Lisbon School of Economics & Management
Universidade de Lisboa

MASTER

MATHEMATICAL FINANCE

MASTER´S FINAL WORK

DISSERTATION

PUMP IT: TWITTER SENTIMENT ANALYSIS FOR CRYPTOCURRENCY PRICE PREDICTION

VLADYSLAV KOLTUN

OCTOBER - 2022

Lisbon School of Economics & Management
Universidade de Lisboa

MASTER

MATHEMATICAL FINANCE

MASTER´S FINAL WORK

DISSERTATION

PUMP IT: TWITTER SENTIMENT ANALYSIS FOR CRYPTOCURRENCY PRICE PREDICTION

VLADYSLAV KOLTUN

SUPERVISION:

IVAN YAMSHCHIKOV

OCTOBER - 2022

Contents

Acknowledgments

I would like to start by saying that I am grateful for the opportunity to have worked on this thesis under the supervision of Ivan Yamshchikov. He was able to join his area of expertise with my personal interests and gave me a theme that allowed me to explore and learn what machine learning is and how it can be used in the financial industry. It was an amazing experience and it was very thoughtful of him to dedicate his time and effort to help me out whenever I needed.

Also, I want to express my appreciation to my mother Olena and to Adrialina, for always being by my side through this 5 year journey and helping me in so many ways. Without them, I probably would've had a much harder time and would not be where I am today.

Abstract

Predicting the prices of the cryptocurrencies can be done by only using historical data related to the price, but adding other sources of information can be beneficial. In this work, we propose to analyse the market sentiment and add that information to the models. This sentiment was analyzed across 567 thousand tweets about 12 coins to get a daily grasp of the sentiment, polarity and subjectivity of the market. The tokens were separated into classes: established, emerging and "meme" tokens. We trained various algorithms, such as OLS, LOGIT, LSTM and NHITS. Two periods were analysed: one corresponding to a bear market and one to a bull market. Due to the high intra-day volatility of cryptocurrencies, LSTM that takes longer periods into consideration did not seem to perform better than the ones without "memory", like OLS and LOGIT. NHITS was the best performing model accuracy wise, but lacked in returns, which we associated with our over-simplistic trading strategy. The information extracted from social media proved to be helpful across the range of models and coins. We successfully showed that "meme" tokens do not represent a viable investing strategy in our study. The forecasting error does not increase significantly from a bear market to a bull market, even though the market changes drastically.

Keywords: Cryptocurrency; Price Prediction; NHITS; Sentiment Analysis; Machine Learning;

1 Introduction

In recent years, crypto assets have been a hot topic in the finance and investment world. Looking at the plot below, we can see how often the word crypto has been searched on Google. There was a small spike before the middle of the 2018, when cryptocurrencies that generated yield for holders became quite popular. The path to the top began in mid 2020 and peaked in the first half of 2021, coinciding with the crypto-market bull run. The assets themselves did not represent a new financial instrument never seen before, but the fact that they were built on blockchain technology is deemed to be a major breakthrough. Nowadays, talking about cryptocurrencies specifically does not encompass the whole market. Instead one can talk about crypto assets, because not everything that exists on a blockchain and has a token is a currency. Examples of currencies are Bitcoin (BTC), Ethereum (ETH), Ripple (XRP), and many others, but there are also utility tokens, that provide the owner of said token some sort of product or service, security tokens, usually distributed during Initial Coin Offerings, as a representation of the investment made by a person into a project. Even art became a digital token in the form of an NFT (non-fungible token) and drew another segment for investors looking to get into the crypto-market.

Figure 1: Trend of the word "crypto" on Google Search

The price of a cryptocurrency, like any other price, can be viewed as a time-series and it is possible to try and predict the future cost of a certain coin based on past information about its' performance. The problem of price prediction can be interpreted as a one dimensional problem where the only driving factor is the price itself and we try to predict the price trajectory based on the historical data. Although it is a valid approach to the task and we explore this in our specific case, it rarely provides actionable insights, since, most of the times, there is more than one source of information at our disposal. Moreover, this external information that is not fully incorporated into the price due to some market inefficiency could be a valuable source of data for price prediction.

Each asset class has some sort of volatility and the magnitude of this noise varies from a class to a class. For cryptocurrencies, and crypto-assets in general, the volatility is much higher, due to several factors. It is a 24/7 market, easy to access market and, as such, it allows more trades to be executed daily by a large amount of participants, amounting to large volumes of cash-flows. Although cryptocurrencies existed for a while, it became much more popular in recent years and, therefore, gathered a large amount of first time traders. With new participants joining in, the market becomes less rational, since it is more common for inexperienced traders to put emotions first and experience FOMO (fear of missed opportunity) and FUD (fear, uncertainty and doubt) [14]. All these factors generate noise and contribute to higher volatility, but there are also macro events that drastically impact the price action of cryptocurrencies. Regulatory news tend to shake up the market with fear, major celebrities/companies endorsing a crypto-project usually drives the price up, and many others global news that might not even be related to crypto, but impact them either way. Therefore, it is very important to forecast the price

action and exploit any future opportunity.

There is evidence that the equity market has a connection with the crypto market as moves of the $S&P500 or the $DJI tend to be replicated, in a way, by $BTC, which in its turn dictates the pace of other cryptocurrencies. As shown in [9], there is a positive correlation between Bitcoin and the $S&P500, thus movements in traditional markets should be monitored and used as another source of information. An illustrative example of this was added by the author of the above mentioned paper and there is a noticeable similarity in the prices' trends and the correlation between the two assets increased significantly from 2017 to 2021.

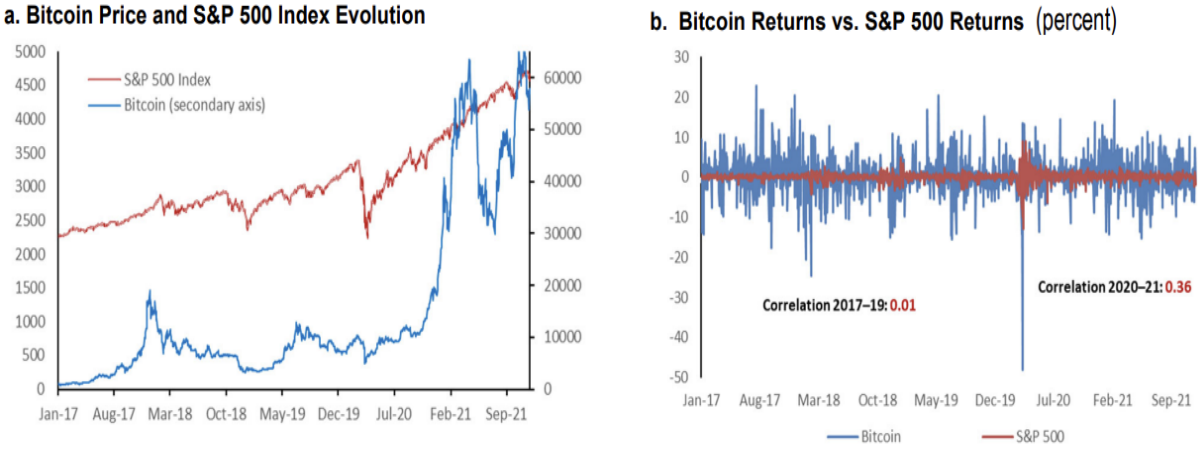

Figure 2: $BTC and $S&P500 price evolution and returns correlation

Also, it is no wonder that any asset depends on the image that the public has of it and nowadays social media dictate what this image looks like. There are several platforms, like Reddit and Discord, where there is plenty of content created everyday about crypto, but the ultimate platform for any crypto-related topic is Twitter. It is in this social network that most of the opinions are shared, news published, announcements made and many use it as their go-to source of information. Tweets spread information much faster than any newspaper and some make an impact on the markets almost instantaneously. For instance, Elon Musk's tweets on cryptocurrencies, like Bitcoin or Dogecoin, drag their prices down or up depending on the sentiment transmitted.

Figure 3: $BTC price in May-June 2021

Looking at the plot above [1], there are clear drops when there is a negative message posted by him and an upward trajectory for good news. By analysing the sentiment of his tweet on Bitcoin, one could create a long or short position on $BTC, depending whether its a positive or a negative message. Even though Elon's impact on the crypto-market is unique, due to his status on social media, there are other people tweeting and swaying the market in a particular direction in a more discrete way. Either way, it is clear that understanding the market sentiment towards a cryptocurrency is important and might prove to be useful in the predicting its price action.

Even though stocks belong to the traditional financial market and tend to exhibit a more reasonable behaviour, authors of [13] and [11] found that, here as well, the market sentiment is important and provides useful inputs in predicting the movement of stocks. More in line with our work, there is [15] which showed that using news data about Ethereum helps predicting the price fluctuations of this coin.

To grasp an understanding of the market sentiment, one has to decide what metrics to use to evaluate the mood and find a way to quantify this in order to feed it to a learning algorithm. From a single tweet, we can gather the number of likes, comments and shares that post had or we can use Natural Language Processing (NLP) algorithms to obtain the sentiment, polarity and subjectivity embedded in the text. The existing literature usually only focuses on extracting the sentiment from the tweets and this work will help us understand if this approach is incomplete or not.

Since NFTs are also an important part of the crypto industry and are largely driven by the general public's opinion of the project and art involved in each NFT, it makes sense to try and predict their valuation using Twitter data. In [12], the authors analysed what drives the valuation of these assets. They concluded that using social media information can boost the accuracy of the predictive model.

A remark must be made that, since NFTs are highly illiquid and a very recent asset class, that peaked in a blossoming bull market, the valuation itself might be misleading of the true value of these assets.

For any time-series there is a wide variety of models that can be used in order to fit the data at hand. Common models for such tasks are Autoregressive models, Moving Average models or a combination of both. Most prices tend to exhibit some sort of seasonality and, because of that, one can use a SARIMA model, just like the authors of [2] did.

Going through the literature, on the other hand, we found that most papers used the Long Short Term Memory (LSTM) model. First introduced in 1997 in [6], it was a novel machine learning algorithm that solved the problem of time consuming back propagation. Even nowadays, it is still used as a go-to model and achieves satisfying results [18]. Going trough [7], we found a similar approach to ours, but with some different aspects. The authors decided to explore the largest Chinese social media platform, Sina-Weibo, to capture the market sentiment and used it with the LSTM model to predict the prices of Bitcoin, Ethereum and Ripple. They outperformed a state of the art auto-regressive model by 18.5% in precision and 15.4% in recall. Unfortunately, in [19] it was shown that, although LSTM performs better than Naive Bayes, it was just slightly better than a simple coin toss for some tasks. On top of that, the machine learning field has evolved a lot and there was a significant progress in a wide variety of tasks, with time-series analysis being one of them.

In this thesis, we propose to use a different model that might achieve better results. In [3], the authors introduced Neural Hierarchical Interpolation for Time Series Forecasting, which we will refer as NHITS from now on.

This architecture was inspired by the Transformer model [17], but is applicable to long horizon time series prediction problems. Its design focuses on solving two problems: volatility of predictions and computational complexity.

[1] Obtained from https://www.vox.com/recode/2021/5/18/22441831/elon-musk-bitcoin-dogecoin-crypto-prices-tesla

The authors proposed to use the following concepts:

1. Multi-Rate Data Sampling: sub-sampling layers, which reduce the memory footprint and computations required, while preserving the model's capacity to capture long-range dependencies;

2. Hierarchical Interpolation: mechanism that enforces smoothness of multi-step predictions by reducing the dimensionality of neural network's prediction;

These allow for each block to specialize on forecasting its own frequency band of the time-series and the model achieved state-of-the-art results on six large-scale benchmark datasets, with exchange rate being one of them.

With this work, we intend to answer 3 questions: can the NHITS outperform the classical approaches used in the literature; does Twitter provide a useful source of information for price prediction; does the market state affect the choice of model and information used. We will compare several models according to an error metric and study how each model would perform if used as a trading tool in 2 separate time intervals, corresponding to a bear and a bull market.

2 Data and Evaluation

In this section, we explain how we obtained the data from Twitter and the prices of cryptocurrencies, what transformations we applied to it, as well as, how we evaluated the performance of each model and the models used.

2.1 Data Processing

For this thesis we used a dataset provided by the authors of [5]. They collected 576 thousand tweets on the top 12 cryptocurrencies, according to their market capitalization at the time, and the prices of these coins. For each tweet we had information on the number of likes, retweets and replies, and the text itself. As for the prices, we had data on daily high and low prices, open and close prices [2] and the daily volume traded.

Our first step was to turn all the data into a single data frame with daily data on the metrics just mentioned. In our case, the open and close prices were not important and we decided to use the following as our daily price for each cryptocurrency:

$$price = \frac{daily\ low + daily\ high}{2} \tag{1}$$

For each coin, we averaged the number of likes, retweets and replies to get daily numbers for these metrics. Lastly, we needed to extract the sentiment from the text and average it to get a numeric representation of the daily market mood.

Using Natural Language Processing it is possible to extract, in a quick way, a variety of information from a written text. Each tweet carries a sentiment with it, positive or negative, and it can provide insights into what the person feels about a certain cryptocurrency, for example. In [16], the authors introduced an alternative model to represent the English language, which was faster and lighter. It captures the sentiment by finding words or expressions commonly linked to positive or negative messages and assigns each text with values between 0 and 1 (0 being extremely negative and 1 being extremely positive). TextBlob [3] is another model used to process textual data and Vader was introduced as sentiment analysis model in [8]. Since there is an author to every tweet, we can argue that sentiment only is not enough, since we have to take into account if the person is being biased or not, subjectivity, and polarity is the measure of the overall combination of the positive and negative emotions in a sentence. We decided to use TextBlob for subjectivity and Vader for to get the polarity, since we wanted to choose the best model for each task. After gathering these scores across all tweets, we averaged them on a daily basis.

	Unnamed: 0	date	avg_reply	avg_like	avg_retweet	avg_sent	avg_subj	avg_pola	price_diff	price	price_norm	volume
0	0	2017-10-03	0.024555	0.088099	0.076437	0.941215	0.120271	0.132945	173.580078	4345.680176	0.076238	0.003670
1	1	2017-10-04	0.017847	0.072070	0.053608	0.954026	0.088497	0.122328	141.890137	4281.364990	0.075110	0.003182
2	2	2017-10-05	0.013919	0.050453	0.049602	0.931472	0.118180	0.115739	198.590332	4263.344971	0.074793	0.003310
3	3	2017-10-06	0.017140	0.050277	0.063330	0.964282	0.077474	0.139177	92.740234	4366.899902	0.076610	0.003049
4	4	2017-10-07	0.026982	0.078032	0.081806	0.937274	0.125561	0.150764	122.830078	4382.464844	0.076883	0.002584
...	...	...	...	...	...	...	...	...	...	...	...	...
1239	1239	2021-02-23	0.156492	0.462831	0.087150	0.968919	0.077981	0.118484	8914.339488	49747.760012	0.872743	0.302314
1240	1240	2021-02-24	0.142370	0.543466	0.099368	0.958695	0.082764	0.129662	4076.638533	49251.817428	0.864043	0.181485
1241	1241	2021-02-25	0.200964	0.449561	0.097886	0.960672	0.083788	0.197053	4855.113963	49521.410001	0.868772	0.155304
1242	1242	2021-02-26	0.168263	0.435177	0.082930	0.965310	0.094404	0.126812	3915.943146	46412.813687	0.814237	1.000000
1243	1243	2021-02-27	0.159421	0.433921	0.080809	0.957557	0.072934	0.164909	2984.244335	46761.147933	0.820348	0.130812

Figure 4: Bitcoin's final version data frame

[2] Cryptocurrencies trade 24/7 so the open and close prices are the prices at the opening or closing of the NYSE

[3] https://textblob.readthedocs.io/en/dev/index.html

2.2 Performance Evaluation

For each model fitted it is possible to evaluate its performance by looking at error metrics like the mean absolute error or the mean square error, or look at the accuracy and recall percentages when considering a classification problem. Though these values are reasonable and intuitive indicators of the goodness of the model, sometimes they might be misleading or show no clear best performer.

In price prediction the errors committed need to be looked into, because we need to know what happens when the model is wrong. Constructing a model for price prediction has a clear purpose, which is help the decision making in the trading process. The wrong decision on opening a long or short position might be critical even when the predicted price is just slightly off the actual truth. At the same time, a model that correctly and reliably opens a long position in a currency that goes up just a notch might and, conversely, shorts a currency that goes down in the future, might be pretty valuable despite being over-optimistic or over-pessimistic in terms of the absolute error. Therefore, in this paper, we suggest constructing some heuristic trading algorithms based on the model's predictions and comparing the profits with some baseline models.

We suggest a straightforward approach: a bot is given a certain initial wealth and is allowed to make a daily decision on whether to buy, hold or sell the coins from one of the portfolios. All wealth allocations are subject to weights, calculated based on traded volume. The following diagram explains the decision algorithm we implemented.

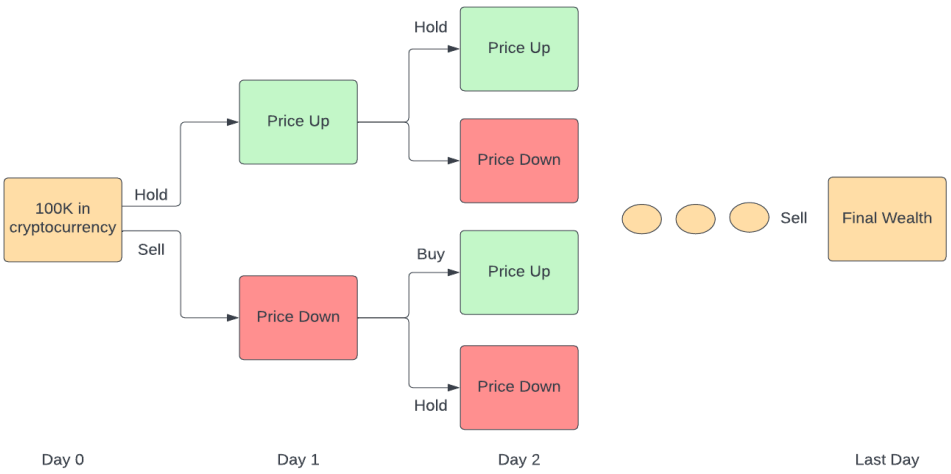

Figure 5: Trading Algorithm

The performance of the model was judged on the performance of the corresponding bot as well as in terms of the overall prediction error. This way we can evaluate the goodness of the fit in a more realistic scenario on top of the traditional error metrics.

3 Model Description

In this section, we will present a short description of the models considered throughout this work, explain some of the parameter choices made when fitting the models and discuss the pros and cons of each one of them.

3.1 Ordinary Least Squares Regression

The OLS regression, introduced in [4], is a common technique for estimating coefficients of linear regression equations which describe the behaviour of one variable in terms of n other. If we have n features, the model equation can be written as:

$$Y = \beta_0 + \Sigma_{j=1}^{n} \beta_j X_j + \epsilon \tag{2}$$

Where y is the target variable, β_0 is the intercept, X_j stands for the $j-th$ feature and ϵ is a random variable with mean 0 and variance σ^2.

Least squares is the method used to compute the optimal model coefficients. It is done by minimizing the sum of square differences between the observed and predicted values. OLS requires the following assumptions to be made about the data:

1. Individuals or observations are independent;
2. Variance is homogeneous;
3. Residuals follow a normal distribution;

It is an easy model to implement, computationally efficient and highly interpretable. The requirements to use it are general enough to be applicable to a plethora of datasets. On the other hand, a linear approximation for irregular trajectories might perform poorly. Also, since OLS uses matrix inversion, when using a high number of features, the model will choose to ignore some, resulting in a loss of information.

In our case, we decided to use this model using lag 1, because we wanted to try to predict the price tomorrow solely based on the information available today, which in general is a quite difficult task.

3.2 Logistic Regression

A LOGIT regression, firstly proposed in [1], analyses the relationship between one or a set of independent variables and classifies the data into discrete classes. Unlike a regular regression, this model is used for classification problems, where the target variable is a binary one, like True/False, 1/0 or Yes/No. In order to do so, the model estimates the probability of an observation belonging to a specific category or not. As we already stated, we are trying to predict prices, which is not a dichotomous variable. However, we can adjust the problem setting by trying to predict whether the price will go up or down tomorrow based on the information we have today.

```
def label(array):
    labels=[0]
    for i in range(1,len(array)):     #labels 1 if the price decreases and 0 if it goes up
        if array[i]>array[i-1]:
            labels.append(1)
        else:
            labels.append(0)
    return labels
```

Figure 6: Code for the labelling of the price data

In this snippet of code from the LOGIT model, we illustrate how we transformed the data in order to be able to apply this model. We evaluate if the price on the i-th day is bigger or smaller than on $i-1$-th.,.. day, thus labelling it 1 or 0.

LOGIT has several advantages in the field, because it is an easy to implement model which can achieve satisfying results if the dataset is linearly separable. It also provides valuable insights about the data, such as how relevant or appropriate an independent variable is and also reveals the direction of their association. Also, we can evaluate important metrics like accuracy, recall and precision, which are important in the field we are working on.

On the other hand, it follows a "majority principle", which works well in general, but does not respond well to outliers. It requires a sufficiently large sample size and the observations should be independent. The model assumes linear relationship of independent variables, which most likely is not true.

3.3 Long Short Term Memory

Long short term memory model or LSTM belongs to the class of Recurrent Neural Network (RNN) models. These models receive an input, pass it on to a neural network, return an output and a loop allows the information to be passed on to the next step. RNN are capable of connecting past information to the present task, only if the gap between them is small enough. LSTM was created to circumvent the long term dependency problem by changing the architecture of the neural network. In most RNNs, the repeating block is quite simple, often made of a single layer. LSTMs, on the other hand, have a more intricate module, containing 4 neural network layers, interacting with each other. [10]

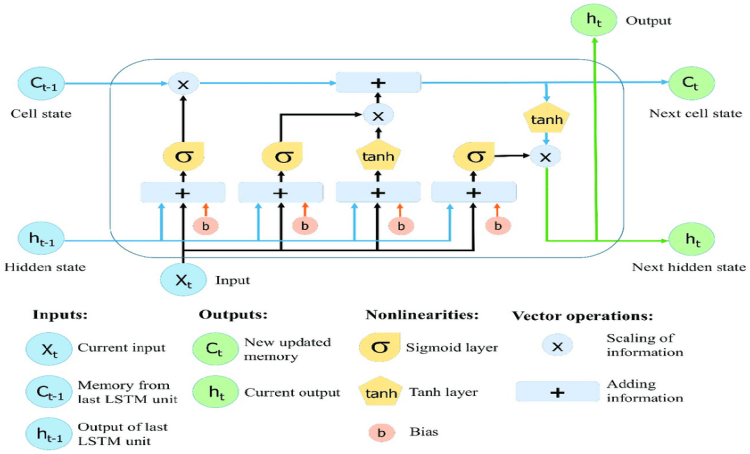

Figure 7: LSTM cell diagram

The core idea behind LSTM is to have a cell state that goes through the top of the diagram and runs straight through the entire chain. It has gates, sigmoid functions, that decide what new information is added to or ignored by the cell state. Each initialization begins with an evaluation of what to keep from the previous cell C_{t-1} by looking at h_{t-1} and x_t (first sigmoid). Next it decides what new information is going to be stored in the cell state (second sigmoid and first tanh) and update the current one. Finally, the cell decides what to output to the next iteration by filtering the current information contained in the cell (third sigmoid and second tanh).

By design, it is capable of solving the vanishing gradient problem and the long term dependency present in RNNs. Because of that, it is commonly used for sequential data like text or time series. There are some drawbacks: the complexity of the model requires high computational power; data mining tasks require even longer memory; LSTMs get affected by different random weight initialization and have a large number of parameters to be chosen; they also

have a tendency to over fit the data.

3.4 Neural Hierarchical Interpolation for Time Series Forecasting

The N-HITS model, introduced in [3], is rather recent and tackles the long term forecasting problem. The authors created a model that incorporates a novel hierarchical interpolation and multi-rate data sampling techniques.

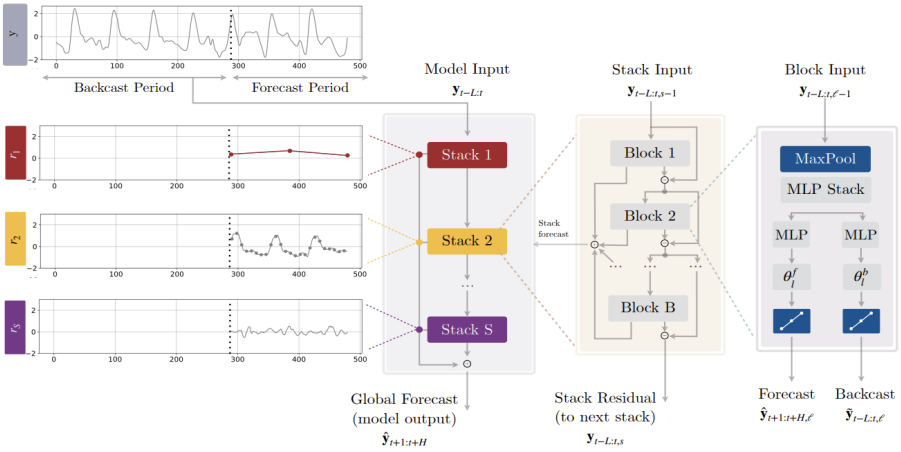

Figure 8: NHITS architecture diagram

The above diagram, taken from the original paper on this model, summarizes the architecture of the model. It is composed of several MLPs [4], which learns to compute coefficients for the backcast and forecast outputs of its basis with ReLU nonlinearities. The backcast output cleans the inputs of subsequent blocks, while the forecasts are joined together to form the final prediction. These blocks are grouped into stacks, each specialized in learning a particular characteristic of the data using a different set of basis functions.

The multi-rate input pooling, hierarchical interpolation and backcast residual connections mechanisms reduce the overall computational and memory costs, while improving the model's accuracy.

[4] MLP stands for multilayer perceptron

4 Results

We decided to first evaluate what features provided the best results across the 2 metrics previously mentioned: RMSEP and Returns, and then we compared the results of the best representatives of each model between themselves. We compare all results across the two time periods: bull and bear markets. Returns for each model were computed for 3 separate portfolios:

1. Established - composed by Bitcoin, Ethereum, Litecoin, Ripple and Monero
2. Emerging - composed by Cardano, Binance and Link
3. Meme - composed by Doge, Iota and Stellar

This way we were able to build our baseline to evaluate the NHITS' performance and evaluate the models performance depending on portfolio type and market state. The formula used to calculate the error committed was:

$$RMSEP = \sqrt{\frac{\Sigma_{i=1}^{n}(y_i - \hat{y}_i)^2}{n}} \tag{3}$$

We built our baseline consisting of the OLS with lag 1 and LOGIT, using price, volume and average sentiment as features, based on their performance. Both achieved lower error and higher returns than the traditional LSTM. NHITS was able to outperform our baseline in term of accuracy, but underperformed in returns. We found that some cryptocurrencies show a stronger dependency of the social media signal, while established tokens were far less determined by such signal.

4.1 OLS with lag 1

OLS1 BULL			
RMSEP	OLS-1-All	OLS-1-P	OLS-1-S
ada	10,08%	**9,10%**	9,15%
bnb	5,76%	**5,40%**	5,60%
btc	3,73%	**3,57%**	3,62%
doge	10,85%	11,09%	**10,52%**
eth	4,53%	4,39%	**4,37%**
iota	7,41%	**5,46%**	7,06%
link	10,14%	**9,14%**	10,27%
ltc	4,85%	**4,62%**	4,63%
trx	8,82%	8,23%	**8,22%**
xlm	9,05%	**8,32%**	8,40%
xmr	4,72%	9,70%	**4,70%**
xrp	15,88%	15,63%	**15,41%**

OLS1 BEAR			
RMSEP	OLS1-ALL	OLS1-P	OLS1-S
ada	7,53%	**4,78%**	6,42%
bnb	5,32%	**4,54%**	4,96%
btc	3,35%	3,22%	**3,20%**
doge	14,37%	**13,22%**	14,83%
eth	4,35%	4,20%	**4,11%**
iota	8,70%	**4,66%**	8,33%
link	**5,02%**	5,03%	5,04%
ltc	8,59%	**5,03%**	5,22%
trx	28,80%	**23,15%**	27,94%
xlm	12,99%	**11,09%**	11,33%
xmr	3,80%	**3,69%**	3,83%
xrp	12,54%	**11,59%**	13,10%

Table 1: Results for OLS with lag 1

The numbers on the left represent the RMSEP for OLS with lag 1 during a bull run fitted on all the variables (OLS-1-All), or on price and volume only (OLS-1-P) or on price, volume and average sentiment (OLS-1-P+S) and on the right we have the same, but for a bear run. In the case of OLS with lag 1, we found that the model using price and volume performed slightly better in both time periods if we are only looking at the error metric. It is important to note that adding sentiment achieved a similar performance and we needed to look at the returns to better understand which is definitely better.

Here as well, we saw similar results, but we observed slight improvements in some cases when sentiment was used. Therefore, we included OLS1 with sentiment in our baseline.

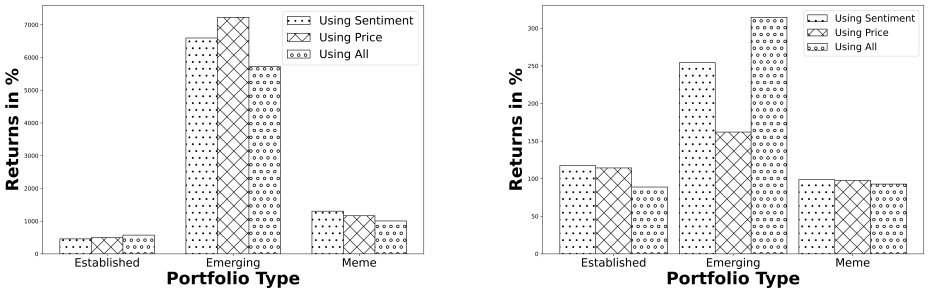

Figure 9: Returns in % using OLS1 for each portfolio type during a bull state on the right and bear state on the left.

4.2 LOGIT with lag 1

Unlike the OLS regressions, LOGIT is used for classification tasks and the MSE metric for error does not work in those scenarios. Therefore, we used the accuracy to help us evaluate the performance of each subset of features.

LOGIT BULL			
Accuracy	LOGIT-All	LOGIT-P	LOGIT-S
ada	**59,95%**	54,30%	59,41%
bnb	**56,99%**	56,99%	56,18%
btc	50,81%	52,69%	**52,69%**
doge	52,42%	54,30%	**55,38%**
eth	58,60%	55,91%	**58,60%**
iota	**55,38%**	54,57%	55,11%
link	54,57%	**56,45%**	55,11%
ltc	53,23%	51,61%	**53,49%**
trx	**54,84%**	54,03%	53,76%
xlm	56,18%	55,65%	**57,26%**
xmr	57,53%	56,72%	**59,68%**
xrp	**55,65%**	52,42%	54,84%

LOGIT BEAR			
Accuracy	LOGIT-All	LOGIT-P	LOGIT-S
ada	60,00%	58,92%	**60,54%**
bnb	60,54%	**61,62%**	61,08%
btc	56,22%	58,11%	**58,11%**
doge	**56,22%**	54,59%	54,59%
eth	**60,27%**	60,00%	59,73%
iota	**54,59%**	54,59%	54,32%
link	**56,49%**	54,05%	55,41%
ltc	54,32%	**59,73%**	59,19%
trx	52,43%	54,05%	**54,32%**
xlm	**57,84%**	55,68%	57,57%
xmr	56,76%	58,38%	**59,73%**
xrp	55,95%	51,89%	**56,49%**

Table 2: Accuracy for LOGIT with lag 1 for both market types

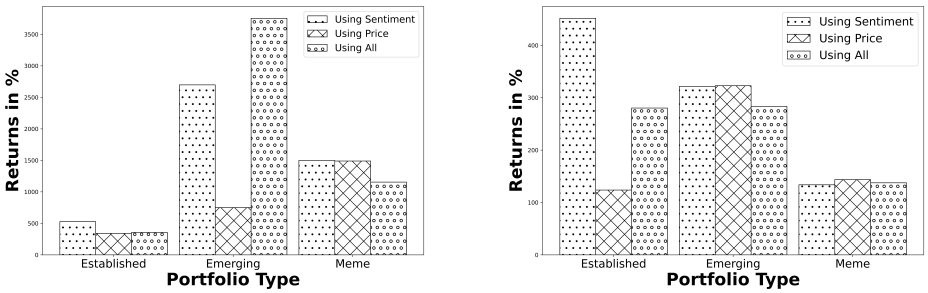

Figure 10: Returns for each portfolio type in % for LOGIT for bull on the left and bear on the right.

From the tables above, we saw a near tie between LOGIT-1-ALL and LOGIT-1-P+S and there is only a couple of instances where the model with volume and price performed better. It was clear that having at least the market sentiment from Twitter improves the accuracy and this fact was sustained by the returns in 10. Independently of the market state, we saw a clear advantage in adding sentiment information for established portfolios. The meme portfolio was unfazed by the change in the number of variables used. There was a remarkable spike in

returns when using sentiment with the emerging portfolio during a bull market, which didn't happen during the bear run. We decided to use LOGIT with price, volume and sentiment as the second element of our baseline.

4.3 Model comparison

With the best features chosen for each model in our baseline, we moved forward to compare the results across all models considered in this work. Even though LSTM is commonly used in the literature, we were not able to improve the baseline accuracy. We believe that one of the two explain this behaviour: either the high intra-day volatility of cryptocurrency prices or the data-set is too small. NHITS, on the other hand, outperformed all previously considered models and achieved the highest accuracy in both time periods.

BULL RMSEP Comparison between Models			
RMSEP	OLS-1-S	LSTM-ALL	NHITS-ALL
ada	9,15%	46,2%	**5,57%**
bnb	5,60%	33,4%	**5,14%**
btc	3,62%	37,6%	**3,60%**
doge	10,52%	53,9%	**8,15%**
eth	4,37%	67,0%	**4,36%**
iota	**7,06%**	38,9%	7,53%
link	10,27%	76,0%	**5,74%**
ltc	4,63%	30,8%	**4,52%**
trx	8,22%	77,4%	**4,75%**
xlm	8,40%	305,0%	**5,81%**
xmr	4,70%	39,3%	**4,30%**
xrp	15,41%	21,3%	**5,99%**

Table 3: RMSEP in % comparison between OLS1, LOGIT, LSTM and NHITS during a bull run

BEAR RMSEP Comparison between Models			
RMSEP	OLS1-P	LSTM-ALL	NHITS-ALL
ada	**4,78%**	46,1%	5,62%
bnb	4,54%	33,4%	**4,03%**
btc	3,22%	37,6%	**3,16%**
doge	13,22%	53,8%	**4,38%**
eth	4,20%	66,9%	**3,80%**
iota	**4,66%**	38,9%	5,03%
link	5,03%	75,8%	**4,85%**
ltc	5,03%	30,8%	**3,72%**
trx	23,15%	77,3%	**5,25%**
xlm	11,09%	304,6%	**10,82%**
xmr	**3,69%**	39,2%	4,02%
xrp	11,59%	21,3%	**3,18%**

Table 4: RMSEP in % comparison between OLS1, LOGIT, LSTM and NHITS during a bear run

Even though LSTM is one of the most prevalent model for time series analysis and is commonly used in the literature, we did not find any evidence that this model performs better than the OLS and LOGIT regressions. On one hand, we can argue that the autoregressive processes, present in both, were not able to capture correlation. On the other, it is possible that the results would improve if the parameters chosen were different.

As stated before, we also need to look into the returns generated by each model to assess their applicability as a trading tool. From the bar charts found below, we were able to conclude that our baseline outperforms significantly all others when we consider the emerging portfolio

and bull market or established portfolio and bear market. For some portfolios it performed quite well, but OLS1 and LOGIT showed stronger results in most cases. Also, it is important to remark that NHITS performed better than LSTM in all cases.

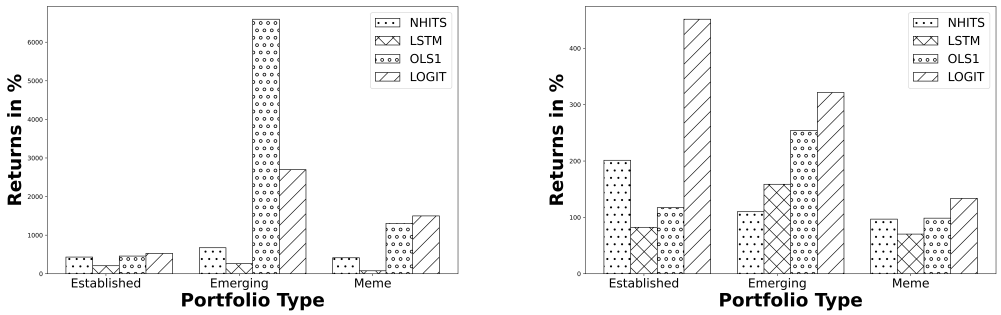

Figure 11: Returns in % achieved by each model separated by portfolio type during the bull market period on the left and bear market on the right

A small mistake in the wrong direction means a possible missed opportunity of entering a short or long position and a big mistake in the right direction does not affect the overall trading strategy. Therefore, we believe that this alone could explain the differences in the returns obtained. If we consider that there might be "luck" involved when evaluating the performance of model with a simplistic trading algorithm, then we should conclude that NHITS is the best performer according to the error metric. On the other hand, the purpose of predicting price is to invest and generate wealth. According to this, we would be better off by following the LOGIT or OLS1 model. For the next part, we used the results obtained for NHITS to further explore what else can we extract.

4.4 Further investigations

We carried out our work considering a bear market period and a bull market. As expected, we were able to achieve lower errors in a bear market and higher returns in the middle of the bull run 12 and 13. It is easier to predict tokens during a bear market, with $XLM being the only outlier. More importantly, the differences in the prediction errors do not increase significantly from a slow market to the fast paced bull period.

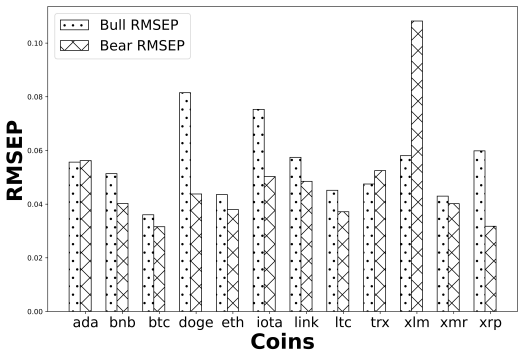

Figure 12: RMSEP % contrast between the bull and bear periods for each coin analysed

Even though a bull market is trickier to predict, because not every coin will appreciate in the same explosive manner as we witness in the latest bull run and some might even go to 0. In

our case, we verified that returns doubled for established tokens and the impact on emerging and meme portfolios was even more significant. Also, we managed to obtain higher returns in a bear market using more stable tokens, but we were better off investing in the emerging portfolio during the bull run. From our analysis, we did not find any justification to enter a portfolio consisting of meme coins, since the returns obtained for this portfolio do not justify the risk taken.

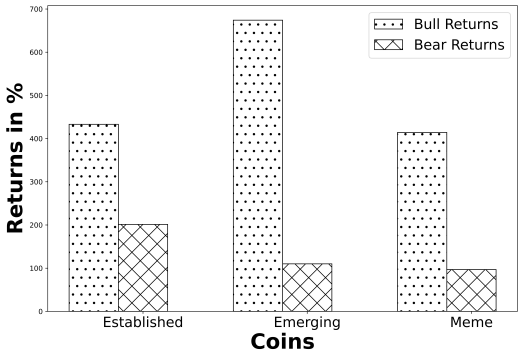

Figure 13: Returns in % comparison during both periods for different portfolios

According to plots 14 and 15, for similar levels of daily average traded volume, we obtained lower errors for established tokens and higher ones for meme coins, independent of the general market state. As expected, throughout both periods, $BTC and $ETH were the easiest to predict, while $DOGE and $XLM, both meme coins, proved to be harder to forecast correctly.

In line with these, we found that, by looking at 16 and 17, the smallest errors corresponded to the lower volatility tokens, with some exceptions in each market. An almost straight line related these in the bull run, with Bitcoin at the bottom and Doge on top, as expected. Although we considered $XRP as an established token, it suffered a major fall during the bull market run, due to its on-going law-suit with the SEC. For the bear period, the tokens were scattered, not forming any clear line, with established tokens being in the down-left corner and Doge and Stellar far away from the rest. With no surprise, we found meme coins spread across the plots, since their price action is largely driven by hype, "FOMO" and "FUD", thus becoming highly volatile.

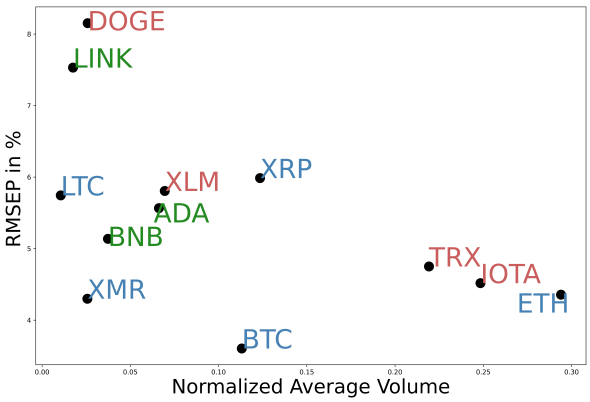

Figure 14: Error metric against the daily average volume of each token during a bull market

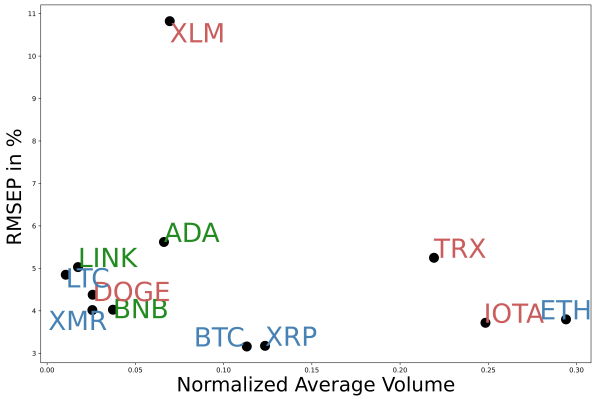

Figure 15: Error metric against the daily average volume of each token during a bear market

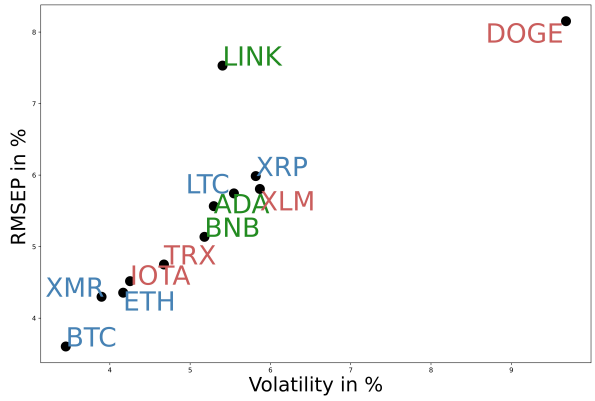

Figure 16: Error and actual volatility comparison during the bull period

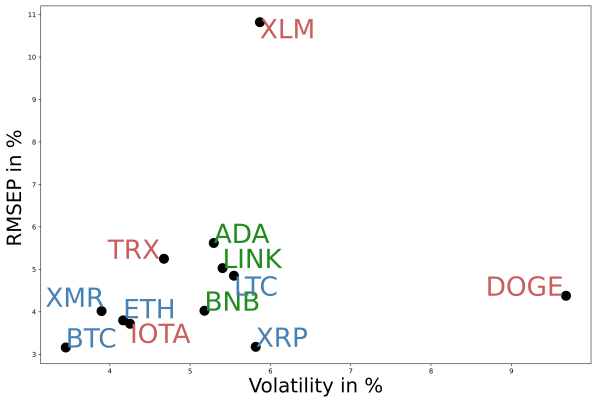

Figure 17: Error and actual volatility comparison during the bear period

5 Conclusion

We set out to study the importance of the market sentiment for cryptocurrency price prediction. We managed to extract that sentiment, using pre-trained Natural Language Processing algorithms, from tweets about these currencies. With that information, we applied several different models to our data: OLS, LOGIT, LSTM and NHITS. To compare the models, we used RMSEP for the error and a trading bot to compare returns based on the predictions made. We conducted our research in two types of market, bull and bear, and split our tokens into portfolios according to their "reputation".

We saw that using daily sentiment, on top of price and volume, we could obtain lower RMSEP and higher returns in most cases. The OLS with lag 1 and LOGIT were the best models, beating the traditional approach LSTM and NHITS, in terms of returns generated, but NHITS was much more accurate. Unfortunately, the latter was not capable of achieving better returns, which we associated with "bad luck", easily fixed if we allow the trading decision to be more frequent. For that we would need to adapt and enlarge the dataset used. We speculate that the results for LSTM were unsatisfactory, because it has an auto-regressive component, which was not able to capture the intra-day volatility or it was related to the parameters chosen.

Besides studying which model was best, we also dug deep in the results we obtained for that model. Firstly, we analysed how the model performed in forecasting the prices of each coin and concluded that established token were easier to predict, the error between the two periods does not vary much for each coin and the returns were higher during the bull run, as it was expected to happen. Notably, we were able to conclude that there is no justification in investing into a meme portfolio, as its higher unpredictably was not rewarded as much as the other 2 in any of the two time intervals.

We were successful in showing that for the same level of average trade volume, we obtain higher accuracy for established tokens, like Bitcoin and Ethereum. A clear correlation was found between volatility and RMSEP during the bull market, with the only outlier being Link, which did not occur in the bear market, where the tokens were spread out. Throughout both time periods, $BTC had the lowest volatility and error, which reinforces its dominance and stableness in the cryptocurrency market.

To sum it all up, the market sentiment improves the forecasting capabilities of any model and that impact varies according to the market cap and type of the currencies. In any market, bearing the risk of holding and trading highly volatile currencies, that rely on the social media hype and market sentiment, does not represent the best strategy. We took the daily average for all features, so our work could be further improved if the dataset was enlarged to allow hourly data points.

List of Figures

List of Tables

References

[1] Berkson, J. Application of the logistic function to bio-assay. *Journal of the American statistical association 39*, 227 (1944), 357–365.

[2] Cabanilla, K. I. M. The future of cryptocurrency: Forecasting the bitcoin-philippine peso exchange rate using sarima through tramo-seats.

[3] Challu, C., Olivares, K. G., Oreshkin, B. N., Garza, F., Mergenthaler, M., and Dubrawski, A. N-hits: Neural hierarchical interpolation for time series forecasting. *CoRR abs/2201.12886* (2022).

[4] Galton, F. Regression towards mediocrity in hereditary stature. *The Journal of the Anthropological Institute of Great Britain and Ireland 15* (1886), 246–263.

[5] Garg, A., Shah, T., Jain, V. K., and Sharma, R. Cryptop12: A dataset for cryptocurrency price movement prediction from tweets and historical prices. In *2021 20th IEEE International Conference on Machine Learning and Applications (ICMLA)* (2021), IEEE, pp. 379–384.

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

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

[8] Hutto, C., and Gilbert, E. Vader: A parsimonious rule-based model for sentiment analysis of social media text. In *Proceedings of the international AAAI conference on web and social media* (2014), vol. 8, pp. 216–225.

[9] Iyer, T. *Cryptic connections: spillovers between crypto and equity markets*. International Monetary Fund, 2022.

[10] Le, X. H., Ho, H., Lee, G., and Jung, S. Application of long short-term memory (lstm) neural network for flood forecasting. *Water 11* (07 2019), 1387.

[11] Li, X., Xie, H., Chen, L., Wang, J., and Deng, X. News impact on stock price return via sentiment analysis. *Knowledge-Based Systems 69* (2014), 14–23.

[12] Luo, J., Jia, Y., and Liu, X. Understanding nft price moves through social media keywords analysis. *arXiv preprint arXiv:2209.07706* (2022).

[13] Mittal, A., and Goel, A. Stock prediction using twitter sentiment analysis. *Standford University, CS229 (2011 http://cs229. stanford. edu/proj2011/GoelMittal-StockMarketPredictionUsingTwitterSentimentAnalysis. pdf) 15* (2012), 2352.

[14] Nagel, P. Psychological effects during cryptocurrency trading, 2018.

[15] Nguyen, T. H., Shirai, K., and Velcin, J. Sentiment analysis on social media for stock movement prediction. *Expert Systems with Applications 42*, 24 (2015), 9603–9611.

[16] Sanh, V., Debut, L., Chaumond, J., and Wolf, T. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108* (2019).

[17] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need. In *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., Curran Associates, Inc.

[18] Vo, A.-D., Nguyen, Q.-P., and Ock, C.-Y. Sentiment analysis of news for effective cryptocurrency price prediction. *International Journal of Knowledge Engineering 5*, 2 (2019), 47–52.

[19] Wong, E. L. X. Prediction of bitcoin prices using twitter data and natural language processing.

Appendices

A Additional graphics

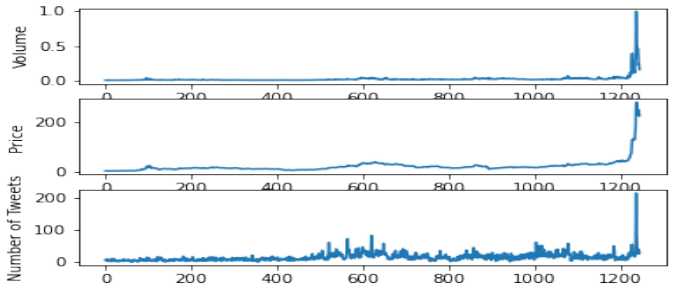

Figure 18: Average Daily Volume vs Daily Number of Tweets for $BNB

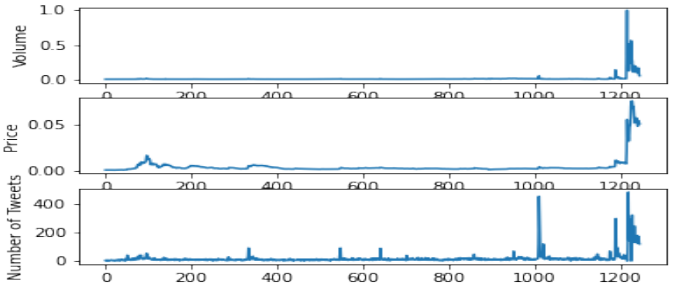

Figure 19: Average Daily Volume vs Daily Number of Tweets for $DOGE

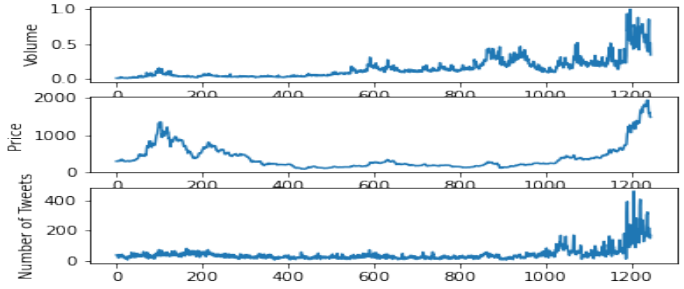

Figure 20: Average Daily Volume vs Daily Number of Tweets for $ETH

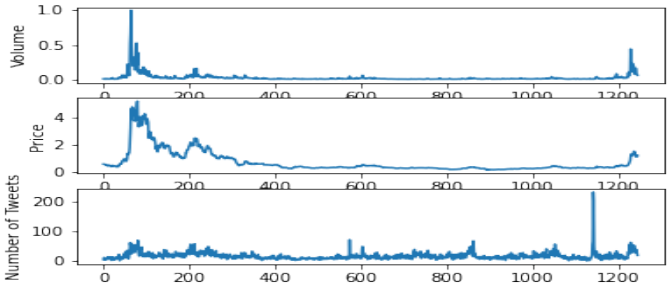

Figure 21: Average Daily Volume vs Daily Number of Tweets for $IOTA

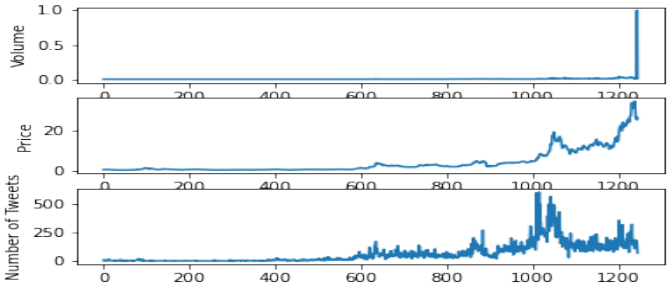

Figure 22: Average Daily Volume vs Daily Number of Tweets for $LINK

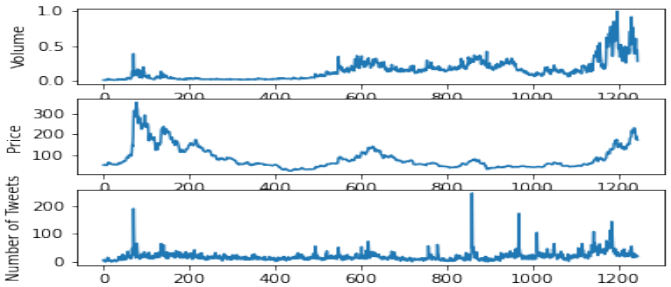

Figure 23: Average Daily Volume vs Daily Number of Tweets for $LTC

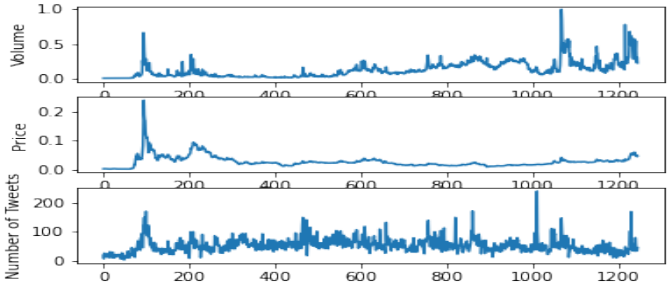

Figure 24: Average Daily Volume vs Daily Number of Tweets for $TRX

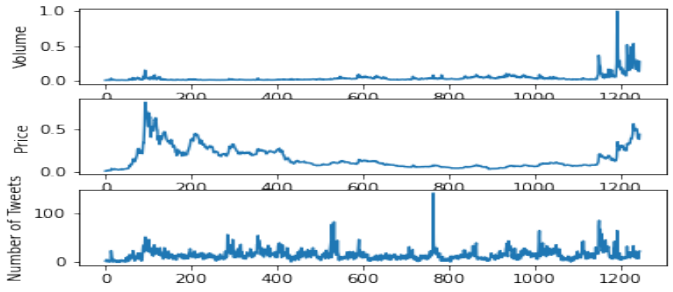

Figure 25: Average Daily Volume vs Daily Number of Tweets for $XLM

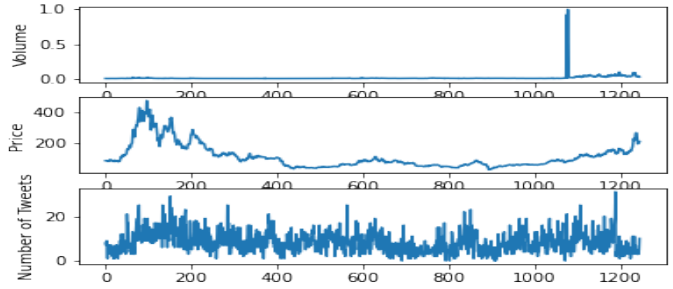

Figure 26: Average Daily Volume vs Daily Number of Tweets for $XMR

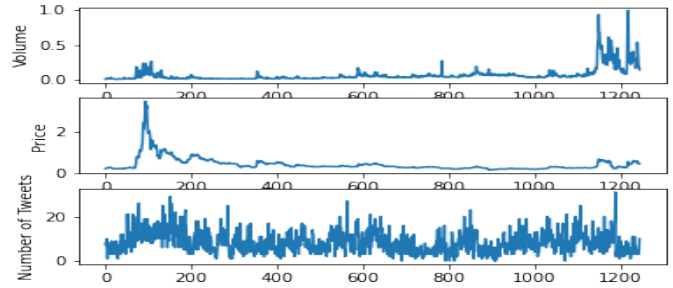

Figure 27: Average Daily Volume vs Daily Number of Tweets for $XRP

B Code Snippets

In the following pages we added the code we used to train and test our OLS with lag 1 and the NHITS. Both were written using Python language and took advantage of **neuralforecast** module [5] to implement the NHITS.

[5] https://pypi.org/project/neuralforecast/

In [3]:

```
import pandas as pd
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, mean_absolute_error    #import necessary modules
import statsmodels.api as sm
import matplotlib.pyplot as plt
import random

def splitter(frame):
    train=frame.iloc[0:871][:]
    test=frame.iloc[871:][:]     #split into traning and test data (70%-30%)
    return train, test

def sell(amount,price):
    wealth=amount*price
    return wealth
                                 #defines the sell and buy function used in the trading algorithm
def buy(w,price):
    coin_amount=w/price
    return coin_amount

def dailyOLS(coin):

    coin_results=[coin]     #a list for the coin results

    df=pd.read_csv('frame_model_'+coin+'.csv')     #obtain the data frame for a particular coin

    train,test=splitter(df)

    df['returns'] = (np.log(df['price'] / df['price'].shift(-1)))
    daily_std = np.std(df['returns'])                    #calculates the daily volatility
    annual_std = std = daily_std * 365 ** 0.5

    #For X we have 3 different possible sets of features: price and volume; sentiment, price and volume; or aLL features;

    #X_train=train.reindex(['volume','price_norm'],axis='columns')
    X_train=train.reindex(['avg_sent','volume','price_norm'],axis='columns')
    #X_train = train.reindex(['avg_reply','avg_like','avg_retweet','avg_sent','avg_subj','avg_pola','volume','price_norm'], axis="columns")
    X_train = X_train.drop(870)
    y_train = np.array(train['price_norm'][1:])     #select the important columns for X and y

    #X ends 1 entry before and y starts 1 entry after

    model = sm.OLS(y_train, X_train)
    results = model.fit()               #training and fitting of the model

    #X_test=test.reindex(['volume','price_norm'],axis='columns')
    X_test=test.reindex(['avg_sent','volume','price_norm'],axis='columns')
    #X_test = test.reindex(['avg_reply','avg_like','avg_retweet','avg_sent','avg_subj','avg_pola','volume','price_norm'], axis="columns")
    X_test = X_test.drop(1243)
    y_test = test['price_norm'][1:]

    y_pred=results.predict(X_test)    #predicted price vector

    volume=np.mean(X_test['volume'])
    #mse=mean_squared_error(y_test,y_pred)
    rmsep=np.sqrt((np.sum(((np.array(y_test)-np.array(y_pred))/np.array(y_test))**2))/371)    #Error metrics
    mae=mean_absolute_error(y_test,y_pred)
    coin_results.append(np.mean(y_test))

    coin_results.append(rmsep)
    coin_results.append(mae)
    portfolio=[100000,0]   # [wealth,number of tokens]

    prediction=np.array(y_pred)

    y_test_returns = (np.log(y_test / y_test.shift(-1)))
    daily_std_test = np.std(y_test_returns)
    annual_std_test = std = daily_std_test * 365 ** 0.5     #actual volatility during test period

    y_pred_returns = (np.log(y_pred / y_pred.shift(-1)))
    daily_std_pred = np.std(y_pred_returns)                 #predicted volatility during test period
    annual_std_pred = std = daily_std_pred * 365 ** 0.5

    portfolio[1]=buy(portfolio[0],df['price'][870])    #buy coin
    portfolio[0]=0
    value=[100000]

    for i in range(0,371):
        if portfolio[0]!=0:
            value.append(portfolio[0])
        else:
            value.append(sell(portfolio[1],df['price'][870+i]))

        if prediction[i+1]<prediction[i] and portfolio[1] != 0:
            portfolio[0]=sell(portfolio[1],df['price'][870+i])    #if tomorrow price goes down and i have coins
            portfolio[1]=0

        if prediction[i+1]>prediction[i] and portfolio[0] != 0:
            portfolio[1]=buy(portfolio[0],df['price'][870+i+1])   #if tomorrow price goes up and i dont have coins
            portfolio[0]=0

    if portfolio[1]!=0:
        portfolio[0]=sell(portfolio[1],df['price'][1240])        #transform everything to cash
        portfolio[1]=0

    value.append(portfolio[0])
    value=np.array(value)
    profit=portfolio[0]-100000

    coin_results.append(profit)

    profits=[]

    for k in range(0,10):
        portfolio_2=[100000,0]                          #inital portfolio
        portfolio_2[1]=buy(portfolio_2[0],df['price'][870])  #buy coin
        portfolio_2[0]=0
        rand_decision=[]
        for i in range(0,372):                          #random decision list
            rand_decision.append(random.randint(0,1))

        for i in range(0,371):
            if rand_decision[i+1]==0 and portfolio_2[1] != 0:
                portfolio_2[0]=sell(portfolio_2[1],df['price'][870+i])   #if tomorrow the price is down and i have coins
                portfolio_2[1]=0
            if rand_decision[i+1]==1 and portfolio_2[0] != 0:
                portfolio_2[1]=buy(portfolio_2[0],df['price'][870+i+1])  #if tomorrow price goes up and i dont have coins
                portfolio_2[0]=0
        if portfolio_2[1]!=0:
            portfolio_2[0]=sell(portfolio_2[1],df['price'][1240])
            portfolio_2[1]=0
        profit_2=portfolio_2[0]-100000
        profits.append(profit_2)

    coin_results.append(np.mean(profits))
    coin_results.append(np.mean(volume))
    coin_results.append(daily_std)
    coin_results.append(annual_std)
    coin_results.append(daily_std_test)
    coin_results.append(daily_std_pred)
    coin_results.append(annual_std_test)
    coin_results.append(annual_std_pred)

    return coin_results, value
```

In [ ]:

```
!pip install neuralforecast datasetsforecast

import torch
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

from ray import tune

from neuralforecast.auto import AutoNHITS
from neuralforecast.core import NeuralForecast
                                                                        #install and import the necessary modules
from neuralforecast.losses.pytorch import MAE
from neuralforecast.losses.numpy import mae, mse
from datasetsforecast.long_horizon import LongHorizon

import logging
logging.getLogger("pytorch_lightning").setLevel(logging.WARNING)

from google.colab import drive
drive.mount('/content/drive')

torch.cuda.is_available()


def auto_nhits_bull(coin):         #def of the nhits to run through all the coins for the bull market

    coin_results=[coin]

    df=pd.read_csv('/content/drive/MyDrive/Crypto_Thesis/frame_model_'+coin+'.csv')
    df['unique_id']=df['Unnamed: 0'].map(lambda x:coin)
    Y_df = df[['date', 'price_norm']].copy()
    Y_df['unique_id'] = df['unique_id']
    Y_df['ds'] = Y_df['date']
    Y_df['y'] = Y_df['price_norm']                          #define y var
    Y_df = Y_df.drop(['date','price_norm'],axis=1)
    X_df = df.drop(['Unnamed: 0','price','price_diff'],axis=1)
    X_df['ds'] = X_df['date']
    X_df = X_df.drop(['date'],axis=1)
    #X_df = X_df[['unique_id','ds','price_norm','avg_sent','volume']]
    X_df = X_df[['unique_id','ds','avg_like','avg_retweet','avg_reply','avg_sent','avg_pola','volume']]    #choose what vars we include
    Y_df = Y_df[['unique_id','ds','y']]
    Y_df['ds'] = pd.to_datetime(Y_df['ds'])


    df2=pd.concat([Y_df,X_df],axis=1,join='inner',ignore_index=True)
    df2=df2.drop([3,4],axis=1)
    df2.columns = ['unique_id','ds','y','avg_like','ave_retweet','avg_reply','avg_sent','avg_pola','volume']
    df2['ds']=pd.to_datetime(df2['ds'])

    #all the previous step serve to prepare the df

    n_time = len(Y_df.ds.unique())
    val_size = int(.298 * n_time)           #number of points for validation and test
    test_size = int(.298 * n_time)

    Y_df.groupby('unique_id').head(2)

    horizon = 7 #weekly horizon

    nhits_config = {
       "learning_rate": tune.choice([1e-3]),                                     # Initial learning rate
       "max_steps": tune.choice([100]),                                          # Number of SGD steps was 1000
       "input_size": tune.choice([3 * horizon]),                                 # input_size = multiplier * horizon
       "batch_size": tune.choice([7]),                                           # Number of series in windows
       "windows_batch_size": tune.choice([256]),                                 # Number of windows in batch
       "n_pool_kernel_size": tune.choice([[2, 2, 2], [16, 8, 1]]),               # MaxPool's Kernelsize
       "n_freq_downsample": tune.choice([[168, 24, 1], [24, 12, 1], [1, 1, 1]]), # Interpolation expressivity ratios
       "activation": tune.choice(['ReLU']),                                      # Type of non-linear activation
       "n_blocks":  tune.choice([[1, 1, 1]]),                                    # Blocks per each 3 stacks
       "mlp_units":  tune.choice([[[512, 512], [512, 512], [512, 512]]]),        # 2 512-Layers per block for each stack
       "interpolation_mode": tune.choice(['linear']),                            # Type of multi-step interpolation
       "random_seed": tune.randint(1, 10),
    }

  # Fit and predict
    fcst = NeuralForecast(
      models=[AutoNHITS(h=horizon, config=nhits_config,
                        num_samples=10)], # control of hyperopt samples
        freq='D')

    fcst_df = fcst.cross_validation(df=df2, val_size=val_size,
                                    test_size=test_size, n_windows=None)

    y_true = fcst_df.y.values
    y_hat = fcst_df['AutoNHITS'].values

    n_series = len(Y_df.unique_id.unique())

    y_true = y_true.reshape(n_series, -1, horizon)
    y_hat = y_hat.reshape(n_series, -1, horizon)

    rmsep=np.sqrt((np.sum(((np.array(y_true[0,:,0])-np.array(y_hat[0,:,0]))/np.array(y_true[0,:,0]))**2))/364)  #calculate error

    coin_results.append(rmsep)

return coin_results, y_hat[0,:,0], y_true[0,:,0]   #store error, predicted values and actual values

nhits_df = pd.DataFrame(columns = ['Coin','RMSEP'], index=[i for i in range(0,12)])     #results df
values=[0,0,0,0,0,0,0,0,0,0,0,0]
values_2=[0,0,0,0,0,0,0,0,0,0,0,0]

coins=['ada','bnb','btc','doge','eth','iota','link','ltc','trx','xlm','xmr','xrp']

for i in range(0,12):
    nhits_df.loc[i], values[i], values_2[i]=auto_nhits_bull(coins[i])                #run nhits on all coins

nhits_df.to_csv('/content/drive/MyDrive/Crypto_Thesis/NHITS_bull_all_results.csv')

df2 = pd.DataFrame(data=values, index=coins)
df2=df2.T
df2.to_csv('/content/drive/MyDrive/Crypto_Thesis/NHITS_bull_predictions.csv')

df3 = pd.DataFrame(data=values_2, index=coins)
df3=df3.T
df3.to_csv('/content/drive/MyDrive/Crypto_Thesis/NHITS_bull_actual.csv')
```