Deloitte.

Master Thesis

The influence of Twitter sentiment on Bitcoin prices

Author: Sebastiaan Peek (2653740)

1st supervisor:	Eduard Belitser
daily supervisor:	Emma Machielse
2nd reader:	Joost Berkhout

September 4, 2022

Abstract

Goal. The main goal of this research is to analyze the impact of Twitter sentiment on predicting Bitcoin prices. Before reaching this goal, an intermediary goal is set to compute a sentiment model which is capable of classifying the three classes negative, neutral, and positive.

Methods. Since this research is divided into two parts, there are two different models used to reach the set goals.

The first part is using a Transformer based Language Model called BERTweet to classify the sentiment of Bitcoin related tweets. The Twitter data was gathered from November 2018 until December 2019 (period prior to the COVID-19 pandemic) using an API. After analyzing the data it was clear to see that there were quite some tweets which did not portrayed any clear positive or negative sentiment, therefore it was decided to add a third sentiment class: *neutral*. Since the model now needed to predict three classes (negative, neutral and positive) and BERTweet is only able to predict two, the model had to be fine-tuned. To accomplish this, an annotated dataset containing tweets labeled as negative, neutral and positive was selected and used to fine-tune the BERTweet model. The evaluation metric used for the sentiment model was the F1 score, which was due to the fact that the dataset had a class imbalance and the F1 score tends to work better when there is a class imbalance.

In the second part the labeled tweets were aggregated on a daily and hourly basis. The dataset per hour was used to train the LSTM forecasting model. Because the Twitter data is per hour, it can be viewed as a time series, thus the train, validation and test splits were not random but they were determined at specific points in time. This way of splitting the data is necessary, since the order of instances should be kept, as the order is a dependent factor. To evaluate the model the Root Mean Squared Error (RMSE) is used, considering it punishes the error more and makes the outcome more understandable as it

is based on real values and not percentages/ratios. A baseline model is also computed in order to compare the model's performance with the other models where the Twitter sentiment is as added as a predictor. The baseline model only used Bitcoin information, which was the price and the trading volume of the past hour.

Results. The results of the sentiment model have shown that on average the majority of tweets are neutral followed by positive and negative. However, sometimes the probabilities of the classes per tweets had small differences. For example, a tweet gets labeled based on the class with the highest probability, if the second highest class has just a slightly lower probability the subtlety of that difference gets lost. This often happens when there has to be made a distinction between positive/negative and neutral, which might impact the number of negative, neutral, and positive tweets. For the LSTM model the results were a bit surprising since the baseline model performed best. Predicting prices when using the baseline model with a time window of 3 days resulted in a RMSE of 193.89. The next best scoring model was the total population with a RMSE of 301.48 which is quite much higher. Yet, the model which used Twitter sentiment as a predictor were able to somewhat predict the trend of the Bitcoin price. Nonetheless, when compared to the baseline model they were off by quite a large margin.

Conclusions. Based on the results of the first model it seems that the sentiment model was able to predict the three sentiment classes quite well. Nevertheless, for the LSTM fore-casting model the adding of sentiment information from tweets did not deliver a better performing model compared to the baseline model. Yet, the baseline model only performed better when there is a time window of 3 days. When a longer period of time is chosen the performance for the baseline model drops quite a bit.

Contents

1

Introduction

Since the launch of Bitcoin, which was introduced by Nakamoto [2008], a lot more cryptocurrencies became available. This created a whole new financial market which nowadays is mostly used as a way of investing money instead of a payment method. Especially since the fact that it is easy to buy cryptocurrencies. Everyone can open an online wallet and start investing in cryptocurrencies. However, the cryptocurrency market is very volatile, with big returns, but also big losses. Although, there are moments where stocks can rise or fall in a short amount of time, it is not even near the price fluctuations the cryptocurrency market has. Take for example Terra (LUNA) coin, in the year 2021 it had a increase of almost 14,000 percent NDTV [2022]. This quickly changed in May, when the coin crashed and became practically worthless. On the 1st of May the coin was worth around $80 and two weeks later its value was $0.0001. This shows that the cryptocurrency market is very volatile. This volatility makes it hard to predict what the price of a cryptocurrency will be. Due to the unpredictability of cryptocurrencies, investing in them would be similar to gambling. But is it possible to get the odds in our favor?

Cryptocurrencies has shown to be quite susceptible to news messages and posts on social media Garcia *et al.* [2014]. Even certain individuals (e.g., Elon Musk) have shown to greatly influence the price of certain cryptocurrencies. This might be due to the amount of followers those users have and the impact they have when the tweet something. Their tweets reach a lot of people and therefore might influence their choices when investing in cryptocurrencies. Also, it has been shown that social media has a quicker "wear-in" time than than conventional online media, which means that the information from social media has a quicker predictable value on the prices Luo *et al.* [2013]; Xie *et al.* [2017].

1. INTRODUCTION

So instead of the conventional news platforms, investors tend to move towards social media for information regarding cryptocurrencies. Twitter in particular is useful for its large-scaled feeds of tweets and therefore lots of information. Because Twitter allows users to express themselves freely, tweets can contain a variety of emotions that can be extracted Balabantaray *et al.* [2012]; Sailunaz & Alhajj [2019]. Also, next to emotions also sentiment can be extracted, which is more a belief emerged from emotions, such sentiment can be shared among certain different groups of people. So, by extracting sentiment it is possible to get a view on how groups of people think about different subjects Pak & Paroubek [2010]. Even though every tweet has a maximum of 140 characters, when you take the aggregated sentiment of millions of tweets it may provide an accurate representation of the sentiment and mood at any given time. It has been shown that certain financial decisions are driven by emotion and mood Nofsinger [2005]. In summary, social media has a quicker "wear-in" time, tweets contain useful sentimental value and it has been shown that certain financial decision are driven by mood and emotion. Together, this indicates that it is reasonable to research to see if public mood and sentiment from twitter can drive stock market and cryptocurrencies as much as conventional news platforms.

As Kahneman & Tversky [2013] states, emotions play a significant role when making certain financial decisions. This makes it quite probable that sentiment gathered from tweets can give interesting insights into human decision-making. As a result, the following situation may arise: someone reads multiple positive tweets about a crypto coin and therefore might be more tempted to invest in that particular coin as opposed to reading negative tweets. It already has been shown that Twitter sentiment can be used to predict changes in the stock market prices Bollen *et al.* [2011]; Li *et al.* [2018]. As a result, it will be interesting to see if this is also works with cryptocurrencies, particularly Bitcoin.

Most papers which performed sentiment analysis on Twitter messages extract sentiment using a rule- and lexicon-based model (e.g., VADER, which stands for *Valence Aware Dictionary for Sentiment Reasoning* by Hutto & Gilbert [2014]). Over the past years more state-of-the-art model are available to use for classifying texts. One of those models is BERT Devlin *et al.* [2018], which has shown very promising results. For this research the pre-trained model **BERTweet**, which specifically trained on English tweets, is used. This model has the same architecture as $\text{BERT}_{\text{base}}$ and used the same pre-training procedure from RoBERTa Liu *et al.* [2019]. This model showed very promising result in three Tweet NLP tasks: Part-of-speech tagging, Named-entity recognition and text classification

Nguyen *et al.* [2020]. Next, in order to predict bitcoin prices using sentiment information from tweets, a model that can deal with multiple variables should be used. Since predicting bitcoin prices is done over time, some assumptions have to be made about stationarity of the data. To overcome making those assumptions an LSTM model will be used as a forecasting model to predict bitcoin prices Hochreiter & Schmidhuber [1997]; Wu *et al.* [2018].

This will give the following research question: *To what extend is it possible to predict the price of Bitcoin using Twitter sentiment?* There is a small extension made to this question based on the possible influence of twitter users who have a lot of followers, which is: *What is the impact of only using sentiment information of the top 20 percent of users based on their number of followers?*

2

Literature Review

2.1 Predictability of cryptocurrencies

Stock market prediction is a topic that both researchers as well as real life business are attracted to. In the early stages, research on the predictability of the stock market was based on the Efficient Market Hypothesis (EMH) and the 'random walk' theory. According to the EMH, all available information reflects the current stock price Fama [1991]. This implies that information from the past and present is immediately incorporated into stock prices, thus prices changes are merely due to 'new' information or news. Since the fact that, in nature, news is unpredictable, the stock market will follow a random walk pattern. Therefore, stock prices cannot be predicted with more than 50% accuracy if a binomial model is used Qian & Rasheed [2007]. If this hypothesis is valid, it would mean that attempting to predict the stock market would be pointless. However, some research has shown that the stock market prices does not always follow a random walk pattern, since they managed to predict stock prices better than random when using sentiment information from Twitter messages Bollen *et al.* [2011]; Vu *et al.* [2012].

When looking at cryptocurrencies it seems that most assumptions are not applicable. One of those assumptions stated by EMH is that, in an efficient market, successive price changes are independent Fama [1970]. However, as Ciaian *et al.* [2018] shows, the Bitcoin and altcoin markets are highly interdependent and that the Bitcoin-altcoin price relationship is very strong in the short run rather than the long run, which contradicts the assumption of independency of price changes. Next, another assumption which is stated by the EMH is that investors are assumed to be rational and value an asset based on it fundamental value. In an online article about Bitcoin by Silverman *et al.* [2017], William Goetzmann, a Yale economist states the following: "Bitcoin is an electronic commodity

used for payment that has become a vehicle for speculation. It does not have future dividends. Its value is set by expectations of future demand and/or expectations of future resale value. There is no way to value it fundamentally." This would mean that the prices of cryptocurrencies are driven by speculation, making the market irrational.

The EMH is a theory which exists for quite some time and might not be an appropriate approach for something as new like Bitcoin, due to the more emotionally driven investments when looking at cryptocurrencies. Therefore, the Adaptive Markets Hypothesis (AMH), proposed by Lo [2004], seems to be a more applicable theory when looking at cryptocurrencies. Lo [2012] states that the EMH is not wrong, but it is merely incomplete. The AMH states that "markets are not always efficient, but are usually competitive and adaptive, varying in their degree of efficiency as the environment and investor population change over time" Lo [2012]. Whereas the EMH assumes homo economicus[1] to be a consistent rational actor, the AMH claims that this only occurs when there is certainty. The actor's behavior during moments of uncertainty is difficult to describe because it is motivated by emotion and instinct. This is best exemplified by the flight-to-safety principle, which is one of the fundamental implications of the AMH and can also be seen in the cryptocurrency market. Investors will shift their risky assets into defensive assets during periods of market instability when (extreme) greed and/or fear rule through the so-called "madness of the masses." This also happens for Bitcoin as mentioned by BaurBaur & Hoang [2021]. When the market is volatile, investors will frequently use stablecoins like Tether, Paxos, or USD Coin as a safe haven. When the volatility decreases, the market returns to the wisdom of crowds, and prices become a more accurate reflection of the available information. Furthermore, Lo [2012] contends that "a relatively new market is likely to be less efficient than a market that has been in existence for decades". This gives support to the proposed argument that the cryptocurrency market is inefficient and thus, to some extent, predictable.

2.2 Sentiment analysis

Sentiment analysis, which sometimes is also called *opinion mining*, is a research area within natural language processing (NLP). In this field of study, people's opinions, emotions and attitudes towards products, services, organizations or individuals are analyzed. Analyzing

[1] **Homo Economicus** is a theoretical abstraction that some economists use to describe a rational human being. In certain neoclassical economic theories, people are portrayed this way: as ideal decision-makers with complete rationality, perfect access to information, and consistent, self-interested goals. Source: `https://www.investopedia.com/terms/h/homoeconomicus.asp`

opinions and sentiment is not new, often people analyse the opinions of others about the decisions they might want to make. For example, consumers often read reviews of products to determine if they want to buy that same product.

In the past years social media had an enormous growth, which makes the use of sentiment analysis more relevant. Organizations can use sentiment analysis to get a grasp of the public opinion about their business. Since there is an increased volume of opinions in blogs, columns, forum posting and posts on social media e.g., Facebook or Twitter, it is hard for people to read all of it, for example on a daily basis there are already over 500 million tweets. Using automated sentiment analysis systems it is possible to extract and summarize these opinions. Another thing sentiment analysis is used for, is predicting the stock prices. In particular the movement of the stock prices, so if a stock is in an upward/downward trend Nguyen *et al.* [2015].

2.2.1 Twitter sentiment analysis

Within the past research based on Twitter sentiment analysis (TSA) there are multiple approaches which are used. A survey performed by Giachanou & Crestani [2016] stated that there are four different classes which are identified by the literature regarding TSA:

- Machine Learning
- Lexicon-Based
- Hybrid (Machine Learning & Lexicon-Based)
- Graph-Based

The machine learning method will train a classifier that is able to detect the sentiment of a tweet based on certain features, while a lexicon-based approach is more manually. Those approaches often use a dictionary of words which are either positive or negative, which are then used to derive the polarity of the message, in this case a tweet. It is possible to combine both lexicon-based methods with machine learning to obtain better performance as shown by Khuc *et al.* [2012].

It has been shown that Twitter messages contain rich information that can affect certain financial markets Bollen *et al.* [2011]. Literature has shown that there is some effectiveness in using TSA when predicting the financial markets. One of the more well-known studies who had some reasonable results is by Bollen *et al.* [2011]. They obtained an accuracy of 86.7% in predicting the daily up and down changes in the closing values of the Dow Jones

Industrial Average (DJIA). It must be said that the outcome is revisited by Lachanski & Pav [2017]. They found that there were some methodology problems; they discovered that several researchers were not able to replicate their accuracy score, raising some serious concerns about the validity of the authors' results Kuleshov [2011]. More recently Li *et al.* [2018] also did research regarding the influence of sentiment analysis on the financial markets. They used a Naive Bayes classifier with regression models, which obtained an accuracy of 86.3%. They have shown that when the price on a day was very volatile the day after the volume of twitter messages increases. This might be an indication that Twitter sentiment can both be an *effect* and a *cause* of the financial markets.

When using the results of the TSA to predict financial markets multiple approaches are used. Most of the time a regression model is used or a statistical causality test Bollen *et al.* [2011]; Li *et al.* [2018]; Porshnev *et al.* [2013]. The differences between the research lies at the sentiment analysis methods, which vary from supervised machine learning models to lexicon-based approaches using annotated corpora. When a statistical causality test is used to see if there is a relation between Twitter sentiment and financial markets, most literature opt for the Granger-causality test introduced by Granger [1969]. However, there are some limitations to this test regarding its bias and assumptions. One of the more impactful limitations of the test is that the data has to be stationary and it requires a linear relation between the variables. Since there are many different factors which can affect the prices of stocks and cryptocurrencies, the relations between a variable and the prices of stocks and cryptocurrencies is likely to be non-linear Bollen *et al.* [2011].

2.3 Twitter sentiment analysis and Bitcoin

In the past years social media has become a well used source of information on cryptocurrencies. Multiple researchers successfully have shown the impact of social media on the price of Bitcoin Kim *et al.* [2017]; Mai *et al.* [2015]. They also performed sentiment analysis to see if it has an effect on the price. Most researchers opt for a lexicon-based model Karalevicius *et al.* [2018]; Kraaijeveld & De Smedt [2020] or use a finance sentiment dictionary Loughran & McDonald [2014] to determine the sentiment scores of the messages. In the past years more State-Of-The-Art (SOTA) Language Models are introduced, being able to outperform a lot of older models. Those models often take a more machine learning approach instead of a rule based approach. To see the difference in performance this research will therefore focus more on the machine learning approach. In previous research the (granger) causality was determined between the gathered sentiment scores

over time and stock prices Bollen *et al.* [2011] and/or cryptocurrency prices Balcilar *et al.* [2017]. In these papers the calculated sentiment was a polarity score which is aggregated per hour of day and would give a representation on how negative/positive that particular hour/day was.

In this research the sentiment model will be used as a classification model and thus will not have a polarity score, thus tweets would be either classified as negative or positive. After reading some tweets, it became clear that some tweets do not portray any positive or negative sentiment and thus can be considered neutral. As a result, when classifying tweets, the neutral class is added to the possible sentiment classes. This means that there are three variables for each hour, namely the number of negative, neutral, and positive tweets, which would make it a multivariate time series analysis and therefore the multivariate granger causality test should be used. This causality test is generally used for medical data since it is often multivariate data Wen *et al.* [2013]. Nonetheless, when conducting such statistical tests certain assumption have to be made. When using a machine learning approach less or even no assumptions are needed to be satisfied. So the model used in this research to forecast the Bitcoin prices is an LSTM model, which have shown to be quite effective when used for time series data regarding financial data Cao *et al.* [2019]; Siami-Namini & Namin [2018]; Siami-Namini *et al.* [2019].

3

Methodology

3.1 Data Collection

In this research the focus lies on Bitcoin. Therefore only tweets which contain the (hash-tags)keywords (#)bitcoin or (#)btc are gathered and all non-English tweets are filtered out. The tweets are gathered from November 2018 till December 2019. The reason for this time period is to make it more comparable with the work of other researchers. Besides that, during the covid-19 pandemic the prices of cryptocurrencies were fluctuating quite much, which makes it harder to predict the prices. All tweets within this time frame containing the earlier stated keywords are included. The number of tweets per day are shown in Figure 3.2.

3.1.1 Tweets

For the gathering of tweets the official Twitter API is used. The Twitter API enables programmatic access to all public information found on Twitter. With the API it is possible to get access to meta information of users and posts. Twitter has different access levels for their API, most of them are only available using a paid plan. For this research, an application for the Academic Research level was filed, which is free if you comply to the requirements, and approved shortly after applying.

The Academic research API allows for gathering 10 million tweets per month and the full-archive of twitter messages is accessible. This means it even has the first tweet ever tweeted available. This created the opportunity to gather tweets before the start of Covid-19.

An annotated dataset of tweets including sentiment information is used to train the sentiment model to classify the gathered tweets. Instead of the two classes, positive and

negative, neutral is also taken into account to classify tweets, slimming down the choice of the number of datasets containing annotated tweets. After some research a dataset which was used during the International Workshop on Semantic Evaluation (SemEval) was selected. During this event in 2017 different NLP related tasks had to be performed, one of them was *Task 4*: sentiment analysis of Twitter messages Rosenthal *et al.* [2017]. This dataset contains around 50k tweets with labels 0, 1 and 2 meaning negative, neutral and positive respectively. The count of each class is shown as a histogram in Figure 3.1. As one can see, the number of negative tweets is much lower than neutral and positive. This class imbalance within the dataset might cause for some issues when training the classification model. The annotated dataset is necessary to have since the training of the sentiment model is a supervised learning process.

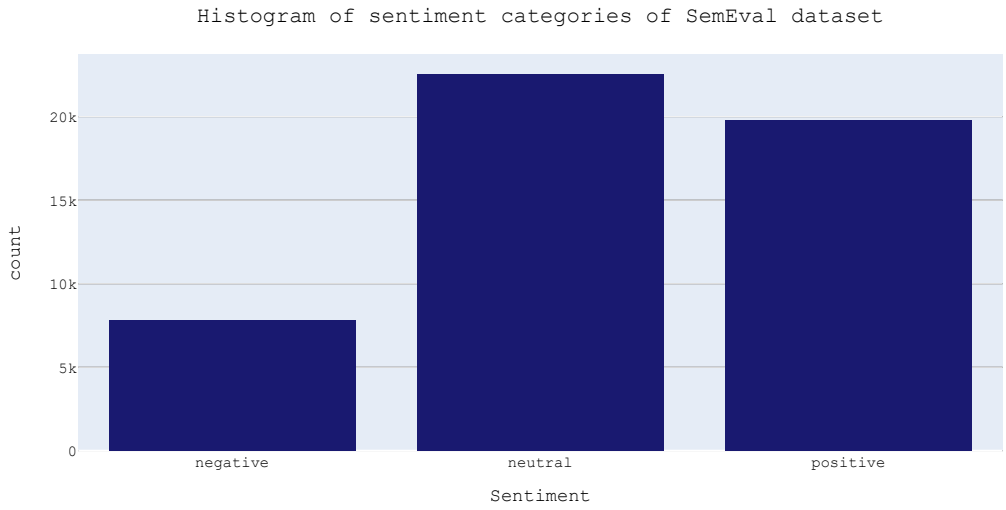

Figure 3.1: Histogram of the three categories within the SemEval dataset.

3.1.2 Bitcoin data

To gather the Bitcoin prices the API of BinanceUS was used, since this was one of the few API's which offered free access to hourly and daily historical OHLC (Open, High, Low, Close) price information. CoinMarketCap would be another good source for price information since it combines prices from a large number of exchanges. The combination of multiple sources of price information creates a more accurate representation of the general value of cryptocurrencies. However, to get historical data from CoinMarketCap, a paid plan was needed and therefore could not be used. Next to the OHLC data the number of trades are gathered. To create a forecasting model, the Bitcoin data has to be converted

into a time series. The conventional stock market has specific open en close times, whereas the cryptomarket is always open. Therefore, for the Bitcoin data, the close price per hour or day is used to be able to distinguish the hours and days from each other. The closing price of Bitcoin per day is shown in Figure 3.2.

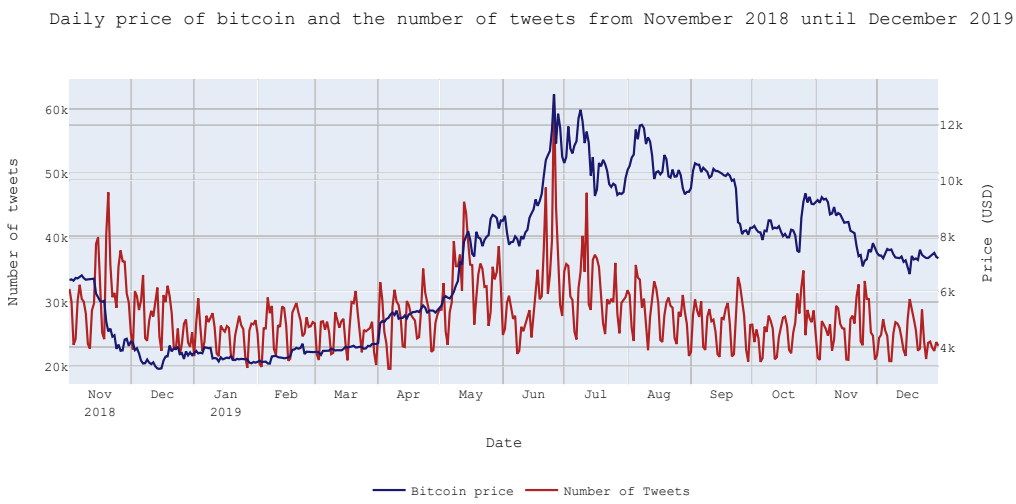

Figure 3.2: The bitcoin price and number of tweets per day from November 2018 until December 2019.

3.2 Data pre-processing

An important part of this research is pre-processing the data. Twitter messages are known for having a lack of structure, since people can write whatever they want, however they want. This creates a lot of noisy data, grammar mistakes, abbreviations, missing punctuation's etc. The authors who proposed the BERTweet model performed two strategies to normalise the tweets. A 'soft' strategy is used which replaces user mentions and web/url links into special token *@USER* and *HTTPURL*, respectively. Also emoticons are converted into strings corresponding to the emoticon, for example 🚀 will be converted to *:rocket:*. They also used a 'hard' strategy which further applying lexical normalization dictionaries to normalize word tokens in tweets. However, they showed that the 'soft' strategy outperformed the 'hard' strategy when performing several NLP tasks. Therefore it was decided that for this research the 'soft' strategy will be used to pre-process the text of the tweets.

3.3 Models

The process of modelling is split up into two parts. The first part a sentiment model is used to classify each tweet as being negative, neutral or positive. For the sentiment analysis a machine learning approach is used instead of lexicon-based & rule-based (e.g. VADER). The sentiment scores are obtained using a Transformer based model called BERTweet Nguyen *et al.* [2020]. This model has the same architecture as $\text{BERT}_{\text{base}}$, which is trained with a masked language modeling objective Devlin *et al.* [2018]. The pre-training procedure of BERTweet is based on the RoBERTa Liu *et al.* [2019], which optimizes the BERT pre-training approach for more robust performance. The BERTweet model is trained on a large corpus which consists of 850M English tweets. The model also uses a tokenizer which performs the 'soft' normalization and pre-processing as stated in section 3.2. It is important to note that BERTweet performs binary classifications when conducting sentiment analysis, resulting in tweets that are only classified as positive or negative, implying that a tweet will not be classified as neutral. Since this research also will take neutral labels into account, the pre-trained model has to be fine-tuned to be able to classify three classes, negative, neutral and positive.

After the sentiment scores are obtained the results are aggregated on a hourly and daily basis. The daily information will be used to get an overview of the sentiment on each day. The hourly information will be used to predict the bitcoin price. The number of negative, neutral and positive tweets will be used as a predictor in the LSTM forecasting model.

3.4 Predicting sentiment model

As stated earlier, the model to gather the sentiment scores of the tweets is called BERTweet. This is a model based on the architecture BERT, which is a transformer model. The special thing about BERT is that it works bidirectional, therefore a BERT model can also be called a *multi-layer bidirectional transformer* model Devlin *et al.* [2018]. So, by taking an already successful model and training it specifically on tweets, it is able to handle and classify shorter texts. Because short texts contain less information, classifying them can be difficult.

However, the pre-trained model BERTweet is only able to classify tweets as positive or negative. Therefore, the model needs to be fine-tuned to be able to properly classify the classes: negative, neutral and positive.

3.4.1 BERT and BERTweet

The architecture of BERTweet is based on BERT, which in itself is based on Transformers. The transformer architecture was released in a Google machine translation paper titled "Attention Is All You Need" in December 2017 Vaswani *et al.* [2017]. This work sought models capable of automatically translating multilingual texts. Prior to this, many machine translation techniques used some automation, but it was backed up by extensive rules and linguistic structures to ensure the translations were acceptable enough for a service like Google Translate. A year later BERT, which stands for Bidirectional Encoder Representation from Transformers, was released in the paper "Pre-Training of Deep Bidirectional Transformer for Language Understanding" Devlin *et al.* [2018]. In previous language models text was processed sequentially in a loop-like approach, an example of this process is shown in Figure 3.3.

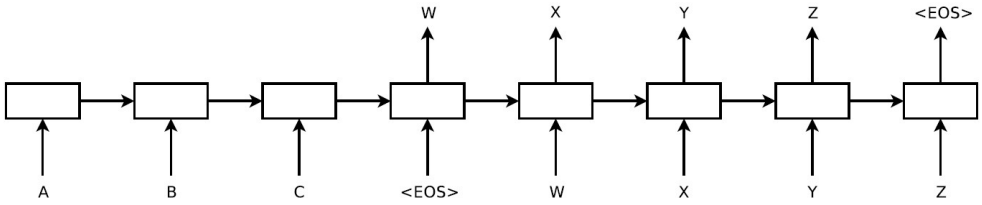

Figure 3.3: This is an simple representation of a sequence to sequence process where the input sentence "ABC" is read by the model which produces "WXYZ" as an output sentence. The model will stop reading inputs and making predictions after the end-of-sentence (<EOS>) token.

To further elaborate on this process lets take the following sentence, "The game crashed when the user started it to play with his friends". This is a simple sentence for a person to understand, but there are a few complications if you process this sequentially. How do you know what "it" refers to once you get there? To identify the "game" as the main protagonist in this sentence, it may be necessary to save some hidden state which contains this information. Then, as you read the sentence, you would have to figure out how to connect the "it" to the "game". Now, if a sentence contains a large number of words, it follows that as more text is processed, there will be more references to keep track of. However, this was a problem for sequential models since they could only prioritize the importance of words that were most recently processed. This problem is also known as the "Vanishing Gradient" problem, special neural networks as Long Short-Term Memory (LSTMs) are known to alleviate the consequences of this phenomenon. Later in section 3.5.2 this model will be discussed more elaborate. Nonetheless, LSTM models were not

efficient enough to find a way to "focus" on the important word in each sentence. This is the problem that the Transformer network addressed by employing the mechanism known as "Attention". The architecture of the transformer is shown in Figure 3.4. As seen in the figure the architecture contains two main parts:

1. **Encoder**: This component of the Transformer model processes the input text, looks for important parts, and generates an embedding for each word based on its relevance to other words in the sentence.

2. **Decoder**: This takes the encoder's output, which is an embedding, and converts it back to text output, i.e. the answer based on the question input text.

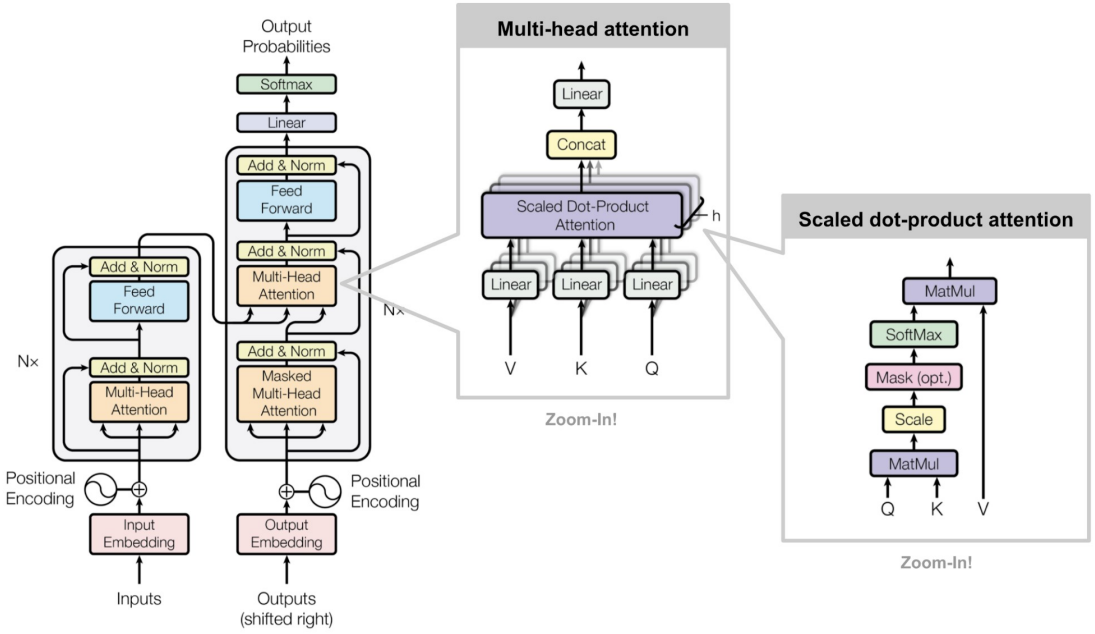

Figure 3.4: The full model architecture of the transformer (left is the encorder, right is the decoder), including zoomed-in figures to visualise the attention process within the Multi-Head Attention block. Vaswani *et al.* [2017]

It is important to note that only the Transformer's encoder mechanism is required for BERT to create a language model. A more detailed explanation of how Transformers work can be found in Google's paper "Attention is all you need" Vaswani *et al.* [2017].

The Transformer encoder reads the entire sequence of words at once, as opposed to directional models, which read the text input sequentially (left-to-right or right-to-left). As a result, it is considered bidirectional, though it is more accurate to say non-directional. This feature enables the model to learn the context of a word based on its surroundings (left and right of the word).

Language models' ability to learn context is inherently constrained because they frequently attempt to predict the next word in a sentence. BERT employs the following two training strategies to overcome this challenge:

Masked LM (MLM) : Before the input of words are fed into BERT, roughly 15% of the words in each sequence are replaced with a [MASK] token. When training the model then attempts to predict the original value of the masked words. This is based on the context provided by the other surrounding, non-masked, words in the sequence. In technical terms, the prediction of the output words requires the following steps:

1. A classification layer is added on top of the encoder output.
2. The output vectors are then multiplied using the embedding matrix, which transforms them into the vocabulary dimension.
3. For each word in the vocabulary the probability is calculated with the softmax function.

This process can been seen in Figure 3.5. The implementation of masking the words is a bit more elaborate than described earlier. For additional information about this process, see Appendix A.

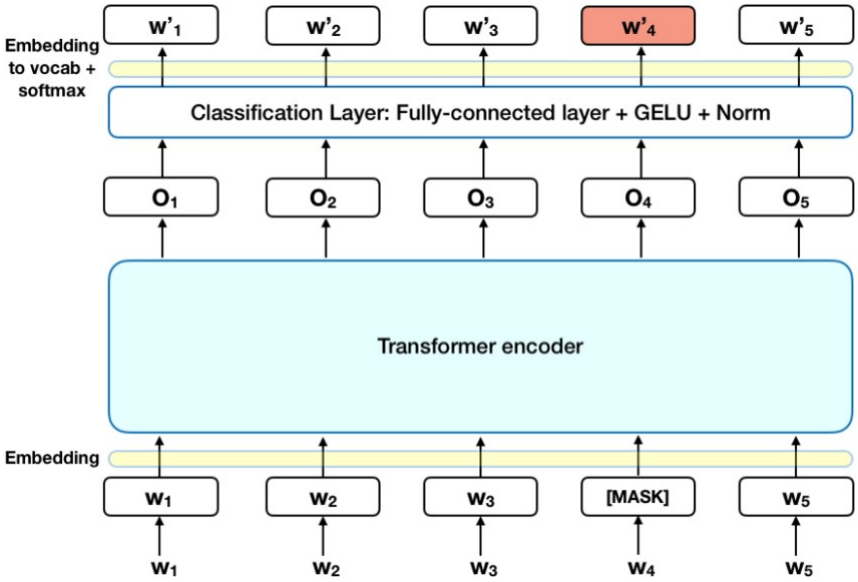

Figure 3.5: The process of predicting the word of the masked value with the use of the Transformer encoder.Horev [2018]

Next Sentence Prediction (NSP) : During the BERT training process, the model receives pairs of sentences as input and learns to predict whether the second sentence in the pair is the subsequent sentence in the original document. During training, 50% of the inputs are pairs in which the second sentence is the following sentence in the original text, while the other 50% are random sentences from the corpus. The random sentence is assumed to be isolated from the first sentence.

In Figure 3.6 a representation of two sentences as an input of BERT model is shown, the distinguishing of the two sentences is performed in the following way:

1. A [CLS] token is placed at the beginning of the first sentence, and a [SEP] token is placed at the end of each sentence.

2. Each token has a segment embedding indicating Segment A or Segment B. Segment embeddings are conceptually similar to token embeddings with a vocabulary of two.

3. Each token receives a positional embedding to indicate its position in the sequence. The Transformer paper describes the positional embedding concept and implementation.

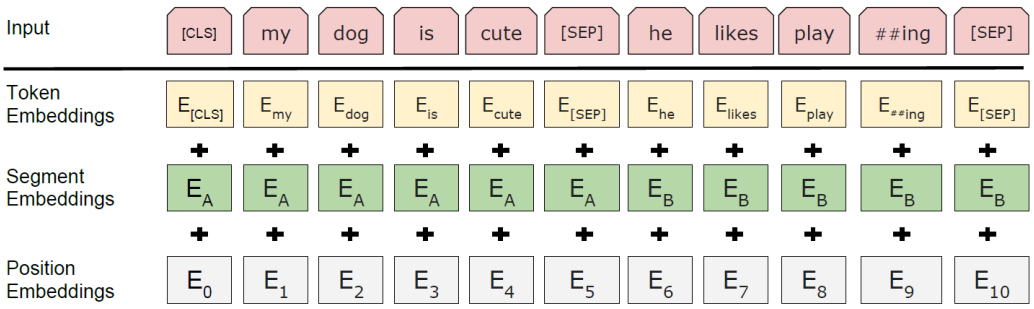

Figure 3.6: BERT input representation including [CLS] and [SEP] tokens. The input embeddings are the sum of the token embeddings, the segmentation embeddings and the position embeddings.Devlin *et al.* [2018]

The following steps are taken to predict wheter or not the second sentence is related to the first:

1. The Transformer model processes the entire input sequence.
2. Using a simple classification layer, the output of the [CLS] token is transformed into a 2x1 shaped vector.
3. Using a softmax function to calculate the probability of IsNextSequence.

Masked LM and Next Sentence Prediction are trained together in the BERT model with the goal of minimizing the combined loss function of the two strategies.

BERTweet BERT has shown promising results when used on different corpora, however tweets are often small messages which do not always contain multiple sentences. This might lead to challenges when applying existing language model, which are pre-trained on large corpora with formal grammar and regular vocabulary, on a large-scale corpus of English tweets. So, in order to fill this gap BERT was pre-trained on a large corpus of 850M tweets. During training BERTweet three NLP tasks where conducted and evaluated, one of which was text classification. The results of training procedure have shown that they outperformed other language models which performed the same tasks e.g., RoBERTa$_{\text{base}}$Liu *et al.* [2019] and XLM-R$_{\text{base}}$Conneau *et al.* [2019].

3.4.2 Model setup

Often when performing sentiment analysis, models will only perform binary classification, which means that a text will be classified as either negative or positive. However, when reading the tweets in the gathered dataset it was clear that some of them did not have any clear negative or positive sentiment and thus could be seen as neutral tweets. But, BERTweet is only able to perform binary sentiment classification and therefore needs to be trained to predict three classes namely: *negative*, *neutral* and *positive*.

To classify tweets into these three classes, a classification layer is added which is able to predict three different classes. Since this would be a newly introduced layer it is necessary to train this layer as it has never seen any data. Typically the BERTweet model outputs two variables, the first variable contains the embedding vectors of all tokens in a sequence, the second variable (often called the pooled output) contains embedding vectors of [CLS] tokens. It is sufficient to use these embedding vectors of [CLS] tokens as an input to the classification layer for a text classification task.

To improve the performance of the classification model, an additional hidden layer is added just before the classification layer. As a result, a two-layer neural network is added to the pre-trained BERTweet model. After adding those layers, they both are initialized using Xavier initialization Glorot & Bengio [2010]. A loss function is also added, and because this is a multi-classification problem, the cross entropy loss function is used. Finally, a dropout layer is added to the model to reduce the possibility of overfitting.

For the optimization operator the AdamW optimizer is chosen, this is a variant of the Adam (Adaptive Moment Estimation) optimizer but with an improved implementation of weight decay. Next, instead of a given static learning rate, a scheduler is implemented, which in this model is a cosine schedule with warmup. This produces a schedule with a decreasing learning rate based on the values of the cosine function between the initial learning rate in the optimizer and 0. A warmup period is used before the schedule, during which the learning rate increases linearly between 0 and the given learning rate. An example how such a schedule would look like is shown in Figure 3.7.

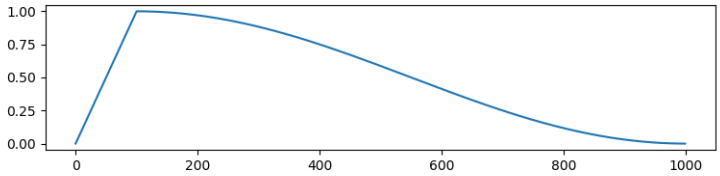

Figure 3.7: Example of a learning rate schedule based on the values of the cosine function including a linear warmup period.Huggingface [2022]

At last, the data used to train the model is split into a training, validation and test set. The split ratio chosen for this dataset is: train - 70%, validation - 15%, test - 15%. Since the data has a class imbalance, as seen in Figure 3.1, it is important that all classes are represented in equal ratios. This is done by performing a stratified random sampling when creating the dataset splits.

BERTweet hyperparameters : Before starting immediately with training the model, certain parameters have to be set. The added hidden layer is the same size as the last layer from the pre-trained BERTweet model. The classification layer can have any number of nodes, where the number of nodes equals the number of classes, which is three in this case. When the input size of the hidden layer and the number of nodes for the classification layer are determined, the optimizer and scheduler parameters must be set. Furthermore, the data is divided into batches to optimize the training procedure. Choosing different batch sizes can affect performance, and changing the batch size may require changing the learning rate as well, because they can both impact each other when changed Kandel & Castelli [2020]. This shows that different combinations of parameters can have different results on the performance of the model. To find the optimal combination of parameters for this model a hyperparameter grid search is conducted. During training the performance of the model was determined by the loss of the validation set. The different choices for the hyperparameters were as follows:

- Batchsize: **64**, 128, 256
- Learning rate: **1e-5**, 2e-5, 5e-5, 1e-4, 2e-4, 5e-4
- warmup: 0.1, **0.2**, 0.3
- weight decay: 1e-4, 1e-3, **1e-2**

The values which are bold were the optimal values when combined and therefore used when training the final model. During the hyperparameter tuning the results have shown that training for more than 6 epochs only had negative impact on the performance, therefore the final training is run for 6 epochs and after each epoch the model is validated on

the validation set. After each epoch a checkpoint is saved and at the end of training the top 3 checkpoints, based on the validation loss, are saved.

Those three model then are evaluated on the test set and the resulting accuracy and F1 scores are calculated to see which model performs the best. The best performing model is then used to predict the sentiment scores of the tweets regarding bitcoin. When predicting the sentiment scores the probabilities of all classes are predicted. Based on the class which has the highest probability the tweet get labeled with that particular class. An example of the results is shown in Table 3.1

Table 3.1: Example of three tweets regarding bitcoin with probability per sentiment class and final label based on highest the class with the highest probability.
Labels: 0 = negative, 1 = neutral, 2 = positive.

Tweet	NEG	NEU	POS	label
Bitcoin is amazing, invest now!!	0.05	0.21	0.85	2
Never buying bitcoin again, lost everything	0.91	0.12	0.09	0
Bitcoin is now legal in 127 countries.	0.11	0.82	0.34	1

3.5 Forecasting model

3.5.1 Data preparation

To be able to use the sentiment information as a predictor for a forecasting model, the data is aggregated on a hourly and daily basis. This means that for each hour and day the number of negative, neutral and positive tweets are counted.

However, since this research also is looking into the possible influence of users which have a high follower count, this process has some extra steps to perform beforehand. Per hour/day the users which are in the top 20 percent based on their followers count will be split. After performing this processing step there are three different datasets. The first is the total population, the second is the data of the influential users based on their followers count and the last is the other 80 percent. For each of those datasets the total number of negative, neutral and positive tweets per hour/day are calculated.

At last, for each Twitter sentiment dataset the bitcoin closing price and volumes are added. For the hourly data the closing price of that particular hour is added and for the daily data the closing price of that date is added. Two examples of the daily and hourly dataset are shown in Table 3.2 and 3.3. Now that the results of the sentiment analysis is

done and combined with the bitcoin prices, it was time to start using a fore-casting model to predict the prices using the sentiment information of tweets.

Table 3.2: Example of the dataset aggregated on a daily basis including the number of occurrences of each class, tweet volume (total number of tweets on that day), the Bitcoin trading volume and closing price.

Date	NEG	NEU	POS	tweet volume	trade volume	closing price
01-01-2019	1628	14265	5689	21582	23.741.687.033	3797.14
02-01-2019	2912	17230	6623	26765	35.156.463.369	3858.56
03-01-2019	2341	19017	9192	30550	29.406.948.359	3766.78

Table 3.3: Example of the dataset aggregated on a hourly basis including the number of occurrences of each class, tweet volume (total number of tweets in that hour), the Bitcoin trading volume and closing price.

Date	NEG	NEU	POS	tweet volume	trade volume	closing price
01-01-2019 0:00	66	638	239	943	68.636.742	3700.31
01-01-2019 1:00	59	490	210	759	613.539.115	3689.69
01-01-2019 2:00	66	567	198	831	895.302.181	3690.0

Before training it was decided to use the hourly data instead of the daily data. This choice is based on the fact that the daily data contained not enough instances to make reliable predictions. Next, the hourly data has to be split into train, validation and test sets. The ratio of the train, validation and test datasets are the same as for the sentiment model: 70%, 15%, and 15% respectively. However, in contrast to the sentiment model there is no random sampling, since this is time-based data it is important to maintain the order of instances. Thus, the first 70% of the dataset will be the train dataset, the next 15% the validation dataset, and the last 15% the test dataset, which is done for the total population, influential users and non-influential users. Also, to be able to see the effect of adding sentiment information a baseline dataset is used which only contains the trading volume and closing price of bitcoin.

To make predictions based on historical information the LSTM model has to look back a certain amount of days, which can be any given amount of days. This can be expressed as a sequence of vectors $[\mathbf{x}_t, \mathbf{x}_{t-1}, \mathbf{x}_{t-2}, \ldots, \mathbf{x}_{t-i}]$ to predict y_{t+1}, where $\mathbf{x_t}$ is a vector containing the predictors at time t and i number of days. For this research three different sequence lengths are chosen to evaluate which are 3, 5, and 7 days. The reason for this is that after

a week the sentiment information would probably be not relevant anymore and thus not reliable to use a predictor.

3.5.2 LSTM

During the research project different methods were considered to predict the bitcoin prices. When performing time series forecasting multiple methods are possible. Often autoregressive models like ARIMA (AutoRegressive Integrated Moving Average) or SARIMA (Seasonal ARIMA) are used. However, such models are known to perform poorly on multivariate data, and adding Twitter sentiment as a predictor will make the data multivariate. As described in section 2.3, this research will focus on using an LSTM model to predict the Bitcoin prices when using the sentiment of Twitter messages as extra variables.

In section 3.4.1 the LSTM model is shortly mentioned as a model which works well when using sequential data. This is due to the "Vanishing Gradient" problem when previous information is a dependent variable when performing forecasts. These issues are often the case when using Recurrent Neural Networks (RNNs). When dealing with short-term dependencies, RNNs perform quitte good. For example, when applied to the following sentence: *The colour of the sky is …*, RNNs have proven to be quite effective. Since the issue has nothing to do with the context of the statement, it is only interested in finding the missing word in that particular sentence. The RNN does not need to remember what was said before this sentence or what it meant; all they need to know is that the sky is usually blue. As a result, the prediction would be: *The colour of the sky is **blue**.* However, basic RNNs are incapable of comprehending the context of an input. Something said a long time ago cannot be easily remembered when making predictions in the present. The fact that the relevant information needed for the context may be separated by a load of irrelevant data, which causes the RNNs to fail. This problem can be solved by using a slightly modified version of RNNs called Long Short-Term Memory Networks.

LSTM, like any other neural network, consists of layers that enables it to learn and recognize patterns for improved performance. The basic operation of LSTM models can be thought of as holding the necessary information and discarding the information that is not necessary or beneficial for further predictions.

An LSTM cell acts as a memory, writes, reads and erases data based on the decisions made by the input, output, and forget gates, respectively. A recurrent learning procedure is then used to train (adapt) the weights associated with the gates. An LSTM with those gates is shown in Figure 3.8.

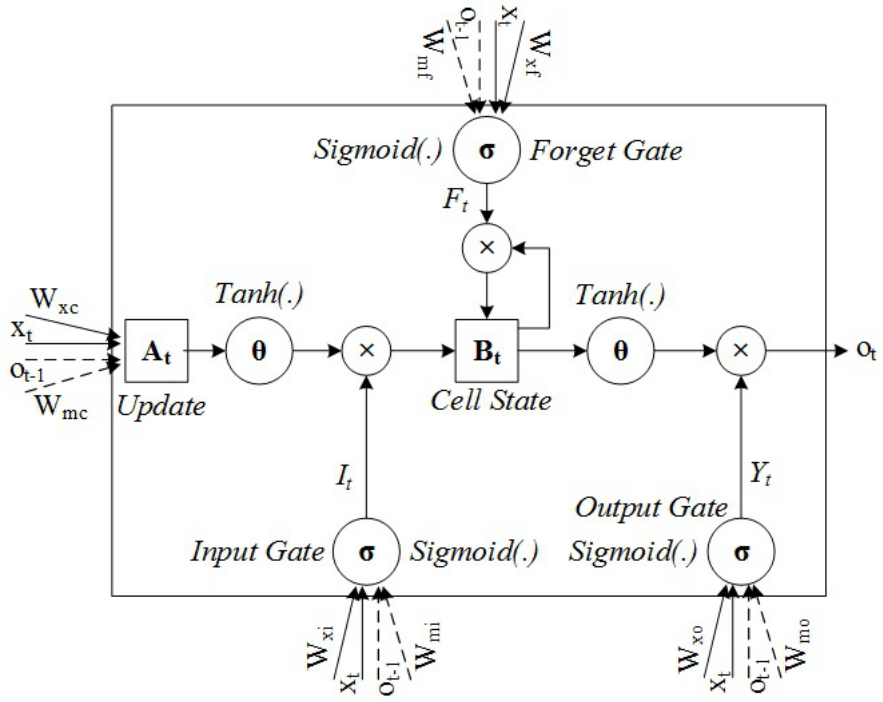

Figure 3.8: An LSTM cell containing the input gate, the forget gate, and the ouput gate. Each gate receives two vectors as input $\mathbf{x}_t$, and previous output, $\mathbf{o}_{t-1}$. Venna *et al.* [2018]

The memory cell shown in Figure 3.8 is implemented as follows:

$$\mathbf{I}_t = \sigma\left(W_{xi}\mathbf{x}_t + W_{mi}\mathbf{o}_{t-1} + \mathbf{b_i}\right)$$
$$\mathbf{F}_t = \sigma\left(W_{xf}\mathbf{x}_t + W_{mf}\mathbf{o}_{t-1} + \mathbf{b_f}\right)$$
$$\mathbf{Y}_t = \sigma\left(W_{xo}\mathbf{x}_t + W_{mo}\mathbf{o}_{t-1} + \mathbf{b_o}\right)$$
$$\mathbf{A}_t = \sigma\left(W_{xc}\mathbf{x}_t + W_{mc}\mathbf{o}_{t-1} + \mathbf{b_c}\right)$$
$$\mathbf{B}_t = \mathbf{F}_t \odot \mathbf{B}_{t-1} + \mathbf{I}_t \odot \theta\left(\mathbf{A}_t\right)$$
$$\mathbf{o}_t = \mathbf{Y}_t \odot \left(\mathbf{B}_t\right)$$

In these equations W_x and W_m are the adaptive weights, which are initialized at random in the range (0,1). The vectors $\mathbf{x}_t$ and $\mathbf{o}_{t-1}$ denote the current input and previous output, respectively. The $\mathbf{b}$ parameters are bias vectors which are not shown in Figure 3.8. The cell state, $\mathbf{B}_t$, is updated by the forget gate, the input gate, and the current input value ($\mathbf{A}_t$). The functions σ and θ detemine the *Sigmoid* and *Tanh* activations respectively Venna *et al.* [2018].

To better understand the importance of each gate and the cell state an explanation per gate and the cell state is given:

Forget Gate: This gate is responsible for deciding whether certain information can pass through the network. As seen in Figure 3.8 the information from the forget gate is processed by the *Sigmoid* function, which eliminates the information which has a tendency towards zero from the network.

Input gate: The input gate aids in determining the significance of the information by updating the cell state. The information goes through the *Sigmoid* and *Tanh* functions where the *Sigmoid* function decides the weight of information and *Tanh* reduces the bias of the network.

Cell State: The output of the forget gate and the input gate are multiplied with each other in the cell state. The information that has a chance of dropping out is multiplied by near-zero values. Also the input and output values are added together which updates the cell state with the information which is relevant for the network.

Output Gate: This gate is used to decide what is going to be output. This is done by multiplying the result of the Output Gate (using *Sigmoid* function) with the updated cell state information which has been put through a *Tanh* function (to push the values between -1 and 1). For example, if a language model just saw a subject, it might want to output information relevant to a verb in case that is what comes next. For instance, it could output whether the subject is singular or plural, letting us know what form a verb should be conjugated into if that is what follows next.

3.5.3 Model setup

With the use of the Pytorch Lightning (PL) library it is really straightforward to create an LSTM model. To create a model only a certain group of parameters has to be given to create a network of LSTM layers. The parameters which are given to create those LSTM cells are discussed in the paragraph **LSTM hyperparameters**. In addition, since this is a fore-casting model a regressor layer needs to be added that outputs a single value, which is the predicted price. Next, an optimization algorithm has to be implemented to be able to train the model. There are numerous optimization methods to choose from in today's deep learning libraries. Since it is not always clear which one would perform optimal, three optimizers are chosen to be tested in order to determine which performs best. For this project the Adam, AdamW, and RMSprop (root mean square prop) optimization

algorithms are tested. To be able to compare the models based on their performance the mean squared error (MSE) loss is used. Finally, a dropout layer is added to the model to decrease the possibility of overfitting.

LSTM hyperparameters As stated before the Pytorch Lightning library makes it really easy to implement an LSTM model. The LSTM layer module takes an input size, which is the number of features used as predictors. In this case that are the number of negative, neutral and positive tweets per hour, the total number of tweets per hour, the trading volume and bitcoin price at the end of that hour. Since input size is statics it won't have to change when performing the gridsearch. The parameters which are altered during the gridsearch are the following:

- Batch size: 32, **64**, 128
- Hidden layer size: **32**, 64, 128
- Number of layers: **1**, 2, 3
- Dropout rate: **0.1**, 0.2, 0.3
- Learning rate: 0.01, **0.001**, 0.0001
- Optimizer: **Adam**, AdamW, RMSprop

The values in bold are the optimal values based on the conducted gridsearch. These parameters are used to train the final model which was run for 20 epochs. Since there are four datasets (baseline, total population, influential users, and non-influential users) and 3 different time windows, there will be 12 models to be trained. For each training loop the top 3 epochs were saved based on the validation loss. After each model has been trained, the top three epochs are evaluated on the test set, and the best performing model is selected to compare against the other models.

3.6 Metrics

Both the sentiment and the forecasting model are evaluated using certain evaluation metrics. The sentiment model is a classification model and therefore is evaluated using different metrics in comparison with the LSTM fore-casting model, which is a regression model. The sentiment model is evaluated using accuracy and F1 score, and the LSTM fore-casting model is evaluated using the Root Mean Squared Error (RMSE). For the LSTM model several metrics could have been chosen, however the RMSE penalizes greater error than for example the Mean Absolute Error (MAE). To get a better grasp what each metric evaluates, they will be shortly discussed in the following paragraphs.

3.6.1 Classification

For classification problems the results often can be visualized in a confusion matrix Grandini *et al.* [2020]. Such matrix is a cross table that records the number of occurrences between the actual classification and the predictions. In Figure 4.1 three confusion matrices are shown, where the columns stand for the model predictions and the rows for the true labels. A confusion matrix can be described using the following four classes:

True Positive (TP)/ True Negative (TN): These are the number of occurrences where the predicted value is correct and thus the same as the actual value.

False Positive (FP)/ False Negative (FN): These are the number of occurrences where the predicted values is incorrect and thus differ from the actual value.

Precision & Recall Both precision and recall are metrics used in F1 score, so to understand their meaning they will be described shortly. The precision is the fraction of True Positive elements of a column in a confusion matrix divided by the total number of that column. For example, all instances that were correctly labeled with class "A" are divided by all instances there were labeled with class "A" even though not all were correct. This gives the following equation:

$$Precision = \frac{TP}{TP + FP}$$

Precision expresses the proportion of units a model says are class "A" and they actually are class "A". In other words, precision indicates how much we can rely on the model when it predicts an individual class.

The recall is the fraction of True Positive elements of a row in a confusion matrix divided by the total number of that row. For example, all instances of class "A" that were correctly labeled with class "A" are divided by all instances of class "A" including the instances that are labeled with different classes. This gives the following equation:

$$Recall = \frac{TP}{TP + FN}$$

Recall measures the model's predictive accuracy for a single class: intuitively, it evaluates the model's ability to find the true instances per class.

Accuracy One of the most popular evaluation metric used in (multi-class) classification is accuracy, which can be directly computed using the confusion matrix. It is the sum of all correctly predicted instances divided by all instances. When using a confusion matrix, it is the sum of the main diagonal by the sum of all instances including the main diagonal. This gives the following equation:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

This metric can be quite useful to determine how much the model is correctly predicting on the entire dataset. However, when there is a class imbalance, this measure loses its value because it will always predict the class with the most instances because it will always be better than random guessing.

F1 score When there is an imbalance of classes the F1 score can give a better view on the performance of the model. The F1 score is a harmonic mean of the precision and recall, the equation is as follows:

$$\text{F1-Score} = 2 \cdot \left(\frac{precision \cdot recall}{precision + recall} \right)$$

The F1 score equation can be viewed as a weighted average of precision and recall, with the best score being 1 and the worst score being 0. Precision and recall contribute equally to the F1 score, and the harmonic mean can be used to discover the best trade-off between the two quantities.

For a binary classification problem the earlier mentioned equation would be sufficient. However, when it is a multi-class classification problem, the F1 score should involve all classes.

This can be accomplished by calculating the multi-class F1 score using two different methods: the Micro-averaged F1 score and the Macro-averaged F1 score.

Micro averaged F1 score For the Micro-averaged F1 score, the Micro-precision and Micro-recall has to be computed. The concept of Micro-averaging is to consider all units collectively, without taking into account any variances across classes. Thus, for each class k the Micro-averaged precision and recall are computed as follows:

$$\begin{aligned} MicroavragePrecision &= \frac{\sum_{k=1}^{K} TP_k}{\sum_{k=1}^{K} TotalColumn_k} = \frac{\sum_{k=1}^{K} TP_k}{GrandTotal} \\ MicroavrageRecall &= \frac{\sum_{k=1}^{K} TP_k}{\sum_{k=1}^{K} TotalRow_k} = \frac{\sum_{k=1}^{K} TP_k}{GrandTotal} \end{aligned}$$

As one can see, the Micro-average of precision and recall is the same, therefore the Micro-average F1 score is also the same. Since the harmonic mean of two equal values is just the value. This gives the following equation:

$$Micro\ average\ F1\ score = \frac{\sum_{k=1}^{K} TP_k}{GrandTotal}$$

For the keen eye, one may see that the Micro-average F1 score is equal to the Accuracy, which shows that both metrics have some similarities when measuring multi-class classifications.

Macro averaged F1 score Just as with the Micro averaged F1 score, the Macro-average precision and recall have to be computed to derive the Macro-average F1 score. They are calculated by averaging the precision for each predicted class and the recall for each actual class. So for each class k the precision and recall will be calculated as follows:

$$Precision_k = \frac{TP_k}{TP_k + FP_k}$$
$$Recall_k = \frac{TP_k}{TP_k + FP_k}$$

By simply computing the arithmetic mean using both metrics for all single classes the following equations are obtained:

$$Macro\ averaged\ Precision = \frac{\sum_{k=1}^{K} Precision_k}{K}$$
$$Macro\ averaged\ Recall = \frac{\sum_{k=1}^{K} Recall_k}{K}$$

To get the Macro averaged F1 score, the harmonic mean of the Macro averaged precision and recall is calculated:

$$Macro\ average\ F1\ score = 2 \cdot \left(\frac{Macro\ averaged\ Precision \cdot Macro\ averaged\ Recall}{Macro\ averaged\ Precision + Macro\ averaged\ Recall} \right)$$

Because the numerators of Macro averaged Precision and Macro averaged Recall are composed of values in the range [0,1], Macro-Average methods tend to generate an overall mean of multiple measurements. There is no relationship to class size because classes of varying sizes are equally weighted in the numerator. When using accuracy, the largest class has a greater impact on the outcome, whereas when using the Macro average, the largest class has the same importance as the smaller ones. So, if the Macro F1 score is high, it means that the model performs well in all classes.

3.6.2 Regression

Root Mean Squared Error (RMSE) The root mean squared error is an extension of the mean squared error (MSE), where $\hat{Y}$ represents the predicted value and Y represents the actual values. The value n denotes the total number of values in the test set. This statistic, like MSE, penalizes larger errors more severely. Both metrics can be written like:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \hat{Y}_I\right)^2$$

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \hat{Y}_I\right)^2}$$

Similarly, the RMSE is always positive, with lower values indicating better performance. This technique has the advantage of having the RMSE number in the same unit as the projected value, which makes it easier to understand when compared to the MSE.

4

Results

4.1 BERTweet sentiment model

4.1.1 Results pre-training SemEval dataset

Before the sentiment of the tweets regarding bitcoin was predicted, the sentiment model was fine-tuned using the SemEval dataset. After training the model, the top 3 epochs are evaluated on the test set. The results are given as confusion matrices, which are shown in Figure 4.1. It is clear to see that, when predicting the neutral and positive classes, the model at epoch 4 performs better than the models at epoch 2 and 3. Yet, the model at epoch 3 seems to be better at recognizing negative tweets, whereas the model at epoch 4 have predicted more neutral tweets when the actual sentiment label was negative.

Based on the confusion matrices both the accuracy and F1 scores of the model at 3 different epochs are computed. The results of those computations are shown in Table 4.1. Because the dataset has a class imbalance, the accuracy is less important when comparing the models, so the F1 score is primarily used. Based on those results one can clearly see that the model at epoch 4 outperforms the model at the other two epochs. As a result, the model at epoch 4 will be used to predict the sentiment scores of Bitcoin related tweets.

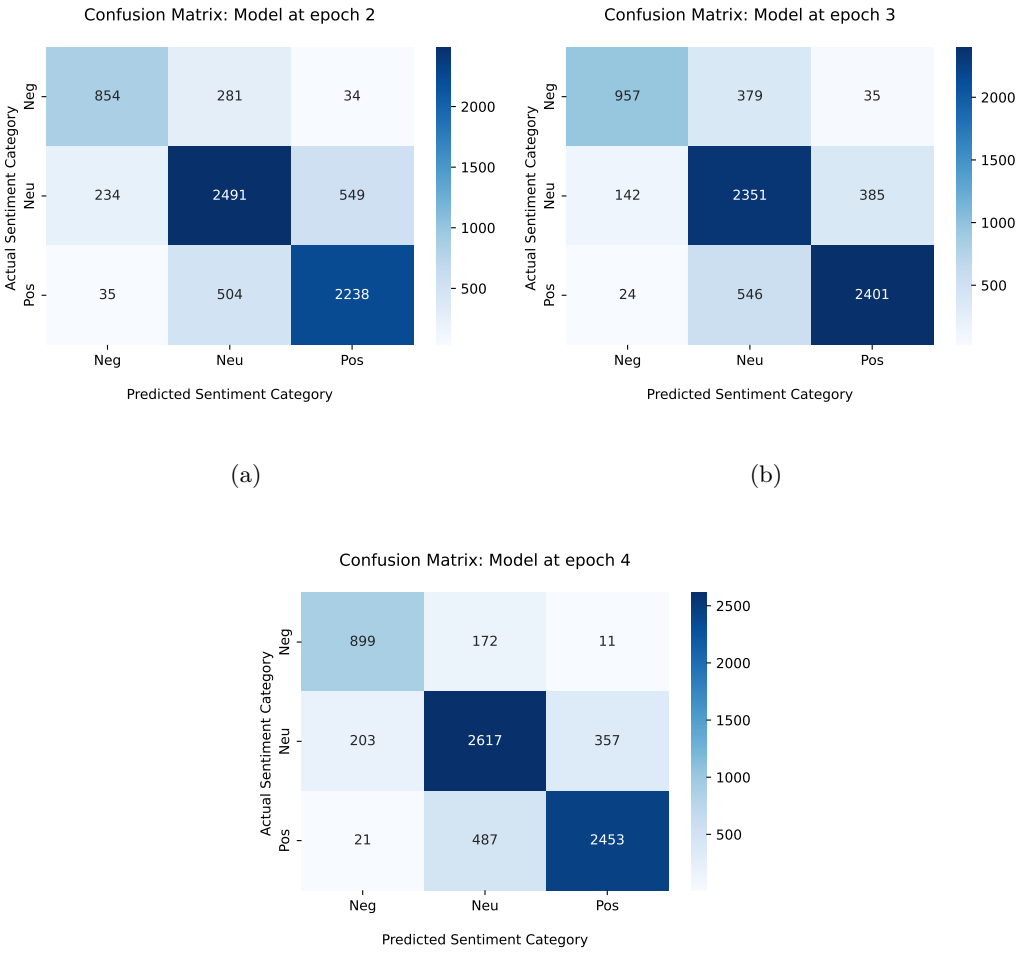

Figure 4.1: Confusion matrices of the fine-tuned BERTweet model at epoch: (a) 2 (b) 3 (c) 4

Table 4.1: The micro, macro and weighed averages accuracy and F1 scores of the fine-tuned BERTweet model at the top 3 epochs evaluated on the test set.

Models	Accuracy			F1 score		
	micro	macro	weighted	micro	macro	weighted
$\text{Model}_{\text{epoch 2}}$	0.7733	0.7658	0.7733	0.7733	0.7685	0.7731
$\text{Model}_{\text{epoch 3}}$	0.7907	0.7744	0.7907	0.7907	0.7869	0.7915
$\text{Model}_{\text{epoch 4}}$	**0.8267**	**0.8277**	**0.8267**	**0.8267**	**0.8250**	**0.8271**

4.1.2 Results predictions of tweets regarding Bitcoin

Following the selection of the best performing model, all tweets were classified and aggregated on a daily and hourly basis. The descriptive statistics of the daily tweets are shown in Table 4.2 and 4.3. As one can see, most of the tweets were classified as neutral followed by positive and negative. The fact that there are far more neutral tweets than positive and negative tweets could be attributed to a huge number of tweets that simply provide information on bitcoin rather than emitting sentiment. This is also evident when compared with the influential and non-influential users.

Table 4.2: Descriptive statistics of each variable of the all tweets including sentiment label aggregated per day.

Variables	Total			
	Mean	SD	Min	Max
Volume	27927	5097	19603	62265
Negative tweets	2371	896	1152	8504
Neutral tweets	19128	3100	13340	38515
Positive tweets	6473	1494	4152	17368

Table 4.3: Descriptive statistics of each variable of the influential and non-influential tweets including sentiment label aggregated per day.

Variables	Influential				Non-influential			
	Mean	SD	Min	Max	Mean	SD	Min	Max
Volume	5631	277	3924	12455	22411	4083	15719	49813
Negative tweets	443	173	211	1699	1934	732	930	6805
Neutral tweets	4020	669	2886	8131	15158	2449	10463	30385
Positive tweets	1167	277	729	3371	5318	1239	3374	13999

4.2 LSTM forecasting model

4.2.1 Results price predictions

As mentioned in Chapter 3 the metric used to evaluate and compare the LSTM models is the Root Mean Squared Error (RMSE). As with the sentiment model, the top 3 epochs of each run were saved, and the best performing model, based on the RMSE score, was

chosen for each run. The results of the best performing models per window size are shown in Table 4.4. It is clear to see that based on the RMSE the *Baseline* model at a window size of 3 performs the best compared to the other models and their corresponding window sizes. However, when the window size is increased for the *Baseline* model the RMSE gets much worse.

What is also interesting to note is that when using the total population of users the model performs best when using a window size of 7 where all other models perform optimal with a window size of 3.

Table 4.4: The root mean squared error (RMSE) of each model evaluated for the windows sizes 3, 5 and 7 (lower is better).

Models	RMSE		
	3 days	5 days	7 days
Baseline	**193.89**	924.60	870.69
Total population	442.40	431.93	**301.48**
Influential users	**325.18**	512.78	359.29
Non-influential users	**349.93**	477.22	879.42

When solely looking at the RMSE results it can be hard to get a good feeling of the actual performance of the model and thus the predictions are plotted against the actual Bitcoin prices to get a better view of the performance. The plots of the best performing models, given in Table 4.4, are shown in Figures 4.2, 4.3, 4.4, and 4.5. It is clear to see that based on the RMSE score and Figure 4.2 the Baseline model with window size 3 seems to predict the prices the most accurate of all models. Yet, in all but one graph it seems that they are constantly predicting above the actual price, only the influential model predicted below the actual prices. Besides being off by quite a large margin, all models seem to follow the trend of the actual price. The plots for all other models can be seen in Appendix B.

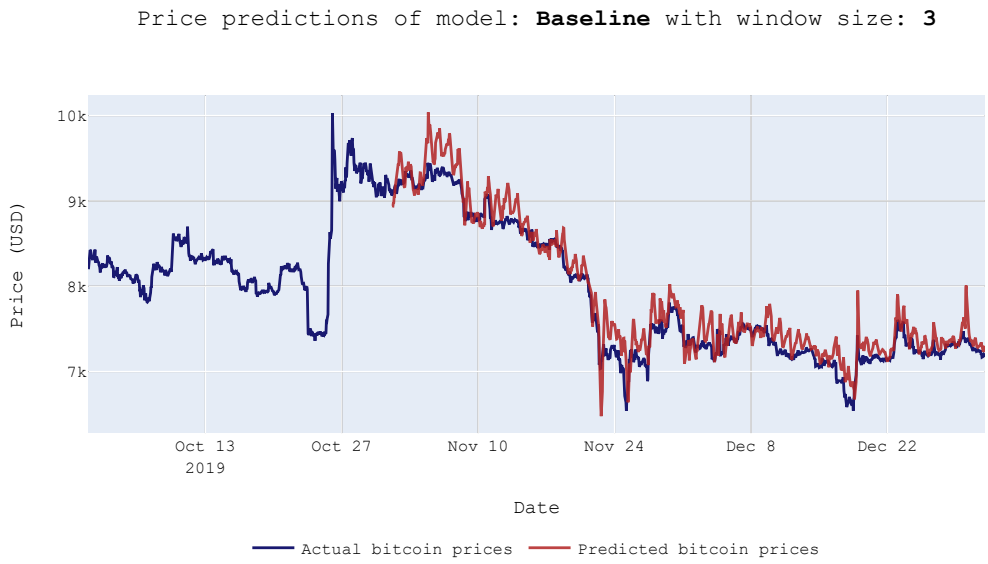

Figure 4.2: Predictions of the bitcoin price of the best performing Baseline model on the test set compared to the actual price.

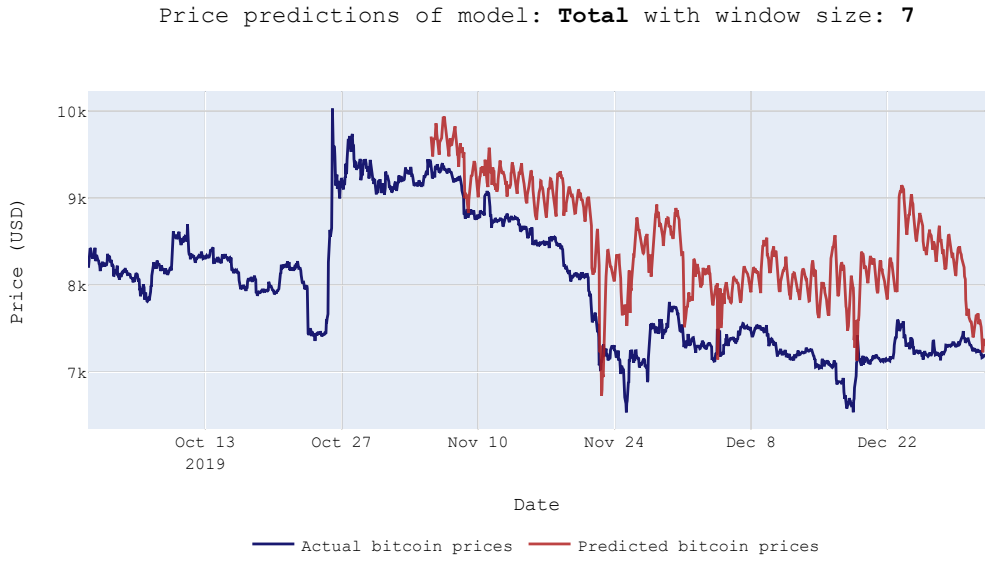

Figure 4.3: Predictions of the bitcoin price of the best performing Total population model on the test set compared to the actual price.

Figure 4.4: Predictions of the bitcoin price of the best performing Influential model on the test set compared to the actual price.

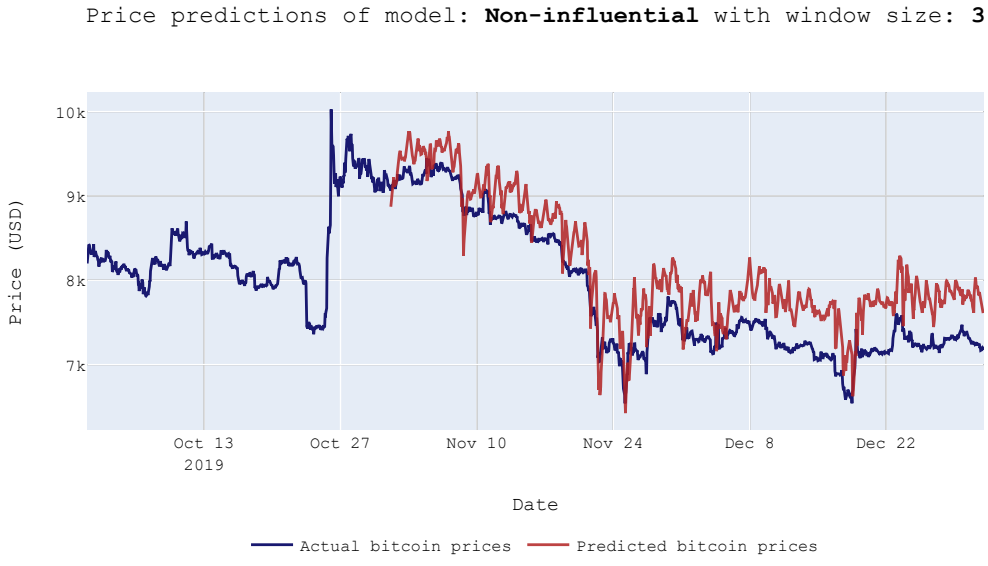

Figure 4.5: Predictions of the bitcoin price of the best performing Non-influential model on the test set compared to the actual price.

5

Discussion

Each model and methods with their corresponding results will be discussed in this chapter.

5.1 Sentiment model

5.1.1 Pre-training BERTweet

Since BERTweet was only pre-trained for using binary classification problems the model had to be fine-tuned to be able to classify three classes. However, since adding more classes can also add complexity to the model, the choice of adding the neutral class could have turned to be a bad choice. Nevertheless, the outcome of the model showed some promising results. Next, the computation of the model took quite some time and resources, so much that it was not possible to train the model on a standard laptop. If this model would be used for future research it therefore might be good to find a way to make it less computational intensive.

5.1.2 Classifying bitcoin tweets

After pre-training BERTweet the model was used to classify the bitcoin tweets. After all tweets were classified it was clear to see that the biggest class was the neutral class. When analyzing the results it seemed that sometimes the model had hard times distinguishing two classes. For example, the class predictions of a tweet could have been: Negative - 0.02, Neutral - 0.82, Positive - 0.81. When selecting the class with the highest probability, the tweet would be labeled as neutral, but the difference in probability between neutral and positive is very small. This could have had some impact on the labels based on those minor differences between probabilities, which could be the reason why there are more neutral tweets than positive and negative tweets.

5.2 LSTM forecasting model

At first there were some thoughts on using a multivariate granger causality test to see if the twitter sentiment statistically causes the price changes of bitcoin. However, such causality tests require a large number of assumptions to be made in order to produce reliable results; thus, the choice was to use an LSTM model, which does not require such assumptions Wu *et al.* [2018].

5.2.1 Prediction bitcoin prices

For the prediction of bitcoin prices it was decided to only use the hourly data since the daily dataset would have too few instances to get reliable results. The results show that when using a short time window the use of Twitter sentiment does not result in better performance when predicting bitcoin prices compared to the baseline model.

However, when a larger time window is used the baseline model performs worse than the model which uses Twitter sentiment. So the information gain of Twitter sentiment might have a positive impact when used for larger time window. This might be due to the inertia of Twitter sentiment, which means that the effect of Twitter sentiment only becomes apparent after more than 3 days.

As one can see in Table 4.4, when using the total population it even performs best when there is a time window of 7. However, the performs for the non-influential users drops quickly when a time window of 7 is chosen, which might indicate that influential users have more impact over non-influential users when a larger time window is used. For the other time windows the scores are quite similar, which might indicate that there is some consistency when predicting prices using the total population and influential users for all time windows.

Next it seems that sometimes the predictions are off from the start, which can heavily impact the predictions made after the initial predictions. Since it was clear to see that for some models. where the initial prediction was off, the predictions actually were following the same trend as the actual values.

6

Conclusion

This chapter will provide some conclusions, answers the research questions and give some ideas for future research.

6.1 Method conclusions

This research was divided into two parts, the first of which involved developing a sentiment model to categorize the sentiment of tweets. In the second part, a forecasting model that was used to predict bitcoin prices included sentiment from the Bitcoin related tweets as a predictor.

Sentiment model As seen in the results the sentiment model was able to predict the three different classes quite well and therefore was also a good model to classify the bitcoin tweets. The results clearly showed that the majority of bitcoin-related tweets are neutral, with more positive tweets than negative tweets. This ratio was the same when the data was divided into influential and non-influential users.

Price predictions The results of the LSTM forecasting model showed that for a short time window adding twitter sentiment information does not improve the price predictions. However, when choosing a larger time window adding sentiment information can make a difference in the performance of price predictions.

6.2 Research Question

In this section both the main and the extension of the main research question are answered:

First answering the main research question:

To what extend is it possible to predict the price of Bitcoin using Twitter sentiment? The results has shown that for a short time window the use of twitter sentiment seems to perform worse compared to the baseline. However, when a bigger time window is chosen adding twitter sentiment had better performance scores and therefore might be of good use when predicting bitcoin prices over a larger time window.

Next, answering the extension of the main research question:

What is the impact of only using sentiment information of the top 20 percent of users based on their number of followers?

When looking at the results it seems that splitting the data based on the number of followers per users does not have a big impact. The overall best performing model, not taking the baseline model into account, was the model using the total population. Nonetheless, the differences were not that big and thus is it hard to say if there is a significant difference in performance.

6.3 Future research

Since there were some choices made which could have impacted the results, some future research ideas are given.

- It was clear that for some models the predictions were off from the start but did follow the trend of the actual price. Therefore the model could be altered into a classification model. So instead of predicting the price the model would predict if the price would go up or down. This probably would make the model less complex.

- Instead of predicting the probability of the three classes, it might be interesting to predict a score between -1 and 1, where -1 is negative, 0 is neutral and 1 is positive. This is somewhat the same as the polarity score which VADER has, which might overcome the issue where the differences between the probabilities of two classes were very small.

- Instead of separating the population based on their number of followers, one could look at the most influential tweets. Thus not taking the top users, but taking the top 20 percent of tweets based on the number of retweets and likes those tweets have.

- When using Language Models there is often an extensive pre-processing step to reduce the noise in texts. This was not done in this research since the authors of BERTweet mentioned that the gain of pre-processing using BERTweet was marginal Nguyen *et al.* [2020]. However, their main goals was not to be able to create a sentiment classifier that predicts three classes, but to improve the performance on several NLP tasks when compared to the other state-of-the-art Language Models. Other authors who did analyze the sentiment of bitcoin tweets did perform more elaborate pre-processing steps Kraaijeveld & De Smedt [2020]. It might be interesting to see whether more extensive pre-processing steps will have a positive impact on the results of the sentiment analysis, which as a consequence might impact the results of the forecasting model.

References

Balabantaray, R.C., Mohammad, M. & Sharma, N. (2012). Multi-class twitter emotion classification: A new approach. *International Journal of Applied Information Systems*, **4**, 48–53. 2

Balcilar, M., Bouri, E., Gupta, R. & Roubaud, D. (2017). Can volume predict bitcoin returns and volatility? a quantiles-based approach. *Economic Modelling*, **64**, 74–81. 9

Baur, D.G. & Hoang, L.T. (2021). A crypto safe haven against bitcoin. *Finance Research Letters*, **38**, 101431. 6

Bollen, J., Mao, H. & Zeng, X. (2011). Twitter mood predicts the stock market. *Journal of computational science*, **2**, 1–8. 2, 5, 7, 8, 9

Cao, J., Li, Z. & Li, J. (2019). Financial time series forecasting model based on ceemdan and lstm. *Physica A: Statistical mechanics and its applications*, **519**, 127–139. 9

Ciaian, P., Rajcaniova, M. *et al.* (2018). Virtual relationships: Short-and long-run evidence from bitcoin and altcoin markets. *Journal of International Financial Markets, Institutions and Money*, **52**, 173–195. 5

Conneau, A., Khandelwal, K., Goyal, N., Chaudhary, V., Wenzek, G., Guzmán, F., Grave, E., Ott, M., Zettlemoyer, L. & Stoyanov, V. (2019). Unsupervised cross-lingual representation learning at scale. *arXiv preprint arXiv:1911.02116*. 19

Devlin, J., Chang, M.W., Lee, K. & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*. 2, 14, 15, 19

REFERENCES

Fama, E.F. (1970). Efficient capital markets: A review of theory and empirical work. *The journal of Finance*, **25**, 383–417. 5

Fama, E.F. (1991). Efficient capital markets: Ii. *The journal of finance*, **46**, 1575–1617. 5

Garcia, D., Tessone, C.J., Mavrodiev, P. & Perony, N. (2014). The digital traces of bubbles: feedback cycles between socio-economic signals in the bitcoin economy. *Journal of the Royal Society Interface*, **11**, 20140623. 1

Giachanou, A. & Crestani, F. (2016). Like it or not: A survey of twitter sentiment analysis methods. *ACM Computing Surveys (CSUR)*, **49**, 1–41. 7

Glorot, X. & Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, 249–256, JMLR Workshop and Conference Proceedings. 20

Grandini, M., Bagli, E. & Visani, G. (2020). Metrics for multi-class classification: an overview. *arXiv preprint arXiv:2008.05756*. 28

Granger, C.W. (1969). Investigating causal relations by econometric models and cross-spectral methods. *Econometrica: journal of the Econometric Society*, 424–438. 8

Hochreiter, S. & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, **9**, 1735–1780. 3

Horev, R. (2018). Bert explained: State of the art language model for nlp. 18

Huggingface (2022). 21

Hutto, C. & Gilbert, E. (2014). 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, 216–225. 2

Kahneman, D. & Tversky, A. (2013). Prospect theory: An analysis of decision under risk. In *Handbook of the fundamentals of financial decision making: Part I*, 99–127, World Scientific. 2

Kandel, I. & Castelli, M. (2020). The effect of batch size on the generalizability of the convolutional neural networks on a histopathology dataset. *ICT express*, **6**, 312–315. 21

Karalevicius, V., Degrande, N. & De Weerdt, J. (2018). Using sentiment analysis to predict interday bitcoin price movements. *The Journal of Risk Finance*. 8

Khuc, V.N., Shivade, C., Ramnath, R. & Ramanathan, J. (2012). Towards building large-scale distributed systems for twitter sentiment analysis. In *Proceedings of the 27th Annual ACM Symposium on Applied Computing*, SAC '12, 459–464, Association for Computing Machinery, New York, NY, USA. 7

Kim, Y.B., Lee, J., Park, N., Choo, J., Kim, J.H. & Kim, C.H. (2017). When bitcoin encounters information in an online forum: Using text mining to analyse user opinions and predict value fluctuation. *PloS one*, **12**, e0177630. 8

Kraaijeveld, O. & De Smedt, J. (2020). The predictive power of public twitter sentiment for forecasting cryptocurrency prices. *Journal of International Financial Markets, Institutions and Money*, **65**, 101188. 8, 43

Kuleshov, V. (2011). Can twitter predict the stock market? *Standford Üniversitesi Çalışma Metni*, 1–5. 8

Lachanski, M. & Pav, S. (2017). Shy of the character limit:" twitter mood predicts the stock market" revisited. *Econ Journal Watch*, **14**, 302. 8

Li, T., van Dalen, J. & van Rees, P.J. (2018). More than just noise? examining the information content of stock microblogs on financial markets. *Journal of Information Technology*, **33**, 50–69. 2, 8

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L. & Stoyanov, V. (2019). Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*. 2, 14, 19

Lo, A.W. (2004). The adaptive markets hypothesis. *The Journal of Portfolio Management*, **30**, 15–29. 6

Lo, A.W. (2012). Adaptive markets and the new world order (corrected may 2012). *Financial Analysts Journal*, **68**, 18–29. 6

Loughran, T. & McDonald, B. (2014). Measuring readability in financial disclosures. *the Journal of Finance*, **69**, 1643–1671. 8

Luo, X., Zhang, J. & Duan, W. (2013). Social media and firm equity value. *Information Systems Research*, **24**, 146–163. 1

REFERENCES

Mai, F., Bai, Q., Shan, J., Wang, X.S. & Chiang, R.H. (2015). The impacts of social media on bitcoin performance. 8

Nakamoto, S. (2008). Bitcoin: A peer-to-peer electronic cash system. *Decentralized Business Review*, 21260. 1

NDTV (2022). Meme coins roared, bitcoin hit all-time high: Know the best performing cryptocurrencies in 2021. 1

Nguyen, D.Q., Vu, T. & Nguyen, A.T. (2020). Bertweet: A pre-trained language model for english tweets. *arXiv preprint arXiv:2005.10200*. 3, 14, 43

Nguyen, T.H., Shirai, K. & Velcin, J. (2015). Sentiment analysis on social media for stock movement prediction. *Expert Systems with Applications*, **42**, 9603–9611. 7

Nofsinger, J.R. (2005). Social mood and financial economics. *The Journal of Behavioral Finance*, **6**, 144–160. 2

Pak, A. & Paroubek, P. (2010). Twitter as a corpus for sentiment analysis and opinion mining. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*. 2

Porshnev, A., Redkin, I. & Shevchenko, A. (2013). Machine learning in prediction of stock market indicators based on historical data and data from twitter sentiment analysis. In *2013 IEEE 13th International Conference on Data Mining Workshops*, 440–444, IEEE. 8

Qian, B. & Rasheed, K. (2007). Stock market prediction with multiple classifiers. *Applied Intelligence*, **26**, 25–33. 5

Rosenthal, S., Farra, N. & Nakov, P. (2017). SemEval-2017 task 4: Sentiment analysis in Twitter. In *Proceedings of the 11th International Workshop on Semantic Evaluation*, SemEval '17, Association for Computational Linguistics, Vancouver, Canada. 12

Sailunaz, K. & Alhajj, R. (2019). Emotion and sentiment analysis from twitter text. *Journal of Computational Science*, **36**, 101003. 2

Siami-Namini, S. & Namin, A.S. (2018). Forecasting economics and financial time series: Arima vs. lstm. *arXiv preprint arXiv:1803.06386*. 9

Siami-Namini, S., Tavakoli, N. & Namin, A.S. (2019). A comparative analysis of forecasting financial time series using arima, lstm, and bilstm. *arXiv preprint arXiv:1911.09512*. 9

Silverman, G., Murphy, H. & Authers, J. (2017). Bitcoin: an investment mania for the fake news era. *Financial Times, December 1st*. 5

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł. & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, **30**. 15, 16

Venna, S.R., Tavanaei, A., Gottumukkala, R.N., Raghavan, V.V., Maida, A.S. & Nichols, S. (2018). A novel data-driven model for real-time influenza forecasting. *Ieee Access*, **7**, 7691–7701. 25

Vu, T.T., Chang, S., Ha, Q.T. & Collier, N. (2012). An experiment in integrating sentiment features for tech stock prediction in twitter. In *Proceedings of the workshop on information extraction and entity analytics on social media data*, 23–38. 5

Wen, X., Rangarajan, G. & Ding, M. (2013). Multivariate granger causality: an estimation framework based on factorization of the spectral density matrix. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, **371**, 20110610. 9

Wu, C.H., Lu, C.C., Ma, Y.F. & Lu, R.S. (2018). A new forecasting framework for bitcoin price with lstm. In *2018 IEEE International Conference on Data Mining Workshops (ICDMW)*, 168–175, IEEE. 3, 40

Xie, P., Wu, J. & Wu, C. (2017). Social data predictive power comparison across information channels and user groups: evidence from the bitcoin market. *The Journal of Business Inquiry*, **17**, 41–54. 1

Appendix A

Word Masking

BERT trains the language model by predicting 15% of the tokens in the input that were chosen at random. These tokens are pre-processed in the following manner: 80% are replaced with a [MASK] token, 10% with a random word, and 10% with the original word. The authors chose this approach based on the following intuition:

1. If [MASK] is used all the time, the model might not produce good token representations for non-maksed words. The model was optimized for predicting the masked words, however the non-masked tokens were still used to determine the context.

2. If [MASK] is used 90% of the time and random words 10% of the time, the model will learn that the observed word is *never* correct.

3. If [MASK] is used 90% of the time and the same word is used 10% of the time, the model might just duplicate the non-contextual embedding.

Appendix B

Model predictions

B.1 Baseline

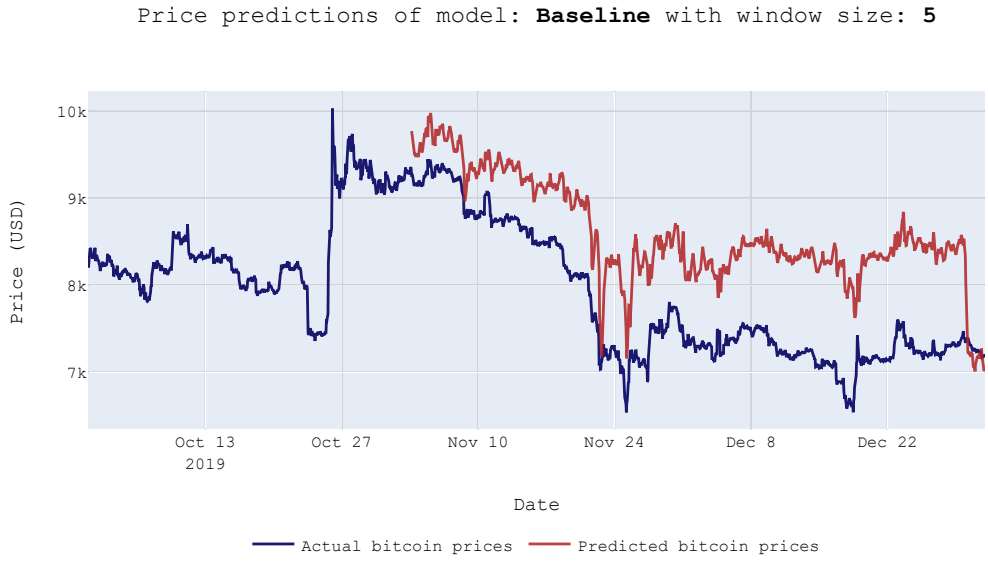

Figure B.1: Predictions of the bitcoin price of the Baseline model with window size 5 on the test set compared to the actual price.

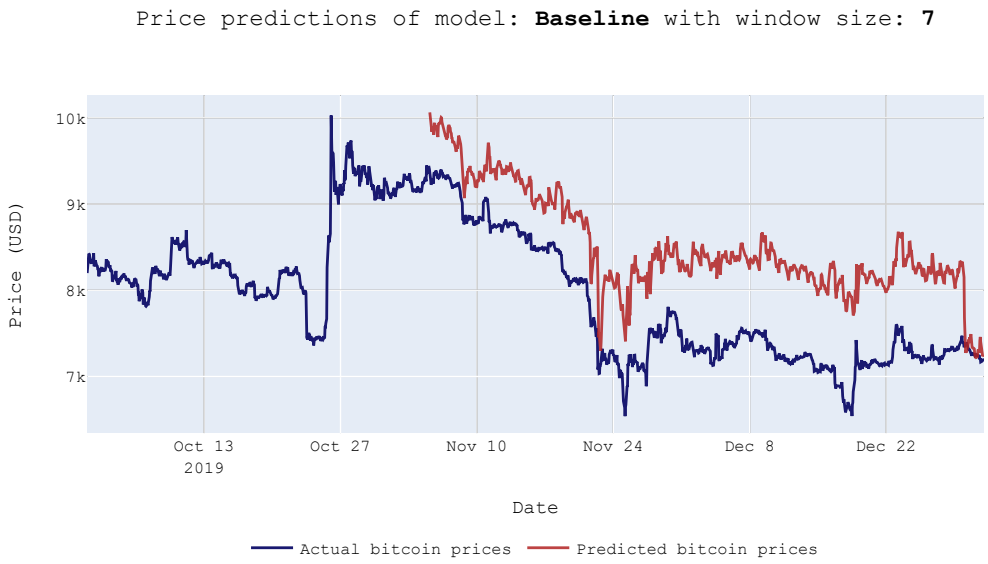

Figure B.2: Predictions of the bitcoin price of the Baseline model with window size 7 on the test set compared to the actual price.

B.2 Total

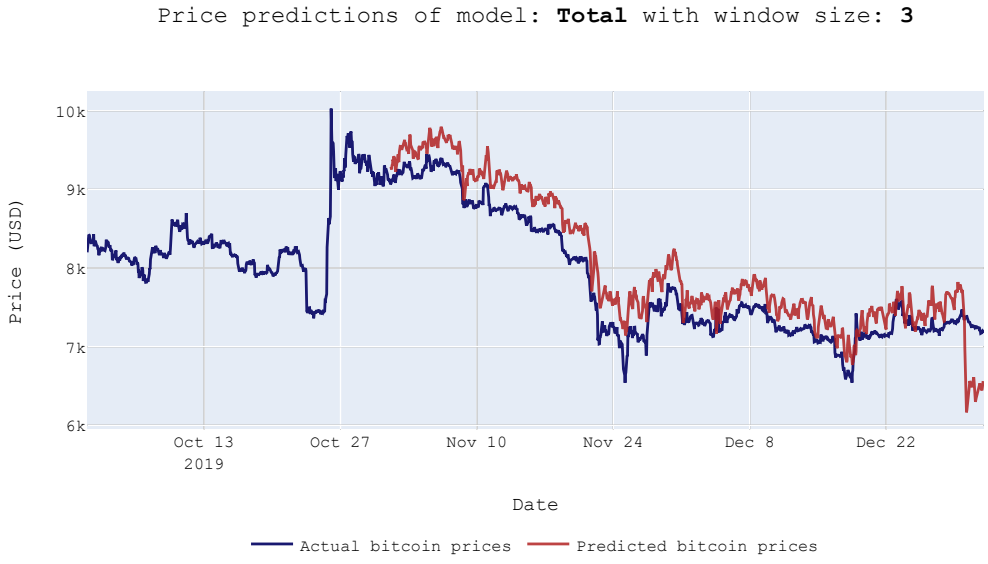

Figure B.3: Predictions of the bitcoin price of the Total model with window size 3 on the test set compared to the actual price.

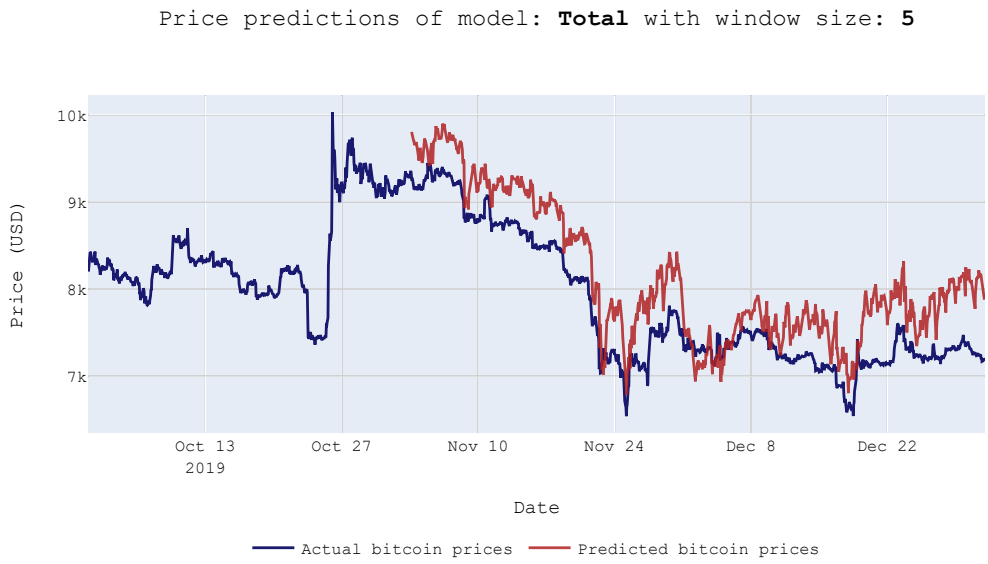

Figure B.4: Predictions of the bitcoin price of the Total model with window size 5 on the test set compared to the actual price.

B.3 Influential

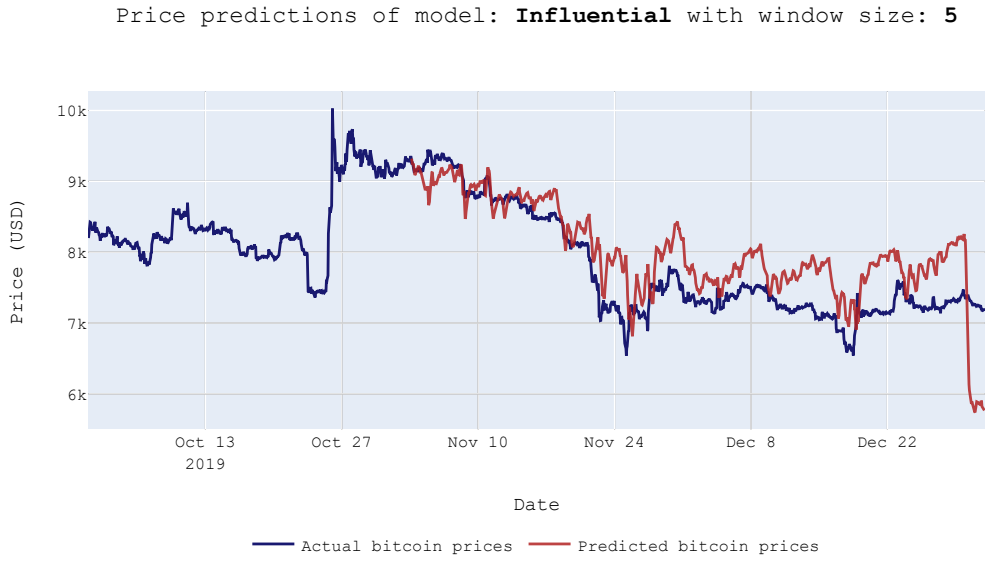

Figure B.5: Predictions of the bitcoin price of the Influential model with window size 5 on the test set compared to the actual price.

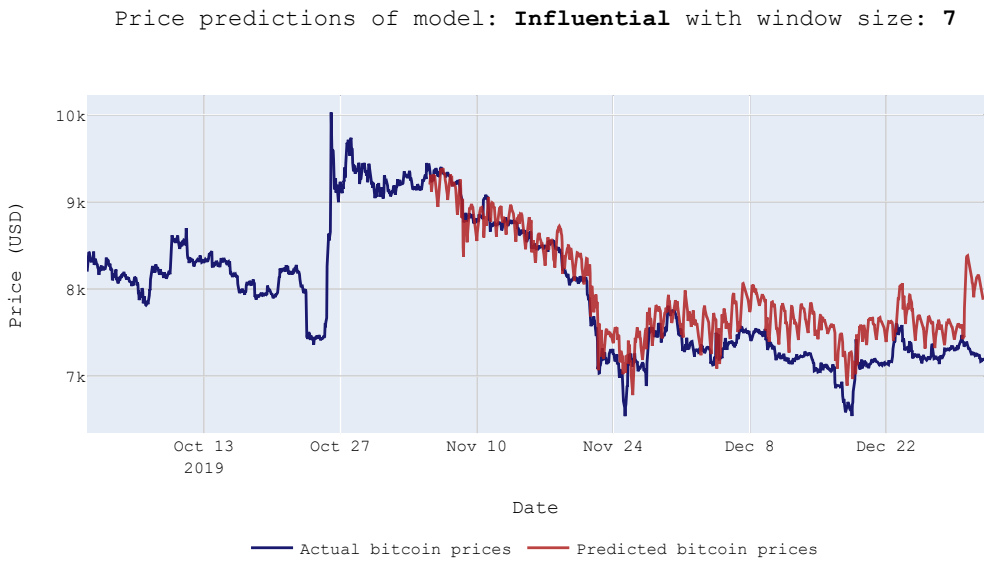

Figure B.6: Predictions of the bitcoin price of the Influential model with window size 7 on the test set compared to the actual price.

B.4 Non-influential

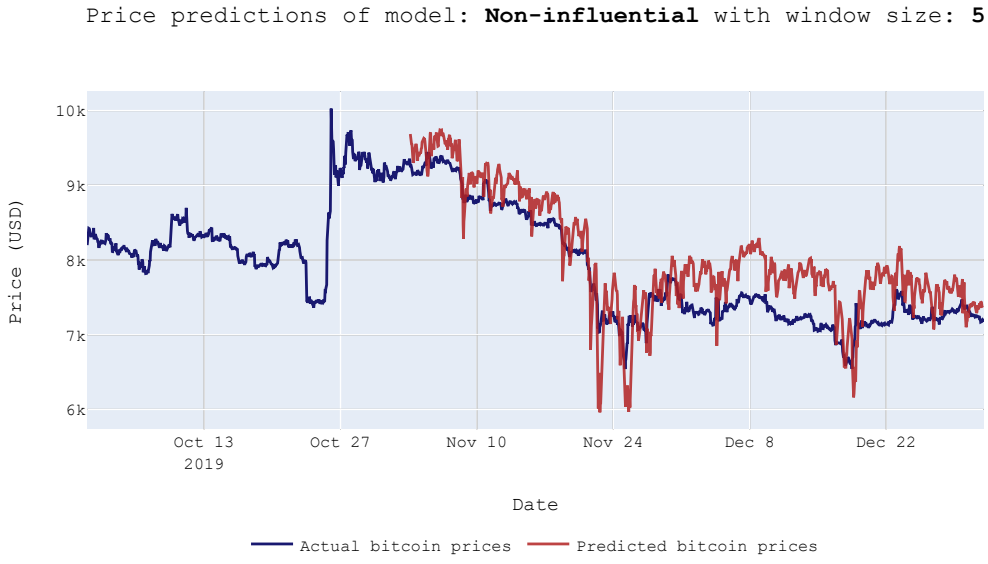

Figure B.7: Predictions of the bitcoin price of the Non-influential model with window size 5 on the test set compared to the actual price.

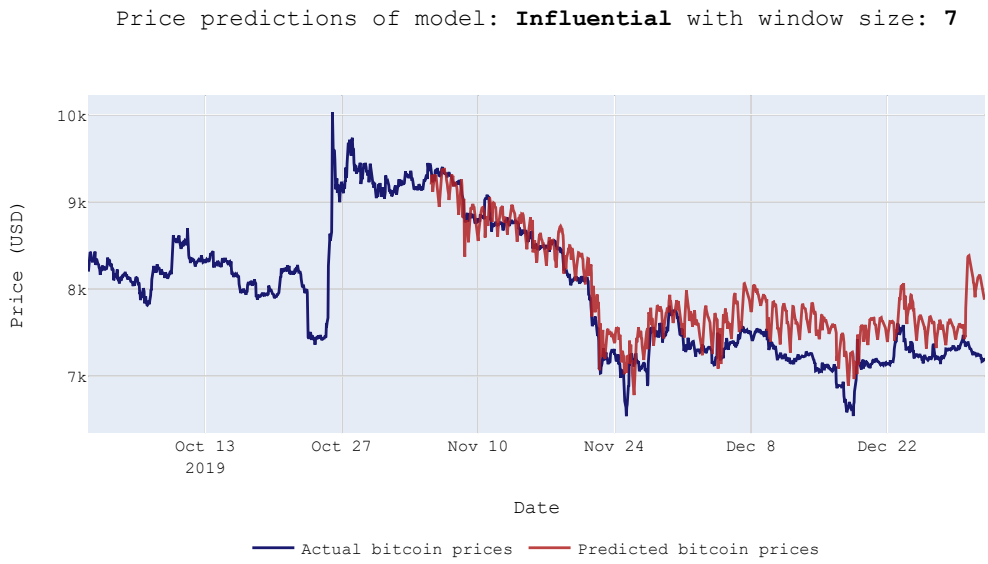

Figure B.8: Predictions of the bitcoin price of the Non-influential model with window size 7 on the test set compared to the actual price.