Acknowledgements

While researching, studying, and writing this work, I received a great deal of support and assistance.

I would first like to thank my supervisor, Professor Francesco Grossetti, for having a deep belief in my work since the first moment I mentioned it and for supporting me with constructive and invaluable advice.

I also want to thank my family, who encouraged me during these years of university, and without whom I would not have been able to complete my studies. Special thanks to my brother Joseph for always feeding my interest in cryptocurrencies and blockchain, without which I would not have ventured into this work.

Finally, I thank my colleagues and friends at DSBA, CLEAM, and Spadolini, without whom the past five years would not have been the same. I thank you for all the days spent in the library, the dinners in apartment 77, and the countless video calls.

Not forgetting the two friends who have been there since day one, Niccolò and Giovanni, for inspiring each other and for encouraging and supporting me whenever I needed them.

Abstract

This thesis aims to test an innovative and untested approach to classify the price of Bitcoin in a way that can later be used on algorithmic trading scenarios. A binary classification approach is preferred, instead of the more widely applied regression approach, to ease the prediction task. This choice is made in accordance with the conceived distribution of the proposed model, which is to be employed within a trading algorithm. The Bitcoin Financial Time Series task is approached similarly to a computer vision problem, leveraging a 2D Convolutional Neural Network. The data include not only the Bitcoin price data, but also Bitcoin on-chain features and metrics. On this data technical indicators are measured to increase the number of time series, which are made stationary by fractional differentiation. The time series are transformed into images using recurrence plots, and used to train the 2D Convolutional Neural Network as a standard image classification problem. On the test set, the proposed model reaches a Macro F1-score of 58%. According to a Financial Evaluation, the returns of the proposed model based algorithm exceed the bitcoin Buy-and-Hold by 7.88% with a 38.95% increase in the sharpe ratio. The performance of the proposed model are also compared with a k-Nearest Neighbors (k-NN) and a Random Forest (RF) classifier. It is interesting how the k-NN, despite being a very simple and intuitive model, was able to outperform the proposed proposed model. In the test set, it achieved a Macro F1-score of 63%. In terms of Financial Evaluation, it also achieved the best results, outperforming the bitcoin Buy-and-Hold by 75.69%, with a sharpe ratio increase of 50.86%.

Contents

Chapter 1

Introduction

1.1 Bitcoin: A Peer-to-Peer Electronic Cash System

On October 31, 2008, Satoshi Nakamoto announced the design of Bitcoin in a mailing list for cryptographers, known as "metzdowd". In the Bitcoin design paper, called "Bitcoin: A Peer-to-Peer Electronic Cash System", there is the first detailed description of the Bitcoin protocol [21]. Bitcoin was conceived as a peer-to-peer payment system based on cryptographic proofs and in which transactions are recorded within a particular distributed database: the Blockchain. The term Blockchain derives from the structure of this database, consisting of a chain of blocks containing transactions. The data recorded on the blockchain is made public and cannot be modified without the consent of the majority of nodes [1]. This simple but revolutionary mechanism has allowed the creation of the first digital currency able to solve the problem of double spending without the need of an intermediary [34]. The first version of Bitcoin software v0.1 was released on January 9, 2009, by Satoshi Nakamoto. In over 12 years since the first release, Bitcoin has established itself as the first cryptocurrency,

[1] This is because the submission of transactions within the blockchain relies on a consensus mechanism known as Proof-of-Work (PoW). PoW requires nodes to solve a mathematical problem before proposing the submission of transactions into the blockchain. Solving this problem is called "mining", and only the first node that manages to solve it is rewarded with new bitcoins.

reaching on April 13, 2021, its highest value valued at over $63,500 per bitcoin with a total capitalization of approximately $1,186B. Its performance has attracted the attention of not only small investors, but also institutional investors and governments. At the moment it is no longer considered as a mere peer-to-peer electronic payment system, mainly due to its high volatility which makes it prohibitively expensive to use it as a medium of exchange and a unit of account. However, over very long periods, it assumes the characteristics of a store of value[2] and this is supported by the design of the protocol which makes it a deflationary asset. The excess volatility at first glance may seem incompatible with the characteristics of a store of value, however if this volatility is high only in the short term, but more stable in the long term, then it may not be a major issue for Bitcoin. On the contrary, it could be a positive factor that suggests a price search, until that asset reaches maturity. Moreover, due to inflation, FIAT [3] currencies themselves are not risk-free and in recent years have depreciated against real assets, such as gold. Consequently, they have not acted as a good store of value, contrary to what Bitcoin has done [32]. For these reasons Bitcoin has earned the name of "Best Performing Asset Of The Decade", with an annualized return of 230% thus outperforming the Nasdaq 100 by 10 times. As a result, investors are increasingly interested in predicting the price. From this point of view, the excess of volatility is both a negative factor, as regards the impossibility of being used as a medium of exchange, and a positive one as regards investment opportunities and its hedging capabilities [10]. This volatility implies a need to be able to make robust predictions, and at the moment research on such models is still an open debate.

1.2 Financial Time Series Forecasting

Financial Time Series Forecasting is considered one of the most challenging problems in the domain of modern time-series forecasting, in particular because these data are inherently

[2]A store of value is defined as a asset, commodity or currency that is able to be exchanged in the future without deteriorating in value. Basically it should be worth the same or more.

[3]FIAT currencies are government issued currencies not backed by a commodity. Their value is only defined by demand and supply and the stability of the issuing government.

noisy, non-stationary and deterministically chaotic [4] [3, 15, 37]. This problem is relevant both from a theoretical and an applied point of view, notably because of the obvious non-linearity of data [37]. In recent years, a number of researches based on Machine Learning models have been published consistently outperforming classical econometric models [24]. More recently, Deep Learning models gained further popularity, emerging as the best performing models [24]. The majority of studies aim at solving regression problems on financial time series, on which the most widely used models turn out to be Recurrent Neural Network (RRN), Convolutional Neural Network (CNN) and Multi Layer Perceptron (MLP) [14, 22, 23, 30]. As of 2018, the most studied topics concern stock price forecasting, trend forecasting, index forecasting and currently cryptocurrency price prediction [24], of which the latter is apparently growing in popularity.

1.3 Problem Definition

With this work, we consider our Bitcoin Financial Time Series task as a binary classification problem. Given our observed price series of length t, our goal is to classify the future value y_{t+1} as either a *buying* or *selling* signal. The y_t^{buy} correspond to the y values in which it is convenient to buy the asset at time t, so we expect at time $t+1, t+2, ...$ a price rise; on the contrary the y_t^{sell} correspond to the y_t values in which it is convenient to sell the asset.

1.4 Motivation

Cryptocurrency combined with Financial Time Series Forecasting is still a very novel and unexplored field. The goal of this thesis is to test an innovative and untested approach to classify the price of Bitcoin in a way that can later be used on algorithmic trading scenarios. A binary classification approach is preferred, instead of the more widely applied regression approach, to ease the prediction task. This choice is made in accordance with the conceived distribution of the proposed model, which is to be employed within a trading algorithm. For

[4] Deterministic chaos is defined as the appearance of irregular chaotic motion in purely deterministic dynamical systems [3].

this purpose, it is sufficient that the model is able to classify the days when it is convenient to buy or sell bitcoin, compared to forecasting its continuous price values.

1.5 Organization of Thesis

The thesis is divided into four sections: the Literature Review presents previous studies regarding Financial Time Series Forecasting on Bitcoin. The Research Design chapter describes the data used to train 2D-CNN, the methodology used to create labels for binary classification, the procedures used to transform the collected data and the structure of the proposed model. In the chapter on the evaluation we highlight the results obtained at the completion of the training.

Chapter 2

Literature Review

2.1 Stock Price Forecasting and Classification

In the literature there are several studies on stock price forecasting which can be categorized into three distinct clusters [31]. The first cluster is dedicated to models that use bivariate or multivariate regression but because of the linearity assumption they fell short with less accurate results [31]. The second cluster uses econometric models, such as Autoregressive Integrated Moving Average (ARIMA) [4], Granger Causality Test and Generalized Autoregressive Conditional Heteroscedasticity (GARCH) [11]. These models are very useful as they have explanatory power, but are limited by their own assumptions, which often do not hold in the real-world cases, particularly with high dimensional data [27]. The third cluster gathers methods based on machine learning and deep learning that include the most modern and best performing models [31]. Nelson et al. (2017) used a Long Short-Term Memory (LSTM) [1] neural network with multiple stock analysis indicators to predict the rise or fall of the stock price via a classification task. To determine if there are any significant improvements between their model and the chosen baselines, they performed the Kruskal-Wallis non-parametric test. In general, the proposed model outperforms the baselines and

[1]LSTM is an Recurrent Neural Network (RNN) used in deep learning, that is able to process entire sequences of data. LSTM are usually used for speech recognition, natural language processing and anomaly detection in network traffic.

in the analyzed stocks it achieves an F1-score of at most 0.431 [23]. Convolutional Neural Networks (CNNs) have dominated the computer vision field in recent years, as they are able to find patterns within images (or data) in a translational invariant manner. Chen et al. (2017) used a CNN in combination with Gramian Angular Fields (GAFs), moving average mapping, and double moving average mapping, to transform stock price time series into images and eliminate noise, proving an increase in accuracy [7]. Kim, T. and Kim, H.Y. (2019) proposed an LSTM-CNN model based on a combination of stock time series and stock trend graphs and showed that this model outperformed the models taken individually [14]. Sezer and Ozbayoglu (2018) trained a CNN by transforming technical indicators of stock prices into a 2D image to be used in an algorithmic trading system. According to the results they were able to achieve accuracy between 58% and 62%, as well as perform better against Buy-and-Hold on the out-of-sample test periods [30].

2.2 Bitcoin Time Series Forecasting and Classification

Although relatively new, the number of studies on cryptocurrency price forecasting is growing, making it the fourth most studied topic in 2018 based on rate of publication counts in financial time series forecasting with deep learning field [24]. Mudassir et al. (2020) trained both regression and classification models to predict the price of Bitcoin and to determine the differences between these models. The trained models include Artificial Neural Network (ANN), Stacked Artificial Neural Network (SANN), Support Vector Machines (SVM), and LSTM. Apart from the usual Open, High, Low, Close (OHLC) [2] price data of Bitcoin, they have also used other data from the blockchain as input, such as blocksize, difficulty, average transaction value, hashrate, etc. Moreover, to increase the number of features, they measured various technical indicators: Simple Moving Average (SMA), Exponential Moving Average (EMA), Relative Strength Index (RSI), Weighted Moving Average (WMA), Stan-

[2]OHLC data is a type data used to describe the price of financial instrument over time. Open and High are respectively the prices of the financial instrument when it began and ended trading in a given time period, while High and Low are the maximum and minimum prices in that same time period. These price data are usually illustrated through candle sticks.

dard Deviation (STD), Variance (VAR), Triple Moving Exponential (TRIX) and Rate of Change (ROC). These technical indicators have been calculated on different periods: raw closing price, 7, 30 and 90 days. According to the results obtained, the best classification model reached an accuracy of 65% for the next-day forecast, while for the daily forecast they reached a MAPE of 1.44% [19]. Mallqui and Fernandes (2019) analyzed the Bitcoin price using ANN and SVM for both a classification task and a regression task. The data used consist of the daily OHLC values and volume of Bitcoin, some blockchain features and some external indicators such as crude oil, gold future prices, S&P500 future. According to their analysis, as a single model, SVM achieved the best performance in both the classification task, with 59.45% accuracy, and the regression task, with a MAPE between 1.52-1.58%. The best results were achieved with an ensemble of RNNs and a tree classifier model, capable of achieving 62.9% accuracy [9]. McNally et al. (2018) implemented a Recurrent Neural Network (RNN) and Long Short Term Memory (LSTM) to predict the price of Bitcoin as a classification task (price up and price down). The data used includes the OHLC price of Bitcoin, collected from five major Bitcoin exchanges and two blockchain features: hash rate and mining difficulty. Additionally, two simple moving averages (SMA) and a de-noised closing price were also included. With both models, they were able to achieve 51-52% accuracy, thus a marginal improvement over the odds of binary classification [18]. Borges and Neves (2020) tested four different machine learning models to predict the price of Bitcoin and find the best entry (buy) and exit (sell) points to gain advantage over the market through a classification approach. The models used are: Logistic Regression (LR), SVM, Random Forest (RF) Decision Tree Gradient Boosting, which in conclusion were combined reaching an accuracy of 59.3% [5]. Greaves and Au (2015) attempted to predict Bitcoin price increases and decreases in the next hour using LR, SVM, Neural Network (NN) and a Baseline (percentage of the average price increase). Besides the Bitcoin price, among the inputs were also introduced some features derived from the blockchain, such as “mined bitcoin in the last hour” or “number of transactions made by new addresses in a given hour”. In this way they were able to achieve 55.1% accuracy with the best model, the Neural Network [13]. Li et al. (2020) used an

attentive LSTM combined with an Embedding Network, renamed ALEN, to predict Bitcoin price fluctuations through a classification approach. The attentive LSTM is an improvement of the LSTM that achieves outstanding performance in time series prediction. In the study it was used to capture the time dependency of Bitcoin price, while the Embedding Network to capture the hidden representations from related cryptocurrencies. Three different types of features were used to train ALEN: basic features, traditional technical trading indicators, and features generated by Denoising Autoencoders (DAEs). According to the results, ALEN was able to achieve a 61.15% accuracy and a 61.01% F1-score [38].

Chapter 3

Research Design

The Bitcoin Financial Time Series task is approached similarly to a computer vision problem, leveraging a 2D Convolutional Neural Network. The data include not only the Open, High, Low, Close (OHLC) values of Bitcoin, but also features coming directly from its blockchain, i.e. on-chain. On this data technical indicators is measured to increase the number of time series, which are made stationary by fractional differentiation. The time series are transformed into images using recurrence plots, and used to train the 2D Convolutional Neural Network as a standard image classification problem.

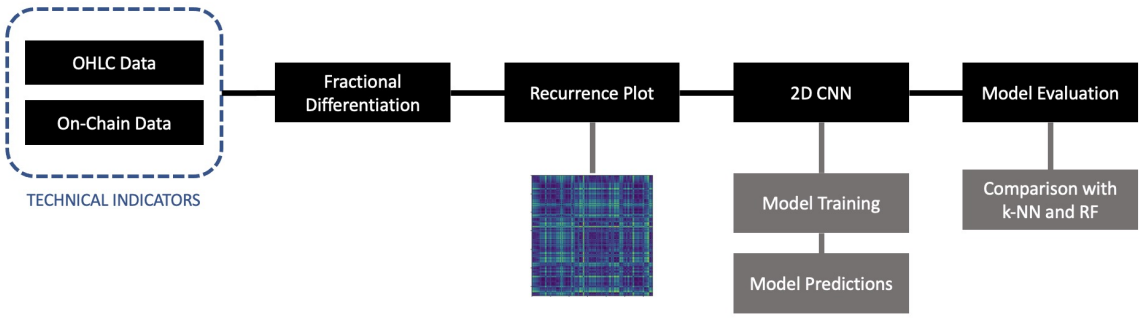

Fig. 3.1 Methodology pipeline

3.1 Bitcoin Data and On-Chain Metrics

In the present study we use daily price data of Bitcoin, in the OHLC format, as well as various metrics and features derived from the Bitcoin blockchain. All data is collected directly on *Glassnode.com*. Table 3.1 and 3.2 contain the list of Bitcoin on-chain features and metrics used, respectively. The data is then structured into matrix $\mathbf{X}$ of features, where each row corresponds to a single day.

Bitcoin on-chain Feature	**Description**
Active Addresses	Number of unique addresses that were active in the network
Addresses with Non-Zero Balance	Number of unique addresses holding a positive (non-zero) amount of bitcoins
Addresses with Balance $\geq$ 1,000 bitcoin	Number of unique addresses holding at least 1,000 bitcoin
Addresses with Balance $\geq$ 10,000 bitcoin	Number of unique addresses holding at least 10,000 bitcoin
Balance on Exchanges (percent)	Percent supply held on exchange addresses
Exchange Net Flow Volume	Net flow of bitcoins into/out of exchanges
Percent of Supply Last Active 1+ Years Ago	Percent of circulating supply that has not moved in at least 1 year

Table 3.1 Bitcoin on-chain features list

The following is a description of on-chain metrics, to understand their interest with respect to Bitcoin price and possible predictive capabilities.

Fee Ratio Multiple (FRM)

The Fee Ratio Multiple (FRM) is a metric introduced by Matthew Leibowitz (2018) to determine the security level of a Proof-of-Work blockchain [17]. FRM is measured as:

$$FRM = \frac{Block\ Reward + Transaction\ Fee}{Transaction\ Fee}$$

basically, the FRM is able to assess how secure a chain is once block rewards disappear, or how much a chain is able to sustain its level of security solely through transaction fees. A

Bitcoin on-chain Metric	Description
Fee Ratio Multiple (FRM)	ratio between the total miner revenue (blocks rewards + transaction fees) and transaction fees
Thermocap	aggregated amount of bitcoins paid to miners
Coin Days Destroyed (CDD)	number of bitcoins in a transaction multiplied by the number of days it has been since those bitcoins were last spent
Cumulative Value-Days Destroyed (CVDD)	ratio of the cumulative USD value of Coin Days Destroyed and the market age (in days)
Dormancy	ratio of coin days destroyed and total transfer volume
Entity-adjusted Dormancy Flow	ratio of the current market capitalization and the annualized dormancy value (measured in USD)
Liveliness	ratio of the sum of Coin Days Destroyed and the sum of all coin days ever created
Realized HODL (RHODL) Ratio	ratio between the 1 week and the 1-2 years RCap HODL bands
Net Realized Profit/Loss Ratio	net profit or loss of all moved coins
Difficulty Ribbon Compression	uses a normalized standard deviation to quantify compression of the Difficulty Ribbon
Adjusted Spent Output Profit Ratio (aSOPR)	SOPR ignoring all outputs with a lifespan of less than 1 hour
Stock to Flow Deflection	ratio between the current Bitcoin price and the S/F model
Percent Supply in Profit	percentage of existing coins whose price at the time they last moved was lower than the current price
Net Unrealized Profit/Loss (NUPL)	difference between Relative Unrealized Profit and Relative Unrealized Loss

Table 3.2 Bitcoin on-chain metrics list

low FRM suggests that the chain is able to maintain its level of security without the need for inflationary subsidy. In contrast, a high FRM suggests that the chain needs large block rewards to maintain its security level.

Thermocap

Thermocap is a metric used to measure true capital flow into the network defined as the total economic work contributed by miners and first described by Nic Carter (2018) [20]. Thermocap at time t is measured as the block rewards received by miners multiplied by the price of Bitcoin when the block reward was received:

$$Thermomcap_t = Block\ Reward_t \cdot Bitcoin\ Price_t$$

the *Thermocap* is also known as the *Aggregate Security Spend*, as it indicates the total value spent by miners to keep the network secure. In addition, since the miners must operate in profit, it can be assumed that the cumulative expenditure by miners should not exceed total rewards provided by the protocol.

Coin Days Destroyed (CDD)

Coin Days Destroyed (CDD) is a metric that indicates the economic activity of the network and gives more weight to unspent bitcoins for longer. An unspent bitcoin is the amount of bitcoin left remaining after executing a transaction. The term comes from the transaction model on which Bitcoin is based, namely "Unspent Transaction Output" (UTXO). CDD is based over the concept of UTXO, and it is measured as:

$$CDD = UTXO\ Value \cdot Lifespan_{days}$$

is considered an effective metric to analyze the spending behavior of smart money, whales and long term investors, as it takes into account both the volume of bitcoins spent and the lifespan. Basically every day, every unspent bitcoin accumulates a "coin day". When that bitcoin is spent, its "coin day" is reset to zero. CDD indicates the aggregated value of all coin days. For example, 2 BTC which are dormant for 100 days, have accumulated 200 coin days.

Cumulative Value-Days Destroyed (CVDD)

Cumulative Value-Days Destroyed (CVDD) is a metric introduced by Willy Woo (2019) [35] historically used to identify market bottoms of bitcoin. CVDD leverages CDD and aims to analyze the cumulative sum of both USD value and time-value of bitcoins spent. It is measured as:

$$CVDD_{USD} = \frac{\sum(CDD \cdot Price)}{Days \cdot 6000000}$$

where the 6 million value is arbitrary and was established only as a chart stabilizer.

Dormancy

Dormancy is a metric introduced by Reginal Smith (2017) [29] to analyze the average number of days each transacted coin remained dormant, unmoved. It is also a metric to measure the economic activity of the network and it is defined as:

$$Dormancy = \frac{CDD}{Onchain\ Transfer\ Volume}$$

high values of *Dormancy* indicate that for that specific day many long term holders are putting their bitcoins into circulation and could indicate the beginning of a bearish period. On the contrary low *Dormancy* values are a bullish indicator as the number of long term holders accumulating bitcoins starts to grow. $Onchain\ Transfer\ Volume$ corresponds to the total number of coins transacted on the blockchain.

Entity-adjusted Dormancy Flow

Entity-adjusted Dormancy Flow was introduced by David Puell (2019) [8] and takes the concept of Dormancy to analyze bullish or bearish Bitcoin trends. It is measured as:

$$Dormancy\ Flow = \frac{MarketCap_{USD}}{365^{MA}(Dormancy \cdot Price_{USD})}$$

basically, when the *Marketcap* remains high relative to the yearly moving average of its realized *Dormancy* in USD, the bull market can be considered as "healthy," because the price remains high relative to the market's annualized spending behavior. When instead the *Dormancy* value increases with respect to the *Marketcap*, it indicates a potential buying point.

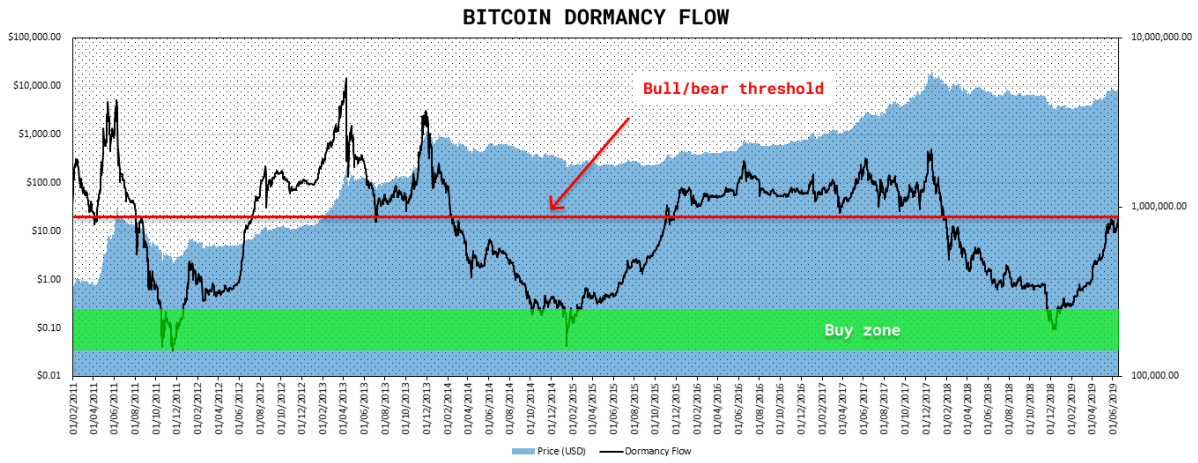

Fig. 3.2 Bitcoin Dormancy Flow — Source: David Puell (2019)

Liveliness

Liveliness is a metric created by Tamas Blummer (2018) [33], which analyzes network saving behavior. It is measured as:

$$Liveliness = \frac{CDD}{\sum Coins\ Created}$$

in this way it gives us information about saving behavior of investors, as it will tend to 1 if long-term positions are moved, and potentially liquidated, while it will tend to 0 if investors accumulate bitcoin.

Realized HODL (RHODL) Ratio

The Realized HODL ($RHODL$) Ratio is a metric used to determine overheated markets and analyze cycle tops. It is measured using a ratio of Realized HODL Waves, and was created by Philip Swift (2020) [25]. The Realized HODL Waves are different age bands of bitcoin UTXO weighted by the Realized Value of bitcoins within each band. Realized Value means the value assigned to each UTXO equal to the bitcoin prices in USD at the time when said UTXO last moved. The RHODL Ratio is measured as:

$$Realized\ HODL\ Ratio = \frac{RHODL_{1w}}{RHODL_{1yr-2yr}} \cdot Total\ Market\ Age$$

taking the ratio between $RHODL$ waves at 1 week, i.e. short term, and $RHODL$ waves at 1yr-2yr, i.e. long term, is able to capture the speculative market psychology of Bitcoin. Additionally, the ratio is adjusted by the increased supply by weighting the ratio by the $TotalMarketAge$. When the 1 week $RHODL$ band is significantly larger than the 1yr-2yr $RHODL$ band, then it can be assumed that the market is becoming overheated. Historically this approach has been able to identify market tops within a few days accuracy by avoiding false signals of a cycle high.

Realized Profit/Loss Ratio

The Realized Profit/Loss Ratio is the ration between all coins moved at a profit and at a loss.

$$Realized\ P/L\ Ratio = \frac{Realized\ Profit}{Realized\ Loss}$$

the $Realized\ Profit$ actually indicates the total profit in USD of all bitcoins whose price the last time they were moved was less than the current price. Conversely, the $Realized\ Loss$ indicates the total loss of all bitcoins whose price the last time they were moved was greater than the current price.

Fig. 3.3 RHODL Ratio (blue line) — Source: Glassnode.com

Difficulty Ribbon Compression

The Difficulty Ribbon Compression is a metric used to quantify the compression of the Difficulty Ribbon and consequently determine buying opportunities zones, it was first introduced by Willy Woo (2019) [36]. The Difficulty Ribbon is an indicator based on simple moving average of Bitcoin mining difficulty used to create ribbons. Periods when the ribbon compresses constitute good buying opportunities. This mechanism is due to the capitulation of miners, which results in a reduction in mining difficulty and is thus reflected by the Difficulty Ribbon. Basically, miners sell part of their block rewards to pay for production costs, and this produces a bearish price pressure. In the moment in which the weakest miners do not succeed to maintain the production costs through the block reward, they capitulate, and consequently the mining difficulty and the hashing power decrease. This leaves only the most efficient miners operative and greatly reduces bearish price pressure, leading to a period of price accumulation and stabilization. Historically capitulation occurs during a

bear market or due to block reward halvening.

Adjusted Spent Output Profit Ratio (aSOPR)

The Adjusted Spent Output Ratio ($aSOPR$) is a Spent Output Profit Ratio (SOPR) in which all UTXOs with a lifespan less than 1 hour are ignored. The SOPR is a metric used to analyze the economic behavior of investors using spent outputs. it was created by Renato Shirakashi (2019) [28]. It is measured as:

$$SOPR = \frac{Realized\ Value}{Value\ at\ Creation}$$

the $Realized\ Value$ is equal to the $Realized\ Profit$ or $Realized\ Loss$ described on the $Realized\ P/L\ Ratio$. The $Value\ at\ Creation$ is the value in USD when that UTXO was last spent. When the $SOPR > 1$ it indicates that the investors are in profit at the time of the transaction, otherwise in loss. During a bull market, SOPR values below 1 are rejected, while during a bear market SOPR values above 1 are rejected. This is because, in the first case, investors are psychologically reluctant to sell at loss, thus decreasing supply when the $SOPR < 1$ and leading to an upper price pressure. On the contrary, in the second case, investors wait to reach a break even to sell, thus determining a strong down price pressure when $SOPR > 1$. We can therefore say that the SOPR can be used as a metric to determine local market tops and bottoms.

Stock to Flow Deflection

Stock to Flow Deflection is a metric based on Stock to Flow (S/F) model and used to determine if Bitcoin is under or over-valued relative to the model. The S/F model is a popular model for determining the value of scarce assets. In the case of Bitcoin, it was first analyzed by PlanB (2019) [26] and it is measured as:

$$S/F\ Deflection = \frac{Bitcoin\ Price}{S/F\ Value}$$

according to the S/F model, Bitcoin's value is significantly determined by its scarcity, which can be quantified as:

$$Scarcity\ (S/F) = \frac{Stock}{Flow}$$

where $Stock$ is the size of the existing stockpiles or reserves, while $Flow$ is the yearly production. For example, gold has a S/F of 62, so at the current level of production/extraction of gold, it takes 62 years to get the current gold stock. According to this reasoning, the market value of a scarce asset can be quantified by its S/F even simply by a linear regression on logarithmic values, achieving an R2 of 95% [26]. In the case of Bitcoin, its S/F is greatly impacted by block reward halvening, which every four years halves the yearly production of bitcoin ($Flow$), thus doubling the S/F. As of May 2020, with the latest halvening, Bitcoin has reached an S/F of around 50, and approaching that of gold.

Fig. 3.4 Stock to Flow Deflection — Source: Glassnode.com

Percent Supply in Profit

The Percent Supply in Profit is the percentage of existing bitcoins whose price the last time they were moved was lower than the current price. This metric can be used to analyze the current stake of the Bitcoin market. Indicatively, when this percentage exceeds 95% it is likely that we are approaching a market top, on the contrary when it falls below 50% we are approaching a market bottom.

Net Unrealized Profit/Loss (NUPL)

Net Unrealized Profit/Loss ($NUPL$) is a metric based on $RelativeUnrealizedProfit$ and $RelativeUnrealizedLoss$ and used to analyze investor sentiment. It was first introduced by Adamant Capital (2019) [6] and it is measured as:

$$NUPL = RUP - RUL$$

Relative Unrealized Profit (RUP) and *Relative Unrealized Loss* (RUL) are measured by normalizing *Unrealized Profit* (UP) and *Unrealized Loss* (UL) against market cap, respectively. *Unrealized P/L* is measured by weighting each circulating bitcoin by the difference between the current price and its realised price and sum up all bitcoins in profit and loss, respectively.

$$UP = \sum_{UTXOs} value \cdot max(0, price_{current} - price_{realized})$$

$$UL = \sum_{UTXOs} value \cdot max(0, price_{realized} - price_{current})$$

$$RUP = UP \cdot MarketCap$$

$$RUL = UL \cdot MarketCap$$

following this approach, roughly when the $RUP > 0.75$ potentially indicates a market top and a good selling opportunity. On the contrary when $RUP < RUL$ it indicates a good buying opportunity. Also according to [6], $NUPL$ can also be measured as:

$$NUPL = \frac{MarketCap - Realized\ Cap}{MarketCap}$$

3.2 Data Labels

Since the task addressed by this work is a classification problem, it is necessary that for each row of our matrix $\mathbf{X}$ of features, a label y is identified. In this way the model proposed can learn the patterns and predict the labels even on unseen features sample. The labels indicate days when there are good buying opportunities or selling opportunities and are created following the *Triple Barrier* method devised by Marco Lopez de Prado (2018) [27]. This type of labeling has not been found in the literature, where fixed-horizon labeling methods are used instead. With the latter, observations are labeled according to the return on some fixed number of steps in the future (see Fig. 3.5). However, this type of labelling presents two problems [16]:

- *The thresholds are fixed, but volatility isn't.* For this reason labels could be affected by different periods of volatility.
- *The label is path-independent*, i.e. it only depends on the return at the horizon and not on the intermediate returns. This unfortunately does not reflect a real trading session in which "take profit" and "stop loss" limits are common [1].

Through the Triple Barrier with Dynamic Thresholds it is possible to overcome both of the above problems by taking into account the volatility and incorporating the "take profit" and "stop loss" scenarios.

3.2.1 Triple Barrier Labelling

The Triple Barrier method labels an observation based on the first barrier touched on three barriers (See Fig. 3.6). There are two horizontal barriers and one vertical barrier. The horizontal barriers identify the "take profit" and "stop loss" limits, which are set dynamically according to the volatility of the asset and are not necessarily symmetrical. Basically, the upper barrier is the threshold an observation's return needs to reach to be considered a buying

[1] "take profit" and "stop loss" limits are types of limits orders used in trading that specifies the exact price at which to close an open position for a profit or a loss.

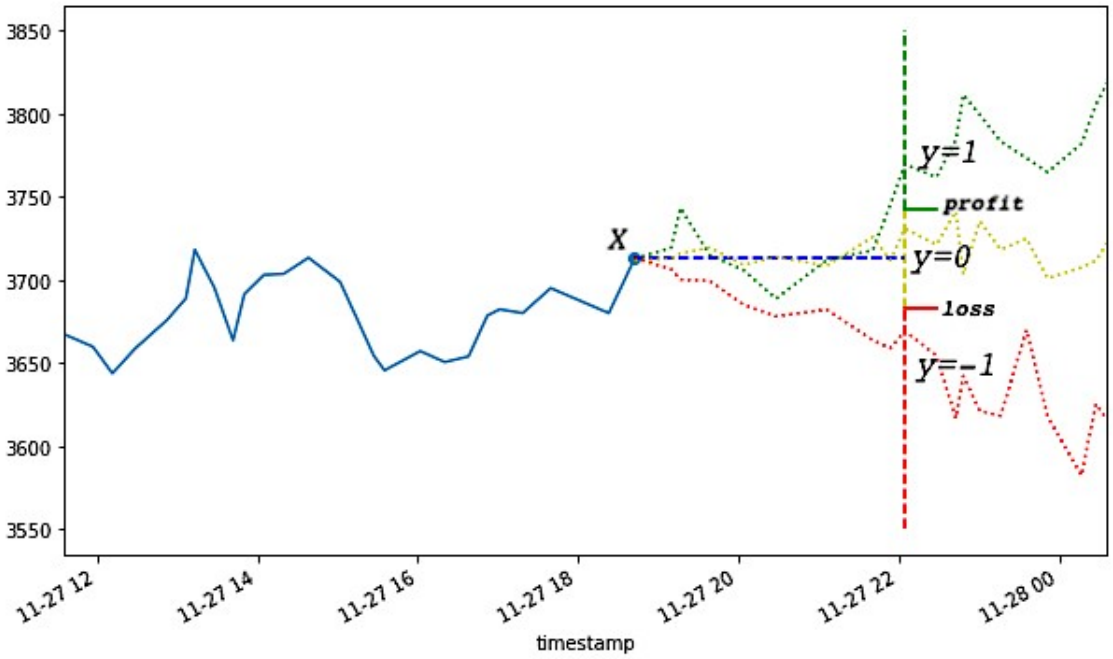

Fig. 3.5 Example: Fixed Horizon Labelling — Source: Maks Ivanov

opportunity, while the lower barrier is the threshold for the observation to be considered a selling opportunity. The vertical barrier instead reflects an expiration limit, based on the number of bars elapsed since the position was taken. Generally if the upper barrier is touched first, then this will be a good buying opportunity and labels the observation as $y_t = 1$. If the lower barrier is touched first, then it will be a good selling opportunity and labels the observation as $y_t = -1$. While if the vertical barrier is touched first it is more appropriate to hold the previous position, and the observation is labeled as $y_t = 0$. In any case, for the purpose of this work no vertical barrier is taken into account, so a position has no expiration limit, and we would have only two types of classes: buying opportunities, labelled as $y_t^{buy} = 1$, and selling opportunities labelled as $y_t^{sell} = 0$. In addition, this strategy makes the labelling method path dependent, so in order to label an observation it is necessary to consider all the path spanning $[t_{i,0}, t_{i,0} + h]$, where h indicates the expiration limit of the vertical barrier. For this work, the value h has been set to a very high value in such a way that it is not evaluated, and thus the whole path is considered for labelling.

Fig. 3.6 Example: Triple Barrier Labelling — Source: De Prado [27]

3.2.2 Comparing Returns: Triple Barrier vs Buy-and-Hold

By using the labels created by the Triple Barrier method, we can make a first comparison of the returns between the Buy-and-Hold strategy and the potential trading algorithm with the true labels. The Buy-and-Hold strategy consists of buying the asset at time t_0 and holds it until time t_{0+h} where h coincides with the last day of the data. The strategy used by the algorithm exploiting the true labels, consists instead in buying the maximum amount of assets when $y_t = 1$, so when there is a buying opportunity, and sell all the asset in our possession when $y_t = 0$, so when there is a good selling opportunity. The data used start from September 6, 2011, through May 30, 2021. In table 3.3 we summarize the returns and the sharpe ratio achieved by these two strategies.

According to these strategies, starting with an initial capital of 1, the returns of the algorithm

Strategy	Total Return	Sharpe Ratio
Buy-And-Hold	$\times 5,264$	0.0640
Triple Barrier Based	$\times 402,543$	0.4318

Table 3.3 Buy-and-Hold vs. Triple Barrier Strategy

exceed the Buy-and-Hold by 39,727,822%, and achieve a significantly higher sharpe ratio. This is a first approach to determine the financial potential of our model, as its goal will be to predict the true labels created by the Triple Barrier method.

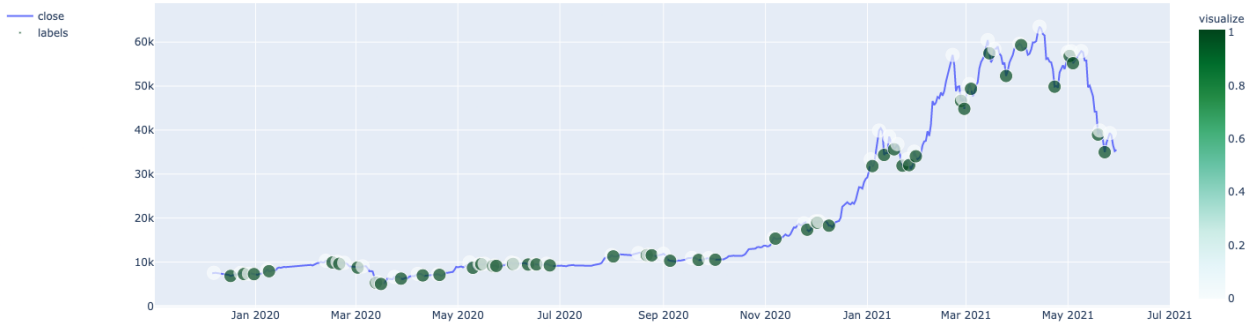

Fig. 3.7 Triple Barrier Strategy - last 500 days sample

3.3 Feature Engineering

Through Feature Engineering it is possible to extract useful patterns from data in order for machine learning models to achieve better performance. To this end, the matrix **X** of features composed of the OHLC price data and the previously defined on-chain metrics, is transformed to increase the number of features and attempt to derive more usable and explanatory features. Technical indicators are measured on all the features within the matrix **X**, also following the strategies identified in the literature [23, 30, 19]. Then, the augmented matrix **X** of features is further transformed to make the time series stationary via fractional differentiation, following the approach defined by Marco Lopez de Prado (2018) [27].

3.3.1 Technical Indicators

For each technical indicator used, 15 different periods are calculated, starting from the 6th period up to the 21st period. These technical indicators are classified into momentum and trend indicators, and some of them measured only on the close price of Bitcoin. Table 3.4 contains the complete list of used technical indicators.

Technical Indicator	Type	Only Close Price
RSI	Momentum	False
CMO	Momentum	False
ROC	Momentum	False
SMA	Trend	False
EMA	Trend	False
WMA	Trend	False
TRIX	Trend	False
DPO	Trend	False
BB MAV	Trend	False
William R	Momentum	True
MACD	Momentum	True
CCI	Trend	True
DMI	Trend	True
HMA	Trend	True

Table 3.4 Technical Indicators List

Relative Strength Index (RSI)

The RSI is a momentum oscillator which measures the speed and change of the time series feature movement. Regarding its use on the close price, it is useful to evaluate if the market is overbought or oversold and it is measured as:

$$RSI(n) = 100 - \frac{100}{1 + RS(n)}, \quad \text{with} \quad RS(n) = \frac{Average\ Gains}{Average\ Losses}$$

n indicates the number of time periods to be analyzed; *Average Gains* and *Average Losses* indicates respectively the average gains and losses of up periods during the last n periods. The RSI assumes values between 0 and 100, where 0 indicates no up periods, while 100 indicates only up periods. Traditionally, values greater than 70 indicate an overbought market, while values below 30 indicate an oversold market.

Chande Momentum Oscillator (CMO)

The CMO is a momentum oscillator used to measure trend strength of a time series feature and takes values between -100 and +100. In the case of price, it can identify relative strength or weakness in a market. In particular, with values greater than +50 the market is considered overbought, while with values lower than -50 the market is considered oversold. The CMO is measured as:

$$CMO = \frac{sH - sL}{sH + sL} \cdot 100$$

in the case of price, sH is the sum of absolute price changes on up days, while sL is the sum of absolute price changes on down days.

Rate Of Change (ROC)

The ROC is a momentum oscillator used to measure the percentual amount the time series features have changed over a defined number of past periods, generally 10 periods. In the case of price, high values of the ROC indicate an overbought market, while low values indicate

an oversold market. The ROC is measured as:

$$ROC(n) = 100 \cdot \frac{TS\ Feature_p - TS\ Feature_{p-n}}{TS\ Feature_{p-n}}$$

where p corresponds to the current period, and n corresponds to the number of periods to be analyzed.

Simple Moving Average (SMA)

The SMA is a trend indicator used to identify time series feature trends and a possible change in an established trend. It is measured as:

$$SMA_k = \frac{1}{k} \sum_{i=n-k+1}^{n} TS\ Feature_i$$

basically it is an arithmetic moving average calculated by adding recent time series feature values and dividing the result by the number of time periods k taken into account for averaging, while n is the number of total periods.

Exponential Moving Average (EMA)

EMA is a type of moving average, like SMA, in which more significance is given to the most recent data points, by applying weighting factors which decrease exponentially. In this way it is able to respond to changes in the values of the time series feature more quickly than SMA. In the case of price, it is usually used in conjunction with other indicators to confirm significant market moves and it is measured as.

$$EMA_t(n) = (TS\ Feature_t - TS\ Feature_p) \cdot Multiplier + EMA_p(n)$$

with

$$Multiplier = \frac{2}{n+1}$$

where t denotes the values of today's time series and today's EMA, and p the values of the previous day's time series and EMA. n instead denotes the number of time periods evaluated

to measure the mean.

Weighted Moving Average (WMA)

The WMA is a type of moving average where more significance is given to the most recent data points. It performs similar tasks to the EMA and it is measured as:

$$WMA = \frac{TS\ Feature_1 \cdot n + TS\ Feature_2 \cdot (n-1) + ...TS\ Feature_n}{\frac{n \cdot (n+1)}{2}}$$

where n denotes the number of time periods.

Triple Exponential Average (TRIX)

The TRIX is a momentum indicator used to filter out movements in the time series that are not considered significant. The value of TRIX oscillates around zero, positive values indicate that the momentum is increasing, while negative values indicate that it is decreasing. When values cross zero, it is usually considered as a buy or sell signal. In the case of price, extremely positive values may indicate an overbought market, or on the contrary oversold. It is measured using the moving average that has been smoothed exponentially three times:

$$TRIX_t = \frac{EMA3_t - EMA3_{t-1}}{EMA3_{t-1}}$$

with

$$EMA1_t = EMA_t(n)$$

$$EMA2_t = EMA(EMA1_t)$$

$$EMA3_t = EMA(EMA2_t)$$

where EMA_t denotes the current value of EMA, and EMA_{t-1} the value of EMA at the previous day.

Detrended Price Oscillator (DPO)

The DPO is an oscillator used to eliminate the trends of the time series, allowing to estimate the length of the cycles between peaks and troughs. In this way it is possible to estimate buy and sell points based on the historical cycle. The DPO is measured as:

$$DPO = TS\ Feature_{\frac{n}{2}+1} - SMA_n$$

where $TS\ Feature_{\frac{n}{2}+1}$ denotes the value of the time series at $\frac{n}{2}+1$ periods ago, while SMA_n denotes the SMA at n periods.

Bollinger Band® (BB)

Bollinger Bands is one of the most used technical indicators by traders, and developed by John Bollinger to determine when an asset is oversold or overbought. BBs measure volatility and how quickly the values of the time series can move up and down, using the bands above and below the Moving Average that expand and contract as volatility increases and decreases. The bands are measured based on the standard deviation of the time series, which can determine how far each individual data point varies from the average for all of the data points. In the case of price, a touch of the lower band identifies an oversold market, while a touch of the upper band an overbought market. BBs are measured as:

$$BOLU = MA(TS\ Feature, n) + m \cdot \sigma[TS\ Feature, n]$$
$$BOLD = MA(TS\ Feature, n) - m \cdot \sigma[TS\ Feature, n]$$

where $BOLU$ is the upper bollinger band, $BOLD$ is the lower bollinger band, MA is the moving average, n is the time period taken into account usually equal to 20, m is the number of standard deviations usually equal to 2 and $sigma[TSFeature, n]$ is the standard deviation of the time series over last n periods.

Williams %R

The Williams Percent Range is a momentum indicator that moves between 0 and -100 and is able to identify entry and exit points in the market. By comparing the closing price with the high-low range over a specific period, it is able to measure overbought and oversold levels. When the William %R takes values greater than -20, the market is considered overbought, while with values less than -80, oversold. It is measured as:

$$Williams\ \%R = \frac{Highest\ High - Close}{Highest\ High - Lowest\ Low}$$

where $Highest\ High$ and $Lowest\ Low$ are respectively the highest and lowest value of the price over the n time periods considered.

Moving Average Convergence–Divergence (MACD)

The MACD is a momentum oscillator measured by the differences between two exponential moving averages. To measure the MACD one usually subtracts the 26-period EMA of the price from the 12-period EMA. Together with the MACD, usually the Signal Line is also measured, measured as an EMA of the MACD and used together with the MACD to detect buy or sell signals.

$$MACD = EMA_{12} - EMA_{26}$$
$$Signal\ Line = EMA_9(MACD)$$

buy and sell signals are detected when the MACD and the Signal Line cross; A crossing by the MACD line above the signal line is considered a buy signal, while the opposite is a sell signal.

Commodity Channel Index (CCI)

CCI is a trend oscillator used to measure the variation of the price from its statistical mean. High values of CCI indicate that the price is high compared to its average, therefore generally

overbought. To the contrary, low values of CCI indicate that the price is low regarding its average, therefore oversold. High values mean CCI values above 100, while low values, below -100. CCI is measured as:

$$CCI(n) = \frac{1}{0.015} \cdot \frac{TP_p - SMA_n(TP_p)}{\sigma_n(TP_p)}$$
$$\text{with} \quad TP_p = \frac{High_p + Low_p + Close_p}{3}$$

TP_p is known as the typical price, while $High_p$, Low_p and $Close_p$ are the price values for the time period p. The SMA_n is the simple moving average of the typical price over the previous n time periods, while $\sigma_n(TP_n)$ is the mean deviation of the SMA during the previous n periods. Generally 14 periods are used for the SMA. The value 0.015 is a constant used to ensure that approximately 70-80% of the CCI values are within the range -100 and +100.

Directional Movement Index (DMI)

The DMI is a technical indicator which can determine both the strength and direction of a price movement and is intended to reduce false signals. This is possible by comparing prior highs and lows using a positive directional movement line (+DI) and a negative directional movement line (-DI). The strength is also estimated with the average directional index line (ADX) which shows the momentum of the asset. If +DI is above -DI there is more upward pressure, while on the contrary there is downward pressure. Also, if the spread between the two lines is very large, the price trend is stronger. The DMI is measured as:

$$+DI = \left(\frac{Smoothed\ +DM}{ATR}\right) \cdot 100$$
$$\text{-}DI = \left(\frac{Smoothed\ \text{-}DM}{ATR}\right) \cdot 100$$
$$DX = \left(\frac{|+DI - \text{-}DI|}{|+DI + \text{-}DI|}\right) \cdot 100$$

where

$$+DM = Current\ High - Previous\ High$$

$$\text{-}DM = Current\ Low - Previous\ Low$$

$$Smoothed\ +/\text{-}DM = \sum_{t=1}^{n} DM - \left(\frac{\sum_{t=1}^{n} DM}{n}\right) + Current\ DM$$

where $+/\text{-}DM$ indicates the directional movement of the asset, n is the number of time periods taken into account and typically equal to 14, ATR is the average true range measured as the average of the true ranges, i.e. given a trading day, the high minus the low of an asset.

Hull Moving Average (HMA)

The HMA is a moving average capable of reducing lag and noise, while increasing responsiveness. It is used both to identify the trend of the time series and to identify entry signals in the direction of the prevailing trend. If the prevailing trend is positive, there is a buy signal when the HMA turns up. On the contrary there is a sell signal if the prevailing trend is negative and the HMA turns down. The HMA is measured as:

$$HMA = WMA_{\sqrt{n}}(2 \cdot WMA1 - WMA2)$$

where

$$WMA1 = WMA_{\frac{n}{2}}(TS\ Feature)$$

$$WMA2 = WMA_{n}(TS\ Feature)$$

where n is the time period considered to measure the moving average, $WMA1$ is the weighted moving average of the time series with respect to $\frac{n}{2}$ time period, $WMA2$ instead the weighted moving average of the time series measured over n time period and $WMA_{\sqrt{n}}$ the weighted moving average measured over $\sqrt{n}$ time period.

3.4 Stationarity by Fractional Differentiation

Supervised Learning models usually need the data to be comparable with each other, both in the train set, in the test set and on future data on which the model will be used. In this way they are able to map a previously unseen observation to a collection of labeled examples, in order to infer from them the label of that new observation. The financial time series are known for their remarkable amount of noise and for their great amount of trends that shift the series' mean over time. These two properties in particular make the modelling of the financial time series extremely difficult. To overcome these difficulties, it is necessary to make the time series stationary, i.e. make sure that the statistical properties of the process, such as mean, variance and covariance, are invariant with respect to the time ordering. To verify that a time series is stationary, we use the Augmented Dickey Fuller (ADF) test, or visually we can verify that the values of the time series return to the same mean, because a stationary process can never drift too far from its mean because of the finite variance. The ADF test is one of the statistical tests used to verify the unit root of a time series. The unit root is a characteristic of time series that determines their non-stationarity. A time series is said to have unit root when $\alpha = 1$ in:

$$y_t = \alpha y_{t-1} + \beta X_e + \epsilon_t$$

where y_t is the value of the time series at time t, X_e is an exogenous variable and ϵ_t is a white noise and corresponds to the part that cannot be predicted given the past values of the time series. So, when $\alpha = 1$ we say that the time series is not stationary. The ADF test verifies the null hypothesis that $\alpha = 1$, so as to identify if a time series is stationary or not.

Usually to make a time series stationary, we apply a first differentiation which in the case of a time series of the price of an asset corresponds, to its returns. The first difference operator is defined as:

$$\Delta y_t = y_t - y_{t-1} \tag{3.1}$$

Instead, the first order lag operator is defined as:

$$Ly_t = y_{t-1} \tag{3.2}$$

where $L^2 y_t = L(Ly_t) = Ly_{t-1} = y_{t-2}$ and more generally $L^k y_t = y_{t-k}$. From this, it is possible to construct polynomials of the lag operator, such as:

$$y_t + \alpha_1 y_{t-1} + \alpha_2 y_{t-2} = (1 + \alpha_1 L + \alpha_2 L^2) y_t$$

and it is possible to rewrite the first difference in equation (3.1) as:

$$\Delta y_t = (1 - L) y_t \tag{3.3}$$

The first difference, even if it succeeds in making the time series stationary, tends to remove memory completely from the original series [1]. Although for inferential purposes, stationarity is a necessary property, memory is the basis of predictive model performance, so we want to avoid removing it completely. Unfortunately, the price in time series is usually non-stationary, while the first difference is stationary but without memory. To address this limitation it is possible to apply a fractional differentiation, proposed by Hosking (1981) [15]. Hosking generalized the $ARIMA(p, d, q)$ models [2] into the Auto Regressive Fractionally Integrated Moving Average $ARFIMA(p, d, q)$ process, which for $0 < d < \frac{1}{2}$ is stationary and possesses long memory. Where p is the lag order and defines the number of lag observations in the model; d is the degree of differencing and defines the number of times the raw observations are differenced; q is the order of the moving average and defines the size of the moving average window.

To apply fractional differentiation, we exploit the lag operator described in equation (3.2). We can see how:

[2]The Auto Regressive Integrated Moving Average (ARIMA) is the most widely used group of econometric model, which interprets a given time series based on its lagged values and errors, to predict future values.

$$(1-L)^2 = 1 - 2L + L^2 \tag{3.4}$$

where $L^2 y_t = y_{t-2}$, as seen above, then $(1-L)^2 y_t = y_t - 2y_{t-1} + y_{t-2}$. It is necessary to highlight how $(x+y)^n = \sum_{k=0}^{n} \binom{n}{k} x^k y^{n-k} = \sum_{k=0}^{n} \binom{n}{k} x^{n-k} y^k$, for n a positive integer. Consequently, for a real number d, $(1+x)^d = \sum_{k=0}^{\infty} \binom{d}{k} x^k$ [27]. In a fractional model, the exponent d can be a real number, and we can then expand (3.4) using binomial coefficients:

$$\begin{aligned}
(1-L)^d &= \sum_{k=0}^{\infty} \binom{d}{k} (-L)^k \\
&= \sum_{k=0}^{\infty} \frac{\prod_{i=0}^{k-1}(d-1)}{k!} (-L)^k \\
&= \sum_{k=0}^{\infty} (-L)^k \prod_{i=0}^{k-1} \frac{d-1}{k-1} \\
&= 1 - dL + \frac{d(d-1)}{2!} L^2 - \frac{d(d-1)(d-2)}{3!} L^3 + ...
\end{aligned} \tag{3.5}$$

By using a non-integer positive value $d \in \mathbf{R}^+$ it is possible to preserve memory. This is evident in Figure (3.8), as the lag weights become asymptotically small for real orders d and large lags. We have:

$$\widetilde{X}_t = \sum_{k=0}^{\infty} \omega_k X_{t-k} \tag{3.6}$$

With weights $\omega = \left\{1, -d, \frac{d(d-1)}{2!}, \frac{d(d-1)(d-2)}{3!}, ..., (-1)^k \prod_{i=0}^{k-1} \frac{d-1}{k!}, ...\right\}$ and values $X = \{X_t, X_{t-1}, X_{t-2}, X_{t-3}, ..., X_{t-k}, ...\}$. When d is a positive integer number, $\prod_{i=0}^{k-1} \frac{d-1}{k!} = 0, \forall k > d$, and the memory beyond that point is deleted. For example, via the first difference (i.e. with $d = 1$), we have $\prod_{i=0}^{k-1} \frac{d-1}{k!} = 0, \forall k > 1$, and $\omega = 1, -1, 0, 0, ...$ Furthermore, it is possible to observe how the sequence of weigths ω, with $k = 0, ..., \infty$ and $\omega_0 = 1$, can be generated iteratively as:

$$\omega_k = -\omega_{k-1} \frac{d-k+1}{k} \tag{3.7}$$

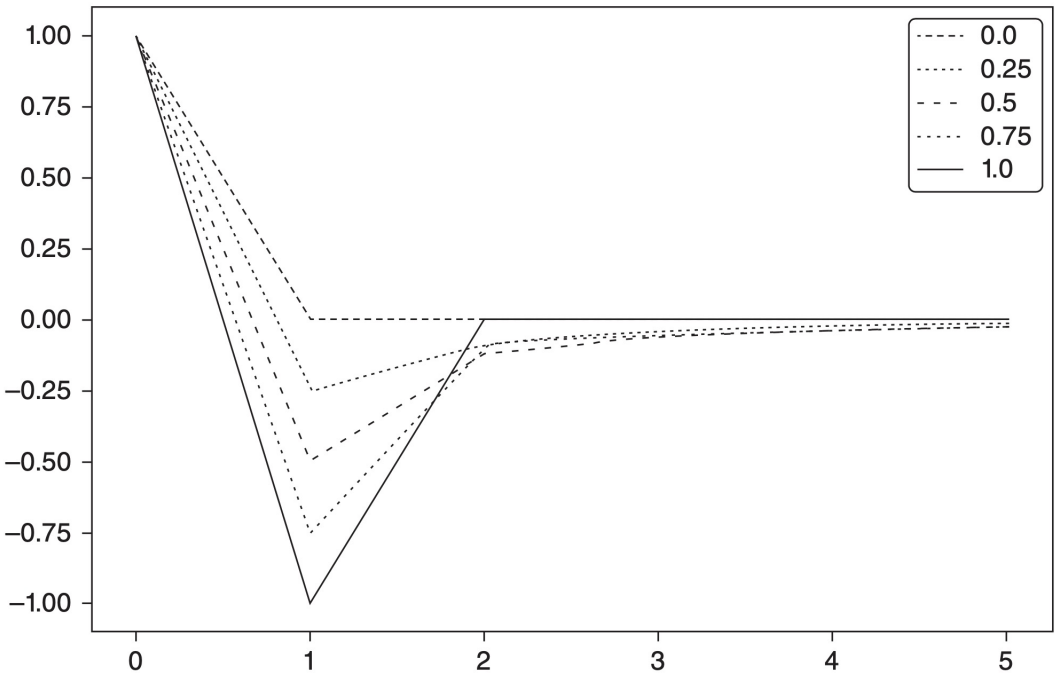

Fig. 3.8 ω_k (y-axis) as k increases (x-axis). Each line is associated with a particular value of $d \in [0, 1]$, in 0.1 increments. — Source: De Prado [27]

3.4.1 Implementation

To perform the transformation of the original time series to fractionally differentiated, we apply the series expansion described in equation (3.5) for a specific $d \in \mathbf{R}^+$. Next, as the lag weights become asymptotically small, as highlighted above, we choose to cut off the weights beyond a certain window size $k = 295$, where they have become small enough to be insignificant. The transformation is applied to all non-stationary time series contained within the matrix $\mathbf{X}$ of features and augmented using the technical indicators. Before carrying out the transformation, it is verified whether the time series, or its log transformation, taken into consideration is already stationary. If not, we proceeded with the fractional differentiation. Thus, given a time series $[y_1, ..., y_T] \in$ matrix $\mathbf{X}$, if $[y_1, ..., y_T]$ or $\log([y_1, ..., y_T])$ is stationary according to ADF test, leave the time series unchanged or with its logarithmic transformation, otherwise find the minimum value of $d \in \mathbf{R}^+$, with $0 < d < 1$, such that $\Delta[y_1, ..., y_T]$ is stationary.

3.5 Time-Series Data to 2D Images with Recurrence Plot

To train the proposed model, the matrix $\mathbf{X}$ of features transformed via fractional differentiation is converted into images via Recurrence Plot. A recurrence plot is a square matrix $RP_{i,j}$, when $i, j = 1, ...N$ where the values of $RP(i, j)$ are equal to 1 if the distance between the points $x(t_i)$ and $x(t_j)$ in phase space do not exceed a predefined value ϵ, otherwise $RP(i, j)$ is equal to 0.

$$RP(i,j) = \begin{cases} 1 & \|\overrightarrow{x_i} - \overrightarrow{x_j}\| \leq \epsilon \\ 0 & \|\overrightarrow{x_i} - \overrightarrow{x_j}\| > \epsilon \end{cases} \quad i, j = 1, 2...N$$

Joint recurrence plots are an extension of recurrence plots applied to multivariate time-series. For each individual time series the recurrence plot is measured and then combined using the Hadamard product. The recurrence plots and the joint recurrence plots can be represented as images, usually we assign to the values $RP(i, j) = 1$ the color black, while to the values $RP(i, j) = 0$ the color white.

To increase the human-visible differences of the joint recurrence plots, a Min-Max scaler with values between $[-1, 1]$ is also applied. In Figure 3.9, we can see on the left the joint recurrence plot of all the days when according to Triple Barrier labelling it was convenient to buy bitcoin, while on the right the days when it was convenient to sell.

The differences between the two classes shown in Figure 3.9 give good insights on the potential performance of the model. To generate the images on a single date, each row of the matrix $\mathbf{X}$ is considered as a single time series and then transformed into an image by recurrence plot. Although this is a counter-intuitive strategy, it reported better results than generating the images using multiple rows of the X feature matrix via joint recurrence plot. In the figures 3.10 and 3.11 it is possible to see respectively the images generated on individual days for the days in which it was convenient to buy and for the days in which it was convenient to sell bitcoin. Figures 3.12 and 3.13 instead show the images generated using 50 rows of our matrix via joint recurrence plot. These last images therefore summarize

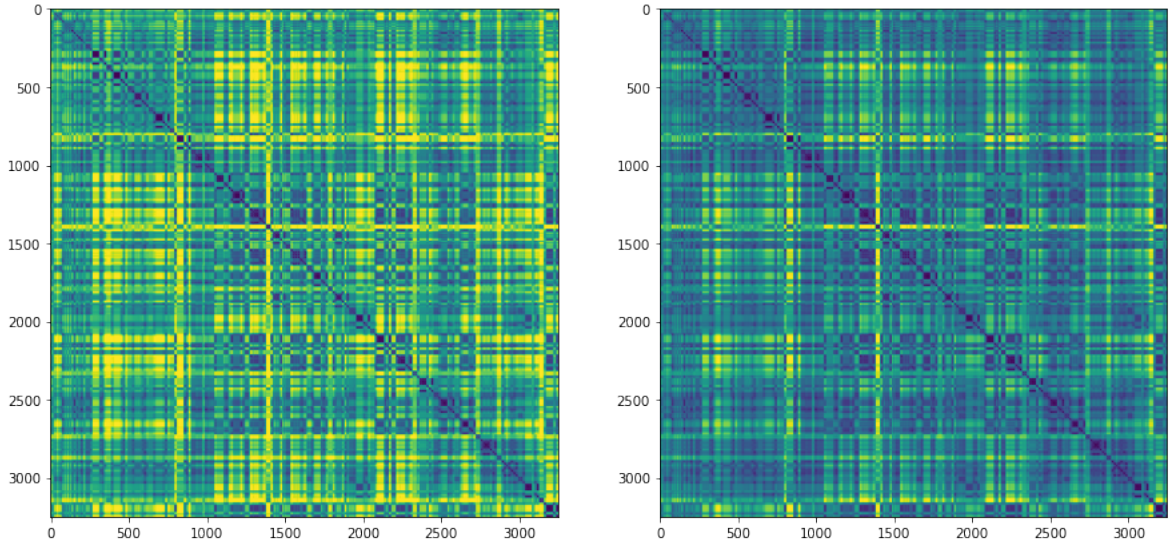

Fig. 3.9 Left: joint recurrence plot of all days when it was convenient to buy bitcoin — Right: joint recurrence plot of all days when it was convenient to sell bitcoin.

the information of all our features for 50 days at the same time, but they have not been used for the training since the previous images have been preferred in terms of performance of the model.

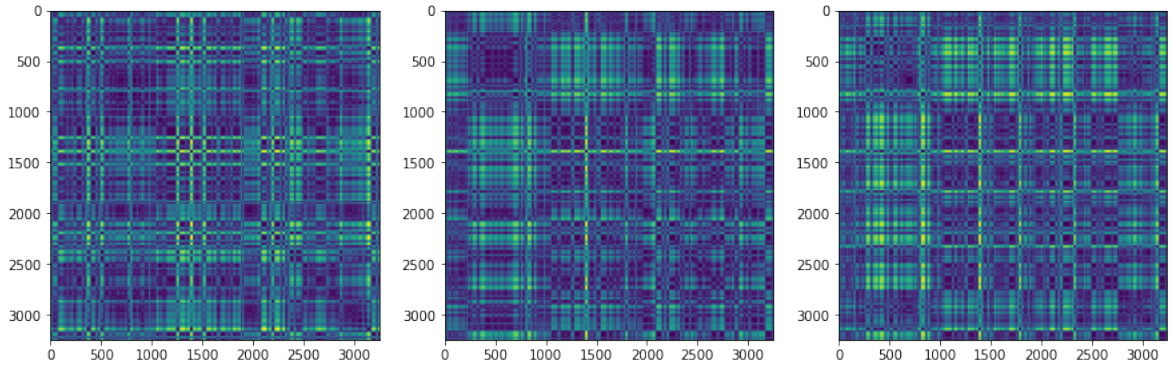

Fig. 3.10 Recurrence plots of a random sample of 3 days when it was convenient to buy bitcoin, using single day data as univariate time series.

In conclusion, to reduce the complexity of the data, and to improve efficiency during model training, the images were resized to 256×256 pixels.

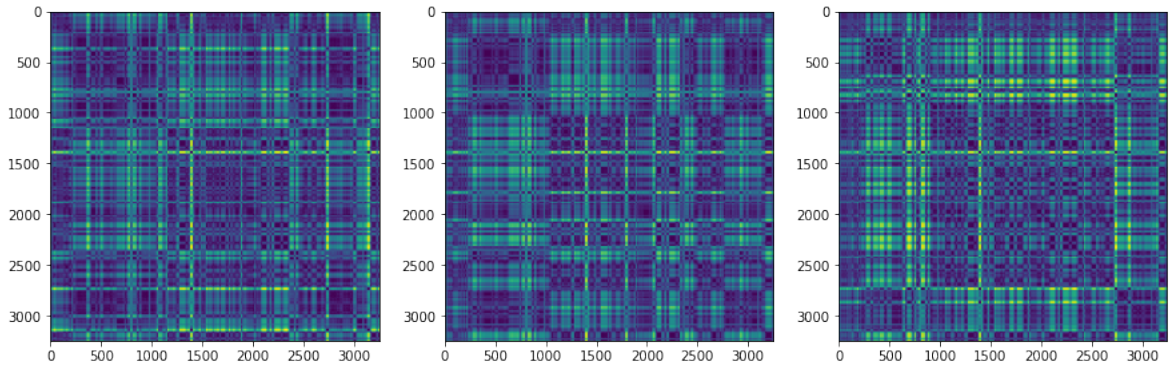

Fig. 3.11 Recurrence plots of a random sample of 3 days when it was convenient to sell bitcoin, using single day data as univariate time series.

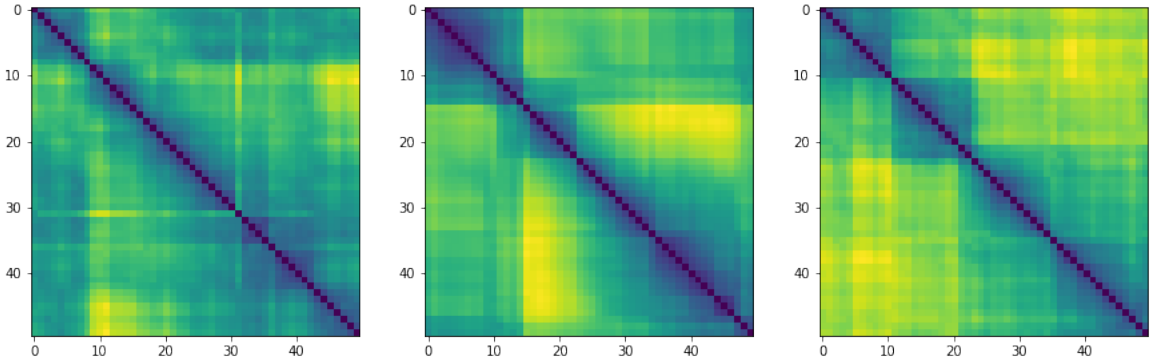

Fig. 3.12 Joint recurrence plots of a random sample of 3 days when it was convenient to sell bitcoin, combining 50 days data.

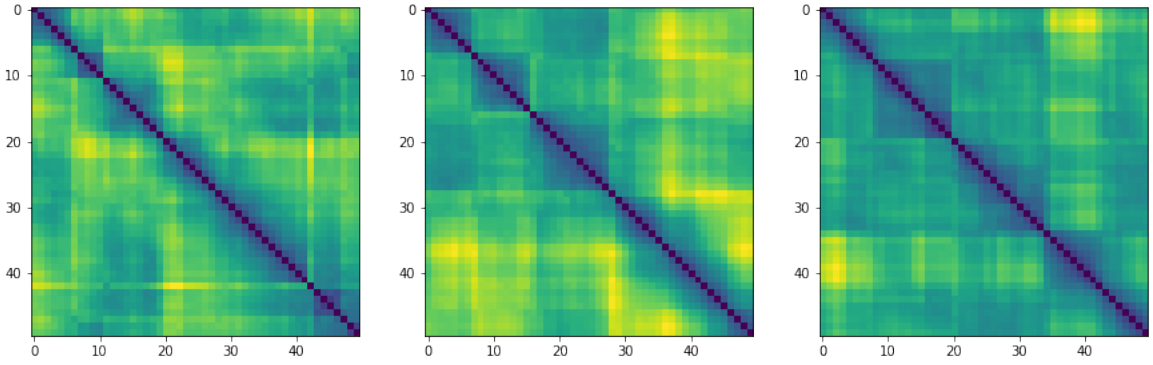

Fig. 3.13 Joint recurrence plots of a random sample of 3 days when it was convenient to sell bitcoin, combining 50 days data.

3.6 Convolutional Neural Networks and Proposed Model

The proposed model is an architecture based on a 2D Convolutional Neural Network (CNN), first introduced by Yann LeCun (1999) [2]. CNN are specialized Neural Networks (NN) for processing data that have a grid-like topology, such as images (2D or 3D tensors). CNN employs a mathematical operation called convolution, which is a linear operation used in place of the general matrix multiplication used on standard NN.

The convolution operator is defined as follows:

$$(f * g)(x) = \int f(z)g(x - z)dz$$

It is then possible to measure the overlap between f and g when a function is "flipped" and shifted by x. In the case of discrete objects, the sum replaces the integral. For two-dimensional tensors we have:

$$(f * g)(i, j) = \sum_a \sum_b f(a, b)g(i - a, j - b)$$

The convolution operation is used to find within our input tensor (image) a specific shape by using a kernel tensor. The kernel tensor (or filter) is passed to each part of the input tensor by sliding it, and at each step we compute the convolution operation. At the end of these procedures, we produce the output tensor called feature map, which is usually smaller than the input tensor. The feature map can be considered as the learned representations, or features, in the spatial dimensions (width and height), and it is passed to the subsequent layer.

The main advantage of using a convolution layer instead of a dense layer used in standard NN, is because a convolution layer is able to learn local patterns, while a dense layer is only able to learn global patterns. Local patterns are translation invariant, thus a convolution layers is able to identify them anywhere in the image, while a dense connected layer would need to learn a new pattern if the same local pattern is found in a different location. Hence convolution layers are more efficient than dense layers in pattern recognition.

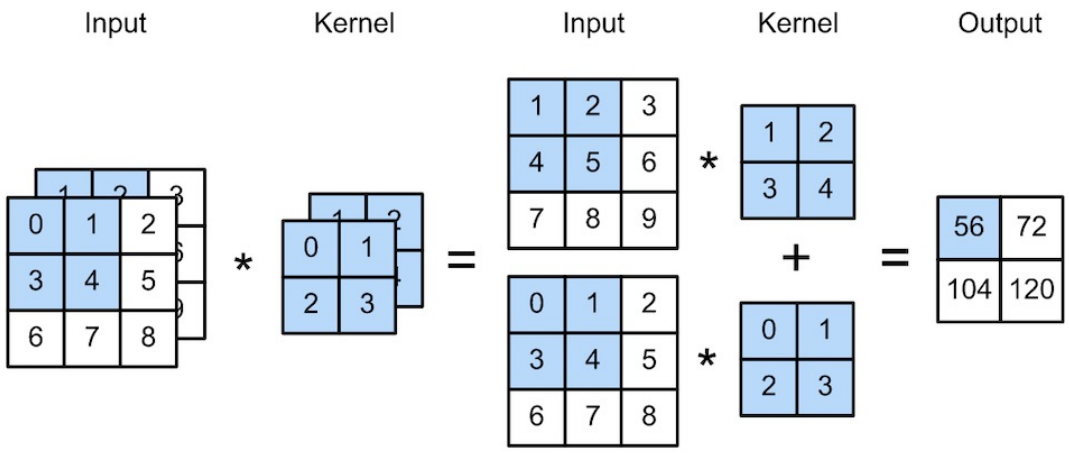

Fig. 3.14 Convolution operation — Source: d2l.ai

After convolution layers we usually introduce non-linearity by applying non-linear functions, such as Rectified Linear Unit (ReLu) or Leaky ReLu[3]. Then, we can further implement Pooling, a function that replaces the output of the convolution layer with a summary statistic of nearby outputs. This is useful to reduce the computational intensity needed for training the CNN, while at the same time not loose too much information. Max pooling is the most popular option of pooling layer.

3.6.1 Architecture of the proposed CNN Model

The proposed CNN model is composed of a two-stage structure. During each stage there is a feature learning phase using convolution, normalization, activation and pooling layers.

As can be seen in Figure 3.14, the proposed CNN model is composed of two convolution layers with 32-3 $\times$ 3 filters and 64-3 $\times$ 3 filters, respectively. After each convolution layer there is a Batch Normalization layer[4], a Leaky ReLu non-linear activation function with

[3]The ReLu activation function is a non-linear function that returns the value provided as input if the value is greater than 0, otherwise it returns 0. Leaky ReLu is a non-linear activation function based on ReLu, that has a small slope for negative values insted of a flat slope. It is used to avoid the sparse gradients problem.

[4]The Batch Normalization layer is used to normalized the output of the previous layers and it ensures that the mean and the variance of the layer inputs remains the same. They are usually used as regularization, to avoid overfitting of the model.

parameter $\alpha = 0.3$, and we end the stage with a max-pooling layer of pooling size 2×2.

After the two stages, the feature map obtained was flattened into a column vector, which was supplied to a feed-forward neural network. The latter is composed of two dense layers of 128 and 64 nodes respectively, and after each of them a dropout of 40% is applied to introduce some kind of regularization.

Throughout the training Adam optimizer was used to update parameters to achieve faster convergence with a learning rate that was automatically reduced every 40 epochs.

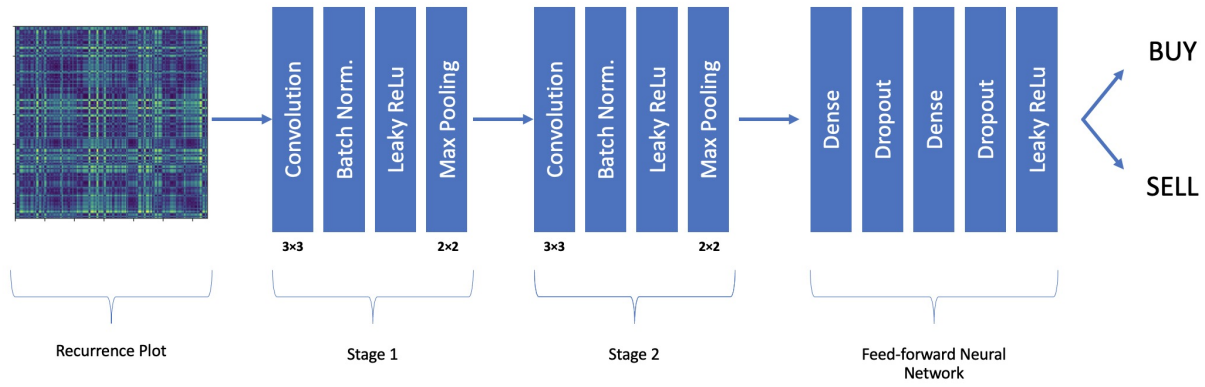

Fig. 3.15 Architecture of the proposed CNN Model

Layer (type)	**Output Shape**	**Param**
Conv 2D	(None, 254, 254, 32)	320
Batch Normalization	(None, 254, 254, 32)	128
Leaky ReLu	(None, 254, 254, 32)	0
Max Pooling 2D	(None, 127, 127, 32)	0
Conv 2D	(None, 125, 125, 64)	18496
Batch Normalization	(None, 125, 125, 64)	256
Leaky ReLu	(None, 125, 125, 64)	0
Max Pooling 2D	(None, 62, 62, 64)	0
Flatten	(None, 246016)	0
Dense	(None, 128)	31490176
Dense	(None, 64)	8256
Leaky ReLu	(None, 64)	0
Dense	(None, 2)	130

Table 3.5 Proposed CNN model Summary — Total params: 31,517,762; Trainable params: 31,517,570; Non-trainable params: 192

Chapter 4

Empirical Results

4.1 Performance of the Proposed CNN Model

The proposed CNN model is trained for 100 epochs with a batch size of 64, on 2070 samples of training data and validated on 518 data samples for each iteration. Then, the trained model is tested on 351 samples in the testing dataset, corresponding to the days from May 27, 2020 to May 27, 2021. To evaluate the performance of the model, the macro average F1-score is used as a metric to overcome the problem of class imbalance in the test set. Indeed, the test set consists of 96 observations y_T^{sell} and 255 observations y_T^{buy}. The macro average F1-score gives equal importance to each class, and is defined as:

$$\text{Macro F1-score} = \frac{1}{N}\sum_{i=0}^{N} \text{F1-score}_i$$

where F1-score is defined as the harmonic mean of the precision and recall:

$$\text{F1-score} = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$$

Precision is the fraction of true positives among the prediction positives: $Precision = \frac{TP}{TP+FP}$, while *Recall* is the fraction of true positives among the positives: $Recall = \frac{TP}{TP+FN}$.

The model is able to learn complex features efficiently, but overfitting the training set.

It is able to reach in the training set a Macro F1-score of 97%, and 62% in the validation set. As for the test set, the model reaches a Macro F1-score of 58%. In the classification report (Table 4.1), the model performs better on the class $y_t^{buy} = 1$, with a $Precision = 0.8$, $Recall = 0.62$, and $F1 - score = 0.70$, than on the class $y_t^{sell} = 0$. This could be partly due to the class imbalance on the training set, consisting of 1883 observations y_T^{buy} and 1056 observations y_T^{sell}.

Class	**Precision**	**Recall**	**F1-Score**	**Support**
0	0.37	0.59	0.45	96
1	0.80	0.62	0.70	255
Accuracy			0.61	351
Macro avg	0.58	0.60	0.58	351
Weighted avg	0.68	0.61	0.63	351

Table 4.1 Proposed CNN model — Classification Report on test set

Financial Evaluation

As carried out in section 3.2.2, below we compare the returns between the Buy-and-Hold strategy and the potential trading algorithm with the labels predicted by the model, by using the same naive trading strategy as defined before. In this case, the data used correspond only to the samples in the test set, from May 27, 2020 to May 27, 2021. In table 4.2 we summarize the returns and the sharpe ratio achieved by these two strategies. According to these strategies, starting with an initial capital of 1, the returns of the proposed CNN model based algorithm exceed the Buy-and-Hold by 7.88% with a 38.95% increase in the sharpe ratio.

Strategy	**Total Return**	**Sharpe Ratio**
Buy-And-Hold	329.63%	0.1291
Proposed CNN model Based	337.51%	0.1793

Table 4.2 Buy-and-Hold vs. Proposed CNN model based algorithm

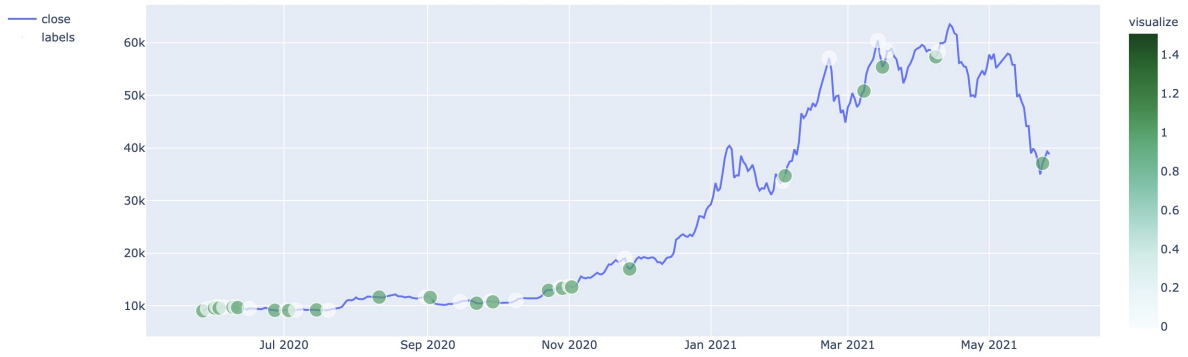

Fig. 4.1 Proposed CNN model based algorithm - Predicted Labels

4.1.1 Other examined Machine Learning models

To compare the performance of the proposed CNN models, other machine learning models are also tested on the matrix $\mathbf{X}$ of stationary features without the recurrence plot image transformation. These other models tested are the k-Nearest Neighbors (k-NN) and the Random Forest (RF) classifier.

k-Nearest Neighbors (k-NN)

The k-Nearest Neighbors (k-NN) is an algorithm used in pattern recognition for classifying observations based on the features of observations close to the one under consideration. The input is the k nearest training examples in the feature space, while in the case of the classifier, the output is a class. An observation is classified by a plurality vote of its neighbors, with the observation assigned to the most common class among its k nearest neighbors. For training the k-NN we used $k = 100$.

The k-NN, despite being a very simple and intuitive model, was able to outperform the proposed CNN model. In the test set, it achieved a Macro F1-score of 63%. In terms of Financial Evaluation, it also achieved the best results, outperforming the Buy-and-Hold of bitcoin by 75.69% in the test set, with a sharpe ratio increase of 50.86%.

Class	Precision	Recall	F1-Score	Support
0	0.45	0.49	0.47	96
1	0.80	0.77	0.79	255
Accuracy			0.70	351
Macro avg	0.62	0.63	0.63	351
Weighted avg	0.70	0.70	0.70	351

Table 4.3 k-NN — Classification Report on test set

Strategy	Total Return	Sharpe Ratio
Buy-And-Hold	329.63%	0.1291
k-NN Based	405.31%	0.1947

Table 4.4 Buy-and-Hold vs. k-NN based algorithm

Random Forest (RF) classifier

A Random Forest is an ensemble classifier obtained from the aggregation by bagging of decision trees. Bagging, or bootstrap aggregation, is a technique for reducing the variance of a forecast function. Bagging seems to work especially well with high variance and low bias methods, such as trees. Random Forests use a modification of bagging that constructs a large collection of de-correlated decision trees, and then computes the average. This solution minimizes the overfitting of the training set with respect to the decision trees and is able to achieve high performance in many problems. For classification tasks, we fit a "committee" of trees, each of which casts a vote for the predicted class.

The RF classifier is first tuned using GridSearch on a grid of parameters, with Time Series Split and a cross validation fold of 10. In performing cross validation on time series, the Time Series Split is essential, and it is not possible to use the more common K-fold-Cross-Validation. Indeed, in the case of time series we cannot choose random samples and split them into train and test sets, because it does not make sense to use future data to predict past data. With Time Series Split we avoid future-looking data during cross-validation, because folds are generated and tested on a rolling basis. We start with a small subset of data for training, and test for the next subset. Then we add the previous testing subset to

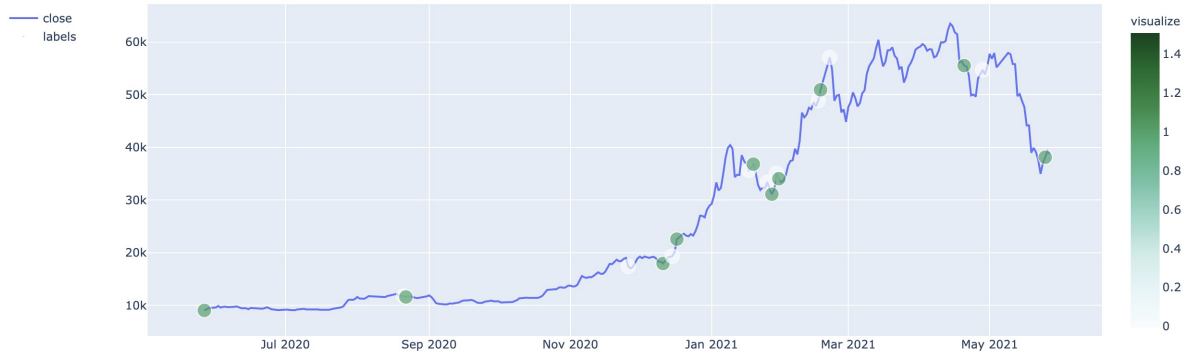

Fig. 4.2 k-NN based algorithm - Predicted Labels

the training data, and continue testing on the next subset, until all folds are used.

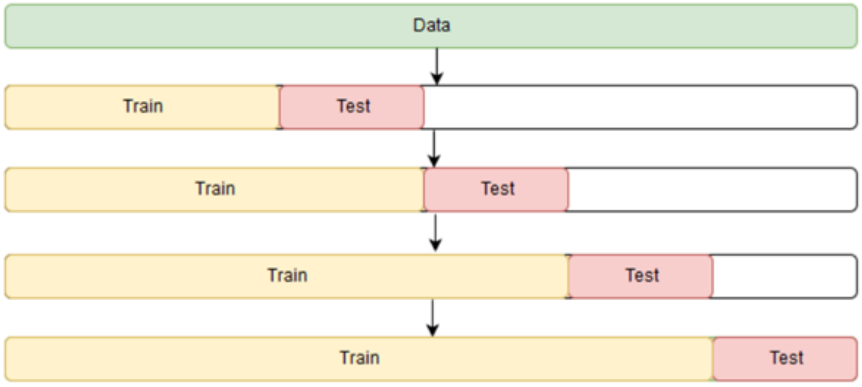

Fig. 4.3 Example of a Time Series Split - Source: Soumya Shrivastava

The RF classifier, compared to the proposed CNN and k-NN, performed the worst, achieving a Macro F1-score of 48% in the test set. Even in terms of Financial Evaluation, it was unable to outperform Buy-and-Hold of bitcoin, worsening both the returns and the sharpe ratio.

Class	Precision	Recall	F1-Score	Support
0	0.28	0.53	0.37	96
1	0.74	0.50	0.59	255
Accuracy			0.51	351
Macro avg	0.51	0.51	0.48	351
Weighted avg	0.61	0.51	0.53	351

Table 4.5 Random Forest Classifier — Classification Report on test set

Strategy	Total Return	Sharpe Ratio
Buy-And-Hold	329.63%	0.1291
RF Based	51.90%	0.0635

Table 4.6 Buy-and-Hold vs. RF based algorithm

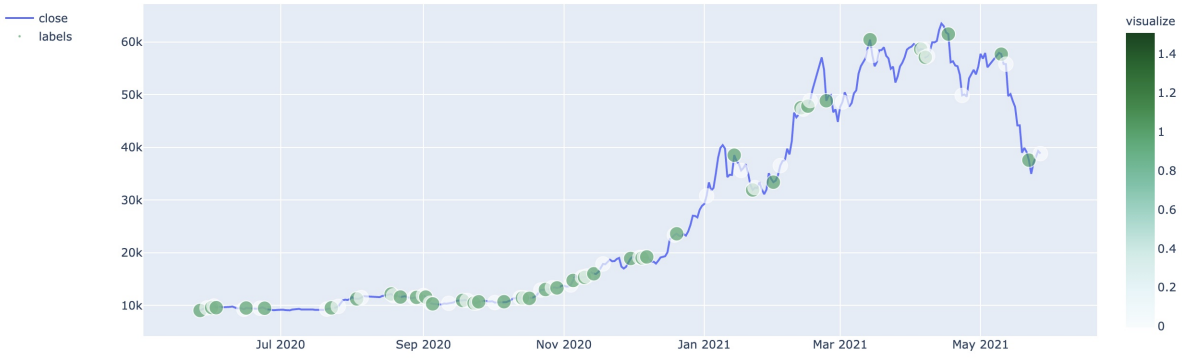

Fig. 4.4 RFC based algorithm - Predicted Labels

Chapter 5

Conclusion

The present study is the first one that attempts to classify the price of Bitcoin using both price data and Bitcoin on-chain metrics through a computer vision approach and by using the Triple Barrier labelling method. The proposed CNN model shows promising results, achieving a Macro F1-score of 58% and outperforming the results obtained from various researches analyzed in the literature. It should be noted that the performance obtained by the k-NN surpasses those of the proposed model significantly, reaching a Macro F1-score of 63%. It would be interesting to examine what are the reasons that lead to this increase in performance. The proposed model can be used in a trading algorithm.

5.1 Future Works

Other improvements and changes may be applied in future implementations of this work.

1. *The model parameters can be further fine-tuned for better performance.* Finding the most efficient architecture of a neural network for a specific purpose is a non-trivial task. In the present study, several architectures were tested and the one found to be empirically more performing is proposed. Other studies could seek, even with more analytical methods such as Bayesian Optimization, architectures more effective than the one proposed;

2. *Test the performance that can be achieved on a regression task.* The present study only analyze the performance of the proposed model for a classification task, further studies could also analyze the performance on a regression task;

3. *Solve class imbalance in training data.* The training set used presents a class imbalance with 1883 observations y_t^{buy} and 1056 observationos y_t^{sell}. It would be interesting to address this class imbalance, e.g. via Synthetic Minority Oversampling Technique (SMOTE), to investigate whether there is any increase in model performance;

4. *Analyze prediction performance on other cryptocurrencies data.* The analysis performed in this study could also be performed on other cryptocurrencies, such as Ethereum;

5. *Integrate both price data and on-chain metrics from other cryptocurrencies.* According to Ghorbel and Jeribi (2021) [12], cryptocurrencies exhibit interdependencies within the cryptocurrency market. Therefore, in addition to using price and blockchain data from Bitcoin, it would be interesting to also integrate data derived from other cryptocurrencies and blockchains such as Ethereum;

6. *Integrate other exogenous data, such as sentiment data or or crude oil, gold futures, SP500 future prices.* According to Ghorbel, A. and Jeribi, A. (2021) [12], current conditional volatilities of stock indices (SP500, Nasdaq, and VIX), gold, and oil depend not only on their own past volatility but also on past volatilities of cryptocurrencies. Therefore, it would be interesting to integrate similar exogenous data within the current input data. In this case it could also be useful to introduce some dimensionality reduction technique;

Bibliography

[1] Carol A. *Market Models, 1st edition.* John Wiley Sons, 2001. ISBN: 978-0-471-89975-4.

[2] LeCun Y. et al. "Object recognition with gradient-based learning". In: Springer Verlag, 1999, pp. 319–345.

[3] Yalamova et al. "Detecting chaos in financial time series". In: *Tech. Print* (2006), pp. 1–9.

[4] Ayo C. K. Ariyo A. Adewumi A. O. "Stock Price Prediction Using the ARIMA Model". In: *2014 UKSimAMSS 16th International Conference on Computer Modelling and Simulation* (2014).

[5] Neves R. F. Borges T. A. "Ensemble of machine learning algorithms for cryptocurrency investment with different data resampling methods". In: *Applied Soft Computing* (2020).

[6] Adamant Capital. *A Primer on Bitcoin Investor Sentiment and Changes in Saving Behavior.* URL: `https://medium.com/@adamant_capital/a-primer-on-bitcoin-investor-sentiment-and-changes-in-saving-behavior-a5fb70109d32`. accessed: 13.07.2021.

[7] Huang C. P. Chen J. F. Chen W. L. "Financial Time-Series Data Analysis Using Deep Convolutional Neural Networks." In: *2016 7th International Conference on Cloud Computing and Big Data* (2016).

[8] Puell D. *Bitcoin Average Dormancy.* URL: `https://medium.com/@kenoshaking/bitcoin-average-dormancy-28f88ea4c2ce`. accessed: 12.07.2021.

[9] Fernandes R. A. S. Dennys C. A. Mallqui. "Predicting the direction, maximum, minimum and closing prices of daily Bitcoin exchange rate using machine learning techniques". In: *Applied Soft Computing* 75 (2018).

[10] Anne Haubo Dyhrberg. "Hedging capabilities of bitcoin. is it the virtual gold". In: *Finance Research Letters* 16 (2016), pp. 139–144.

[11] Stambaugh R. F. French K. R. Schwert G. "Expected Stock Returns and Volatility". In: *Journal of Financial Economics* 19 (1987), pp. 3–29.

[12] Jeribi A. Ghorbel A. "Investigating the relationship between volatilities of cryptocurrencies and other financial assets". In: *Decisions Econ Finan* (2021), pp. 1–9.

[13] Au B. Greaves A. "Using the Bitcoin Transaction Graph to Predict". In: *Applied Soft Computing* (2015).

[14] Kim H. Y. Kim T. "Forecasting stock prices with a feature fusion LSTM-CNN model using different representations of the same data." In: *PLoS ONE* 14 (2019), e0212320.

[15] Hosking J. R. M. "Fractional differencing". In: *Biometrika* (1931), pp. 165–176.

[16] Ivanov M. *Financial Machine Learning Part 1: Labels*. URL: `https://towardsdatascience.com/financial-machine-learning-part-1-labels-7eeed050f32e?gi=9f8dab388ce0`. accessed: 16.07.2021.

[17] Leibowitz M. *Introducing: Fee Ratio Multiple (FRM)*. URL: `https://medium.com/coinmonks/introducing-fee-ratio-multiple-frm-1eada9ac9bec`. accessed: 11.07.2021.

[18] Caton S. McNally S. Roche J. "Predicting the Price of Bitcoin Using Machine Learning". In: *26th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP)* (2018).

[19] Unal D. Mudassir M. Bennbaia S. "Time-series forecasting of bitcoin prices using high-dimensional features: a machine learning approach". In: *Neural Computing and Applications* (2020), pp. 1–15.

[20] Carter N. *Bitcoin as a Novel Economic Istitution*. URL: https://www.docdroid.net/FbgH1WS/bitcoin-institution-riga-pdf. accessed: 12.08.2021.

[21] Satoshi Nakamoto. "Bitcoin: A Peer-to-Peer Electronic Cash System". In: *Cryptography Mailing list at https://metzdowd.com* (2008).

[22] Dittmar R. Neely C. Weller P. "Is technical analysis in the foreign exchange market profitable? A genetic programming approach". In: *Financ Quant Anal.* 43 (1997).

[23] Pereira A. de Oliveira R. Nelson D. "Stock market's price movement prediction with LSTM neural networks". In: *2017 International Joint Conference on Neural Networks (IJCNN)* (2017), pp. 1419–1426.

[24] Mehmet U. G. Ahmet M. O. Omer B. S. "Financial time series forecasting with deep learning : A systematic literature review: 2005–2019". In: *Applied Soft Computing* 90 (2020), p. 106181.

[25] Swift P. *Bitcoin Realized HODL Ratio*. URL: https://positivecrypto.medium.com/bitcoin-realized-hodl-ratio-9023db15a559. accessed: 13.07.2021.

[26] PlanB. *Modeling Bitcoin Value with Scarcity*. URL: https://medium.com/unconfiscatable/introducing-sopr-spent-outputs-to-predict-bitcoin-lows-and-tops-ceb4536b3b9. accessed: 13.07.2021.

[27] de Prado M. L. *Advances in Financial Machine Learning*. Wiley Publishing, 2018. ISBN: 978-1-119-48208-6.

[28] Shirakashi R. *Introducing SOPR: spent outputs to predict bitcoin lows and tops*. URL: https://medium.com/unconfiscatable/introducing-sopr-spent-outputs-to-predict-bitcoin-lows-and-tops-ceb4536b3b9. accessed: 13.07.2021.

[29] Reginald S. "Bitcoin Average Dormancy: A Measure of Turnover and Trading Activity". In: *Electronic Journal* (2017).

[30] Murat O. Sezer O. "Algorithmic Financial Trading with Deep Convolutional Neural Networks: Time Series to Image Conversion Approach". In: *Applied Soft Computing* (2018).

[31] Jaydip S. Abhishek D. Sidra M. "Stock Price Prediction Using Machine Learning and LSTM-Based Deep Learning Models". In: (2020).

[32] Baur D. G. Dimpfl T. "The volatility of Bitcoin and its role as a medium of exchange and a store of value". In: *Empir Econ* (2021).

[33] Blummer T. *Liveliness of Bitcoin*. URL: `https://medium.com/@tamas.blummer/liveliness-of-bitcoin-174001d016da`. accessed: 12.07.2021.

[34] Chohan U. W. "The Double Spending Problem and Cryptocurrencies". In: *SSRN* (2021).

[35] Woo W. *Experiments on Cumulative Destruction*. URL: `https://woobull.com/experiments-on-cumulative-destruction/`. accessed: 12.07.2021.

[36] Woo W. *Introducing the Difficulty Ribbon, signaling the best times to buy Bitcoin*. URL: `https://woobull.com/introducing-the-difficulty-ribbon-the-best-times-to-buy-bitcoin/`. accessed: 13.07.2021.

[37] Huang H. Wang X. Wang B. "A novel text mining approach to financial time series forecasting." In: *Neurocomputing* 83 (2012), pp. 136–145.

[38] Dai H. Yang L. Zheng Z. "Enhancing bitcoin price fluctuation prediction using attentive lstm and embedding network". In: *Applied Sciences* (2020).