously complicated to process at once so we decided to only take tweets from accounts with above 10k followers in a first time, and then accounts with more than 500k followers.

We then noticed that even with this first filter, the quality of the accounts were not guaranteed: for example, there were a lot of scam accounts trying to offer "Free BTC" that Elon Musk apparently had to give away.

4.2 Forecasting Data

Daily Bitcoin prices data were downloaded from Yahoo Finance, spanning from the same date set than the tweet corpus.

4.3 Data analysis and summary statistics

One problem is that the unlabeled tweets dataset had some missing dates on the timespan that we were analyzing. To deal with this issue, we considered that on those days, the daily sentiment would have simply been "Neutral".

For the labeled dataset, it was split 2 ways, with 302 Positives and 260 Negatives.

5 Feature engineering

Since we tried to predict Bitcoin prices at a daily frequency, we decided to sum the sentiments within a day. We also wanted to capture the impact of the reach of a tweet, so we created a feature from sentiment that we called "Reach', constructed from the sentiment times the number of followers the tweeter had. Having the number of impression a tweet had would have been even better but we thought it was a good compromise to construct our indicator this way.

6 Natural Language Processing (NLP) for Sentiment Analysis

The metrics used to compare the models are accuracy (proportion of correctly predicted instances out of all instances) and F1 score, which is the harmonic mean of precision and recall. Precision is the proportion of correctly predicted positives out off all positive predicted instances, while recall is the is the proportion of correctly predicted positive instances out of all actual positive instances.

For models which need to be trained or custom fine-tuned, the calculation is done on a "test" set which consists of a subset of the pre-labeled dataset. For models which do not need training, accuracy and F1 scores are calculated on the entirety of the pre-labeled dataset.

6.1 BaseLine Model: VADER

6.1.1 Principle

VADER (Valence Aware Dictionary and sEntiment Reasoner) is a lexicon and rule-based model. It provides both polarity (positive/negative) and score (intensity), proposed by Hutto and Gilbert (2014). This model was developed to work well on social media style text, which is why we decided to use it as baseline model. To test this model, we used the pre-labeled dataset of reddit comments.

6.1.2 Data Pre-processing

The pre-processing included removing stop-words, lowercasing, removing urls/mentions/hashtags, removing duplicate spaces, line-breaks and special characters.

6.1.3 Implementation and results

Because VADER provides both polarity and intensity, we adapted the output to be consistent with our dataset labels. Generally, one can include a "Neutral" class for scores between $[-0.05, 0.05]$, however, we considered all scores above 0 to mean "Positive", and all scores below zero to mean "Negative". The overall accuracy we find is 59%, and the F1 score is 0.64.

6.2 Multinomial Naive Bayes

6.2.1 Principle

The equation for Naive Bayes is the following:

$$P(\text{sentiment class}|\text{word}_1, \text{word}_2, ..., \text{word}_n) = \frac{P(\text{word}_1, \ldots \text{word}_n|\text{sentiment class})P(\text{sentiment class})}{P(\text{word}_1, \text{word}_2, ..., \text{word}_n)}$$

If features, ie words, are "naively" assumed to be independent of each other, this is equivalent to:

$$P(\text{sentiment class}|\text{word}_1, \text{word}_2, ..., \text{word}_n) = \frac{P(\text{sentiment class}) \prod_{i=1}^{n} P(\text{word}_i|\text{sentiment class})}{P(\text{word}_1, \text{word}_2, ..., \text{word}_n)}$$

This means that the likelihood probability (amount of times word i appears among all words in all documents of class k) is calculated in the following way: first concatenate all documents of class k into one big "class k" text. Then use the frequency of word i in this concatenated document to give the likelihood probability:

$$\hat{P}(\text{word}_i|k) = \hat{P}_{i,k} = \frac{N_{i,k}}{N_k}$$

where $N_{i,k}$ is the count of word i in a sample of class k in the training set, N_k is the total count of all words for class k. To avoid problems where a word is not present in the training sample, an alternative, smoothed version of maximum likelihood is:

$$\hat{\theta}_{i,k} = \frac{N_{i,k} + \alpha}{N_k + \alpha n}$$

with n the total number of words in the training set, and $\alpha \geq 0$ is the Laplace smoothing coefficient.

6.2.2 Data Pre-processing

The Naive Bayes classifier classifies the dataset based on features. To do so, words must be turned into quantitative numerical values. Thus a bag of words model is used. To implement the bag of words model, we first clean the data: lowercasing, removing urls/mentions/hashtags, removing duplicate spaces, line-breaks and special characters.

6.2.3 Implementation and results

We implement two bag of words models: a CountVectorizer and a TF-IDF (Term Frequency-Inverse Document Frequency) Vectorizer. The CountVectorizer simply counts the number of times a word appears in a document. For the TF-IDF, where terms are the words of a tweet, and documents are the different tweets, a weighting factor takes into account not only how many times a word appears in a document but also how important that word is to the whole corpus of tweets. In both cases, we split the pre-labeled dataset into 90% train, 10% test. With CountVectorizer, we find an accuracy of 80% and an F1 Score of 0.80. With TF-IDFVectorizer we find an accuracy of 78% and an F1 Score of 0.78. CountVectorizer performs slightly better than TFIDF in our case, probably because the lengths of texts in the dataset vary a lot. The hyperparameter α was tuned using a gridsearch.

6.3 BERT Models: CryptoBERT and custom fine-tuned FinBERT

6.3.1 Principle

BERT (Bidirectional Encoder Representations from Transformers), introduced by Devlin et al. (2018), is designed to pre-train deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers, making it optimal for sentence-level tasks. Originally, BERT was trained on BookCorpus and English Wikipedia.

The pre-trained BERT model can be (inexpensively) fine-tuned with one additional output layer: both FinBERT and CryptoBERT are fine-tuned BERT models. FinBERT (Araci, 2019) was further pre-trained using a subset of Reuters' news articles published between 2008 and 2010, and trained on classification using around 5k english financial news sentences that were human labeled by people with background in finance and business.

On the other hand, CryptoBERT is based upon the already pre-trained model BERTweet (Nguyen et al., 2020), which follows an optimized training procedure following RoBERTa (Liu et al., 2020), and trained on a corpus of 850M English Tweets from 2012 to 2019. CryptoBERT was further fine-tuned on the cryptocurrency domain, using a corpus of 3.2M cryptocurrency-related social media posts from StockTwits, Twitter, Reddit and Telegram.

6.3.2 Data Pre-processing

BERT models do not require heavy pre-processing as the whole sentence is taken into account for classification. The only preprocessing done was to remove mentions and URLs.

6.3.3 Implementation and results

CryptoBERT did not adhere to the classification of our test data. It answers Bearish, Neutral or Bullish and our test data did not include a Neutral. When only considering False Positives and False Negatives (i.e. dropping the neutrals), its performance were the best of all models. However, it discarded most of the data by doing so, as it thought most of the data was "Neutral". It had much better accuracy at predicting positive sentiment rather than negative, and this may be due to it being trained on datasets in which sentiment is generally pro-crypto.

Considering all data points, our Finetuned BERTweet had better performance. It achieved an overall accuracy of 93% on the test data.

6.4 ChatGPT

6.4.1 Principle

The GPT (Generative Pretrained Transformer) model, a transformer-based architecture, is leveraged for sentiment analysis due to its proficiency in understanding context and generating human-like text. The model is pretrained on a large corpus of text data and fine-tuned for specific tasks like sentiment analysis. It processes input text token by token, utilizing its self-attention mechanism to weigh the influence of each token on others in the sequence. This enables GPT to capture intricate patterns and dependencies in the text, thereby accurately classifying the sentiment expressed.

Here, through OPEN AI's commercially-available GPT3.5, we used it to that purpose only. We chose OPEN AI's version over other GTPs because the availability of their API and its superior performance to other GPT models when it comes to following a prompt.

6.4.2 Data Pre-processing

No preprocessing of the data was done.

6.4.3 Implementation and results

The implementation of a GPT model to do sentiment analysis required to give it a role so that the Text Generation feature's output was what we expected. Despite this, sometimes the model behaved out of character and answered completely beside the point.

Local GPT options were nowhere near as effective, both in term of speed (which was already a big issue of ChatGPT) and accuracy.

The key deciding factor was however price. To test on 500 tweets, it already cost 21 cents. Scaled up to a 400k tweet corpus, that was already 160 USD. However, the model performed quite well on the task and achieved an overall accuracy of 83%.

6.5 Final NLP Model Selection

A summary table of the accuracy and F1 score for each model is given below:

Model	Accuracy	Positive Acc.	Negative Acc.	F1 Score
VADER	59	68	48	65
Naive Bayes	84	93	74	84
CryptoBERT	63	78	46	73
ChatGPT	83	77	90	91
BERTweet (Finetuned)	93	91	94	93

Table 1: Performance metrics for the five models (in %)

Positive accuracy is the percentage of correctly classified positives out of all actual positives, while negative accuracy is the percentage of correctly classified negatives out of all actual negatives. We chose to use our Finetuned BERTweet for its high accuracy and relative ease of implementation.

7 Price Forecasting

The models are compared using 2 different performance metrics: Root Mean Squared Error (RMSE) and Mean Directional Accuracy (MDA). Both are calculated for the test/prediction series when the model is used for forecasting.

7.1 BaseLine Model: Time Series Forecasting

Traditional time series methods such as ARIMA (AutoRegressive Integrated Moving Average) have been used widely in the modelling of financial markets. However, statistical models like ARIMA do not have flexibility in handling non-linearity in the financial data. We implement a traditional time series method mainly to present a basis for comparison of time series forecasting with regards to machine learning models, but these models can also incorporate exogenous explanatory variables such as sentiment scores.

7.1.1 Brief Background

ARIMA models consist of:

AutoRegressive (AR): The AR component represents the relationship between an observation and a certain number of lagged observations (auto-correlation). It models the dependence of the current value on previous values from the series.

Integrated (I): The I component accounts for the differencing required to make the time series stationary by removing the trend.

Moving Average (MA): The MA component represents the relationship between the observation and the residual errors from a moving average model applied to lagged values of the series. It helps capture the short-term dependencies in the series.

According to Wold's decomposition theorem (Wikipedia contributors, 2022), an ARMA model is sufficient to describe a (wide-sense) stationary time series. Thus one of the first requirements of applying ARIMA is to test for the stationarity of the BTCUSD time series, and if required making the appropriate transformations to make it stationary:

For a given process Y_t with trend, if we can construct $W_t = (1-L)^d Y_t$ with L the lag operator such that $LX_t = X_{t-1}$ and $d \geq 0$ some integer such that W_t is stationary, then we have an ARMA(p,q) process with mean μ which satisfies $\varphi(L)(W_t - \mu) = \theta(L)\epsilon_t$. Here, $\varphi(z) = 1 - \varphi_1 z - ... - \varphi_p z^p$ is the AR characteristic polynomial, and $\theta(z) = 1 - \theta_1 z + ... + \theta_q z^q$ is the MA characteristic polynomial.
Thus, Y_t satisfies the equation $\varphi(L)(1-L)^d Y_t = \mu(1 - \sum_{k=1}^{p} \varphi_k) + \theta(L)\epsilon_t$, meaning that Y_t is an ARIMA(p,d,q) process.

SARIMA (Seasonal AutoRegressive Integrated Moving Average) is an extension of ARIMA which takes into account seasonal patterns by having 4 additional parameters: P, D, Q, and s. P, D, and Q are similar to the p, d, and q parameters of ARIMA: however, they apply to the seasonal component of the data. The s parameter represents the length of the seasonal period or cycle.

7.1.2 Implementation and results

Due to our available data of Tweets, we adapted our BTCUSD data by cropping it to the timeframe 2021-02-05 (yyyy-mm-dd) up to 2023-01-09, which gives us 704 datapoints of daily BTCUSD value. We then split the data into training and test sets:

- training set: 2021-02-05 to 2022-09-30
- test set: 2022-10-01 to 2023-01-09

The time series plot is shown in Figure 1.

To assess the stationarity of our (train) series, we used the Augmented Dickey Fuller (ADF) test in which the null hypothesis is that a unit root is present in a time series sample (ie the series is non-stationary). We choose a significance level of 5%. The test statistics are shown for zero ($d = 0$) and first order ($d = 1$) differencing of the series in Table 2.

	$d = 0$	$d = 1$
ADF statistic	-1.672	-28.607
p-value	0.446	0.000

Table 2: ADF test statistic

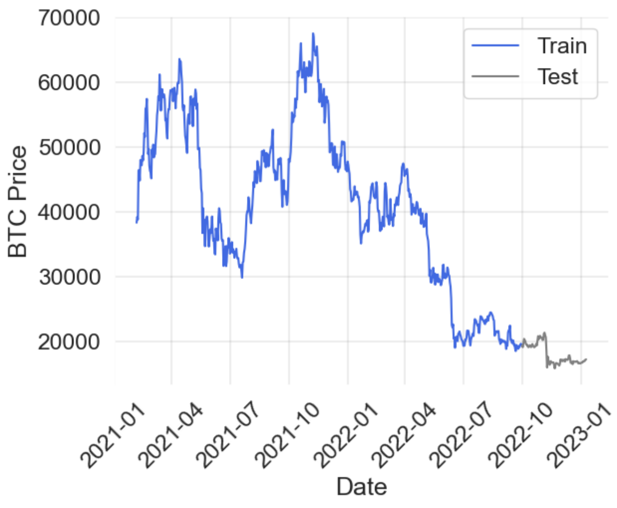


Figure 1: Line plot of the BTCUSD value and training/testing set

For first order differencing, the p-value is below 0.05 thus we can reject the null hypothesis that the first order differenced series is non-stationary. The zero and first order differenced series rolling means and standard deviations are plotted in Figures 2 and 3.

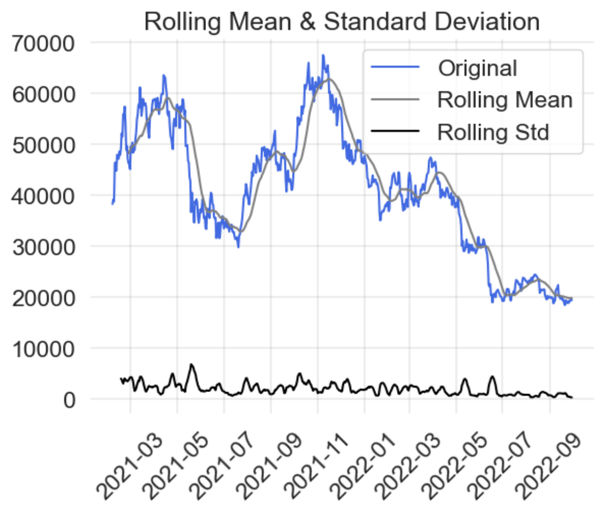


Figure 2: Rolling mean and standard deviation of zero differenced series

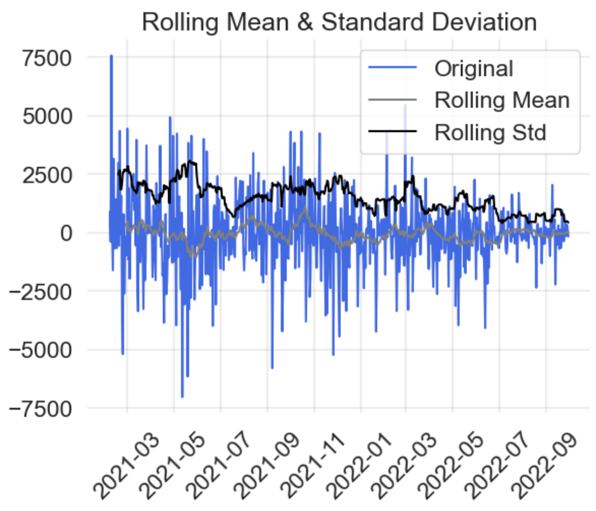


Figure 3: Rolling mean and standard deviation of first order differenced series

We also inspected the first order differenced series for any seasonal trends (shown in Figure 6). Although there is clearly a repeating pattern of seasonality, it is hard to say with certainty what the period of seasonality can be taken to be, but weekly is a plausible estimation.

Evidence regarding the existence of the day-of-the-week effect in Bitcoin returns was shown by Aharon and Qadan (2019), further pushing us to test whether adding weekly seasonality provides us with a better model. However, because it is hard to conclude whether there really is seasonality or not, and the period of seasonality, we decided to implement both an ARIMA and a SARIMA model.

In addition, the ACF (Auto-Correlation Function) and PACF (Partial Auto-Correlation Function) are plotted in Figures 4 and 5. The ACF plot displays the correlation between a time series observation and its lagged values. It provides the Moving Average (MA) order of the ARIMA model.

On the other hand, the PACF plot represents the correlation between an observation and its lagged values while also accounting for the correlation contributed by shorter lags.Thus the PACF plot provides the Autoregressive (AR) order of the ARIMA model.

By analyzing the ACF and PACF plots, we can obtain initial estimates for the order of the AR and MA components in the ARIMA model. The order of the AR component can be determined by the lag at which the PACF plot exhibits significant peaks. Similarly, the order of the MA component is determined by the lag at which the ACF plot exhibits significant peaks. Our initial estimates are $(p, q) = (1, 1)$. Regarding seasonality, there may be some repeating peaks around every 7 days possibly suggesting weekly seasonality, but it is important to note that the peaks remain very close to the 95% confidence interval around 0. For that reason, we keep both an ARIMA and a SARIMA with weekly seasonality $(s = 7)$ models.

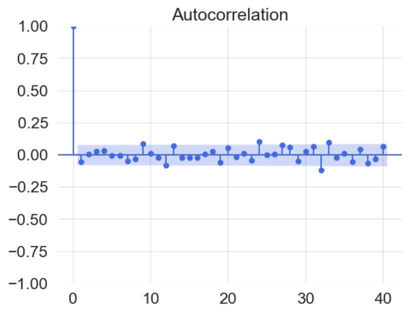


Figure 4: ACF plot of of first order differenced series

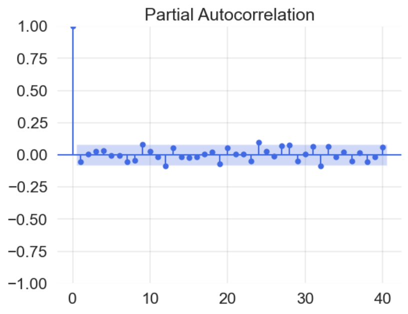


Figure 5: PACF plot of first order differenced series

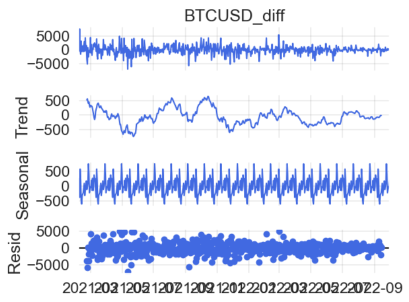


Figure 6: Seasonal decomposition of first order differenced series

We then run a grid search to find the best fitting model. Our criterion of choice is the Akaike Information Criterion (AIC) which is a widely used measure of goodness of fit because it balances the tradeoff between both the model's ability to explain the data (goodness of fit) as well as the models complexity (number of parameters). A better model would have a lower AIC value.

In the grid search process, the range of potential parameter values for the ARIMA and SARIMA models were $(p, d, q) = (\{0, 1, 2\}, 1, \{0, 1, 2\})$ and for SARIMA we also added the seasonal order $(P, D, Q, s) = (\{0, 1, 2\}, \{0, 1, 2\}, \{0, 1, 2\}, 7)$. All combinations were tested and their AIC values compared. The final models selected were ARIMA$(0, 1, 2)$ and SARIMA$(2, 1, 2)_{\times}(0, 0, 1, 7)$.

We predict the bitcoin value for the period of the test set. To do so, we use rolling (or recursive) forecasting. That is, we predict the next day BTCUSD value sequentially for each future time point as new data becomes available. The model is initially fitted to historical data, and then updated and re-fitted with each new observation. Graphs with the prediction as well as actual values are given in Figures 7 and 8. The 2 comparison metrics, RMSE and MDA, are provided in Table 3. We can conclude that although the simple ARIMA model results in a slightly smaller RMSE, the SARIMA model is better at predicting the direction of future prices.

	ARIMA$(0, 1, 2)$	SARIMA$(2, 1, 2)_{\times}(0, 0, 1, 7)$
RMSE	474.28	483.04
MDA	0.48	0.49

Table 3: RMSE and MDA of ARIMA and SARIMA models

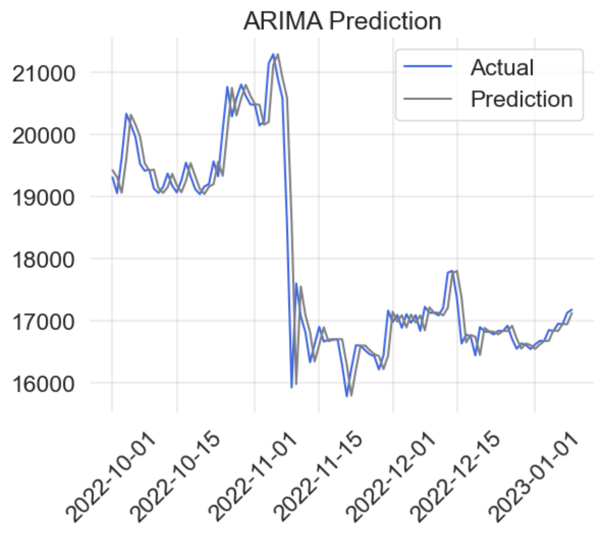


Figure 7: ARIMA$(0, 1, 2)$ prediction

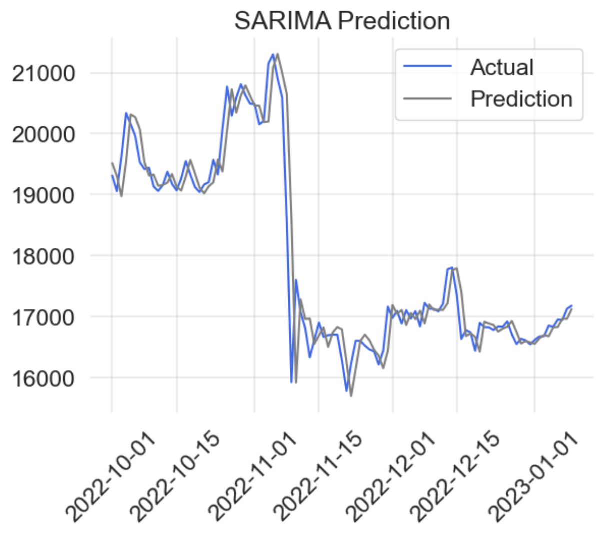


Figure 8: SARIMA$(2, 1, 2)_{\times}(0, 0, 1, 7)$ prediction

7.2 Long-Short Term Memory (LSTM)

Long Short-Term Memory (LSTM) networks are a type of recurrent neural network (RNN) that are designed to remember information for long periods of time. They were introduced by Hochreiter and Schmidhuber (1997) to overcome the limitations of traditional RNNs, particularly the problem of vanishing and exploding gradients, which make it difficult for a standard RNN to learn from information where the gap between the relevant information and the point where it is needed is large.

7.2.1 Brief Background

LSTMs have a chain-like structure, but the repeating module has a different structure than a standard RNN. Instead of having a single neural network layer, there are four, interacting which each other. These four layers work together to decide what information should be kept or discarded at each time step. LSTMs have the ability to remove or add information to the cell state, regulated by structures called gates. Gates are a way to optionally let information through. They are composed out of a sigmoid neural net layer and a pointwise multiplication operation.

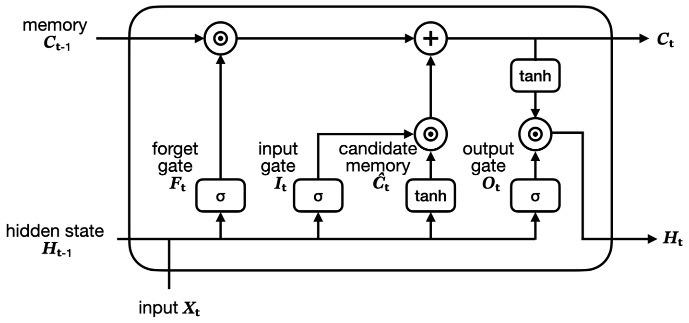


Figure 9: LSTM working principle

There are three types of gates within a unit:

1. Forget Gate: Decides what information we're going to throw away from the cell state.
2. Input Gate: Decides which values we'll update.
3. Output Gate: Decides what the next hidden state should be. LSTMs are widely used for sequence prediction problems and have proven to be extremely effective. They can be used in various applications such as time series prediction, natural language processing, speech recognition, and even in music composition.

7.2.2 Implementation

We built our LSTMs with 100 days sequences. The way we implemented our LSTMs, they take the specified sequence to predict one single day. Other approaches like Direct Multi-step Forecast (where you train a separate model for each future time step - to predict 7 days into the future, you would train 7 different models, where the i-th model is trained to predict i days into the future) are very interesting but the further time away you want to predict, the less likely it is to be correctly predicted.

Again, due to our dataset of Tweets we limited the BTC data to 2021-02-05 to 2023-01-09. We then applied the same split as for Time Series.

Our LSTM model was trained on 200 epochs using the Adam optimizer. It had 32 Hidden dimensions, 2 hidden layers and one input series and one output. Figure 10 is the model's fitted values to the training set. The training loss is also plotted.

Using its predictive abilities over a 100 days, we obtained this prediction. It has to be noted however it is not predicting the full 100 days at once. Each day it uses the past 100 days to predict the following day. Figure 11 shows the prediction plotted against actual values. We tried Bi-directionnal LSTM but could not get it to work.

As seen in Figure 12 rolling LSTM was quite promising. We retrained the model after each day, doing reinforcement learning in a way. Its ability to predict directionality of price change improved with respect to the standard model, but its overall ability to predict price was worse (higher RMSE).

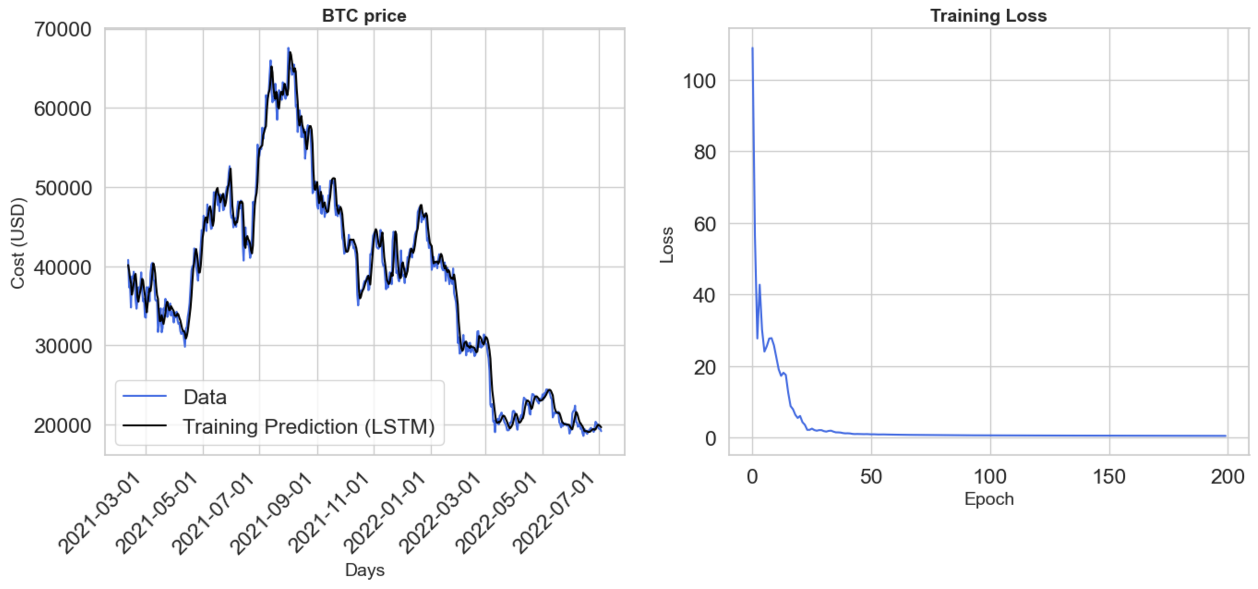


Figure 10: LSTM training, no sentiment

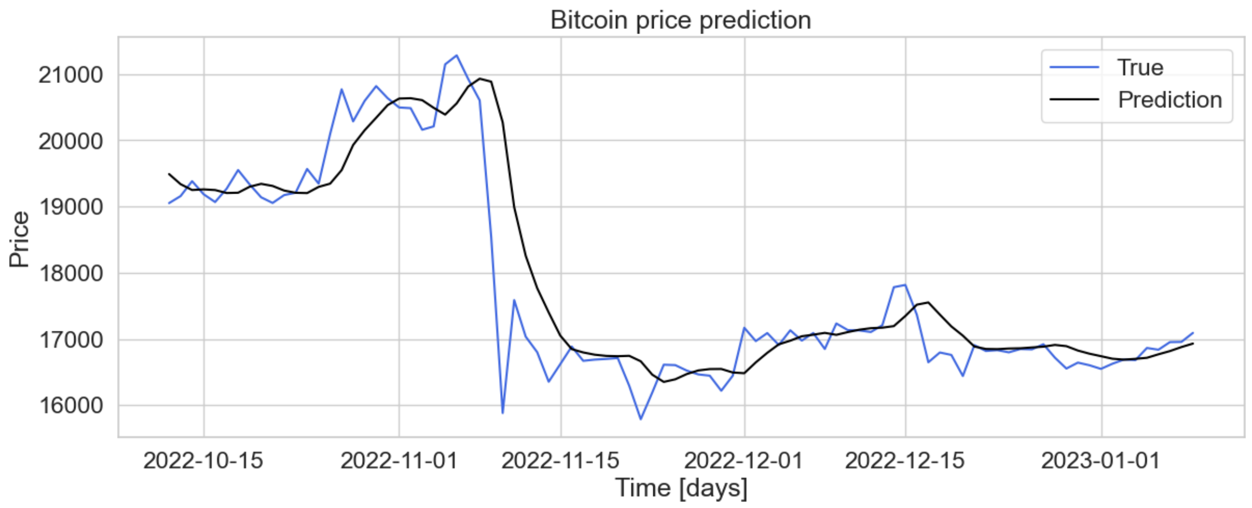


Figure 11: LSTM prediction, no sentiment

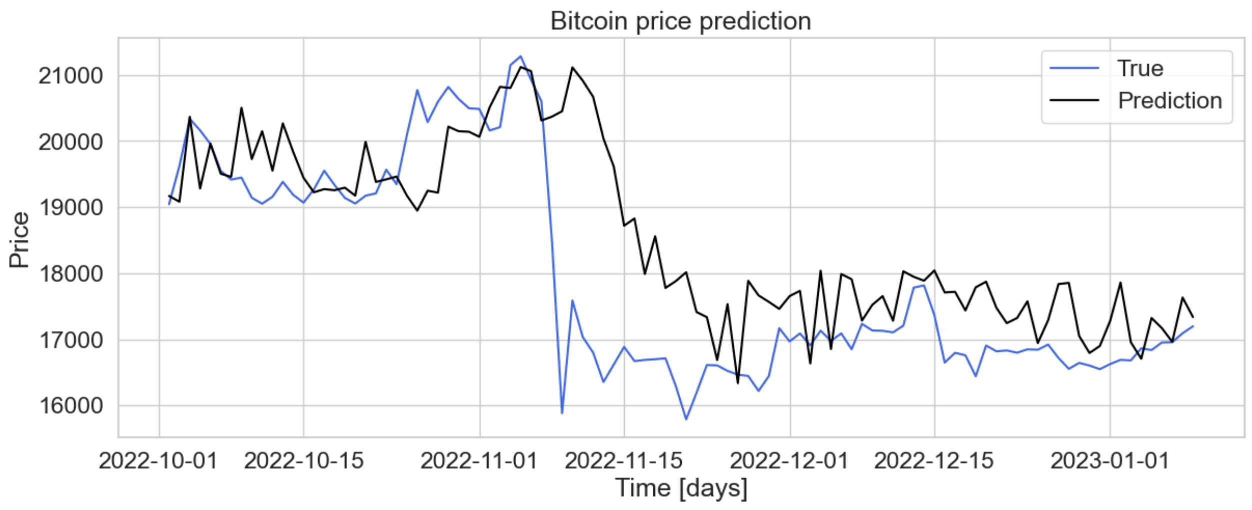


Figure 12: Rolling LSTM prediction, no sentiment

As for the performance metrics, results are in Table 4.

	RMSE	MDA
LSTM	126.53	38%
Rolling LSTM	625.99	57%

Table 4: Performance metrics (RMSE and MDA) for LSTM and Rolling LSTM models

7.3 Preliminary conclusion on price forecasting models

Forecasting prices, particularly in the realm of cryptocurrencies, is a highly intricate task. This is consistent with the theoretical proposition that Bitcoin behaves more like a true Martingale than a stock, implying that its future price changes are independent of past changes and are essentially random (Schilling and Uhlig, 2019).

In their paper, "Trend Prediction Classification for High Frequency Bitcoin Time Series with Deep Learning," Shintate and Pichl (2019) propose a trend prediction classification framework for cryptocurrency time series. They suggest that LSTM models may be outperformed by other deep learning methods in this context. Furthermore, they found that the profit rates based on LSTM did not exceed those of a uniform random process with three outcomes, reinforcing the notion that LSTM's performance in predicting Bitcoin prices is no better than pure randomness.

The inherent complexity and unpredictability of price movements render traditional forecasting methods such as Time Series Forecasting or Long Short-Term Memory (LSTM) networks relatively ineffective. In the context of predicting Bitcoin's daily price direction, these methods perform marginally better than a random guess at best. Rolling LSTM however obtained a 57% success rate on our test data. Even if it is not very high, it is somewhat encouraging.

8 Implementation of chosen models

8.1 BERTweet

Applying our sentiment analysis to our first corpus of tweet was a challenge. It amounted to 400k tweets in total and despite using all the possible hardware acceleration (Neural Engine on Apple Silicon Macs, CUDA on Google Colab), it took roughly 15 secs to analyze a thousand tweets. This meant a total execution time of 1h30min. Separate scripts were created to do this task. Once the data was processed, we decided to regroup them by day and to create a feature taking into account the audience a tweet had and the reach its message could bear, by multiplying the sentiment value by the number of followers of the writer, then normalizing it by dividing by the maximum amount of followers a single user has.

8.2 (S)ARIMAX

ARIMAX (ARIMA with eXogenous variables) incorporates exogenous variables by including them as additional input features in the time series forecasting. We added the daily tweets sentiment to the rolling forecasting to see whether it made the model a better predictor for BTCUSD value.

The incorporation of exogenous variables expands the ARIMA model equation as follows:

$$\varphi(L)(1-L)^d Y_t = \mu(1-\sum_{k=1}^{p}\varphi_k) + \beta X(t) + \theta(L)\epsilon_t$$

Here, $X(t)$ is the exogenous variable (sentiment) at time t, and β is the coefficient associated with the exogenous variable, which is estimated during the model fitting process.

The RMSE and MDA metrics of incorporating the daily sentiment are given in Table 5. The plots of actual vs predicted values are also shown in Figures 13 and 14.

	ARIMAX$(0,1,2)$+dailysentiment	SARIMAX$(2,1,2)_\times(0,0,1,7)$+dailysentiment
RMSE	472.78	481.99
MDA	0.48	0.49

Table 5: RMSE and MDA of ARIMAX and SARIMAX models

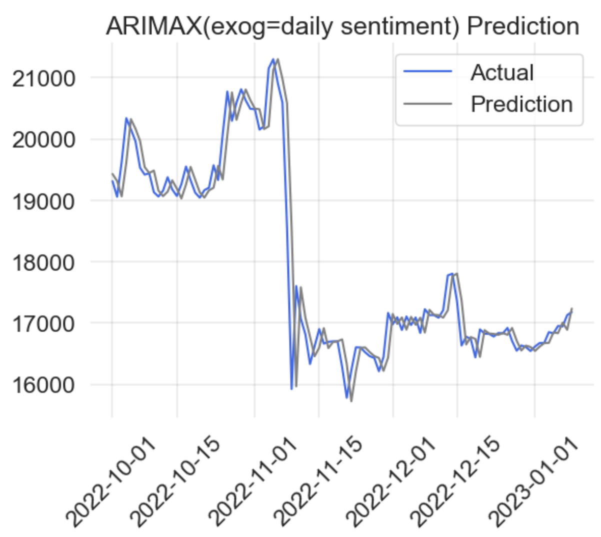


Figure 13: ARIMAX$(0,1,2)$ prediction vs actual

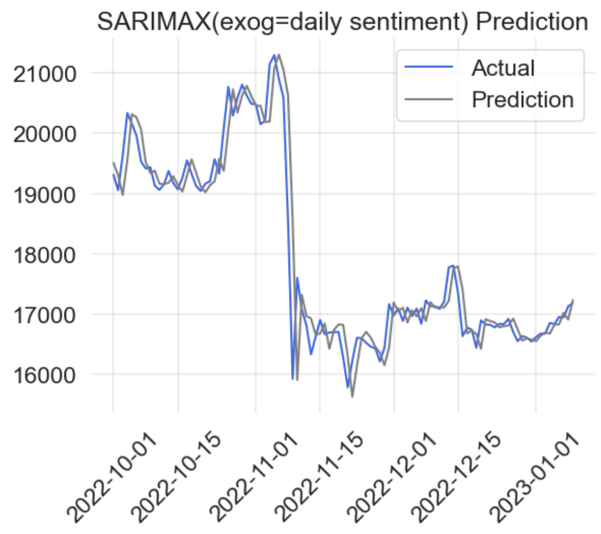


Figure 14: SARIMAX$(2,1,2)_\times(0,0,1,7)$ prediction vs actual

Although adding sentiment slightly reduces RMSE for both models compared to when sentiment is not incorporated, the MDA does not change. More than that, it remains very close to 0.5, meaning that the directionality of price change is only correctly predicted 50% of the time, which is no better than simply guessing. It is important to note that there may however be better ways to pick the initial model by using metrics other than AIC. For example, we could pick the RMSE minimising model, or the MDA maximizing model.

The daily rolling forecasting was done as it was found to give much more interesting results than a longer term forecast. In general, ARIMA-based models work much better for short period forecasts (ie only a few observations in the future) rather than longer period forecasts (Hua, 2020; Pagariya et al., 2022; Serafini et al., 2020). This also helped us better see the effect of daily sentiment.

8.3 LSTM

To incorporate our sentiment analysis to the LSTM models, we increased the input dimensions of the neural network to two and retrained them. We then adapted the number of hidden dimensions and layers to better incorporate these added datapoints. The final settings are reported in Table 6

Model	Number of Dimensions	Hidden Dimensions	Hidden Layers	Epochs	Learning Rate
LSTM	1	32	2	200	0.01
Rolling LSTM	1	32	2	100	0.01
LSTM with Sentiment	2	128	3	1000	0.01
Rolling LSTM with Sentiment	1	32	2	200	0.01

Table 6: Model settings for LSTM and Rolling LSTM with and without sentiment analysis

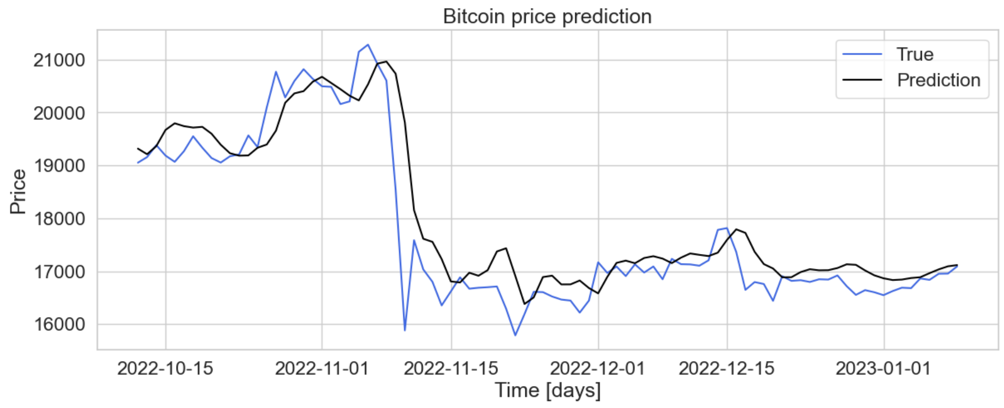


Figure 15: LSTM prediction, with sentiment

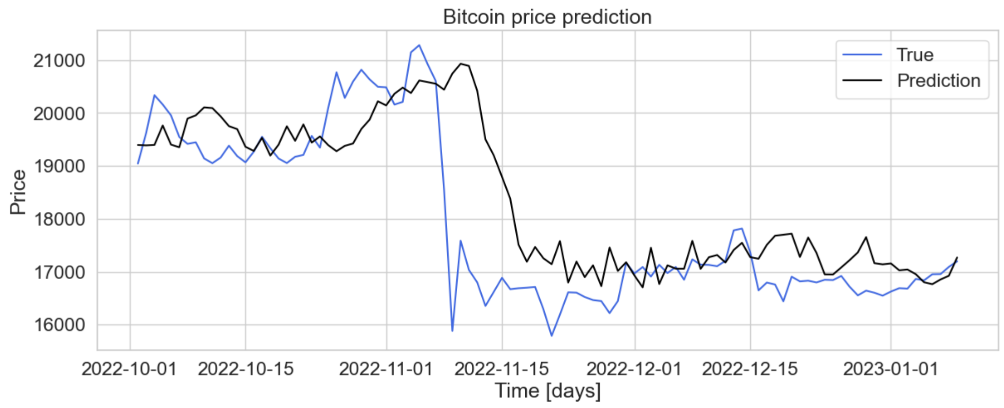


Figure 16: Rolling LSTM prediction, with sentiment

Overall, our augmented models achieved the performance reported in Table 7.

In all cases, adding the sentiment greatly improved MDA. In the case of Rolling LSTM, it even decreased RMSE. Compared with LSTM, Rolling LSTM has a greater RMSE but a much higher MDA: the model is more accurate when predicting the direction of price change, but less precise when it comes to the predicted value.

	Without sentiment		With sentiment	
	LSTM	Rolling LSTM	LSTM	Rolling LSTM
RMSE	126.53	625.99	245.14	440.72
MDA	38%	57%	45%	60%

Table 7: Performance metrics (RMSE and MDA) for LSTM and Rolling LSTM models with and without sentiment analysis

9 Conclusion

From our findings, price alone is not an efficient way to predict future prices. Augmenting features with NLP is interesting but the effect is limited and that could be due to multiple reasons. It seems to have improved the mean directional accuracy in both cases but values are still relatively low. It remains to be seen what trading performances can be achieved with these indicators.
Moreover, in our case, the data containing the tweet is not of sufficient quality. Too many days are missing, too many bots made their way into the corpus... Also, in its entirety, Crypto-Twitter is very pro Crypto. The bias is really strong and even during "Crypto-winter" they are hopeful. This makes it difficult to gauge the change of perception of the general public, which would be interesting to have to predict the price. Conversely, most of the negative views on twitter come from news source we could find outside of twitter and sometimes the data would even be of better quality from another medium. Take Whale_alert for example. From another project on cryptocurrency price prediction, we concluded one of the best feature to predict prices was order book imbalance. Whales play a big role in these order book imbalance, but getting the information on twitter simply makes it second-grade in term of usefulness. However, we still think NLP is a great technology and could have interesting and profitable applications to Finance. Whether you consider the ability to automate and speed up research about a asset or simply to gather amounts of data not digestible for a human, it is a great tool.

References

Aharon, D. Y. and Qadan, M. (2019). Bitcoin and the day-of-the-week effect. *Finance Research Letters*, 31.

Araci, D. (2019). Finbert: Financial sentiment analysis with pre-trained language models.

Azari, A. (2019). Bitcoin price prediction: An arima approach.

Baker, M. and Wurgler, J. (2007). Investor sentiment in the stock market. *Journal of Economic Perspectives*, 21(2):129–151.

Bollen, J., Mao, H., and Zeng, X. (2011). Twitter mood predicts the stock market. *Journal of Computational Science*, 2(1):1–8.

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):28–43.

Critien, J. V., Gatt, A., and Ellul, J. (2022). Bitcoin price change and trend prediction through twitter sentiment and data volume. *Financial Innovation*, 8(1):1–20.

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

Fazlija, B. and Harder, P. (2022). Using financial news sentiment for stock price direction prediction. *Mathematics*, 10(13).

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

Hua, Y. (2020). Bitcoin price prediction using ARIMA and LSTM. *E3S Web of Conferences*, 218:01050.

Hutto, C. and Gilbert, E. (2014). VADER: A parsimonious rule-based model for sentiment analysis of social media text. *Proceedings of the International AAAI Conference on Web and Social Media*, 8(1):216–225.

Jagini, A., Mahajan, K., Aluvathingal, N., Mohan, V., and TR, P. (2023). Twitter sentiment analysis for bitcoin price prediction. In *2023 3rd International Conference on Smart Data Intelligence (ICSMDI)*. IEEE.

Kim, G., Kim, M., Kim, B., and Lim, H. (2023). CBITS: Crypto BERT incorporated trading system. *IEEE Access*, 11:6912–6921.

Liapis, C. M., Karanikola, A., and Kotsiantis, S. (2023). Investigating deep stock market forecasting with sentiment analysis. *Entropy*, 25(2):219.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. (2020). Ro{bert}a: A robustly optimized {bert} pretraining approach.

Mcnally, S., Roche, J., and Caton, S. (2018). Predicting the price of bitcoin using machine learning. *2018 26th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP)*, pages 339–343.

Mittal, A. (2011). Stock prediction using twitter sentiment analysis.

Nakamoto, S. (2009). Bitcoin: A peer-to-peer electronic cash system.

Nazemi, A., Jakubik, J., Geyer-Schulz, A., and Fabozzi, F. J. (2020). Incorporating financial news for forecasting bitcoin prices based on long short-term memory networks. *SSRN Electronic Journal.*

Nguyen, D. Q., Vu, T., and Nguyen, A. T. (2020). BERTweet: A pre-trained language model for English Tweets. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 9–14.

Pagariya, P., Shinde, S., Shivpure, R., Patil, S., and Jarali, A. (2022). Cryptocurrency analysis and forecasting. In *2022 2nd Asian Conference on Innovation in Technology (ASIANCON)*, pages 1–6.

Preisler, B., Mieskes, M., and Becker, C. (2019). Bitcoin value and sentiment expressed in tweets. In *Swiss Text Analytics Conference.*

Rao, T. and Srivastava, S. (2012). Analyzing stock market movements using twitter sentiment analysis. pages 119–123.

Sattarov, O., Jeon, H. S., Oh, R., and Lee, J. D. (2020). Forecasting bitcoin price fluctuation by twitter sentiment analysis. In *2020 International Conference on Information Science and Communications Technologies (ICISCT)*. IEEE.

Schilling, L. and Uhlig, H. (2019). Some simple bitcoin economics. *Journal of Monetary Economics*, 106:16–26.

Serafini, G., Yi, P., Zhang, Q., Brambilla, M., Wang, J., Hu, Y., and Li, B. (2020). Sentiment-driven price prediction of the bitcoin based on statistical and deep learning approaches. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8.

Sezer, O. B., Gudelek, M. U., and Ozbayoglu, A. M. (2020). Financial time series forecasting with deep learning : A systematic literature review: 2005–2019. *Applied Soft Computing*, 90:106181.

Shintate, T. and Pichl, L. (2019). Trend prediction classification for high frequency bitcoin time series with deep learning. *Journal of Risk and Financial Management*, 12(1):17.

Tandon, S., Tripathi, S., Saraswat, P., and Dabas, C. (2019). Bitcoin price forecasting using lstm and 10-fold cross validation. pages 323–328.

Wen, C., Li, T., and Qiu, Z. (2022). Bitcoin price prediction using autoregressive integrated moving average (arima) model. *Proceedings of the 4th International Conference on Advanced Information Science and System.*

Wikipedia contributors (2022). Wold's theorem — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/w/index.php?title=Wold%27s_theorem&oldid=1069896324`.