Università degli Studi di Cagliari

PhD DEGREE

Mathematics and Computer Science

Cycle XXXIV

TITLE OF THE PhD THESIS

Cryptocurrency Price Prediction Improvements Through Artificial Intelligence and Model Feature Refinement

Scientific and Disciplinary Sector(s)

INF/01 Computer Science

PhD Student: Nicola Uras

Supervisor Michele Marchesi, Giuseppe Destefanis

Final exam. Academic Year 2020/2021

Thesis defence: February 2022 Session

Dedicated to God, my family and mankind.

Contents

List of Figures

*

List of Tables

*

Chapter 1

Introduction

Time series forecasting has always been one of the most investigated scientific areas and this is probably due to the large amount of fields involving a time component including, meteorology, finance, econometric, astronomy, earthquake prediction, just to mention a few. Time series forecasting involves two sub-areas of study that are time series analysis and time series prediction. Time series analysis consist of using statistical tools to gain relevant insights from time series data. Time series prediction includes the use of prediction models to forecast time series future values from the knowledge of past values. The present work studies the field of financial time series forecasting, mainly focusing on the Cryptocurrency market. This field poses very sophisticated problems for several reasons. Due to its recent birth, this market has a dynamic nature continuously giving rise to new cryptocurrencies leading to frequent and sudden variations in their prices. Cryptocurrencies are virtual currencies based on an innovative technology known as Blockchain. As a consequence, their economy, along with traditional macroeconomic variables, depends on technology variables that can be directly measured from the Blockchain platform. All this features lead to high volatility in cryptocurrency prices.

At the same time, the analysis of a market whose price behaviour is still largely unexplored has a fundamental impact not only in the scientific field but also within economic and financial fields, serving as a source of information for speculators and investors.

The two main approaches used to study this research topic are traditional econometric methods and artificial intelligence algorithms. Econometric theory, firstly described by econometricians Ciompa and Tinbergen in the early 1900s, consist of applying statistical and mathematical models to economic time series data in order to outline actual economic phenomena. The main purpose of econometric is to model the stochastic process that is generating the data produced by the phenomena that you are observing. Econometric models

are simplifications of reality that help us approximate a very complex economic world and uncover basic relations between variables. Economic variables are most likely generated by an immensely complex model involving potentially infinitely many variables. A complex model is a model whose parameters estimator cannot be described with an analytic expression. As a consequence, a complex model depends not only on the model itself, but also, on the type of estimator being used. Basic econometrics models are linear regression models and autoregressive models, while basic estimators are those based on *Ordinary Least Squares* principles, discovered by Carl Friedrich Gauss in 1795.

Artificial Intelligence approach make use of statistical and mathematical algorithms to learn and emulate relationships between specific inputs and outputs on a sample data, known as training data. In time series field the inputs are time series past values while outputs are one or more time series future values. Two main subfields of Artificial Intelligence are machine end deep learning whose most popular algorithms are support vector machines and neural networks models.

The purpose of the present work is to study an innovative approach, based on the techniques described above, to gain relevant insights in order to enrich the state-of-the-art in the field of cryptocurrency time series forecasting. This study started with a regression forecasting problem of cryptocurrency time series prices where the main goal was to compare the results obtained through traditional econometrics models with those obtained using state-of-the-art artificial intelligence algorithms. Consequently, classification problems were conducted with the aim of predicting cryptocurrency price changes and verifying that the addition of technical analysis features to the dataset can lead to an effective improvement in the prediction of cryptocurrency price movements. Cryptocurrencies arouse keen interest not only in the scientific and financial fields but also within social media communities, making the analysis of their price beahviours one of the most discussed topics of the last fiew years. Several are also the studies that tried to use online information, including social media topics discussions, to predict cryptocurrencies price changes, proving the existence of possible cause-effect relationships between the cryptocurrency price changes and online information. These considerations led to the development of a study based on identifying and modeling relationships between cryptocurrencies market price changes and topic discussion occurrences on social media. This analysis is a further confirmation on how the addition of social variables to the dataset lead to an effective improvement on cryptocurrency price prediction. For this reason we decided to conduct a deeper investigation on whether developers emotions can effectively provide insights that can improve the prediction of cryptocurrency prices.

This approach proved how the progressive addition of variables of different

sources allows the development of more accurate cryptocurrencies prediction models.

1.1 Thesis Overview

This thesis is organized as follows: in Chapter 2 we present the main concepts that are extensively used in the rest of this thesis.

Chapter 3 presents a novel approach of analyzing cryptocurrencies price prediction, namely the progressive addition of features of different sources to the dataset that proved an effective improvement in the accuracy of the predictions. Starting from considering only trading variables as inputs models we gradually add to the dataset technical analysis variables and online information, such as variables generated from social media discussions and online developers comments.

Chapter 4 presents a literature review of cryptocurrency time series prediction.

Chapter 5 finally draws the conclusions of this thesis highliting results and contributions and presenting futures works.

Chapter 2

Background

2.1 Cryptocurrencies and Blockchain

Cryptocurrencies are virtual currencies based on Blockchain technology. The Blockchain concept is built on the distributed ledger technology that offers a consensus validation mechanism through a network of computers that facilitates peer-to-peer transactions without the need for an intermediary or a centralized authority to update and maintain the information generated by the transactions. The name Blockchain derives from the fact that when a transaction is validated is added as a new "block" to an already existing chain of transactions. Blockchain is one of the most promising technologies of the moment whose applications range from proprietary networks used to process financial transactions or insurance claims to platforms that can issue and trade equity shares and corporate bonds [81]. The most valueble cryptocurrency at the moment is the Bitcoin, a form of electronic cash invented by an unknown person or group of people using the pseudonym Satoshi Nakamoto, whose network of nodes was started in 2009. Although this ecosystem was introduced in 2009, its actual use began to grow only from 2013. Therefore, Bitcoin is a new entry in currency markets, though it is officially considered as a commodity rather than a currency and its price behaviour is still largely unexplored, presenting new opportunities for researchers and economists to highlight similarities and differences with standard financial currencies, also in view of its very different nature with respect to more traditional currencies or commodities. Ethereum and Litecoin are two other valuable cryptocurrencies at the moment. Ethereum was invented in 2013 by programmer Vitalik Buterin and its network went live on 2015. This cryptocurrency is second only to Bitcoin in market capitalization. Litecoin is a peer-to-peer cryptocurrency and open-source software project developed by Charlie Lee and its network went live on 2011. The price volatility of

cryptocurrencies, above all the Bitcoin one, is far greater than that of fiat currencies, providing significant potential in comparison to mature financial markets. Hence, forecasting cryptocurrencies price has also great implications not only in the scientific field but also within economic and financial fields, serving as a source of information for speculators and investors.

2.2 Time Series

2.2.1 Definition

A time series is a set of observations x_t, each one being recorder at a specific time t. A *discrete-time* time series is one in which the set T_0 of times at which observations are made is a discrete set, as is the case, for example, when observations are made at fixed time intervals. *Continuous-time* time series are obtained when observations are recorded continuously over some time interval, for example when $T_0 = [0,1]$.

2.2.2 Time Series Decomposition

Any time series is supposed to consist of three systematic components that can be described and modelled. These are 'base level', 'trend' and 'seasonality', plus one non-systematic component called 'noise'. The base level is defined as the average value in the series. A trend is observed when there is an increasing or decreasing slope in the time series. Seasonality is observed when there is a repeated pattern between regular intervals, due to seasonal factors. Noise represents the random variations in the series. Every time series is a combination of these four components, where base level and noise always occur, whereas trend and seasonality are optional. Depending on the nature of the trend and seasonality, a time series can be described as an additive or multiplicative model. This means that each observation in the series can be expressed as either a sum or a product of the components [57]. An additive model is described by following the linear equation:

$$x(t) = BaseLevel + Trend + Seasonality + Noise \tag{2.1}$$

A multiplicative model is instead represented by the following non-linear equation:

$$x(t) = BaseLevel * Trend * Seasonality * Noise \tag{2.2}$$

An additive model would be used when the variations around the trend does not vary with the level of the time series whereas a multiplicative model

would be appropriate if the trend is proportional to the level of the time series. This method of time series decomposition is called "classical decomposition".

2.2.3 iid Noise

The simplest model for a time series is one in which there is no trend or seasonal component and in which the observations are simply independent and identically distributed (iid) random variables with zero mean. In this model there is no dependence between observations. From a mathematical point of view, we can state that for all $h \geq 1$ and all $x, x_1, ..., x_n$ equation 2.3 is valid.

$$P\big[X_{n+h} \leq x | X_1 = x_1, ..., X_n = x_n\big] = P\big[X_{n+h} \leq x\big] \tag{2.3}$$

Equation 2.3 shows that the knowledge of $X_1, ..., X_n$ is of no value for predicting the behaviour of $Xn+h$. Iid sequences are carachterized by constant mean, variances and higher order moments.

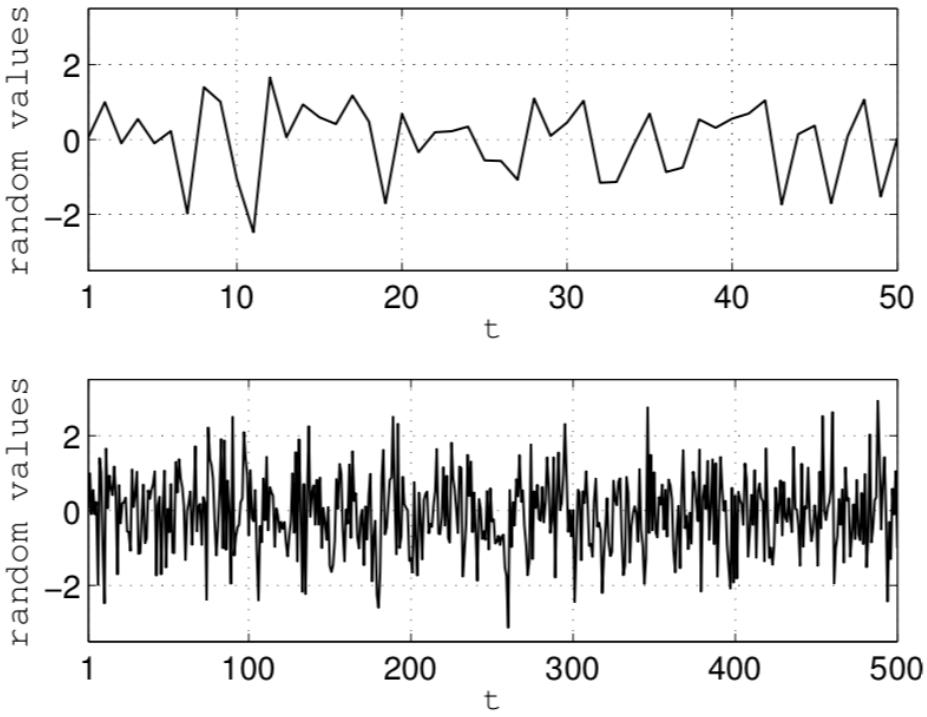


Figure 2.1: Gaussian IIDs - Examples of iid sequences

2.2.4 Stationarity

A time series $\{X_t\}$ is said to be stationary if its statistics does not change over time. $\{X_t\}$ is stationary if its statistical properties are similar to those of the *time-shifted* series $\{X_{t+h}\}$, for each integer h. More specifically, a time series can be *strict* stationary or *weakly* stationary.

A time series $\{X_t\}$ is said to be strictly stationary if equation 2.4 occurs for every t and h.

$$(X_1, ..., X_h) := (X_{t+1}, ..., X_{t+h}), \quad (t,h) \in N \times N \tag{2.4}$$

Strict stationarity is a more restrictive concept implying that the entire distribution does not change over time.

A time series $\{X_t\}$ is said to be weakly stationary or covariance stationary, if the mean, variance and covariances are invariant in time t.

In stationary data, the unconditional moments are time-invariant while the conditional moments can change. Let us consider the stochastic process X_t that generates our time series observations that is well approximated by an $AR(1)$ model. The unconditional mean of Xt is given by equation 2.5.

$$E(X_t) = \mu, \quad \text{with } \mu \text{ time-invariant and } X_t \sim N\left(\mu, \sigma^2\right) \tag{2.5}$$

The conditional mean of $X_t = \phi X_{t-1} + \epsilon_t$, with $\epsilon_t \sim N(0, \sigma^2)$, is given by equation 2.6.

$$E(X_t | X_{t-1}) = E(\phi X_{t-1} | X_{t-1}) + E(\epsilon_t | X_{t-1}) = \phi E(X_{t-1} | X_{t-1}) = \phi X_{t-1} \tag{2.6}$$

In 2.5 we assume $E(\epsilon_t | X_{t-1}) = 0$ for exogeneity.

2.2.5 White Noise

If $\{X_t\}$ is a sequence of uncorrelated random variables, each with zero mean and variance σ^2, then $\{X_t\}$ is stationary. Such a sequence is referred to as a *white noise* process with zero mean and variance σ^2. From a mathematical point of view, this process is represented by equation 2.7.

$$\{X_t\} \simeq WN(0, \sigma^2) \tag{2.7}$$

More specifically, a *white noise* process is an example of a weakly stationary process.

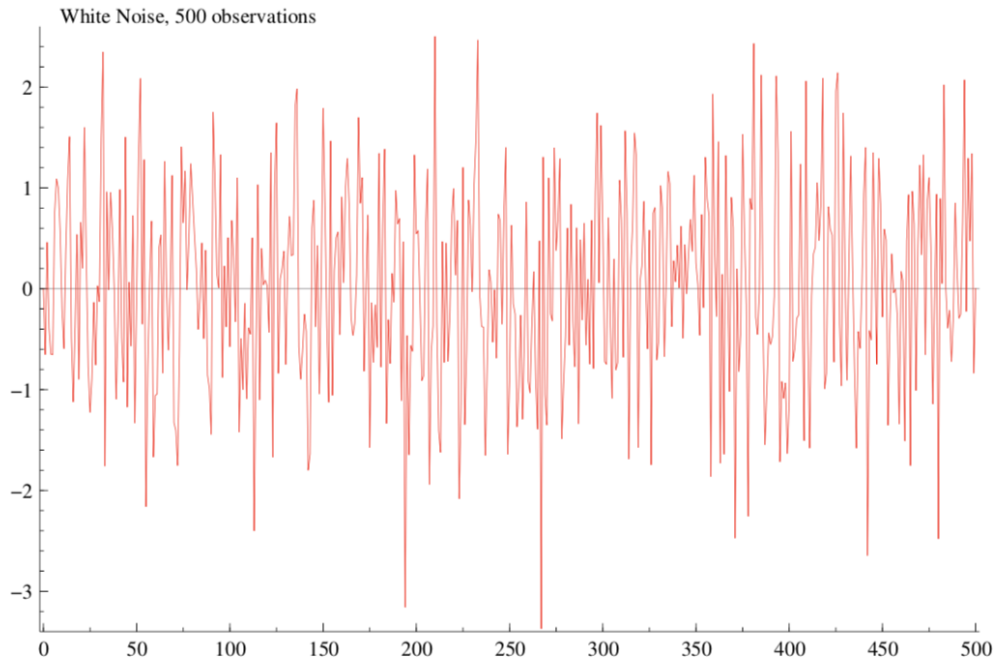


Figure 2.2: Realization of a white noise process with 500 observations

2.2.6 Random Walk

A stochastic process with a unit root is non-stationary, namely shows statistical properties that change over time, including mean, variance and covariance, and can cause problems in predictability of time series models. A common process with unit root is the random walk.

A time series $\{X_t\}$ is said to be a *random walk* if as defined as in equation 2.8, where $\{\epsilon_t\} \simeq WN(0,\sigma^2)$ is a white noise process.

$$X_t = \epsilon_1 + \epsilon_2 + ... + \epsilon_t, \quad \forall t \in N \tag{2.8}$$

The more common way to define the random walk is in recursive form as the sum of all white noise processes until t, as described in equation 2.9.

$$X_t = X_{t-h} + \epsilon_{t-h-1} + ... + \epsilon_{t-1} + \epsilon_t, \quad \forall h \in N \tag{2.9}$$

The random walk is a non-stationary process because it contains a stochastic trend formed by the accumulation of random shocks.

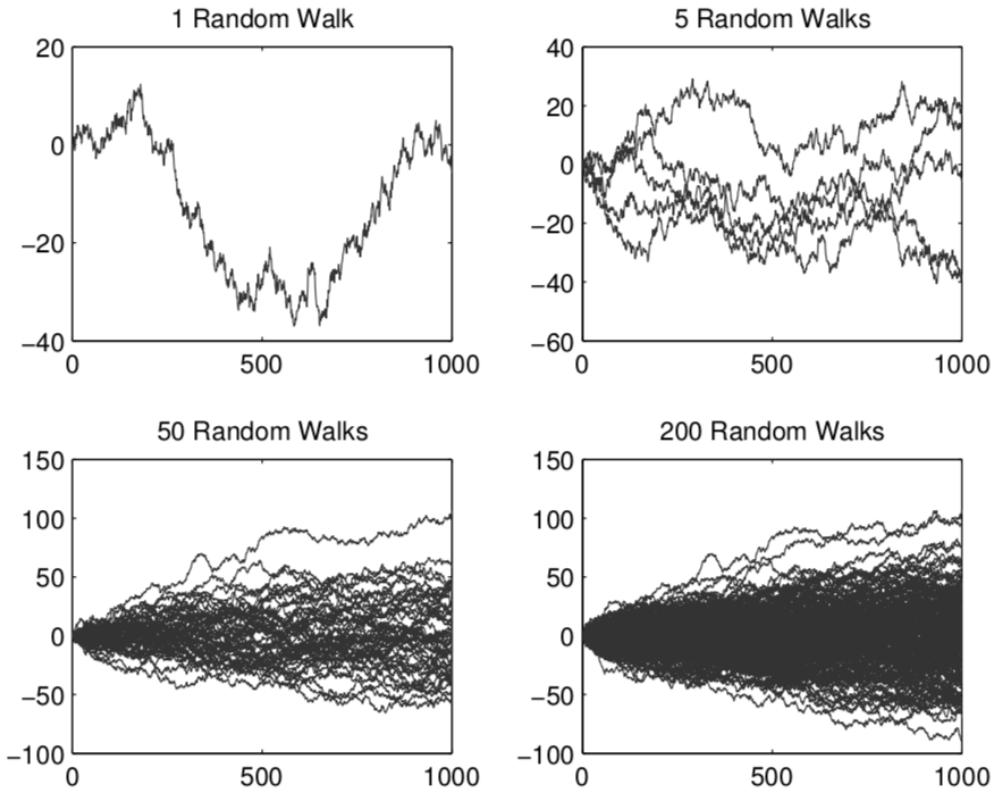


Figure 2.3: Realization of random walks

2.2.7 Linear Autoregressive Models

Autoregressive models are used to model temporal dependence and dynamic behaviour of stationary time series. Their popularity derives from the Wold's Theorem which states that most weakly stationary time-series $\{X_t\}$ can be well approximated by autoregressive processes.

A time series $\{X_t\}$ is said to be generated by an autoregressive model of order 1, denoted with *AR(1)*, if and only if, equation 2.10 yields.

$$X_t = \phi X_{t-1} + \epsilon_t, \quad \{\epsilon_t\} \simeq WN(0, \sigma^2) \tag{2.10}$$

Since the model is of order 1, its equation is characterized by a single regressor and recursive substitution lead to equation 2.11.

$$\begin{aligned} X_t &= \phi X_{t-1} + \epsilon_t = \phi^2 X_{t-2} + \phi\epsilon_{t-1} + \epsilon_t = \dots \\ &= \phi^n X t - n + \phi^{n-1}\epsilon_{t-(n-1)} + \dots + \phi\epsilon_{t-1} + \epsilon_t \end{aligned} \tag{2.11}$$

The autoregressive model of order 1 can also feature an "intercept" or "constant" parameter α that determines the unconditional mean of the time-series. An $AR(1)$ model with intercept is represented by equation 2.12.

$$X_t = \alpha + \phi_1 X_{t-1} + \epsilon_t, \quad \epsilon_t \sim WN(0, \sigma_\epsilon^2) \tag{2.12}$$

The unconditional mean of the time series is given by equation 2.13.

$$E(X_t) = E(\alpha + \phi X_{t-1} + \epsilon_t) = \alpha + \phi E(X_{t-1}) = \alpha + \phi\mu \tag{2.13}$$

Reminding that the unconditional mean of a stationary stochastic process is time-invariant, we have $E(X_t) = E(X_{t-1}) = \mu$. From this consideration we prove that the unconditional mean of the process is given by equation 2.14.

$$\mu = \frac{\alpha}{1-\phi} \tag{2.14}$$

For a random walk process $\phi = 1$ and $\mu \longrightarrow \infty$. The unconditional moments of a random walk process are not defined.

The autoregressive model of order 1 with equation 2.10, can be generalized to a model of order p.

A time series $\{X_t\}$ is said to be generated by an autoregressive model of order p, denoted $AR(p)$, if and only if is represented by equation 2.15.

$$X_t = \phi_1 X_{t-1} + \phi_2 X_{t-2} + \dots + \phi_p X_{t-p} + \epsilon_t, \quad \{\epsilon_t\} \sim WN(0, \sigma_\epsilon^2) \tag{2.15}$$

An autoregressive model of order p generates stationary time series if

$$|\phi_1 + \phi_2 + \dots + \phi_p| < 1.$$

2.3 Statistical Learning

Nowadays vast amounts of data are being generated in many fields, from academic research to agricultural, finance and industrial experiments. By analysing these data it is possible to extract important patterns and insights that can be used to improve solutions and business processes. This approach of learning from data is known as *statistical learning*. Typical use cases of learning problems are predict the price of a stock on the basis of company performance measures and economic data, identify and group identity documents from digitized images, identify the risk factors for prostate cancer, based on clinical and demographic variables and so on and so forth. The learning problems can be categorized as either supervised or unsupervised. In supervised learning, the goal is to predict the value of an outcome measure based on a number of input measures. In this scenario the outcome variable can be quantitative, for example a stock price or categorical, such as heart attack/no Heart attack. The sample data are grouped into a *training* set of data, in which the outcome and feature measurements for a set of objects are observed. Using this data a prediction model, or learner, can be built and used to predict the outcome for new unseen objects. A good model is one that accurately predicts such an outcome. This kind of problem is called "supervised" because of the presence of the outcome variable to guide the learning process. In the unsupervised learning problem there is no measurements of the outcome and only the input features are observed. The task in this problem is to describe how the data are organized or clustered.

The distinction in output type, namely between quantitative and categorical variables, has led to a naming convention for the prediction tasks. We refer to *regression* tasks when we predict quantitative outputs, and *classification* when we predict qualitative outputs. Inputs also vary in measurement type and can includes both qualitative and quantitative input variables. These have also led to distinctions in the types of methods that are used for prediction: some methods are defined most naturally for quantitative inputs, some most naturally for qualitative and some for both. Qualitative variables are typically represented numerically by codes. The easiest case is when there are only two classes or categories, such as "success" or "failure", "survived" or "died". These are often represented by a single binary digit or bit as 0 or 1, or else by 1 and 1. Such numeric codes are sometimes referred to as targets. When there are more than two categories, several alternatives are available. The most useful and commonly used coding is via dummy variables. Here a K-level qualitative variable is represented by a vector of K binary variables or bits, only one of which is “on” at a time. Although more compact coding schemes are possible, dummy variables are symmetric in the levels of the factor [51].

2.3.1 Linear Regression Models and Least Squares

Let us consider an input set of data represented by the vector $X^T = \left(X_1, X_2, ..., X_p\right)$, and want to predict a real-valued output Y. The linear regression model has the form of equation 2.16.

$$f(X) = \beta_0 + \sum_{j=1}^{p} X_j \beta_j. \tag{2.16}$$

The linear model either assumes that the regression function $E(Y\big|X)$ is linear, or that the linear model is a reasonable approximation. Here the β_j 's are unknown parameters or coefficients, and the variables X_j can come from different sources, such as quantitative inputs, transformations of quantitative inputs, such as log, square-root or square, basis expansions, such as $X_2 = X_1^2, X_3 = X_1^3$, leading to a polynomial representation, numeric or "dummy" coding of the levels of qualitative inputs, interactions between variables, for example, $X_3 = X_1 \cdot X_2$. No matter the source of the X_j , the model is linear in the parameters. Typically we have a set of training data $(x_1, y_1), ..., (x_N, y_N)$ from which to estimate the parameters β. Each $x_i = (x_{i1}, x_{i2}, ..., x_{ip})^T$ is a vector of feature measurements for the ith case. The most popular estimation method is least squares, in which we pick the coefficients $\beta = (\beta_0, \beta_1, ..., \beta_p)^T$ to minimize the residual sum of squares represented in equation 2.17.

$$RSS(\beta) = \sum_{i=1}^{N} \left(y_i - f(x_i)\right)^2 = \sum_{i=1}^{N} \left(y_i \beta_0 - \sum_{j=1}^{p} x_{ij} \beta_j\right)^2. \tag{2.17}$$

From a statistical point of view, this criterion is reasonable if the training observations $\left(x_i, y_i\right)$ represent independent random draws from their population. Even if the x_i's were not drawn randomly, the criterion is still valid if the y_i's are conditionally independent given the inputs x_i. Figure 2.4 illustrates the geometry of least-squares fitting in the R^{p+1}-dimensional space occupied by the pairs $\left(X, Y\right)$.

The purpose is to minimize equation 2.17. Let us denote $\mathbf{X}$ the $N \times \left(p+1\right)$ matrix with each row an input vector (with a 1 in the first position), and similarly let y be the N-vector of outputs in the training set. With these considerations the residual sum-of-squares can be written as equation 2.18.

$$RSS(\beta) = \left(\mathbf{y} - \mathbf{X}\beta\right)^T \left(\mathbf{y} - \mathbf{X}\beta\right). \tag{2.18}$$

This is a quadratic function in the $p+1$ parameters. Differentiating with respect

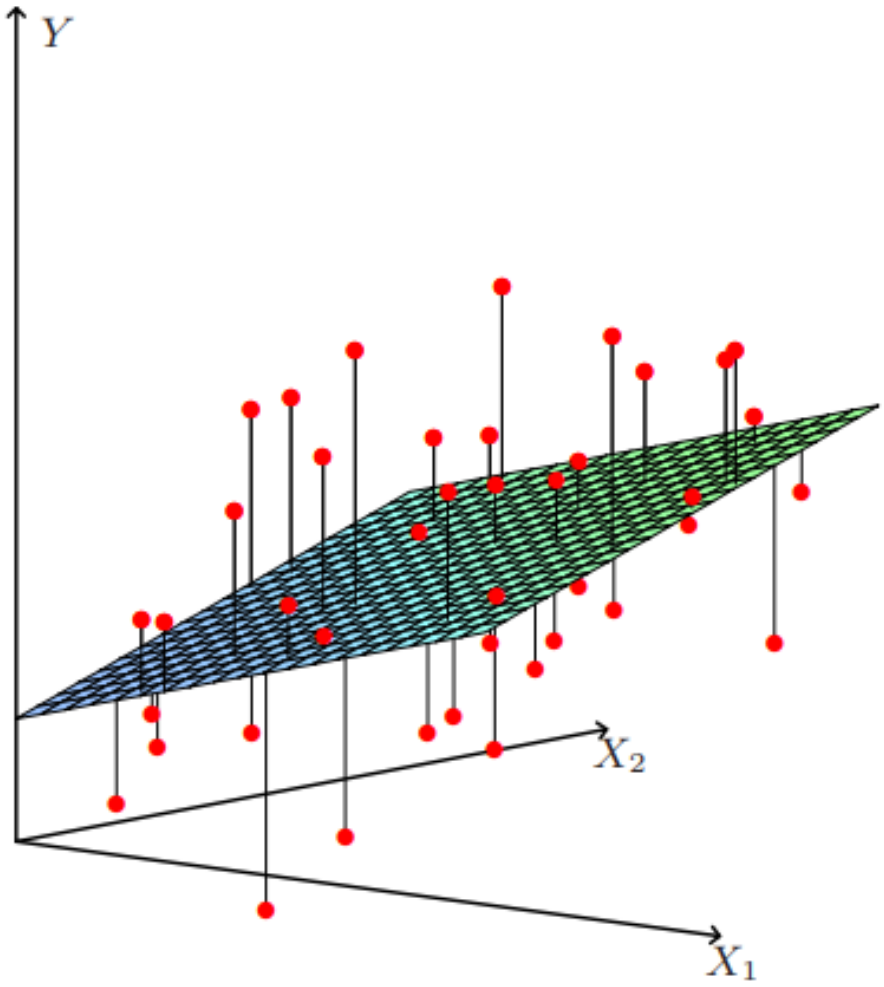


Figure 2.4: Linear least squares fitting with $X \in R^2$

to β we obtain equation 2.19.

$$\begin{aligned} \frac{\partial RSS}{\partial \beta} &= -2\mathbf{X}^T\Big(\mathbf{y}-\mathbf{X}\beta\Big) \\ \frac{\partial^2 RSS}{\partial \beta \partial \beta^T} &= 2\mathbf{X}^T\mathbf{X}. \end{aligned} \tag{2.19}$$

Assuming that X has full column rank, and hence $\mathbf{X}^T\mathbf{X}$ is positive definite, the first derivative can be set to zero, leading to the unique solution represented by equation 2.20.

$$\hat{\beta} = \Big(\mathbf{X}^T\mathbf{X}\Big)^{-1}\mathbf{X}^t\mathbf{y}. \tag{2.20}$$

The predicted values at an input vector x_0 are given by $\hat{f}\big(x_0\big) = \big(1 : x_0\big)^T\hat{\beta}$. The fitted values at the training inputs are expressed by equation 2.21 with $\hat{y}_i = \hat{f}\big(x_i\big)$.

$$\hat{\mathbf{y}} = \mathbf{X}\hat{\beta} = \mathbf{X}\Big(\mathbf{X}^T\mathbf{X}\Big)^{-1}\mathbf{X}^T\mathbf{y}. \tag{2.21}$$

2.3.2 Linear Methods for Classification: Logistic Regression

In this scenario the predictor takes values in a discrete set and it is always possible to divide the input space into a collection of regions labeled according to the classification. The boundaries of these regions can be rough or smooth, depending on the prediction function. For an important class of procedures, these decision boundaries are linear and they are known as *linear methods for classification.*

The logistic regression model arises from the desire to model the posterior probabilities of the K classes via linear functions in x, while at the same time ensuring that they sum to one and remain in $\big[0,1\big]$. The model has the form represented in equation 2.22.

$$\begin{aligned} log\frac{Pr\Big(G=1\Big|X=x\Big)}{Pr\Big(G=K\Big|X=x\Big)} &= \beta_{10} + \beta_1^T x \\ log\frac{Pr\Big(G=2\Big|X=x\Big)}{Pr\Big(G=K\Big|X=x\Big)} &= \beta_{20} + \beta_2^T x \\ &\vdots \\ log\frac{Pr\Big(G=K-1\Big|X=x\Big)}{Pr\Big(G=K\Big|X=x\Big)} &= \beta_{(K-1)0} + \beta_{K-1}^T x. \end{aligned} \tag{2.22}$$

The model is specified in terms of $K-1$ log-odds or logit transformations, reflecting the constraint that the probabilities sum to one. Although the model uses the last class as the denominator in the odds-ratios, the choice of denominator is arbitrary in that the estimates are equivariant under this choice. Equation 2.23 shows that the probabilities clearly sum to one.

$$\begin{aligned} Pr\Big(G=k\Big|X=x\Big) &= \frac{exp\big(\beta_{k0}+\beta_k^T x\big)}{1+\sum_{l=1}^{K-1} exp\big(\beta_{l0}+\beta_l^T x\big)}, \quad k=1,...,K-1, \\ Pr\Big(G=K\Big|X=x\Big) &= \frac{1}{1+\sum_{l=1}^{K-1} exp\big(\beta_{l0}+\beta_l^T x\big)} \end{aligned} \tag{2.23}$$

Logistic regression can be binomial, ordinal or multinomial. When $K=2$, the model is binomial and it is especially simple, since there is only a single linear function and the observed outcome for a dependent variable can have only two possible types, encoded with 0 and 1 for example. It is widely used in biostatistical applications where binary responses (two classes) occur quite frequently. For example, patients survive or die, have heart disease or not, or a condition is present or absent. Multinomial logistic regression deals with situations where the outcome can have three or more possible types that are not ordered while ordinal logistic regression deals with dependent variables that are ordered.

2.3.3 Non-linear Methods for Classification: Support Vector Machines

This section describes generalizations of linear decision boundaries for classification, covering extensions to the non-separable case, where the classes overlap. These techniques are then generalized to what is known as the support vector machine, which produces nonlinear boundaries by constructing a linear boundary in a large, transformed version of the feature space. Support Vector Machines are supervised learning algorithms used for machine learning applications to solve classification and regression problems. The original *SVM* algorithm was introduced by Vapnik and Chervonenkis [112] in 1963. In 1992 Boser, Guyon and Vapnik [15] developed a new *SVM* version for non-linear classification problems by applying kernel functions to maximum margin hyperplanes. *SVM's* are based on statistical learning frameworks and they are one of the most robust and commonly used prediction methods in machine learning applications. The original *SVM* implementation is a linear classifier involving a linear kernel for linearly separable data. In this problems the input space can always be divided into a collection of regions with linear decision boundaries. In a p-dimensional space a linear classifier is a $(p-1)$-dimensional hyperplane described by equation 3.10, where w is the normal vector to the hyperplane,

$(x_1, y_1), ..., (x_n, y_n)$ are n points of a given training data set and y_i is the class to which the point x_i belongs.

$$\mathbf{w}^T\mathbf{x} - b = \mathbf{0} \tag{2.24}$$

The hyperplane that best classifies the data is the one that represents the largest separation, better known as *margin*, between the classes. Therefore, this hyperplane is chosen so that the distance from it to the nearest data point on each side is maximized. If the data is linearly separable, then such a hyperplane exists and it is known as the maximum-margin hyperplane. In 1992 Boser, Guyon and Vapnik introduce an extension of the original *SVM* algorithm useful when dealing with non-linearly separable data where the classes overlap. In this cases the *SVM* produces non-linear boundaries by constructing a linear boundary in a large, transformed version of the feature space. More specifically, this extended version creates non-linear classifiers by simply replacing every dot product in the 3.10 with a non-linear kernel function. These kernel functions allow the original space to be mapped to a higher dimensional feature space, thus making the separation between classes more obvious in this transformed space. The most commonly used kernels are listed below.

- Polynomial: $k(\vec{x}, \vec{y}) = (\vec{x} \cdot \vec{y})^d$
- Gaussian radial basis function (*rbf*): $k(\vec{x}, \vec{y}) = e^{-\frac{||\vec{x}-\vec{y}||^2}{2\sigma^2}}$
- Hyperbolic tangent: $k(\vec{x}, \vec{y}) = tanh(k\vec{x} \cdot \vec{y} + c)$, where $k > 0$ and $c < 0$

2.3.4 Neural Networks

The concept of artificial neural network was introduced in 1958 by Frank Rosenblatt proposing the first single feed forward neural network, also known as perceptron. The term neural network has evolved to encompass a large class of models and learning methods. A neural network is a two-stage regression or classification model, typically represented by a network diagram as in 2.5. This network applies both to regression or classification. For regression, typically $K = 1$ and there is only one output unit Y_1 at the top. However, these networks can handle multiple quantitative responses in a seamless fashion. For K-class classification, there are K units at the top, with the kth unit modeling the probability of class k. There are K target measurements Y_k, $k = 1, ..., K$, each being coded as a $0-1$ variable for the kth class. Derived features Z_m are created from linear combinations of the inputs, and then the target Y_k is modeled as a function of linear combinations of the Z_m as descriebed in equation 2.25, where

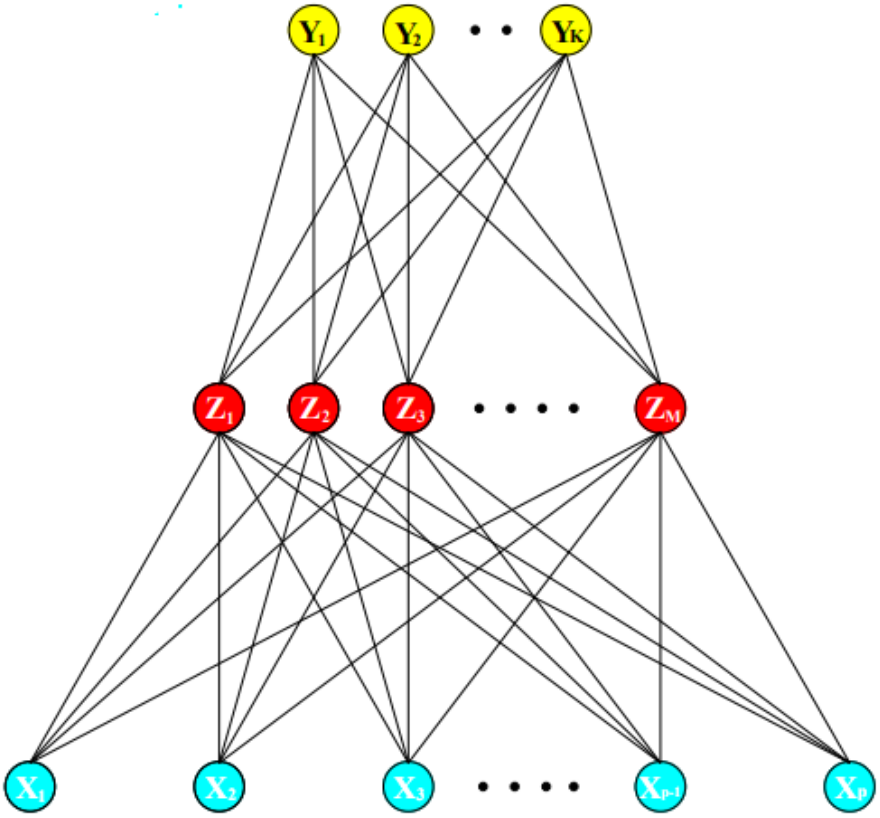


Figure 2.5: Schematic of a single hidden layer, feed-forward neural network

$Z = (Z_1, Z_2, ..., Z_m)$ and $T = (T_1, T_2, ..., T_k)$.

$$\begin{aligned} Z_m &= \sigma(\alpha_{0m} + \alpha_m^T X), \quad m = 1, ..., M \\ T_k &= \beta_{0k} + \beta_k^T Z, \quad k = 1, ..., K \\ f_k(X) &= g_k(T), \quad k = 1, ..., K. \end{aligned} \tag{2.25}$$

The activation function (v) is usually chosen to be the sigmoid $(v) = 1/(1+e^{-v})$. Sometimes Gaussian radial basis functions are used for the (v), producing what is known as a radial basis function network. Neural networks are usually structured with an additional bias unit feeding into every unit in the hidden and output layers. Thinking of the constant 1 as an additional input feature, this bias unit captures the intercepts α_{0m} and β_{0k} in model. The output function $g_k(T)$ allows a final transformation of the vector of outputs T. For regression tasks the identity function $g_k(T) = T_k$ is usually used. Early work in K-class classification also used the identity function, but this was later abandoned in favor of the softmax function described in equation 2.26.

$$g_k(T) = \frac{e^{T_k}}{\sum_{l=1}^{K} e^{t_l}}. \tag{2.26}$$

The units in the middle of the network, computing the derived features Z_m, are called hidden units because the values Z_m are not directly observed. In general there can be more than one hidden layer. We can think of the Z_m as a basis expansion of the original inputs X; the neural network is then a standard linear model, or linear multilogit model, using these transformations as inputs.

The neural network model has unknown parameters, often called weights, and the main goal is to seek values for them that make the model fit the training data well. Let us denote the complete set of weights by θ showed in equation 2.27.

$$\begin{aligned} \{\alpha_{0m}, \alpha_m; m = 1, 2, ..., M\} \quad & M(p+1) weights, \\ \{\beta_{0k}, \beta_k; k = 1, 2, ..., K\} \quad & K(M+1) weights. \end{aligned} \tag{2.27}$$

For regression purposes, let us use the sum-of-squared errors as error function 2.28.

$$R(\theta) = \sum_{k=1}^{K} \sum_{i=1}^{N} \left(y_{ik} - f_k(x_i)\right)^2. \tag{2.28}$$

For classification purposes we can use either squared error or cross-entropy as our measure of fit 2.29 with the corresponding classifier $G(x) = \text{argmax}_k f_k(x)$.

$$R(\theta) = -\sum_{i=1}^{N} \sum_{k=1}^{K} y_{ik} log\, f_k(x_i). \tag{2.29}$$

With the softmax activation function and the cross-entropy error function, the neural network model is exactly a linear logistic regression model in the hidden units, and all the parameters are estimated by maximum likelihood. The generic approach to minimizing $R(\theta)$ is by gradient descent, called back-propagation in this setting. Because of the compositional form of the model, the gradient can be easily derived using the chain rule for differentiation. This can be computed by a forward and backward sweep over the network, keeping track only of quantities local to each unit. Let $z_{mi} = \sigma(\alpha_{0m} + \alpha_m^T x_i)$, from 2.25 and let $z_i = (z_{1i}, ..., z_{Mi})$. Then the error function $R(\theta)$ is described by equation 2.30 with derivatives 2.31.

$$R(\theta) = \sum_{i=1}^{N} R_i = \sum_{i=1}^{N} \sum_{k=1}^{K} \big(y_{ik} - f_k(x_i)\big)^2. \tag{2.30}$$

$$\begin{aligned} \frac{\partial R_i}{\partial \beta_{km}} &= -2\big(y_{ik} - f_k(x_i)\big) g_k'\big(\beta_k^T z_i\big) z_{mi}, \\ \frac{\partial R_i}{\partial \alpha_{ml}} &= -\sum_{k=1}^{K} 2\big(y_{ik} - f_k(x_i)\big) g_k'\big(\beta_k^T z_i\big) \beta_{km} \sigma'\big(\alpha_m^T x_i\big) x_{il}. \end{aligned} \tag{2.31}$$

Given these derivatives, a gradient descent update at the $(r+1)$st iteration has the form of equation 2.32 where γ_r is a constant usually known as *learning rate*.

$$\begin{aligned} \beta_{km}^{(r+1)} &= \beta_{km}^{(r)} - \gamma_r \sum_{i=1}^{N} \frac{\partial R_i}{\partial \beta_{km}^{(r)}}, \\ \alpha_{ml}^{(r+1)} &= \alpha_{ml}^{(r)} - \gamma_r \sum_{i=1}^{N} \frac{\partial R_i}{\partial \alpha_{ml}^{(r)}}. \end{aligned} \tag{2.32}$$

The derivatives of the error function 2.31 can be written as in equation 2.33.

$$\begin{aligned} \frac{\partial R_i}{\partial \beta_{km}} &= \delta_{ki} z_{mi}, \\ \frac{\partial R_i}{\partial \alpha_{ml}} &= s_{mi} x_{il}. \end{aligned} \tag{2.33}$$

The quantities δ_{ki} and s_{mi} are "errors" from the current model at the output and hidden layer units, respectively. From their definitions, these errors satisfy equation 2.34 known as the *back-propagation equations*.

$$s_{mi} = \sigma'\big(\alpha_m^T x_i\big) \sum_{k=1}^{K} \beta_{km} \delta_{ki}. \tag{2.34}$$

Using this, the updates in 2.32 can be implemented with a two-pass algorithm. In the forward pass, the current weights are fixed and the predicted values $\hat{f}_k(x_i)$ are computed from formula 2.25. In the backward pass, the errors δ_{ki} are computed, and then back-propagated via 2.34 to give the errors s_{mi}. Both sets of errors are then used to compute the gradients for the updates in 2.32, via 2.33. This two-pass procedure is what is known as back-propagation.

Chapter 3

Cryptocurrency Predictions

3.1 Forecasting Bitcoin closing price series using linear regression and neural networks models

In this thesis work we forecast daily closing price series of Bitcoin, Litecoin and Ethereum cryptocurrencies, using data on prices and volumes of prior days. Cryptocurrencies price behaviour is still largely unexplored, presenting new opportunities for researchers and economists to highlight similarities and differences with standard financial prices. We compared our results with various benchmarks: one recent work on Bitcoin prices forecasting that follow different approaches, a well-known paper that uses Intel, National Bank shares and Microsoft daily NASDAQ closing prices spanning a 3-year interval and another, more recent paper which gives quantitative results on stock market index predictions. We followed different approaches in parallel, implementing both statistical techniques and machine learning algorithms: the Simple Linear Regression (*SLR*) model for uni-variate series forecast using only closing prices, and the Multiple Linear Regression (*MLR*) model for multivariate series using both price and volume data. We used two artificial neural networks as well: Multilayer Perceptron (*MLP*) and Long short-term memory (*LSTM*). While the entire time series resulted to be indistinguishable from a random walk, the partitioning of datasets into shorter sequences, representing different price "regimes", allows to obtain precise forecast as evaluated in terms of Mean Absolute Percentage Error(*MAPE*) and relative Root Mean Square Error (*relativeRMSE*). In this case the best results are obtained using more than one previous price, thus confirming the existence of time regimes different from random walks. Our models perform well also in terms of time complexity, and provide overall results better than those obtained in the benchmark studies, improving the state-of-the-art.

Bitcoin is the world's most valuable cryptocurrency, a form of electronic cash, invented by an unknown person or group of people using the pseudonym Satoshi Nakamoto [87], whose network of nodes was started in 2009. Although the system was introduced in 2009, its actual use began to grow only from 2013. Therefore, Bitcoin is a new entry in currency markets, though it is officially considered as a commodity rather than a currency, and its price behaviour is still largely unexplored, presenting new opportunities for researchers and economists to highlight similarities and differences with standard financial currencies, also in view of its very different nature with respect to more traditional currencies or commodities. The price volatility of Bitcoin is far greater than that of fiat currencies [19], providing significant potential in comparison to mature financial markets [77] [29] [30]. According to *[31]* website, one of the most popular sites that provides almost real-time data on the listing of the various cryptocurrencies in global exchanges, on May 2019 Bitcoin market capitalization value is valued at approximately 105 billion of USD. Hence, forecasting Bitcoin price has also great implications both for investors and traders. Even if the number of bitcoin price forecasting studies is increasing, it still remains limited [74]. In this work, we approach the forecast of daily closing price series of the Bitcoin cryptocurrency using data on prices and volumes of prior days. We compare our results with three well-known recent papers, one dealing with Bitcoin prices forecasting using other approaches, one forecasting Intel, National Bank shares and Microsoft daily NASDAQ prices and one on stock market index forecasting using fusion of machine learning techniques.

The first paper we compare to, tries to predict three of the most challenging stock market time series data from NASDAQ historical quotes, namely Intel, National Bank shares and Microsoft daily closed (last) stock price, using a model based on chaotic mapping, firefly algorithm, and Support Vector Regression (SVR) [62]. In the second one [74] used different machine learning techniques such as Artificial Neural Networks (ANN) and Support Vector Machines (SVM) to predict, among other things, closing prices of Bitcoin. The third paper we consider in our work proposes a two stage fusion approach to forecast stock market index. The first stage involves SVR. The second stage uses ANN, Random Forest (RF) and SVR [95]. We decided to predict these three share prices to give a sense of how Bitcoin is different from traditional markets. Moreover, to enrich our work, we applied the models also to two other two well-know cryptocurrencies: Ethereum and Litecoin. In this work we forecast daily closing price series of Bitcoin cryptocurrency using data of prior days following different approaches in parallel, implementing both statistical techniques and machine learning algorithms. We tested the chosen algorithms on two datasets: the first consisting only of the closing prices of the previous days; the second adding the volume data. Since Bitcoin exchanges are open 24/7, the closing

price reported on *Coinmarketcap* we used, refers to the price at 11:59 PM UTC of any given day. The implemented algorithms are Simple Linear Regression (SLR) model for univariate series forecast, using only closing prices; a Multiple Linear Regression (MLR) model for multivariate series, using both price and volume data; a Multilayer Perceptron and a Long Short-Term Memory neural networks tested using both the datasets. The first step consisted in a statistical analysis of the overall series. From this analysis we show that the entire series are not distinguishable from a random walk. If the series were truly random walks, it would not be possible to make any forecasts. Since we are interested in prices and not in price variations, we avoided the time series differencing technique by introducing and using the novel presented approach. Therefore, each time series was segmented in shorter overlapping sequences in order to find shorter time regimes that do not resemble a random walk so that they can be easily modeled. Afterwards, we run all the algorithms again on the partitioned dataset.

The reminder of this work is organized as follows. Section 2 presents the methodology, briefly describing the data, their pre-processing, and finally the models used. Section 3 presents and discuss the results. Section 4 draws the conclusions.

3.1.1 Methods

In this section we first introduce some notions on time series analysis, which helped us to take the operational decisions about the algorithms we used and to better understand the results presented in the following. Then, we present the dataset we used, including its pre-processing analysis. Finally we introduce our proposed algorithms with the metrics employed to evaluate their performance and the statistical tools we adopted.

3.1.2 Time Series Analysis

Time Series Components

Any time series is supposed to consist of three systematic components that can be described and modelled. These are 'base level', 'trend' and 'seasonality', plus one non-systematic component called 'noise'. The base level is defined as the average value in the series. A trend is observed when there is an increasing or decreasing slope in the time series. Seasonality is observed when there is a repeated pattern between regular intervals, due to seasonal factors. Noise represents the random variations in the series. Every time series is a combination of these four components, where base level and noise always occur, whereas trend

and seasonality are optional. Depending on the nature of the trend and seasonality, a time series can be described as an additive or multiplicative model. This means that each observation in the series can be expressed as either a sum or a product of the components [57]. An additive model is described by following the linear equation:

$$y(t) = BaseLevel + Trend + Seasonality + Noise \tag{3.1}$$

A multiplicative model is instead represented by the following non-linear equation:

$$y(t) = BaseLevel * Trend * Seasonality * Noise \tag{3.2}$$

An additive model would be used when the variations around the trend does not vary with the level of the time series whereas a multiplicative model would be appropriate if the trend is proportional to the level of the time series. This method of time series decomposition is called "classical decomposition" [57].

Statistical Measures

The statistical measures we calculated for each time series are the mean, labelled with μ, the standard deviation σ and the trimmed mean $\bar{\mu}$, obtained discarding a portion of data from both tails of the distribution. The trimmed mean is less sensitive to outliers than the mean, but it still gives a reasonable estimate of central tendency and can be very helpful for time series with high volatility.

3.1.3 Collected data

We tested our algorithms on six daily price series. Three of them are stock market series, all the data were extracted from the 'Historical Data' available on *[116]* website; the other ones are cryptocurrencies, namely Bitcoin, Ethereum and Litecoin price daily series, all the data were extracted from *[31]* website.

- Daily stock market prices for Microsoft Corporation (MSFT), from 9/12/2007 to 11/11/2011.
- Daily stock market prices for Intel Corporation (INTC), from 9/12/2007 to 11/11/2010.
- Daily stock market prices for National Bankshares Inc. (NKSH), from 6/27/2008 to 8/29/2011.

- Daily Bitcoin, Ethereum and Litecoin price series, from 15/11/2015 to 12/03/2020.

We state once more that we choose these price series and the related time intervals as benchmark to compare our results with well known literature results obtained by using other methods.

Specifically, we have chosen for the stock market series the same time intervals chosen in [62]. The choice of Bitcoin as criptocurrency is quite natural since it represents about 60% of the Total Market Capitalization. We chose Ethereum and Litecoin since they are among the most important and well-known cryptocurrencies. It is worth noting that, for the stock market series we used the same data of the work we compare to, whereas for the cryptocurrencies we used all the available data to have more significant results.

The dataset was divided into two sets, a training part and a testing part. After some empirical test the partition of the data which lead us to optimal solutions was 80% of the daily data for the training dataset and the remaining for the testing dataset.

3.1.4 Data pre-processing

For both models we prepared our dataset in order to have a set of inputs (X) and outputs (Y) with temporal dependence. We performed a one-step ahead forecast: our output Y is the value from the next (future) point of time while the inputs X are one or several values from the past, i.e. the so called *lagged* values. From now on we identify the number of used lagged values with the *lag* parameter. In the Linear Regression and *Univariate* LSTM models the dataset includes only the daily closing price series, hence there is only one single lag parameter for the *close* feature. On the contrary, in the Multiple Linear Regression and *Multivariate* LSTM models the dataset includes both *close* and *volume (USD)* series, hence we use two different lag parameters, one for the *close* and one for the *volume* feature. In both cases, we attempted to optimize the predictive performance of the models by varying the *lag* from 1 to 10.

3.1.5 Univariate versus Multivariate Forecasting

A univariate forecast consists of predicting time series made by observations belonging to a single feature recorded over time, in our case the closing price of the series considered. A multivariate forecast is a forecast in which the dataset consists of the observations of several features. In our case we used:

- for BTC, ETH and LTC series all the features provided by *Coinmarketcap* website: Open, High, Low, Close, Volume.

- for MSFT, INTC, NKSH series all the features provided by *Yahoofinance* website: Date, Open, High, Low, Close, Volume.

We observed that adding features to the dataset did not lead to better predictions, but performance and sometimes also results worsened. For this reason, we decided to use in the multivariate analysis only the *close* and *volume* features, that provided the best results.

3.1.6 Statistical Analysis

As a first step we carried out a statistical analysis in order to check for non-stationarity in the time series. We used the *augmented Dickey-Fuller test* and *autocorrelation plots* [9] [17]. A stochastic process with a *unit root* is non-stationary, namely shows statistical properties that change over time, including mean, variance and covariance, and can cause problems in predictability of time series models. A common process with *unit root* is the *random walk*. Often price time series show some characteristics which makes them indistinguishable from a random walk. The presence of such a process can be tested using a *unit root* test.

The *ADF* test is a statistical test that can be used to test for a *unit root* in a univariate process, such as time series samples. The null hypothesis H_0 of the *ADF* test is that there is a *unit root*, with the alternative H_a that there is no *unit root*. The most significant results provided by this test are the *observed test statistic*, the Mackinnon's approximate *p-value* and the *critical values* at the 1%, 5% and 10% levels.

The test statistic is simply the value provided by the *ADF* test for a given time series. Once this value is computed it can be compared to the relevant critical value for the Dickey-Fuller Test.

Critical values, usually referred to as α levels, are an error rate defined in the hypothesis test. They give the probability to reject the null hypothesis H_0. So if the observed test statistic is less than the critical value (keep in mind that ADF statistic values are always negative [9]), then the null hypothesis H_0 is rejected and no *unit root* is present.

The *p-value* is instead the probability to get a "more extreme" test statistic than the one observed, based on the assumed statistical hypothesis H_0, and its mathematical definition is shown in equation 3.3.

$$p_{value} = P\Big(t \geq t_{observed} \Big| \; H_0\Big) \tag{3.3}$$

The *p-value* is sometimes called *significance*, actually meaning the closeness of the *p-value* to zero: the lower the *p-value*, the higher the significance.

In our analysis we performed this test using the *adfuller()* function provided by the *statsmodels* Python library, and we chose a *significance level* of 5%.

Furthermore, the *autocorrelation plot*, also known as *correlogram*, allowed us to calculate the correlation between each observation and the observations at previous time steps, called *lag values*. In our case we employed the *autocorrelation_plot()* function provided by the python *Pandas* library [78].

3.1.7 Forecasting

We decided to follow two different approaches: the first uses two well-known statistical methods: Linear Regression (LR) and Multiple Linear Regression (MLR). The second uses two very common neural networks (NN): Multilayer Perceptron (MLP) NN and Long Short-Term Memory (LSTM) NN. The reasons of this choices are explained below.

Linear Regression and Multiple Linear Regression

Linear regression is a linear approach for modelling the relationship between a dependent variable and one independent variable, represented by the main equation:

$$y = b_0 + \vec{b}_1 \cdot \vec{x}_1, \tag{3.4}$$

where y and $\vec{x}_1$ are the dependent and the independent variable respectively, while b_0 is the intercept and $\vec{b}_1$ is the vector of slope coefficients. In our case the components of the vector $\vec{x}_1$, our independent variable, are the values of the closing prices of the previous days. Therefore, $\vec{x}_1$ size is the value of the *lag* parameter. In our case y represents the closing price to be predicted.

This algorithm aims to find the curve that best fits the data, which best describes the relation between the dependent and independent variable. The algorithm finds the best fitting line plotting all the possible trend lines through our data and for each of them calculates and stores the amount $(y - \bar{y})^2$, and then choose the one that minimizes the squared differences sum $\sum_i (y_i - \bar{y}_i)^2$, namely the line that minimizes the distance between the real points and those crossed by the line of best fit.

We then tried to forecast with multiple independent variables, adding to the *close* price feature the observations of several features, including *volume*, *highest value* and *lowest value* of the previous day. These information were gained from *Coinmarketcap* website. In these cases we used a Multiple Linear

Regression model (MLR). The MLR equation is:

$$y = b_0 + \vec{b}_1 \cdot \vec{x}_1 + ... + \vec{b}_n \cdot \vec{x}_n = b_0 + \sum_{i=1}^{n} \vec{b}_i \cdot \vec{x}_i \tag{3.5}$$

where the index i refers to a particular independent variable and n is the dimension of the independent variables space.

We used the Linear and Multiple regression model of *scikit learn* [97]. We decided to use this two models for several reasons: they are simple to write, use and understand, they are fast to compute, they are commonly used models and fit well to datasets with few features, like ours. Their disadvantage is that they can model only linear relationships.

Multilayer Perceptron

A multilayer perceptron (MLP) is a feedforward artificial neural network that generates a set of outputs from a set of inputs. It consists of at least three layers of neurons: an input layer, a hidden layer and an output layer. Each neuron, apart from the input ones, has a nonlinear activation function. MLP uses backpropagation for training the network. In our model we keep the structure as simple as possible, with a single hidden layer. Our inputs are the closing prices of the previous days, where the number of values considered depends on the *lag* parameter. The output is the forecast price. The optimal number of neurons were found by optimizing the network architecture on the number of neurons itself, varying it in an interval between 5 and 100. We used the Python *Keras* library [26].

LSTM Networks

Long Short-Term Memory networks are nothing more than a prominent variations of Recurrent Neural Network (RNN). RNN's are a class of artificial neural network with a specific architecture oriented at recognizing patterns in sequences of data of various kinds: texts, genomes, handwriting, the spoken word, or numerical time series data emanating from sensors, markets or other sources [54]. Simple recurrent neural networks are proven to perform well only for short-term memory and are unable to capture long-term dependencies in a sequence. On the contrary, LSTM networks are a special kind of RNN, able at learning long-term dependencies. The model is organized in cells which include several operations. LSTM hold an internal state variable, which is passed from one cell to another and modified by Operation Gates (forget gate, input gate, output gate). These gates control how much of the internal state is passed to the output and work in a similar way to other gates. These three gates have

independent weights and biases, hence the network will learn how much of the past output and of the current input to retain and how much of the internal state to send out to the output.

In our case the inputs are the closing prices of the previous days and the number of values considered depends on the *lag* parameter. The output is the forecast price. We used the Keras framework for deep learning. Our model consists of one stacked LSTM layer with 64 units each and the densely connected output layer with one neuron. We used Adam optimizer and MSE (mean squared error) as a loss.

We optimized our *LSTM* model searching for the best set of *epochs* and *batch size* "hyperparameters" values. These hyperparameters strongly depend on the number of observations available for the experiment. Due to the recently birth of the cryptocurrency markets, the dimensions of our datasets are quite limited (around 1000 observations), therefore we decided to vary the *epochs* hyperparameter from 300 to 800 with a step of 100. The Keras LSTM algorithm we used sets as default value for *batch size* 32. So, for each fixed *epoch*, we trained the model varying the *batch size* within the interval $[22,82]$ with a step of 10. We did not take into account values less than 300 epochs, nor greater than 800 in order to avoid *underfitting* and *overfitting* problems. Furthermore, we did not consider *batch size* values less than 22, since they would lead to extremely long training times. Similarly, *batch size* values greater than 82 would not allow to find a good local minimum point of the chosen loss function during the learning procedure. The results obtained during the hyperparameters tuning are shown in figure 3.1.

This figure shows the *MAPE* error as a function of the *batch size* hyperparameter, for each fixed epoch. As can be seen from the figure, we considered the *batch size* equal to 72 to be the optimal value. In fact, it is an excellent compromise, having a low MAPE value, which is also practically the same for all tested *epochs*. The optimal choice for the *epochs* hyperparameter is 600, which is the one that minimizes the MAPE error for *batch size* equal to 72, and is consistently among the best choices for almost all batch sizes considered. Therefore, the best set of *epochs* and *batch size* "hyperparameters" values we chose is 600 and 72, respectively.

3.1.8 Time Regimes

The time series considered are found to be indistinguishable from a random walk. This peculiarity is common for time series of financial markets, and in our case is confirmed by the predictions of the models, in which the best result is obtained considering only the price of the previous day.

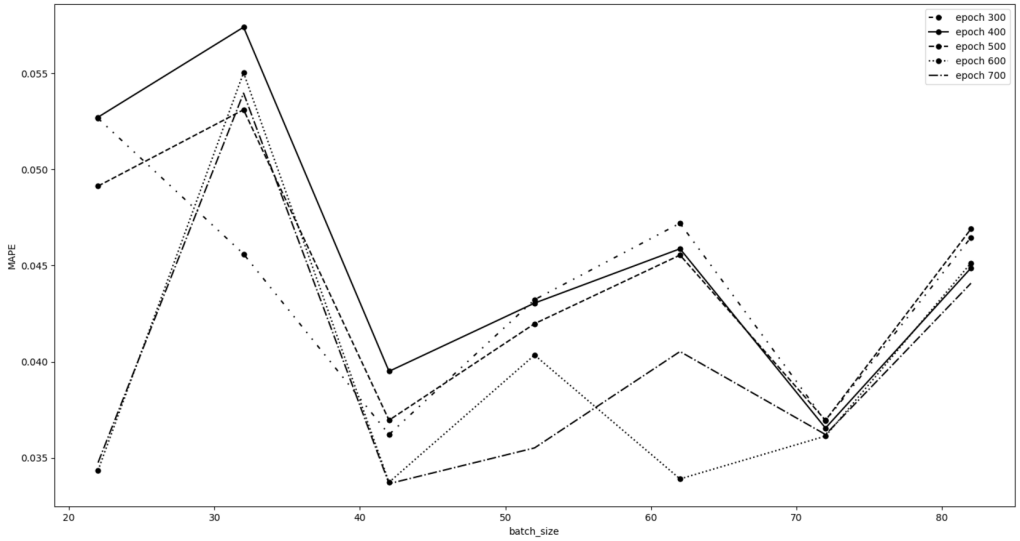


Figure 3.1: Bitcoin hyperparameters tuning results

The purpose is to find an approach that allow us to avoid time series differencing technique, in view of the fact that we are interested in prices and not in price variations represented by integrated series of d-order. For this reason, each time series was segmented into short partially overlapping sequences, in order to find if shorter time regimes are present, where the series do not resemble a random walk. Finally, to continue with the forecasting procedure, a train and a test set were identified within each time regime.

For each regime we always sampled 200 observations - namely 200 daily prices. The beginning of the next regime is obtained with a shift of 120 points from the previous one. Thus, every regime is 200 points wide and has 80 points in common with the following one.

We chose a regime length of 200 days because, in this way, we obtain at least 5 regimes (from 5 to 12) for each time series to test the effectiveness of our algorithms, without excessively reducing the number of samples needed for training and testing. The choice was determined also according to the following: we performed the augmented Dickey-Fuller test on subsets of the data, starting from the whole set and progressively reducing the data window and sliding it through the data. The first subset of data that does not behave as random walks appears at time interval of 230 days, which we rounded to 200.

Since the time series considered have different lengths, the partition in regimes has generated:

- Bitcoin, Ethereum and Litecoin: 12 regimes

- Microsoft: 8 regimes
- Intel and National Bankshares: 5 regimes

From a mathematical point of view, the used approach can be described as follows.

Let us target a vector $\overrightarrow{OA}$ along the t axis, with length 200. This vector is identified by the points $O(1,0)$, $A(a,0) \equiv (200,0)$. The length of this vector represents the width of each time regime.

Let $\overrightarrow{OH}$ be a fixed translation vector along the t axis, identified by the points $O(1,0)$ and $H(h,0) \equiv (120,0)$. The length of $\overrightarrow{OH}$ represents the translation size.

For the sake of simplicity, let us label the $\overrightarrow{OA}$ and $\overrightarrow{OH}$ vectors with $\vec{A}$ and $\vec{H}$.

Let $\vec{A}'$ be the vector $\vec{A}$ shifted by $\vec{H}$ and $\vec{A}^n$ the vector $\vec{A}$ shifted by n times $\vec{H}$.

Therefore, the vector that identifies the nth sequence to be sampled along the series is given by:

$$\vec{A}^n = \vec{A} + n\vec{H} \tag{3.6}$$

where $n \in \left[0, \frac{D-A}{h}\right]$, being D the dimension of the sampling space, A the time regimes width and h the translation size.

So the nth time regime is given by:

$$R^n = f\left(\vec{A}^n\right) = f\left(\vec{A} + n\vec{H}\right) \tag{3.7}$$

where f is the function that maps the values along the t axis (dates) to the respective regimes y values (actual prices).

3.1.9 Performance Measures

To evaluate the effectiveness of different approaches, we used the *relative* Root Mean Square Error (rRMSE) and the Mean Absolute Percentage Error (MAPE), defined respectively as:

$$relativeRMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(\frac{y_i - f_i}{y_i}\right)^2} \tag{3.8}$$

$$MAPE = \frac{1}{N}\sum_{i=1}^{N}\left|\frac{y_i - f_i}{y_i}\right| \tag{3.9}$$

In both formulas y_i and f_i represent the actual and forecast values, and N is the number of forecasting periods. These are scale free performance measures, so that they are well appropriate to compare model performance results across series with different orders of magnitude, as in our study.

3.1.10 Results

3.1.11 Time Series Analysis

In figure 3.2 we report the decomposition of Bitcoin (a-d) and Microsoft (e-h) time series, for comparison purposes, as obtained using the *seasonal_decompose()* method, provided by the Python *statsmodels* library [105].

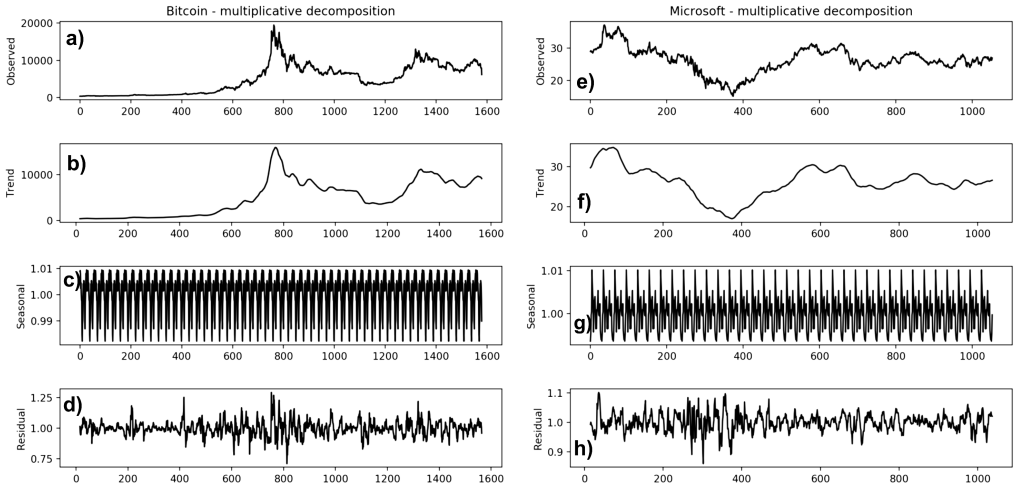


Figure 3.2: Decomposition of Bitcoin (a-d) and Microsoft (e-h) time series

The *seasonal_decompose()* method requires to specify whether the model is additive or multiplicative. In the Bitcoin time series, the trend of increase at the beginning is almost absent (from around 2016-04 to 2017-02); in later years, the frequency and the amplitude of the cycle appears to change over time. The Microsoft time series shows a non-linear seasonality over the whole period, with frequency and amplitude of the cycles changing over time. These considerations suggest that the model is multiplicative. Furthermore, if we look at the residuals, they look quite random, in agreement with their definitions. The Bitcoin residuals are likewise meaningful, showing periods of high variability in the later years of the series.

It is also possible to group the data at seasonal intervals, observing how the values are distributed and how they evolve over time. In our work we grouped the data of the same month over the years we considered. This is achieved with the 'Box plot' of month-wide distribution, shown in figure 3.3 (a: Bitcoin; b: Microsoft). The Box plot is a standardized way of displaying the distribution of data based on five numbers summary: minimum, first quartile, median, third quartile and maximum. The box of the plot is a rectangle which encloses the middle half of the sample, with an end at each quartile. The length of the box

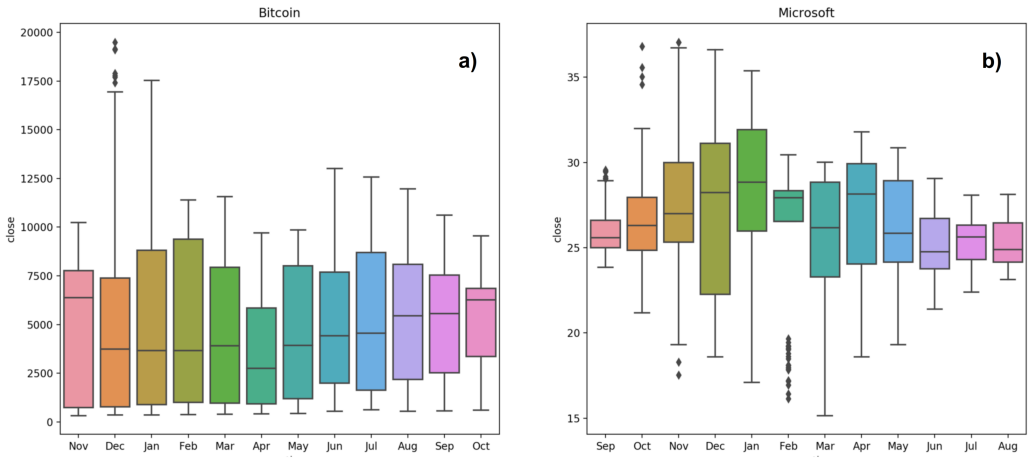


Figure 3.3: Seasonality of Bitcoin (a) and Microsoft (b) time series

is thus the inter-quartile range of the sample. The other dimension of the box has no meaning. A line is drawn across the box at the sample median. Whiskers sprout from the two ends of the box defining the outliers range. The box length gives an indication of the sample variability, and for the Bitcoin samples shows a large variance, in almost all months, except for April, September and October. Not surprisingly, bitcoin volatility is much higher than Microsoft one. The line crossing the box shows where the sample is centred, i.e. the median. The position of the box in its whiskers and the position of the line in the box also tell us whether the sample is symmetric or skewed, either to the right or to the left. The plot shows that the Bitcoin monthly samples are therefore skewed to the right. The top whiskers is much longer than the bottom whiskers and the median is gravitating towards the bottom of the box. This is due to the very high prices that Bitcoin reached throughout the period between 2017 and 2018. These large values tend to skew the sample statistics. In Microsoft, an alternation between samples skewed to the left and samples skewed to the right occurs, except for the sample of October that shows a symmetric distribution. Lack of symmetry entails one tail being longer than the other, distinguishing between heavy-tailed or light-tailed populations. In the Bitcoin case we can state that the majority of the samples are left skewed populations with short tails. Microsoft shows an alternation between heavy-tailed and light-tailed distributions. We can see that some Microsoft samples, particularly those with long tails, present outliers, representing anomalous values. This is due to the fact that heavy tailed distributions tend to have many outliers with very high values. The heavier the tail, the larger the probability that you will get one or more disproportionate values in a sample.

Table 3.1: Time Series Statistical Measures

Series	μ	σ	$\bar{\mu}$
BTC	4931,3	3970,0	4593,1
ETH	216,8	239,8	171,2
LTC	55,9	58,0	45,6
MSFT	26,2	3,9	26,3
INTC	19,9	3,6	19,9
NKSH	24,3	3,9	24,5

Tables 3.1 and 3.2 show the statistics calculated for each time series and for each short time regime. The unit of measurement of the values in the tables is the US dollar ($). In table 3.1 we can observe that the only series for which the trimmed mean, obtained with *trim_mean()* method provided by the Python *scipy* library [60], with a cut-off percentage of 10%, is significantly different from the mean are Bitcoin, Ethereum and Litecoin. In particular the trimmed mean decreased. This is due to the fact that these cryptocurrencies, for a long period of time, registered a large price increment and this implies a shift of the mean to the right (i.e. to highest prices). This confirms that cryptocurrencies distribution is right-skewed. Table 3.2 shows that stock market series time regimes present a lower σ than BTC, ETH and LTC ones, namely that cryptocurrencies distribution has higher variance.

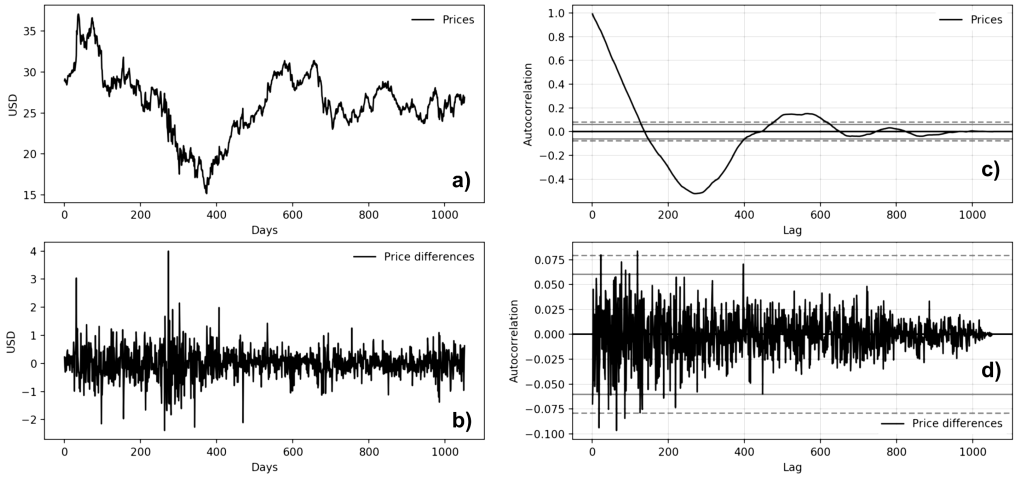


Figure 3.4: Microsoft time series autocorrelation plots

Figures 3.4 and 3.5 show the autocorrelation plots of BTC and MSFT series.

Table 3.2: Regimes Statistical Measures

Series	**h**	μ	σ	$\bar{\mu}$
BTC	0	419,7	39,6	421,6
	120	551,2	97,3	549,6
	240	707,9	122,5	693,2
	360	1110,1	358,8	1048,8
	480	2481,2	1107,4	2414,0
	600	7446,4	4808,8	6870,7
	720	10359,6	3082,8	9966,1
	840	7536,5	1130,1	7424,8
	960	5810,9	1382,3	5859,4
	1080	4509,6	1101,3	4349,9
	1200	8016,9	2752,9	8048,3
	1320	9154,5	1477,4	9080,2
ETH	0	6,0	4,6	5,8
	120	11,7	2,0	11,6
	240	10,8	1,7	10,8
	360	34,6	39,0	26,3
	480	195,8	114,6	194,5
	600	441,9	281,8	385,5
	720	695,9	251,4	682,0
	840	487,4	159,1	486,4
	960	239,6	118,0	228,2
	1080	144,6	34,0	141,8
	1200	204,7	52,5	201,1
	1320	186,8	42,5	181,7
LTC	0	3,5	0,4	3,4
	120	3,9	0,5	3,9
	240	3,9	0,2	3,9
	360	8,2	8,1	6,2
	480	33,8	19,3	33,3
	600	102,6	85,4	86,1
	720	167,0	65,0	163,7
	840	107,6	40,2	105,3
	960	52,9	17,9	52,2
	1080	50,5	19,9	48,7
	1200	87,4	23,8	85,7
	1320	67,2	22,3	64,5
MSFT	0	30,7	2,8	30,5
	120	26,1	3,2	26,4
	240	20,6	3,9	20,4
	360	22,8	3,8	22,8
	480	28,2	2,3	28,4
	600	26,8	2,2	26,7
	720	26,1	1,3	26,1
	840	26,0	1,2	26,0
INTC	0	23,5	2,4	23,5
	120	20,0	3,6	20,3
	240	15,4	2,3	15,1
	360	17,3	2,3	17,4
	480	20,6	1,4	20,4
NKSH	0	18,5	0,9	18,5
	120	22,2	3,0	22,2
	240	26,5	1,4	26,5
	360	25,9	1,9	26,0
	480	26,5	2,5	26,3

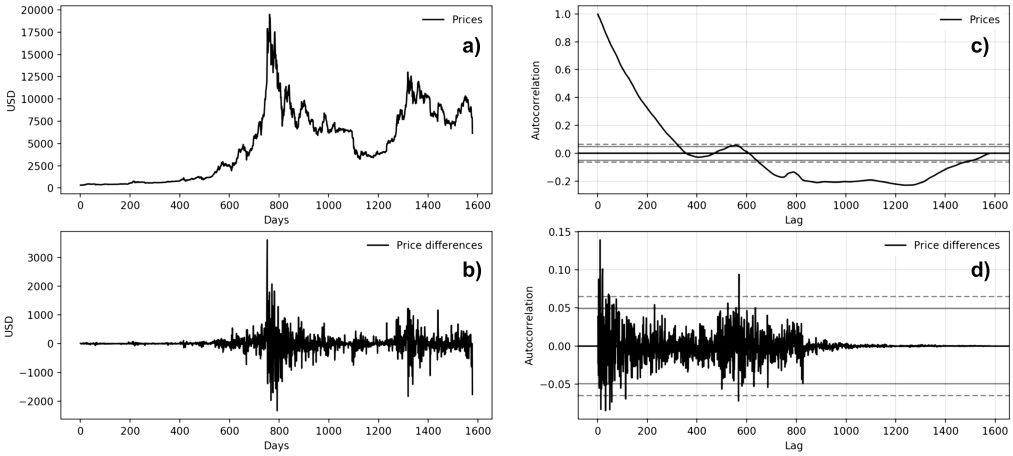


Figure 3.5: Bitcoin time series autocorrelation plots

Table 3.3: Augmented Dickey-Fuller test results

Series	**ADF statistic**	**p-value**
BTC	-2,12	0,24
ETH	-2,17	0,22
LTC	-2.34	0,16
MSFT	-1,98	0,29
INTC	-1,98	0,29
NKSH	-2,10	0,25

The others stock market series are not presented because they show the same features of the MSFT series. Both autocorrelation plots (sub-figures c) show a strong autocorrelation between the current price and the closest previous observations and a linear fall-off from there to the first few hundred lag values. We then tried to make the series stationary by taking the *first difference*. The autocorrelation plots of the 'differences series' (sub-figures d) show no significant relationship between the lagged observations. All correlations are small, close to zero and below the 95% and 99% confidence levels.

As regards the *augmented Dickey-Fuller* results, shown in table 3.3, looking at the observed *test statistics*, we can state that all the series follows a unit root process. We remind that the null hypothesis H_0 of the *ADF* test is that there is a *unit root*. In particular, all the observed *test statistics* are greater than those associated to all significance levels. This implies that we can not reject the null hypothesis H_0, but does not imply that the null hypothesis is true.

Observing the *p-values*, we notice that for the stock market series we have a low probability to get a "more extreme" test statistic than the one observed under the null hypothesis H_0. Precisely, for both *MSFT* and *INTC* we got a probability of 29%, for *NKSH* a probability of 25%. The same considerations also apply to the Bitcoin, Ethereum and Litecoin cryptocurrency time series. We conclude that H_0 can not be rejected and so each time series present a *unit root* process.

We conclude that all the considered series show the statistical characteristics typical of a *random walk*.

3.1.12 Time Series Forecasting

Table 3.4 and 3.5 show the best results, in terms of MAPE and rRMSE, obtained with the different algorithms applied to the entire series. From now on, let us label the closing and the volume features *lag* parameters with k_p and k_v respectively. In particular, table 3.4 reports the results obtained using the *Linear Regression* algorithm for univariate series forecast, using only closing prices, and the *Multiple Linear Regression* model for multivariate series, using both price and volume data.

Table 3.5 shows the results obtained with the *LSTM* neural network, distinguishing between *univariate LSTM*, using only closing prices, and *multivariate LSTM*, using both price and volume data.

Small values of the $MAPE$ and $rRMSE$ evaluation metrics suggest accurate predictions and good performance of the considered model.

From the analysis of the series in their totality, it appears that linear models outperforms neural networks. However, for both models, the majority of best results are obtained for a *lag* of 1,thus confirming our hypothesis that the series are indistinguishable from a random walk.

In order to perform the time series forecasting, we also implemented a *Multi-Layer Perceptron* model. Since the *LSTM* network outperforms the *MLP* one, we decided to show only the *LSTM* results. This is probably due to the particular architecture of the LSTM network, that is able to capture long-term dependencies in a sequence.

It should be noted that better predictions are obtained for stock market series rather than for the cryptocurrencies one. In particular, the best result is obtained for Microsoft series, with a MAPE of $0,011$ and k_p equal to 1. This is probably due to the high price fluctuations that Bitcoin and the other cryptocurrencies have suffered during the investigated time interval. This is confirmed by the statistics shown in table 3.1. It must be noted that the addition of the *volume* feature to the dataset does not improve the predictions.

Table 3.4: Linear and Multiple Linear Regression results

	Linear Regression			**Multiple Linear Regression**			
Series	**MAPE**	**rRMSE**	k_p	**MAPE**	**rRMSE**	k_p	k_v
BTC	0,026	0,040	1	0,026	0,037	1	1
ETH	0,031	0,049	1	0,039	0,053	6	3
LTC	0,034	0,050	1	0,045	0,058	2	2
MSFT	0,011	0,015	1	0,011	0,015	1	1
INTC	0,013	0,017	1	0,013	0,017	1	1
NKSH	0,014	0,019	12	0,013	0,018	7	5

Table 3.5: Univariate and Multivariate LSTM results

	Univariate LSTM			**Multivariate LSTM**			
Series	**MAPE**	**rRMSE**	k_p	**MAPE**	**rRMSE**	k_p	k_v
BTC	0,027	0,041	1	0,038	0,048	2	1
ETH	0,034	0,052	6	0,057	0,076	2	1
LTC	0,035	0,051	1	0,039	0,054	1	1
MSFT	0,012	0,015	1	0,012	0,015	1	2
INTC	0,013	0,017	2	0,013	0,017	1	1
NKSH	0,014	0,020	7	0,013	0,018	1	2

In order to perform prices forecast we changed the approach and decided to split the time series analysis using shorter time windows of 200 points, shifting the windows by 120 points, with the aim of finding local time regimes where the series do not follow the global random walk pattern.

Table 3.6 and 3.7 show the results obtained with our approach of partitioning the series into shorter sequences. Let us label the moving step forward with h. Particularly, in table 3.6 are presented the results obtained using the *Linear Regression* algorithm for univariate series forecast, using only closing prices, and the *Multiple Linear Regression* model for multivariate series, using both price and volume data. This approach, has the advantage of being simple to implement and requires low computational complexity. Nevertheless, has led to good results, similar to those present in the literature, if not better as in the Microsoft, Bitcoin and National Bankshares cases, where the MAPE error is lower that 1%.

Table 3.7 shows the results obtained with the *LSTM* neural network, distin-

guishing between *univariate LSTM*, using only closing prices, and *multivariate LSTM*, using both price and volume data. For each time regimes we show the best results obtained on a specific time window defined by the k_p and k_v values reported in Tabs. 3.6 and 3.7. Note that we highlighted the best results in bold. In particular, it is worth noting that introducing the time regimes, the best result is obtained for the Bitcoin time series, outperforming also the financial ones.

These results show how such innovative partitioning approach allowed us to avoid the "random walk problem", finding that best results are obtained using more than one previous price. Furthermore, this method leads to a significant improvement in predictions. It is worth noting that, from this analysis the best result arise from the Bitcoin series, with a *MAPE* error of $0,007$, a temporal window k_p of 7 and a translation step h of 120, obtained using both regression models and LSTM network.

Another interesting consideration that arises from the results is that, as stated previously in the analysis of the series in their entirety, the linear regression models generally outperform the neural networks ones, while in the short-time regimes approach the different models yielded to similar results.

For a direct feedback we report in table 3.8 the best results obtained in the papers we compared to and our best ones. In the event that the best MAPE error results from different models, we consider the model whose computational complexity is the least as best. It is noticeable that our results outperform those obtained in the benchmark papers, providing notable contribution to the literature.

3.2 Blockchain Cryptocurrencies' Price Movements Classification Using Deep Learning

This part of the Ph.D. work shows the results obtained from a comparison between a restricted and a unrestricted Bitcoin price classification, verifying whether the addition of technical indicators to the classic macroeconomic variables leads to an effective improvement in the prediction of Bitcoin price changes. The goal was achieved implementing different machine learning algorithms, such as *Support Vector Machine (SVM)*, *XGBoost (XGB)*, a *Convolutional Neural Network (CNN)* and a *Long Short Term Memory (LSTM)* neural network. Macroeconomic variables data were gained from *Yahoo Finance* website spanning a 4-year interval with a hourly resolution, while technical indicators data are provided by the python *talib* library. The variance problem on test samples has been taken into account through the cross validation technique which also al-

Table 3.6: LR and MLR results with time regimes

		Linear Regression			**Multiple Linear Regression**			
Series	**h**	**MAPE**	**rRMSE**	k_p	**MAPE**	**rRMSE**	k_p	k_v
BTC	0	0,015	0,025	4	0,012	0,014	8	10
	120	**0,007**	**0,010**	**7**	**0,007**	**0,011**	**1**	**1**
	240	0,029	0,050	4	0,031	0,052	5	1
	360	0,034	0,041	1	0,037	0,045	1	2
	480	0,041	0,062	2	0,039	0,061	2	1
	600	0,065	0,082	2	0,065	0,080	2	2
	720	0,028	0,035	1	0,026	0,035	1	5
	840	0,017	0,024	7	0,018	0,024	7	1
	960	0,030	0,040	4	0,029	0,040	1	10
	1080	0,029	0,039	1	0,022	0,031	3	3
	1200	0,018	0,025	8	0,021	0,026	8	2
	1320	0,020	0,026	5	0,021	0,027	7	7
ETH	0	0,045	0,060	7	0,042	0,056	10	6
	120	**0,022**	**0,029**	**1**	0,022	0,028	1	1
	240	0,031	0,047	4	0,033	0,046	1	3
	360	0,053	0,078	1	0,053	0,078	2	2
	480	0,048	0,077	1	0,050	0,077	1	1
	600	0,060	0,080	1	0,053	0,069	3	8
	720	0,039	0,051	1	0,036	0,049	1	7
	840	0,048	0,070	7	0,064	0,084	5	1
	960	0,051	0,068	1	0,055	0,071	4	1
	1080	0,032	0,046	3	**0,020**	**0,027**	**10**	**7**
	1200	0,024	0,031	8	0,022	0,029	1	8
	1320	0,025	0,033	1	0,028	0,035	1	1
LTC	0	0,027	0,034	4	0,023	0,027	8	8
	120	**0,011**	**0,018**	**3**	**0,011**	**0,017**	**1**	**4**
	240	0,030	0,046	5	0,031	0,047	5	2
	360	0,075	0,098	1	0,074	0,094	3	3
	480	0,073	0,111	1	0,074	0,112	1	1
	600	0,077	0,096	2	0,058	0,074	8	7
	720	0,040	0,049	1	0,040	0,047	1	1
	840	0,032	0,045	9	0,031	0,043	9	3
	960	0,047	0,060	3	0,048	0,062	1	1
	1080	0,037	0,047	9	0,023	0,028	7	7
	1200	0,026	0,032	8	0,027	0,034	8	1
	1320	0,026	0,036	1	0,026	0,037	1	1
MSFT	0	0,015	0,018	1	0,015	0,017	1	3
	120	0,037	0,045	6	0,035	0,044	6	4
	240	0,015	0,019	7	0,015	0,019	9	6
	360	0,010	0,014	3	0,012	0,018	1	1
	480	0,011	0,015	2	0,010	0,012	3	7
	600	0,009	0,011	4	0,009	0,011	5	1
	720	**0,008**	**0,011**	**7**	**0,007**	**0,009**	**10**	**8**
	840	0,012	0,015	1	0,012	0,015	1	10
INTC	0	0,014	0,019	5	0,013	0,017	6	10
	120	0,036	0,045	7	0,035	0,043	7	4
	240	0,017	0,022	5	0,017	0,022	2	3
	360	**0,012**	**0,015**	**1**	**0,012**	**0,015**	**1**	**1**
	480	0,016	0,020	1	0,016	0,020	3	5
NKSH	0	0,019	0,023	8	0,019	0,023	9	6
	120	0,014	0,018	9	0,013	0,017	10	4
	240	0,014	0,018	4	0,012	0,016	1	4
	360	0,019	0,026	2	0,019	0,026	2	1
	480	**0,009**	**0,012**	**7**	**0,009**	**0,012**	**10**	**5**

Table 3.7: Univariate and Multivariate LSTM results with time regimes

		Univariate LSTM			**Multivariate LSTM**			
Series	**h**	**MAPE**	**rRMSE**	k_p	**MAPE**	**rRMSE**	k_p	k_v
BTC	0	0,022	0,034	3	0,021	0,030	3	1
	120	**0,007**	**0,011**	**4**	**0,007**	**0,010**	**2**	**1**
	240	0,044	0,058	3	0,065	0,077	3	1
	360	0,088	0,105	2	0,187	0,233	3	3
	480	0,043	0,066	4	0,041	0,061	1	1
	600	0,068	0,088	1	0,078	0,127	2	1
	720	0,027	0,035	2	0,027	0,043	1	2
	840	0,017	0,023	1	0,017	0,031	3	1
	960	0,027	0,035	6	0,033	0,067	2	1
	1080	0,025	0,038	3	0,030	0,106	3	1
	1200	0,021	0,028	1	0,024	0,033	1	1
	1320	0,018	0,025	1	0,020	0,028	1	2
ETH	0	0,051	0,065	6	0,054	0,068	3	1
	120	0,022	0,028	1	0,023	0,031	1	3
	240	0,034	0,049	1	0,035	0,048	1	2
	360	0,217	0,248	5	0,284	0,349	3	3
	480	0,049	0,077	2	0,050	0,076	1	1
	600	0,074	0,109	3	0,164	0,396	1	1
	720	0,039	0,052	3	0,037	0,079	3	1
	840	0,067	0,092	1	0,052	0,252	1	1
	960	0,053	0,067	1	0,062	0,101	1	1
	1080	0,031	0,042	3	0,039	0,082	1	1
	1200	0,026	0,035	1	0,025	0,049	1	3
	1320	**0,021**	**0,031**	**2**	**0,022**	**0,031**	**1**	**1**
LTC	0	0,045	0,054	5	0,063	0,079	3	1
	120	**0,010**	**0,016**	**2**	**0,011**	**0,018**	**3**	**1**
	240	0,035	0,052	6	0,051	0,069	1	1
	360	0,395	0,409	6	0,397	0,443	3	2
	480	0,086	0,117	3	0,090	0,120	3	1
	600	0,136	0,164	1	0,167	0,431	1	3
	720	0,040	0,051	3	0,040	0,075	1	2
	840	0,034	0,045	1	0,035	0,062	1	2
	960	0,047	0,059	1	0,053	0,107	2	1
	1080	0,047	0,055	1	0,034	0,121	1	3
	1200	0,026	0,035	1	0,026	0,048	1	3
	1320	0,028	0,038	2	0,028	0,038	1	1
MSFT	0	0,014	0,017	1	0,014	0,017	1	2
	120	0,121	0,139	1	0,054	0,064	3	1
	240	0,017	0,023	2	0,017	0,023	1	3
	360	0,017	0,021	4	0,031	0,044	3	1
	480	0,012	0,015	1	0,012	0,016	1	2
	600	0,009	0,012	3	**0,009**	**0,012**	**3**	**1**
	720	**0,008**	**0,011**	**4**	0,010	0,014	2	1
	840	0,012	0,016	4	0,012	0,016	3	1
INTC	0	0,015	0,019	1	0,014	0,018	1	1
	120	0,056	0,068	1	0,069	0,091	3	3
	240	0,017	0,021	3	0,017	0,022	3	1
	360	**0,012**	**0,015**	**1**	**0,013**	**0,017**	**1**	**1**
	480	0,017	0,021	1	0,020	0,025	1	1
NKSH	0	0,021	0,027	1	0,023	0,027	3	1
	120	0,015	0,018	6	0,014	0,019	1	3
	240	0,016	0,022	1	0,017	0,022	1	3
	360	0,020	0,027	1	0,023	0,030	1	3
	480	**0,010**	**0,014**	**1**	**0,010**	**0,013**	**1**	**1**

Table 3.8: Best Benchmarks Results compared to ours

Reference	**Series**	**Model**	**MAPE**
[74]	BTC	SVM:0.9-1(Relief)	0,011
[95]	S&P BSE SENSEX	SVR	0,009
[62]	MSFT	SVR-CFA	0,052
	INTC	SVR-CFA	0,045
	NKSH	SVR-CFA	0,046
Our Work	BTC	LR	0,007
	ETH	MLR	0,020
	LTC	Univariate LSTM	0,010
	MSFT	MLR	0,007
	INTC	LR	0,012
	NKSH	LR	0,009

lowed to evaluate a more reliable estimate of the model's performance. Furthermore, the *Grid Search* technique was used to find the best *hyperparameters* values for each implemented algorithm. The results were evaluated in terms of the well known classification metrics, i.e. *accuracy*, *precision*, *recall* and *f1 score*. Based on the results, it was possible to demonstrate that the unrestricted case outperforms the restricted one, verifying that the addition of the technical indicators to the macroeconomic variables actually improves the accuracy on Bitcoin price classification.

During the last decade we have witnesses an exponentially growing over Cryptocurrencies traded and exchanged with a every day market cap of hundreds of billions of USD Dollars globally (1 trillion to January 2021). The technology behind these crypto-markets are based on distributed ledgers and blockchains, and cryptocurrencies rely upon these technologies to be mined, namely created, and transferred among users and in general stakeholders. These technologies guaranties anonymity and safe transfer of value, which in the end generated the crypto-markets mentioned above.

In January 2021 the number of live projects connected to blockchain technologies and distributed ledgers[1] is more than 7000, and the trend is constantly growing.

Bitcoin project is the first and the pioneering one, whose associated tokens

[1] `https://www2.deloitte.com/content/dam/insights/us/articles/4600\_Blockchain-five-vectors/DI\_Blockchain-five-vectors.pdf`

dominate the crypto-markets scene by both market volumes and capitalisation.

Due to the high volatility of crypto-markets, investors and stakeholders are attracted by this technology, its outlooks, the potential to a constant growth in values and the excess returns on their investments are exploited by investors since the dawn of the Bitcoin in 2009.

The valuation and pricing of cryptocurrencies and digitally native tokens and seets remains a non-trivial activity, due to the novelty of the platforms, millions of users and investors in the space. For the above reasons and the fact that is has proven to be also safe and resilient to network attack, we focused in particular to Bitcoin for our analysis.

Several researchers have aimed their effort in quantitative investigations to exploit information from cryptocurrencies prices time series and forecasting prices insights [99] or to predict the next most likely jump in value, range from theoretical models of pricing and adoption of digital tokens [12, 28, 32] to machine learning [3, 59] and neural network-driven [70] predictions of prices and returns.

In this field, the current state of art of research have yielded insights on the maturity, efficiency and structure of the cryptocurrency markets [11, 40, 106, 111, 118].

Special attention has been devoted to the analysis of factors underneath the high volatility of cryptocurrencies, ranging from extrapolating the mechanisms driving the fluctuations to the model estimation point of view [61, 71]: some research for instance highlighted a strong correlation with global economic activity [33, 114] and volume of trades [16], and with open source development communities supporting the cryptocurrencies [76, 90, 93].

In this part of Ph. D., we focus our investigation precisely on a comparison of four different machine learning algorithms for Bitcoin price classification, verifying whether the addition of technical indicators to the classic macroeconomic variables leads to an effective improvement in the prediction of Bitcoin price changes. The four machine learning algorithms implemented are: Support VectorMachine (SVM), XGBoost (XGB), Convolutional Neural Network (CNN) and a Long Short Term Memory neural network (LSTM).

Macroeconomic variables data are extracted from Yahoo Finance website spanning a 4-year interval with a hourly resolution, while technical indicators data are provided by the python *talib* library. For each algorithm we further compared a *restricted* vs *unrestricted* model. The *restricted model* consists of Bitcoin price movements using only macroeconomic variables : Close, Open, Low, High and Volume. The *unrestricted model* along with the macroeconomic variables includes also technical analysis variables: Simple Moving Average, Weighted Moving Average, Relative Strength Index, Rate of Change Percentage,

Momentum and On-Balance Volume.

In order to take into account the variance problem on test samples we implemented a cross validation technique which also allowed to evaluate a more reliable estimation of the model's performance. Furthermore, we fine-tuned all four algorithms using the Grid Search technique to find the best hyper-parameters values. The results were evaluated in terms of the well known classification metrics, i.e. accuracy, precision, recall and f1 score. Based on the results, it was possible to demonstrate that the multivariate case outperforms the univariate one, verifying that the addition of the technical indicators to the macroeconomic variables actually improves the accuracy on Bitcoin price classification.

This work is organised as follows. In Sec. 3.2.1, we describe how the affect metric time series are constructed and the preliminary analysis performed. In Sec. 3.2.8, we present the results and their implications. In Sec. 3.2.11, we discuss the limitations of this study.

3.2.1 Methodology

In this section we first introduce some notions on time series in order to better understand the analysis presented in the following and we present the dataset we used, including its pre-processing analysis. Finally we introduce our proposed algorithms with the metrics employed to evaluate their performance.

3.2.2 Collected data

We conduct our analysis on the Bitcoin hourly price series. We considered all the available macroeconomic variables, extracted from the *yahoofinance* website, for a 4-year period spanning from 2015/10/08 to 2019/10/03, for a total of 34922 observations. These macroeconomic features are listed below.

- *Close*: the last price at which the Bitcoin traded during the regular trading hour.
- *Open*: the price at which the Bitcoin first trades upon the opening of an exchange on a trading hour.
- *Low*: the lowest price at which the Bitcoin trades over the course of a trading hour.
- *High*: the highest price at which the Bitcoin traded during the course of the trading hour.
- *Volume*: the number of Bitcoin trades completed.

From the knowledge of these variables it was possible to calculate the technical indicators values thanks to some specific function provided by the *talib* python library. The selected technical analysis variables are listed below.

- Simple Moving Average (SMA): calculated as the arithmetic average of the Bitcoin closing price over some period (known as *timeperiod*.
- Weighted Moving Average (WMA): is a moving average calculation that more heavily weights recent price data.
- Relative Strength Index (RSI): is a momentum indicator that measures the magnitude of recent price changes to evaluate overbought or oversold conditions in the price of a stock or other asset (in this case Bitcoin price).
- Price Rate Of Change (ROC): it measures the percentage change in price between the current price and the price a certain number of periods ago.
- Momentum: is the rate of acceleration of a security's price or volume that is, the speed at which the price is changing (useful to identify trends).
- On Balance Volume (OBV): is a technical trading momentum indicator that uses volume flow to predict changes in stock price.

For each selected variable, except for the OBV feature, we set a *timeperiod* of 10.

3.2.3 Restricted versus Unrestricted Classification

The *restricted* classification analysis consists of classifying Bitcoin price movements using only macroeconomic variables, so the dataset in this case is made by the following features: *Close*, *Open*, *Low*, *High* and *Volume*. A *unrestricted* classification is a classification in which the dataset consists not only of the macroeconomic variables listed above, but also of technical analysis variables. In this case the purpose is to ascertain if the addition of the technical analysis features to the dataset can lead to an effective improvement in the Bitcoin price changes classification.

3.2.4 Exploratory Data Analysis

The target variable is a categorical variable that includes three unique classes that are listed in the following.

- Upwards movements. This class it has been encoded with 1 and represents a condition of rising prices.

- Stability. This category stands for price stability, when the price has not changed in the hour, and it has been encoded with 0.
- Downwards movements. This class, labeled with -1 represents a condition of falling prices.

Figure 3.6 shows the class distribution along the dataset, highlighting that we are dealing with a very imbalanced classification problem.

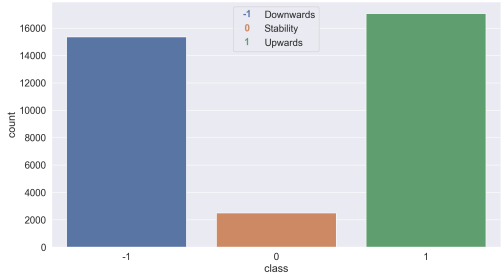


Figure 3.6: Classes distribution

Table 3.9 shows that most of the data are instances of classes 1 and -1, with 48,8% and 44% respectively, while the remaining 7,2% of the data are instances of class 0.

Table 3.9: Class instances counts and percentages

Classes	***Counts***	***Percentages***
1	17062	48,8%
-1	15355	44%
0	2504	7,2%

These findings led us to verify whether the instances of class 0 are evenly distributed along the dataset or whether they are more concentrated in some periods than in others. For this reason, the whole dataset, ranging from October 2015 to October 2019, was splitted into one year bins, with the purpose to test how many instances of class zero fall in each interval.

Table 3.10 and figure 3.7 show that price stability, in the Bitcoin market, is a market condition that occurred in the years immediately following the birth of the cryptocurrency, when its price behaviour was almost constant. Another thing that it is worth noting is that in this analysis we are working with hourly

Table 3.10: Distribution of *stability* class instances into 1-year bins

Bins	***Amount of* 0 *instances***
2015 - 2016	1838
2016 - 2017	545
2017 - 2018	66
2018 - 2019	55

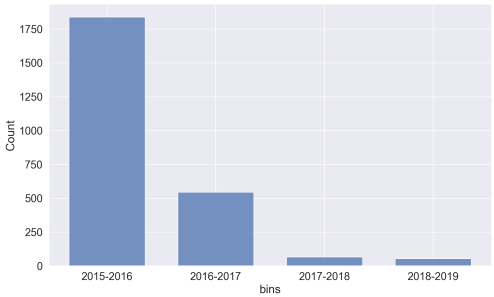


Figure 3.7: Count plot of *stability* class instances into 1-year bins

time series prices, therefore the condition of price stability is more likely than in the case with daily frequency dataset. Table 3.11 reports the results obtained with the same dataset, ranging from October 8, 2015 to October 3, 2019, but with daily frequency, showing that in 1456 data only one instance of class zero occurs, thus confirming our hypotheses.

Table 3.11: Class instances counts and percentages of daily data

Classes	***Counts***	***Percentages***
1	824	56,6%
-1	631	43,3%
0	1	0,1%

Considering that 73% of the 0 class instances fall in the first bin, ranging from October 2015 to October 2019, and that Bitcoin prices are characterised by high volatility, let us conclude that the condition of price stability in the current Bitcoin market is almost non-existent.

At this point, the dataset results in a dataset with a balanced class distribution. In this cases the accuracy metrics turns out to be a well-defined metric to evaluate the performance of a classification model.

3.2.5 Data pre-processing

Since we are dealing with a supervised learning problem we prepared our dataset in order to have a set of X inputs and y outputs with temporal dependence. The X inputs includes the predictors of the model, i.e. one or several values from the past, the so called *lagged* values, of the selected features discussed in the *Collected Data* section. The target variable y is a binary variable of zeros and ones. The 0 instance stands for downward price movements, which is a falling price condition. A 0 instance is obtained when the difference between the price at time t and the price at previous time $t-1$ is less than 0. The 1 instance represents upward price movements, which is a rising price condition. A 0 instance is obtained when the difference between the price at time t and the price at previous time $t-1$ is greater than 0. In both the *restricted* and *unrestricted* model, a number of lagged values equal to 5 were taken for each considered variable.

Principal Component Analysis

Principal component analysis (PCA) is a machine learning technique used to solve several tasks such as, exploratory data analysis problems and predictive model implementation. PCA is also one of the most commonly used technique for dimensionality reduction, that is a method used to reduce the number of input variables for a predictive model. This procedure may lead to a simpler predictive model with better performance, both in terms of a specific monitored prediction error and execution speed. PCA was introduced for the first time in 1901 by Pearson [96] and it was later developed by Hotelling [55], [56] in the 1930s. This technique is commonly used when dealing with data set with a large number of inputs, often very correlated. PCA consist of producing a small number of linear combinations Z_m with a specific method, where $m = 1, ..., M$, of the original predictors X_i, where $i = 1, ..., N$ and $M \leq N$. The Z_m are then used in place of the X_i of the original data set as inputs in the considered predictive model. The methods used to create the new shrinkage features differ in how the linear combinations are constructed, each leading to a different result. A popular approach to create these linear combinations Z_m is to use techniques from the field of linear algebra known as *feature projection* and the algorithms used are referred to as *projection methods*. These methods are able to reduce the number of dimensions in the feature space by performing a orthogonal projection of the data onto a lower dimensional linear space, such

that the variance of the projected data is maximized, whilst also capturing the most important structure and essence of the original data [13]. The resulting features Z_m can then be fed as inputs of a considered predictive model for the training phase.

In this analysis the *PCA* method [98] provided by the sklearn python library was used to implement the PCA technique. The *PCA* method was applied to our original data set and the resulting features were used to train all the implemented models.

3.2.6 Algorithms

Support Vector Machine

Support Vector Machines are supervised learning algorithms used for machine learning applications to solve classification and regression problems. The original *SVM* algorithm was introduced by Vapnik and Chervonenkis [112] in 1963. In 1992 Boser, Guyon and Vapnik [15] developed a new *SVM* version for non-linear classification problems by applying kernel functions to maximum margin hyperplanes. *SVM's* are based on statistical learning frameworks and they are one of the most robust and commonly used prediction methods in machine learning applications. The original *SVM* implementation is a linear classifier involving a linear kernel for linearly separable data. In this problems the input space can always be divided into a collection of regions with linear decision boundaries. In a p-dimensional space a linear classifier is a $(p-1)$-dimensional hyperplane described by equation 3.10, where w is the normal vector to the hyperplane, $(x_1, y_1), ..., (x_n, y_n)$ are n points of a given training data set and y_i is the class to which the point x_i belongs.

$$\mathbf{w}^T\mathbf{x} - b = \mathbf{0} \tag{3.10}$$

The hyperplane that best classifies the data is the one that represents the largest separation, better known as *margin*, between the classes. Therefore, this hyperplane is chosen so that the distance from it to the nearest data point on each side is maximized. If the data is linearly separable, then such a hyperplane exists and it is known as the maximum-margin hyperplane. In 1992 Boser, Guyon and Vapnik introduce an extension of the original *SVM* algorithm useful when dealing with non-linearly separable data where the classes overlap. In this cases the *SVM* produces non-linear boundaries by constructing a linear boundary in a large, transformed version of the feature space. More specifically, this extended version creates non-linear classifiers by simply replacing every dot product in the 3.10 with a non-linear kernel function. These kernel

functions allow the original space to be mapped to a higher dimensional feature space, thus making the separation between classes more obvious in this transformed space. The most commonly used kernels are listed below.

- Polynomial: $k(\vec{x}, \vec{y}) = (\vec{x} \cdot \vec{y})^d$
- Gaussian radial basis function (*rbf*): $k(\vec{x}, \vec{y}) = e^{-\frac{||\vec{x}-\vec{y}||^2}{2\sigma^2}}$
- Hyperbolic tangent: $k(\vec{x}, \vec{y}) = tanh(k\vec{x} \cdot \vec{y} + c)$, where $k > 0$ and $c < 0$

In this analysis we used the Support Vector Classification (*SVC*) method provided by the *sklearn* python library.

XGBoost

We refer to the term *XGBoost* as a gradient boosting algorithm, but actually *XGBoost* is a famous open-source software library which provides a gradient boosting framework for several programming languages including *C++*, *Java*, *Python*, *R*, *Julia*, *Perl* and *Scala*. *XGBoost* was initially conceived as a research project by Tianqi Chen [24] with the purpose of providing a terminal application which could be configured using a suitable configuration file. Later it became one of the most famous and used libraries for machine learning applications with several package implementations for different programming languages such as *Python*, *R*, *Java*, *Scala* and so on and so forth. Gradient boosting is one of the most powerful machine learning techniques for regression and classification problems. This technique was firstly introduced by Leo Breiman [18] by observing that boosting can be interpreted as an optimization algorithm on a suitable cost function. The main intuition behind boosting is a procedure that combines the outputs of many "weak" classifiers to produce a powerful "committee" [52]. A weak classifier is one whose error rate is only slightly better than random guessing, typically decision trees. The purpose of boosting is to sequentially apply the weak classification algorithm to repeatedly modified versions of the data, therefore producing a sequence of weak classifiers. The predictions from all of them are then combined through a weighted majority vote to produce the final prediction. In this analysis we implemented the Python API reference of *xgboost* provided by the *sklearn* python library.

Convolutional Neural Network

Convolutional Neural Network (*CNN*) is a specific class of neural networks most commonly used for deep learning applications involving image processing, image classification, natural language processing and financial time series. The

most important part in the CNN network is the *convolutional* layer that gives the network its name. This layer employs a mathematical operation called *convolution*. In this context, a convolution is a linear operation that involves a multiplication between a matrix of input data and a two-dimensional array of weights, known as filter. This networks use convolution operation in at least one of their layers.

Convolutional neural netowrks have a similar architecture to traditional neural networks, including an input and an output layer, as well as multiple hidden layers. The main feature of a CNN is that their hidden layers typically consist of a series of convolutional layers that convolve with the multiplication described above.

For their implementation we used the Keras framework [27] for deep learning. Our model consists of two stacked 2-dimensional CNN layers, one densely connected layer with 64 neurons and finally the densely connected output layer with one neuron.

Long Short Term Memory

Long Short-Term Memory networks are a prominent variations of Recurrent Neural Network (RNN) used in the field of deep learning. RNN's are a class of artificial neural network with a specific architecture specialized in recognizing patterns in sequences of data of various kinds: texts, genomes, handwriting, the spoken word, or numerical time series data emanating from sensors, markets or other sources. The main disadvantage of traditional recurrent neural networks is their inability to capture long-term dependencies in a sequence of data, thus they perform well only for short-term memory dependencies. On the contrary, LSTM networks are a special kind of RNN, able at learning long-term dependencies. The model is organized in cells which include several operations. LSTM hold an internal state variable, which is passed from one cell to another and modified by Operation Gates (forget gate, input gate, output gate). These gates control how much of the internal state is passed to the output and work in a similar way to other gates. These three gates have independent weights and biases, hence the network will learn how much of the past output and of the current input to retain and how much of the internal state to send out to the output. In this analysis we used the Keras framework [27] for deep learning. Our model consists of one stacked LSTM layer and the densely connected output layer with one neuron.

3.2.7 Hyper-parameters tuning

The *hyper-parameters tuning* technique is a technique for optimising the hyper-parameters of a given algorithm in order to identify the best hyper-parameters configuration that allow the algorithm to achieve the best performance, in terms of the monitored prediction error. For each implemented algorithm, the hyper-parameters to be optimised are selected, and for each hyper-parameter an appropriate searching interval is defined, including all the values to be tested. Once the hyper-parameters of the model with their searching intervals have been selected, the algorithm is fitted on a specific portion of the data set with the first chosen hyper-parameter configuration, after that the fitted model is tested on a portion of data not seen during the training phase. This test procedure returns a specific value for the chosen prediction error.

Table 3.12: SVM's hyper-parameter searching intervals

SVM	***hyper-parameters***		
	C coefficient	**kernel**	**gamma (kernel coefficient for *rbf*)**
searching interval	(1, 5, 10, 100)	(rbf, linear)	(0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9)

To make the hyper-parameter optimisation procedure as robust as possible, a model validation technique is used, for assessing how the performance achieved by a given model will generalize to an independent data set. This validation technique involves the partition of a data sample into a training set, used to fit the model, and a test set used to validate the fitted model. In our analysis we implemented the *KFold cross-validation* method provided by the *sklearn* python library.

Table 3.13: XGB's hyper-parameter searching intervals

XGB	***hyper-parameters***		
	booster	**eta**	**max_depth**
searching interval	(gbtree, gblinear, dart)	(0.1, 0.2, 0.3, 0.4)	(4, 5, 6, 7)

The optimisation procedure ends when all possible combinations of hyper-parameter values have been tested. The hyper-parameter configuration that

will be chosen will be the one with which the algorithm has achieved the best performance in terms of the chosen prediction error. Tables 3.12, 3.13, 3.14 and 3.15 show the chosen searching intervals for each implemented algorithm.

Table 3.14: CNN's hyper-parameter searching intervals

CNN	***hyper-parameters***				
	epochs	**batch_size**	**optimizer**	**activation**	**neurons**
searching interval	(100, 200, 300)	(32, 64, 128)	(adam, Nadam, Adamax)	(relu, tanh)	(8, 16, 32)

Table 3.15: LSTM's hyper-parameter searching intervals

LSTM	***hyper-parameters***				
	epochs	**batch_size**	**optimizer**	**activation**	**neurons**
searching interval	(50, 100, 200)	(32, 64, 128)	(adam, Nadam, Adamax)	(relu, tanh)	(64, 128)

3.2.8 Results and Discussion

In this section the results obtained for the restricted and unrestricted model are reported, evaluated in terms of the well-known classification error metrics, namely *accuracy*, *f1_score*, *precision* and *recall*. For each of these metrics we report their mean and standard deviation obtained on the 10 folds in which the dataset was split during the *Gridsearch* procedure for the identification of the best hyperparameter configuration.

3.2.9 Restricted Model

Table 3.17 shows the best results obtained for the Neural Networks models, through the *Gridsearch* technique in terms of the classification error metrics. The best identified parameters with the related results obtained for the *SVM* and *XGBoost* models are reported in table 3.18 and 3.19 respectively.

The neural network that achieved the best accuracy is *CNN*, with an average accuracy of 53.7% and a standard deviation of 2.9%. Among the implemented machine learning models, the one that achieved the best accuracy is *SVM*, with an average accuracy of 54% and a standard deviation of 2.8%. In the restricted

analysis, the model that produced the best performance is *SVM*. It is worth noting that the model's performances in terms of the prediction error is quite similar, with an average accuracy of around 54%, while from the point of view of execution speed, the *SVM* and *XGB* models outperforms neural networks.

Table 3.16: Unrestricted model - *SVM* and *XGB* results

SVM				***XGBoost***			
accuracy ($\mu \pm \sigma$)	***f1_score*** ($\mu \pm \sigma$)	***precision*** ($\mu \pm \sigma$)	***recall*** ($\mu \pm \sigma$)	***accuracy*** ($\mu \pm \sigma$)	***f1_score*** ($\mu \pm \sigma$)	***precision*** ($\mu \pm \sigma$)	***recall*** ($\mu \pm \sigma$)
(0.546 ± 0.027)	(0.420 ± 0.170)	(0.537 ± 0.023)	(0.404 ± 0.224)	(0.546 ± 0.029)	(0.374 ± 0.194)	$()0.539 \pm 0.052$	(0.354 ± 0.255)

Table 3.17: Restricted model - Neural networks results

Model	***epochs***	***batch size***	***optimizer***	***activation***	***neurons***	***accuracy*** ($\mu \pm \sigma$)	***f1_score*** ($\mu \pm \sigma$)	***precision*** ($\mu \pm \sigma$)	***recall*** ($\mu \pm \sigma$)
CNN	50	32	Nadam	tanh	16	(0.537 ± 0.029)	(0.472 ± 0.143)	(0.511 ± 0.025)	(0.495 ± 0.027)
LSTM	300	128	Adamax	relu	128	(0.535 ± 0.034)	(0.456 ± 0.200)	(0.485 ± 0.082)	(0.503 ± 0.285)

Table 3.18: Restricted model - *SVM* and *XGB* best identified parameters

SVM			***XGBoost***		
C coefficient	**kernel**	**gamma**	**booster**	**eta**	**max_depth**
100	rbf	0.1	gbtree	0.2	4

3.2.10 Unrestricted Model

Table 3.20 shows the best results obtained for the Neural Networks models, through the *Gridsearch* technique in terms of the classification error metrics. The best identified parameters with the related results obtained for the *SVM* and *XGBoost* models are reported in table 3.21 and 3.16 respectively.

The results obtained for the unrestricted model, highlights that in general the addition of technical analysis variables to the dataset leads to an effective improvement in the average accuracy, namely the prediction error. This finding occurs for all the implemented models and this allows one to exclude that

Table 3.19: Restricted model - *SVM* and *XGB* results

SVM				***XGBoost***			
accuracy ($\mu \pm \sigma$)	***f1_score*** ($\mu \pm \sigma$)	***precision*** ($\mu \pm \sigma$)	***recall*** ($\mu \pm \sigma$)	***accuracy*** ($\mu \pm \sigma$)	***f1_score*** ($\mu \pm \sigma$)	***precision*** ($\mu \pm \sigma$)	***recall*** ($\mu \pm \sigma$)
(0.540 ± 0.028)	(0.385 ± 0.206)	(0.515 ± 0.035)	(0.374 ± 0.242)	(0.528 ± 0.025)	(0.521 ± 0.080)	(0.504 ± 0.038)	(0.555 ± 0.135)

Table 3.20: Unrestricted model - Neural networks results

Model	***epochs***	***batch size***	***optimizer***	***activation***	***neurons***	***accuracy*** ($\mu \pm \sigma$)	***f1_score*** ($\mu \pm \sigma$)	***precision*** ($\mu \pm \sigma$)	***recall*** ($\mu \pm \sigma$)
CNN	50	128	adam	tanh	16	(0.547 ± 0.025)	(0.384 ± 0.180)	(0.538 ± 0.033)	(0.356 ± 0.241)
LSTM	50	64	Nadam	relu	32	(0.545 ± 0.027)	(0.435 ± 0.175)	(0.541 ± 0.060)	(0.430 ± 0.228)

Table 3.21: Unrestricted model - *SVM* and *XGB* best identified parameters

SVM			***XGBoost***		
C coefficient	**kernel**	**gamma**	**booster**	**eta**	**max_depth**
1	rbf	0.1	gblinear	0.4	6

this result is a statistical fluctuation and its dependence on the particular forecasting algorithm implemented. The best result obtained with the unrestricted model is achieved by the *CNN* model, with a mean accuracy of 54.7% and a standard deviation of 2.5%. It is worth noting that in the unrestricted analysis, neural networks models outperforms machine learning models in terms of prediction error, while from the point of view of execution speed *SVM* and *XGB* models still outperforms neural networks.

3.2.11 Threats to Validity

Threats to external validity concern the generalisation of our results. In this study, we analysed the time series of only one cryptocurrency and thus different ecosystems may yield different results.

Threats to internal validity concern confounding factors that can influence the obtained results. Based on empirical evidence, we assume that technical indicators and classic macroeconomic variables are exhaustive for our model but there are other factors which may influences the price movements and are not taken into account in this study.

Threats to construct validity focus on how accurately the observations describe the phenomena of interest. The detection and classification of price's movements are based on objective data that describe the whole phenomena with respect of the time behaviour analyzed.

3.3 On the Mutual-Influence between Blockchain Development Communities and Cryptocurrency Price Changes

This thesis work aims to identify and model relationships between cryptocurrencies market price changes and topic discussion occurrences on social media. The considered cryptocurrencies are the two highest in value at the moment, Bitcoin and Ethereum. At the same time, topics were realized through a classification of the comments gained from the *Reddit* social media platform, implementing a Hawkes model. The results highlight that it is possible to identify some interactions among the considered features, and it appears that some topics are indicative of certain types of price movements. Specifically, the discussions concerning issues about government, trading and Ethereum cryptocurrency as an exchange currency, appear to affect Bitcoin and Ethereum prices negatively. The discussions of investment appear to be indicative of price rises,

while the discussions related to new decentralized realities and technological applications is indicative of price falls.

Cryptocurrencies arouse keen interest not only in the scientific and financial fields but also within social media communities, making the analysis of their price behaviours one of the most discussed topics of the last few years. Over the years, several approaches have been developed in seeking to forecast cryptocurrency price movements [68, 69]. S. McNally et al. tried to ascertain with what accuracy the direction of Bitcoin price in USD can be predicted using machine learning algorithms like LSTM (Long short-term memory) and RNN (Recurrent Neural Network) [79]. Naimy and Hayek tried to forecast the volatility of the Bitcoin/USD exchange rate using GARCH (Generalized AutoRegressive Conditional Heteroscedasticity) models [85]. Several are also the studies that tried to use online information, including social media topics discussions, to predict cryptocurrencies price changes, for example, Google searches for Bitcoin-related terms have been shown to have a relationship with the Bitcoin price [67]. D. Garcia and F. Schweitzer have considered the strength and polarisation of opinions displayed in Twitter submissions, founding that an increase in the polarisation of sentiment (disagreement of sentiment) preceded a rise in the price of Bitcoin [45]. In another work, several machine learning pipelines were implemented with the objective of identifying cryptocurrency market movement, in order to prove whether Twitter data relating to cryptocurrencies can be utilized to develop advantageous crypto coin trading strategies [108]. F. Valencia et al. proposed to use most common machine learning tools, such as neural networks, support vector machines and random forest, and available social media data for predicting the price movement of the Bitcoin, Ethereum, Ripple and Litecoin cryptocurrency market movements, showing that it is possible to predict cryptocurrency markets using machine learning and sentiment analysis and that, in this case study, neural networks outperform the other models [44]. A recent exploration monitored the activity on the social media platform Reddit in order to detect the epidemic-like spread of investment ideas beneficial in the prediction of cryptocurrency price bubbles [101]. Thanks to the many works in literature, which helped to prove the existence of possible cause-effect relationships between the cryptocurrency price changes and online information, we can state that the knowledge of the discussion topics that affect price would be a useful component of any trading model.

The purpose of this work is to ascertain the existence of possible relationships between cryptocurrency market prices and social media discussions in order to understand what topics have the potential to predict price movements. It is in fact well known that developers moods can affect software quality [91] and if this concept is applied in the field of cryptocurrency software production

and to their quality metrics [38] this may affect cryptocurrency market prices as well. The first step was to retrieve the discussions comments from the social media platform *Reddit*, which has been shown to be one of the most valuable sources of information relating to cryptocurrency markets. Secondly, the occurrence of particular topics from social media content were evaluated, using dynamic topic modelling, that is an extension of Latent Dirichlet Allocation (LDA). Finally, a Hawkes model was implemented to identify hidden interactions between topics and cryptocurrency market prices.

3.3.1 Methodology

3.3.2 Data Sources

The social media platform *Reddit* is an American social news aggregation, web content rating, and discussion website that reaches about 8 billion page views per month. Reddit is built over multiples subreddits, where each subreddit is dedicated to the discussion of a particular subject. Therefore, there are specific subreddits related to major cryptocurrency projects. In this work, for each cryptocurrency considered, two subreddits are analyzed, one technical and one trading related. In Tab. 3.22 are shown the considered subreddits. For each sub-

Table 3.22: Considered subreddits

Cryptocurrency	***Technical Discussions***	***Trading Discussions***
Bitcoin	r/Bitcoin	r/BitcoinMarkets
Ethereum	r/Ethereum	r/EthTrader

reddit a given amount of comments were fetched, for a total of almost one million of comments analyzed. Regarding the cryptocurrencies choice, we decided to use the two highest in value at the moment: Bitcoin and Ethereum. The historical price data were extracted from the "Historical Data" section available on *Crypto Data Download* website, specifically from the *Coinbase* trading exchange. The hourly prices time series were retrieved, stored and then aggregated to the required granularity. The sample period considered in this work is the entire 2019 year, that is from January 1^{th} 2019 to December 31^{th} 2019. The chosen data period is a suitable one, since in September 2019 a significant fall in the Bitcoin price occurred with consequent ripple effects on Ethereum prices, allowing us to investigate the interaction between prices and social media during this considered period.

3.3.3 Topic Modelling

A topic model is a specific statistical model used to identify the abstract topics within a collection of documents. Topic modelling is a frequently used text-mining tool for discovery themes occurring within a corpus automatically, finding a distribution of words in each topic and the distribution of topics in each document. In this work, we used the *Latent Dirichlet Allocation* (LDA) [34], which is a popular unsupervised learning technique for topic modelling, introduced in 2003. This type of topic model assumes that each document contains multiple topics to different extents. In the following, we briefly discuss, for the sake of brevity, the generative process by which LDA assumes each document originates.

- The first step is to choose, for each document, the number of words N to generate.
- The process then randomly chooses a distribution over topics. This parameter is usually labelled as θ.
- Finally, for each word to be generated in the document, the process randomly chooses a topic, Z_n, from the distribution of topics, and from that topic chooses a word, W_n, using the distribution of words in the topic.

If we consider a given document d and topic t, so the variables of interest in this model are the distribution of topic t in document d and the distribution of words in topic t. These variables are latent, hidden parameters that can be estimated via inference for any specific dataset. It has been proven that the standard LDA model can not understand both the ordering of words within a document and the ordering of documents within a corpus. For this reason, in 2006, an extension of this model was developed [14]. This LDA model extension is known as *dynamic topic model* and even if it still has no understanding of the order of words in a document, at least the order of documents in the corpus is accounted for.

3.3.4 Hawkes Model

The Hawkes process is a point process class [88], also known as a self-exciting counting process, in which the impulse response function explicitly depends on past events [53]. In this type of process, the observation of an event causes the increase of the process impulse function. From a mathematical point of view, a point process is a Hawkes process if the impulse function $\lambda\left(t|H_t\right)$ of the

process takes the form of (3.11).

$$\lambda\big(t|H_t\big) = \lambda_0(t) + \sum_{i:t_i<t} \phi\big(t - t_i\big) \tag{3.11}$$

In equation (3.11) H_t represents the history of given past events, $\lambda_0(t)$ is a positive function that determines the basic intensity of the process and ϕ is another positive function known as *memory kernel*, since it depends on past events occurred before time t. Hawkes models can be used to identify dynamics interactions between a group of K processes. The occurrence of an event on a particular process can cause an impulse response on that process (self-excitation), determining an increase of the likelihood of further events, and on other processes (mutual-excitation). Given events occurring on a number of processes, a Hawkes model can be used in order to quantify previously hidden connections between the processes. In this work, we applied a Hawkes model with the purpose of deciphering how topics are related to one another, and how price changes are related to the topic occurrence. Once the Hawkes model is fitted on the consider data, it will contain some weights representing the directional strength of any interaction between processes interpreted as the expected number of events on a specific process resulting from an event on another process.

3.3.5 Results

Before applying topic modelling, the corpus has been pre-processed. Topics were therefore obtained removing stop words (such as the word “the”), links, special characters and varied punctuation. Part-of-speech (POS) tagging was used to categorize words into types; nouns and adjectives are maintained while other types are removed. Furthermore, stemming techniques were applied in order to reduce derived words to their base root. These techniques allow to group different terms into one unique root term and thus to simplify the number of features, to increase attention to the most critical terms. After this data pre-processing step, we applied the *LdaModel* method provided by the *Gensim* python library [102] in order to identify distinct topics through topic modelling technique, thus generating a time series of topic occurrence. For insight into this topics selection process, Fig. 3.8 shows all topics selected for their coherent cryptocurrency-related content. For the sake of brevity, we only report those for the *r/Bitcoin* subreddit.

The chosen topics are then analysed in a Hawkes model, alongside market prices. We used the *HawkesConditionalLaw* method provided by the *tick* python library [6]. Once the topics were created, the creation of the features in

r/Bitcoin		
N. Topic	Label	Topic words
0	personal investment	'money', 'time', 'wallet', 'way', 'thing', 'work', 'coin', 'transact', 'crypto', 'point', 'actual', 'try', 'mean', 'mine', 'sure', 'person'
1	Bank	'price', 'exchange', 'thank', 'day', 'dollar', 'atm', 'wrong', 'shit', 'purchase', 'withdraw', 'satoshi', 'check', 'list', 'demand', 'name', 'hope', 'google', 'com', 'lt'
2	Bitcoin & Blockchain	'bitcoin', 'btc', 'year', 'new', 'look', 'post', 'currency', 'question', 'remove', 'address', 'node', 'network', 'change', 'free', 'month', 'internet', 'user', 'power', 'believe'
3	Government	'people', 'good', 'market', 'value', 'govern', 'right', 'world', 'account', 'reason', 'everyone', 'country', 'maybe', 'talk', 'idea', 'guy', 'last'
4	Trading	'gt', 'use', 'bank', 'pay', 'fee', 'cash', 'gold', 'trade', 'tax', 'scam', 'lol', 'need', 'word', 'coinbase', 'rate', 'kyc', 'ask', 'trust'

Figure 3.8: Selected topics from *r/Bitcoin* subreddit

events and processes was performed. Data were aggregated into sixty-minute groups ($\Delta t = 60 min$).

Furthermore, two features were processed from each cryptocurrency prices, namely the $delta\_price$ feature, which is the difference between the close price and the hourly open price, and the log_return feature, i.e. the logarithm of the difference between higher price and hourly lower price. We then considered a total of fourteen features, five topics and two price feature for each cryptocurrency. Let us label the maximum time parameter for which an individual event can affect with max_lag. In this work, different values of max_lag were tested. We chose these max_lag values according to the length of the period and the number of events associated per interval.

Fig. 3.9 shows the connections strength between the considered processes for Bitcoin and Ethereum technical discussions for a max_lag of 24. The coefficients of the Hawkes matrix represents the weights extracted from the Hawkes model fitted to the dataset and they are displayed from the vertical to the horizontal axis. There is a general pattern of soft *self-excitement* positive relationships between all the variables, highlighted by the coefficients placed on the diagonal. Some causal relationship are established between $topic\_0$ (related to discussion investments) of the Bitcoin and $topic\_3$ and 4 of Ethereum (related to discussions about decentralized realities and deployment applications).

The Bitcoin and Ethereum log_return feature is negatively affected by Bitcoin $topic\_0$ feature, while positively affected by Ethereum $topic\_3$ and 4. The results obtained with a max_lag of 48, shown in Fig. 3.10, highlights that the *self-excitement* relationship are no longer present and the appearance of some interesting causal relationship between btc_log_return feature and Bitcoin and Ethereum $delta\_return$ features. It also appears that Bitcoin and Ethereum topics negatively affect the $delta\_price$ feature of both cryptocur-

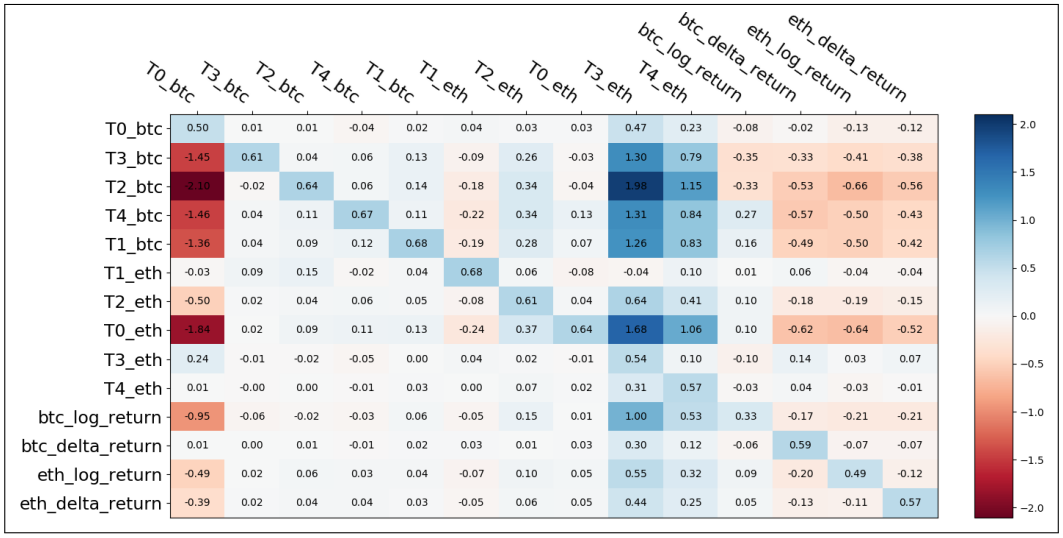


Figure 3.9: Hawkes matrix for *r/Bitcoin* and *r/Ethereum* with max_lag 24.

rencies.

Fig. 3.11 shows the connections strength between the considered processes for Bitcoin and Ethereum trading discussions for a max_lag of 24. In this case the relationships between topics features and the variables related to the prices are almost zero, instead the *self-excitement* relationships between all the features are stronger than those occurred in the technical discussion case.

In Fig. 3.12 are reported the results obtained with a max_lag of 48 always for Bitcoin and Ethereum trading discussions.

It appears that there are no substantial differences compared to the previous case with max_lag equal to 24; the only interesting observation that is worth noting is that all the negative relationships established between the topics features become positive in this case.

A study focused only on September 2019 period was also performed, since in September 24 a significant fall in the Bitcoin price was occurred with several ripple effects on Ethereum prices.

Specifically, the Hawkes model was fitted in different time period:

- In the period before the price fall, that is from 1^{th} to 20^{th} September with a max_lag of 24.
- To the period just before the fall, 21−23 September with a max_lag equal to 12.
- To the turbulent period of 24 – 26 September, with a max_lag of 6.

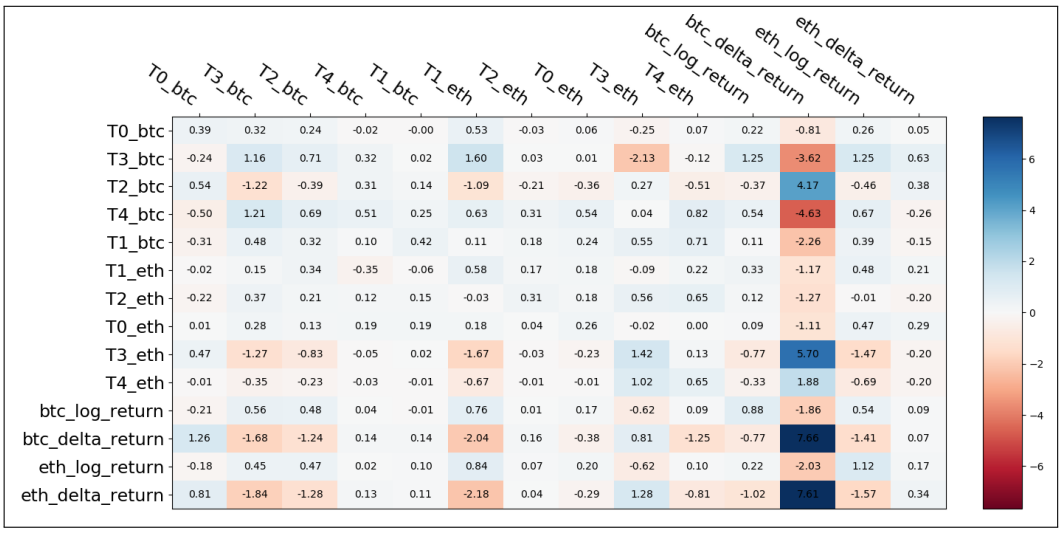


Figure 3.10: Hawkes matrix for *r/Bitcoin* and *r/Ethereum* with max_lag 48.

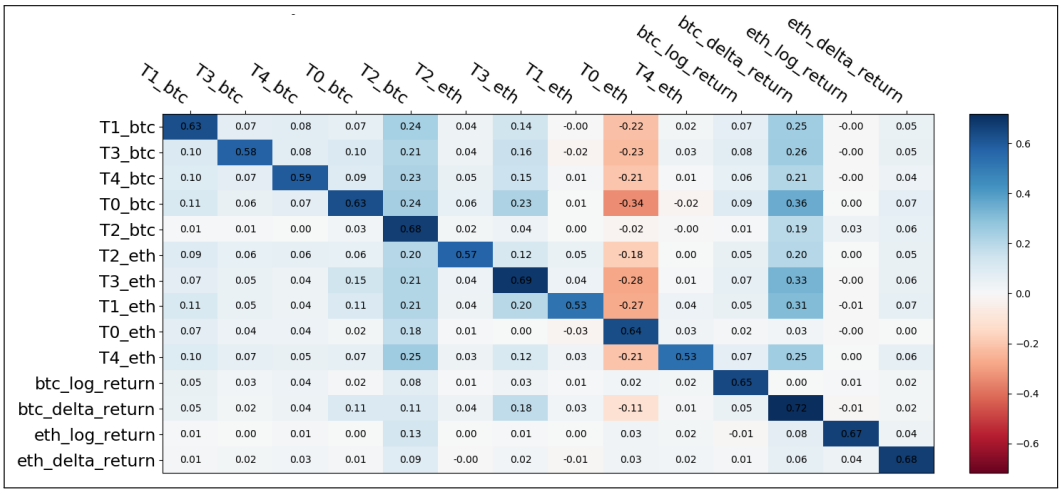


Figure 3.11: Hawkes matrix for *r/BitcoinMarkets* and *r/EthTrader*, max_lag 24.

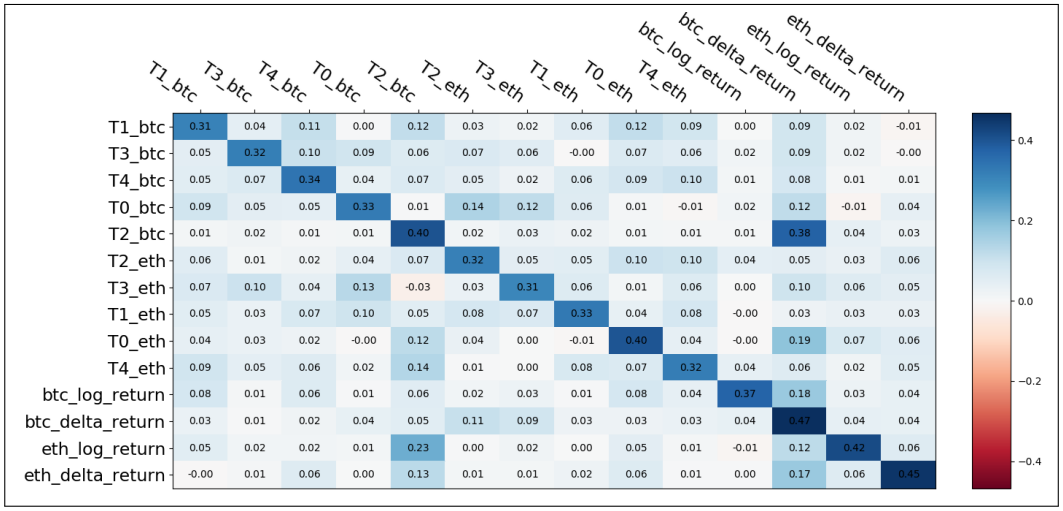


Figure 3.12: Hawkes matrix for *r/BitcoinMarkets* and *r/EthTrader*, max_lag 48.

For the sake of brevity, we only discuss the results without showing the Hawkes matrices, since they are similar to those already analyzed. Let us start by discussing the results related to the technical discussions. The results obtained for the period before the price fall, with a max_lag of 24, does not show relevant information, In general, only Bitcoin's $topic\_0$ (personal investment) and Ethereum's $topic\_3$ (decentralized AI) features appear to affect other variables, but only weakly on prices. More interesting results are obtained for the period just before the Bitcoin price fall, i.e. the period between $20-23$ September with max_lag set to 12. It appears that the Hawkes model fitted in this periods yields strong weights for the log_return features, above all on $topic\_0$ with positive effects and on Ethereum $topic\_3$ and 4 with significant negative effects. In this period, the relationships between *topics* and the features related to the cryptocurrency prices are almost zero, while very strong causal effects among the topics are evident. The analysis related to the trading discussions performed in the turbulent period of $24-26$ September with a max_lag of 6, shows that Ethereum $topic\_2$ and 3 (Ethereum domain and Government) strongly influence the Ethereum $delta\_return$ and the Bitcoin log_return features.

3.4 On the Impact of Development Practices on Cryptocurrency Prices

The network of developers in distributed ledgers and blockchains open source projects is essential to maintaining the platform: understanding the structure of their exchanges, analysing their activity and its quality (e.g. issues resolution times, politeness in comments) is important to determine how "healthy" and efficient a project is. The quality of a project affects the trust in the platform, and therefore the value of the digital tokens exchanged over it.

In this Ph.D. work, we investigate whether developers' emotions can effectively provide insights that can improve the prediction of the price of tokens. We consider developers' comments and activity for two major blockchain projects, namely Ethereum and Bitcoin, extracted from Github. We measure sentiment and emotions (joy, love, anger, etc.) of the developers' comments over time, and test the corresponding time series (i.e. the affect time series) for correlations and causality with the Bitcoin/Ethereum time series of prices. Our analysis shows the existence of a Granger-causality between the time series of developers' emotions and Bitcoin/Ethereum price. Moreover, using an artificial recurrent neural network (LSTM), we can show that the Root Mean Square Error (RMSE) – associated with the prediction of the prices of cryptocurrencies – significantly decreases when including the affect time series.

The ecosystem of cryptocurrencies traded and exchanged every day has been exponentially growing over the past ten years. The platforms – distributed ledgers and blockchains – cryptocurrencies rely upon to be created and transferred are developed (in most cases) in the form of open source projects. Developers from across the globe are constantly contributing to open source projects maintaining the codes and software that ensure the platform's correct functioning. According to a Deloitte report[2], there are currently more than 6500 active projects connected to distributed ledger technologies (DLT) and blockchains. The pioneering ones are the Bitcoin and Ethereum projects, whose associated tokens also dominate the crypto scene by market capitalisation.

Investors are attracted to this technology, not only by its future outlooks and potential but also by the excess returns on their investments that can be achieved by exploiting the highly volatile crypto-market. Nonetheless, valuation and pricing of cryptocurrencies and digitally native tokens remains a non-trivial task, due to the peculiarity of the platforms, users and investors in the space.

Quantitative investigations aimed at extracting information from the time

[2] https://www2.deloitte.com/content/dam/insights/us/articles/4600_Blockchain-five-vectors/DI_Blockchain-five-vectors.pdf

series of cryptocurrency prices and predicting prices drivers [99] or the next most likely jump, range from theoretical models of pricing and adoption of digital tokens [12, 28, 32] to machine learning [3, 59] and neural network-driven [70] forecasts of prices and returns. Analyses of the cryptocurrency markets [40, 106, 111] yielded insights on their maturity, efficiency and structure. A large body of literature is also looking at the volatility of cryptocurrencies, from the model estimation point of view [61, 71] as well as by extrapolating the mechanisms driving the fluctuations. Studies showed, for example, a strong correlation with global economic activity [33, 114] and volume of trades [16].

In the crypto space, where everything is decentralised and shared in a peer-to-peer fashion between users, developers and investors, “social" aspects appear to play a crucial role: discussions about platforms' quality are held over public forums (e.g. Reddit), news about next developments are shared over the informal news channel of Twitter and updates on development activities are publicly accessible over open source development platforms such as Github. Investors' sentiment and trading activities, which in turn impact prices, are, therefore, inevitably informed and influenced via those channels. For this reason, new types of data have been recently used to improve models and predictions. “Social" sentiment is extracted using data gathered from users' online communities [65] – e.g. online forums such as BitcoinTalk[3] – and from online news and tweets [5, 63, 72]. For instance, a suitably built sentiment index can be used to test for speculative bubbles in cryptocurrency prices [23]. More broadly, Google search data related to cryptocurrencies can be relevant to characterise the set of Bitcoin users [119]. Temporal topic analysis of Bitcoin and Ethereum discussions on Reddit also show correlations with variations of cryptocurrency prices [100].

Developers in open source blockchain and DLTs projects are also crucial entities, responsible for the maintenance and updates of the platforms. The idea that the human aspects of software development are of paramount importance to ensure high team productivity, software quality and developers satisfaction is already well-established in software engineering [39, 49, 82]. These studies have shed light on the importance of all the social and human aspects associated with the software development processes, and empirically demonstrated how a positive environment may have an impact on team productivity, software quality and developers' satisfaction [47, 64]. Moreover, standard metrics extracted from open source development platforms such as Github[4] and Bitbucket[5] can be used to rank the top crypto tokens [89]: metrics include number

[3] `https://bitcointalk.org`
[4] `https://github.com`
[5] `https://bitbucket.org`

of commits and issues, forks and number of contributors to the code.

Tools to extract sentiment specifically built for the software engineering domain and language are also available. For example, Murgia et al. [82] demonstrated the feasibility of a machine learning classifier using emotion-driving words and technical terms to identify developers' comments containing gratitude, joy and sadness. Islam et al. [58] studied how developers' emotions affect the software development process. They reconstructed the emotional variations by means of a sentiment analysis on commit messages using the SentiStrength tool [37], showing how emotions influence different typologies of software development tasks.

In this analysis, we focus our investigation precisely on the impact of developers' activities and emotions, sentiment and politeness on the cryptocurrencies issued and transferred over the platform they contribute to develop. In particular, we consider comments written by GitHub contributors of the two main blockchain projects, Bitcoin and Ethereum, and we perform emotions mining (love, joy, anger, sadness), sentiment analysis [82], politeness and VAD analysis[6] of the comments [39, 75]. In the following, we will generally refer to emotions, sentiment, politeness and VAD metrics as affect metrics, in line with recent works in psychology and computer science (e.g. [48, 104]), where *affect* is an umbrella term for discrete emotional states as well as emotional dimensions and moods. In Sec. 3.4.1, we will describe in more details the meaning of the affect metrics and how they are measured.

The main idea of this study is to understand whether emotions mining, sentiment analysis, politeness, and VAD analysis can be used to improve the prediction power of machine learning algorithms for the returns of the Bitcoin/Ethereum cryptocurrency. More generally, these metrics could be useful to monitor the health and quality of projects and platforms from a software engineering point of view.

We aim at understanding the interplay between developers' affect and cryptocurrency returns and we will focus on the two following aspects:

- Does the affect of Bitcoin and Ethereum communities influence variations in returns?
  Using Granger causality tests we will show that the affect metrics extracted from the contributors' comments influence the cryptocurrency prices.

- Is the affect of Bitcoin and Ethereum communities able to improve the error on the prediction of returns?

[6] Valence, Arousal and Dominance: these metrics are used to respectively evaluate the (i) engagement, (ii) confidence and (iii) responsiveness of a person in conducting a task or an activity. More details will follow in Sec. 3.4.2.

Using a LSTM neural network we will show that including the affect time series as features in the training set significantly improves the prediction error.

This work is organised as follows. In Sec. 3.4.1, we describe the dataset, the process to construct the affect time series and the tools used for the analyses (Granger causality test and Long-Short term memory for the prediction of returns). In Sec. 3.2.8, we present the results and their implications. In Sec. 3.2.11, we discuss the limitations of this study.

3.4.1 Dataset and Methods

In this section, we describe how affect time series are constructed using the comments of Ethereum and Bitcoin developers on Github for the period of December 2010 to August 2017.

Both the Bitcoin and Ethereum projects are open source, hence the code and all the interactions among contributors are publicly available on GitHub [92]. Active contributors are continuously opening, commenting on, and closing so-called "issues". An issue is an element of the development process, which carries information about discovered bugs, suggestions on new functionalities to be implemented in the code, or new features actually being developed. Monitoring the issues constitutes an elegant and efficient way of tracking all the phases of the development process, even in complicated and large-scale projects with a large number of remote developers involved. An issue can be "commented" on, meaning that developers can start sub-discussions around it. They normally add comments to a given issue to highlight the actions being undertaken or to provide suggestions on its possible resolution. Each comment posted on GitHub is timestamped, hence it is possible to obtain the exact time and date and generate a time series for each affect metric considered in this study.

An example of a developer's comment extracted from Github for Ethereum can be seen in Table 3.23. Quantitative measures of sentiment and emotions associated with the comments, as reported in this example, are computed using state-of-the-art tools of textual analysis (further details below). The affect metrics computed for each comment are emotions such as love (L), joy (J), anger (A), sadness (S), VAD (valence (Val), dominance (Dom), arousal (Ar)), politeness and sentiment (Pol and Sent respectively).

The Bitcoin and Ethereum price time series were extracted from the API of CoinMarketCap [7] using daily closing prices.

[7] `https://coinmarketcap.com/`

Comment	L	J	A	S	Val	Dom	Ar	Pol	Sent
Perhaps there's simply nothing new to translate? The reason I updated Transifex in the first place was to be sure the strings with subtle English changes (that don't change the meaning) didn't reset the translation - so those were imported from the old translations. Though I seem to recall at least one truly new string - Transaction or such.	0	0	0	1	1.93	1.88	1.26	imp	1

Table 3.23: Example of comments and the corresponding values of affect (love (L), joy (J), anger (A), sadness (S)), VAD (valence (Val), dominance (Dom), arousal (Ar)), politeness and sentiment (Pol and Sent respectively).

3.4.2 Measuring Affects Metrics

In our analysis, we focus on four main classes of affect metrics: emotions (love, joy, anger, sadness), VAD (valence, arousal, dominance), Politeness, Sentiment. As we specify below, for each affect metric class, we use a tailor-made tool to extract it from the text of the comments.

For the detection of emotions, we use the tool developed by Ortu et al. [84] and extended by Murgia et al. [83]. This tool is particularly suited for our analysis as the algorithm has been trained on developers' comments extracted from Apache, a Jira-based data repository, hence within the Software Engineering domain. The classifier is able to detect love, anger, joy and sadness with an F_1 score[8] close to 0.8 for all of them.

Valence, Arousal and Dominance (VAD) represent conceptualised affective dimensions that respectively describe the interest, alertness and control a subject feels in response to a certain stimulus. In the context of software development, VAD measures may give an indication of the involvement of a developer in a project as well as their confidence and responsiveness in completing tasks. Warriner et al.'s [115] has created a reference lexicon containing 14,000 English words with VAD scores for Valence, Arousal, and Dominance, that can be used to train the classifier, similarly to the approach by Mantyla et al. [75]. In [75], the authors extracted valence-arousal-dominance (VAD) metrics from 700,000 Jira issue reports containing over 2,000,000 comments. They showed that issue reports of different type (e.g., feature request vs bug) had a fair variation in terms of valence, while an increase in issue priority would typically increase arousal.

For politeness detection, we use the tool proposed by Danescu et al. [35], which output a binary classification of the text as *polite* or *impolite*. This tool is particularly suitable in the context of our analysis as the algorithm has been trained using over 10,000 manually labelled requests from Wikipedia and StackOverflow. Indeed, in both data sources–but more specifically StackOverflow–contributors make use of technical terms and jargon, similarly to conversations among developers in online forum or development platforms.

Finally, the sentiment is measured using Senti4SD tool [20]. The algorithm extracts the degree of positive (ranging from 1 to 5), neutral (0) and negative (ranging from -1 to -5) sentiment in short texts. This tool is also trained on developers' comments.

[8]The F_1 score tests the accuracy of a classifier and it is calculated as the harmonic mean of precision and recall.

3.4.3 Affect Time Series

Once numerical values of the affect metrics are computed for all comments (as shown in the example in Table 3.23), we consider the timestamps (i.e. dates when the comments were posted) to build the corresponding affect time series. The affect time series are constructed by aggregating sentiment and emotions of multiple comments published on the same day. For a given affect metric, e.g. anger, for a specific day, we construct the time series by averaging the values of the affect metric over all comments posted on the same day.

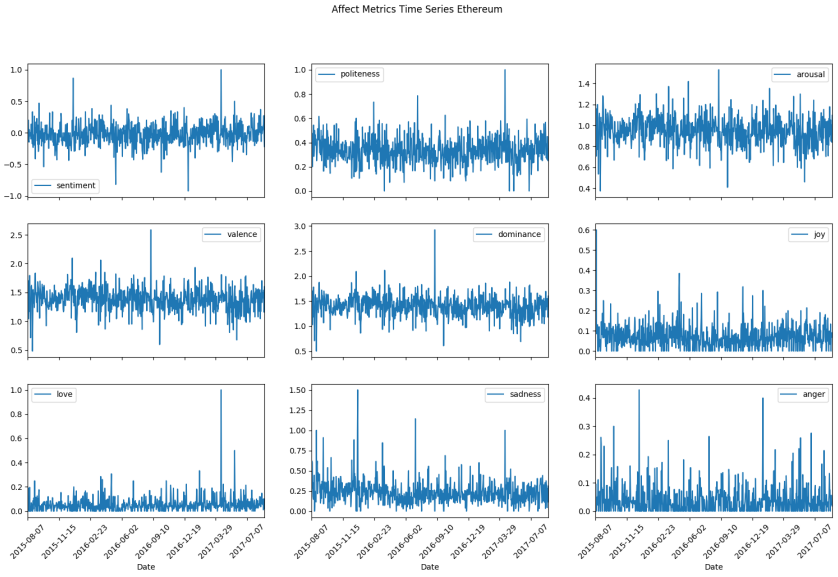


Figure 3.13: **Bitcoin affect time series.** Affect time series reconstructed from the comments of Bitcoin developers.

In Figure 3.13 and 3.14 we show the time series for all affect metrics for Bitcoin and Ethereum, respectively. We also report in Table 3.24 and 3.25 in more details the summary statistics of the affect time series for both cryptocurrencies respectively.

In Figure 3.15, we also show the boxplots of the data distributions for each affect time series for the Bitcoin and Ethereum case.

The box width gives an indication of the sample's variability. In the Bitcoin case, all the affect metrics show a small variance, particularly if we consider *anger*, *joy* and *love* time series. Moreover, all distributions are symmetric, except that for the *anger*, *joy*, *sadness* and *love* samples. For Ethereum, instead, the time series of sentiment, arousal, valence and dominance present a broader distribution compared to the corresponding Bitcoin ones. Further analyses of

Figure 3.14: **Ethereum affect time series.** Affect time series reconstructed from the comments of Ethereum developers.

Statistic	sentiment	arousal	valence	dominance	anger	sadness	joy	love
mean	-0.043081	0.984999	1.431276	1.442303	0.040907	0.244030	0.077308	0.049961
std	0.807408	0.541344	0.821061	0.805969	0.227526	0.595813	0.291157	0.227564
min	-4.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.750000	1.080000	1.100000	0.000000	0.000000	0.000000	0.000000
50%	0.000000	1.040000	1.470000	1.490000	0.000000	0.000000	0.000000	0.000000
75%	0.000000	1.270000	1.840000	1.860000	0.000000	0.000000	0.000000	0.000000
max	4.000000	6.150000	7.890000	7.220000	12.000000	38.000000	12.000000	4.000000

Table 3.24: **Summary statistics of affect metrics for Bitcoin.** Mean, standard deviation, min-max values considering all Github comments for sentiment, arousal, valence, dominance, anger, joy, love.

Statistic	sentiment	arousal	valence	dominance	anger	sadness	joy	love
mean	0.039835	1.271606	1.831578	1.829091	0.035780	0.224817	0.056057	0.132588
std	0.794382	0.837046	1.264279	1.188470	0.217238	0.547257	0.273674	0.365146
min	-3.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.900000	1.290000	1.300000	0.000000	0.000000	0.000000	0.000000
50%	0.000000	1.150000	1.640000	1.660000	0.000000	0.000000	0.000000	0.000000
75%	0.000000	1.420000	2.050000	2.060000	0.000000	0.000000	0.000000	0.000000
max	4.000000	5.570000	8.210000	7.000000	6.000000	15.000000	14.000000	3.000000

Table 3.25: **Summary statistics of affect metrics for Ethereum.** Mean, standard deviation, min-max values considering all Github comments for sentiment, arousal, valence, dominance, anger, joy, love.

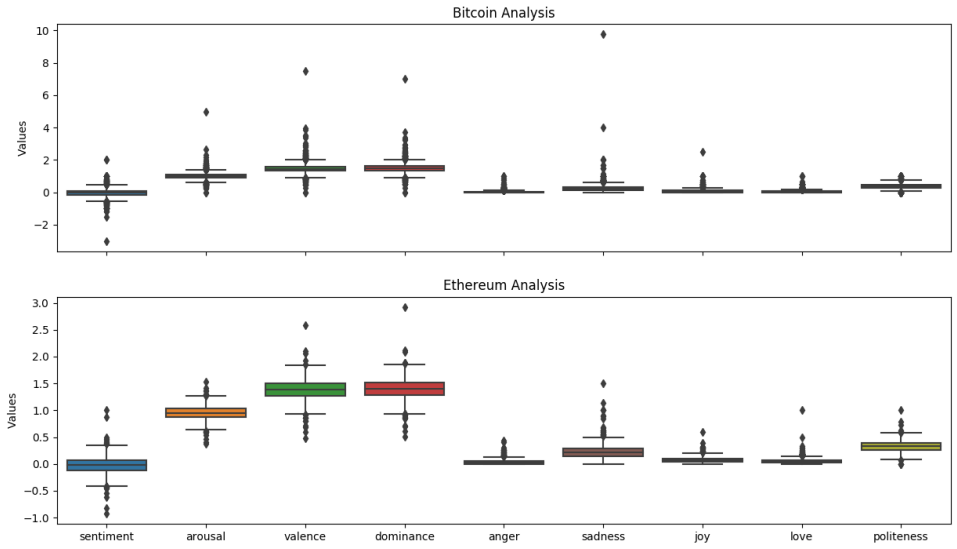


Figure 3.15: **Distributions of the affect time series.** Boxplot of the Bitcoin (top panel) and Ethereum (bottom panel) distributions for all affect metrics and all Github comments.

the stationarity of the time series can be found in the Supplementary Material (Sec. 1).

3.4.4 Granger Causality test

The Granger causality test is a statistical hypothesis test useful to assess whether a given time series shows some potential predictability power on another time series. In the *Granger-sense*, a time series X *Granger-causes* Y if X is able to improve the prediction of Y with respect to a forecast, considering only past values of Y [46]. Equivalently, if we define Γ'_τ as the information set of the form $(x_t, ..., x_{t-\tau}, y_{t-1}, ..., y_{t-\tau})$ (where τ is the number of lags or observations included in the regression), then x_t *Granger-causes* y_t if the variance of the optimal linear predictor of y_t based on Γ'_τ is smaller than the variance of the optimal linear predictor of y_t based on the information set $\Gamma_\tau = (y_{t-1}, ..., y_{t-\tau})$ of lagged values of y_t only:

$$\sigma^2(y_t|y_{t-\tau}, x_{t-\tau}) < \sigma^2(y_t|y_{t-\tau}), \forall \tau \in \mathbf{N}\,. \tag{3.12}$$

The procedure of the *Granger-causality* test is as follows.

1. The time series of cryptocurrency returns (e.g. BTC returns) (y_t) is regressed on its past values excluding the metric series in the regressors.

The so-called *restricted* regression can be written as

$$y_t = \alpha + \sum_{i=1}^{\tau} \rho_i y_{t-i} + \xi_t \,, \tag{3.13}$$

where α is a constant and the error term ξ_t is an uncorrelated white-noise process. We, then, calculate the restricted sum of squared residuals (SSR_r)

$$SSR_r(\tau) = \sum_{i=1}^{N} \left[y_i - \hat{y}_i(\Gamma_\tau) \right]^2 , \tag{3.14}$$

where N is the number of observations, τ is the number of lags included in the regression, Γ_τ is the information set, and $\hat{y}_i$ are the predicted values.

2. We compute a second regression including the lagged values of the affect time series in the regressors. This *unrestricted* regression reads

$$y_t = \alpha + \sum_{i=1}^{\tau} \rho_i y_{t-i} + \sum_{i=1}^{\tau} \gamma_i x_{t-i} + \xi'_t \,. \tag{3.15}$$

As before, we evaluate the unrestricted sum of squared residuals (SSR_u) as follows

$$SSR_u(\tau) = \sum_{i=1}^{N} \left[y_i - \hat{y}_i(\Gamma'_\tau) \right]^2 . \tag{3.16}$$

3. Finally, if $SSR_u < SSR_r$ the affect time series considered for the analysis *Granger-causes* the cryptocurrency returns series.

To determine the presence of (direct and reverse) Granger causality between affect and return time series we use a two-step approach: (i) we first tested the null-hypothesis rejecting it if the p-values are below the chosen significance level and then (ii) we restricted the set of time series to the ones minimising also the information loss (using the Akaike criterion specified below). Both approaches are standard tools used for the optimal lag selection in the econometric literature [8, 73, 110].

For this analysis we have implemented the *grangercausalitytest* test using the *statsmodels*[9] Python library. This tool tests for *Granger non-causality* of two time series, i.e. the null hypothesis H_0 is that the chosen affect metric series does not *Granger-cause* the Bitcoin or Ethereum returns series. We reject the null hypothesis if the p-values are below a desired size of the test, choosing a 5% significance level. The p-values are computed using the Wald test as per the

[9] `https://www.statsmodels.org/stable/index.html`.

standard Python Statsmodel libraries [42]. The number of lags included in the regression models can be tuned by the τ parameter. For any fixed value of τ, a *Granger causality* test is computed for all lags up to τ.

The two possible outcomes of the Granger test are:

- The observed p-values are less than the 5% significance level: rejection of the null hypothesis H_0. The affect time series *Granger cause* the cryptocurrency returns one.

- The observed p-values are greater than the 5% significance level: H_0 cannot be rejected. The affect time series does not *Granger cause* the cryptocurrency returns one.

The *AIC* metric was also monitored for the two models (restricted and unrestricted) for each lag value to check for consistency with the results obtained with the *Granger causality* test. The *Akaike Information Criterion* (AIC) is a statistical tool, based on information theory, that can be used for model selection.

The *AIC* metric provides an estimate of the quality of a given model, based on the loss of information: the best model minimises the information loss. *AIC* for least squares model fitting can be mathematically defined as

$$AIC = 2(k+1) + n\log(SSR)\,, \tag{3.17}$$

where n is the sample size and k is the number of parameters [10]. If the AIC is not minimal for the model with the smallest p-value, we cannot validate the test and we conclude that the Granger causality test has highlighted a spurious, non-statistically significant correlation. Therefore, we restrict the set of affect time series effectively showing Granger causality with returns, to the ones for which not only the p-value is below the chosen significance level but also the AIC is minimal.

We perfom the Granger test on the stationary affect time series selected via the analysis available in Sec. 1 of the Supplementary Material. According to our analysis, the stationary affect time series that we will consider for the Bitcoin case are *sentiment*, *sadness*, *arousal*, *valence*, *love* and *dominance*. For the Ethereum case we will use *sentiment*, *anger*, *arousal*, *valence*, *love*, *dominance*, *joy* and *politeness*. It is worth noting that the Granger causality test is sensitive to the number of lags input in the model. For this reason, we have analysed a large range of lags, of the order of five months. More specifically, the τ parameter was set to 150.

3.4.5 Long Short-Term Memories and predictions

For our prediction task of the cryptocurrency prices, we use a Recurrent Neural Network (RNN). A RNN, at its most fundamental level, is simply a type of densely connected neural network. However, the key difference with respect to normal feed-forward networks is the introduction of time, with the output of the hidden layer in a recurrent neural network being fed back into itself. RNNs are often used in stock-market predictions [36, 79, 103] and more recently also for Bitcoin and cryptocurrency prices [25, 79].

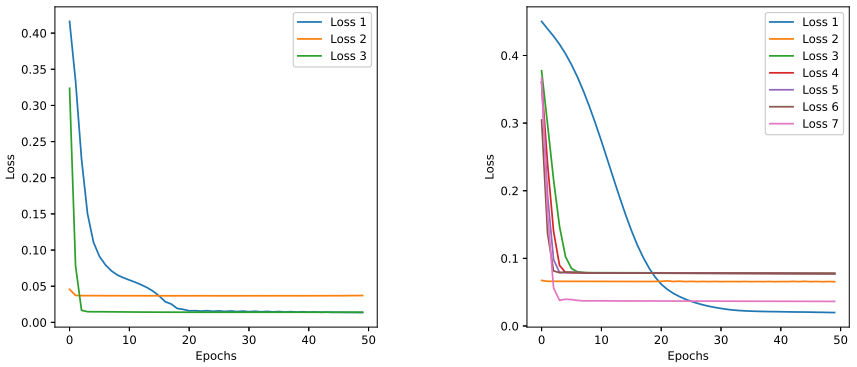
In this analysis, we use a Long-Short Term Memory (LSTM) RNN to predict Bitcoin and Ethereum returns. In our model, we use the previous day returns and affect metrics for the prediction of the returns of the current day (1-day forecast horizon). We decided to use this short forecast horizon model – which is normally a benchmark of more sophisticated prediction algorithms – as we are mostly concerned about demonstrating a possible improvement in Root Mean Square Error (RMSE)[10] when inputting in the model the affect time series rather than building a sophisticated prediction model.

The affect time series used for this analysis are the ones that showed Granger-causality with the Bitcoin and Ethereum returns time series. Indeed, the test assessed whether a given affect time series had some potential predictive power over the cryptocurrency returns time series. As reported in Sec. 3.4.7, we selected *sentiment* and *sadness* for Bitcoin and *sentiment*, *anger*, *arousal*, *dominance*, *valence* and *love* for Ethereum.

We designed the LSTM with 50 neurons in the first hidden layer and 1 neuron in the output to predict the cryptocurrency returns. To configure the LSTM, we use a sigmoid activation function, we calculate the Mean Absolute Error (MAE) loss function and we use the efficient Adam version of stochastic gradient descent [66] for the optimal choice of models' parameters. We train the LSTM, first, using only data related to the cryptocurrency (Ethereum or Bitcoin) returns time series and, then, we incrementally add the correlated (via Granger-causality) affect metrics features.

We first apply the LSTM using only the cryptocurrency returns (Bitcoin or Ethereum returns time series) as a feature, i.e. solving in this case a univariate regression problem. Then, we incrementally add the affect metrics, i.e. considering a multivariate regression problem, to analyse potential effects on the RMSE associated with the predictions. Our analysis is performed by training the LSTM for 50 epochs and recording for each epoch the corresponding RMSE value. Figures 3.16a and 3.16b show the loss of the RNN models against the

[10] The RMSE is defined as the standard deviation of the residuals or prediction errors, i.e. $\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$, where n is the number of observation, $y_i, i = 1,\ldots,n$ are the observed values and $\hat{y}_i, i = 1,\ldots,n$, the predictions.



(a) Bitcoin (b) Ethereum

Figure 3.16: **RNN Loss against number of epochs.** Panel (a) RNN trained with Bitcoin only data (1) first and then sequentially adding sentiment (2) and sadness (3).
Panel (b) RNN trained with Ethereum returns data only (1), then adding sequentially sentiment (2), anger (3), arousal (4), valence (5), dominance (6), love (7).

epochs. We can see that after 50 epochs the loss converges to a stationary value for all models. Finally, all models where trained using 70% of data for training and 30% for testing.

3.4.6 Results

In this section, we summarise the results of our analysis concerning testing for (i) causality between affect time series and cryptocurrency returns and (ii) improvement in Root Mean Square Error (RMSE) for the prediction of returns when including affect time series.

3.4.7 Does the affect of Bitcoin and Ethereum communities influence variations in returns?

In this section, we focus on understanding if there exists a causal relationship between affect time series and the time series of Bitcoin/Ethereum returns. The analysis is performed using the *Granger causality* test [46], which informs on whether changes in a time series – in our case the returns time series – are induced or connected to a variation in a second correlated time series – in our case the affect time series. Details on the Granger test can be found in Sec.

3.4.4. As we will show in the following, the Granger test is detecting significant Granger-causality (both direct and reverse causality) between affect time series and cryptocurrency returns. Via this analysis, we are also able to give an estimate of the time lag or delay after which effects of variations in the affect time series are “visible" in the cryptocurrency returns time series.

Granger causality test - Bitcoin

Let us start with the Bitcoin returns time series analysis. The Granger test highlights that only *sentiment* and *sadness* metrics *Granger-cause* the Bitcoin series. According to the test, instead, there is no casual relationship between the Bitcoin returns and *arousal*, *valence*, *love* and *dominance* time series, for any considered lag value. In order to select the time lag for the Granger causality, we monitor the p-values as a function of the time lag and select – among the time lags with p-values falling below the significance level – the time lag associated with the minimal p-value.

As an illustration, we show in Fig. 3.17 the p-values obtained for each lag value, up to the chosen τ for the two affect time series that displayed statistically significant Granger causality with the Bitcoin returns time series.

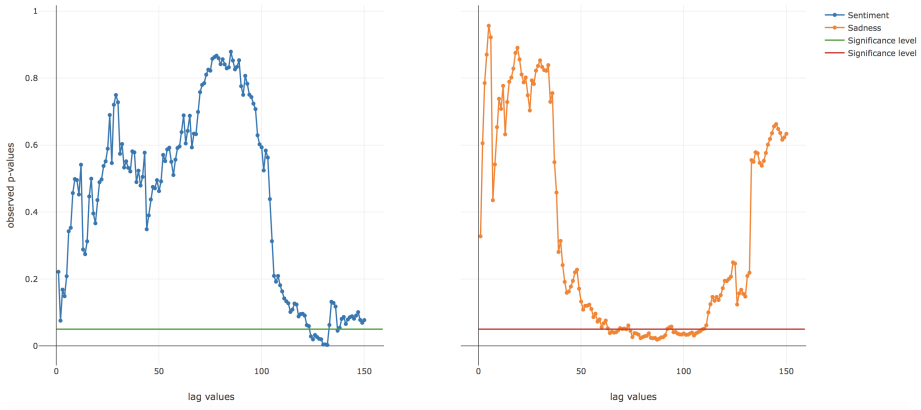


Figure 3.17: **Bitcoin direct causality - p-values as a function of lags.**
Left: Bitcoin returns - sadness time series direct causality. Right: Bitcoin returns - sadness time series direct causality.

A *reverse Granger-causality test* was also conducted, in order to test whether cryptocurrency prices influence the affect time series, hence developers' behaviour and feelings. Specifically, we obtain that Bitcoin returns Granger-cause only *sentiment* and *sadness* affect metrics. In this case we therefore deal with a *bidirectional causality*, whereby *sadness* and *sentiment* series increase the pre-

diction of the Bitcoin price returns and vice versa. The complete table containing p-values and associated time lags for all affect metrics for direct and reverse causality can be found in Sec. 1 of the Supplementary Material.

We have also checked of the *AIC* values of the models to ascertain that the Granger test was not capturing spurious effects.

AIC values as a function of the time lags can be found for all affect metrics Granger-causing the Bitcoin returns in Sec. 2 of the Supplementary Material. We have, then, selected only the affect time series showing a significant causal relationship, and also minimising the information loss.

The final set of time series showing a robust Granger (direct and/or reverse) causality with Bitcoin returns according to both the p-value and the AIC tests, including the minimum observed p-values and their corresponding time lag, is reported in Table 3.26.

Table 3.26: **Bitcoin Granger Causality tests.** Minimum observed p-values and corresponding lag values for sadness and sentiment times series.

	Granger Causality Test		*Reverse Granger Causality Test*	
Series	**p-value**	**lag value**	**p-value**	**lag value**
sadness	0.019	87	9.504e-8	41
sentiment	0.003	132	0.040	137

Granger causality test - Ethereum

We repeat the analysis for the Ethereum returns time series. In this case, we find significant (direct) Granger-causality between the *anger*, *sentiment*, *valence*, *arousal*, *love* and *dominance* metrics series and the Ethereum returns time series. Instead, we can conclude that *joy* and *politeness* metrics do not *Granger cause* the Ethereum returns series.

As an illustration of the process for lag selection, we show in Fig. 3.18 the p-values obtained for each lag value, up to the chosen τ, for the affect time series that are correlated with the Ethereum returns (direct causality).

The *reverse Granger-causality test* results highlight, instead, that *joy*, *valence*, *arousal*, *love* and *dominance* affect metric influences the returns of Ethereum.

As for the Bitcoin analysis, we select as time lag for the Granger causality, the value associated with the minimal significant p-value. The complete table containing p-values and associated time lags for all affect metrics for direct and reverse causality can be found in Sec. 1 of the Supplementary Material.

We, then, further restrict the set of time series showing Granger-causality, by checking using the AIC criterion that the AIC associated with the models with

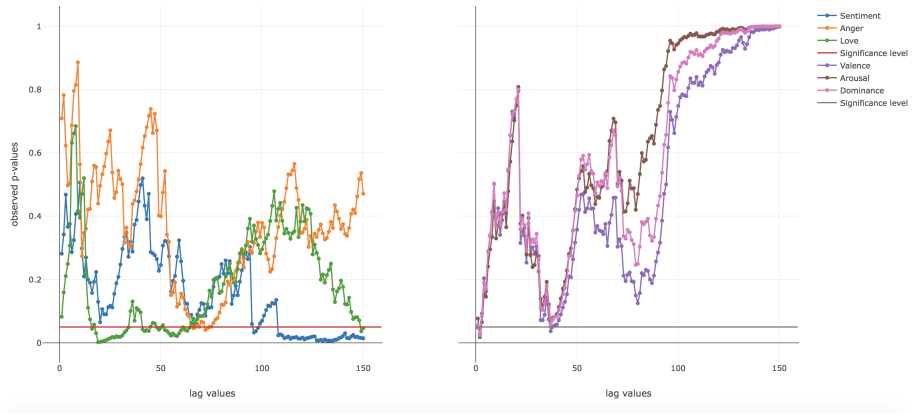


Figure 3.18: **Ethereum direct causality - p-values as a function of lags.** Left: p-values for the direct causality tests between sentiment, anger, love affect time series and returns. Right: p-values for the direct causality tests between valence, arousal and dominance affect time series and returns.

the selected time lag is also minimal, therefore removing cases with possibly spurious correlations.

As an example, we show here the analysis for the *love* metric. In Figure 3.19 we report the AIC values as a function of the lag parameter. We notice that the lowest values of AIC (corresponding to minimal information loss) are recorded in correspondence with the time lag values (~ 19) associated with the lowest p-value. This analysis is, therefore, consistent with the Granger test results. Similar conclusions can be drawn for other affect time series "Granger-causing" the Ethereum returns. AIC values as a function of the time lags can be found for all affect metrics Granger-causing the Ethereum returns in Sec. 2 of the Supplementary Material.

The final set of time series showing a robust Granger (direct and/or reverse) causality with Ethereum returns according to both the p-value and the AIC tests is reported in Table 3.27. As before, we summarise the results of the Granger test, including time lags and the associated p-values.

To summarise, in this case, we have a *unidirectional Granger-causality* from *anger*, *sentiment*, *valence*, *arousal*, *love* and *dominance* series to Ethereum returns and a reverse *unidirectional causality* from Ethereum returns to *joy* metric series.

General remarks for the Bitcoin and Ethereum analysis

In general, for both cryptocurrencies, the observed p-values are well below the chosen 5% significance level. In particular, the p-values obtained for the *sen-*

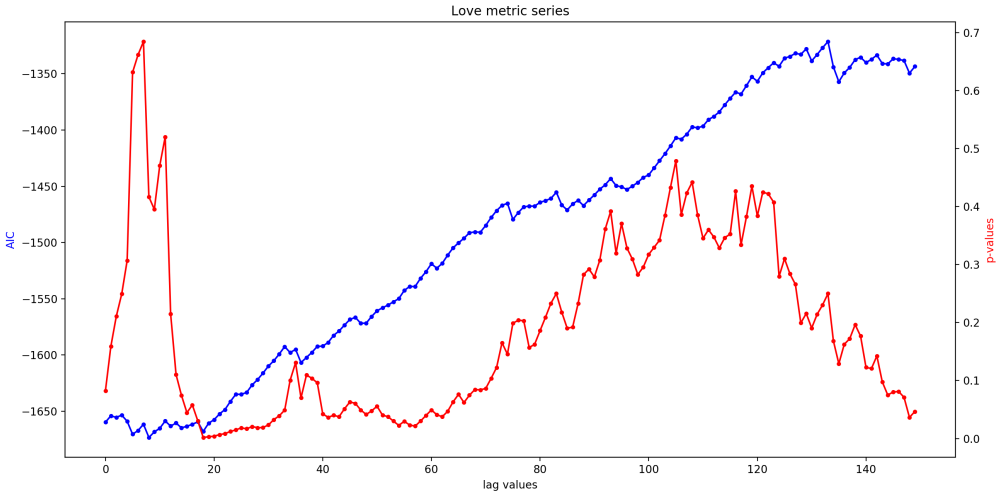


Figure 3.19: **Ethereum - Love AIC and p-values.** AIC values (see Eq. (3.17)) (blue line) and p-values (red lines) as a function of the number of lags.

Table 3.27: **Ethereum Granger Causality tests.** Minimum observed p-values and corresponding lag values for different affect time series.

	Granger Causality Test		*Reverse Granger Causality Test*	
Series	**p-value**	**lag value**	**p-value**	**lag value**
joy	-	-	0.011	1
love	0.002	19	-	-
anger	0.041	72	-	-
valence	0.028	2	-	-
arousal	0.018	2	-	-
sentiment	0.006	134	-	-
dominance	0.024	2	-	-

timent metric is even below the 1% significance level, in both the Bitcoin and Ethereum analysis.

In terms of time lag, the test highlights that the Bitcoin returns time series seems to be affected by sadness metrics and developers sentiment only after a period of the order of 3−5 months (see Tables 3.26, 3.27). Similar considerations can be made in the case of Ethereum for the *Anger* and *Sentiment* affect time series. *Dominance*, *Arousal*, *Valence* and *Love* metrics series appears, instead, to have short-term effects on the Ethereum returns time series.

We could speculate that the short-term and long-term nature of the effects of affect metrics on returns is related to the nature of cryptocurrencies itself. For instance, on the Ethereum platforms, developers can issue multiple tokens with different features and often the developers themselves are those advertising the tokens and making transactions to increase their values. As highlighted in a recent research[11] a dominant fraction of the transactions on the Ethereum blockchain appears to be handled by token teams giving new tokens for free (airdrops) to Ethereum users, therefore possibly impacting the total valuation of the platform.

In the Bitcoin case, the long-term effect of changes in developers' affect metrics may be correlated with the market efficiency. Indeed, in [71, 111] the authors show that the Bitcoin market is not efficient, i.e. that all information is not instantly incorporated into prices, hence the large time lag of the causality.

Finally, disagreements among developers of a platform may signal and lead to a fork event, which in turn generates price movements as shown in [22]. From the onset of a disagreement within the community to the actual fork attempt there is generally a significant time lag, possibly of weeks or months, compatible with our results.

Regarding the reverse causality, in the Bitcoin case we notice rather high lag values (as for the direct causality, i.e. affect metrics → Bitcoin returns), hence the Bitcoin community does not react immediately to price news. In Ethereum, instead, price movements impact the community with a time lag of 1-day. We could speculate that this effect is once again related to the different uses of the two blockchain platforms (e.g. multiple tokens issued on the Ethereum blockchain). In a related study on topic analysis of tech forums on Reddit [100], authors also find that topics on "fundamental cryptocurrency value" are very frequent in the Ethereum community threads and are correlated with increase in prices.

[11] `https://research.tokenanalyst.io/classifying-ethereum-users-using-blockchain-data/`

3.4.8 Is the affect of Bitcoin and Ethereum communities able to improve the error on the prediction of returns?

As we discussed in the previous analysis, the decisions taken by the community of developers may have a non-negligible impact on the crypto-market. In this section, we further investigate the predictive power of the affect time series over the cryptocurrency returns. In particular, we use a deep learning algorithm to predict the cryptocurrency returns in two scenarios, (i) using only the cryptocurrency returns as a feature or (ii) incrementally adding the affect metrics to determine whether the additional affect metrics features yield an improvement in the prediction of the Root Mean Square Error (RMSE). By prediction of the RMSE we mean the average squared error of the correct estimation of the daily returns compared with the actual returns. The details of the algorithm we used were described previously in Sec. 3.4.5.

The results obtained for the RMSE of the predictions (measured at the end of the test phase, i.e. after 50 training epochs) are summarised in Table 3.28 and 3.29 for Bitcoin and Ethereum respectively. We compute the RMSE value by varying the number of features used in the algorithm. We consider as features the affect time series that showed direct Granger causality with the Bitcoin returns (see Table 3.26). For the Bitcoin analysis (Table 3.28), the 1-feature case corresponds to including only the time series of Bitcoin returns, while the 3-feature case includes the return time series together with the sadness and sentiment time series. We proceed in a similar way for the Ethereum case, where we incrementally include affect time series to the prediction model for the returns (considering the affect metrics that showed causality with the returns, summarised in Table 3.27).

Features	RMSE
1	0.129
2	0.032
3	0.013

Table 3.28: **Bitcoin prediction errors.** Root Mean Square Error (RMSE) of predictions considering (1) only Bitcoin returns and then sequentially adding sentiment(2) and sadness (3) time series as features.

Interestingly, we find that including the affect time series in models (based on LSTM neural networks) for the prediction of cryptocurrency returns yield a decrease in the RMSE. This result holds true for the prediction of the time series of both the Bitcoin and Ethereum returns. Indeed, in both Table 3.28, 3.29, we

Features	RMSE
1	0.178
2	0.082
3	0.113
4	0.114
5	0.114
6	0.114
7	0.048

Table 3.29: **Ethereum prediction errors.** Root Mean Square Error (RMSE) of predictions considering Ethereum returns (1), then adding sequentially sentiment (2), anger (3), arousal (4), valence (5), dominance (6), love (7) time series.

can see that when adding all the affects metrics, the RMSE of the predictions is significantly improved, from 0.129 to 0.013 (90% of improvement) for Bitcoin and from 0.178 to 0.048 (73% of improvement) for Ethereum.

We compared the distributions of the RMSEs for the 1-feature model (including only cryptocurrency returns) and the final model with all the features. The distributions of RMSEs include the RMSE values for each one of the 50 training epochs for the two models (including 1-feature only or all affect metrics respectively). For this comparison we used the Wilcoxon Rank-Sum test, a nonparametric test that does not assume specific characteristics of the distributions, e.g. normality, compared to equivalent tests (e.g. the Welch test) [43]. We find that the two distributions are statistically different with a $p-value$ of 0.0002 for Bitcoin (effect size of 0.56) and 0.00001 for Ethereum (effect size 0.48).

To summarise, we show that (i) by aggregating all the features (i.e. all affect metrics) we obtain – for both Bitcoin and Ethereum – a significant increase in predictive power than when considering them separately. Moreover, (ii) we provide examples of cases where also the partial aggregation (using only some of the affect metrics that Granger-cause the returns, e.g. considering Ethereum returns and the anger time series) is better than inputting only the time series of returns for the prediction task. These examples are non-exhaustive of all possible combinations of affect time series and returns as input of the neural network, but serve as illustrations that a decrease in prediction error can be induced by the addition of the affect metric.

3.4.9 Threats To Validity

Threats to external validity concern the generalisation of our results. In this study, we analysed comments from GitHub for Bitcoin and Ethereum open source projects. Our results cannot be representative of all other cryptocurrencies and this could, indeed, affect the generality of the study. Replication of this work on other open source cryptocurrency-related projects is needed to confirm our findings. Additionally, the politeness tool can be subject to bias due to the domain used to train the machine learning classifier.

Threats to internal validity concern confounding factors that can influence the obtained results. Based on empirical evidence, we assume a relationship between the emotional state of developers and what they write in issue reports [94]. Since the main goal of developers' communication is the sharing of information, the consequence of removing or camouflaging emotions may make comments less meaningful and cause misunderstandings. This work is focused on sentences written by developers for developers. To illustrate the influence of these comments, it is important to understand the language used by developers. We believe that all the tools used for measuring the affect metrics are valid in the software development domain. The comments used in this study were collected over an extended period from developers unaware of being monitored, therefore, we are confident that the emotions, sentiment, politeness and VAD metrics we analysed are genuine ones.

Threats to construct validity focus on how accurately the observations describe the phenomena of interest. The detection of emotions from issue reports presents difficulties due to vagueness and subjectivity. Emotions, sentiment and politeness measures are approximated and cannot perfectly identify the precise context, given the challenges of natural language and subtle phenomena like sarcasm.

Chapter 4

Related Works

Over the years many algorithms have been developed for forecasting time series in stock markets. The most widely adopted are based on the analysis of past market movements [1]. Among the others, [4] proposed a prediction system using a combination of genetic and neural approaches, having as inputs technical analysis factors that are combined with daily prices. [41] discussed a hybrid prediction model that combines differential evolution-based fuzzy clustering with a fuzzy inference neural network for performing an index level forecast. [62] presented a forecasting model based on chaotic mapping, firefly algorithm, and support vector regression (SVR) to predict stock market prices. Unlike other widely studied time series, still very few researches have focused on bitcoin price prediction. In a recent exploration [80] tried to ascertain with what accuracy the direction of Bitcoin price in USD can be predicted using machine learning algorithms like LSTM (Long short-term memory) and RNN (Recurrent Neural Network). [86] tried to forecast the volatility of the Bitcoin/USD exchange rate using GARCH (Generalized AutoRegressive Conditional Heteroscedasticity) models. [109] studied and applied α-Sutte indicator and Arima (Autoregressive Integrated Moving Average) methods to forecast historical data of Bitcoin. [107] proposed the use of Fast Wavelet Transform to forecast Bitcoin prices. [117] examined a few complexity measures of the Bitcoin transaction flow networks, and modeled the joint dynamic relationship between these complexity measures and Bitcoin market variables such as return and volatility. [7] presented a forecasting Bitcoin exchange rate model in high volatility environment, using autoregressive integrated moving average (ARIMA) algorithms. [21] studied the predictability of cryptocurrencies time series, comparing several alternative univariate and multivariate models in point and density forecasting of four of the most capitalized series: Bitcoin, Litecoin, Ripple and Ethereum, using univariate Dynamic Linear Models and several multivariate Vector Autoregressive models with different forms of time variation. [113] used knowledge

of statistics for financial time series and machine learning to fit the parametric distribution and model and forecast the volatility of Bitcoin returns, and analyze its correlation to other financial market indicators . Other approaches try to predict stock market index using fusion of machine learning techniques [95]. [2] introduced a novel concept of chainlets, or bitcoin subgraphs, to evaluate the local topological structure of the Bitcoin graph over time and the role of chainlets on bitcoin price formation and dynamics. [50] predicted the future price of bitcoin investigating the predictive power of blockchain network-based, in particular using the bitcoin transaction graph.

Quantitative investigations aimed at extracting information from cryptocurrencies prices time series and predicting prices drivers [11, 99] or the next most likely jump, range from theoretical models of pricing and adoption of digital tokens [12, 28, 32] to machine learning [3, 59] and neural network-driven [70] forecasts of prices and returns.

Several are also the studies that tried to use online information, including social media topics discussions, to predict cryptocurrencies price changes, for example, Google searches for Bitcoin-related terms have been shown to have a relationship with the Bitcoin price [67]. For this reason, new types of data have been recently used to improve models and predictions. “Social" sentiment is extracted using data gathered from users’ online communities [65] – e.g. online forums such as BitcoinTalk[1] – and from online news and tweets [5, 63, 72]. For instance, a suitably built sentiment index can be used to test for speculative bubbles in cryptocurrency prices [23]. More broadly, Google search data related to cryptocurrencies can be relevant to characterise the set of Bitcoin users [119]. Temporal topic analysis of Bitcoin and Ethereum discussions on Reddit also show correlations with variations of cryptocurrency prices [100]. D. Garcia and F. Schweitzer have considered the strength and polarisation of opinions displayed in Twitter submissions, founding that an increase in the polarisation of sentiment (disagreement of sentiment) preceded a rise in the price of Bitcoin [45]. In another work, several machine learning pipelines were implemented with the objective of identifying cryptocurrency market movement, in order to prove whether Twitter data relating to cryptocurrencies can be utilized to develop advantageous crypto coin trading strategies [108]. F. Valencia et al. proposed to use most common machine learning tools, such as neural networks, support vector machines and random forest, and available social media data for predicting the price movement of the Bitcoin, Ethereum, Ripple and Litecoin cryptocurrency market movements, showing that it is possible to predict cryptocurrency markets using machine learning and sentiment analysis and that, in this case study, neural networks outperform the other models [44].

[1] `https://bitcointalk.org`

A recent exploration monitored the activity on the social media platform Reddit in order to detect the epidemic-like spread of investment ideas beneficial in the prediction of cryptocurrency price bubbles [101]. Developers in open source blockchain and DLTs projects are also crucial entities, responsible for the maintenance and updates of the platforms. The idea that the human aspects of software development are of paramount importance to ensure high team productivity, software quality and developers satisfaction is already well-established in software engineering [39, 49, 82]. These studies have shed light on the importance of all the social and human aspects associated with the software development processes, and empirically demonstrated how a positive environment may have an impact on team productivity, software quality and developers' satisfaction [47, 64]. Moreover, standard metrics extracted from open source development platforms such as Github[2] and Bitbucket[3] can be used to rank the top crypto tokens [89]: metrics include number of commits and issues, forks and number of contributors to the code.

[2] https://github.com
[3] https://bitbucket.org

Chapter 5

Conclusion

These three years of PhD research aimed to study the well-known time series prediction problem in the field of financial markets, specifically cryptocurrency market, in order to develop an innovative approach to enrich the state-of-the-art. The main goal of the proposed approach is to ascertain if the progressive addition of variables of different sources can effectively improve the prediction of cryptocurrency prices. The variables included in this work can be grouped into three major categories. The first one contains the trading variables, such as *close* and *open* prices, volumes and so on and so forth. The second class of variables consist of technical variables, i.e. technical analysis variables calculated from the trading ones. Finally the social variables generated from online information and social media comments, such as users' sentiment, politeness and emotions. The analysis were conducted using traditional econometrics methods and state-of-the-art Artificial Intelligence algorithms.

Forecasting Bitcoin closing price series using linear regression and neural networks models.
This part of the Ph.D. work can be considered as a first approach to cryptocurrency price prediction problem with the purpose of forecasting daily closing price series of Bitcoin, Litecoin and Ethereum cryptocurrencies, using data on prices and volumes of prior days. In this work both statistical techniques and artificial intelligence algorithms were implemented. The considered financial time series resulted to be indistinguishable from a random walk, so the series were partitioned into shorter overlapping sequences in order to identify different stationary price regimes. The results confirmed the correctness of our hypothesis of identifying time regimes that do not resemble a random walk and that are easier to model, finding that best result are obtained using more than one previous prices. It is important to emphasize that the identification of short-time regimes within the entire series, allowed us to obtain leading-edge

results in the field of financial series forecasting. This study is a starting point to the development of our innovative proposed approach since the analysed data only includes trading variables, specifically close, open, high, low prices and volumes.

Blockchain Cryptocurrencies' Price Movements Through Deep Learning. Technical Analysis is a trading discipline employed to gain speculative insights and trading opportunities in price trends, always used as a support in predicting traditional stock market price series. These considerations led us to verify whether this trading technique could also affect cryptocurrency prices. For this reason we decided to assess whether the addition of technical indicators to the classic macroeconomic variables can lead to an effective improvement in the prediction of Bitcoin price changes. This purpose was achieved implementing machine learning techniques, such as *Support Vector Machine* and *XGBoost* models and deep learning algorithms, namely *CNN* and *LSTM* neural networks. In the restricted analysis the model that achieved the best performance, both in terms of the chosen prediction error and execution speed, is *SVM* with an average accuracy of 54% and a standard deviation of 2.8%. In the unrestricted case, the best result is achieved by the *CNN* neural network with an average accuracy of 54.7% and a standard deviation of 2.5%. The most important thing that is worth noting is that, the results obtained for the unrestricted model highlights that in general the addition of technical analysis variables to the dataset leads to an effective improvement in the average accuracy. We can state that this finding is not the result of a statistical fluctuation since all the implemented models yielded the same achievements. For the same reason, we can furthermore exclude the dependence of the obtained results on the particular implemented algorithm. These promising findings led us to deeply investigate the discussed approach.

Does online information influence cryptocurrency prices? The strong interest aroused from cryptocurrencies not only in the scientific and financial fields but also within social media communities, led us to the conjecture that the inclusion of social media variables and online information to the datasets could further improve cryptocurrency price predictions. This conjecture was confirmed by this analysis which proved the existence of possible causal-effect relationships between web and social events and cryptocurrency price variations. This goal was achieved by implementing the well-known Hawkes model to indentify and and model relationships between cryptocurrencies market price changes and topic discussion occurrences on social media. The results obtained in this analysis highlighted that it is possible to identify some interactions among the considered features, and it appears that some topics are indicative of certain types of price movements. Specifically, the discussions concerning issues about government, trading and Ethereum

cryptocurrency as an exchange currency, appear to affect Bitcoin and Ethereum prices negatively. The discussions of investment appear to be indicative of price rises, while the discussions related to new decentralized realities and technological applications is indicative of price falls.

Impact of Development Practices on Cryptocurrency Prices.
The confirmation of the presence of correlation patterns between online information and cryptocurrency prices led us to conduct a more deeply investigation on the existence of possible relationships between network developers comments and cryptocurrencies. Blockchain development processes have deep foundations within the community, with the community itself being the "heart and brain" of all critical decisions around the improvements and changes on the platforms. Investors and crypto-market players look at the development activities and read the technical reports of the developers to try to predict the success of the platforms they are betting on. There is, indeed, a connection between the development activities and the valuation of cryptocurrencies. For this reason this study investigated whether developers' emotions can effectively provide insights that can improve the prediction of the price of tokens. Our analysis proved the existence of a Granger-causality between the time series of developers' emotions and Bitcoin/Ethereum price. Moreover, using an artificial recurrent neural network, we showed that the error (RMSE) associated with the prediction of the prices of cryptocurrencies significantly decreases when including the affect time series. These results highlights the existence of a connection between the development activities and the valuation of cryptocurrencies.

Several are the papers already in the literature that try to study and predict the behaviour of Bitcoin prices, or more generally of cryptocurrency prices. Nevertheless, the analysis of the cryptocurrency market is still one of the most discussed topics and at the same time difficult to interpret and analyse. The main focus of this work is the introduction of a novel approach that lead to an effective improvement in cryptocurrency price prediction through the progressive addition to the dataset of variables of different sources, particularly trading, technical and social variables. This findings allow the development of more accurate cryptocurrencies prediction models, serving as a source of information for speculators and investors.

Bibliography

[1] J. Agrawal, V. Chourasia, and A. Mittra. State-of-the-art in stock prediction techniques. International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering, 2(4):1360–1366, 2013.

[2] C. Akcora, A. K. Dey, Y. R. Gel, and M. Kantarcioglu. Forecasting bitcoin price with graph chainlets. Pacific-Asia Conference on Knowledge Discovery and Data Mining, 2018.

[3] L. Alessandretti, A. ElBahrawy, L. M. Aiello, and A. Baronchelli. Anticipating cryptocurrency prices using machine learning. Complexity, 2018.

[4] G. Armano, M. Marchesi, and A. Murru. A hybrid genetic-neural architecture for stock indexes forecasting. Information Sciences, 170(1):3–33, 2015.

[5] T. Aste. Cryptocurrency market structure: connecting emotions and economics. special issue of digital finance on cryptocurrencies. Digital Finance, 2018.

[6] E. Bacry, M. Bompaire, S. Gaiffas, and S. Poulsen. tick: a python library for statistical learning, with a particular emphasis on time-dependent modeling. ArXiv e-prints, 2017.

[7] N. Bakar and S. Rosbi. Autoregressive integrated moving average (arima) model for forecasting cryptocurrency exchange rate in high volatility environment: A new insight of bitcoin transaction. International Journal of Advanced Engineering Research and Science, 4:Issue 11, 2017.

[8] M. Balcilar, E. Bouri, R. Gupta, and D. Roubaud. Can volume predict bitcoin returns and volatility? a quantiles-based approach. Economic Modelling, 64:74–81, 2017.

[9] A. Banerjee, J. Dolado, J. Galbraith, and D. Hendry. Cointegration, error correction, and the econometric analysis of non-stationary data. Oxford: Oxford University Press, Chapter 4, 1993.

[10] H. T. Banks and M. L. Joyner. Aic under the framework of least squares estimation. Applied Mathematics Letters, 74:33–45, 2017.

[11] S. Bartolucci, G. Destefanis, M. Ortu, N. Uras, M. Marchesi, and R. Tonelli. The butterfly "affect": Impact of development practices on cryptocurrency prices. EPJ Data Science, 9(1):21, 2020.

[12] S. Bartolucci and A. Kirilenko. A model of the optimal selection of crypto assets. ArXiv preprint, (arXiv:1906.09632), 2019.

[13] C. M. Bishop. Pattern Recognition and Machine Learning. New York: Springer, 2006.

[14] D. Blei and J. Lafferty. Dynamic topic models. ICML, pages 113–2006.

[15] B. Boser, I. Guyon, and V. Vapnik. A training algorithm for optimal margin classifier. Proceedings of the Fifth Annual ACM Workshop on Computational Learning Theory, 5, 08 1996.

[16] E. Bouri, C. K. M. Lau, B. Lucey, and D. Roubaud. Trading volume and the predictability of return and volatility in the cryptocurrency market. Finance Research Letters, 29:340–346, 2019.

[17] G. E. P. Box and G. Jenkins. Time series analysis: Forecasting and control. Holden-Day, 1976.

[18] L. Breiman. Arcing the edge. 1997.

[19] M. Briere, K. Oosterlinck, and A. Szafarz. Virtual currency, tangible return: Portfolio diversification with bitcoins. 2013.

[20] F. Calefato, F. Lanubile, F. Maiorano, and N. Novielli. [journal first] sentiment polarity detection for software development. In 2018 IEEE/ACM 40th International Conference on Software Engineering (ICSE), pages 128–128. IEEE, 2018.

[21] L. Catania, S. Grassi, and F. Ravazzolo. Forecasting cryptocurrencies financial time series. BI Norwegian Business School, Centre for Applied Macro- and Petroleum Economics, 2018.

[22] P. Chaim and M. P. Laurini. Volatility and return jumps in bitcoin. Economics Letters, 173:158–163, 2018.

[23] C. Y.-H. Chen and C. M. Hafner. Sentiment-induced bubbles in the cryptocurrency market. Journal of Risk and Financial Management, 12(2):53, 2019.

[24] T. Chen and C. Guestrin. Xgboost: A scalable tree boosting system. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '16, page 785–794, New York, NY, USA, 2016. Association for Computing Machinery.

[25] Z. Chen, C. Li, and W. Sun. Bitcoin price prediction using machine learning: An approach to sample dimension engineering. Journal of Computational and Applied Mathematics, 365:112395, 2020.

[26] F. Chollet. Keras. Available at https://keras.io (accessed 20 June 2019), 2015.

[27] F. Chollet et al. keras, 2015.

[28] P. Ciaian, M. Rajcaniova, and D. A. Kancs. The economics of bitcoin price formation. Applied Economics, 48(19):1799–1815, 2016.

[29] L. Cocco, R. Tonelli, and M. Marchesi. An agent-based artificial market model for studying the bitcoin trading. IEEE Access, 7, 2019.

[30] L. Cocco, R. Tonelli, and M. Marchesi. An agent based model to analyze the bitcoin mining activity and a comparison with the gold mining industry. Future Internet, 11(1):8, 2019.

[31] Coinmarketcap. http://www.coinmarketcap.com. (accessed 21 April 2020), 2020.

[32] L. W. Cong, L. Yen, and W. Neng. Tokenomics: Dynamic adoption and valuation. Becker Friedman Institute for Research in Economics Working Paper, (2018-49), 2018.

[33] C. Conrad, A. Custovic, and E. Ghysels. Long-and short-term cryptocurrency volatility components: A garch-midas analysis. Journal of Risk and Financial Management, 11(2):23, 2018.

[34] A. N. D. Blei and M. Jordan. Latent dirichlet allocation. Journal of Machine Learning Research, vol. 3:993–1022, 2003.

[35] C. Danescu-Niculescu-Mizil, M. Sudhof, D. Jurafsky, J. Leskovec, and C. Potts. A computational approach to politeness with application to social factors. In Proceedings of ACL, 2013.

[36] R. K. Dase and D. D. Pawar. Application of artificial neural network for stock market predictions: A review of literature. International Journal of Machine Intelligence, 2(2):14–17, 2010.

[37] J. C. de Albornoz, L. Plaza, and P. Gervás. Sentisense: An easily scalable concept-based affective lexicon for sentiment analysis. In LREC, pages 3562–3567, 2012.

[38] G. Destefanis, S. Counsell, G. Concas, and R. Tonelli. Software metrics in agile software: An empirical study. 15th Int. Conf. on Agile Software Development, XP 2014, vol. 179, 2014.

[39] G. Destefanis, M. Ortu, S. Counsell, S. Swift, M. Marchesi, and R. Tonelli. Software development: do good manners matter? PeerJ Computer Science, 2:e73, 2016.

[40] S. Drożdż, L. Minati, P. Oświęcimka, M. Stanuszek, and M. Wątorek. Signatures of crypto-currency market decoupling from the forex. arXiv preprint arXiv:1906.07834, 2019.

[41] D. Enke and N. Mehdiyev. Stock market prediction using a combination of stepwise regression analysis, differential evolution-based fuzzy clustering, and a fuzzy inference neural network. Intelligent Automation and Soft Computing, 19(4):636–648, 2013.

[42] L. Fahrmeir, T. Kneib, S. Lang, and B. Marx. Regression. Springer, 2007.

[43] M. P. Fay and M. A. Proschan. Wilcoxon-mann-whitney or t-test? on assumptions for hypothesis tests and multiple interpretations of decision rules. Statistics surveys, 4:1, 2010.

[44] B. V.-A. Franco Valencia, Alfonso Gómez-Espinosa. Price movement prediction of cryptocurrencies using sentiment analysis and machine learning. Computer Science, 2019.

[45] D. Garcia and F. Schweitzer. Social signals and algorithmic trading of bitcoin. Royal Society Open Science, vol. 2, 2015.

[46] C. W. Granger. Investigating causal relations by econometric models and cross-spectral methods. Econometrica: Journal of the Econometric Society, 37:424–438, 1969.

[47] D. Graziotin, X. Wang, and P. Abrahamsson. Happy software developers solve problems better: psychological measurements in empirical software engineering. PeerJ, 2:e289, 2014.

[48] D. Graziotin, X. Wang, and P. Abrahamsson. How do you feel, developer? an explanatory theory of the impact of affects on programming performance. PeerJ Computer Science, 1:e18, 2015.

[49] D. Graziotin, X. Wang, and P. Abrahamsson. Understanding the affect of developers: theoretical background and guidelines for psychoempirical software engineering. In Proceedings of the 7th International Workshop on Social Software Engineering - SSE 2015, pages 25–32, New York, New York, USA, 2015. ACM Press.

[50] A. Greave and B. Au. Using the bitcoin transaction graph to predict the price of bitcoin. Computer Science, 2015.

[51] T. Hastie, R. Tibshirani, and J. Friedman. The Elements of Statistical Learning. Springer Series in Statistics. Springer New York Inc., New York, NY, USA, 2001.

[52] T. T. Hastie and R. J. H. Friedman. The elements of statistical learning: Data mining, inference, and prediction. New York: Springer, 2001.

[53] A. Hawkes. Spectra of some self-exciting and mutually exciting point processes. Biometrika, vol. 58:83, 1971.

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

[55] H. Hotelling. Analysis of a complex of statistical variables into principal components. Journal of Educational Psychology, 24:417–441, and 498–520, 1933.

[56] H. Hotelling. Relations between two sets of variates. Biometrika, 28(3/4):321–377, 1936.

[57] R. Hyndman and G. Athanasopoulos. Forecasting: principles and practice. Chapter 6:157–182, 2014.

[58] M. R. Islam and M. F. Zibran. Towards understanding and exploiting developers' emotional variations in software engineering. In 2016 IEEE 14th International Conference on Software Engineering Research, Management and Applications (SERA), pages 185–192, June 2016.

[59] H. Jing-Zhi, H. William, and N. Jun. Predicting bitcoin returns using high-dimensional technical indicators. The Journal of Finance and Data Science, 2018.

[60] E. Jones, T. Oliphant, and P. Peterson. Scipy: Open source scientific tools for python. Available at: http://www.scipy.org/ (accessed 20 June 2019), 2001.

[61] P. Katsiampa. Volatility estimation for bitcoin: A comparison of garch models. Economics Letters, 158:3–6, 2017.

[62] A. Kazem, E. Sharifi, F. K. Hussain, S. Morteza, and O. K. Hussain. Support vector regression with chaos-based firefly algorithm for stock market price forecasting. Applied soft computing, 13(2):947–958, 2013.

[63] Z. Keskin and T. Aste. Information-theoretic measures for non-linear causality detection: application to social media sentiment and cryptocurrency prices. arXiv:1906.05740, 2019.

[64] I. A. Khan, W.-P. Brinkman, and R. M. Hierons. Do moods affect programmers' debug performance? Cognition, Technology & Work, 13(4):245–258, 2011.

[65] Y. B. Kim, J. G. Kim, W. Kim, J. H. Im, T. H. Kim, S. J. Kang, and C. H. Kim. Predicting fluctuations in cryptocurrency transactions based on user comments and replies. PLOS ONE, 11(8):1–17, 08 2016.

[66] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980, 2014.

[67] L. Kristoufek. Bitcoin meets google trends and wikipedia: Quantifying the relationship between phenomena of the internet era. Scientific Reports, vol. 3, 2013.

[68] M. M. L. Cocco, R. Tonelli. An agent-based artificial market model for studying the bitcoin trading. IEEE Access, vol. 7:42908–42920, 2019.

[69] M. M. L. Cocco, R. Tonelli. An agent based model to analyze the bitcoin mining activity and a comparison with the gold mining industry. Future Internet, vol. 11, 2019.

[70] S. Lahmiri and S. Bekiros. Cryptocurrency forecasting with deep learning chaotic neural networks. Chaos, Solitons and Fractals, 118:35 – 40, 2019.

[71] S. Lahmiri, S. Bekiros, and A. Salvi. Long-range memory, distributional variation and randomness of bitcoin volatility. Chaos, Solitons & Fractals, 107:43–48, 2018.

[72] T. R. Li, A. S. Chamrajnagar, X. R. Fong, N. R. Rizik, and F. Fu. Sentiment-based prediction of alternative cryptocurrency price fluctuations using gradient boosting tree model. Frontiers in Physics, 7:98, 2019.

[73] V. K.-S. Liew. Which lag length selection criteria should we employ? Economics bulletin, 3(33):1–9, 2004.

[74] D. Mallqui and R. Fernandes. Predicting the direction, maximum, minimum and closing prices of daily bitcoin exchange rate using machine learning techniques. Applied Soft Computing, 75:10.1016/j.asoc.2018.11.038, 2018.

[75] M. Mäntylä, B. Adams, G. Destefanis, D. Graziotin, and M. Ortu. Mining valence, arousal, and dominance: possibilities for detecting burnout and productivity? In Proceedings of the 13th International Conference on Mining Software Repositories, pages 247–258. ACM, 2016.

[76] L. Marchesi, M. Marchesi, G. Destefanis, G. Barabino, and D. Tigano. Design patterns for gas optimization in ethereum. In 2020 IEEE International Workshop on Blockchain Oriented Software Engineering (IWBOSE), pages 9–15. IEEE, 2020.

[77] K. H. McIntyre and K. Harjes. Order flow and the bitcoin spot rate. Applied Economics and Finance, pages 42908–42920, 2014.

[78] W. Mckinney. pandas: a foundational python library for data analysis and statistics. Python High Performance Science Computer, 2011.

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

[80] S. McNally, J. Roche, and S. Caton. Predicting the price of bitcoin using machine learning. 26th Euromicro International Conference on Parallel,and Network-Based Processing, PDP:339–343, 2018.

[81] J. Michael, A. Cohn, and J. R. Butcher. Blockchain technology. The Journal, 1(7), 2018.

[82] A. Murgia, M. Ortu, P. Tourani, B. Adams, and S. Demeyer. An exploratory qualitative and quantitative analysis of emotions in issue report comments of open source systems. Empirical Software Engineering, pages 1–44, 2017.

[83] A. Murgia, M. Ortu, P. Tourani, B. Adams, and S. Demeyer. An exploratory qualitative and quantitative analysis of emotions in issue report comments of open source systems. Empirical Software Engineering, 23(1):521–564, 2018.

[84] A. Murgia, P. Tourani, B. Adams, and M. Ortu. Do developers feel emotions? an exploratory analysis of emotions in software artifacts. In Proceedings of the 11th Working Conference on Mining Software Repositories, pages 262–271. ACM, 2014.

[85] V. Y. Naimy and M. R. Hayek. Modelling and predicting the bitcoin volatility using garch models. International Journal of Mathe-matical Modelling and Numerical Optimisation, vol. 8:197–215, 2018.

[86] V. Y. Naimy and M. R. Hayek. Modelling and predicting the bitcoin volatility using garch models. International Journal of Mathematical Modelling and Numerical Optimisation, 8:197–215, 2018.

[87] S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system. 2008.

[88] M. F. Neuts. A versatile markovian point process. Journal of Applied Probability, vol. 16, 1979.

[89] B. Ong, T. M. Lee, G. Li, and D. L. K. Chuen. Evaluating the potential of alternative cryptocurrencies. In Handbook of digital currency, pages 81–135. Elsevier, 2015.

[90] M. Ortu. Mining software repositories: measuring effectiveness and affectiveness in software systems. 2015.

[91] M. Ortu, G. Destefanis, M. Kassab, S. Counsell, M. Marchesi, and R. Tonelli. Would you mind fixing this issue? an empirical analysis of politeness and attractiveness in software developed using agile boards. Lecture Notes in Business Information Processing, pages 122–129, 2019.

[92] M. Ortu, T. Hall, M. Marchesi, R. Tonelli, D. Bowes, and G. Destefanis. Mining communication patterns in software development: A github analysis. In Proceedings of the 14th International Conference on Predictive Models and Data Analytics in Software Engineering, pages 70–79. ACM, 2018.

[93] M. Ortu, M. Orrú, and G. Destefanis. On comparing software quality metrics of traditional vs blockchain-oriented software: An empirical study. In 2019 IEEE International Workshop on Blockchain Oriented Software Engineering (IWBOSE), pages 32–37. IEEE, 2019.

[94] B. Pang and L. Lee. Opinion Mining and Sentiment Analysis. Foundations and Trends in Information Retrieval, 2(1-2):1–135, Jan. 2008.

[95] J. Patel, S. Shah, P. Thakkar, and K. Kotecha. Predicting stock market index using fusion of machine learning techniques. Expert Systems with Applications, 42:2162–2172, 2015.

[96] K. Pearson. On lines and planes of closest fit to systems of points in space. The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science, 2(11):559–572, 1901.

[97] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, A. Müller, J. Nothman, G. Louppe, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in python. Journal of Machine Learning Research, 12, 2012.

[98] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12:2825–2830, 2011.

[99] R. C. Phillips and D. Gorse. Cryptocurrency price drivers: Wavelet coherence analysis revisited. PloS one, 13(4):e0195200, 2018.

[100] R. C. Phillips and D. Gorse. Mutual-excitation of cryptocurrency market returns and social media topics. In Proceedings of the 4th International Conference on Frontiers of Educational Technologies, pages 80–86. ACM, 2018.

[101] D. G. R. C. Phillips. Predicting cryptocurrency price bubbles using social media data and epidemic modelling. Entropy, 2017.

[102] S. R. Rehurek, P. Software framework for topic modelling with large corpora. In proceedings of the LREC 2010 workshop on new challenges for NLP frameworks, pages 45–50, 2010.

[103] J. Roman and A. Jameel. Backpropagation and recurrent neural networks in financial analysis of multiple stock market returns. In Proceedings of HICSS-29: 29th Hawaii International Conference on System Sciences, volume 2, pages 454–460. IEEE, 1996.

[104] J. A. Russell. Emotion, core affect, and psychological construction. Cognition & Emotion, 23(7):1259–1283, Nov. 2009.

[105] S. Skipper and Perktold. Statsmodels: Econometric and statistical modeling with python. In proceedings of the 9th Python in Science Conference, 2010.

[106] D. Stanisław, G. Robert, M. Ludovico, O. Paweł, and W. Marcin. Bitcoin market route to maturity? evidence from return fluctuations, temporal correlations and multiscaling effects. Chaos: An Interdisciplinary Journal of Nonlinear Science, 28(7):071101, 2018.

[107] M. Stocchi and M. Marchesi. Fast wavelet transform assisted predictors of streaming time series. Digital Signal Processing, 77:5–12, 2018.

[108] M. S. Stuart G. Colianni, Stephanie M. Rosales. Algorithmic trading of cryptocurrency based on twitter sentiment analysis. Computer Science, 2015.

[109] D. U. Sutiksno, A. S. Ahmar, N. Kurniasih, E. Susanto, and A. Leiwakabessy. Forecasting historical data of bitcoin using arima and α-sutte indicator. Journal of Physics: Conference Series, 1028:conference 1, 2018.

[110] D. L. Thornton and D. S. Batten. Lag-length selection and tests of granger causality between money and income. Journal of Money, credit and Banking, 17(2):164–178, 1985.

[111] A. Urquhart. The inefficiency of bitcoin. Economics Letters, 148:80–82, 2016.

[112] V. Vapnik and A. Chervonenkis. A note on one class of perceptrons. Automation and Remote Control, 25, 01 1964.

[113] N. Vo and G. Xu. The volatility of bitcoin returns and its correlation to financial markets. IEEE, 2017.

[114] T. Walther, T. Klein, and E. Bouri. Exogenous drivers of bitcoin and cryptocurrency volatility–a mixed data sampling approach to forecasting. University of St. Gallen, School of Finance Research Paper, (2018/19), 2019.

[115] A. B. Warriner, V. Kuperman, and M. Brysbaert. Norms of valence, arousal, and dominance for 13,915 English lemmas. Behavior Research Methods, 45(4):1191–1207, dec 2013.

[116] Yahoofinance. http://www.finance.yahoo.com. (accessed 21 April 2020), 2020.

[117] S. Y. Yang and J. Kim. Bitcoin market return and volatility forecasting using transaction network flow properties. IEEE, 2016.

[118] S. H. YD, P. Matjaž, and R. H. V. Clustering patterns in efficiency and the coming-of-age of the cryptocurrency market. Scientific reports, 9(1):1440, 2019.

[119] A. Yelowitz and M. Wilson. Characteristics of bitcoin users: an analysis of google search data. Applied Economics Letters, 22(13):1030–1036, 2015.

List of Publications Related to the Thesis

Published papers

Journal papers

1. N. Uras, L. Marchesi, M. Marchesi, and R. Tonelli. Forecasting bitcoin closing price series using linear regression and neural networks models. PeerJ Computer Science, 6:e279, July 2020.

2. S. Bartolucci, G. Destefanis, M. Ortu, N. Uras, M. Marchesi, and R. Tonelli. The Butterfly "Affect": impact of development practices on cryptocurrency prices. EPJ Data Science, vol. 9, July 2020.

Conference papers

1. N. Uras, S. Vacca, and G. Destefanis. Investigation of mutual-influence between blockchain development communities and cryptocurrency price changes. WETSEB workshop proceedings, May 2020.

2. N. Uras, and M. Ortu. Investigation of Blockchain Cryptocurrencies Price Movements through Deep Learning: A Comparative Analysis. 2021 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER), March 2021.

3. S. Vacca, C. L. Costerbosa, A. Spada, G. Riotta, and N. Uras. Investigation of coronavirus impact on blockchain and cryptocurrencies markets. WETSEB workshop proceedings, May 2021.