UCLouvain

LOUVAIN
School of Management

Louvain School of Management

Predicting the return and volatility of cryptoassets using Machine Learning

Author(s): Alessandro Coco
Supervisor(s): Mikael Petitjean and Christophe Desagre
Academic year 2021-2022
Dissertation for the master of
Ingénieur de gestion, à finalité spécialisée
Daytime schedule

Abstract :

Recently, the interest for artificial intelligence, and more specifically for Machine Learning, has been constantly growing. This method, which has the advantage of being able to self-learn, but also to analyze large amount of data in a short time, is progressively becoming widely used in large companies. Indeed, they apply this approach to enable solving complex problems as well as helping them in decision-making. In parallel, recent events such as the massive purchase of Bitcoin by several countries and global firms have created a craze around cryptocurrencies. In this dissertation, we will relate these two elements by predicting the price of ten cryptoassets using Machine Learning techniques. The results of this analysis show that linear regression, gradient boosting and random forest are the most suitable algorithms to predict the price of cryptocurrencies. We will also demonstrate that it is possible to outperform buy-and-hold approaches using momentum investing based strategies.

UNIVERSITÉ CATHOLIQUE DE LOUVAIN
Louvain School of Management

Place des Doyens, 1 bte L2.01.01, 1348 Louvain-la-Neuve
Boulevard Emile Devreux 6, 6000 Charleroi, Belgique
Chaussée de Binche 151, 7000 Mons, Belgique

www.uclouvain.be/lsm

Acknowledgements

I would first like to acknowledge my two co-supervisors Professor Mikael Petitjean and Professor Christophe Desagre, who made this thesis possible thanks to their advice and guidance during all the process of researching and testing for this thesis.

I would also like to thank my family that allows me to pursue my studies at the university and for providing me support and motivation throughout my years of study.

Table of Contents

List of Tables

List of Figures

1 Introduction

This master's thesis studies the prediction of cryptoasset prices with Machine Learning algorithms. Throughout this thesis, we will attempt to answer this research question: "Can we beat the market by using Machine Learning techniques in the cryptoasset world?".

I chose this subject for many reasons. I am interested in finance, that is why I decided to have a specialization in financial engineering and it strengthened my interest in financial and cryptocurrency markets. Furthermore, the interest for artificial intelligence, and especially Machine Learning, has been booming for few years in several areas. Being a cryptocurrency investor, I found it interesting to combine these two subjects to experiment if Machine Learning is suitable for the cryptoasset market and, if it is, to find new investing strategies that could help making more profits. My motives are therefore mainly personal. However, I believe that the core of this study can be relevant to other volatile assets or stocks.

The contribution of this thesis, compared with the literature review, is to predict returns of cryptoassets using Machine Learning techniques to build different strategies, notably momentum investing portfolios, in order to attempt beating buy-and-hold strategies (equal weight and risk parity).

Regarding the structure of this work, we will begin by the theoretical part. This section will be organized as follows.

First, we will explain what is cryptocurrency and the different types of investing. Then we will clarify the concept of Machine Learning by outlining some fields of application, its benefits and how it operates. We will describe several major Machine Learning algorithms as well. After that, we will explain some features added in our models for the empirical analysis and describe four important measures of performance to evaluate the accuracy of models. Finally, we will end this section by discussing the related literature on price predictions with Machine Learning.

Concerning our empirical analysis, we will rely on datasets containing the close price of 10 major cryptoassets (ADA, BNB, BTC, DASH, DOGE, ETH, LINK, LTC, XLM, XRP) which already existed in 2017. We will use this data to predict the prices of each of them with 6 different Machine Learning algorithms. Then, we will select the three best fitting models and establish three portfolios based on momentum investing strategy and observe if it is possible to beat the market with such approaches.

Ultimately we will finalize this work with the presentation of our results for the empirical analysis and make a general conclusion. We will also mention the limits encountered during this study and suggest different approaches to go deeper in the analysis for future researches.

2 Theory

2.1 Cryptocurrency

A cryptocurrency (or cryptoasset) can be defined as a subset of digital currency (Lam and Lee, 2015). In reaction to the global financial crisis, Satoshi Nakamoto created in 2008 the first cryptocurrency: the Bitcoin. This invention allows "online payments to be sent directly from one party to another without going through a financial institution" (Nakamoto, 2008). Since the creation of Bitcoin, thousands of new cryptocurrencies have emerged and are now commonly called altcoins. Cryptocurrencies rely on the blockchain technology, which is a shared and immutable ledger that records all transactions and it also provides a solution for the double-spending problem. In contrast to fiat currencies, these digital currencies are characterized by a limited supply, for instance Bitcoin has a supply of 21 million coins. The other key features of cryptocurrencies are the fast speed of transactions, the decentralization and the transparency thanks to all transaction records stored in the blockchain (Lee et al., 2017).

In addition to being a means of payment for certain goods such as cars or apartments in some countries, cryptocurrencies can also be considered speculative assets and it is in this perspective that we will use them in our thesis. The list below contains the 10 cryptoassets we will use in this thesis, as well as their ticker symbol:

- ▷ Cardano (ADA);
- ▷ Binance Coin (BNB);
- ▷ Bitcoin (BTC);
- ▷ Dash (DASH);
- ▷ Dogecoin (DOGE);
- ▷ Ethereum (ETH);
- ▷ Chainlink (LINK);
- ▷ Litecoin (LTC);
- ▷ Stellar (XLM);
- ▷ Ripple (XRP)

2.1.1 Passive investment

Passive investment is the easiest strategy to implement in a portfolio. This type of investing is based on the buy-and-hold strategy and therefore is usually used in a long term horizon. As its name suggests, it consists in buying the assets and holding them for the entire duration of the investment. This strategy supports the idea that markets turn out

to be efficient, in the sense they assimilate and adjust to new information (Malkiel, 2003).

The main benefits of passive investing over active investing are the low transaction costs, as there is no need of rebalancing, and the saving of time that would have been lost in searching for information to keep up to date. Over the years, the popularity for the different forms of buy-and-hold investments have increased, especially in ETFs, which are trackers that replicate stock indices such as the S&P 500, the CAC 40 or the NASDAQ (Babiak and Bianchi, 2022).

In this thesis, two weighting approaches in passive investment will be discussed: the equal weight portfolio and the risk parity.

The equal weight portfolio is obtained by giving the same importance to each asset, no matter their volatility or their market capitalization. In this way, it allows the portfolio to reduce the concentration of the unsystematic risk (Statman, 1987). On the other hand, this method is more susceptible to increase the systematic risk due to the higher exposure in small capitalization assets (Solactive, 2018).

The second approach is the risk parity. Popular since the 2008 crisis, the purpose of this strategy is to equalize risk contributions between the different asset classes (Anderson et al., 2012).

2.1.2 Active investment

In contrast with passive investment, active investment aims at rebalancing the portfolio to attempt to beat the market by taking advantage of price fluctuations (Keane, 1986). Consequently, investors have to analyze the market and keep abreast with financial and political news. This strategy therefore rejects the efficient markets hypothesis, and some studies show that "most investors must underperform the market average" (Malkiel, 2003). The main drawback of this investing method is the high transaction costs involved in rebalancing that may affect profitability.

The type of active investments which we will focus on in this study is momentum strategy. As opposed to contrarian strategies, this approach, based on asset returns, recommends riding the trend by selling past losers and buying past winners (Liu et al., 1999). We can subdivide momentum strategy in 3 categories.

First, the absolute momentum which only selects assets with positive returns in the portfolio. Therefore, this technique compares each asset with its own historical performance (Ryou et al., 2020). If all assets are losing value, the absolute momentum investors will avoid investing. The advantages of this approach are its simplicity, as it holds for

long-only investing, and its applicability to all types of assets or portfolios (Antonacci, 2013).

Second, the relative momentum strategy consists in selling underperforming assets and buying outperforming assets. Thus, the investor has to compare the past returns of all assets to rebalance his portfolio. The drawback of this approach is that even if the returns of all assets are negative, this strategy forces the investor to invest. Indeed, in this case he will invest in assets that are going down the least (Ryou et al., 2020).

Third, the dual momentum is the combination of absolute and relative momentum. In this way, the dual momentum investor will buy the strongest winner portfolio and short sell the weakest loser portfolio. This strategy generally provides better returns than the two other ones (D'Souza et al., 2016).

2.2 Machine Learning

Nowadays, Machine Learning is more and more applied in several fields. This scientific area is a sub-category of artificial intelligence that is able to emulate human intelligence (El Naqa and Murphy, 2015). It allows algorithms to detect patterns without receiving literally programming instructions (Jones and Rasekhschaffe, 2019). To that end, they need large databases in input to train and learn, so it explicitly relies on Big Data. These datasets can be from any type such as images, prices, words or numbers (Bastien, 2021). Machine Learning algorithms can deliver high quality predictions that may improve over time as they autonomously learn. The increasing performance throughout the years with these models can be explained by the growing computing power, the increase of data availability and the emergence of new algorithms thanks to the combination of the previous points (Jones and Rasekhschaffe, 2019).

Currently, a main field of application of Machine Learning is finance. It is used notably in risk assessment, credit score calculations, asset management and even loan approvals (CFI Team, 2022). Aziz et al. (2021) made a classification of major researches in this area (Aziz et al., 2021). We can also find numerous studies focusing on price predictions with Machine Learning models because of the capacity of algorithms to recognize complex patterns in multiple situations (Henrique et al., 2019). These techniques also help investors in decision-making thanks to their very fast analysis speed. Another major benefit of Machine Learning is the absence of emotional biases that are present in human decision making, it only relies on facts.

When constructing a Machine Learning model, there is a risk of overfitting. That means the model does not generalize well from observed data to unseen data. Consequently, it will perform greatly on training set but predict poorly on testing set. The causes of this problem can be varied. For instance, it may be the result of a too small training set size (Ying, 2019).

2.2.1 Algorithms

Machine Learning algorithms can be categorize in 2 classes (Ren et al., 2016):

- ▷ Classification algorithms: Classification is used when the objective is to find the decision boundary which can divide the dataset into different classes. The output variables are therefore represented by discrete values.
- ▷ Regression algorithms: Regression is used when the objective is to find the best fitting line that can predict the output more accurately. The predicted variables are therefore represented by continuous values.

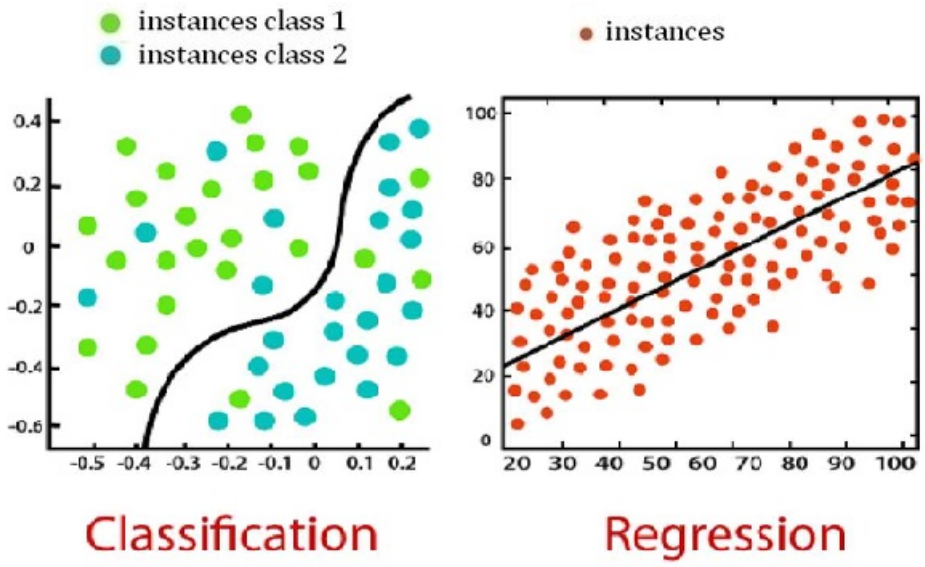

Figure 1: Diagram of Classification VS Regression (Belkacem, 2021)

Given that the aim of this thesis is the prediction of returns, we will only focus on regression models with the six following algorithms.

2.2.1.1 Random Forest

Random forest is a learning method that operates by combining multiple decision trees. The random forest model makes the final decision by taking the majority of the trees. The main benefit of this algorithm is the reduction of the overfitting risk and the required training time. Another advantage of using this model is that the data does not need to be transformed or rescaled, so the pre-processing required is small. In addition, the accuracy level provided by the random forest algorithm is high. This algorithm performs efficiently in large datasets, whether for classification or regression problems (Akyildirim et al., 2020).

2.2.1.2 Linear Regression

Linear regression is one of the most common machine learning algorithms that can be used to predict discrete or continuous variables. This algorithm allows to find the linear relationship between one or more independent variables and a dependent variable. If there is only one independent variable this is a simple linear regression, otherwise we are dealing with a multiple linear regression (Maulud and Abdulazeez, 2020).

2.2.1.3 Decision Tree

Decision tree is a widely used Machine Learning algorithm thanks to its ease of interpretation. It can perform both classification and regression tasks. Decision trees are composed by three types of nodes.

The first one is the Root Node which represents the entire population or sample and may be divided into several homogeneous sets. Then, the Interior Nodes constitute the features of a dataset, the attribute test and the decision rules are represented by the branches. Ultimately, the Leaf Nodes correspond to the predicted class labels for a classification, or the outcome for a regression problem. Using numerous iterations, the algorithm can perform the prediction of a target value (Gurucharan, 2020).

We can see the depiction of a decision tree on Figure 2.

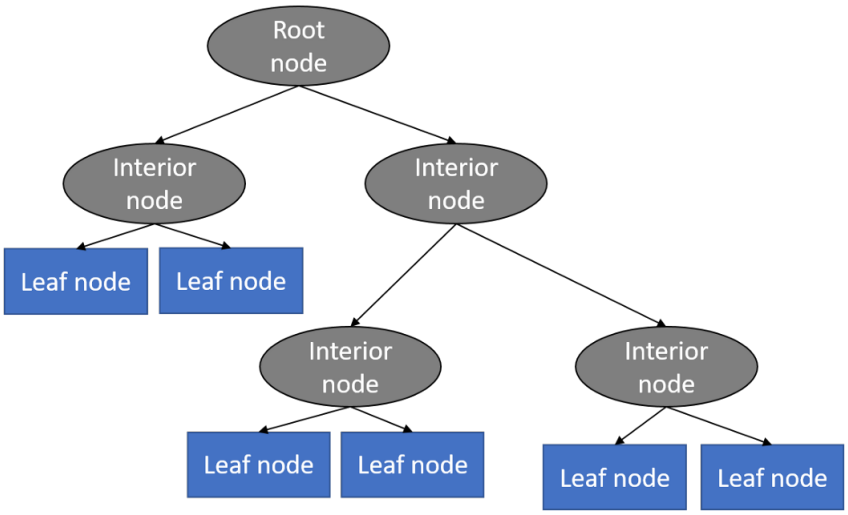

Figure 2: Decision tree diagram (Schlagenhauf, 2022)

2.2.1.4 Gradient Boosting Regression

Gradient boosting regression is a popular Machine Learning algorithm for large and complex datasets. This method allows to find any nonlinear relationship between features and the model target, with minimal data pre-processing. It is also able to deal with outliers and missing values without any special treatment (Masui, 2020).

The process of gradient boosting is to build models sequentially, where these successive models attempt to reduce the errors of the previous ones (Saini, 2021).

2.2.1.5 K-Nearest Neighbors

K-nearest neighbors is a Machine Learning algorithm that can be used to solve classification and regression problems. The idea of this method is based on the assumption of locality in data space. K-nearest neighbors relies on similarity to predict the value of a testing point by making a weighted average of the k-nearest neighbors to the point (Hu et al., 2014).

2.2.1.6 Multilayer Perceptron

The multilayer perceptron algorithm is a class of feedforward artificial neural network

that has three or more layers. These layers are organized into 3 categories: a single input layer, one or more hidden layers, and a single output layer of perceptrons, as we can see on the figure 3 (Jaokar, 2019). Each node in this model is a neuron that uses a nonlinear activation function, except for the input nodes. This function defines the output of that node given an input or set of inputs. The formulas below describes the two basic sigmoidal activation functions currently employed in applications.

$$\mathrm{y(x)} = tanh(x)$$

$$\mathrm{y(x)} = \frac{1}{1 + e^{-x}}$$

The first formula is a hyperbolic tangent ranging from -1 to 1. The second one is a logistic sigmoid which ranges from 0 to 1. These activation functions are utilized because they have practical mathematical qualities and because the MLP network can easily express close-to-linear mappings thanks to their nearly linear behavior near the origin (Valpola, 2000). This algorithm is suitable for both classification and regression tasks (Mohanty, 2019).

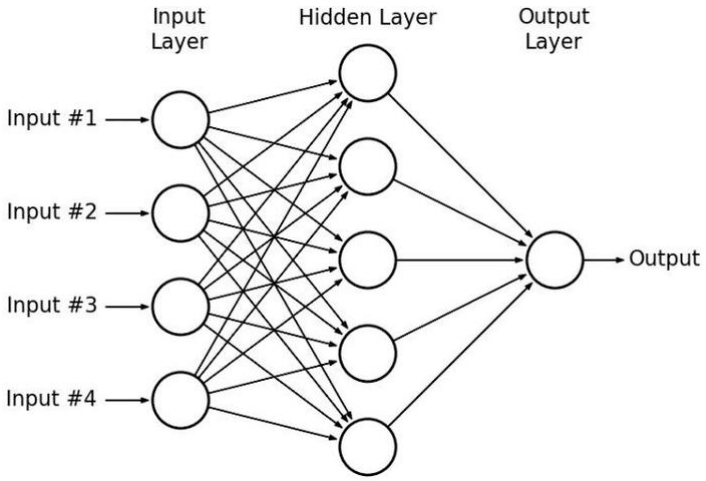

Figure 3: Multilayer perceptron diagram (Hassan et al., 2015)

2.3 Features - Technical indicators

To help the models to make predictions, we can add features for the training of Machine Learning algorithms. Any traditional technical indicator can be used as a feature to improve a model (Hilpisch, 2020).

2.3.1 Simple Moving Average

A simple moving average (SMA) computes the average of a selected range of prices by the number of periods (n) in that range. This indicator is often used to determine the trend direction of an asset. It is calculated by adding the price of the past n periods and dividing the total by the number of time periods.

$$SMA_t = \frac{A_{t-1} + A_{t-2} + \cdots + A_{t-n}}{n}$$

Where :

A_t = asset price at period t

n = total number of periods

2.3.2 Moving standard deviation

The moving standard deviation measures the market volatility. It permits to calculate the standard deviation of prices from the moving average of the prices by specifying the number of periods. This indicator does not predict a market direction but it can be used to confirm a forecast (Zaner, 2022).

$$\mathrm{d} = \frac{(A_1 - SMA)^2 + (A_2 - SMA)^2 + \cdots + (A_n - SMA)^2}{n}$$

Where :

A_n = asset price at period n

n = total number of periods

2.3.3 Relative Strength Index

The Relative Strength Index (RSI) is a momentum oscillator, developed by J. Welles Wilder, which measures the speed and change of price movements. This indicator oscil-

lates between 0 and 100, where a value above 70 considers that the asset is overbought, therefore a trend reversal may happen. In contrast, a value below 30 indicates an oversold asset. The chart may also reveal divergences or failure swings that are understandable signals for traders. J. Welles Wilder suggested 14 as the default number of periods for the RSI. However, it also operates well for other number of periods. To compute the RSI, we first have to calculate the Relative Strength (RS) which is obtained by dividing the average gain over the n past periods by the average loss over the n past periods (Fernando, 2022).

$$\text{RS} = \frac{\text{Average gain}}{\text{Average loss}}$$

Then, we can get the RSI with a value between 0 and 100 thanks to the following formula.

$$\text{RSI} = 100 - \frac{100}{1 + RS}$$

2.3.4 Lags

The lag operator is a function that shifts a time series in a manner that the "lagged" values are aligned with the actual time series. In time series, this process is valuable for the autocorrelation effect. The latter represents the correlative tendency between the values within a time series and previous copies of itself. This phenomenon permits to identify seasonality which is the recurrence of patterns at periodic frequencies (Dancho, 2018).

The idea behind lag features is that what happened in the past contains a form of intrinsic information about the future. This implies that today's outcomes are dependent of yesterday's outcomes. Some models aim at choosing the "best lag", it refers to the lag with the most power to explain some outcome.

2.4 Measures of performance

The performance measurement of models is quite different for classification and regression algorithms. Indeed, the assessment for classifications is simply an accuracy formula computed by the number of correct predictions divided by the total number of predictions. On the other hand, in regression analysis no consensus has been reached to adopt a standard metric to measure the performance. As a result, studies employs various evaluation methods which the most popular ones are explained in this section (Dancho, 2018).

2.4.1 Mean Absolute Error

The Mean Absolute Error (MAE) measures the average of the absolute difference between predicted values and actual observations over the test sample. This indicator gives the same weight to all errors and ranges from 0 to ∞, where a value of 0 translates a perfect model. Therefore, the smaller the MAE, the closer the fit is to the data (Chai and Draxler, 2014).

$$\text{MAE} \ = \ \frac{1}{n} \ \sum_{j=1}^{n} \mid y_j - \hat{y}_j \mid$$

Where :

n = the number of observations

y_j = the j^{th} observed value

$\hat{y}_j$ = the j^{th} predicted value

2.4.2 Mean Squared Error

The Mean Squared Error (MSE), also known as the Mean Squared Deviation (MSD), measures the average squared difference between predicted values and actual observations. A large value of the MSE reflects that the data points are dispersed widely around its mean, consequently, a low MSE indicates high quality predictions (Frost, 2022).

$$\text{MSE} \ = \ \frac{1}{n} \ \sum_{j=1}^{n} (\ y_j - \hat{y}_j \ )^2$$

Where :

n = the number of observations

y_j = the j^{th} observed value

$\hat{y}_j$ = the j^{th} predicted value

2.4.3 Root Mean Squared Error

The Root Mean Squared Error (RMSE) is the square root of the average of squared differences between predictions and actual observations. It is basically the square root of the MSE previously explained so here again, lower values are preferred. The value of this indicator tends to be larger than MAE because the RMSE penalizes large errors while the MAE assigns the same weight to all errors. This performance measurement, which is one of the most used for regression problems, can be calculated with the following formula (Chai and Draxler, 2014).

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{j=1}^{n}(y_j - \hat{y}_j)^2}$$

Where :

n = the number of observations

y_j = the j^{th} observed value

$\hat{y}_j$ = the j^{th} predicted value

2.4.4 Coefficient of determination

The coefficient of determination, also known as R-squared (R^2), is a statistical measure in a regression model that represents the proportion of the variance for a dependent variable that is explained by the independent variable(s). It shows the goodness of a fit on a 0 to 100% scale therefore, a high percentage indicates a great accuracy of the model. Consequently, a single value says much about the performance of a regression, contrary to the MAE, MSE and RMSE. However, the coefficient of determination can sometimes be negative, it means that the trend of the data is not followed by the model. A negative R^2 indicates that the model fits the data really poorly in such a way that a horizontal line would fit better. To compute the R-squared, we first have to calculate the mean of the observed data ($\bar{y}$), the residual sum of squares (RSS) and the total sum of squares (TSS) thanks to the formulas below (Chicco et al., 2021).

$$\bar{y} \;=\; \frac{1}{n} \sum_{j=1}^{n} y_j$$

$$\text{RSS} \;=\; \sum_{j=1}^{n} \left(y_j - \hat{y}_j \right)^2$$

$$\text{TSS} \;=\; \sum_{j=1}^{n} \left(y_j - \bar{y}_j \right)^2$$

$$R^2 \;=\; 1 \;-\; \frac{\text{RSS}}{\text{TSS}}$$

According to Chicco et al.(2021) studies, the coefficient of determination is the most informative and truthful measure of performance in regression analysis, better than the other approaches seen in previous sections, and they suggest the application of R^2 as standard metric to evaluate regression analysis in any scientific field (Chicco et al., 2021).

The limit of R-squared is that the more you add, the higher the coefficient of determination is, so even if irrelevant variables are added in the model, the R-squared will increase. To prevent this problem, we can use the Adjusted Coefficient of Determination (Adjusted R-squared). This assessment method is an adjustment for the R-squared that takes into account the number of variables in a data set and penalizes the points that don't fit the model (Glen, 2022).

$$\textit{Adjusted } R^2 \;=\; 1 \;-\; \frac{\left(1 - R^2 \right) \left(n - 1 \right)}{n - p - 1}$$

Where :

n = the number of observations

p = the number of independent variables (predictors)

2.4.5 Sharpe ratio

Invented by William Sharpe, the Sharpe ratio is a measure of investment portfolio performance that helps investors to evaluate the return on an investment in relation to its risk. The Sharpe ratio describes the return of a portfolio, without taking into consideration

the "risk-free" interest rate and estimates the return percentage for each risk unit carried by the investment (Iania, 2021).

It is calculated as follows:

$$\text{Sharpe ratio} = \frac{R_p - R_f}{\sigma_p}$$

Where :

R_p = The portfolio return

R_f = The risk-free rate

σ_p = The standard deviation of the portfolio

The risk-free rate, as the name suggests, represents the return of an investment with zero risk, for instance the yield for a U.S. Treasury bond. The higher the Sharpe ratio is, the more the risk-adjusted return is attractive (Jason Fernando, 2022).

2.5 Literature review

In this section, important studies about Machine Learning performances and portfolios that are able to outperform buy and hold strategies will be gathered and compared. The purpose is to find out what has been done in this area and to draw a global conclusion. It will allow us to observe if we reach the same findings in our empirical analysis and how can we go further. Recently, several studies have been conducted in Machine Learning to predict stock prices and analyze price patterns to such an extend that most traders now depend on intelligent trading systems. Indeed, it assists them by making instant investment decisions and by predicting price in various circumstances (Shah, 2007).

The first study we will focus on was conducted by Reddy and Sriramya (2020). After noticing that many papers attempting to forecast asset prices had a low accuracy, they decided to predict the Bitcoin price by taking into consideration multiple parameters that affect the Bitcoin value. To reduce the time complexity in their model, Reddy and Sriramya used a regularization technique called least absolute shrinkage selection operator (LASSO). The latter has the advantage of finding results in a large database very rapidly. This paper also provides a comparison of LASSO algorithm with other Machine Learning algorithms such as k-nearest neighbors, linear regression, random forest and others. The results of this study show linear regression algorithm has the highest accuracy, and is therefore the most efficient algorithm. Concerning the time complexity in Bitcoin price prediction, Reddy and Sriramya conclude that using LASSO algorithm permits to more reduce the running time than the other algorithms (Reddy and Sriramya, 2020).

Another important study was undertaken by Kaczmarek and Perez in 2021. Their paper aims at predicting the price of stocks from the S&P500 and STOXX600 using Machine Learning algorithm techniques, and then optimizing different portfolios based on the cross-section of expected excess-return predictions to attempt to outperform the equal-weighted portfolio. They constructed two types optimized portfolios based on different techniques to try to beat the 1/N rule. The first ones rely on mean-variance analysis, also called modern portfolio theory (MPT) and conceptualized by Markowitz in 1952, that maximizes the Sharpe ratio. The second portfolios are optimized with the modern Hierarchical Risk Parity (HRP) proposed by López de Prado, 2016 that does not use expected returns as input data. For this study, Kaczmarek and Perez used stocks from the S&P500 index with history from 1999 to 2019 and they repeated all calculations with the data for stocks from the STOXX600 index to confirm the robustness of their results. The predictions are made with the random forest algorithm and the n stocks with the highest monthly predictions are chosen to be in the portfolios, and then their weights are allocated using three approaches: equal weight, mean-variance and Hierarchical Risk Parity optimization. The findings of this study highlight that according to the Sharpe

ratio, the mean-variance optimized portfolios outperform the equal-weighted portfolios by 16.5%, which is in total opposition with DeMiguel et al., 2009 who demonstrate that a traditional mean-variance optimizer underperforms a equal-weighted portfolio by 41.1%. Moreover, none of the 11 other modified mean-variance optimizers outperforms an equal-weighted portfolio.
The Hierarchical Risk Parity (HRP) optimizer gets also great results with an outperformance of 18% over the equal-weighted portfolios. Kaczmarek and Perez's paper proves that optimizers are very powerful tools that cooperate greatly with Machine Learning predictions and even enhance the performance of portfolios (Kaczmarek and Perez, 2021).

Many papers attempted to argue with the efficient market hypothesis by operating different strategies using Machine Learning techniques to overperform the buy-and-hold strategy supporting the idea that stock prices follow a random walk. This is the case of the Skabar and Cloete's study (2002) that employs a trained Neural Network algorithm to identify buy and sell points for financial assets. In their empirical analysis, they use the close values, from 1 July 1996 to 30 June 2001, and compare the returns of four financial assets applying a bootstrapping procedure: the Dow Jones Industrial Average, the Australian All Ordinaries, the S&P500 and the NASDAQ. The results show that, in contrast with the efficient market hypothesis, some time series are not completely random and that a trading strategy relying only on historical price data may be employed to get better returns than with a buy-and-hold strategy (Skabar and Cloete, 2002).

When we forecast, predictive indicators may be included in the model to strengthen it. This is especially what Wolff and Neugebauer (2019) made for their study. Indeed, they analyze equity premium predictions using traditional linear models and tree-based Machine Learning algorithms with additional indicators as an expand to the dataset. These several indicators that are classified in four classes : fundamental, macroeconomic, sentiment and risk. In their empirical analysis, Wolff and Neugebauer compare individually the predictive power for each indicator and the result shows that at least one indicator of each panel has statistically significant predictive power. That is why by combining predictors in multivariate models, they achieve higher forecast performance in comparison with the historical average. The other important finding in this paper is that contrary to most linear models, the tree-based Machine Learning approaches fail to outperform the historical average benchmark forecast. This outcome is in contradiction with the results of Gu et al., 2018 who state a higher prediction accuracy of Machine Learning algorithms. Wolff and Neugebauer justify this inconsistency by the different size of the datasets used in their model. Actually, Gu et al. (2018) make predictions on a considerable dataset of more than 900 signals, while Wolff and Neugebauer relies on a small database of 168 monthly observations. Therefore, Machine learning algorithms are

stronger than regression-based models for high-dimensional prediction problems, while for low-dimensional problems this is the contrary (Wolff and Neugebauer, 2019).

The growing interest in explaining or predicting the behavior of cryptocurrency prices leads Koker and Koutmos, 2020 to research in this area. In their paper, they build a direct reinforcement model for active trading in cryptocurrency and compare it with a naive buy-and-hold approach. Reinforcement is a domain of Machine Learning that aims to learn actions from successive experiments to maximize the notion of cumulative reward. The study considers the five major cryptocurrencies in circulation, namely, Bitcoin, Ethereum, Litecoin, Ripple and Monero, with a sample range from 26 August 2015 to 12 August 2019. The results of this study demonstrate that reinforcement Machine Learning is capable of optimizing actively managed portfolios when trading cryptocurrencies, even when actual transaction costs are taken into account. Furthermore, it proves that if the model is properly trained, active trading technique can provide investors greater risk-adjusted returns relative to buy-and-hold approach, and even lower portfolio downside risk over time. The empirical analysis also shows that cryptocurrency prices may not follow a random walk process (Koker and Koutmos, 2020).

Finally, we notice the most extensive researches, such as the second study described in this section by Wolff and Neugebauer (2019), are based on stocks and very rarely on cryptocurrencies. Furthermore, we also see that the findings of some papers can sometimes contradict the results of others, therefore, it is appropriate to rely only on studies based on similar assets or equivalent database size. That is why we will focus our research on predicting returns of cryptoassets using Machine Learning techniques with the aim of outperforming the buy-and-hold strategy.

3 Empirical analysis

3.1 Objective

As mentioned in the preceding section, few studies focus on Machine Learning performances on cryptoassets. For this reason, in this master's thesis we will concentrate on this area by using Machine Learning algorithms to predict cryptocurrency returns and try to outperform buy-and-hold strategies (equal-weighted and risk parity portfolio). To that end, we will rely on 6 different algorithms (random forest, linear regression, decision tree, gradient boosting regression, k-nearest neighbors and multilayer perceptron) and consider 10 major cryptoassets below:

- ▷ Cardano (ADA);
- ▷ Binance Coin (BNB);
- ▷ Bitcoin (BTC);
- ▷ Dash (DASH);
- ▷ Dogecoin (DOGE);
- ▷ Ethereum (ETH);
- ▷ Chainlink (LINK);
- ▷ Litecoin (LTC);
- ▷ Stellar (XLM);
- ▷ Ripple (XRP)

To strengthen the models, we add four sorts of features in our variables, which are simple moving averages, moving standard deviation, relative strength index and lags.

First, we will compare the forecast of prices with the forecast of cryptoasset returns to find out which method is more accurate and pursue our researches with it. Then, among our six Machine Learning models we will choose the three best fitting ones to build three different portfolios based on momentum investing strategies. Finally, we will compare the returns provided by these active investment approaches with buy-and-hold strategies. The summary outline below describes the whole process of our empirical analysis.

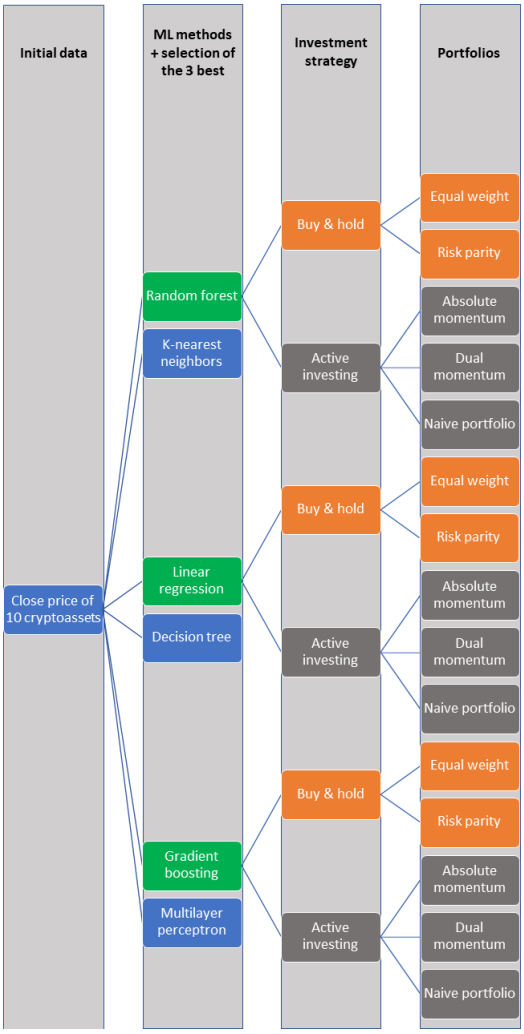

Figure 4: Summary outline

This analysis will be performed on Python and on Excel with the objective of answering this research question : Can we beat the market by using Machine Learning techniques in the cryptoasset world?

3.2 Methodology for price/return predictions

To accomplish this empirical analysis, we had two choices: directly predicting the returns (appendix 1.1) or making predictions of cryptoasset prices (appendix 1.2) and transform them into returns after. The former approach implies working with a stationary time series, as we can observe on the histogram of ADA returns in Figure 5, while the second deals with a non-stationary time series. Many studies only work in stationary conditions to avoid having biased coefficients, but in pure forecasting it can be appropriate to use non-stationary time series. We decided to perform both alternatives separately on different models and select the option that offers the highest accuracy for the continuation of our analysis. This section is carried out on Python.
For the sake of clarity, the appendices are attached on a separate document and the Python codes as well as the Excel files used in this master's thesis are made available in this **shared folder**.

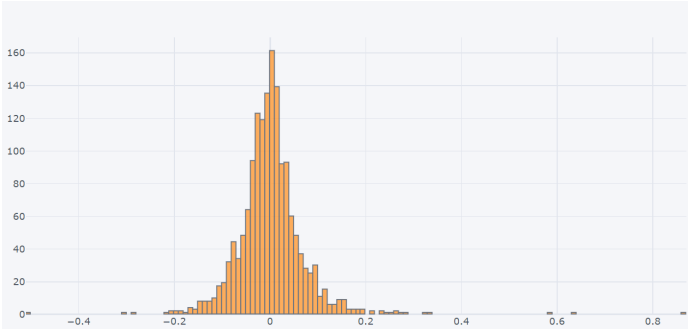

Figure 5: Histogram of ADA returns

3.2.1 Data import

To perform this study, we downloaded the historical data of the ten cryptoassets we are interested in (ADA, BNB, BTC, DASH, DOGE, ETH, LINK, LTC, XLM, XRP) on Yahoo Finance, and only the close values. These cryptocurrencies have been picked on the basis of their market capitalization, as they are all in the top 100, but also on their high longevity, to get a large database. The sample chosen goes from 9 November 2017 to 5 March 2022 composed of daily observations. Therefore, all periods specified in this empirical analysis will refer to days.

```
dataset = f"C:/Users/Lenovo/Downloads/Mémoire/Database/cut/Close/ADA.xlsx"
data = pd.read_excel(dataset, index_col=0, parse_dates=True).dropna()
```

Figure 6: Import the downloaded ADA database

3.2.2 Data transformation

Historical data are raw data that may need to be modified to be more operable, this is why we made some transformations.
After having loaded the datasets, we have to check if there are missing values and remove them if this is the case. To do this, we use the *dropna* fonction in Python which will drop the rows containing a missing value.

```
data.dropna(inplace=True)
```

Figure 7: Remove missing values

Furthermore, as we work in parallel with price and return data to see which method offers the best results in terms of predictions, the downloaded closing prices must be transformed into returns for the second method. Therefore, we transform these data into logarithmic returns with the formula below.

```
rets = np.log(data / data.shift(1)).dropna()  # log returns
```

Figure 8: Calculate the returns

3.2.3 Add features

In trading, it is not uncommon to use technical indicators to produce buy or sell signals based on patterns seen in the market. Consequently, to help our algorithms to predict, we add three types of features:

- ▷ Simple moving averages (SMA) of 14, 30 and 50 periods;
- ▷ Relative strength index (RSI) of 14, 30 and 50 periods;
- ▷ Moving standard deviation of 14 periods.

```
data['std14'] = data['ADA'].rolling(14).std()
feature_names = ['std14'] #empty list to hold the feature names.
for n in [14, 30, 50]:
    data['ma' + str(n)] = talib.SMA(data['ADA'].values, timeperiod=n)
    data['rsi' + str(n)] = talib.RSI(data['ADA'].values, timeperiod=n) #SMA and RSI methods to calculate the SMA and RSI: 14, 3
0, 50.

    feature_names = feature_names + ['ma' + str(n), 'rsi' + str(n)] #Add the ma and rsi variable names to the feature_names lis
t.
```

Figure 9: Add the features

3.2.4 Add lags

To take decisions in advance we must add lag variables in our models otherwise, we would have to wait the end of the day to get the close price which is too late to benefit from

the returns of the day. Therefore, we insert 7 lags to each variable. This number of lags has been chosen because it represents a trading week in the cryptocurrency market.

```
col_list = ['ADA', 'std14', 'ma14', 'rsi14', 'ma30', 'rsi30', 'ma50', 'rsi50']
for col in col_list:
    for x in range (1,8):
        df[col+f'__lag{x}'] = df[col].copy().shift(x)
```

Figure 10: Add the lags

3.2.5 Define independent variables

As for all regression models, we need to define independent variables on which our Machine Learning algorithms will rely on to make predictions. These predictors, represented by X in our models, gather all lagged variables of the dataset. In other words, all variables except the cryptoasset close prices, 'std14', 'ma14', 'rsi14', 'ma30', 'rsi30', 'ma50' and 'rsi50'.

```
X = data.drop(['ADA', 'std14', 'ma14', 'rsi14', 'ma30', 'rsi30', 'ma50', 'rsi50'],axis=1)
```

Figure 11: Define the independent variables

3.2.6 Define the dependent variable

The dependent variable corresponds to the variable we want to predict using the rest of the dataset. Therefore, this target Y is the cryptoasset price or return, depending on which model we work on.

```
Y = data['ADA']  #Extract the price column
```

Figure 12: Define the dependent variable

3.2.7 Split the data

Once the dependent and independent variables are defined, we need to split the data into two parts. The first one is the training set which will train the models by teaching them how to make predictions. The second portion is the test set, it will allow to test the models and compare the predictions with the actual values. For our analysis, we decided to use 90% of the data to train the models and 10% to test them. Given that we feed the algorithms with a very large part of the data, we can expect a relatively high accuracy.

```
X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size=0.1,shuffle = False,random_state=0)
```

Figure 13: Data splitting

3.2.8 Implement the different Machine Learning algorithms

After performing the preceding steps, we can finally implement the six different Machine Learning algorithms (random forest, linear regression, decision tree, gradient boosting regression, k-nearest neighbors and multilayer perceptron) to make predictions. Among these six models, we choose the three algorithms that best fit the data by relying on performance indicators (R^2, MAE, MSE, RMSE), and continue the analysis with them (appendix 1.3).

3.2.8.1 Random Forest

In the construction of the random forest model, we keep the default values of all parameters, except the number of trees, *n_estimators*, that we set to 100.

Random Forest

```
# Create Random Forest regressor
regressor = RandomForestRegressor(n_estimators=100)
```

```
# Train the model
regressor.fit(X_train,Y_train)
```

```
RandomForestRegressor()
```

```
test_data_prediction = regressor.predict(X_test)
```

Figure 14: Random forest implementation

3.2.8.2 Linear Regression

Concerning the linear regression model, we keep the default values of all parameters.

Linear Regression

```
from sklearn.linear_model import LinearRegression

# Creation of the Regression Model
model = LinearRegression()
```

```
# Train the model
model.fit(X_train, Y_train)
```

```
LinearRegression()
```

```
# Make predictions
y_pred_LR= model.predict(X_test)
```

Figure 15: Linear regression implementation

3.2.8.3 Decision Tree

For the decision tree algorithm, we use a random state equal to 0 which allows to get the same train and test sets across different executions. The other parameters remain by default.

Decision Tree

```
from sklearn.tree import DecisionTreeRegressor
# Create Decision Tree regressor
dtree_regressor = DecisionTreeRegressor(random_state=0)
```

```
# Train the model
dtree_regressor.fit(X_train,Y_train)
```

```
DecisionTreeRegressor(random_state=0)
```

```
# Make predictions
pred_dtree = dtree_regressor.predict(X_test)
```

Figure 16: Decision tree implementation

3.2.8.4 Gradient Boosting Regression

In this model, we use 100 as number of trees and keep the default values for the remaining parameters as well.

Gradient Boosting Regressor

```
from sklearn.ensemble import GradientBoostingRegressor
# Create Gradient Boosting Regressor
GB_regressor=GradientBoostingRegressor(n_estimators=100)
```

```
# Train the model
GB_regressor.fit(X_train,Y_train)
```

```
GradientBoostingRegressor()
```

```
# Make the predictions
GB_predict=GB_regressor.predict(X_test)
```

Figure 17: Gradient boosting regression implementation

3.2.8.5 K-Nearest Neighbors

For the k-nearest neighbors model, we do not make any change in the default values of parameters.

K Nearest Neighbors

```
from sklearn import neighbors
# Create K Nearest Neighbors regressor
knn = neighbors.KNeighborsRegressor()
```

```
# Train the model
knn.fit(X_train,Y_train)
```

```
KNeighborsRegressor()
```

```
# Make predictions
knn_pred = knn.predict(X_test)
```

Figure 18: K-nearest neighbors implementation

3.2.8.6 Multilayer Perceptron

In the multilayer perceptron model, we also keep the default values of all parameters.

Multilayer Perceptron

```
from sklearn.neural_network import MLPRegressor
from sklearn.datasets import make_regression
# Create MLP Regressor
MLP = MLPRegressor()
```

```
# Train the model
MLP.fit(X_train, Y_train)
```

```
MLPRegressor()
```

```
# Make predictions
MLP_pred = MLP.predict(X_test)
```

Figure 19: Multilayer perceptron implementation

3.2.9 Comparison and selection of the best algorithms

When all algorithms are implemented, we calculate the performance indicators introduced in the theory section (R^2, MAE, MSE, RMSE) for each of them and make a comparison. Indeed, we observe which algorithms have the highest R^2 and the lowest MAE, MSE, RMSE overall and select the 3 algorithms that best fit (appendices 1.4; 1.5).

```
#Performance indicators
r_score_RF=metrics.r2_score(Y_test,test_data_prediction)
print("R squared value_RF :",r_score_RF)
print("MAE_RF:", mean_absolute_error(Y_test, test_data_prediction))
meanSquaredError_RF=mean_squared_error(Y_test, test_data_prediction)
print("MSE_RF:", meanSquaredError_RF)
rootMeanSquaredError_RF = sqrt(meanSquaredError_RF)
print("RMSE_RF:", rootMeanSquaredError_RF)
```

Figure 20: Calculate the value of each indicator

3.3 Methodology for portfolio building in active investing strategies

Once we have our price predictions, we can calculate the expected returns and start the active investment strategies. We decided to implement three strategies inspired by the momentum investing strategies that argue for following the trend (appendix 1.6). Indeed, momentum investing approaches are theoretically constructed by relying on past performances but in our case, we will take decisions based on the expected returns computed while maintaining the core of the momentum strategies described in the section 2.1.2. The other adaptation is that we will just take long only decisions. Concerning the transaction fee, we consider a cost of 0.10% per transaction in our study, as this is the charge of the largest cryptocurrency platform, Binance.

3.3.1 Absolute Momentum

The first portfolio we build is based on absolute momentum approach. Indeed, for each period (day) we buy or keep the cryptoassets which have a positive expected returns and sell the others. Therefore, if no cryptoasset has a positive expected return at a given time, we do not invest during this period.

3.3.2 Dual Momentum

The next portfolio we construct is inspired by the dual momentum strategy. As a matter of fact, in each period, we buy the two cryptoassets having the highest expected returns but in contrast with the traditional method, we will not short the weakest assets. As a result, our portfolio is always composed of two cryptoassets.

3.3.3 Naive portfolio

Our last strategy is a naive portfolio that also consists in following the trend of the assets. We start with an equal weight portfolio (10% of each cryptoasset) and rebalance by buying 2% of the two cryptoassets with the highest expected returns, and selling 2% of the two cryptoassets with the lowest expected returns in every period. If at a given day the weights of the lowest assets are already at 0%, we therefore reduce the weights of the subsequent cryptoassets to compensate with the buying.

3.4 Methodology for buy-and-hold strategies

Concerning the buy-and-hold strategies, we constructed two portfolios: an equal weight portfolio and a risk parity portfolio. The development and computations of these passive investment strategies are performed on Excel (appendix 1.7).

3.4.1 Equal weight portfolio

The first buy-and-hold strategy we build is the equal weight portfolio. To set up this naive approach we simply allocate the same weight to each cryptoasset, given that we have 10 assets, each one has a weight of 10% in the portfolio (1/N).

3.4.2 Risk parity portfolio

The second buy-and-hold approach we consider in this thesis is risk parity. The allocation of the assets in this strategy is based on a risk-weighted basis to keep the volatility tolerable. To construct this strategy, we followed the method proposed by NEDL, 2021. Consequently, we started from the equally-weighted portfolio which has been optimized by the Excel to minimize the sum of deviations from parity, it involved the following steps:

1. We determined the covariance matrix;
2. We calculated the matrix product of the cryptoasset weights with the covariance matrix;
3. We computed the risk contribution of each cryptocurrency in the portfolio;
4. We used the Excel solver to calculate the weight of each cryptoasset.

On Figure 21, the objective is to minimize the sum of deviations from parity, the changing variable cells are the weights of each cryptoasset and the constraint is that the sum of all weights has to be equal to 1. For a better understanding of the selected cells in the solver, the Excel *cut-multi_database* is available **here**.

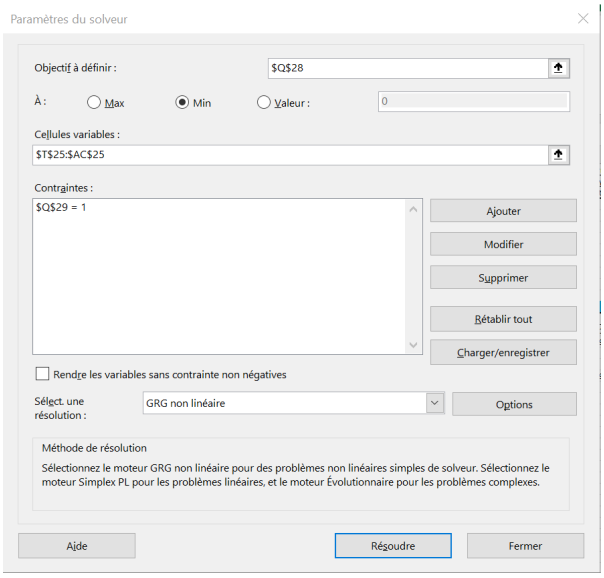

Figure 21: Calculate the weight of cryptoassets in risk parity with Excel solver

3.5 Compute returns and performance of the different portfolios

Once all our portfolios are constructed, we can calculate the global yields of each of them for the considered period (from 4 October 2021 to 5 March 2022). To that end, we compute the cumulative return of each strategy with the following formula.

$$\text{Cumulative return} \;=\; (1+R_1)(1+R_2)...(1+R_n)-1$$

Where :

R_i = The return during period i

n = The number of periods

We also calculate the average return and volatility of each portfolio to be able to compute the Sharpe ratio. This measurement permits to observe the performance of a portfolio in relation to its risk. The higher this ratio is, the more returns it offers.

In our study, as we do not have any risk-free rate, this parameter will be assumed equal to zero.

3.6 Analysis and results

Our first test was the comparison between predictions on prices and predictions on returns. We remark that even if the MAE, MSE and RMSE seem to provide good results, the R-squared indicator of models working with returns is negative for all algorithms, meaning that the models fit very poorly. As we can see on Figure 22, we reach to the same conclusion by looking at the plots depicting predicted values VS actual values.

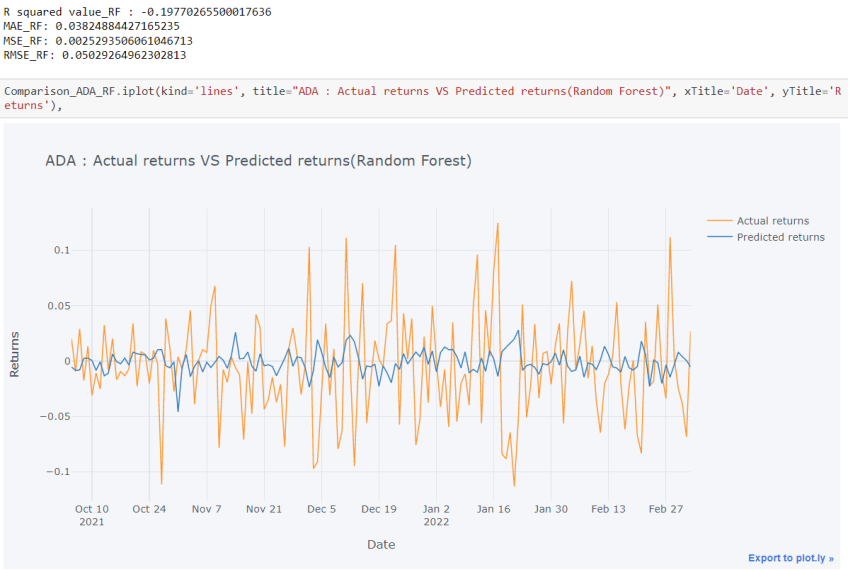

Figure 22: Performance of return predictions of ADA

In contrast, for the models forecasting prices, both performance indicators and plots give great results, that is why we decided to predict prices and then transform them in returns for our analysis.

Figure 23: Performance of price predictions of ADA

After having predicted the price with each model, the next step consisted in selecting the three algorithms that best fit the data. To achieve this, we relied on the performance indicators and compared the results of every cryptoasset. Based on performance indicators, we classified in Table 1 the three algorithms that are the most adapted for each cryptoasset for the period from the 04/10/2021 to the 05/03/2022. We noticed that all performance indicators give the same result which made it easier for us to select the most suitable models. Table 1 shows that the three algorithms that best fit are linear regression (LR), gradient boosting regression (GB) and random forest (RF) as they are the most present in the top 3. The reason of the great accuracy with gradient boosting regression and random forest can be explained by the fact that these algorithms only require minimal data pre-processing to be efficient thanks to their high flexibility. They do not need any specific transformation to be operable. Linear regression meanwhile, is very suitable for price predictions, that is why many studies apply this model when they forecast prices. Therefore, we use the predictions of these three algorithms to build our different portfolios.

Best fitting algorithms			
	Best fit	2nd best fit	3rd best fit
ADA	LR	GB	RF
BNB	LR	MLP	RF
BTC	LR	MLP	GB
DASH	LR	GB	RF
DOGE	LR	GB	DT
ETH	LR	MLP	RF
LINK	MLP	LR	RF
LTC	LR	GB	MLP
XLM	GB	LR	RF
XRP	LR	RF	GB

Table 1: Most adapted algorithms for each cryptoasset

We built our two buy-and-hold strategies, equal weight and risk parity, by relying on the predictions of the 3 selected algorithms and we notice that the returns are not profitable. Indeed, we can see in Table 2 that all the average returns are negative, meaning that we would lose money if we use these strategies to invest. We also notice that the random forest model forecast greater returns and Sharpe ratio for both strategies. Knowing that we assume having these portfolios already built at time 0, we do not have any transaction cost. Consequently, these results show that such strategies are not profitable in the cryptocurrency market in a short-term horizon. It is not really surprising as we know that passive investment approaches are supposed to be employed over the long run.

Results of buy-and-hold strategies				
	Equal weight		Risk parity	
	Return	Sharpe ratio	Return	Sharpe ratio
LR	-0.213%	-0.0613481	-0.205%	-0.0597916
GB	-0.212%	-0.072419	-0.2%	-0.0705117
RF	-0.192%	-0.0592003	-0.18%	-0.0578585

Table 2: Return and Sharpe ratio of buy-and-hold strategies

Concerning the active investment strategies, we observe quite different results in Table 3.

Results of active investing strategies						
	Absolute momentum		Dual momentum		Naive strategy	
	Return	Sharpe ratio	Return	Sharpe ratio	Return	Sharpe ratio
LR	1.969%	0.9501554	3.353%	0.822175	-0.382%	-0.1132034
GB	1.98%	1.0499765	3.743%	1.0469161	-0.407%	-0.1370858
RF	1.489%	0.7777537	3.79%	0.9344194	-0.431%	-0.120479

Table 3: Return and Sharpe ratio of active investing strategies

The absolute momentum and dual momentum portfolios are profitable as they provide respectively an average daily return of 1.813% and 3.629%. Because of the high number of transactions at every period, the absolute momentum portfolio is the strategy that is the most penalized by the transaction costs. It is not really surprising to observe that the dual momentum portfolio provides the best performance as we only select the two highest expected returns therefore, the return is supposed to be maximized. In addition, there is at most four transactions per period, which is lower than the two other active investing strategy. The drawback of this portfolio is the small diversification it induces. If we look at the Sharpe ratio, we remark that the values are very close between the absolute momentum and the dual momentum portfolios even if the returns of the second one are almost doubled compared to the first one, meaning that the volatility of the absolute momentum strategy is nearly two times lower. Regarding the different algorithms, gradient boosting is the one that foresees the highest returns as well as the highest Sharpe ratio for the absolute momentum portfolio. For the dual momentum strategy, there are two different algorithms that predict the highest return and Sharpe ratio. Indeed, the random forest algorithm gets the greatest average returns while the gradient boosting regression model has the highest Sharpe ratio which means that the random forest model forecasts more volatility than the gradient boosting regression.

In contrast, the naive portfolio strategy, in which we increased the weight of the 2 highest predicted returns by 2% and reduced the weight of the 2 lowest predicted returns by 2%,

has negative average returns. It can be partly explained by the transaction costs of 0.4% at each period, except at the first day. The other reason may be that a rebalancing of 2% is not enough to take advantage of the trend. Consequently, knowing that the buy-and-hold strategies provide negative returns, it was foreseeable to also observe negative returns for this naive strategy as there are relatively low changes at each period and we are penalized by 0.4% in transaction costs every day. The linear regression algorithm provides the most optimistic results for the naive strategy as the return and the Sharpe ratio are higher in comparison with the other algorithms.

If we take a look at the cumulative return of each approach, we remark on Table 4 that the dual momentum portfolio offers by far the highest profit, especially when we use random forest, with a cumulative return of 26358.47%, transaction costs included, in about five months. This cumulative return seems quite high, but we know that prices are very volatile in cryptoassets so it remains achievable, especially with dual momentum strategy as we take advantage of the two highest upward trends each day. We also notice an important cumulative return gap between linear regression and the two other algorithms that are almost two times higher. Regarding the absolute momentum approach, we observe the same difference with the random forest algorithm providing more than two times lower cumulative return than the others. However, this portfolio remains quite profitable. In contrast, as seen earlier the naive portfolio offers very poor results. If we used this strategy from the 4 October 2021 to the 5 March 2022, we would have lost about half of our investment which is considerable. The poor returns for the naive strategy can be explained by the fact that at each period (except the first day) four transactions are made which represent daily transaction costs of 0.4% to benefit from only 2% more of the two highest expected returns. These ones can also be negative with this strategy, therefore, the transactions may not always be profitable.

Cumulative return of active investing strategies

	Absolute momentum	Dual momentum	Naive strategy
LR	1815.97%	13707.71%	-49.03%
GB	1856.36%	25213.55%	-50%
RF	835.2%	26358.47%	-53.3%

Table 4: Cumulative return of active investing strategies

The results of this empirical analysis show that it is possible to outperform buy-and-hold approaches in the cryptocurrency market with machine learning based strategies. In fact, our absolute momentum and dual momentum portfolios not only beat equal weight and risk parity strategies which offer negative returns, but they also provide profitable returns. The other finding in this analysis is that none of the three selected algorithms

(linear regression, gradient boosting and random forest) really outperforms the others, each one offers a higher return or Sharpe ratio for one or the other strategy.

4 Conclusion & Limits

In this last section, we will finally conclude this thesis with our key findings during this empirical analysis. We will compare our results with those found in the literature review and answer our research question. Ultimately, we will describe some limits to our work and try to find ways to go deeper in this area for future researches.

In this study, we show that predictions on a non-stationary time series (price data) can be more accurate than predictions on a stationary one (return data) which is in accordance with the paper of Mackay and Jonathan, 2020.

We also prove that linear regression gives the most precise price predictions for cryptoassets which is consistent with the results of Reddy and Sriramya, 2020. It is followed by gradient boosting regression and random forest that also provide accurate forecasts.

Thanks to this analysis, we can now answer our research question : "Can we beat the market by using Machine Learning techniques in the cryptoasset world?". We demonstrated that it is possible to outperform buy-and-hold approaches with active investing strategies,even by including transaction costs. Indeed, we showed that dual momentum offers by far the highest returns. However, absolute momentum also provides positive and greater returns than buy-and-hold approaches and Sharpe ratios close to the dual momentum strategy. The only portfolio underperforming equal weight and risk parity strategies is the naive portfolio in which we increased by 2% the weight of the two cryptoassets with the highest expected return and decreased by 2% the weight of the two cryptoassets with the lowest expected return at each period, starting from an equal weight portfolio. It delivers negative returns and is thus not profitable.

4.1 Limits

An obvious limit in our empirical analysis is the limited number of trees or number of estimators we can set in our models due to the small power of our computer. Increasing the number of estimators using a more powerful system would allow a higher accuracy which can be relevant to improve our testing.

The global conditions during the period we focused on to make our analysis is quite turbulent with a climate of financial uncertainty party due to the coronavirus crisis. On the political level, the situation is even more hectic because of the conflict between Ukraine and Russia. These two countries being major actors in the cryptocurrency market, the asset prices may have been affected, which may have made the prediction more difficult to perform, and thus reduced its accuracy. In addition, the period we based our predictions on can be considered neither purely in bull market nor in bear market. Therefore, in

future researches it could be interesting to operate a similar analysis during one of these two situations to observe if we can still beat the market and reach to the same results.

5 References

Akyildirim, E., Goncu, A., & Sensoy, A. (2020). Prediction of cryptocurrency returns using machine learning. *Annals of Operations Research*, *297*, 3–36. https://doi.org/https://doi.org/10.1007/s10479-020-03575-y

Anderson, R. M., Bianchi, S. W., & Goldberg, L. R. (2012). Will my risk parity strategy outperform? *Financial Analysts Journal*, *68*(6), 75–93. https://doi.org/https://doi.org/10.2469/faj.v68.n6.7

Antonacci, G. (2013). Absolute momentum: A simple rule-based strategy and universal trend-following overlay. *SSRN Electronic Journal*. https://doi.org/http://dx.doi.org/10.2139/ssrn.2244633

Aziz, S., Dowling, M., Hammami, H., & Piepenbrink, A. (2021). Machine learning in finance: A topic modeling approach. *European Financial Management*, *28*(3), 744–770. https://doi.org/https://doi.org/10.1111/eufm.12326

Babiak, M., & Bianchi, D. (2022). On the performance of cryptocurrency funds. *Journal of Banking Finance*, *138*. https://doi.org/https://doi.org/10.1016/j.jbankfin.2022.106467

Bastien, L. (2021). *Machine learning: What is it and why does it change the world?* Retrieved 2022, from https://datascientest.com/en/machine-learning-what-is-it-and-why-does-it-change-the-world

Belkacem, S. (2021). *Machine learning approaches to rank news feed updates on social media* (Doctoral dissertation).

CFI Team. (2022). *Machine learning (in finance).* Retrieved 2022, from https://corporatefinanceinstitute.com/resources/knowledge/other/machine-learning-in-finance

Chai, T., & Draxler, R. R. (2014). Root mean square error (rmse) or mean absolute error (mae)? *Geoscientific Model Development*, *7*(3), 1525–1534. https://doi.org/https://doi.org/10.5194/gmd-7-1247-2014

Chicco, D., Warrens, M. J., & Jurman, G. (2021). The coefficient of determination r-squared is more informative than smape, mae, mape, mse and rmse in regression analysis evaluation. *PeerJ Computer Science*, *7*(e623). https://doi.org/https://doi.org/10.7717/peerj-cs.623

Dancho, M. (2018). *Tidy time series analysis, part 4: Lags and autocorrelation.* Retrieved 2022, from https://www.business-science.io/timeseries-analysis/2017/08/30/tidy-timeseries-analysis-pt-4.html

DeMiguel, V., Garlappi, L., & Uppal, R. (2009). Optimal versus naive diversification: How inefficient is the 1/n portfolio strategy? *Review of Financial Studies*, *22*(5), 1915–1953. https://doi.org/https://doi.org/10.1093/rfs/hhm075

D'Souza, I., Srichanachaichok, V., Wang, G. J., & Yao, Y. (2016). The enduring effect of time-series momentum on stock returns over nearly 100-years. *SSRN Electronic Journal*. https://doi.org/https://dx.doi.org/10.2139/ssrn.2720600

El Naqa, I., & Murphy, M. J. (2015). Machine learning in radiation oncology - what is machine learning?, 3–11. https://doi.org/https://doi.org/10.1007/978-3-319-18305-3_1

Fernando, J. (2022). *Relative strength index (rsi)*. Retrieved 2022, from https://www.investopedia.com/terms/r/rsi.asp#:~:text=The%5C%20Relative%5C%20Strength%5C%20Index%5C%20(RSI)%5C%20is%5C%20a%5C%20measurement%5C%20used%5C%20by,scale%5C%20of%5C%200%5C%20to%5C%20100.

Frost, J. (2022). *Mean squared error (mse)*. Retrieved 2022, from https://statisticsbyjim.com/regression/mean-squared-error-mse/

Glen, S. (2022). *Coefficient of determination (r squared): Definition, calculation*. Retrieved 2022, from https://www.statisticshowto.com/probability-and-statistics/coefficient-of-determination-r-squared/

Gu, S., Kelly, B. T., & Xiu, D. (2018). Empirical asset pricing via machine learning. *Working Paper*. https://doi.org/10.3386/w25398

Gurucharan, M. K. (2020). *Machine learning basics: Decision tree regression*. Retrieved 2022, from https://towardsdatascience.com/machine-learning-basics-decision-tree-regression-1d73ea003fda

Hassan, M., Negm, A., Zahran, M., & Saavedra, O. (2015). Assessment of artificial neural network for bathymetry estimation using high resolution satellite imagery in shallow lakes: Case study el burullus lake. *International Water Technology Journal*, *5*(4).

Henrique, B. M., Kimura, H., & Sobreiro, V. A. (2019). Machine learning techniques applied to financial market prediction. *Expert Systems with Applications*, *124*, 226–251. https://doi.org/https://doi.org/10.1016/j.eswa.2019.01.012

Hilpisch, Y. (2020). *Artificial intelligence in finance: A python-based guide*. O'Reilly Media.

Hu, C., Jain, G., Zhang, P., Schmidt, C., Gomadam, P., & Gorka, T. (2014). Data-driven method based on particle swarm optimization and k-nearest neighbor regression for estimating capacity of lithium-ion battery. *Applied Energy*, *129*, 49–55. https://doi.org/10.1016/j.apenergy.2014.04.077

Iania, L. (2021). *Investments: Performance measurement (slides)* [Unpublished], Université catholique de Louvain.

Jaokar, A. (2019). *The mathematics of data science: Understanding the foundations of deep learning through linear regression*. Retrieved 2022, from https://www.

datasciencecentral.com/the-mathematics-of-data-science-understanding-the-foundations-of

Jason Fernando. (2022). *Sharpe ratio*. Retrieved 2022, from https://www.investopedia.com/terms/s/sharperatio.asp

Jones, R. C., & Rasekhschaffe, K. C. (2019). Machine learning for stock selection. *Financial Analysts Journal*, *75*(3), 70–88. https://doi.org/https://doi.org/10.1080/0015198X.2019.1596678

Kaczmarek, T., & Perez, K. (2021). Building portfolios based on machine learning predictions. *Economic Research-Ekonomska Istraživanja*, *35*(1), 19–37. https://doi.org/https://doi.org/10.1080/1331677X.2021.1875865

Keane, S. M. (1986). The efficient market hypothesis on trial. *The Journal of Alternative Investments*, *42*(2), 58–63. https://doi.org/https://doi.org/10.2469/faj.v42.n2.58

Koker, T. E., & Koutmos, D. (2020). Cryptocurrency trading using machine learning. *Journal of Risk and Financial Management*, *13*(8), 1–7. https://doi.org/https://doi.org/10.3390/jrfm13080178

Lam, P. N., & Lee, D. K. C. (2015). Introduction to bitcoin. *Handbook of Digital Currency*, 5–30. https://doi.org/https://doi.org/10.1016/B978-0-12-802117-0.00001-1

Lee, D. K. C., Guo, L., & Wang, Y. (2017). Cryptocurrency: A new investment opportunity? *The Journal of Alternative Investments*, *20*(3), 16–40. https://doi.org/https://doi.org/10.3905/jai.2018.20.3.016

Liu, W., Strong, N., & Xu, X. (1999). The profitability of momentum investing. *Journal of Business Finance Accounting*, *26*(9-10), 1043–1091. https://doi.org/https://doi.org/10.1111/1468-5957.00286

López de Prado, M. (2016). Building diversified portfolios that outperform out of sample. *The Journal of Portfolio Management*, *42*(4), 59–69. https://doi.org/https://doi.org/10.3905/jpm.2016.42.4.059

Mackay, E., & Jonathan, P. (2020). Assessment of return value estimates from stationary and non-stationary extreme value models. *Ocean Engineering*, *207*, 107406–. https://doi.org/https://doi.org/10.1016/j.oceaneng.2020.107406

Malkiel, B. G. (2003). Passive investment strategies and efficient markets. *European Financial Management*, *9*(1), 1–10. https://doi.org/https://doi.org/10.1111/1468-036X.00205

Masui, T. (2020). *All you need to know about gradient boosting algorithm*. Retrieved 2022, from https://towardsdatascience.com/all-you-need-to-know-about-gradient-boosting-algorithm-part-1-regression-2520a34a502

Maulud, D., & Abdulazeez, A. M. (2020). A review on linear regression comprehensive in machine learning. *Journal of Applied Science and Technology Trends*, *1*(4), 140–147. https://doi.org/https://doi.org/10.38094/jastt1457

Mohanty, A. (2019). *The mathematics of data science: Understanding the foundations of deep learning through linear regression*. Retrieved 2022, from https://becominghuman.ai/multi-layer-perceptron-mlp-models-on-real-world-banking-data-f6dd3d7e998f

Nakamoto, S. (2008). *Bitcoin: A peer-to-peer electronic cash system*. https://bitcoin.org/bitcoin.pdf

NEDL. (2021). Risk parity portfolio explained: Risk contributions of asset classes (excel). https://www.youtube.com/watch?v=MnUERF8DoUY&ab_channel=NEDL

Reddy, L. S., & Sriramya, P. (2020). A research on bitcoin price prediction using machine learning algorithms. *International Journal of Scientific Technology Research*, *9*(4).

Ren, Y., Zhang, L., & Suganthan, P. (2016). Ensemble classification and regression-recent developments, applications and future directions. *IEEE Computational Intelligence Magazine*, *11*(1), 41–53. https://doi.org/10.1109/MCI.2015.2471235

Ryou, H., Bae, H. H., Lee, H. S., & Oh, K. J. (2020). Momentum investment strategy using a hidden markov model. *Sustainability*, *12*(17), 7031–. https://doi.org/https://doi.org/10.3390/su12177031

Saini, A. (2021). *Gradient boosting algorithm: A complete guide for beginners*. Retrieved 2022, from https://www.analyticsvidhya.com/blog/2021/09/gradient-boosting-algorithm-a-complete-guide-for-beginners/

Schlagenhauf, T. (2022). *Decision trees in python*. Retrieved 2022, from https://python-course.eu/machine-learning/decision-trees-in-python.php

Shah, V. H. (2007). *Machine learning techniques for stock prediction*. Retrieved 2022, from https://bigquant.com/community/uploads/default/original/1X/5c6d3b9959a8556a533a58e0ac4568dfc63d6ff4.pdf

Skabar, A., & Cloete, I. (2002). Neural networks, financial trading and the efficient markets hypothesis. *Australasian Computer Science Conference*, *2*, 241–249. https://doi.org/10.1145/563857.563829

Solactive. (2018). *When size doesn't matter: Equal weighting (vs. market cap weighting)*. Retrieved 2022, from https://www.solactive.com/wp-content/uploads/2018/04/Solactive-Equal-Weighting-vs.-Market-Cap-Weighting.pdf

Statman, M. (1987). How many stocks make a diversified portfolio? *Journal of Financial and Quantitative Analysis*, *22*(3), 353–363.

Valpola, H. (2000). *Multi-layer perceptron networks*. Retrieved 2022, from http://users.ics.aalto.fi/harri/thesis/valpola_thesis/node43.html

Wolff, D., & Neugebauer, U. (2019). Tree-based machine learning approaches for equity market predictions. *Journal of Asset Management*, *20*, 273–288. https://doi.org/https://doi.org/10.1057/s41260-019-00125-5

Ying, X. (2019). An overview of overfitting and its solutions. *Journal of Physics: Conference Series*, *1168*(2). https://doi.org/10.1088/1742-6596/1168/2/022022

Zaner. (2022). *Moving standard deviation (mstd)*. Retrieved 2022, from https://www.zaner.com/3.0/education/technicalstudies/MSD.asp