PONTIFICIA UNIVERSIDAD CATOLICA DE CHILE
SCHOOL OF ENGINEERING

BITCOIN PRICE PREDICTION TROUGH STIMULUS ANALYSIS: ON THE FOOTPRINTS OF TWITTER'S CRYPTO-INFLUENCERS

GERMÁN ALFREDO CHEUQUE CERDA

Thesis submitted to the Office of Research and Graduate Studies
in partial fulfillment of the requirements for the degree of
Master of Science in Engineering

Advisor:
JUAN REUTTER

Santiago de Chile, June 2021

PONTIFICIA UNIVERSIDAD CATOLICA DE CHILE
SCHOOL OF ENGINEERING

BITCOIN PRICE PREDICTION TROUGH STIMULUS ANALYSIS: ON THE FOOTPRINTS OF TWITTER'S CRYPTO-INFLUENCERS

GERMÁN ALFREDO CHEUQUE CERDA

Members of the Committee:
JUAN REUTTER
DENIS PARRA
FELIPE BRAVO
CHRISTIAN OBERLI

Thesis submitted to the Office of Research and Graduate Studies
in partial fulfillment of the requirements for the degree of
Master of Science in Engineering

Santiago de Chile, June 2021

To the people who have faced the worst of these tough times.

ACKNOWLEDGEMENTS

I am deeply grateful for all those who have helped me get to this moment.

Juan Reutter de la Maza, for many great conversations, our discussions were always enriching in many ways, for your support with the decisions made and your help in the presented opportunities for sharing our work. For being committed to your students as well as the well-being of everyone at the DCC.

Denis Parra, for promoting research and sharing it with colleagues and the community. For generating spaces for conversation and being a participant, sharing your knowledge with others beyond the academy. For the joy, you share in your work.

All at the DCC, for making it a welcoming place to work.

Every teacher who has bequeathed me their love for science, as well as the ones that have taught me about the responsibility behind the knowledge which must be aware of its applications and their consequences as well as the goals pursued by their financiers.

Alonso Medina, being with you has given me great learning and has filled me with love as I have never known before. Having you as my support strengthens me. For every moment of joy by your side, soaking me up by your wisdom.

My parents, for being people full of love and dedication to their loved ones and the things they like. The kindness, the appreciation of our cultural roots, and the affection for what we do are some values I recognize from both of you in me.

My brothers César and Marcos, for reminding me what is important. Life is fragile but also resilient, we have each other.

Marisol Ramírez, for your care and concern for the good living of everyone, by creating community and trust among all individuals. For inspiring changes in my relationship with the environment.

Nayareth, Andrés, Elena, Bárbara, Ignacio, Nicolás, Seba and all my friends. Sharing time with you is just fun and a good reminder that happiness can be in simple things.

TABLE OF CONTENTS

LIST OF FIGURES

ABSTRACT

The Bitcoin protocol and its underlying cryptocurrency have started to shape the way we view digital currency and opened up a large list of new and interesting challenges. Amongst them, we focus on the question of how is the price of digital currencies affected, which is a natural question, especially when considering the price roller-coaster we witnessed for bitcoin in 2017-2018. We work under the hypothesis that price is affected by the web footprint of influential people, we refer to them as crypto-influencers.

In this research, we provide models based on machine learning (neural networks and random decision trees) to predict the price of bitcoin. We compare what these models learn from recent price history versus what they can learn when they are fed additionally with encoded textual information obtained from influential users as opinions. According to our results, we show that humans have an average performance at interpreting Bitcoin referred opinions of 75%. For this reason and to increase our possibilities, we have explored seven models trained for encoding this kind of information, including two of the most recent models on inductive transfer learning techniques: ULMFIT and BERT.

We show preliminary evidence that Twitter data should help predict the price of bitcoin by being able to support the decision process at the task of predicting the price of Bitcoin one day in the future. We managed to make our models go from projecting the current price into the future as the only and best prediction to making more realistic ones that we can only attribute to the introduction of an external price stimulus retrieved from the feature selection process. To challenge our models, we explored the price simulation task. Our results are not conclusive in this regard, since we have models that show a different level of performance in a different time span in the future. However, these results do not show awareness of changes to more than a week from the simulation started.

Our best model generates predictions supported by the opinion of these influential users, but it is a prediction with an acceptable error up to a maximum future window of

only one day. Recent evidence shows that Bitcoin's high variability behavior is happening again. By February 2021, Bitcoin reached a new all-time high of almost $57000 USD, which is evidence of how volatile is the behavior of this crypto asset.

Keywords: Bitcoin, Recurrent Neural Networks, XGBoost, Twitter, Influential Users Identification, Sentiment Analysis, Price Prediction, Price simulation, Stance Detection, Inductive Transfer Learning, ULM-FIT, BERT.

RESUMEN

El lanzamiento del protocolo de Bitcoin y su subyacente cripto moneda, han comenzado a dar forma a la manera en que vemos las monedas digitales, abriendo una gran lista de nuevos e interesantes desafíos. Entre ellos, nos centramos en el objetivo de investigar cómo se ve afectado el precio de las monedas digitales. Lo cual por cierto, es una pregunta natural, especialmente cuando consideramos la montaña rusa de precios que presenciamos para Bitcoin entre 2017 y 2018. En esta investigación trabajaremos bajo la hipótesis de que el precio se ve afectado por la huella digital de personas influyentes. Nos referiremos a ellos como criptoinfluencers.

En esta investigación proporcionamos modelos basados en aprendizaje automático (redes neuronales y árboles aleatorios de decisión) para predecir el precio del bitcoin. Comparamos lo que sucede cuando estos modelos se alimentan solo con el historial de precios reciente versus lo que sucede cuando se alimenta adicionalmente con información textual codificada obtenida desde estos usuarios influyentes a partir de sus opiniones. De acuerdo a nuestros resultados, mostramos que los humanos tienen un desempeño promedio al interpretar opiniones referentes a Bitcoin de un 75%. Por esta razón y para aumentar nuestras posibilidades, hemos explorado siete modelos entrenados para codificar este tipo de información, incluyendo dos de los modelos más recientes en técnicas de transferencia inductiva del aprendizajes: ULMFit y BERT.

Mostramos evidencia preliminar de que los datos de Twitter deberían ayudar a predecir el precio de bitcoin al ser capaz de respaldar el proceso de decisión en la tarea de predecir el precio de Bitcoin un día en el futuro. Logramos hacer que nuestros modelos pasaran de proyectar el precio actual hacia el futuro como la única y mejor predicción, a hacer unas más realistas, cosa que solo podemos atribuir a la introducción de un estímulo externo al precio y al proceso de selección de variables. Para desafiar nuestros modelos, exploramos la tarea de simulación de precios. Nuestros resultados no fueron concluyentes en este aspecto, ya que tenemos modelos que muestran diferentes niveles de rendimiento

en diferentes lapsos de tiempo hacia el futuro. Sin embargo, estos resultados en general no muestran consciencia a cambios de precio más allá de una semana de iniciada la simulación.

Nuestro mejor modelo genera predicciones siendo respaldado por la opinión de estos usuarios influyentes, pero es una predicción con un error aceptable hasta una ventana futura máxima de solo un día. La evidencia reciente muestra que el comportamiento de alta variabilidad de Bitcoin está sucediendo nuevamente. Para febrero de 2021, Bitcoin alcanzó un nuevo máximo histórico de casi 57000 USD, lo que es evidencia de cuán volátil es el comportamiento de este cripto-activo.

Palabras Claves: Bitcoin, Redes Neuronales Recurrentes, XGBoost, Twitter, Identificación de Usuarios Influyentes, Análisis de Sentimiento, Detección de Posturas, Predicción de Precios, Simulación de Precios, Transferencia Inductiva de Aprendizaje, ULMFIT, BERT.

1. INTRODUCTION

It was 2008 when Bitcoin's white paper appeared to the world for the first time. Since then, the well-called cryptocurrency has been involved in more than one news, sparking an interest that at times left no one indifferent, whether for its mysterious and unknown author Satoshi Nakamoto, who signed the paper or by the way Bitcoin's price has behaved through these last years, becoming a valuable asset nowadays.

In this research, we aim to analyze a specific aspect of this currency: its price evolution and the high volatility exhibited so far, which has surprised the leading financial experts since it is not clear what this variability depends on. In this research, we propose to support some prediction models with the information obtained from textual information about what influential personalities think on Twitter about the Bitcoin market.

1.1. The reasons behind Bitcoin's success

Some of the reasons behind this successful scenario are partially explained by the smart way 'they'[1] solved some historical troubles in creating a safe and anonymous way of transacting through digital cash. The release of the Bitcoin's Protocol [**Nakamoto, 2009**] unleashed bitcoin as the world's first decentralized cryptocurrency that works quite well in terms of cash. Numerous previous attempts such as hash cash, digicash, or bit-gold were unsuccessful solutions in developing the same idea.

Historically, there have existed two systems in which people have exchange goods and services that also solve the problem of coordination[2]: cash and credit. Through the last decades, credit has found apogee to the eaves of banks that offer this service remotely, as well as online debit payment systems, that are intrinsically associated with an intermediary architecture such as PayPal or Web Pay, which while may not have further implications in

[1]Some people say that the original Satoshi Nakamoto, the name of the user who released the paper, was not a man or woman but an organization or a group.

[2]This refer to the characteristic of money (in the form of cash or credit) of being useful for trading anything, making any exchange to occur in "easily" coordinated agreements, which is a missing feature in barter, for example.

privacy, make lose simplicity for the user by interacting indirectly with the seller. Thus, there was no such equivalent for cash for a long.

Two main advantages have cash systems over the other ones [**Narayanan et al., 2016**], [**Levy, 2001**]. The first is anonymity and the second that cash can enable offline transactions. While it is real that Bitcoin does not quite ensure totally both properties, it comes close enough to be useful as cash. The first solution to ensures anonymity came from cryptography, the idea is generating some mathematical problem to encrypt the user information making transaction not traceable. The second property is accomplished partially by solving the "double spending" problem[3], with the idea of secure timestamping, by recording a ledger of transactions, which was first discussed in 1991 by Haber and Stornetta.

In the original paper, Satoshi detailed the methods of using a peer-to-peer network to generate what was described as *a system for electronic transactions without relying on trust*. So, Nakamoto's contribution was more than the Bitcoin itself, but the development of a whole decentralized system that ensures trust but without relying on a central authority or even in the trust of each of their participants. Indeed, blockchain technology, a ledger in which all Bitcoin transactions are securely recorded by using cryptography hashes and digital signatures, is where Bitcoin's wealth resides.

The way how a currency is likely to acquire real value is being scarce by design. In fact, this is the reason why gold or diamonds have been used as a backing for money. In the digital realm, one way to achieve scarcity is to design the system so that minting money requires solving a computational problem (or "puzzle") that takes a while to crack, then as more people become interested in using it, more valuable in money equivalent Bitcoin becomes[4].

[3]This problem refers to the possibility of using digital cash multiple times while the system is offline and the information cannot be corroborated.

[4]Because there are no so many bitcoins to acquire. The protocol ensures the creation of 21 million bitcoins in total, today approximately 18 million from them are already in circulation.

While it took to Bitcoin some time to gain traction, in the last few years a huge number of alternate cryptocurrencies have appeared, and they have become popular to the point that almost every financial agent has at least considered investing in them. At the time of writing, the price of one single bitcoin has stabilized around 9100 USD, after an all-time high of almost 20000 USD just four years ago. But while this price bubble is now understood to have burst –at least partially–, some other concerns about the condition of the bitcoin market are still not well understood.

1.2. The Bitcoin price volatility problem

One of the most recurrent questions about the bitcoin market has to do with its volatility: what factors influence the variation in bitcoin price? There has been some research that links bitcoin price with Google searches [**Matta et al., 2015**], which is remarkable because other comparable assets such as gold are known to have very little correlation with this indicator. However, in conversation with people in the cryptocurrency market, we have suggested another indicator: tweets and posts from a number of international influencers.

As a monetary good, Bitcoin is not free from speculation, which reinforces our theory about the influence of users, specifically what they are sharing and commenting with everyone else in the media, because there are not only opinions but events happening in real-world shared as news, that also influences the markets. Some recent examples of these kinds of situation could be this [**Browne, 2019**], the cited article refers to the suspicious and distant look of the past U.S. president to the launch of a new cryptocurrency (linked to Facebook) in 2019, surprisingly a day later an important fall in the value price of the Bitcoin was registered. Similarly in this other article [**Bambrough, 2020**], the journalist discusses how Bitcoin price fell down, after the tension of Iranian relations with the United States. There are also some experiences at supporting prediction models by using sentiment information from Twitter posts ([**Gabrovšek et al., 2017**,**Pagolu et al., 2016**]).

1.3. Objectives and research approach

The goal of this research is to validate the hypothesis that the price of bitcoin is affected by the web footprint of popular actors in the international crypto market. Preliminary research has told us that the amount of digital material does not appear to be correlated with bitcoin price, so we cannot settle for a simple answer like in [**Matta et al., 2015**]. But we can do more: what if supportive messages affect the price in a positive way, and disagreement messages in a negative way? This idea gives us a clear map of what needs to be done: First, gather tweets from the most influential users in the cryptocurrency world. Next, analyze what stance (supporting, disagreement, or neither) each user takes with respect to each message they communicate and then analyze through predictive models if this data actually does affect bitcoin's price.

Influential users can be selected in a greedy way, starting from some well-known population of popular users. But how can we show that this data affects the price? We can proceed as in [**Zhang et al., 2018b**] and [**Zhang et al., 2018a**], and compare the prediction capabilities of some model that uses only the information of bitcoin's price in the past with the same model that incorporates this time the external information in some way into the time series.

The price prediction problem is not new, there are numerous studies that address it from the most diverse approximations, using Random matrix theory analysis over stock market prices [**Utsugi et al., 2004**] or through feeding neural networks [**Wanjawa, 2016**, **Jia, 2016**] and studying it against volatility [**Petneházi and Gáll, 2018**], also there is an experience in such aspect by using a Bayesian approach [**Ning and Shephard, 2017**]. The simplest one seems to be to generate a regression from the observed points of the series to try to forecast their variations in the future. While a lot of algorithms have been developed for these purposes, the emerge of sophisticated Machine Learning techniques, such as Neural Networks and boosted randomized decision trees, motivate us to explore these models.

A pending question is: How can we extract the meaning of the messages and their relation to the Bitcoin behavior?

Natural Language Processing (NLP) is known the field that combines linguistics, computer science, and empirical knowledge retrieved from artificial intelligence with the objective of developing comprehension within the interactions between computers and human language. The task studied by this area ranges from the most diverse classification tasks, question answering, information retrieval as most web browsers do, summarization, information extraction or machine translation to sentiment analysis, spam filter in our emails or the well-known applications in our smartphones like the auto-predictor and auto-corrector when we write a message.

The simplest way to try out our objective is to treat it like a classification problem to some label's representation of *stance* of the messages retrieved from social media refereeing Bitcoin. While, to our knowledge, there exist some previous work in analyzing stance toward a target [**Mohammad et al., 2017**], there is no exist known algorithm that generates similar classification for Bitcoin specifically, so, we are strongly interested in some recent applications of inductive transfer learning techniques, which are very popular in the area of computer vision but recently developed for NLP's concerns. The inductive transfer learning consists exactly in not training a model from scratch, but benefits from already obtained knowledge of language models previously trained in huge amounts of data to, through a fine-tuning process make our model aware of the idiosyncrasy of the new data and adapting their outputs to the desired task, in this case, to generate a good *stance classification* of these messages. Inductive transfer learning is very promising in understanding how language comprehension is learned by those models. The main reason for considering transfer techniques is because we do not see this task as trivial. Indeed, we do not want to know the sentiment that messages state, which generally correlates with the election of used words. In this way, the task that we are seeking is more complex. Also, the scarcity of labeled data for this purpose is difficult to deal with, making it virtually impossible to train a learning model from scratch, something that transfers algorithms

as [**Howard and Ruder, 2018**] and [**Devlin et al., 2019**] promise to face using previous learning.

The main contributions of this work are two. First, the application of NLP state-of-the-art algorithms and their adaption to the stance detection problem, as well as the creation of well-located labeled examples for training these stance algorithms. On the other hand, we contribute with a novel receipt for studying the price prediction problem, introducing external stimulus as indicatives of the outgoing influence of some user in media, we show promising evidence about the usefulness of this information.

Organization. In this work we show promising preliminary results, showing that incorporating Twitter data can lead to better forecasting models. However, our predictive power is limited to just a couple of days in the future. We think there is still room for improvements, as there are many different ways to follow our general map, and we have only studied a small number of them. This research is presented as follows: in Section 2 we deepen our objectives; present the data we are working with and explain the type of prediction we will focus on in the rest of this research. Section 3 contains the architecture of our model, discusses the need for an appropriate scaling, and presents the results of the realistic problem of predicting price one day in the future, with their new complications on more distant predictions in time. Section 4 introduces the opinion information obtained from Twitter, we refer to our experience at a labeling session, in which we obtained the data used at fine-tune process. Section 5 presents our discussion about how to include the retrieved information so far, we describe the feature selection process made for including stance. The final results of prediction and simulating price with the embedded data are presented in section 6. We present promising results in predicting price one day in the future but also some limitations. We conclude this research in section 7 with our main ideas and discoveries as well as the challenges to be explored in future work.

2. ON THE PREDICTION OF MONETARY GOODS

Predicting the price of monetary goods is a problem that has been thoroughly studied for several decades and from a series of different areas. As such, it is almost impossible to give a full overview of all the different versions of these problems, or the techniques used to solve them. In the case of Bitcoin, the data presented in section 2.1 is highly variable, so a particularly challenging problem to deal with. However, we do have one important aspect to discuss: the problem of predicting the immediate price of monetary goods that are subject to a high volume of transactions can be dealt with (up to a very reasonable error) with standard machine learning techniques. For the case of Bitcoin, in Section 2.2 we provide a neural network that can effectively predict the price of bitcoin in the next 5 minutes, and we test it to show that the error is indeed quite small. But then, if one can predict the price, why is not everyone arbitraging the market? The problem is that, if one wants to arbitrage bitcoin prices by taking advantage of a model that predicts the price in the next 5 minutes, one would need a prediction that is essentially perfect. For this reason, we focus instead on other problems which cannot be directly dealt with with standard techniques. One problem is predicting the price of Bitcoin after 24 hours of reading the last input, and the other problem has to do with training a model that can properly simulate bitcoin prices for a long period of time. We define these in more detail in Section 2.3

2.1. Materials: samples and range of prediction

We draw Bitcoin price data from bitcoincharts.com and focus on the interval between July 1st, 2018, and January 21st, 2019. The data contains information about the opening and closing prices of Bitcoin in intervals of 5 minutes, giving a total of 83560 samples. To avoid complications between these two measures (open and close values), we decided to work with the average price of each instant. This price evolution over the referred dates is presented in Figure 2.1. It is important to mention that some of the points (less than 30) of the original data were missing, in such case we repeat the last observation

completing the number of registries. In all our models we use the first 67160 data points as training data [1], and use the remaining to test our predictions.

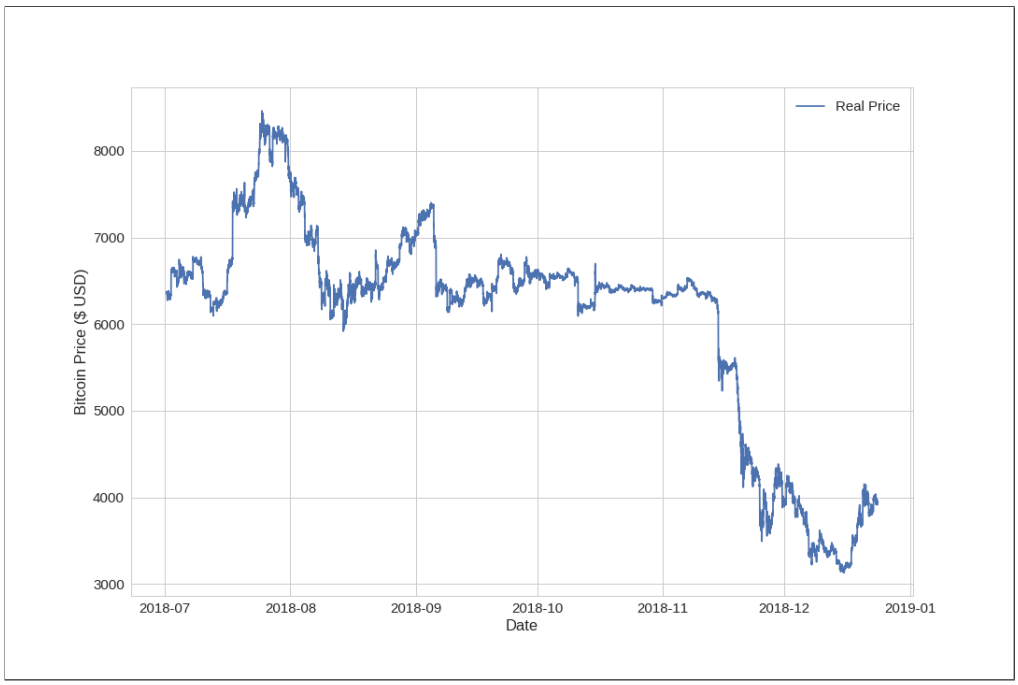


FIGURE 2.1. Bitcoin price time series.

As everyone in the industry knows, the price of Bitcoin is still far from reaching a stable behavior. And while the high frequency of registries is definitely good news for learning models in predicting its price, it also carries a curse: a large number of variations can be very difficult to learn, and occasional points with high variability would tend to be magnified by the necessary scaling that we must perform on our data, which can also generate that some local variations can be underestimated in relation with the higher ones. This means that predictors may either choose to focus on these high variability changes, and therefore loosing constant small changes, or focus on small changes and treating big raises or drops as outliers. Among the many possible solutions to this, we bet for a simple and well-known measure in market behavior studies. To solve this noisy effect without losing our high-frequency chart, we introduce the well-known quantity called *moving mean* or *moving average* measure, defined at equation 2.1

[1]We always train with a random 10% of the sample reserved for cross-validation.

$$MM = \frac{1}{N}\sum_{i=1}^{N} p_i \tag{2.1}$$

where p_i is the current price at date i and N indicates the number of samples that are considered on that average. The idea behind this measure is to work with a constant window of size N and consecutively move their bounds to get a continuous measure of how the price varies on average, creating smoothness in the micro and preserving variability in the macro.

Figure 2.2 shows us how moving average captures the variability of the price with a smoother approximation than the one presented in figure 2.1. We remark that moving average is one of the most used metrics in stock exchange markets, to the point that its inclusion is considered folklore.

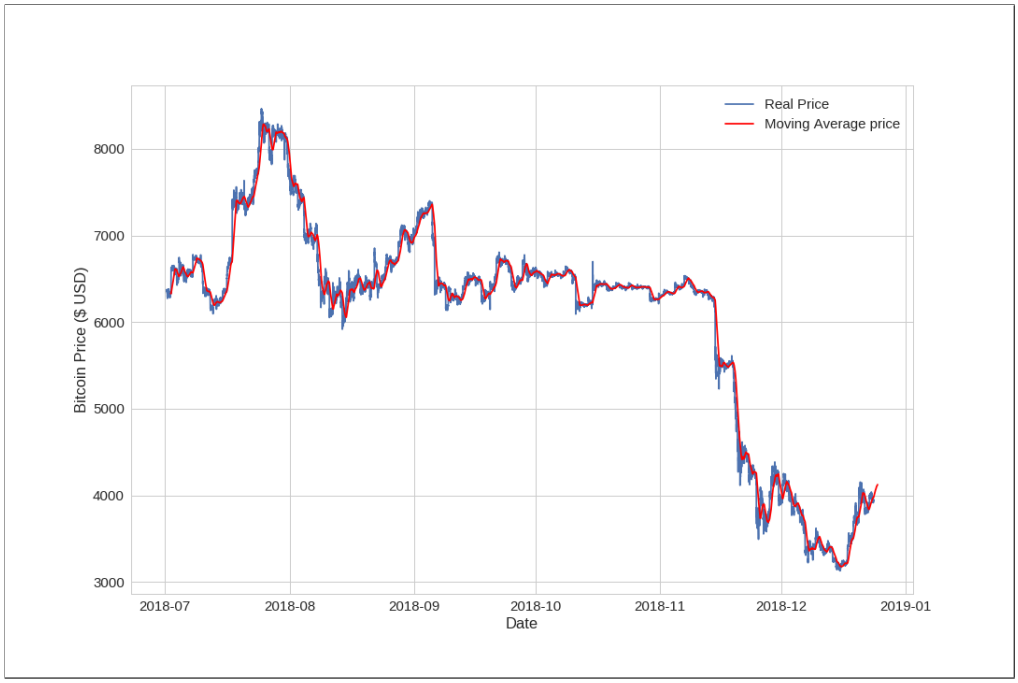


FIGURE 2.2. Comparison between price and moving average with a windows size of 288 measures, equivalent to 1 day of observations with a frequency of 5 minutes.

2.2. Predicting the next data point.

If we want to predict the evolution of a series, the first idea that comes to mind is that this value should depend in some way on the incoming history of recent variations. If all we want is to predict the next data point in the series, we can obtain reasonable predictions by using state-of-the-art neural networks. Many studies have addressed the problem of forecasting by using *LSTM*s in the most diverse fields. Some studies have worked on e-commerce [**Bandara et al., 2019**], others have made the same by analyzing the weather-based in history [**Karevan and Suykens, 2018**], even studying the human blood pressure as a continuous sequence of measures [**Girkar et al., 2018**]. Other studies have made important advances in popular computer tasks, such as question Answering [**Nguyen et al., 2018**], [**Jung et al., 2018**] or speech recognition [**K et al., 2019**] and even in the development of autonomous vehicles [**Wehbe et al., 2018**], all with the use of LSTM.

An LSTM is a special kind of Neural Network that stands for Long-Short Term Memory and is one of the most popular applications of *recurrent* neural networks. This kind of net is characterized by benefiting themselves from sequential data, just like for example, a time series, a sequence of words forming a message, paragraph, or even a sequence of the different events that a person activates by interacting with a sensor, a web page and so on. Within the recurrent networks, we can find the RNNs, GRUs, and finally the LSTMs. The novelty of these last ones from the original RNNs is found in the definition of their neurons, which are specially designed to solve the vanishing gradient problem [**Hochreiter, 1998**], better than the other models, this is accomplished by the introduction of several gates that update and control the cell states, the knowledge encoded in these state vectors captures long-term dependencies and relations in the sequential data, preventing to be forgotten while new data is seen [**Hochreiter and Schmidhuber, 1997**].

We chose to build a Recurrent Neural Network based on LSTM. Our network consists of three layers, each one of 30 neurons, except for the first one, with 288 neurons equal to the number of registries used as the previous history to feed our network. To each

layer follows a respective dropout regularization, the final two layers are fully connected to retrieve a single value for prediction. Further details about the configuration and design decisions can be found in Section 3, where we focus on a more complex problem. The network is fed, as told, with a moving window of N = 288 data points, i.e., one full day of observations. On the other hand, the outputs are selected to predict the price of the next data point in the sequence. The predictor optimizer is trained to minimize both the root mean square (RMSE) and the mean absolute error (MAE).

As we briefly comment in the last section, we also scale the values that feed our network, as it is known to accelerate the learning process of machine learning algorithms (as commented in [**Li et al., 2017**]). Since we just need to scale the scalar value of a prediction, we use a simple MinMax function, applied over the total number of observations. This function is defined in equation 2.2.

$$X_{scaled} = \frac{X_{original} - X_{min}}{X_{max} - X_{min}} \tag{2.2}$$

where X_{min} and X_{max} are the minimum and maximum value of the range scale, respectively. Clearly, this function maps the original values to the range [0,1]. With these considerations, we proceed to train our model.

Figure 2.3 shows the quality of this simple regression. As we see, our model is able to predict the price of the next point with great accuracy, both in training (yellow line) and testing (orange line) phases, achieving a score of 9.91 of RMSE in the training set on average, and 10.87 of RMSE in the testing set, after recovering the scale. While the predictive power of this simple network seems to be great, the main problems come when we try out to simulate the price evolution over time, starting after the test set validation (red line in figure 2.3).

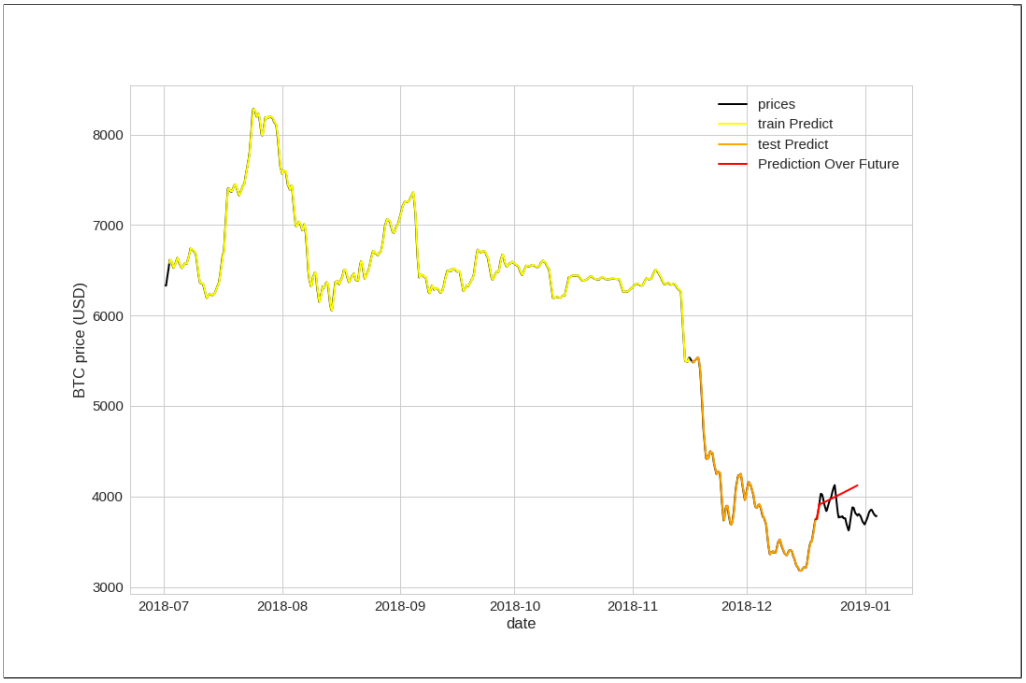


FIGURE 2.3. Network predictions after training. The black line represents the moving average measure introduced in the last subsection, the yellow line the predicted value on the training part of the data, and the orange line the prediction on the test set. The part with the red line is zoomed-in and explained in Figure 2.4.

2.3. Problems studied in this work.

As we have seen, the idea of predicting the next point of a time series is a simple task for a neural network, even for a simple one as presented in section 3. Yellow and orange lines show great fit with respect to the original moving averaged series and thus, we prefer to focus on two more difficult problems.

The first problem we study is to predict the price of bitcoin in the next day in the future (that is, to predict the price after 288 data points). This objective comes as a natural extension of the regression problem we have just defined, and it is both, realistic and useful. To achieve this, we just have to redefine the output ”labels” for the regression task. In the following section (3) of this work, we will refer to this objective as the *next-day price* prediction problem.

The second problem we study has more to do with our intention of using external data to predict prices. The motivation comes from Figure 2.3. If we see the orange line, every orange point takes as input the moving window of the latest 288 real data points. But what would happen if we start feeding the network with its own predicted prices? How well can we simulate the moving curves of bitcoin prices? We refer to this problem as the *Price Simulation* problem.

To understand how changeling can the price simulation task be, we present in figure 2.4 a closer view of the last part of figure 2.3.

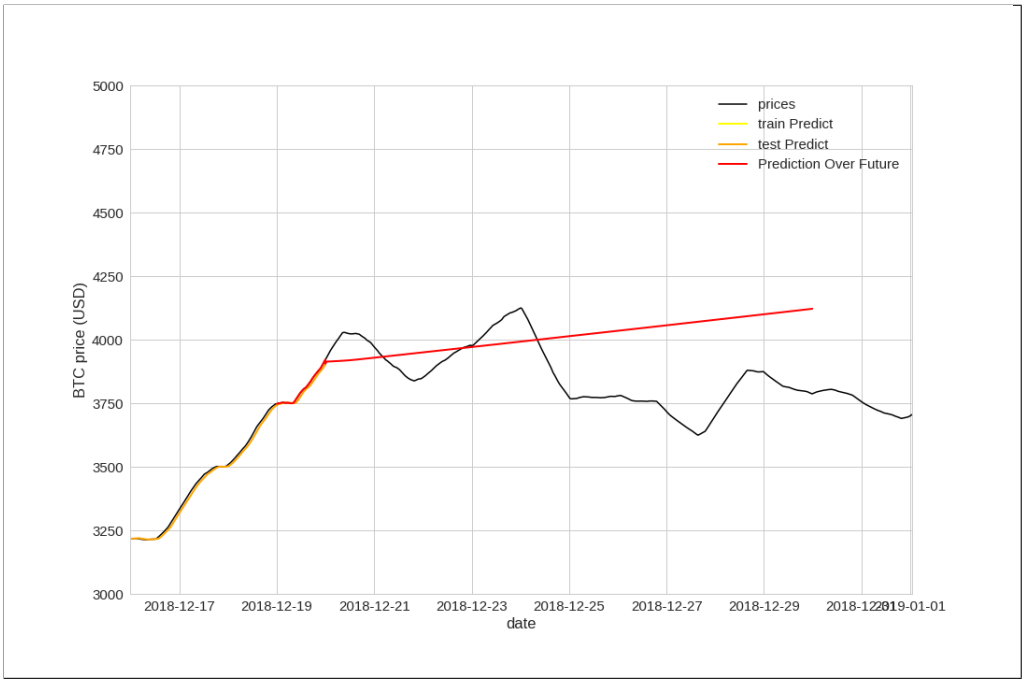


FIGURE 2.4. Simulation of Bitcoin Price. As the red curve shows, once we start feeding the network's own prediction as data, we immediately lose generality and prediction power, and we get quickly stuck in an average value.

As we said, the orange line represents the predictions made over our test set, which is well behaved when we feed our network with the recent history of variations. On the other hand, the red line represents the price simulation which is generated from the same model described in the previous section but now has been fed with already predicted data. The performance in this task is poorer than our previous results and clearly, these results

follow strongly some instantaneous momentum at the start of the simulation. Clearly, one expects that this price simulation cannot be carried away with price data only, probably the last resulting series will continue growing without awareness of changes of the real price as high-raises and low-drops are probably related to external phenomena.

Selecting which information, is appropriate to generate prediction is not evident, as a relatively new valuable asset in the market is not obvious which dependencies are the important ones. A previous study from 2012 [**Mehmet Levent, 2018**] studied the existence of dependencies between Bitcoin's prices with respect to the price of different valuable assets as gold, tangible currencies, and some market indicators. The authors worked over a causality test frame, where variations overtime series are made through the analysis of series decomposition, specifically, they separate the original series in a summation of positive and negative "shocks". Under this definition, any value at any time can be calculated as a cumulative function of these variations. The results of these tests revealed that the null hypothesis (not the existence of correlations between these "shock series") cannot be rejected with the available data, in most of the cases they do not find evidence of strong dependencies except for one market indicator: S&P 500 Index. Although the statistic test is good enough to rule out independence, the dependency's direction is not as expected, the authors discovered an inverse relation, where Bitcoin's price variations appear to have an influence on the investor's decision over the S&P 500 Index's market value. They are very clear when say they cannot settle evidence of positive causality over Bitcoin's price variations under the studied variables, revealing a changeling task and a necessary turn in the direction of analysis carried out.

While recent literature has not been to establish a strong relationship with the variations of other valuable assets in different investment markets, the results presented in [**Matta et al., 2015**] are promising in discovering some relation that links bitcoin's price with Google searches. The relation found is quite surprising because there has not been such currency in existence that showed this behavior before. The correlation of virtual currencies with virtual indicators seems a sensible thing to consider and an interesting field of study. Although is not clear what indicators should be considered in our analysis,

we explore deep in this work the hypothesis that the bitcoin's price is affected by the web footprint of popular actors in the international crypto market.

Preliminary exploration of the data told us that the amount of digital material, i.e., the number of messages retrieved from the influential users in Twitter media over time, does not appear to be strongly correlated with bitcoin's price. A fast check through Pearson correlation calculation reveals a slightly negative correlation by a coefficient of -0.243, the significance test of the null hypothesis for this statistical measure is 2.1%, so is clear that there is an inverse relationship that we cannot reject, but that is not what we want to find. Figure 2.5 analyzes the time-lagged cross-correlation between both series.

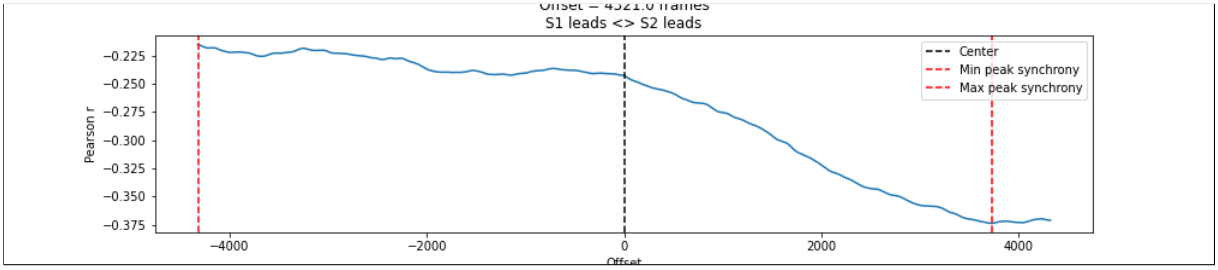


FIGURE 2.5. Time lagged cross Pearson correlation between Bitcoin's price and the number of messages retrieved from influential users in Twitter talking about Bitcoin. The second series is temporally lagged in time after and before the price variations. The numbers on the x-axis represent the number of steps in which the second series is moved, each step represents a difference of 5 minutes. The vertical lines mark the zero-time lag (the black one) and values in which the correlation reaches the max and min value (red ones).

From this figure it is seen that by lagging the number of messages to the future, the previously detected inverse correlation starts to grow in the module in time, which should be interpreted as "variations in Bitcoin price have a consequence in the number of comments that people in media do", or more specifically, "if Bitcoin decrease/increase in price, people turn to talk more/less about it in time". So again, as well as conclusions from [**Mehmet Levent, 2018**], we are able to set an inverse causality relation between the number of influential users' comments in media and Bitcoin. If we want to continue exploring our hypothesis, we must introduce a new kind of analysis. This is where Natural Language Processing (NLP) comes in handy, as is discussed in section 4.

3. NEXT-DAY PRICE PREDICTION AND PRICE SIMULATION BY USING PRICE DATA ITSELF

In this section, it is introduced the model design, created for studying both: the next-day price prediction task and the price simulation problem. We recall that this architecture is the same as the one used to generate the curves presented in the previous chapter. Again, we are still training our model with just price data alone but, unlike the last section, we will deepen in a more realistic and useful prediction task. More specifically, we want to answer the following two questions: First, can we get the same good results for the next-day price as we got when predicting the next data point in the price history? The second question has to do with price simulation. As we have mentioned, in this setting we do not expect good results on the simulation, but the question remains: how far we can push this simulation using only price data?

3.1. Creating a prediction model: Architecture behind predictions

Let us briefly describe the architecture of our network. As told, we are experimenting with Deep Neural Networks. Since our data are time series, choosing recurrent networks (RNN) makes sense in order to take advantage of the recent history of variations to predict a single value in the future. Among the different RNNs, we are specifically working with LSTMs networks, which are well known for their ability to exploit temporal connections and solving the gradient vanishing problem. Figure 3.1 depicts the main components of our model.

The introduced architecture consists of 2 LSTM recurrent layers, each consisting of 128 neurons. We drew these numbers as design hyper-parameters from our preliminary testing and through grid search iterations. After some time, we discover that such depth and hidden units were good enough for our purposes. The input layer receives the 288 data points representing a complete day of observations, then an embedding function transforms the sequence in just 24-time steps observation but now including some statistical information as the min, max, median, and mean of the price along the hour of data. Then,

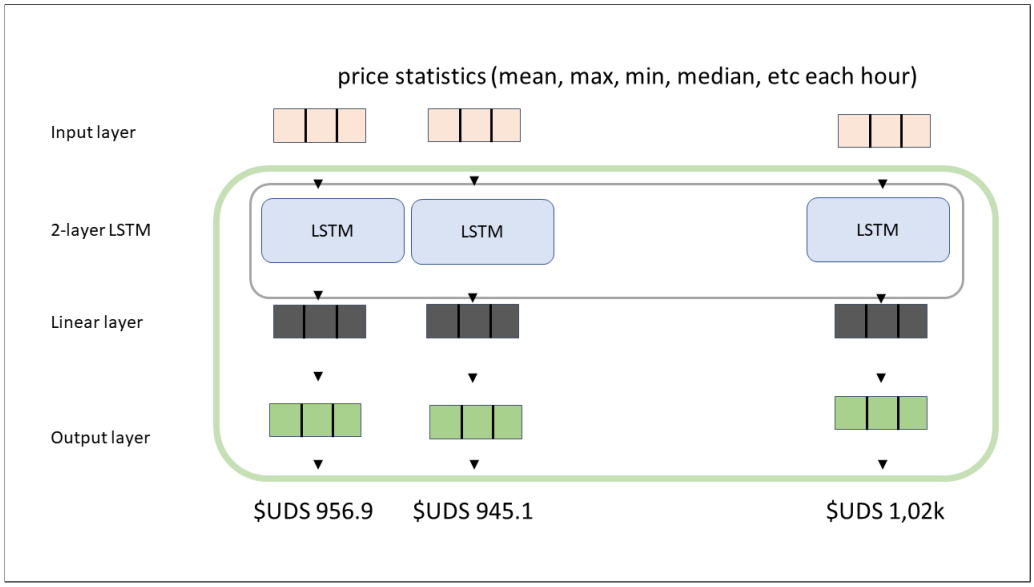


FIGURE 3.1. The architecture of our model is essentially a composition of multiple LSTM layers and a last feed forward layer to generate the output.

all this information is fed as sequential data. The output is then retrieved to be fed to a linear layer.

Following good practices, we initialized the first layer with random biases and weights. To avoid overfitting, every LSTM layer is followed by a Dropout regularization; the first layer has the biggest dropout probability and it decreases with each subsequent layer. Two fully connected layers are added, at last, to obtain in first place new latent factors of the variables, and second to generate a unique output so in every stage our neural network can be trained for prediction. When an output value is obtained then is compared with the target value which corresponds. The loss function is set as the quadratic difference between each value also known as the RMSE metric, thus can be then minimized through backpropagation. In every training step, we feed our network with a batch of 288 examples to ensure system generalization.

The introduction of Dropout in each layer has a double purpose. The first as mentioned is to achieve generalization so when the model receives new data, it is more robust and less likely to reproduce the same predictions like the ones done at the training set. The second reason comes from an important result described in [**Gal and Ghahramani,**

2016]. In the referred work the authors claim they had *"develop a new theoretical framework, casting dropout training in deep neural networks as approximate Bayesian inference in deep Gaussian processes"*. The results are very impressive as well as we can now have a framework in which we can quantify our model's uncertainty through the use of dropout. That is, in each realization, we work with a slightly different network that is affected by a different noise (caused by the dropout itself), so we could now quantify how robust is our model to this noise. The authors showed that training a network over the dropout objective, in effect, minimizes the Kullback–Leibler divergence between an approximate distribution and the posterior of a deep Gaussian process similarly as Bayesian's methods do, which are mathematically designed to do.

To test our model design, we are comparing the score of the RMSE error of an XGBoost model, trained for predicting the next day's price.

3.2. The relevance on scaling correctly

From the previously introduced model, we proceed just as in Section 2 and start training our network for the new task of predicting price one day next in the future. To do so, we scale the whole data with the MinMax scaling function as described before and then fed the embedded data to the network with measures of 288 registries at each training batch instance. Instantly we can see in figure 3.2 how now our predictions are surprisingly worse. The yellow line, representing our prediction over training data, is stuck in a constant value, which is close to the mean of the series over the considered time span. Moreover, this value is projected in the test set as the unique possible value learned by the model. In short, the trained model is practically useless.

The reason for this setback is that the prices of 1 day in the future have a much higher variation than the ones that could be seen in consecutive data points. Our scaling function just projected our values to the range [0,1] but preserves the proportion in the differences, this means that the network could not properly distinguish these changes. In this point

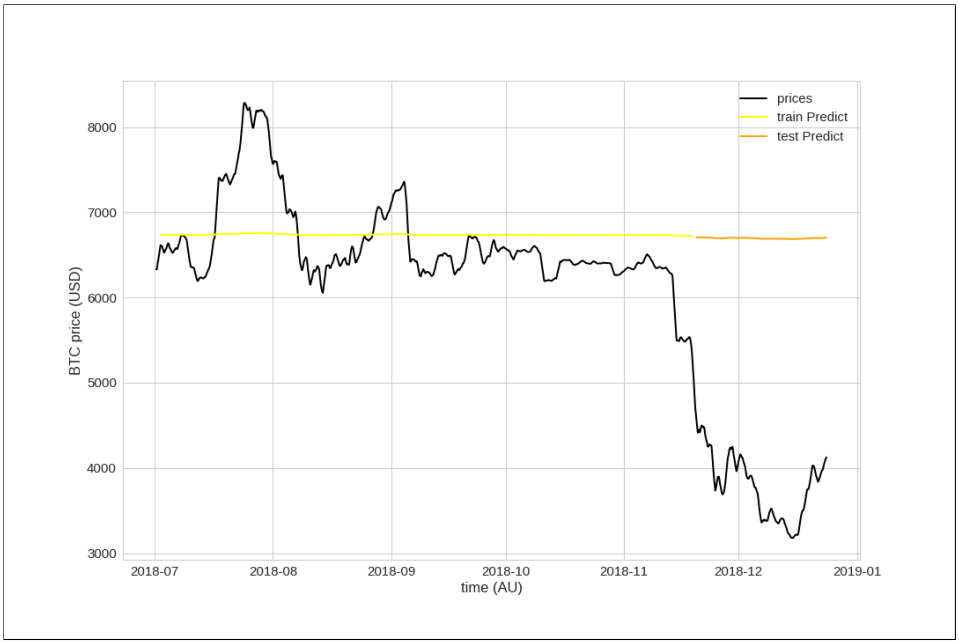


FIGURE 3.2. Next-day price prediction when all data is scaled as in Section 2. The curve is markedly displaced to an average value, not being able to recover the shape of the prediction curve and being flattened on all its extension

we have to emphasize that the main reason for applying a scaling function over the observed data is to optimize the gradient descent process during the training phase, it is a well-known practice that scaling data generate a faster push to reach a local minimum. In this sense, the issue is the following: If we scale all data points first, then most probably all data points of a 24 hours window would be mapped to very similar values. This would not be an issue for predicting values in the same range, but when forced to predict values for the next day, which are mapped to other ranges by the scaling, the network was no longer working how it should. It is probably that in this case, the variations in a window of 24 hours are infeasible to learn, so the network progressively understands that predicting a central value as the mean, is a good way of reducing loss.

To solve the referred problem, we use instead of a scaling function that scales every 24 hours window separately, i.e., each observation of 288 points that feed our network is scaled with a particular MinMax scale function. Of course, there is a new problem with

this solution: if the prices in a day vary in an (a, b) interval and the price in the next day is outside this interval, what should we expect the network to predict? To cope with this problem we alter the MinMax bounds so that the lowest point (x_{min}) fed into the network does not get mapped to value 0 but to its equivalent value in scale as a distance from the new lower bound in dataset $0.8 \times X_{min}$ and likewise for the higher point $1.20 \times X_{max}$, in this way we are trusting in the *naive* assumption that the values we want to predict reside in a new range ($0.8 \times X_{min}$, $1.20 \times X_{max}$) and while these weights are learned to cover a high range of variations, there is a considerable number of extreme values that our model will have to treat as outliers. These design decisions are mainly supported by our results. Figure 3.3 shows the performance of our model after these modifications.

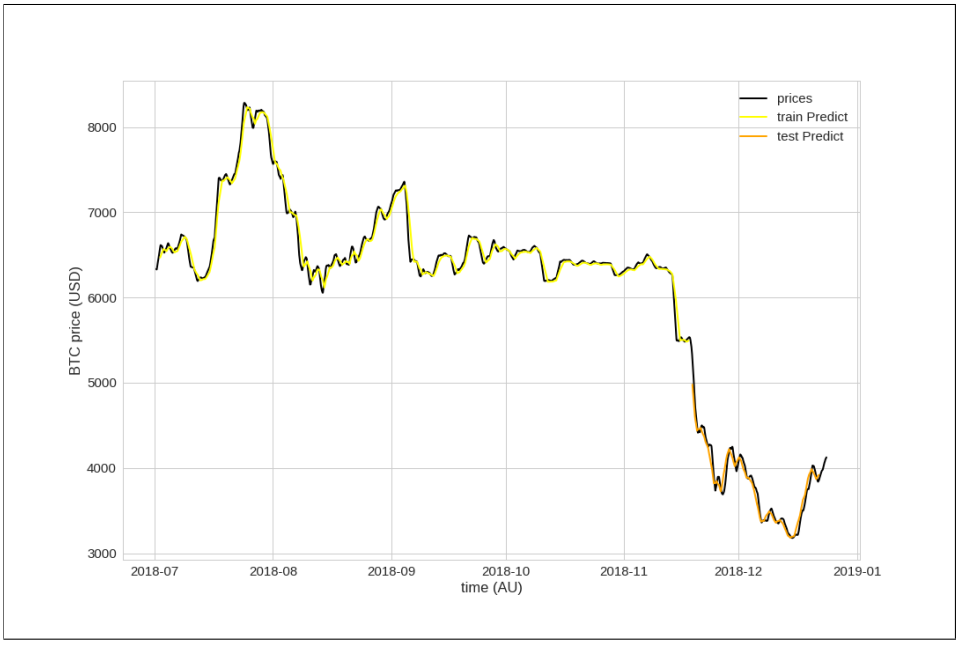


FIGURE 3.3. Next-day price prediction when data is particularly scaled in a per-day basis. Results are much more well placed than the ones obtained with a general scaling function.

As we see, the performance of the model is recovered with no modifications needed at the architecture design but only at the scaling phase, this reveals the importance of data engineering even when we rely on self-trainable systems as neural networks. We must emphasize that although the selected solution is naive, it is also simple and more importantly

good enough for the network's learning. There are of course more sophisticated ways of improving the same results: For example, one can consider more complex networks, as in [**Zhou et al., 2018**] where the authors develop a generative adversarial network to create a prediction model in two parts, one called Generator, who is also called as an 'adversary' that generate samples from a real-time series and tries to compete with a Discriminator who determinates if the samples are real or were generated by the adversary. The two training phases create a robust model that is able to discriminate a precise range in which the possible values move and adjust their predictions to it. Another possible solution for this problem could be considering other scaling functions among the several defined ones or perhaps creating a simile to the technique introduced in [**Zhong et al., 2019**]. In this study, authors use *attention* in the training process so that the network is aware of some important relations for the prediction task, a possibility would be to use this idea to relieve the scale-boundaries learning to the network itself. However, since the goal of this study is to later combine these models with Twitter data, we choose to move onto the next problem and work with the introduced daily scale, as performs good enough.

3.3. Price Simulation using per-day scaling

Since we have seen that per-day scaling gives us a better-trained model for the next-day problem, it also makes sense to update our price-simulation network so that the training is likewise done with this other scaling. Results are presented in Figures 3.4 and 3.5. Again, the first image presents the results as in 3.3 with an extra red line for the price simulation task, complementary the second image shows a closer view to last results.

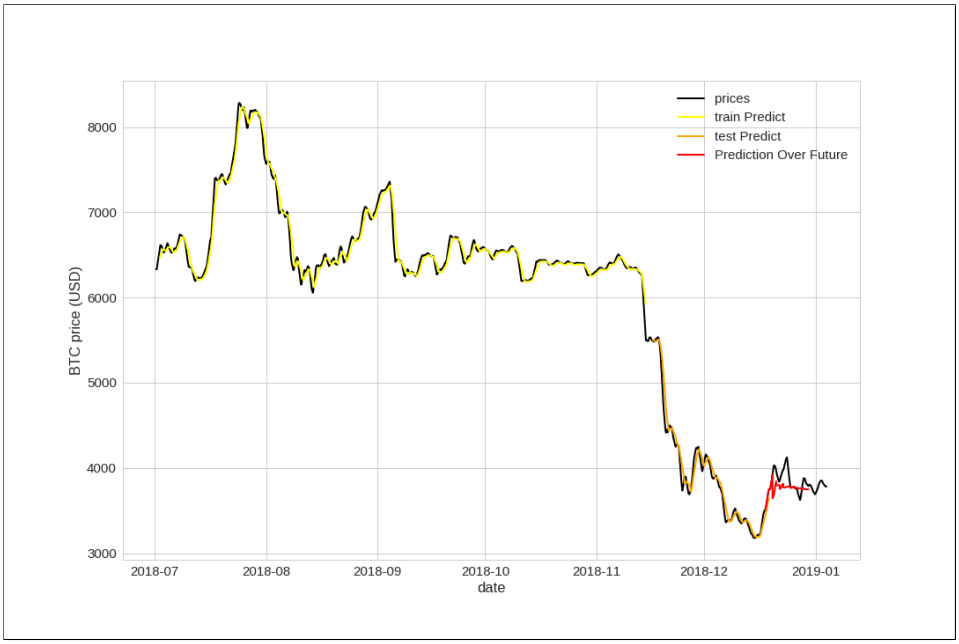


FIGURE 3.4. Price Simulation with per-day scaling. Test data is depicted on the yellow line while simulation is on the red line.

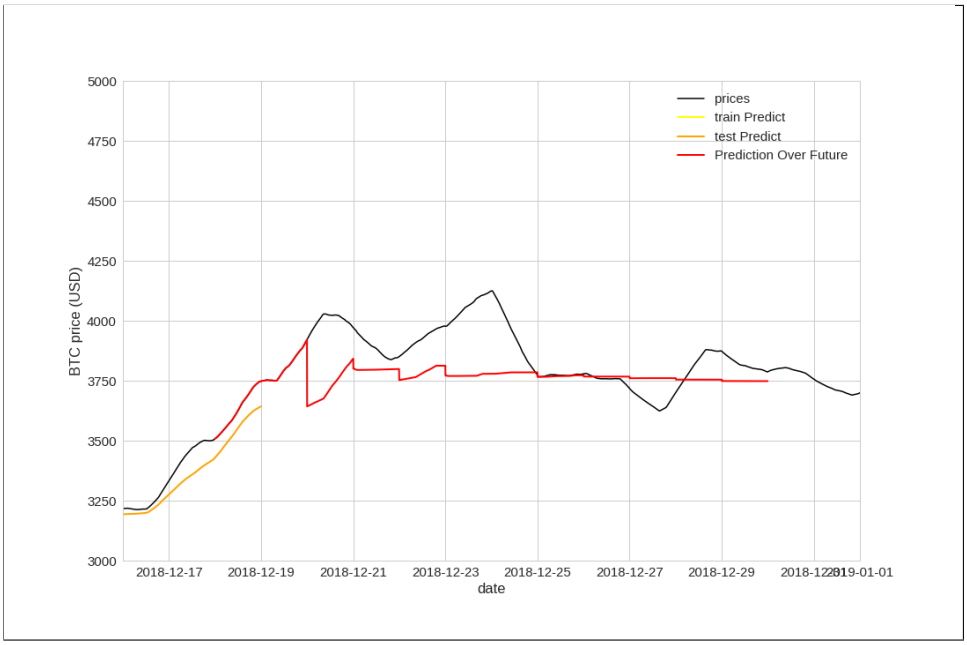


FIGURE 3.5. Zoom of the model's performance while simulating price. This model now manages to predict a decrease in price, but the prediction is still too coarse.

The new scaling gives our model the ability to identify slight trends over price data (red line in Figure 3.5), however, we are still not able to detect the subsequent rise in price by the end of the test period (data left out of the training and validation process): at this point, the model has detected that prices tend to drop over time, and, unless we provide another form of external stimulus to the model, it will never have enough information to change its decision direction. Our hypothesis is that Twitter data may well be the external stimulus that we are looking for, and that "good" or "bad" tweets will give the network enough information to change the trend of what it is predicting. At this point, some natural question emerges: ¿How to recognize who is influential in Twitter media? ¿How use to define what is a good or bad tweet? ¿Which is the best way to use this information in making our network aware of changes in price? This is the focus of the next section where we settle down a framework in which we size up the exactitude of the introduced hypothesis.

4. OPINION AS A QUANTIFIABLE MEASURE

The idea that the price of assets is affected by certain influential spokespersons is not new and has already been studied when analyzing prices of other types of assets (see e.g., [**Zhang et al., 2018b**, **Zhang et al., 2018a**]). Our belief that Twitter's data is a good external source that should help in predicting Bitcoin data comes from gathering opinions of people working in the crypto market itself [**Rada, 2018b**].

4.1. Extracting Twitter opinion data

There is a large body of work on how to compute the most central nodes in a social network, and to the best of our knowledge, there is no agreement on a single method that fits all uses. Moreover, most well-known centrality measures (PageRank, betweenness centrality, degree centrality, etc.) require full access to the network to be computed, which is not feasible in the case of Twitter. Instead, we use the following algorithm as an approximation for computing users with higher page rank that are related to cryptocurrencies. This algorithm is inspired by [**Rada, 2018a**]:

(i) We start with a seed of 10 users that are known to be influential.

(ii) In each iteration, we create a list of users corresponding to all the people followed by the users in the seed.

(iii) We order the list with the most followed users first, take the 10 most followed that are not already in the seed, and adding them to the seed.

We selected a total of 135 influential users after 10 iterations of the algorithm. Then, we extract all tweets by these users between July 2018 and January 2019, but only store those tweets referring to one of the following terms: *bitcoin*, *btc* or *cryptocurrency* as well as their grammatical variations. This method gave us a total of 9146 tweets, which represent an average of 43.5 tweets per day across the seven months considered. Note that we could have obtained more tweets by simply enlarging our list of crypto influencers. However, there is a danger in doing this: with more users in our list, the tweets made by

the real influential people would be diluted by tweets from regular people, which we do not trust to affect the price in the same way.

In this study, we decided to not include retweets, because a lot of this information could correspond to advertising or propaganda and we are more interested in the diversity of the speech to be analyzed against price.

While this data represents a good material for use in a prediction model, is not obvious how to represent this information, as an opinion can be qualified in many ways. Within them, the simplest way appears to characterize opinions as incentives of Bitcoin's price behavior, the next section will discuss this extensively. A big problem to deal with is that gathered data is raw and not labeled at all.

4.2. Characterizing opinion from influential users.

When we thought about what is information, we would like an extract from an opinion and more specifically in our case, from a tweet; we realized this is not obvious to achieve. The general idea is to extract the sense of an opinion referring to Bitcoin, however, the objectives we pursue are even more ambitious and require specific analysis. More than getting an idea about Bitcoin, we want to recognize if the information expressed by tweets is going to act as incentives or not, but precisely for the behavior of the price itself and the Bitcoin's market. At this point, is important to comprehend that this information is very specific, so is not easy to accomplish with any algorithm of the area as we will see.

Among the many NLP's applications, there are two that match in somehow our ideas of getting a valuation starting from an idea or opinion. Both *Sentiment analysis*, as well as *Stance detection*, represent applications of Natural Language in developing comprehension through the introduction of polar measures, but with the deep difference between them [**Krejzl et al., 2017**]. The first one, Sentiment analysis, is a fruitful area of NLP applications that consists of the classification of text according to the *mood* that emerges from itself, in short, is a measure of positiveness and negativeness about something. According to this, we can argue that the sentiment measure is strongly related to *words* used in the

message's construction. Concepts as happiness, support, negation, anger, and so on, tell us immediately information about the sentiment that people have about something. Nevertheless, there are as many algorithms as ways to compute this measure of sentiment. Some algorithms ([**PappuRajan and Victor, 2014**], [**El Alaoui et al., 2018**], [**Baccianella et al., 2010**]) generate a score based justly on the occurrence and co-occurrence of words in a pre-designed dictionary of positive/negative ones. Other algorithms generate training of more sophisticated classification algorithms, where sometimes the classification rules are learned directly from data [**Mohammad et al., 2013**]. Experimental results are varied but they suggest that is a naive assumption that isolated words by themselves can represent the sense of the speech. The fact is that semantics relations of language are difficult to represent and learn, but also are important to completely understand what has been said. So, an ideal approximation must be aware of these throwbacks.

While sentiment approximation and their algorithms seem interesting, unfortunately, we cannot use standard sentiment analysis tools and expect that they would be useful to Bitcoin's world and our purposes. For instance, some texts regarding Bitcoin found in our data set say

I. *Banks in India are now saying that they will now close costumer's accounts if they deal in cryptocurrencies*

II. *For this reason, governments worldwide are having a tough time adjusting to the reality of a world with #Bitcoin: They are losing their tight grip on #Money, which historically, has been in their full control. No longer! Money can now truly belong to the people. #FreedomOfMoney*

Box 1. Tweets about Bitcoin, examples.

As can be read from the first example, terrible news for Bitcoin's market is announced, however, the chosen language is very neutral. By its side, the second one is supporting Bitcoin as an alternative to the traditional transaction method dominated by cash and credit, both classically linked to banks and governments; the language used in this case is a more

negative one, expressions like *tough time*, *losing* and *no longer* are evidencing some information with clear purposes: to criticize banks and governments as well as their relations with money's markets. This second tweet is one of the typical examples where a polar measure based on sentiment is not in tune with the meaning we want to keep: determining a sense of support or disagreement regarding Bitcoin's market behavior. Those terms are in some way closer to a *stance* concept than the circumstantial sentiment that an opinion gives. The *posture detection* task is about determining whether, who gives an opinion, adopts a position (a.k.a. stance) of supporting, agreeing, defending, or simply liking something, which agrees to detect positive stimulus when the opinion comes from influential users. In similar terms, the negative stimulus, so an indication that price should be decreasing, must be detected from opinions from influential users that do not see optimistically or disagreeing with Bitcoin's market behavior. Of course, a good classifier should also understand the scenario in which the opinion does not express a stance or is not even clear, in which case should be classified as neutral. This encourages us to consider stance as the embedding unit to carry out our study.

Most of the state-of-the-art solutions for stance prediction are usually task-specific classifiers based on neural networks and as is well known, their success relies on the huge number of labeled examples that the algorithm has access to. In our scenario, this does not seem reachable. First, because there are no public datasets available for this task. But more importantly, the labeling of tweets by humans would come at a considerable cost, since this task must be done by people knowledgeable both with economics and with the bitcoin ecosystem. So, using large training datasets is outside the scope of this work. Despite this unfortunate scenario, some recent results could be very useful for us. In the papers [**Howard and Ruder, 2018**], [**Devlin et al., 2019**], [**Radford et al., 2019**] (2017-2018) authors introduce novel techniques that refer about the possibility of **fine-tuning** language models for a specific task with even just 100 labeled examples to train [**Howard and Ruder, 2018**]. This idea seems very interesting because the language models usually found in the literature are always trained in a huge corpus of data, so have learned a huge number of rules and language relations that can be beneficial in re-using for another task.

This particular scenario of lacking data and new results in fine-tuning models suggest a novel methodology for building our task-specific classifier. First, as we will discover, the fine-tuning process will still need some labeled-by-experts examples, to adapt to their knowledge for new tasks and contexts; so, in the beginning, we have to develop the biggest as possible training dataset using experts knowledgeable in the problem. Then, the next step should be fine-tuning some of the state-of-the-art NLP models for our task-specific purposes by using transferring learning techniques. Finally, by getting stance detection results for our list of messages from influential users, we could easily use them as new inputs of the price prediction model. To this end, the next sections present how we carry out this methodology. In section 4.3 we discuss two of the most recent results in transfer learning's area to be applied in the most diverse NLP tasks. Section 4.4 introduces our experience in creating the mentioned Bitcoin-stance-labeled Database for our specific fine-tune purposes. Finally, in section 4.5 we discuss our results in diverse models, tasks, and algorithms.

4.3. Fine-tuning language models for stance detection: ULMFIT & BERT

The models that assign probabilities to sequences of words are called language models. The simplest model to assign these probabilities to sentences and sequences of words is the n-gram. Today there are much more complex characterizations that are the result of analyzing a large corpus of documents.

The idea of fine-tuning learning models (a.k.a. inductive transfer learning) comes from the area of Computational Vision (a.k.a. CV). In these investigations, authors trained convolutional neural networks for image recognition and discovered differences within the information learned by different layers. The first one learned more general information about images than the final ones. Indeed, when they visualized the activation of the neurons over the first layers, they saw shapes, colors, and shadows; as well that none of them by itself solely could represent any of the known data. So, these authors had an idea: keep the first trained layers and replace the last ones that are mandated to make the classification. The results were that after just a few numbers of extra training epochs to make the

network adapted to its new task, they were able to recover their accuracy in classification in a completely different task.

Nowadays, fine-tuning is widely used and very popular for image recognition. However, despite this was introduced for the first time in 2011 ([**Tian et al., 2011**]) this application did not see the light in NLP until 2018-2019, where many ideas ([**Howard and Ruder, 2018**], [**Devlin et al., 2019**], [**Radford et al., 2019**]) came out with promising results. Until that moment, most of the state-of-the-art results in NLP were models trained from scratch, and the inductive transfer learning was relegated to the generation of pre-trained word embeddings [**Almeida and Xexéo, 2019**], in which case the transfer techniques only target the first model's layer, still requiring task-specific modifications and training from scratch. The impossibility of not being able to use fine-tuning directly was because most of these applications do not share the same model architectures used in CV, and also that semantics and syntax are not as simple structures for language as shapes and shadows are for images. Particularly, Twitter data is a very interesting use case for transfer learning, mainly because the typical language syntax seen in Tweets is quite different from that used to train typical language models.

For purposes of this work, we will focus our discussion on two fine-tuning proposals: ULMFiT and BERT.

4.3.1. ULMFiT: a recipe for correctly fine-tune your model.

ULMFiT was presented for the first time in [**Howard and Ruder, 2018**], where the authors define the acronym that stands for Universal Language Model Fine-tuning. In this publication authors more than introducing embeddings, introduce a standard process to effectively fine-tune any language model, for the many desired tasks.

The fine-tuning process for a novel classification task is based on a 3-stage methodology.

1. General-domain Language Model pre-training: When we talk about a language model, we refer to any learning model used for the simplest task in NLP: language modeling. The concept of *simple* must be understood in the sense that any source of text works for these purposes, however, this task could be very complex. An example of language modeling is training a model to recognize the following more probable word in a sequence. The idea behind this is training any model for a task as general as possible to capture many facets of the language that can be relevant for downstream tasks, such as long-term dependencies [**Linzen et al., 2016**], hierarchical relations [**Gulordava et al., 2018**], and sentiment [**Radford et al., 2017**]. These relations are learned by our model at their neuron's weights in what we call a *general-domain language model*. In the paper, the authors tested ULMFiT with a pre-trained model generated using an AWD-LSTM model such as the introduced in [**Merity et al., 2017**] which consist of very simple architecture, but has demonstrated to be good behaved for the language modeling task. This architecture consists of a 3-deep layer recurrent neural network based in LSTM and trained over 28,595 pre-processed Wikipedia articles, totaling 103 million words. What comes next, it's about adapting those weights to a novel language's context (Twitter media) and task (Stance classification).

2. Target task Language Model fine-tuning: The second step is telling the model that the new corpus comes from a different distribution of words and a specific context, things we called *the idiosyncrasy* of the data. To accomplish this, the authors introduce some interesting techniques to update the language model without suffering from forgetting the already learned parameters. The main two are:

- **Discriminative Fine-tuning**: From the ULMFiT paper: "As different layers capture different types of information [**Yosinski et al., 2014**], they should be fine-tuned to different extents" and this is done in ULMFiT after extensive empirical testing and implementation updates in discovering specific hyper-parameters. In this sense, rather than change each layer manually, they adapt a multiplier that influences the learning rate for

each of them, so the weight modification rate is not equal for each layer. The value found was 2.6 and the varying for this is of the form,

$$\eta^l = \frac{\eta^{l-1}}{2.6} \tag{4.1}$$

where η^l is the learning rate of the l-th layer of the model language. This modification traduces in changes to the SGD [**Poggio et al., 2011**] update rule, where the learning rates are involved as follow,

$$\theta_t^l = \theta_{t-1}^l - \eta^l \cdot \nabla_{\theta^l} J(\theta) \tag{4.2}$$

where the θ_t^l are the weights of the $l - th$ layer at the $t - th$ epoch and $\nabla_{\theta^l} J(\theta)$ is the currently gradient calculated by backpropagation.

- **1-cycle learning rate policy:** In the fine-tuning stage, a *1-cycle learning rate* policy is also applied, idea that comes originally from this paper [**Smith, 2018**].

  This step is now about varying η across different epochs, so this will be η_t on each epoch. The version of the policy used by ULMFiT it is a modification of the cyclical learning rate policy, which has been around in the academic community for a long time. The novel here is that their 1-cycle policy allows a large initial learning rate (LR_{max} = 10^{-2}, for example), decreasing it by several orders of magnitude just at the last epoch steps. The reason for this policy is adapting the model parameters to quickly converge to a suitable region of the task-specific parameter space at the beginning of training, then just refine its parameters in the last epoch steps. We will call as *slanted triangular learning rate* to this policy. The learning rate's schedule exhibited by ULMFiT is shown in figure 4.1

  According to the authors, this specific policy seems to provide greater final accuracy.

3. Target task *classifier* fine-tuning: Once we save the updated weights from the language model fine-tuning step, we can fine-tune the classifier (which is the final layer of

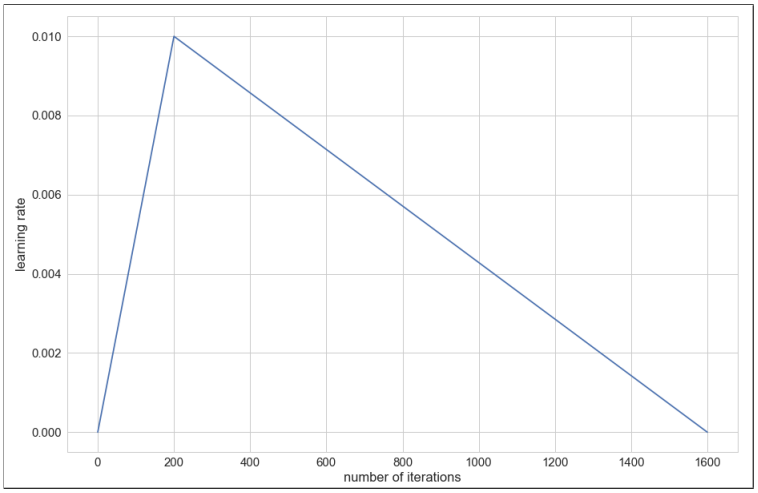


FIGURE 4.1. Slanted triangular learning rate policy for fine-tuning ULMFIT.

the learning model), this is accomplished by using the techniques described above and the introduction of two new ones to perform task-specific class prediction.

- **Gradual unfreezing**: This refers to a new policy in the way the network is trained. Rather than training all the layers at once during classification, the first layers are ”frozen” and just the last one is fine-tuned first, in the next epoch, the two last layers (the last followed by the next layer before it) will be trained and the remaining ones will be continuing to be ”frozen”, and so on during the following epochs. This avoids the phenomenon known as *catastrophic forgetting* which is something that happens when the network changes its weights too aggressively to achieve good classification scores, not relying on the pre-learning phase.
- **Concatenated pooling**: Because an input text can consist of hundreds or thousands of words, information might get lost if we only consider the last hidden state of the neural network. Hence, the hidden state at the last time step, h_T is concatenated with both the max-pooled (the max activation scores within the states) and mean-pooled (an average activation score from all states) representation of the hidden states over as many time steps as can fit in GPU memory.

$$h_C = [h_T, maxpool(H), meanpool(H)] \tag{4.3}$$

Where H is the vector of all hidden states.

The coupling of these parts and all of their components allowed the authors to reach state-of-the-art results for different tasks and data with just 100 labeled examples for the classifier fine-tuned, after applying a language modeling fine-tuning with 50,000 unlabeled examples. In this work, we will use the same language model as the one used in the original job.

4.3.2. BERT: a novel attention-based architecture, easy to adapt to contexts and tasks.

Contrary to ULMFiT who relies on a set of instructions and some empirically discovered hyper-parameters to fine-tune a given (any) language model; BERT, which stands for Bidirectional Encoder Representations from Transformers, is a language model itself. BERT was designed to learn deep bidirectional representations from unlabeled text, this means that in every moment, BERT computes weights by reading text forward and backward. Those representations have demonstrated learning properly many language particularities. As a result, the BERT model can be fine-tuned by adding just one output layer to create state-of-the-art models for a wide range of tasks, without substantial task-specific architecture modifications.

From an architectural point of view, BERT is a trained Transformer Encoder, but formed by many copies stacked as a block, more or less simply, BERT's architecture is presented in 4.2. The term *Transformer* refers to the self-attention system developed by Ashish Vaswani et. al in the paper [**Vaswani et al., 2017**]

Figure 4.2 shows how the model takes the words as input (usually represented by some embedding) and after passes through 12 Transformer's Encoders layers, BERT generates representations learned in the process. While the architecture seems very simple, there is a lot of mathematics happening there. To understand this totally, we have to see t what a Transformer Encoder is, this is shown in figure 4.3.

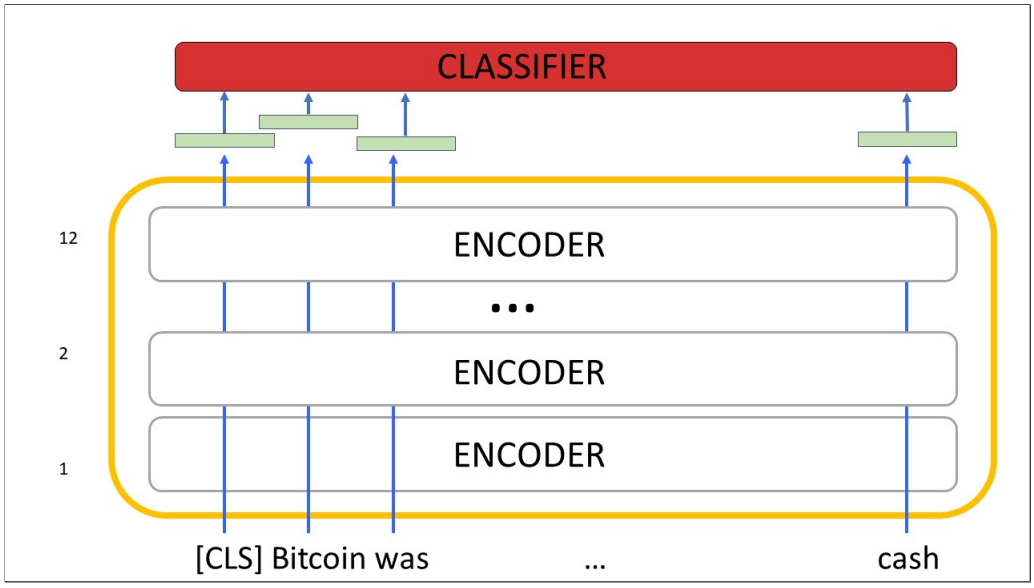


FIGURE 4.2. BERT architecture, as discussed in [**Devlin et al., 2019**]

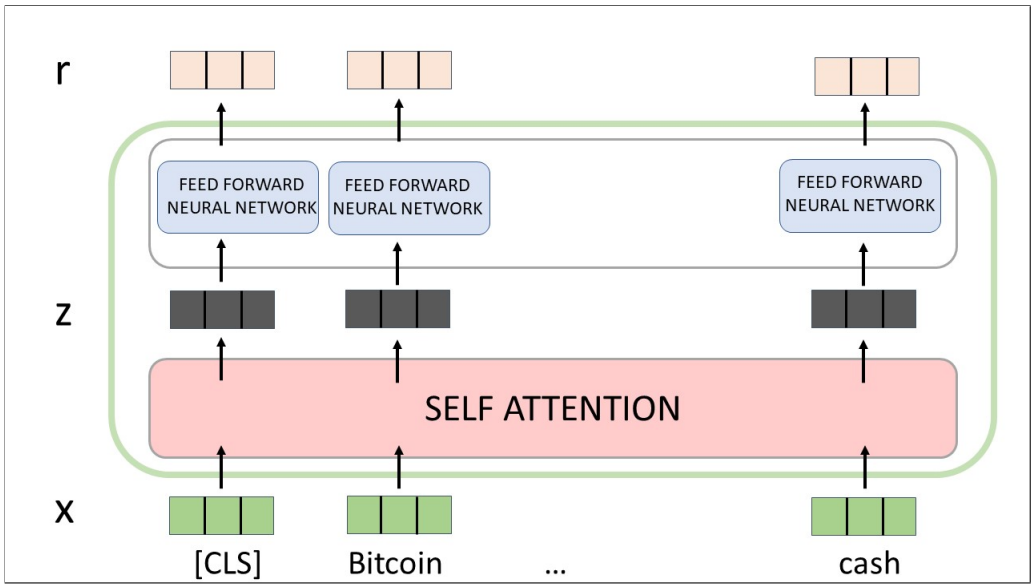


FIGURE 4.3. Transformer base architecture: Encoder.

Every Encoder is formed by two layers, the first one calculates self-attention scores while the second one is formed by just a feed-forward network. Because the last layer is a quite known architecture, our discussion will be centered on understanding the attention's concept. To this end, let's analyze again the first example presented in Box 1,

*Banks in India are now saying that **they** will now close costumer's accounts if **they** deal in cryptocurrencies*

in this tweet two subjects are involved: Banks in India and costumers, nevertheless the word **they** is used indiscriminately for them; an attentive mechanism is expected to recognize this type of differences and associate each **they** with their respective entity. In BERT this is done in a very smart way, in simple words, if you are reading the first **they** from the last example, you would expect a high score resulting from an attention metric with words *Banks* and probably *India*, the solution created in BERT introduces useful self-learned abstractions to calculate these scores. Thus, for each word a Query vector ($\vec{q}$), a Key vector ($\vec{k}$), and a Value vector ($\vec{v}$) will be created, then computed attention between two words (i, j) will be a score between their query $\mathbf{q}_i$ and key $\mathbf{k}_j$ vectors,

$$z = SoftMax(\frac{q_i \cdot k_j}{\sqrt{d_k}})v_i \tag{4.4}$$

where z is the output shown in figure 4.2, the term $\sqrt{d_k}$ is the square root of the dimension of the key vectors, which along with the SoftMax function lead to having more stable gradients by normalizing the score. Finally, the intuition behind the value vector is to assign a relevance score to maintain intact values of the word/s we want to focus on and drown out irrelevant words. The final step is, to sum up, the weighted value vectors. The novel in BERT is that these representations are calculated by the model itself through the whole process.

The unique representations created by BERT are so powerful that carry on a lot of information, so the fine-tuning process concentrates uniquely on modifying/adding the last classifier layer to the task-specific purposes.

While BERT's seems to be more promising in their results (by achieving the best performance in at least 8 of the most used NLP's test datasets) authors are cautious in mentioning that fine-tuning is unstable on small datasets, being the smallest used by them

formed from 2,500 labeled examples. This number while is not unreachable is way more expensive to achieve than the number needed for ULMFiT

4.4. A labeling session

Mining opinions from the internet is not a difficult task. Media like Twitter are huge sources of information where people share their thoughts for free. However, there are two important limitations, the first is related to the maximum number of tweets that can be retrieved from the media in a single query. Twitter's API service allows consulting up to 3,200 tweets back-in-time on the user timeline, starting from the date at the moment of the consulting and of course, the number reduces when you keep filtering over topics of interest. The second throwback comes from the rawness of data, which is hardly labeled for specific purposes like the one we are looking for.

Generating labeled data is expensive in most cases, you always have to be willing to invest in manual classification made by humans. Because of this, web pages for crowdsourcing and surveys have found recently a niche of success. However, these services are uncommonly carried out by experts' annotators in the topic. Our task is especially critical because, as commented, the pursued task is easily misunderstood when *sentiment* is considered instead of *stance regarding price*, in the same line, one can expect that a minimal knowledge in cryptocurrencies markets, as well as economics, may be needed to ensure complete comprehension of what influential user is saying about Bitcoin markets. For those reasons, before using a web crowdsourcing service, we adopted the decision of recreating a labeling session with undergraduate students from the Computer Science Department at Pontificia Universidad Católica de Chile. These people were especially interested in the world of cryptocurrencies when we interviewed and selected them, on the other hand, their formation at the Engineering School reinforce our confidence in their knowledge about economics and computing, things that added to their English skills turn them into ideal experts for the labeling task around the Bitcoin topic.

Our objectives for this session were modest, because we considered labeling as a job, paying to the students for their responses when our resources were limited. So, in this way, we wanted to generate a database made from at least, the minimum number of labeled examples to fine-tune a pre-trained language model. As we remember, this number was 100 for ULMFiT and more than 10 times this number for BERT. As the last one was too ambitious, we decided to at least obtain labels for 200 examples, drawing them randomly from all messages extracted from influential users.

The session was planned as follow:

- We decide to work with six students doing the same job, to reduce random error and also giving more relevance and trust to the classification.
- The 200 examples were divided into two sessions. In each one, every student had to put labels on 100 tweets. The idea was to prevent bad answers due to fatigue or mechanization of the task.
- The students were part of a little class about differences between concepts of Sentiment and Stances with real examples.
- They worked each day with a document containing the 100 tweets and were asked to classify them according to 4 labels: SUPPORT, AGAINST, NONE and I DO NOT KNOW/ I AM NOT SURE. The tweets answered with the last label were not considered in the following stages.
- To verify the attention and comprehension about what each tweet talked about, the students were asked also to identify the **target subject** of the opinion, the variable that was also used for control.
- No time limit was given to students to deliver their answers.
- An instructive was given to students to resume all this information, which is included in the Appendix A.

Any topic-specific session, like the one we created with these students, will still need a review of the agreement levels of the participants in their choices. To this end, we compared their answers. We starting using the control variable, the one we called the *target subject* of an opinion. This variable is about recognizing who the opinion is about. Specifically, we asked the students to choose one of these categories: the opinion goes to Bitcoin or somebody related with the market, the opinion goes to somebody or something that is not related to Bitcoin; the tweet does not express an opinion but is telling a fact (usually informing news).

By identifying the target subject, we can control the reading comprehension of the students as this is a more general task, so it's easier to find agreements. Answers with high disagreement in the target subject were considered as controversial (more than one answer with at least two votes for each one), so taken out from the final database. This means that people did not understand what the messages were telling, so this is an important filter for what is coming next. The second cleaning step consists of eliminating tweets in which the deviation score of the stance classification answers was too high, for our purposes more than two opposite labels were considered as disagreement.

After cleaning, we were left with a database of 150 labeled examples, more than enough for our purpose of using ULMFiT. We will also test how BERT behaves with this data anyway. An important conclusion of this little session is again that this task is not easy to accomplish: at least 1 of 4 tweets were controversial; so, we must recognize that algorithms will also face this difficulty.

4.5. Results on Tweet Classification task

The last part of this chapter is dedicated to summarizing our results at the *stance regarding Bitcoin's price* classification task. From the last chapter, we obtained 150 labeled examples, from which 100 must be reserved for training the task-specific classifier in the case of ULMFiT and BERT. Nevertheless, there exist a considerable number of pre-trained

algorithms for general sentiment/stance detection ([**Baccianella et al., 2010**], [**Mohammad et al., 2013**], [**Bindal and Chatterjee, 2016**]). As well as most algorithms in the area, they have been trained for general contexts and tasks by analyzing huge amounts of documents, so will not be a surprise that such algorithms will not perform that well over our specific task-driven data. While the knowledge learned by those models does not apply to our purposes, testing them should be informed about how difficult is to target our problem.

Following our beliefs, we state that differences in contexts, idiosyncrasy, and task purposes, will be decisive for the algorithm's performances, some of which have enormous differences at these initial conditions. To check this out, we are testing some of them in the *stance regarding Bitcoin's price* task. First, we will briefly review some differences in these initial conditions. *Sentiment140* is an algorithm that was trained to exactly match the idiosyncrasy of Twitter, i.e., was trained for comprehending the informal language of the media and for being adapted to the max fixed length of these messages, which originally was 140 (hence its name); however, this model was not trained for Bitcoin specifically, neither for stance detection but detecting the general sentiment expressed. Similarly, *Tweetment* is another known algorithm that was also trained for Twitter-like data, recognizing up to three sentiment labels. The case of *SentiWordNet* is very special, this one was developed as an opinion mining application capable of classifying sentiment expressed in the text within a continuous range [-1, 1], for instance, an opinion analyzed by this algorithm can retrieve a score of 0.3, meaning "somehow positive", this is achieved starting from a pre-trained dictionary that synthesizes n-grams relations, which were previously scored in positiveness, negativeness and objectiveness [**Baccianella et al., 2010**], again this algorithm was trained at general corpus and not even for Twitter's context. However, this one has been applied in a variety of NLP's tasks, such as language representation [**Ke et al., 2020**] and their applications like strategies for Curriculum Learning [**Rao et al., 2020**].

The last algorithm will be used as a baseline, which is one not even created for detecting sentiment but that can be easily adapted for any classification task by training it for a

few epochs, this is XG-Boost, a randomized decision trees-based algorithm, by using this we can explore from a general perspective the data we are using.

In the following results, we assume for most of the cases the same text pre-processing strategy, which is close to the well-known practices in the area. This includes:

- Change all text to lower case (Not used for ULMFIT and BERT).
- Tokenization: separate every tweet into tokens (words, symbols)
- Removing stop words, usually connectors, that repeat many times across the corpus and do not provide specific information (Not used for ULMFIT and BERT).
- Lemmatization: change similar words to their linguistic root, only when words mean the same.

All these pre-processing can be achieved with language packages as *nltk* in Python. The major differences at the pre-processing phase are found for the ULMFiT and BERT algorithms. ULMFiT does not consider the steps of converting to lowercase or removing information from the base text, such as stop words, their authors argue that these simplifications could result in a tremendous loss of information for these models, which are capable to generate more complex language understanding. Instead of the mentioned steps, in the case of BERT, some tokens are included at the starting/ending of paragraphs, as well as one special token for masking unknown words, so that minimal information is lost and only additional information is aggregated. Meanwhile, ULMFIT uses tokenization based on *subwords*. They are a special way of language representation that strikes a balance by using a mixture of character and word tokens. Indeed, they have shown two improvements from prior tokenization rules:

- Sub words more easily represent inflections, including common prefixes and suffixes, and are thus well-suited for morphologically rich languages.
- Sub word tokenization is a good fit for open-vocabulary problems and eliminates out-of-vocabulary tokens.

The discussed algorithms will be compared to get the one with the best performance for using it in the next chapter. Our results are summarized in table 4.1. We remark that the first three algorithms are pre-trained for the contexts and tasks previously referred to, so is not possible to extrapolate a *train's accuracy result*, so just XG-Boost, ULMFiT, and BERT will have informed training scores. In every case, we used 100 labeled examples for training classifiers, while always the same remaining 50 messages were reserved for testing them. The case of ULMFiT is special, as we saw in section 4.3 its second fine-tuning step is about adapting the language model to the idiosyncrasy of data, for which even unlabeled data is useful. This freedom allows us to try two different sources of data. The first one, a database created from general tweets about random topics that were discussed at a specific that moment in the media, which would allow, in theory, a good comprehension of the writing manner at the Twitter ambiance. The second one is tweeting regarding the specific Bitcoin topic but retrieved from **any** user in the media tweeting at the moment of consulting. We discovered abysmal differences at this election.

TABLE 4.1. Summary of model's performances. SA stands for Sentiment analysis while SD for Stance detection. Sub-index *w* stands for the weighted metric's version.

Algorithm's Name	Task	Context where was trained	Train Recall	Test_w Precision	Test_w Recall	Test F1 Score
Sentiment140	SA	General twitter data	-	0.58	0.46	0.44
Tweetment	SA	General twitter data	-	0.41	0.4	0.38
SentiWordNet	SA	Wiki corpus	-	0.48	0.48	0.47
XGBoost	SD	100 labeled Bitcoin tweets	1	0.41	0.42	0.38
BERT	SD	100 labeled Bitcoin tweets	0.87	0.72	0.63	0.67
ULMFiT	SD	LM with General twitter data	0.55	0.46	0.46	0.46
ULMFiT	SD	LM with specific Bitcoin's tweets	0.71	0.63	0.62	0.62

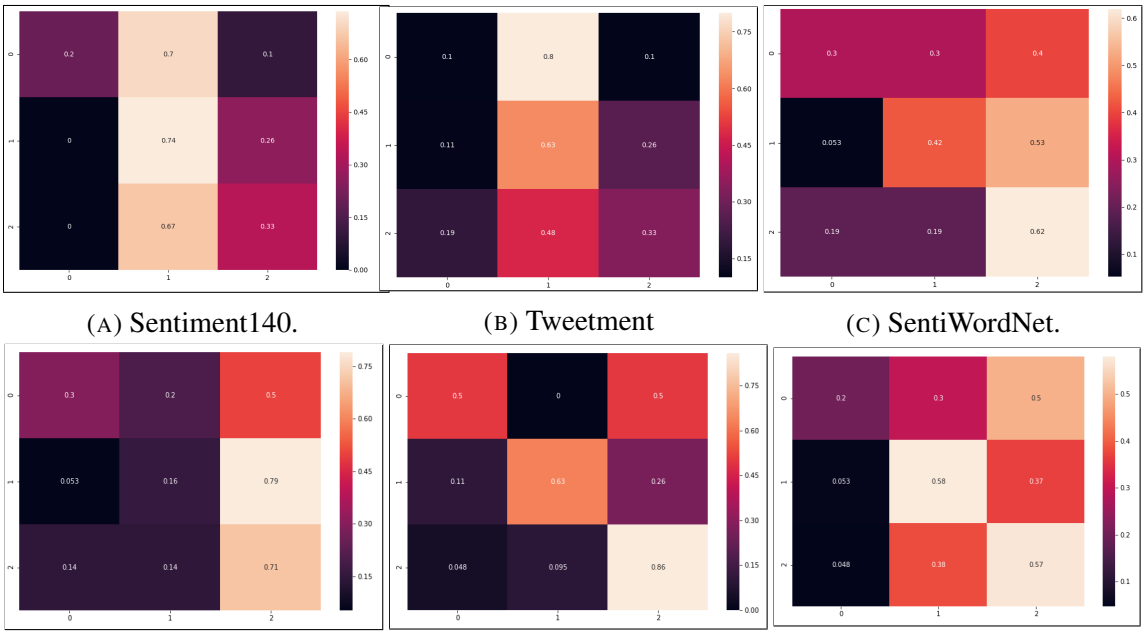
As shown in table 4.1, the algorithm's task specification has a dramatic effect on the algorithm's performance. All sentiment analysis ones performed worse than random. Here is clear that these algorithms have learned classification rules that are not useful for our purposes, mainly because they were thought for different ends, even though the training data coincide in context with one of our data comes (Twitter media). SentiWordNet reaches the best performance among sentiment's algorithms, achieving an F1-score of 0.47, outperforming ULMFiT fine-tuned from general Twitter's data. However, the result is tricky. As we remember, SentiWordNet retrieves a continuous value, so to accomplish comparisons with discrete classification algorithms we changed their scores to labels, which bounds were set for maximizing the classification score. So, the result is

not a surprise and it is only useful for demonstrating that even in this configuration, sentiment analysis is not a suitable task for our purposes, as results for the remaining sentiment analysis algorithms demonstrate, by not outperforming SentiWordNet. Our baseline, XGBooost, was outperformed by all other algorithms, including the sentiment ones, the result in the training set demonstrates that this algorithm adjusts its classification rules to classify correctly every example in this set, so is clear that over-fits easily influencing the test results directly. By using XGBoost we realized once again that the task is not easy to accomplish.

All stance classification algorithms outperform the sentiment analysis F1 scores, being proof of the potential of these inductive transfer learning algorithms. ULMFiT's performance depends highly on the quality of the fine-tuning process. The adaptation to the task-specific idiosyncrasy, by analyzing documents related to the context, is essential in achieving state-of-the-art results in this algorithm. Here we reached a new train specification level, where even for similar set-in language structure, the topic they discuss is determinant for the performance of the classifier in the desired task. Both, BERT and ULMFiT fine-tuned over specific Bitcoin data, reach similar performances in terms of F1 scores, even when BERT achieve better results in learning the train set, in this case, differences in the training stage, do not turn into significantly better performances for the test F1 score of the models, so it remains interesting to try out both, at the predicting model.

Figure 4.4 shows the classification made by different algorithms over the test set. This figure contains six different Confusion Matrix, whose purpose is to inform directly the quality of the classification made and the kind of understanding that algorithms have developed. Both sentiment140 and Tweetment classify mostly of Bitcoin's referring tweets as neutral, this result is very interesting because effectively, as we can check by reading many examples of them, the language used for referring the Bitcoin's market is indeed neutral in sentiment. This result is once again a confirmation of how different the sentiment analysis is from the task we seek to analyze, therefore that is not possible pretending these kinds of the algorithm to adapt well to our data. On the other hand, both SentiWordNet and XG-Boost, made to maximize the classification of the training set, show indeed

the distribution of this set as these algorithms over-fit this data, particularly, the case of XG-Boost is dramatic as even the neutral examples are considered positive.



(A) Sentiment140. (B) Tweetment (C) SentiWordNet.

(D) XG-Boost. (E) BERT. (F) ULMFiT best.

FIGURE 4.4. Confusion Matrices (CM) over the reserved 50 examples for testing under each of the different algorithms considered.

Finally, our best classification results are achieved by both ULMFiT and BERT. Although both algorithms perform equally on average, the Confusion Matrices are more informative of how their classification is distributed. In particular, ULMFiT has more deviation in its classification; this algorithm is characterized by recognizing well neutral and positive tweets with some misclassification between them, in the other hand ULMFiT does not classify that good the negative ones, which are easily misclassified as positives. BERT by its part has the better classification report as positives and neutral tweets have some of the better scores, the case of negative tweets is similar but, in this case, is better recognized but equally well and misclassified among positives. The classification errors at negative messages alert us that if we want to use these algorithms to generate novel inputs for the prediction model, we have to be aware of these misclassifications and weigh them in some way. In the following section, the results for the studied task introduced in section

3 are shown starting from architecture as the one presented in 3.1, that now is fed with both price and opinion information.

5. INCORPORATING OPINION TO THE PRICE PREDICTION TASK

While we have found a way to represent the meaning of an opinion so that it is measurable and thus can be compared to any other one in the same terms, the best way of feeding this information to the model can still be explored.

5.1. How to feed the stance information?

In the case of price, we were previously using the moving average measure over a day, as well as other statistical features from the 288 observations. In the case of stances, as we discussed in the last section, we only have 43.5 tweets per day across the nine months considered in this study. While the last number is not small at all, is also almost 7 times lower than the number of measures per day that we have for studying price, so at the time of using this information, we will find a lot of time steps without an opinion measure. For this reason, we propose include some statistical indicator used in studying economics assets, we refer to the *momentum*, *moving averages*, *correlation score* against price. In the case of momentum, we are going to report the sum of the opinion score over a movable window. We expect these measures will be a good information resource about the general appreciation of the cryptocurrency and how is changing in time.

Another important consideration is how to proceed with the different information retrieved from opinion. From BERT we are getting signals of positive, negative, and neutral interactions in media referring to Bitcoin. The most direct option is, to sum up, all opinions to get an overall measure of what is happening in media according to these influential users, however, one natural throwback that comes with this is that positive, negative, and even neutral signals have different distributions, which not only depend on what is the overall opinion about Bitcoin on Twitter, but also of the level of participation of each user at the actual ongoing debate, which may vary quite a bit. Indeed, figure 5.1 shows differences between the two main stance signals: The negative tweet's distribution is more centered to zero than the positive distribution.

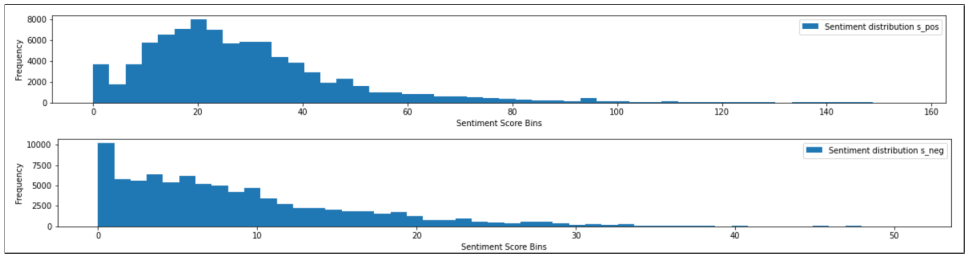


FIGURE 5.1. Stance score distribution about positive and negative appreciation on Bitcoin as retrieved from BERT algorithm.

Because of this, the alternative is to consider counting positive and negative tweets separately and characterizing each *mood* as a varying signal according to its own distribution just as presented in figure 5.1, which leverage the mentioned difficulties but also introduces a natural doubt ¿What should be the model interpretation when there are no opinions? Indeed, this is a natural question when we see that negative stances are less represented in our sample, a huge number of data points in this signal add up to a zero score. If we use a scaling function to characterize the variability of the signal, the absence of opinions will have an equivalent in the scale that will not be necessarily equal to 0 depending on the scale. Every possibility will be studied in the next section.

5.2. Analyzing Twitter's opinions polarity from BERT.

The distribution previously showed is complemented by the 5-minutes stance distribution presented in figure 5.2. In the image, the positives, negatives, and neutrals messages are counted and presented for every time step as a series.

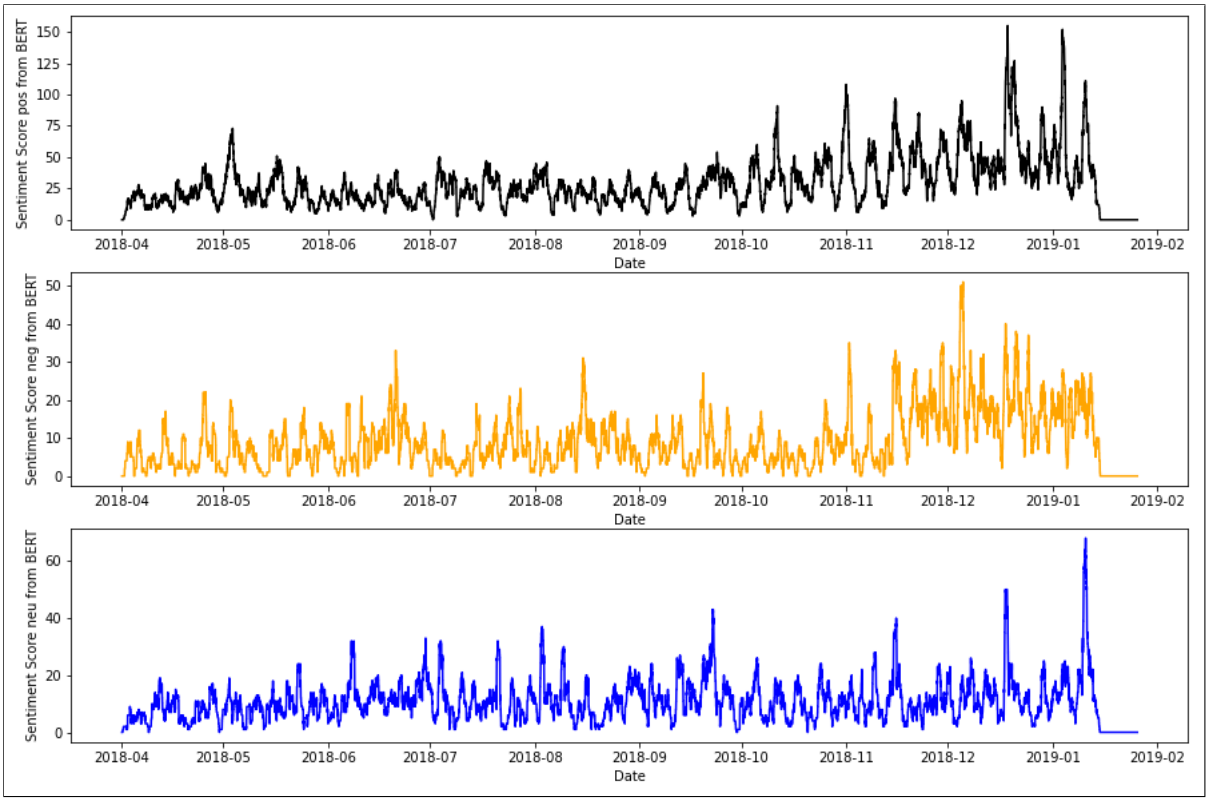


FIGURE 5.2. Stance distribution of tweet polarity. Neutral messages are included for completeness.

The first thing we conclude from figure 5.2 is that the retrieved signal is very noisy across the entire period considered. Probably there are some temporary effects involved as seasonality (daily for example). There exists some frequency at which Bitcoin users left speak about it. On the other hand, the number of detected tweets associated with each stance polarity seems to get increased while we are getting data closer to more current dates. This increasing number of retrieved messages could be partially explained by limitations in Twitter API services for recovering older messages. Indeed, when you query for a particular user, the API will allow you to just get a maximum number of messages starting from the date of the query. One last concern is that the number of positive opinions tends to highlight in number over the other tendencies. While negative ones show a relatively plain behavior at the daily maximum reached value and just recently showing a major cumulative value. On the other hand, the positive ones show higher peaks along

with all the studied history. These properties at the data encourage us to also applying daily scales rather than general scales, to ensure the correct awareness in daily changes. Regarding using polar series over overall score series, we try a previously introduced experiment. Similarly, to the exercise made in chapter 3, we analyze in figure 5.3 the Pearson correlation coefficient between price and stance polar signals against multiple retarding windows between both series.

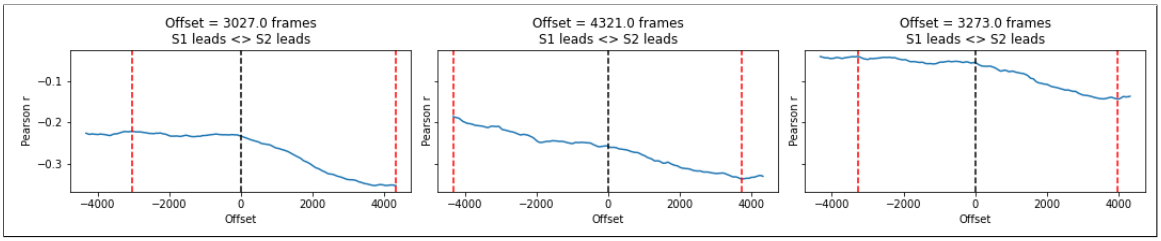


FIGURE 5.3. Pearson correlation coefficient between Bitcoin's price and the stance signals (From right to left: Positive, negative and neutral stances), across a movable retarding window between series. The level of correlation can be though as an *naive hint* of causality.

By interpreting the coefficient as a causality indicator, we can see that the inverse correlation gets stronger for a positive retarding factor across all stance signals, the correct interpretation here should be that changes in positive and negative stance signals are results from previous changes in Bitcoin's price, which is not our purposes as we would like to establish an opposite relation's direction, so deeper analysis is probably recommended to get more benefits from these stimuli. We are not able to set a similar or stronger relation between price and number of emitted neutral messages, as they appear to be the ones more similarly distributed along the time span studied.

Finally, we present in figure 5.4 the Pearson correlation coefficient between Bitcoin's price and the overall stance (negatives + positives messages) expressed in media regarding Bitcoin at the time of price's variations.

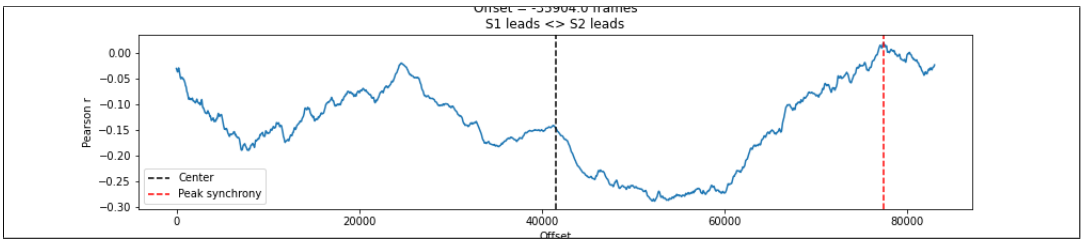
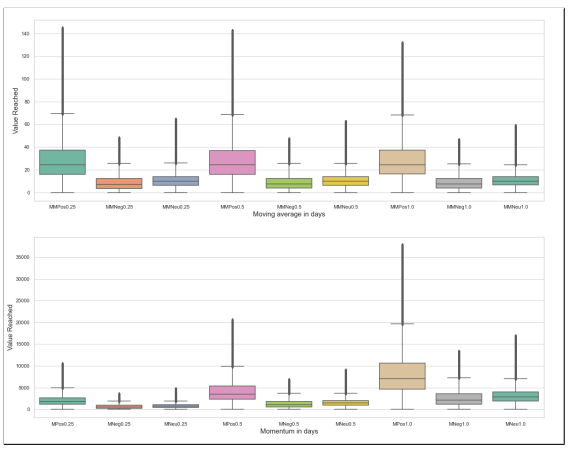


FIGURE 5.4. Pearson correlation coefficient between Bitcoin's price and the overall stance in Twitter media about Bitcoin across the time.

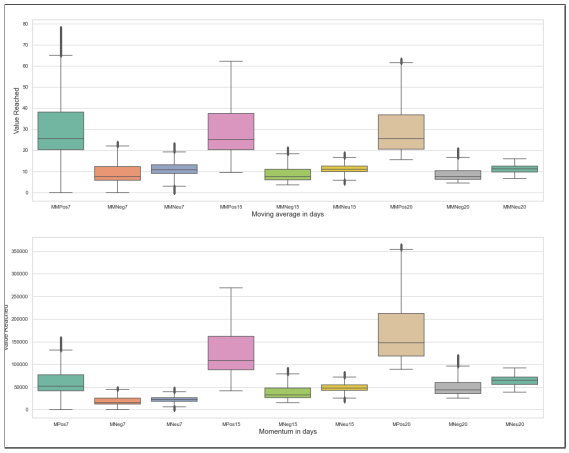
This figure exhibits an important difference from the previous similar results shown in figures 2.5 & 5.3. While we see that there exists a growth in the negative correlation when we apply a positive lag to the overall stance series, we also see that exists a negative correlation decrease when we apply a negative lag to the opinion signal. This is the first time we detect an approximation of the increase of this inverse causality correlation at negative lags. After these discoveries, we conclude two main things, first that the information from stances is not as promising as we would like so we could be only able to set relation in this negative causality way. The second is that however, we are especially interested in using the overall stance signal over each one separated, but for completeness, we will explore both ways of feeding the opinion information, and increase our chances, we are introducing more statistical measures.

5.3. Statistics measures for stance analysis

From the results discussed previously, it is proposed to include some statistical measures, to the stance series retrieved by using BERT. To this far we are including moving averages and feature momentum series as possible assets useful in characterizing the Bitcoin price's distribution.



(A) Moving average (up) and momentum (down) measures on horizons of 6, 12 and 24 hours.



(B) Moving average (up) and momentum (down) measures on horizons of 7, 15 and 20 days

FIGURE 5.5. Statistic features

In order to get an idea of which windows are the ones more useful in representing variability but that also allows us to relax the distributive difference between different polarities, we create boxplot figures for moving averages and momentum measures with variable window size as presented in figure 5.5. From these figures we can drive out many ideas, in general, the mean distribution of the moving averages of positives stances is centered in a very specific value, we only reveal that outliers tend to be less when we average over larger windows. This is interesting because the *participation* of influential people with a positive outlook towards bitcoin at the Twitter's debate, appears to converge to a mean number of interactions, while negatives and neutral ones tend to be more noise even when we extend the window size, how can be appreciated with the outliers that disappear in figure 5.5b in the positive distribution when larger sizes are considered. This discovery makes stronger our thoughts about considering a unique overall measure for sentiment discussed previously, this is because, if the positive participation is more stable than the negative one, then we can learn something about the *difference* of their distribution rather than each one separated, to analyze the effect of this variable we are including it in our models. Finally, by analyzing the momentum boxplot's it follows that the moving cumulative distribution grows in time, being the positive ones the ones with the biggest

increasing through time, again their distribution tends to regularize themselves as more data is seen, but again the negative signal remains with more outliers at the end, in this case, the richest information appears to be at low momentum's windows, as differences between the mean of the distributions are not too large and there are a good amount of outliers that can be indicative of *price's stimulus* through an increase at the cumulative Bitcoin's appreciation by our influential users.

We hope this feature engineering will be helpful in what follows, the prediction itself. However, we have concerns about how much it can be improved with the available data.

6. PREDICTING PRICE USING OPINION AS A EXTERNAL STIMULUS

From the learning collected at the future price projection task and the treatment of the stances retrieved from BERT, we ended up with an objective proposal for combining these two resources of information. In this section, we are going to explore preliminary results in using SentiWordNet as the resource of opinion information. Then, its performance will be compared against models trained with our data treatment previously described. All our experiments will be tested through the model introduced in section 3.1 and an XGBoost tree model trained for prediction.

6.1. Some preliminary results in introducing opinion as an external stimulus.

In chapter 4 we experimented with 7 different models for interpreting opinions. In table 4.1 we show our main results at the Stance Detection and Sentiment Analysis tasks. Among them, the trained ones for "stance detection" purposes showed the best performance for the test data retrieved from the labeling session; however, one of the models trained for "Sentiment Analysis" is very interesting so it deserves a particular analysis. We refer to the SentiWordNet project, which is unique in our literature review. As was discussed in section 4.5, the main reason for highlighting it is because the sentiment interpretation made by this algorithm is more flexible, being able to return a continuous interpretation of the positiveness or negativeness expressed. Under this scenario, we can get an overall measure of the general opinion regarding Bitcoin at every moment. With the introduction of the sentiment measure, we implement a model as the one described in figure 3.1 with the difference that the input layer is fed now with 288*2 observations, where the 2 represents an additional dimension per data point for the 1-day sentiment momentum at any point in time. We trained this model for prediction and price simulation; its performance is presented in figure 6.1 with a closer view to the price simulation task in figure 6.2.

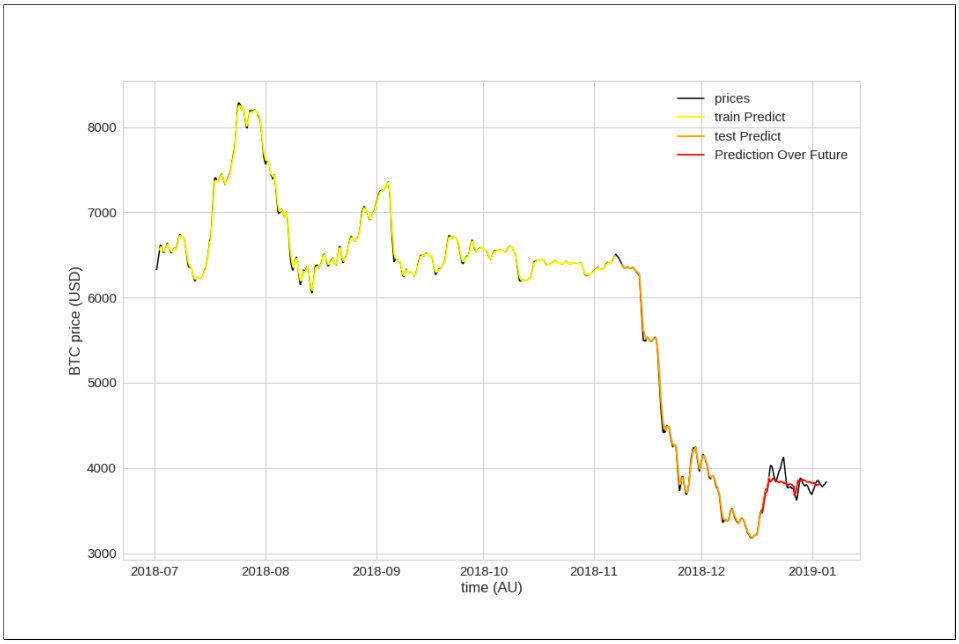


FIGURE 6.1. Prediction of the model when is trained with sentiment data by using SentiWordNet. The predictions for the next-day price are on the same level as the model trained without twitter data.

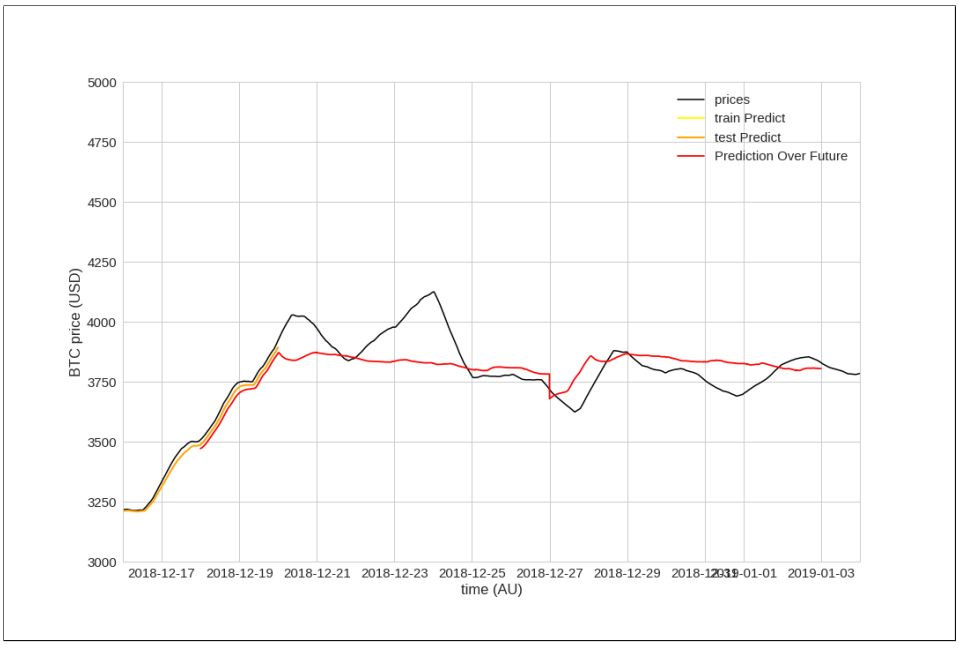


FIGURE 6.2. Zoom to final predictions. We note a fall over the seventh day that is recovered by the prediction.

Regarding the next-day problem, we noticed essentially the same predictive power as the model presented in the chapter 3 when we compare them via RMSE as showed in table 6.1, the results in this metric reveal that the model gets better train performance but worse in the test set, which is indicative of over-fitting. However, the most interesting result comes from the performance at the price simulation problem when looking at figure 6.2 and comparing it with our previous results at figure 3.5.

TABLE 6.1. Summary of rmse scores by model at the price prediction task.

Model	RMSE train	RMSE test
Next day Prediction general scale	4.762	25.248
Next day Prediction daily scale	1.895	1.941
Next day prediction SentiWord overall sentiment	1.494	2.143

While in previous results we have not seen awareness in price changes, this is the first time that a model is able to respond to changes in price evolution as we see in the figure 6.2. There is a drop in price at 27th December 2018, with a subsequent increase in price by the end of the predicted period and even many days after that the simulation started. By looking at 3.5, we can only suspect that it was Twitter's data the information that helped shape the curve in this way this time, by introducing well-interpreted stimulus, as in this case the overall appreciation of Twitter's influential users.

We have included the sentiment momentum distribution retrieved from SentiWordNet in figure 6.3. We notice some trends by the end of December, but clearly, there is not a huge visual correlation with price. It is surprising that this data still allows us to predict spike changes within the price simulation framework.

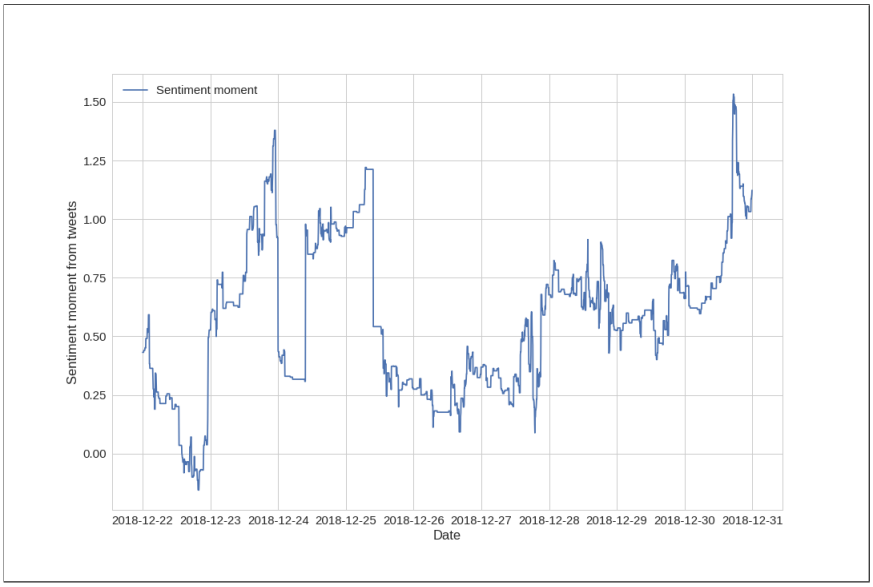


FIGURE 6.3. Evolution of sentiment movement retrieved from using SentiWordNet for interpreting opinion data, during prediction dates.

Despite the described improvements at the simulation task, we must notice that our model is not still capable of noticing the initial increases in price at the start of the considered time, so there is still room for improvement. The obvious question is ¿Can our predictions benefit from the *stance*-based interpretation of the opinions?

6.2. Results in prediction using stance stimulus

As mentioned in the last section, we want to test our model in predicting price by supporting their decision instance stimulus obtained from the analysis of a BERT algorithm fine-tuned for this task. To this end, we propose feeding our network with 2 resources of information, extracting the following measures,

- Price Information
  - 1-day moving average at the current price. Price is embedded through a variable function scale across each moving day.
  - 5-minutes moving difference between the last price and the actual price.

  - Growing rate of volume of transaction every 5 minutes.
  - Correlation score between 1-day moving price and volume.

- Stance Information
  - 7-days moving average from positive, negative, and number of neutral messages.
  - 6-hours moving momentum from a positive, negative, and neutral cumulative appreciation for Bitcoin.
  - Difference between polar signals as an overall web's opinion at the time of prediction.
  - Difference between polar signals weighted by the number of neutral messages as an overall web's opinion at the time of prediction.

For further discussion about how this data is called in our experiments and it is presented in some figures, you may refer to appendix C. We are using these features for training both LSTM and XGBoost models. The results presented in table 6.2 summarize the performance of the models as well as the two main ways used in this research for feeding stance: the overall stance and separated polarity.

TABLE 6.2. Summary of model performances at train and test phases.

Algorithm's	Stance Embedding	RMSE Score Train	RMSE Score Test
LSTM Bitcoin	Separate Polarity	1.552	2.14
LSTM Bitcoin	Overall Stance	1.82	2.53
XGBoost Bitcoin	Separate Polarity	0.67	2.451
XGBoost Bitcoin	Overall Stance	0.152	1.78

This table reveals that each model has its own data configuration where benefits more of the knowledge retrieved; the LSTM reaches a better performance by using separated signals for embedding opinion, instead, the XGBoost model does the same by using an overall stance measure. This difference deserves its own analysis but we have to recognize that each one makes different use of the same information, while the XGBoost model uses punctual values for prediction, the LSTM model uses series (in this case 288 measures)

for the same task. In this way, the more data is better read by LSTM because this model can learn by itself the latent representation that is more useful for prediction, while for XGBoost, which creates structures like trees, it is easier to make a decision with few but more representative information. The scores achieved by each model are unequal by a maximum of one order of magnitude which can be an indicator of overfitting at the tree-like models as well as the big difficulties of LSTMs of taking more advantage of this data. In what follows we will analyze each model with the stance embedding that produces the best model performance at table 6.2.

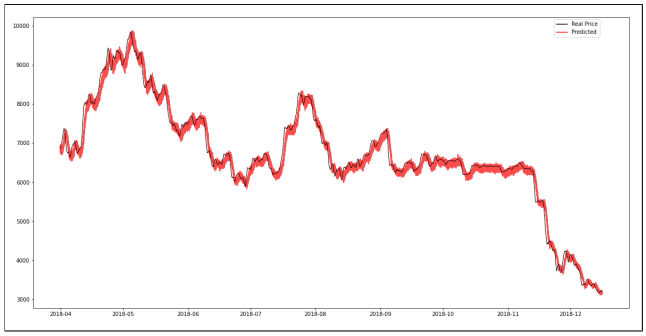


FIGURE 6.4. Results in prediction for train set using LSTM's based network.

In figure 6.4 we show our results at the trained LSTM network in the case of separated polarity for feeding stance. This figure shows at first good performance of the model at adjusting the original price curve. However, is significantly visible that now the red curve domains the figure. Instead, the black one which represents the real price is completely overshadowed. To understand what is happening we have to take a closer view of this distribution. In figure 6.5 we show the model performance over our test set, in which price simulations have been carried out.

Some interesting insights do appear when we see the last image in detail. First of all, that the predicted line, presented as the red one, is not as smooth as previously was in the last chapters, now, the predictions at each point are highly variable. In this sense, does appear that stance is influencing predictions at a very fine level but not supporting the decision process at a general level, i.e., not being able of creating awareness for more

general changes in price. Effectively we also see that the red curve is very similar to the black one but displaced some number of measures to the future.

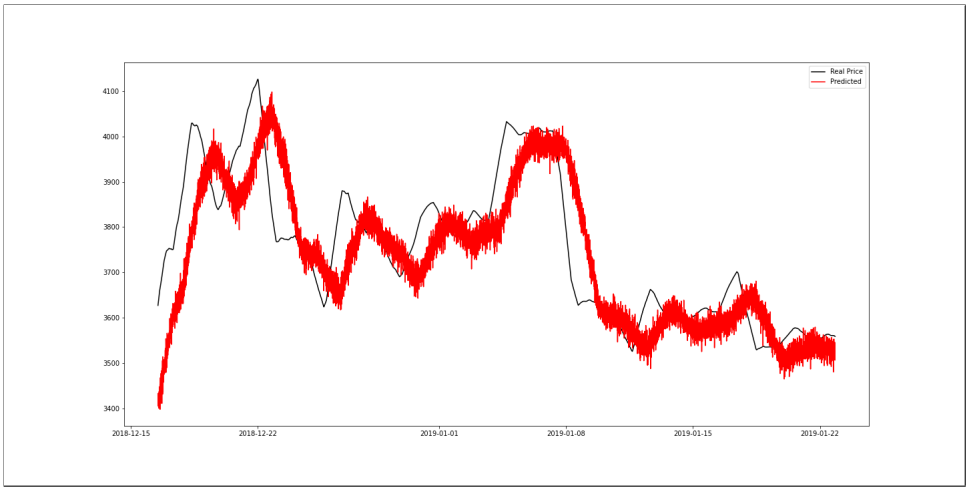
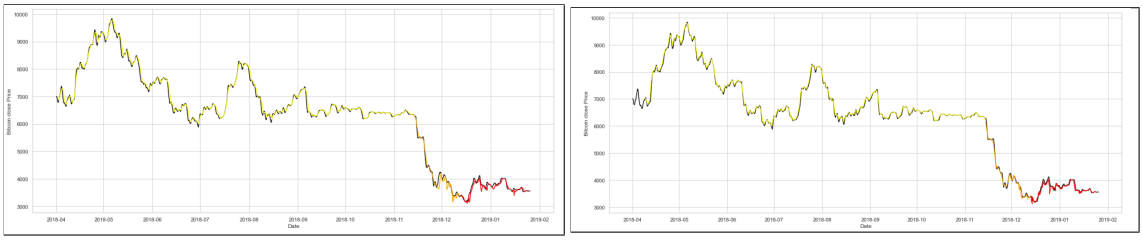


FIGURE 6.5. Results in prediction for the test set using Bitcoin LSTM.

At this point it is clear that the network does learn that projecting the actual value as a prediction of the price evolution, is a good approximation of what is going to happen. While this result is not as desirable as we expect, it is also undeniable that any solution must show a relation between last seen values and the time gap of separation between the original series and prediction, in this way we recognize that these predictions are results from a training process, but this model is not using the external stimulus information at supporting the prediction decision task in the coarse changes of the series. These results are not promising at all, because implies our model is unable to set a more relevant relation between stance and price.

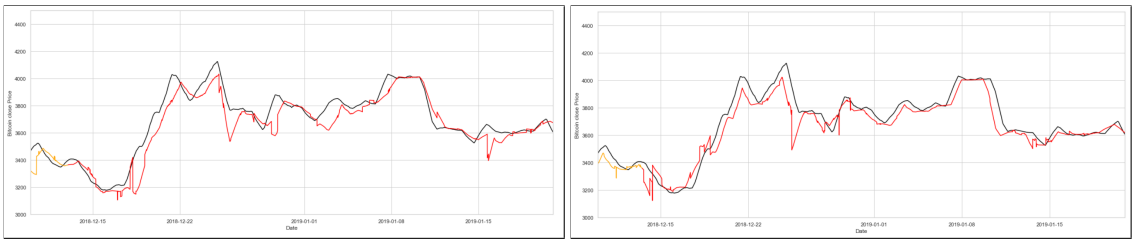
In order to confirm our predictive capacity with the data obtained, we trained an XGBoost model to predict the price one day in the future, for further details on the trained task and the hyper-parameters chosen in the model design, you may ask the appendix B. We show in figure 6.6 the results in price prediction by using the XGBoost's models over polar and overall stance information.



(A) Results in prediction for polar stance (B) Results in prediction for overall stance

FIGURE 6.6. Results in one-day price prediction using stance measures for training a XGBoost model.

At figures 6.7 there are also some close views of predictions at the test set, the timespan where price simulation will be handled soon.



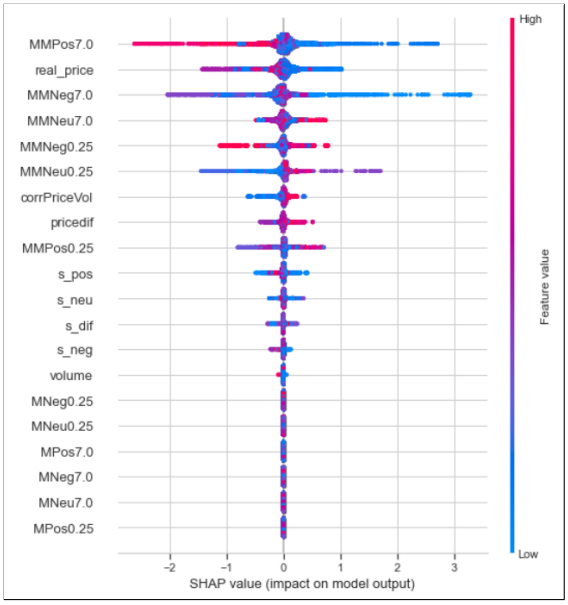
(A) Results in prediction for polar stance (B) Results in prediction for overall stance

FIGURE 6.7. Results in prediction using XGBoost model for test set.

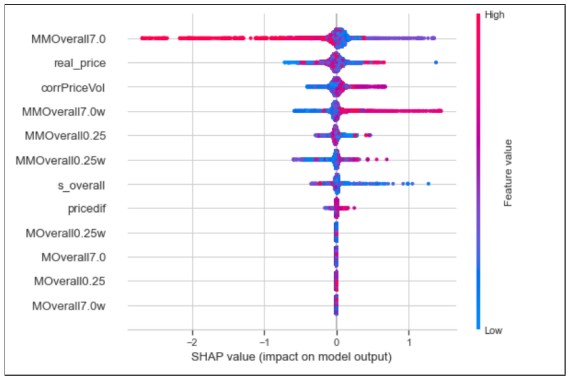
Instead of what was obtained the first time from LSTMs, figure 6.6 shows the model's better adaptation and performance at the prediction task. The models do not show a determinant effect on the time gap dependence between the original series and prediction, on the other hand, the retrieved values do not seem to be sampled randomly, the predicted curve reproduces coarse changes in price but also that there are some local variations. This behavior is explained by the introduction of the XGBoost model which appears to be more efficient for the proposed data and task. Although to be fair, the XGBoost model is also easier to parametrize through grid search. We also believe that the definition of the scale function must play a crucial role.

The behavior of both models in terms of predictive performance is almost identical, but the trained model with overall stance has a slightly better one. In this section, we

will not discuss in detail the differences between these adjustment levels, since they are quite similar to those obtained in section 2, so the gain cannot just be at the RMSE level achieved at this limited task, in the last paragraph we will focus on the performance and differences of these models in the price simulation task, which seems to be a more challenging problem for us. In the meantime, we will discuss an interesting feature of these models. The tree-based models create hierarchical decision structures that depending on their complexity, are even possible to be graphed. This property of tree models allows us to know the decisions that the tree is making exactly at all times and in that way, weigh the importance that has each feature when the model makes a tree-based prediction. As usual, we can visualize this information by plotting the Shapley values of each feature, which is a very know measure in games theory. In figure 6.8 we show this distribution for each XGBoost's models.



(A) Shapley value's graph for features used in price prediction and polar stance measures.



(B) Shapley value's graph for features used in price prediction and overall stance measure.

FIGURE 6.8. Shapley value's graphs from the XGBoost models trained for price prediction.

There are some really interesting conclusions we can make up starting from these figures. First of all, from figure 6.8a the feature with the higher impact is the positive

stance moving average over a window of 7 days. The relation is very unusual, according to this figure while the higher is the moving average, the higher is the negative importance that this variable has to do with prediction. That is, while people are having a greater positive stance with respect to Bitcoin, our model detects that higher is the probability of breaking the bubble price, so a fall. In this sense, we can settle down that this tree agrees that *every rise is followed by a fall* ¿could these opinions serves as alarms of these events? is an interesting question that is born from these results

The following important feature for prediction is the price itself, with similar behavior as the moving average has for a positive opinion. Then appears the moving average for the negative stance measure, the values achieved by this one has a great dispersion as the lower values (blue points in 6.8a) appear to have a positive and negative impact on the prediction, so this feature could be decisive in determining growth, but the relation is widely dispersed. The other important features are moving averages measures over smaller windows of time, with the lowest importance appears the Momentum measures at any window, we discard this information based on the importance revealed by these measures.

On the other hand in figure 6.8b we analyze the feature's relevance obtained from the model trained with overall stance data. In this case, from each of the introduced polar measures, we create two new ones: An overall measure starting from the difference between polar signals, and a weighted version that takes into account the number of neutral messages. Again, the impact of price and 7-days moving average for the overall stance is higher and the direction of the relations remains similar. Then real price and the correlation factor between price and the volume of the transaction appears as relevant features for prediction with some changes in their relevance although very similar to the polar stance's model, however, our main interest is in the fourth place which is taken by the weighted version of the first placed moving average measure for overall stance. This calculated feature has a positive impact when it grows and a negative one when it lowers which may be an indicator of a great correlation. In figure 6.9, we present the evolution of these two series.

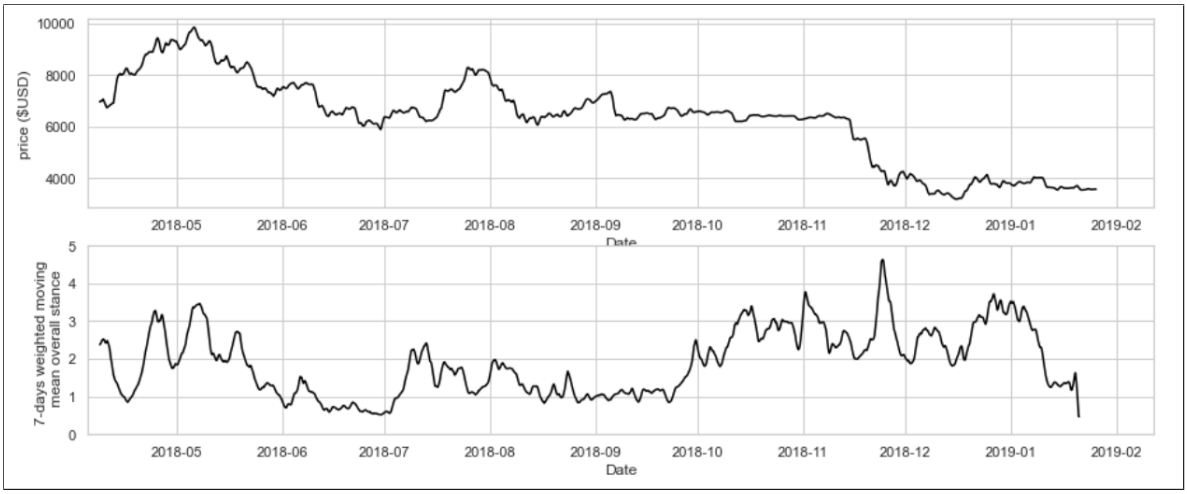


FIGURE 6.9. Top: Bitcoin price evolution for the studied timespan. Bottom: Moving average of the overall stance that influential Twitter users have on Bitcoin

By plotting side by side both series, visually there seems to be no greater correlation in coarse changes, however, there are some points in the very first half of the studied history in which some peaks in moving average are happening consecutively to the ones in price. The overall stance at the second half history is very irregular. One reason may be in the amount of data collected for this time-span which is remarkable bigger than the one for the older time-span, this could explain a great diversity in discussion as more messages are collected. What is undoubted is that this method allows us to identify remarkable outliers to the series. We did not include Momentum in this analysis from our previous conclusions. The feature engineering process was superior to the best neural network solution, confirming its irreplaceable importance in the analysis process.

Finally, we present in figure 6.10 our results using the proposed models for the price simulation task, previously discussed. At first glance, it is evident that the simulation is no longer aware of variations after the 4th day of being started. Some stimulus causes the simulated curve to vary again around December 29, 2018, to finally converge to an average value. The direction of the variations seems almost correct for the first 4 days.

The presented results are very similar to the ones obtained previously in figure 6.2 by using the SentiWordNet information, but this time, our awareness is more limited in

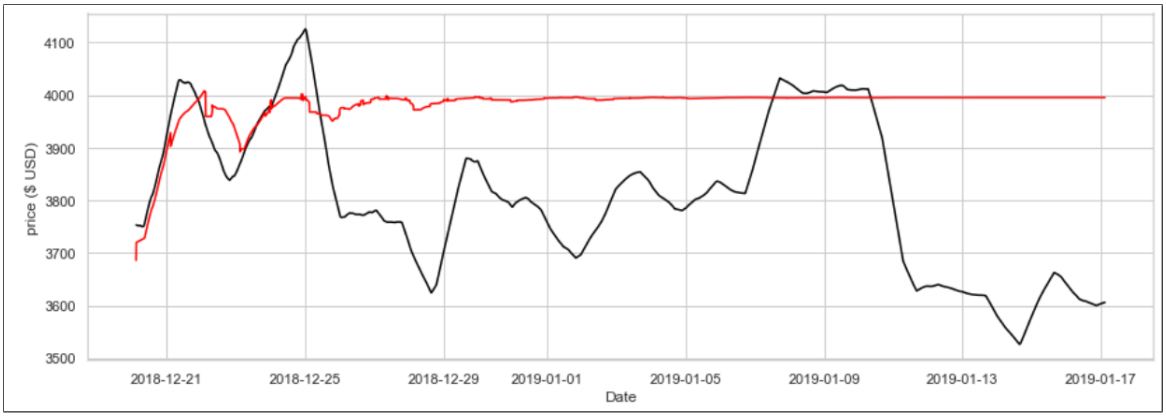


FIGURE 6.10. Price simulation task handled by the XGboost model trained with overall stance measures.

time than before. However, the new model reproduces better some initial variations that were completely lost in the previous study. The cause of these differences is not obvious. Apparently, in the case of SentiWordNet, the model is capable of recovering strong opinion stimuli to predict certain specific future behavior. In the recent case, it seems that the prediction is irretrievably bent to an average value after several iterations. However, this effect seems to be more associated with the proposed variable scale method than with the actual behavior of the tree model.

The information revealed in these models seems promising, there are not few examples in the literature that discuss the potential to build models based on the ensemble of other multiple models, where each one has the ability to generate relevant, accurate, and useful predictions on different windows of time. We are not able to settle down a strong relationship between price and *crypto-influencers*'s opinion, but at least, we found evidence that reveals the usefulness of this data in short-time elapsed predictions. While we are making some progress in reveal these relations, there is still a big room for improvements.

6.3. Thoughts in Bitcoin's price prediction task

In the last sections, we have selected features and models good enough for predicting Bitcoin's price one day in future evolution for the studied period between June 2018 and February 2019, however it is not clear if this consciousness is inherited to more recent

times. The time span studied of just nine months is quite short compared with the 3 years that Bitcoin has remained online at the time of analyzing its price in this research. The main reason for not extending our analysis to earlier times is because Twitter information is not easy to obtain which is a natural throwback for the proposed methodology.

At the studied time, the price reached by Bitcoin was around 4000 USD, however, by May 2019 this value has doubled, stabilizing at this value for most of that year and even during the beginning of 2020 until March. From then until now the price of bitcoin has only increased most of the time, reaching just a few weeks ago a new all-time high of almost 57000 USD, which is more than two times higher than the last maximum value reached in December 2017. This new maximum is also many times higher than the prices analyzed in this research, so it is difficult to assure that the models tried here will remain with similar levels of performance at more recent times. Indeed, it is highly unlikely that the learning achieved with such low-price data could be used to assume that the price would reach low-price levels up to 13 times the maximum value at the time span of the price simulation task.

In this way, it is undeniable that Bitcoin is an asset with a one-of-a-kind behavior and therefore, all our hypotheses that it would reach stability fell with the last registered prices. A great concern is whether this behavior will be just a new price bubble that will bring with it a precipitous fall, as a result of the burst. Certainly, some recently recorded prices show a drop of almost 10000 USD in just a few days. However, these drops are not as high as the moving averages arise in price registered since November 2020. These kinds of behavior are difficult to predict and it is not clear what they respond to. The effects of the pandemic and the role that virtual communities in social networks are promoting in stock markets[1] are new actors that could probably be playing a role in the described behavior, so that new areas of research are opened to this regard.

[1]As discussed in https://www.nytimes.com/2021/01/27/business/gamestop-wall-street-bets.html with the case of Reddit's users, playing a decisive role in the valuation of GameStop's stocks.

7. CONCLUSIONS

In this research, we have made advancements towards testing out if the activity of influential personalities in the digital world does affect the price of Bitcoin. Our main results suggest first that there is a negative causality relation between Bitcoin and Twitter number of messages referring to Bitcoin, which means that the participation of the influential users in Twitter media is a result of Bitcoin price variations. This causality relation is slightly positive when we compared the price to the overall stance that influential users have about Bitcoin. With this in mind, we did experiments for studying the capability of predicting price using this information. We have first shown a possible architecture that goes a long way into predicting the price when fed only with past price data: the error for predicting the next price point was almost negligible, however, the predictions for the next-day price were not accurate as previously were.

When fed with both price and Twitter data, the resulting model had essentially the same predictive score as the model fitted only with price data but now allows us to make more spaced predictions. We show that the sentiment score retrieved from SentiWordNet is very useful in supporting this process. At the simulating price task, the results lead us to believe that there were promising relations to be discovered between these two measures.

We tried two of the newer methods in fine-tuning language models for training a new NLP task over Twitter's context in order to refine the way we interpret opinions. We observed a 75% success rate for humans in this task, indeed 1 over 4 tweets referring to Bitcoin was controversial under our definition. Using these results, we check out the adaptability of the UMLFIT method to learn correctly the new context, while BERT reached the best performance for interpreting the stance expressed with just 100 labeled examples. We conclude that the fine-tune technique is a very well-suited tool to take advantage of the knowledge learned from language models trained in a large corpus of text containing a wide variety of information. We have proved the potential of the inductive transfer learning method.

The LSTM model showed difficulties at simulating the future price even with stance data, the unique possible prediction made by the model was the actual price which was projected to the future. On the other hand, the XGBoost model showed better performance at predicting future prices, recovering the capacity of varying the prediction based on the stance stimulus as was previously explored, by using the sentiment from SentiWordNet and even exceeding its performance. However, the price simulation task was not well behaved as we expected, the performance at this task was three times lower than the scores achieved at predicting the next-day price. In this way, we cannot conclude that our models have the full capacity to predict price correctly for a long time. Indeed, starting the fourth day of our simulation, our best model is only able to predict an average value.

Most of the correlation test studied between Bitcoin's price and all of the ways of quantifying opinions here revealed a negative causality measure, i.e., these opinions are results of what is happening with this cryptocurrency. Despite we found a positive impact relation between price and moving average for overall stances, we cannot be able to settle down a full correlation in the inverse. However, our main contribution is to create a reproducible methodology that allows studying the evolution of any asset price that also proposes a novel way to represent the opinion so that it can be used as an additional input on another prediction task.

The roller coaster behavior we saw in 2017 repeats itself, and this time the curve reached has us all even more expectant and excited than before, wondering when the crash will come. However, there are other topics that should be of immediate interest. The cost of a Bitcoin transaction is evaluated as the most expensive in terms of the amount of energy consumed in the process. All explained by the technology behind Bitcoin's success: the blockchain. The reliability obtained from decentralization is thanks to identical historical copied databases distributed between all the nodes of the network. The size of this database is currently more than 250GB. In order to add a new transaction to the blockchain, a complex mathematical problem must be solved, which at the moment is mostly solved randomly, which leaves the miners with no other option but to invest in higher computing speed in their machines and use them during extended periods of time,

while they are competing for the Bitcoins. According to the research of the University of Cambridge, [**Rauchs et al., 2021**], if Bitcoin were a country, the electricity consumption of this industry would have reached 30th place among the countries that most energy consumes. In this position, Bitcoin surpasses Finland, Switzerland, and Argentina. The main concern about these numbers is how electricity is currently generated and obtained, which still being mainly through coal combustion due to its low price. The still unknown effects of climate change make us raise the alarm against this technology, as long as its relationship with the environment does not change.

Future Work

After the results presented in this research, we are able to settle down at least three lines of improvements. The first one refers to the prediction model itself, without a doubt more complex architectures can be created and from a diversity of approaches like Generative Adversarial Networks, Variational Autoencoders or through a model based on Reinforcement Learning, however, the latest price hikes make us question how predictable this behavior can be by leaving everything to models. It is clear for us that a good prediction model needs for external stimulus. Logically, the second and third line of improvement has to do with the quality of the information that we extract as a stimulus. In this regard, making improvements to the language model for interpreting opinions is suggested as well. In third place, we could extend the definition of crypto-influencer to other platforms and see, for example, if at Reddit we find more evidence of a link to the observed behavior.

The problem of predicting Bitcoin's price has very interesting approaches to study that we have not yet explored, such as what is its relationship with many resources of information. However, this work is undoubtedly challenging for the amount of big data to be retrieved. The knowledge acquired in this sense covers various topics, so another line of work is also proposed, our main concern is about the energetic cost of Bitcoin transactions, or more accurately ¿can we set a growth function to the energy required to

create a transaction? indeed we hope this measure must be proportional to the number of miners and the complexity of the mathematical problem to be solved at including the transaction to the blockchain, but there are some open questions ¿Could we support this function by market's prices? Looks reasonable, but what if we use influencer opinions to create virtual alarms that trigger our prediction of energy consumption? That is no that clear anymore, but also a reasonable question to further explore NLP's techniques.

APPENDIX A. INSTRUCTIONS FOR THE LABELING SESSIONS

The stance expressed about a topic is related to the sentiment that a speaker expresses in his speech, however, they are very different. On sentiment analysis, we are usually interested in know if the expressed idea is positive, negative, or neutral based on the content of the used language. Typically, the sentiment correlates strongly with the election of the words and language used in the speech. On the other hand, stance detection is defined with respect to an objective topic, so can be independent of how positive or negative is the used language. The topic even may not be mentioned directly and every topic can be related to the opinion's objective that we want to measure, for instance:

Topic: Abortion Legalization.
Tweet: The pregnant are more than walking incubators. They have rights too!
Stance: IN FAVOR.

Clearly in the last example if the topic were "Pro-life Movement" the detected stance would change to AGAINST, without modifying in any way the feeling that the text might express.

At the shared folder, you will find a file titled "INFLUENTIALTWEETS.CSV". In this labeling session, we are looking for you to identify the STANCE that every one of these tweets expresses by themselves about the virtual cryptocurrency: BITCOIN. To achieve this objective, firstly, you will be asked to identify who is the target subject of the Tweet in a column called OPINION TOWARDS, which must be completed with the integers 1, -1, 0, or 2, according to the following criteria:

- If the tweet explicitly expresses an opinion about the topic, a dimension of the topic, or a characteristic of the topic, the tweet will be classified as 1.
- If the tweet does not express an opinion with respect to the topic, but has an opinion about something or someone else related to the Bitcoin industry, the tweet will be classified with 0.
- If the tweet does NOT express explicitly an opinion, the tweet will be classified with -1.

- If you are not sure about your answer, you may complete this field with a 2.

From your first analysis, we ask you secondly to identify the STANCE that each tweet express. For the particular case of the BITCOIN currency, we want a stance to be classified as IN FAVOR if the tweet expresses an idea that supports its existence, use, and the overcrowding of its network, as well as if it presents facts that support its good behavior in the world market (for example, its acceptance as a valid currency by a bank). Similarly, a position will be classified as AGAINST if the tweet expresses an idea of rejection of the use of the currency, as well as if there is bad news regarding its valuation, prohibitions on its use, etc. In the event that a tweet does not express a specific stance, you may classify it as NONE. If you are not sure of your answer, you can complete this field with the string IDK (I do not know).

APPENDIX B. MODEL DESIGN AND HYPER-PARAMETERS TUNING FOR XGBOOST

The XGBoost model introduced in section 6 was created for the same purpose as the Neural Networks (NNs) presented in section 3. However, the training process was slightly different.

In the first place in the case of XGBoost, we fed it with just the recent observations for predicting price evolution, instead, NNs are an instance of a Recurrent Neural Network so were fed with tensors made from series over movable windows, rather than a single vector of features. On the other hand, the pre-processing applied for data remained the same, which includes the treatment of the scales.

In the second place, the results discussed in section 6 regarding the XGBoost prediction score, were obtained by tuning hyper-parameters of an instance of a XGBREGRESSOR model (from the XGBoost package in python).

TABLE B.1. Parameters in which we carry out the grid search process to select their best combination.

Parameter	Description	Explored Values	Best Combination
learning rate	Rate in which information is learned by the model	[$1e^{-2}$, $1e^{-3}$, $1e^{-4}$, $1e^{-5}$]	$1e^{-3}$
max depth	Maximum depth of the tree	[5, 10, 30, 50]	30
min child weights	Minimum sum of weights required in a child	[10, $1e^{2}$, $1e^{3}$, $1e^{4}$,]	$1e^{2}$
reg alpha	L1 regularization term on weight	[$1e^{-2}$, $1e^{-3}$, $1e^{-4}$, $1e^{-5}$]	$1e^{-3}$
reg lambda	L2 regularization term on weight	[$1e^{-2}$, $1e^{-3}$, $1e^{-4}$, $1e^{-5}$]	$1e^{-5}$
booster	The type of model to run	[gbtree, gblinear]	gbtree

To achieve this, we were carrying out a grid search over different parameter combinations. In table B.1 we show the explored parameters, their values, and the combination that maximize our model performance.

The first tuned parameter was the learning rate. This value kept very conservative on purpose, indeed the maximum learning rate explored was 1/100 because we observed that our model quickly starts to over-fit the task. On the other hand, parameters as regularizers and the min child weight feature, have also a role to play in the degree of over-fitting exhibited by the model. In this sense, we see that both regularizers, which are introduced with the objective of penalizing the loss function, remained small in their values which could be an indicator that the model at this task is not too prone to over-fitting. The L1 regularization term is two orders of magnitude bigger than L2, so the quadratic regularization term is less important. Indeed, it is known that L2 is not robust to outliers, as square terms blow up the error. Indeed, we are interested in a model capable of predict with some probability the high jumps that price exhibits.

The min child weight parameter has a different purpose, in simple, by setting this parameter we are telling the model when to stop trying to split the decision tree, i.e., once the sample size in a node goes below a given threshold. This value besides the max depth is the one that shapes the structure of the tree. The first one sets a limit about when to create a decision branch in the tree while the second one creates a threshold in the maximum number of child branches, from the origin of the tree to their leaves. The structure found has a max depth of 30 levels, so we can affirm that the decision tree still being very flexible in terms of the flow and number of decisions to make before a final prediction.

Finally, we tried the two model structures available in XGBboost to try the proposed task which we call the booster. We found that the tree-like model achieves the best performance score in terms of the RMSE error for the prediction task. It is obvious to us now that the relationship between the variables introduced in this research and the price is not linear.

APPENDIX C. NOMENCLATURE OF THE VARIABLES USED ON XGBOOST EXAMPLES

In figure 6.8 we show two Shapley values graphs for the features used to fed our XGBoost models. These features were obtained from the processes described in the previous sections, but we are aware that the nomenclature used in these figures may not be entirely explanatory. This is because in this appendix we deepen in their meaning.

All introduced features are instances of two main measures: the *moving mean* or moving average and *momentum* of the STANCE signals previously obtained from BERT. We are using the **MM** and **M** nomenclature to refer to these measures, respectively. As we well know, these measures are defined over a certain window. We represent this amount by the number of days considered in the calculus made for the measure, for example, **7.0** stands for 7 days, i.e., a window of 2016 data points or measures of size. On the other hand **0.25** represents 6 hours or 72 measures, as is a quarter of a day of 288 data points.

Finally, we totally 5 STANCE measures or signals. Three of them are the original polar NEGATIVE, POSITIVE measures and the NEUTRAL one. They are called as **NEG**, **POS** and **NEU**, respectively. From them we obtain two OVERALL stance representation, they only differ in their name by the inclusion of a **W** at the end. This is because the last one is a representation of a weighted version of the original OVERALL measure as equation C.1 shows:

$$overall_W = \frac{overall}{neutral} = \frac{positive - negative}{neutral} \tag{C.1}$$

REFERENCES

Almeida, F. and Xexéo, G. (2019). Word embeddings: A survey.

Baccianella, S., Esuli, A., and Sebastiani, F. (2010). SentiWordNet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, Valletta, Malta. European Language Resources Association (ELRA).

Bambrough, B. a. F. (2020). Bitcoin crashes back as the u.s. and iran send mixed messages.

Bandara, K., Shi, P., Bergmeir, C., Hewamalage, H., Tran, Q., and Seaman, B. (2019). Sales demand forecast in e-commerce using a long short-term memory neural network methodology.

Bindal, N. and Chatterjee, N. (2016). A two-step method for sentiment analysis of tweets. In *2016 International Conference on Information Technology (ICIT)*, pages 218–224.

Browne, R. B. a. C. (2019). Bitcoin plunges, briefly falling below $10,000, as trump slams crypto.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). Bert: Pre-training of deep bidirectional transformers for language understanding.

El Alaoui, I., Gahi, Y., Messoussi, R., Chaabi, Y., Todoskoff, A., and Kobi, A. (2018). A novel adaptable approach for sentiment analysis on big social data. *Journal of Big Data*, 5(1):12.

Gabrovšek, P., Aleksovski, D., Mozetič, I., and Grčar, M. (2017). Twitter sentiment around the earnings announcement events. *PLOS ONE*, 12(2):e0173151.

Gal, Y. and Ghahramani, Z. (2016). Dropout as a bayesian approximation: Representing model uncertainty in deep learning.

Girkar, U. M., Uchimido, R., wei H. Lehman, L., Szolovits, P., Celi, L., and Weng, W.-H. (2018). Predicting blood pressure response to fluid bolus therapy using attention-based neural networks for clinical interpretability.

Gulordava, K., Bojanowski, P., Grave, E., Linzen, T., and Baroni, M. (2018). Colorless green recurrent networks dream hierarchically.

Hochreiter, S. (1998). The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6:107–116.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9:1735–80.

Howard, J. and Ruder, S. (2018). Universal language model fine-tuning for text classification.

Jia, H. (2016). Investigation into the effectiveness of long short term memory networks for stock price prediction.

Jung, H., Han, M., Kang, M., and Hwang, S. (2018). Learning what to remember: Long-term episodic memory networks for learning from streaming data.

K, V. P., S, A., R, V., and KP, S. (2019). A deep learning approach for similar languages, varieties and dialects.

Karevan, Z. and Suykens, J. A. K. (2018). Spatio-temporal stacked lstm for temperature prediction in weather forecasting.

Ke, P., Ji, H., Liu, S., Zhu, X., and Huang, M. (2020). Sentilare: Sentiment-aware language representation learning with linguistic knowledge.

Krejzl, P., Hourová, B., and Steinberger, J. (2017). Stance detection in online discussions.

Levy, S. (2001). *Crypto: How the Code Rebels Beat the Government–Saving Privacy in the Digital Age*. Penguin USA, USA.

Li, T., Jing, B., Ying, N., and Yu, X. (2017). Adaptive scaling.

Linzen, T., Dupoux, E., and Goldberg, Y. (2016). Assessing the ability of lstms to learn syntax-sensitive dependencies.

Matta, M., Lunesu, M. I., and Marchesi, M. (2015). Bitcoin spread prediction using social and web search media.

Mehmet Levent, A. E. (2018). Analysis of the relationships between bitcoin and exchange rate, commodities and global indexes by asymmetric causality test. *Eastern Journal of European Studies*, 9:27–45.

Merity, S., Keskar, N. S., and Socher, R. (2017). Regularizing and optimizing lstm language models.

Mohammad, S. M., Kiritchenko, S., and Zhu, X. (2013). Nrc-canada: Building the state-of-the-art in sentiment analysis of tweets.

Mohammad, S. M., Sobhani, P., and Kiritchenko, S. (2017). Stance and sentiment in tweets. *ACM Trans. Internet Technol.*, 17(3).

Nakamoto, S. (2009). Bitcoin: A peer-to-peer electronic cash system. *Cryptography Mailing list at https://metzdowd.com*.

Narayanan, A., Bonneau, J., Felten, E., Miller, A., and Goldfeder, S. (2016). *Bitcoin and Cryptocurrency Technologies: A Comprehensive Introduction*. Princeton University Press, USA.

Nguyen, D. T., Sharma, S., Schulz, H., and Asri, L. E. (2018). From film to video: Multi-turn question answering with multi-modal context.

Ning, S. and Shephard, N. (2017). A nonparametric bayesian approach to copula estimation.

Pagolu, V. S., Challa, K. N. R., Panda, G., and Majhi, B. (2016). Sentiment analysis of twitter data for predicting stock market movements.

PappuRajan, A. and Victor, S. (2014). Web sentiment analysis for scoring positive or negative words using tweeter data. *International Journal of Computer Applications*, 96:33–37.

Petneházi, G. and Gáll, J. (2018). Exploring the predictability of range-based volatility estimators using rnns.

Poggio, T., Voinea, S., and Rosasco, L. (2011). Online learning, stability, and stochastic gradient descent.

Rada, D. P. (2018a). Crypto-influencers algorithm.

Rada, D. P. (2018b). Personal communication.

Radford, A., Jozefowicz, R., and Sutskever, I. (2017). Learning to generate reviews and discovering sentiment.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. (2019). Language models are unsupervised multitask learners.

Rao, V. A., Anuranjana, K., and Mamidi, R. (2020). A sentiwordnet strategy for curriculum learning in sentiment analysis.

Rauchs, M., Blandin, A., Dek, A., and Wu), Y. (2021). cambridge bitcoin electricity consumption index.

Smith, L. N. (2018). A disciplined approach to neural network hyper-parameters: Part 1 – learning rate, batch size, momentum, and weight decay.

Tian, X., Tao, D., and Rui, Y. (2011). Sparse transfer learning for interactive video search reranking. *CoRR*, abs/1103.2756.

Utsugi, A., Ino, K., and Oshikawa, M. (2004). Random matrix theory analysis of cross correlations in financial markets. *Physical Review E*, 70(2).

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need.

Wanjawa, B. W. (2016). Predicting future shanghai stock market price using ann in the period 21-sep-2016 to 11-oct-2016.

Wehbe, B., Arriaga, O., Krell, M. M., and Kirchner, F. (2018). Learning of multi-context models for autonomous underwater vehicles.

Yosinski, J., Clune, J., Bengio, Y., and Lipson, H. (2014). How transferable are features in deep neural networks?

Zhang, X., Li, Y., Wang, S., Fang, B., and Yu, P. S. (2018a). Enhancing stock market prediction with extended coupled hidden markov model over multi-sourced data.

Zhang, X., Zhang, Y., Wang, S., Yao, Y., Fang, B., and Yu, P. S. (2018b). Improving stock market prediction via heterogeneous information fusion. *Knowledge-Based Systems*, 143:236–247.

Zhong, G., Lin, X., Chen, K., Li, Q., and Huang, K. (2019). Long short-term attention.

Zhou, X., Pan, Z., Hu, G., Tang, S., and Zhao, C. (2018). Stock market prediction on high-frequency data using generative adversarial nets. *Mathematical Problems in Engineering*.