I wish to thank everyone who has helped, followed, and mentored me along the journey. Thank you for the strength and the courage that never allow me to give up. Thank you for your patience and for your commitment that always reminds me how much you love and care about me. This would have never been achievable without you.

Table of Contents

1 Introduction to Price Movement Forecasting

Machine Learning in the financial industry sits at the intersection of several emergents and settled disciplines such as pattern recognition, time series forecasting, financial econometrics, statistical computing, probabilistic and dynamic programming. With the ongoing trend towards increasing computational power and larger datasets, machine learning has grown into a central computational engineering field focusing on placed plug-and-play algorithms available to scientists through open-source machine learning modules.

The surge of machine-accessible data to record and communicate activities in the financial system combined with the rapid development of computing capabilities and data storage have had necessary implications on every aspect of the financial world. Since the financial crisis of 2007-2008 (Flood et al. 2016 [7]), regulatory entities have stressed the need for more accurate data and information from bank loans to trading stress-testing books. For this reason, the financial industry is permeated with high volumes of data of different nature and usage. Financial data can be categorized into two different subsets of data: ”traditional” and ”alternative”. The former type of data refers to securities pricing, companies fundamentals, and micro or macro indicators that professionals have used for decades. However, in recent years, social networks and web browsers have enhanced the spectrum of information that final users can access: tweets, news, images, videos, IOTs, global capital flows are considered ”alternative” data because they entail the reliability on new sources of information that must be stored, analyzed and interpreted departing from the old perspective of data analysis (de Prado *et al.*, 2019 [29]).

Traditionally the focus of financial practitioners was merely related to linear regression, logistic regression and autoregressive iterated moving average (ARIMA) or generalized autoregressive conditional heteroscedasticity (GARCH) time series models. These models aim to predict the price level of the given financial security of the time series’s future real-

ization. These traditional statistical methods assume the linear generation of time series and do not account for data variability in non-linear contexts. Trading data such as closed prices and volumes highly depend on several exogenous micro and macro factors such as consumers behaviour, unpredictable news, monetary and fiscal policies. Caginalp and DeSantis [2] illustrate how financial markets dynamics are governed by non-linear factors. Moreover, they assess how the total return (reward of an investing strategy, μ) and volatility (dispersion of returns for a single financial security, σ) relationship is strictly non-linear. It is clear that the traditional linear assumption, under which time series are generated, is just a trivial approximation of a complicated environment like the stock market is; therefore, the need for other methodologies arises.

Forecasting the direction of stock prices plays a substantial role in determining whether to buy or sell financial security; thus, generating profits for many hedge funds, investments banks, and other financial institutions may be a non-trivial task. More than 56% of quantitative funds worldwide [27] have introduced AIs specifically built to analyze massive amounts of data to create and foster investing strategies or optimize asset allocation and portfolio management. This approach merges Fundamental Analysis (FA [28]) and Technical Analysis (TA [25]). The former measures a firm's intrinsic value by assessing economic and financial factors; the latter uses data to gather statistical trends on trading activities such as price movement and volume. The peculiar aspect of such is related to the fact that this analysis, traditionally performed by financial experts, are now exclusively performed by algorithms with little need of human interaction and intervention (besides the practical implementation of them, of course).

Machine Learning algorithms have been adapted to the stock market to several degrees of sophistication with various computational models (Cavalcante *et al.*, 2016 [6]). These algorithms are used in financial securities analysis, especially in price level forecasting for stocks, ETFs (exchange-traded funds) and options. Artificial Neural Networks (ANN), evolutionary algorithms, Support Vector Machines (SVM) are among the easiest and most implemented machine learning methods. Accordingly, due to the great success of deep learning algorithms in non-linear fields, financial practitioners have gradually begun to intro-

duce models such as Recurrent Neural Networks (RNN) (Krauss and Do and Huck, 2017 [13]), Convolutional Neural Networks (CNN) (Chen *et al.*, 2016 [9]) and Long-Short Term Memory (LSTM) (Fischer and Krauss, 2017 [10]) to analyze economic and financial data. Deep Learning pipelines are revealed to perform better than traditional Machine Learning algorithms [18, 15]. An ensemble methodology constituted by Machine and Deep Learning models stacked together is a common way to enhance the performances of the overall proposed model, e.g. SVM-ANN [5], CNN-SVM [17], CNN-LSTM [14]. However, developed deep learning tools for financial data analysis are still limited in number. Thereby, this paper focuses on the practical implementation of deep learning models for forecasting financial securities prices.

In this study, Machine Learning (ML) and Deep Learning (DL) architectures are implemented and compared on the classification problem of forecasting the price direction of financial securities. The focus has been redirected to forecasting price levels for a given security at a given point in the future; this can be achieved by using traditional statistical methods or forecasting machine learning methods. However, it turns out to be an arduous task as many refer to the stock market as a Random Walk problem where it is practically impossible to guess the next step of the ”walk”. Therefore, we turn our attention to the problem of classifying the movement of financial securities (long or short, up or down) in a specified time frame, and we leave out from the study the forecast of future prices of the underlying. Prices movement classification can be achieved through the implementation of ML and DL classification algorithms cited previously.

The idea underneath the analysis is straightforward. We take the time series of the prices (returns as well) for given financial security. We empirically divide the observations into two classes: the first group will be identified as ”up” or ”long”, and the second group will be identified as ”down” or ”short” (see 2.2.1). The algorithm has to be trained in order to recognize occurrences belonging to one of the two classes and infer predictions on future occurrences whose belonging class is not known in advance. In order to tackle this problem, a hybrid model, inspired by the one proposed by Yang, Xhai, Tao (2020) [20], is used to classify the financial security a day ahead. The deep neural network architecture

can be divided into two main parts: a CNN layer to extract features from the historical series and an LSTM to identify the time series's short and especially long-term trends. The model is built on historical daily close price data of the widely known World financial indexes ranging from 2000 to 2020 (included).

The remaining sections of this paper are developed as follows. In section 2, data are presented, engineered and loaded for the training and testing of the models. In section 3, ANN, RNN, CNN and LSTM models are theoretically presented to the reader. In section 4, the architecture of the final model is discussed. In section 5, the results of the experiments are given in comparison to other architectures. In section 6, conclusions are drawn.

2 Data Collection, Preparation and Loading

2.1 Data Collection

The decision to use Open-High-Low-Close-Volume (OHLCV) data of 10 main World financial indexes such as S&P 500, DJI, NASDAQ Etc., is based on the possibility to account for the overall trends of the world's economy, assess more significant variability in the data as well as generalizing the study itself. The world's indexes average several stocks whose single performances contribute to the performance of the overall index; therefore, the index itself reflects how the stocks belonging to it behave. It is relevant to mention that each index collects stocks of a given kind based on country of origin, market capitalization, the industry of belonging. Nonetheless, it is possible to study and forecast price movements for stocks and ETFs as well. The idea of using only indexes comes from the adoption of Yang, Xhai, Tao (2020) [20] as standard, and it is not binding at all.

For each stock index data are gathered leveraging Python coding and the open-source API of Yahoo Finance. Up to this point, we have collected three dataframes for each of the 10 Stock Indexes that will be used for data preparation purposes.

Each dataframe comprises a *Date* column with unique observations, *High* and *Low* columns in which the highest and lowest price of the security, for a given observation, is reported. *Open* and *Close* refer to the market opening and closing prices of the security. *Volume* refers to the trading volumes of the securities, the number of shares that are exchanged in the market at a given point in time. Finally, *Adj Close* is the closing price adjusted for any further corporate action that might have been taken, reflecting a different price than the closing one.

The NASADAQ 100 dataframe's head for daily data is shown in Figure 1.

Date	High	Low	Open	Close	Volume	Adj Close
2000-01-03	3836.860107	3643.25000	3755.739990	3790.550049	1510070000	3790.550049
2000-01-04	3766.570068	3542.72998	3766.570068	3546.199951	1511840000	3546.199951
2000-01-05	3576.169922	3371.75000	3543.129883	3507.310059	1735670000	3507.310059
2000-01-06	3513.550049	3334.02002	3488.310059	3340.810059	1598320000	3340.810059
2000-01-07	3529.750000	3314.75000	3337.260010	3529.600098	1634930000	3529.600098

Figure 1: Head of NASDAQ 100 dataframe

2.2 Data Preparation

Data Preparation is among the most challenging aspects when it comes to model deep learning architectures. Enriched data engineering can enhance the models' performances and deliver valuable and more accurate insights into the analyzed data. According to Forbes [26], data preparation tasks account for 59% of the total time spent by data scientists on a project, given the fact that data loading is so crucial and delicate at the same time.

This study aims to deploy an ensemble deep learning architecture of CNN-LSTM; CNN is widely used to recognize and detect images, while LSTM is used for the long-short trend of a series of data. The data preparation and data loading section, according to referential

literature [20, 19], consists in the creation of a three-dimensional input tensor that will be given as an ”image” to the CNN in order to recognize instances and extract useful features that the LSTM layer will later use to account for trends in the historical realizations of the observations collected.

In 2.2.1, 2.2.2 and 2.2.3 the three main data preparation techniques are explained and exemplified while in 2.2.4 we show the procedure for 3D input tensor building.

2.2.1 Observations Labelling

The task of forecasting the stock price movement often entails the categorization of each observation in two classes: ”up” and ”down”. In their publication, Hoseinzade and Haratizadeh (2019) [19] conduct a price movement forecasting analysis using a two label framework to classify observations. Gunduz, Yaslan and Cataltepe (2017) [11] conduct a similar analysis on the Turkish stock market where intraday observations are divided again into two distinct classes. According to these studies, we will conduct a binary classification analysis through ensemble deep learning architectures on daily, weekly and monthly close prices data.

Class targets exemplify the direction of the movement of the stock price and the grouping can be done empirically following a straight-forward deterministic reasoning. Let C_t be the close price of the stock index on observation t, the class label relative to observation t is define as follows:

$$\text{label}_t = \begin{cases} 1, & C_{t+1} > C_t \\ 0, & C_{t+1} \leq C_t \end{cases} \tag{1}$$

A given observation is identified as 1 (”up”, ”long”) if the closing price of the following observation is higher than the closing price of the observation considered. Vice versa, an observation of the time series is labelled as 0 (”down”, ”short”) if the closing price of the subsequent observation is lower than the closing price of the observation considered. It must be noted that observations are labelled following a deterministic method, no calcula-

tions are performed, no machine learning or deep learning models are applied to separate instances, but a mere comparison, easily achievable, is performed on time series data.

2.2.2 Transformation of Deterministic Signals

It is well known that technical indicators ("technicals") play a substantial role when used for stock prices forecasts (Neely *et al.*, 2014 [4]). Technical indicators [22] are deterministic and heuristic pattern-based signals produced by price, volume and open interest relative to given financial security merely used for technical analysis. In this paper, we will deploy ten leading technical indicators converted into deterministic trend signals. Jiar *et al.* (2015) [5] show that deterministic values of technicals are better suited for price movements forecasting instead of using the native values from the indicators themselves.

Technical indicators are mathematical functions used to calculate the indicator's value for an associated observation. Each unique observation in the dataset has an associated value relative to each of the ten technicals, which will be converted into a deterministic binary signal using simple comparisons. The detailed formulation of the ten technical indicators can be found in Table 1 while Table 2 presents the conversion from technical indicators' values to binary signals.

During the phase of data preparation, the technical indicators parameters are chosen manually, the n period that is used by the indicators is set to 10.

Indicators name	Formulas
Simple Moving Average	$SMA_t = \frac{1}{n}\sum_{i=1}^{n} C_{t-i}$
Weighted Moving Average	$WMA_t = \frac{1}{\sum_{i=0}^{n-1} n-i}\sum_{i=1}^{n}(n-i) * C_{t-i}$
Momentum	$MOM_t = C_t - C_{t-n}$
Stochastic K%	$K_t = (C_t - LL_{t-(n-1)}/HH_{t-(n-1)} - LL_{t-(n-1)}) * 100$
Stochastic D%	$D_t = \sum_{i=0}^{n-1} K_{t-i}/n$
Moving Average Convergence Divergence	$MACD_t = MACD_{t-1} + \frac{2}{(n+1)(DIFF_t - MACD_{t-1})}$ $DIFF_t = EMA(12)_t - EMA(26)_t$
Relative Stremght Index	$RSI_t = 100 - \frac{100}{1+\frac{\sum_{i=0}^{n-1} UP_{t-i}/n}{\sum_{i=0}^{n-1} DW_{t-i}/n}}$
William's %R	$WR_t = (H_n - C_t/H_n - L_n) * 100$
Commodity Channel Index	$CCI_t = M_t. - SM_t/0.015D_t$, $M_t = H_t + L_t + C_t/3$ $SM_t = \sum_{i=1}^{n} M_{t-i+1}/n$, $D_t = \sum_{i=1}^{n} \lvert M_{t-i+1} - SM_t\rvert/n$
Accumulation Distribution Oscillator	$AD_t = (H_t - C_{t-1})/(H_t - L_t)$

C_t , L_t , and H_t refers the close price, low price, and high price at time t, respectively; LL_t and HH_t represent, respectively, lowest low and highest high in the last t days; UP_t means upward price change while DW_t is the downward price change at time t. EMA refers to the exponential moving average, $EMA(k)_t = EMA(k)_{t\text{-}1} + \alpha * (C_t - EMA(k)_{t\text{-}1})$, $\alpha = 2/k + 1$, and k denotes the time period of k−day exponential moving average.

Table 1: Detailed description of technical indicators

Indicators name	Deterministic trend signals rules
Simple Moving Average	If $C_t > SMA_t$, label 1; otherwise, label 0
Weighted Moving Average	If $C_t > WMA_t$, label 1; otherwise, label 0
Momentum	If $MOM_t > 0$, label 1; otherwise, label 0
Stochastic K%	If $K_t > K_{t-1}$ label 1; otherwise, label 0
Stochastic D%	If $D_t > D_{t-1}$ label 1; otherwise, label 0
Moving Average Conv. Div.	If $MACD_t > MACD_{t-1}$ label 1; otherwise, label 0
Relative Strength Index	If $RSI_t <= 30$ or $RSI_t > RSI_{t-1}$, label 1; If $RSI_t >= 70$ or $RSI_t < RSI_{t-1}$, label 0;
William's %R	If $WR_t > WR_{t-1}$ label 1; otherwise, label 0
Commodity Channel Index	If $CCI_t < -200$ or $CCI_t > CCI_{t-1}$, label 1; If $CCI_t > 200$ or $CCI_t < CCI_{t-1}$, label 0;
Accumulation Distribution Oscillator	If $AD_t > AD_{t-1}$ label 1; otherwise, label 0

Table 2: Rules for converting technical indicators into deterministic trend signals

2.2.3 Correlation Between Assets

Correlation [24] is a way to interpret the statistical and logical relationship between financial securities; it identifies the degree of motion of the two securities. Correlation is a value that ranges between -1 and +1: correlation towards -1 entails a negative correlation where the two financial securities move in opposite directions; if the correlation moves to +1 suggest

accordance in the movement, while a 0 correlation means no correlation at all but just random movements.

Gunduz, Yaslan and Cataltepe (2017) [11] show how ordinate features, rather than randomly chosen, as input tensor increases the performances of the CNN. Accordingly, we will use the Pearson product-moment correlation coefficient (PPMCC) to define the stocks index ranking that will be applied to the creation of the input tensor. The detailed PPMCC formula (formula 2) is reported below:

$$\rho_{x,y} = \frac{\sum_{t=1}^{n}(x_t - \bar{x})(y_t - \bar{y})}{\sqrt{\sum_{t=1}^{n}(x_t - \bar{x})^2 \sum_{t=1}^{n}(y_t - \bar{y})^2}} \tag{2}$$

where x,y represent the x-th and y-th feature on the t-th observation. $\bar{x}$, $\bar{y}$ represent the averages of the previous n observation. In the study, the correlation time-delta between indexes is ten as the length of the period for the input cube. For each instance, correlation is not fixed, recalculated, and the correlation order is kept constant within the cube, but it can vary along with the time series.

2.2.4 Input Tensor Building

From the datasets collected *Open*, *Volume*, *Adj Close* are immediately deleted as these columns will not be used for the analysis. For each Stock Index, the above mentioned technical indicators in 2.2.2 are computed for each observation, and the resulting value is appended to the dataframe underneath a specific column referring to the technical indicator itself. We now have 10 dataframes (converted to *.csv* format for practical purposes) with 14 columns each where the *Date* column is the reference column for each unique observation, 3 (*High*, *Low*, *Close*) columns are from the original dataframe and the remaining 10 have just been appended being the columns referring to the technical indicators.

The following task consists in building the 3D-image Figure 2. The image (*x*,*y*,*z*), is a cube whose dimensions are all equal to 10. Furthermore, it is unique, and each cube is associated with it respective observation in the dataset. Each 3D-image is constructed to represent each observation on the timeline, and it will be filled up with 0s and 1s. From

Figure 2 it can be noticed the presence of white and black dots representing 0s and 1s, respectively. The dimensions of the input tensor (*x*,*y*,*z*) represent the ten stock indexes chosen, the ten technical indicators defined and the previous ten realizations to the associated observation. Therefore, each image establishes a relationship between each observation studied and ten trading instances prior to the realization of the observation itself to study and hopefully justify the price movement in one of the two directions.

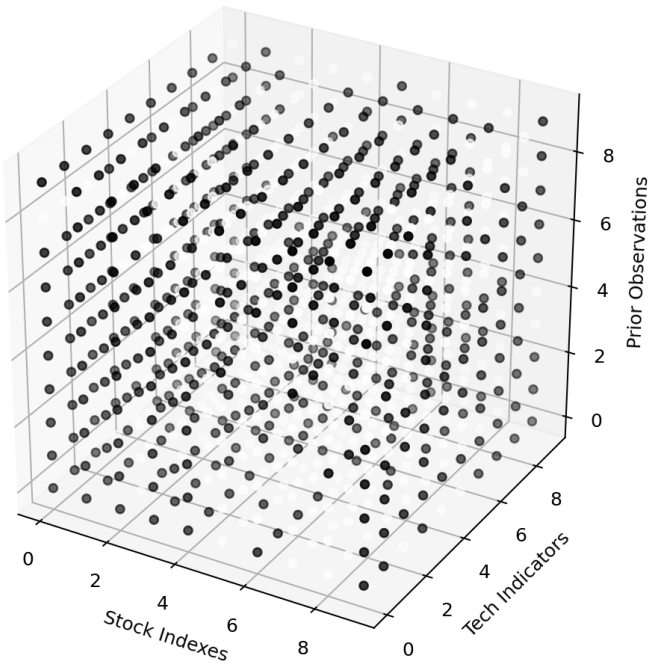

Figure 2: 3D-image of Input Tensor

The following represents the pseudocode for the creation of the input tensor and the associated label (Algorithm 1).

We firstly instantiate two null arrays: $\overline{tensor}$ will contain the 3D images for each observation, $\overline{labels}$ will be filled with the corresponding labels of each observation. The first loop runs from the first observation up to the last, and it is the main loop as it keeps the position i both for $\overline{tensor}$ and $\overline{labels}$. Inside this first loop, we set two variables: $label$ is the resulting label from the deterministic comparison explained in 2.2.1 and $assets$ the ordered list of stock indexes to the S&P 500, obtained through PPMCC explained in 2.2.3. Three nested

Algorithm 1 Input Tensor and Label building

$\overline{tensor} \leftarrow$ null array
$\overline{labels} \leftarrow$ null array
for $i \leftarrow 1$ to N **do** ▷ N are the total days in the dataframe
  $label \leftarrow$ label of i obs. (2.2.1)
  $assets \leftarrow$ list of the ranked indexes (2.2.3)
  for $j \leftarrow 1$ to M **do** ▷ M days used to construct 3D image
    for k in $assets$ **do** ▷ use each ordered stock index at the time
      for z in T **do** ▷ T are the technical indicators
        $\overline{tensor}[i, j, z, k] \leftarrow$ binary signal (Table 2)
      end for
    end for
  end for
  $\overline{labels}[i] \leftarrow label$
end for
return $\overline{tensor}, \overline{labels}$

loops of equal length (10 in our case) are needed to fill the 3D image with deterministic binary signals (see Table 2) coming from the chosen technical indicators explained in 2.2.2 and detailed reported in Table 1. Finally, at the end of these three loops, we save the corresponding label of the i-th observation before iterating again on the main loop. After the main loop as concluded, we simply return $\overline{tensor}, \overline{labels}$.

2.3 Data Loading

Data Loading refers to the procedure of injecting data into the model for training and testing purposes. Model architectures belonging to the field of Machine and Deep Learning require observations to learn from and acquire knowledge regarding a particular task, in our case, the prediction of price movements for stock indexes. Subsequently to the training phase, the architecture is tested on unseen data called testing data. For this reason, we are going

to split the dataframe into training and testing data. Given that the data we are studying is a financial time-series, data cannot be shuffled, and the observations presented to the models must be in chronological order; otherwise, the inferred predictions would be vague without a helpful meaning. We will study the effectiveness and reliability in the prediction of the models following a common-knowledge approach: we are going to train over some years and then we are going to test the models on the following year, the structure of the market keeps changing and evolving therefore is impossible to aspire to predict it through non-updated models, it would just be a failure. We are going to train it iteratively, over multiple time frames backtesting the model on five years chosen at random, to mimic what scientists would have done years back.

2.3.1 Time-Series Cross Validation

The practice of cross-validation is a fundamental aspect in the field of Machine and Deep Learning. Cross-validation allows the model to learn during the training phase with static hyperparameters and update them on an embedded set in the training data called validation set. The hyperparameters chosen in the validation set, obtained through cross-validation techniques, are ultimately used in the test set to infer better and more accurate predictions.

The cross-validation technique that has been performed is the Blocking-Time-Series. The model is trained and validated on a particular instance that would not be seen by the model in the following epoch. For this reason, the model experiences only a portion of data at the time, updating the hyperparameters at every iteration. We are studying a completely different scenario than a ”traditional” machine learning train/test experiment, and the results obtained in one test rather than another cannot be directly compared. Time series evolve and behave differently from time to time. The stock market is so rapidly evolving that untrained and unprepared models for the right and specific time points are useless.

3 Theoretical Frameworks

This section provides a concise theoretical explanation of the main machine and deep learning models used in the study: Artificial Neural Networks (ANNs), Convolutional Neural Networks (CNNs) and Long-Short Term Memory (LSTM).

3.1 Artificial Neural Network and Perceptron

An Artificial Neural Network [3] is a machine learning model whose learning approach is inspired by the functioning of biological neurons in processing, understanding and transferring information (Figure 3). ANNs comprise multiple layers of single units called perceptrons or, more easily, neurons, which are in charge of the functioning of the entire system.

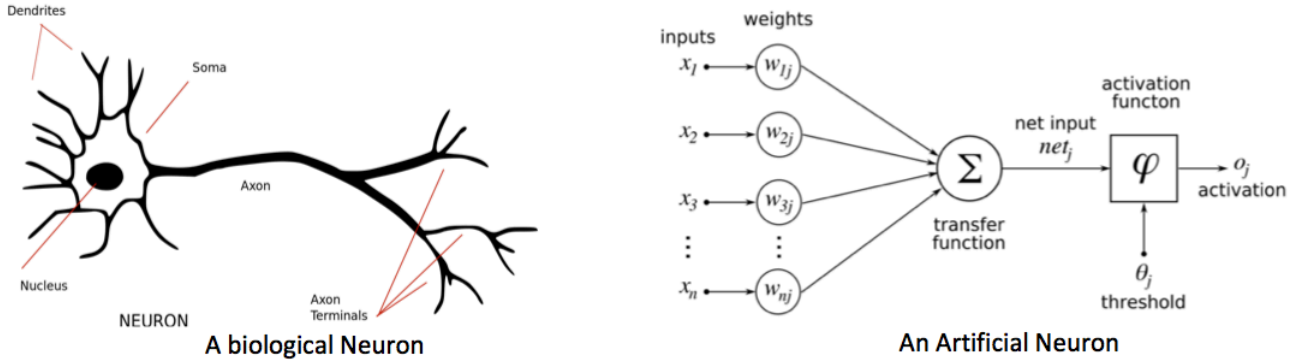

Figure 3: Comparison between biological and artificial neurons

A biological neuron is connected to previous ones through receptors called ”dendrites”, which are structures in charge of gathering information in the surrounding area of the neuron. The main structure of the biological neuron is the ”soma” that collects inputs, processes information and generates an output signal that, through the ”axon”, will reach other neurons continuing with the information flow. In comparison, each artificial neuron is composed of several inputs (numerical values) where a piece of information is stored. A numerical value called weight strengthen or soften the connection between previous neurons and the following one. A transfer function is applied to collect all the input signals with their corresponding weights. Moreover, an activation function is used on the transfer

function, determining the value of the output. If the activation function's value exceeds a certain threshold, the neuron will "spark" or "activates", passing information to the next Perceptron. This process is called "feed-forward", as information is passed from one layer to the next without going backwards in a recursive manner.

The Perceptron is the most basic neural network structure as it is characterized by a unitary or single structure. The Perceptron receive as input a vector $(x_1, x_2, ..., x_n)$ with associated weights $(w_1, w_2, ..., w_n)$ for each entry. The simplest transfer function $\nu = \sum_{i=1}^{n} w_i x_i$ is applied to get the resulting value from the input values and weight vectors. The net input is passed to a linear activation function called "step function" (equation 3):

$$\phi(\nu) = \begin{cases} 1, & \nu \geq \theta \\ 0, & \nu < \theta \end{cases} \tag{3}$$

If the value of $\phi(\nu) = 1$, the Perceptron is activated, whereas it is not activated if the value is equal to 0. From the output, generation is possible to grasp that the Perceptron can only deal with linearly separable binary data. However, if stacked together, many Perceptrons together form a feed-forward ANN, the Multi-Layer Perceptron (MLP), which is able to deal with non-linearly separable data [1].

The Perceptron learning phase consists in an iterative process of modification of the weight vector $(w_1, w_2, ..., w_n)$ in order to correctly classify every instance or at least the great majority of data. Every observation is given to the perceptron singularly and if the predicted label $\phi(\nu)$ is a correct prediction than no weight modification is performed. If the perceptron prediction is not correct than the weight vector is updated following the perceptron learning rule: $\bar{w_i} = w_i + \eta(y_i - \hat{y_i})x_i$, where $\bar{w_i}$ is the new weight for input i, w_i is the current weight, η is the learning rate of the model, y is the prediction of the perceptron, $\hat{y}$ is the correct prediction of the instance and x_i is the numerical value of input i.

3.2 Convolutional Neural Network

Convolutional Neural Network (CNN) [8] is a class of deep learning neural networks that is highly versatile in the field of image, video recognition, natural language processing (NLP), text mining and financial time series analysis. Once again, this Neural Network architecture takes inspiration from the biological processes where the visual cortex of animals is used as a standard to recreate a model that would mimic its behaviour. Moreover, Haoyue and Chenwei (2021) [21] have assessed that, according to their findings, the best models to predict and represents the brain's activities are, in fact, Convolutional Neural Networks and Recurrent Neural Networks. CNN's peculiar ability to be space and time-invariant has fostered these architectures' usage for recognition and pattern detection tasks. Moreover, the fundamental aspect of these networks resides in the reduction of data pre-processing and features localization as the architecture is built so that these tasks are performed internally.

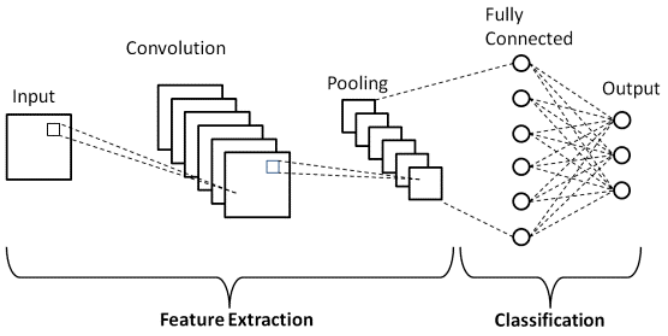

Figure 4: Typical CNN architecture [23]

The CNN typical structure is illustrated in Figure 4, and it comprises three main structures: a Convolutional Layer, a Pooling Layer and a Fully Connected Layer.

The convolutional layer takes an image called "tensor" as input, and transforms it into a feature map or activation map performing convolutional operations on it. Essentially, the convolutional operation entails sliding the kernel over the input matrix and applying matrix multiplication among the local neighbours to the pixel considered. Two important aspects of kernel multiplication are the stride value and the padding. The former refers to the posi-

tions that have to be shifted when performing matrix multiplication; the latter is a practice that places 0s on the borders of the input image so that relevant information on the edges are not lost when the convolution is applied. The pooling layer is adopted at particular stages in the CNN architecture to reduce the data load of the entire process, reducing computational complexity. Pooling consists in selecting specific information and neglecting others by applying a well-specified ruling. The Max Pooling methodology is the most common one where among particular neighbours, the highest value pixel is passed onto the next layer. However, Yang *et al.* (2018) [16] demonstrated that applying a pooling layer to a financial time-series would cause information loss; thereby, we would not adopt this practice for the model construction phase (Section 4). The last block of the CNN architecture is the fully connected neural network where information is exchanged not only forwardly but also in the backward direction. The presence of a convolutional neural network allows for non-linear dependencies recognition in data with a subsequent output layer with as many neurons as data labels shall be classified.

3.3 Long-Short Term Memory

LSTM networks represent a substantial step forward in the recognition of time-dependent sequences. LSTMs have been introduced to overcome Recurrent Neural Networks (RNNs) difficulties in solving the long-term dependencies in data. RNNs are not suitable for long time series in which information must be carried along from beginning to end to predict the subsequent realization of the sequence efficiently. Therefore LSTM networks, a particular form of RNN nets, were introduced by Hochreiter and Schmidhuber in 1997 to deal with long-period structures hidden in data efficiently, nowadays are widely used for various purposes.

In Figure 5 each black arrow carries a vector from the previous node's output to the following node's input. The blue circles represent point-wise operations performed on the vectors. The boxes are instead Neural Network Layers that learn on the data that pass through. Moreover, we can define four main internal structures of the LSTM cell: a forget

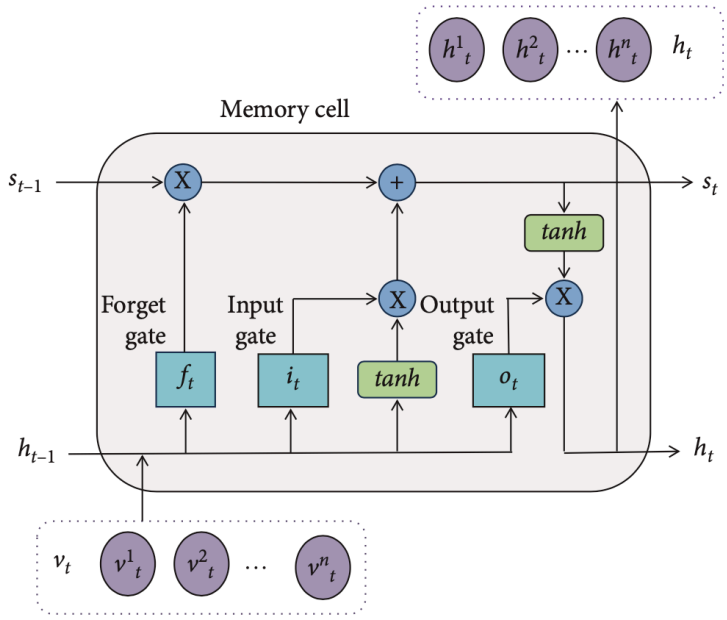

Figure 5: LSTM cell [20]

gate, an input gate, a cell state (s) and an output gate.

The Forget gate (f_t) decides which information must be truncated from the sequence and which shall be passed onto the next cell. Piece of information from the previous hidden state h_{t-1} together with data at the current stage ν_t^i is fed to the Sigmoid function that outputs a value between 0 and 1. If the value outputted from this function is close to 0, then the network will forget the data received; otherwise, if the value is close to 1, the information will be kept. The second structure is the input gate that works as an updater for the cell state. h_{t-1} and ν_t^i are passed again into the Input gate (i_t) characterized by a Sigmoid function to decide if the information shall be kept or lost. Subsequently, the same data are given to the Tanh (green box) function to regularize values between -1 and +1, allowing for better network regulation. As it can be seen the output coming from the Input gate and Tanh are multiplied together through vector multiplication. The cell state is identified by the black line on the top part of the LSTM cell and conveys the state in which the structure is at a given point in time. The previous cell state s_{t-1} is point-wise multiplied by the Output of the Forget gate (if the value of the forget gate is close to 0, then the immediate value of the cell state tends to 0 as well). After that, the cell state becomes

s_t by vector addition operation between the updated old cell state and the value spitted out from the operations of the network. This vector operation conveys its new value to the cell state. The last structure is the Output gate (o_t) which is in charge of determining the hidden state h_t for the cell at time t. The previous observation hidden state h_{t-1} is passed into the Sigmoid function while the cell state s_t just found is passed into the Tanh. These two values are point-wise multiplied together, and the result is the hidden state for the current observation h_t which is then used for forecasting. In Figure 5 it can be noticed that the top arrow exiting the cell is s_t while the bottom one is the hidden state h_t which is passed to the next cell and it is outputted from the cell as prediction for the observation considered.

4 Deep Learning Architectures

This section includes the six models that have been built for the study. The models' building phase takes inspiration from existing models. Subsequently, the models are directly refined on the data used through manual hyper-parameters twisting and cross-validation techniques. The values for the hyperparameters reported in this section are the "base-line" parameters used for the initial analysis. During each cross-validation phase, each model is specifically tuned.

1. **ANN-1** The first model is a simple feed-forward Neural Network composed of four layers with a pyramidal structure. The model takes as data the input tensor built with 3D images, it flattens the structure of the data, and then the first layer of 100 neurons is applied, subsequently one of 50, one of 25, and finally a layer composed of 2 neurons in charge of outputting the categorical label for the binary output. The first layers adopt the ReLu activation function in the structure, while the output layer uses a Sigmoid activation.

2. **CNN-ANN-1** The second model deployed is the CNN structure proposed in [20] with the difference that it does not include any feed-forward part. The model is only com-

posed of a CNN merged with a Dense layer of 2 neurons (Sigmoid activation function) deployed for binary classification of the instances. The CNN part comprises four layers: the first layer is a parallel convolutional layer in which there is two separate Convolutional structure that performs operations on the input data with a kernel size of (10,1,1) and (10,3,3) respectively. The second layer is composed of a Concatenate layer in which the resulting images from the previous two convolutions are unified together. The third layer is again a Convolutional layer with kernel size (10,3,5). Lastly, we have another Convolutional layer with kernel size (10,3,1). The resulting data is flattened and given to the Dense layer for immediate classification. The CNN number of filters equals 10, the activation function is ReLu, padding is set to ”same,” and batch size is 32.

3. **CNN-ANN-2** This model is a combined model of the ones described in 1 and 2. It uses the CNN structure to extract the features from the images and flattens the information fed to an ANN with four layers.

4. **CNN-LSTM-ANN-1** The fourth is a deep learning model composed of a CNN structure, an LSTM framework, and a shallow bi-neuron dense layer. The model proposes a CNN as the one described in 2, with a four-layer structure, with the first is characterized by parallel Convolutional operations. Once the features are extracted from data, an LSTM network with 150 units and a ReLu activation function is fed to keep track of evolving features and temporal structures. Finally, a dense layer with two neurons (Sigmoid activation function) is enabled to undertake predictions on the observations.

5. **CNN-LSTM-ANN-2** The last model realized is a CNN-LSTM-ANN model. Data are given to the CNN for feature extraction; the CNN framework is the same in all models. Once the CNN has concluded its operations, a 50 unit LSTM network is deployed, followed by an ANN feed-forward layer composed of four layers with 100, 50, 25, and 2 units each. The activation function for the ANN is ReLu in all the layers as it is for CNN and LSTM structures. This model proposes an advance to the primary model in 4 by implementing a Neural Network structure after the LSTM network.

6. **META-MODEL** We have decided to create an ulterior model that takes decisions on a voting procedure base. The meta-model labels the instance according to a majority voting rule for every trading day in the test set. In this way, it is possible to recreate a model that takes decisions according to the most chosen action of the other models. The model aims to express the average sentiment of the models in the testing years since, as shown in the next section, it is almost impossible to find a steady best-performing one.

In the empirical analysis, we deploy all models on different time-frames. According to exhaustive empirical research, having a testing set of more than one year reduces the models' performances as they are not adapted to the rapidly evolving environment in which they perform the classification tasks. There is no empirical evidence about which model shall be used. All models behave poorly in some instances and behave well in others. For this reason, we have decided to implement the synthetic model, which should average the others' predictions as we cannot conclude which model is the best performing one. In the next section, we aim to compare the results obtained by the meta-model strategy in terms of base metrics and return concerning a Buy-Hold NASDAQ strategy.

5 Empirical Results

The NASDAQ is presented as a benchmark because the meta-strategy presented studies the hidden patterns of NASDAQ returns. Moreover, it is also considered the most straightforward strategy for an investor who wants to expose him/herself to the stock market without any prior academic knowledge. The ultimate goal of a trading strategy is to beat the benchmark's returns for the period in which the strategy is deployed. The testing period ranges from 2017 to 2020, and the models have been tested on these years for three different windows: year, semester, and quarter. We do not know in advance the optimal strategy for model retraining, and for this reason, we aim to find the best-suited time window to allow the models to better adapt to the new market conditions. Due to computa-

tional complexity issues, all models have been trained on each time-frame for 100 epochs with simple cross-validation. The Binary Cross-Entropy loss function is minimized using the Adam optimizer, a standard loss function optimizer used in computer vision and deep learning [12]. The models are built with Python coding language leveraging open-source libraries such as Keras and Tensorflow.

5.1 Models' Metrics

Before assessing the final results of the model and the performances obtained on each of the years listed above, we introduce the mathematical formulation of the metrics used to evaluate the models. In the analysis, we have adopted as loss function the Binary Cross Entropy or Log Loss (equation 4) which is the most common loss function used for binary classification problems:

$$H(p) = -\frac{1}{n}\sum_{i=1}^{n} y_i log(p(y_i)) + (1-y_i) log(1-p(y_i)) \tag{4}$$

We immediately see that the probability of belonging to the positive class (1) is log weighted to represent the entropy of a single class. y_i is the predicted label for each observation, $p(y_i)$ is the probability associated with the observation's target. In a nutshell, $H(p)$ takes every observation multiplying being positive class to the associated log probability and sums the log probability of belonging to the negative class. Low values for the loss function indicate good predictions and vice versa.

We also use four different metrics to assess ulterior models' performances: Accuracy, Precision, Recall, and F1-Score; Accuracy is the most important and straightforward. Accuracy reflects the proportion of instances correctly classified over the total observations.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

Precision is a measure of quality; it reflects the percentage of correct, true observations classified as positive over the total observations that the model has predicted as positive.

$$\text{Precision} = \frac{TP}{TP + FP}$$

The Recall is a measure of quantity; it accounts for how many observations are labeled as positive over the total number of observations belonging to the positive class.

$$\text{Recall} = \frac{TP}{TP + FN}$$

F1-Score is a combined measure of Precision and Recall, it is the harmonic mean of these two metrics and it reflects the test's accuracy.

$$\text{F1-Score} = 2 * \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$$

5.2 Results

In 2017 the synthetic model did not beat the NASDAQ benchmark, which had an annual return of 30%. However, the model achieved with the yearly window an annual return of 8%, an annual return of 10% with the semester strategy, and an 8.5% re-tuning the algorithm every three months. The synthetic model achieved in the monthly time window an accuracy of 0.54, precision of 0.63, Recall of 0.63, and an F1-score of 0.62.

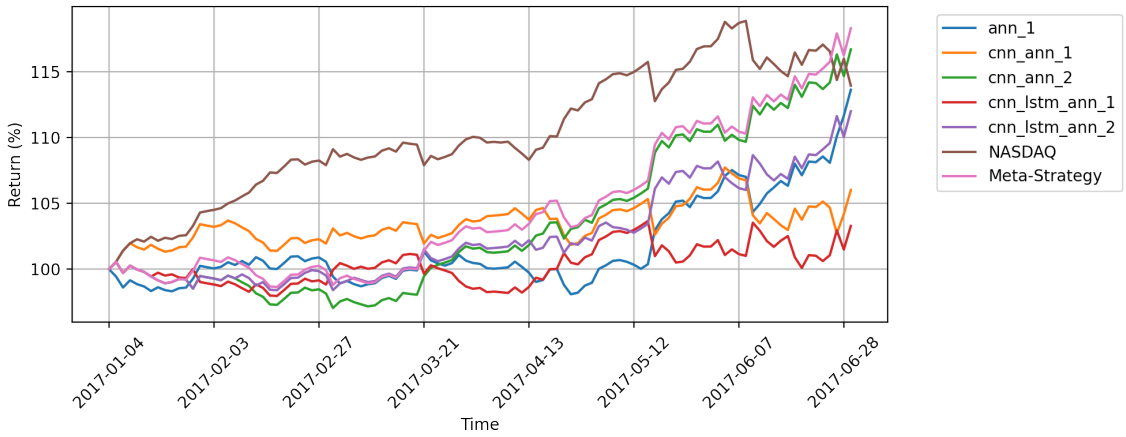

Figure 6: 2017 first semester

The year 2018 is a turning point as the proposed model can beat the NASDAQ return that for this year is attested to be -9%. The annual return for the year and the quarter-time window is -10% and -4.5%, respectively. However, the annual return for the semester time window strategy is 5% with an initial return of 7% (Fig. 7) followed by a negative return

of 2% in the second semester; accuracy is 0.53, precision is 0.53, Recall is 0.70 and F1-score is 0.61. Accuracy decreased by 0.01, precision by 0.1, recall increased by 0.07, and F1-score decreased by 0.01.

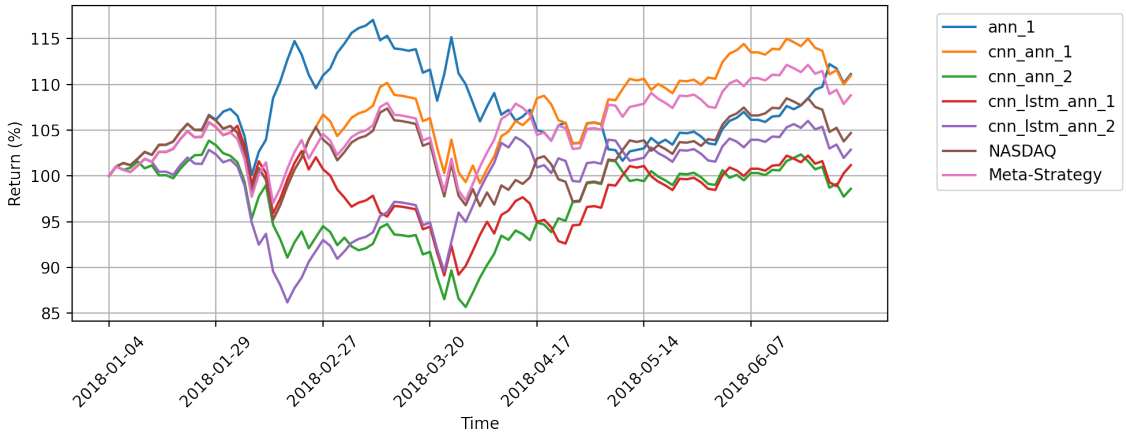

Figure 7: 2018 first semester

In 2019 the strategy did not beat the benchmark again because the annual return of the NASDAQ was around 33% while the best performing strategy, which again is the semester one, only returned 13%. The annual strategy returned 12%, while the quarterly strategy had an annual return of 10%. The best performing strategy had an accuracy of 0.56, precision of 0.58, Recall of 0.85, and F1-score of 0.72.

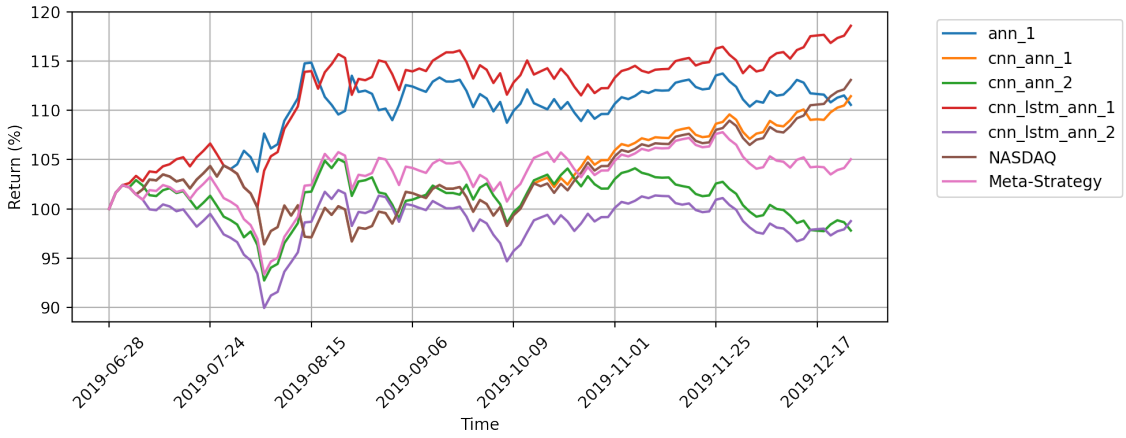

Figure 8: 2019 second semester

In 2020, last testing year, the models performed well despite Covid-19 turbulence in the market. Both the semester and quarter strategy outperformed the benchmark with an-

nual returns of 36% and 29%, respectively. The NASDAQ had an annual return of 25%, being more than 20% down in March. The exciting part is that in both semester and quarter strategies, the models understood the tumbling of the stock market and started to go ”short”, and they turned out to be correct. For this reason, the synthetic model could beat the benchmark with two strategies. In this case, the best strategy, the quarter ones, had an accuracy of 0.58, precision of 0.64, Recall of 0.6, and F1-score of 0.62. Incredibly, ANN-1 concluded the year with a 150% annual return as it correctly classified 63% of the observations in the first semester. The first semester was characterized by extremely high volatility that ANN-1 precisely leveraged, obtaining a 120% return just in the first half of the year.

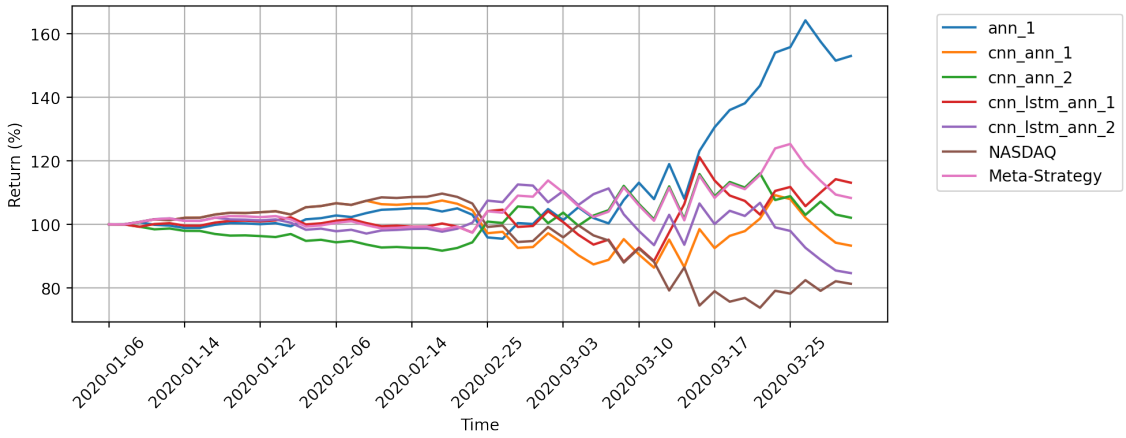

Figure 9: 2020 first quarter

Considering the results obtained, we could consider the semester strategy for the synthetic algorithm a decent strategy even though the proposed strategy did not systematically beat the benchmark, a challenging and ambitious goal. However, it is a valuable tool, especially in moments of general plummeting for the stock market. The NASDAQ index had great performances in the four years considered. It almost constantly grew, and when it crashed, it recovered the losses immediately, unlike the other global indexes presented. For this reason, the meta-strategy could not follow the NASDAQ performances in the first three years, characterized by extremely high returns. The model behaved well in the periods of downturn, recovering from moments of negative tracking, resulting in a decent model overall.

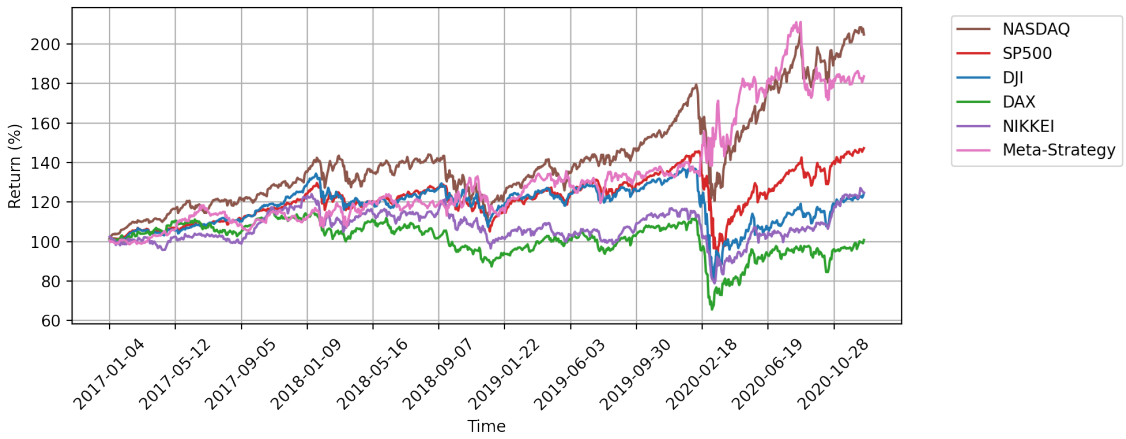

Figure 10: Meta-model cumulative return

In Figure 10 it is possible to grasp that the proposed model underperformed the NASDAQ for the first year and a half; it subsequently regained territory when the NASDAQ crashed in late 2018. The model did not follow the skyrocketing performances of the index in 2019 again. At the beginning of 2020, the index lost 30% while the model leveraged betting against the index, which turned out to be a correct prediction. The proposed model concluded the four-year testing period with a cumulative return of 80%. Moreover, the model's volatility and drawdown are lower than the benchmark's, suggesting a lower return for lower volatility in the strategy proposed.

6 Conclusions and Further Analysis

The study presents a novel algorithmic trading strategy that can leverage deep learning for the binary classification problem of stock price movement forecasting. We have first introduced the field of study as well as models deployed for similar tasks. We then have introduced the logical and mathematical framework underlying the crucial data preparation procedures for the input tensor building. Moreover, we have introduced theoretical concepts behind the structure and realization of the models adopted in the study for a more precise and comprehensive framework. The model is structured so that it can enter ”long” and ”short” positions, thus earning, not only when the index returns positively but also in scenarios when the index has negative, consistent returns. The results of the proposed meta-model are presented, showing that the cumulative return of the strategy did not beat the benchmark's return for the testing period of 2017-2020. However, the model performed better in periods of global downturn, for example, in the second part of 2018 and during the Covid-19 March crash. Accordingly, the meta-strategy would have revealed a performing strategy in the first semester of 2020, when the whole stock market was plummeting. We have also highlighted that the best performing meta-model is obtained through a six-month rolling period of testing with subsequent retraining for better adaptation to the environmental changes.

There is plenty of work to undertake next. It would be interesting to study non-daily observations and test the performances of the proposed model. We could develop a deeper structure for modifying and adapting the technical indicators' parameters, which are now chosen manually. Furthermore, we could seek new indicators and opt for a better deterministic ruling regarding the data preparation part. The meta-model could be further improved with a more sophisticated and computationally expensive cross-validation structure and an improved voting system for the meta-model with the introduction of a weighted average. This problem could also be studied for other financial securities implementing a complex and diversified trading portfolio that could improve the overall performances and metrics obtained so far.

Bibliography

[1] PRAMOD GUPTA and NARESH K. SINHA. “Neural Networks for Identification of Nonlinear Systems: An Overview”. In: *Soft Computing and Intelligent Systems*. Ed. by NARESH K. SINHA and MADAN M. GUPTA. Academic Press Series in Engineering. San Diego: Academic Press, 2000, pp. 337–356. ISBN: 978-0-12-646490-0. DOI: https://doi.org/10.1016/B978-012646490-0/50017-2. URL: https://www.sciencedirect.com/science/article/pii/B9780126464900500172.

[2] Gunduz Caginalp and Mark DeSantis. “Nonlinearity in the dynamics of financial markets”. In: *Nonlinear Analysis: Real World Applications* 12.2 (2011), pp. 1140–1151. ISSN: 1468-1218. DOI: https://doi.org/10.1016/j.nonrwa.2010.09.008. URL: https://www.sciencedirect.com/science/article/pii/S1468121810002282.

[3] Amit D.Kachare A.D.Dongare R.R.Kharde. “Introduction to Artificial Neural Network”. In: *International Journal of Engineering and Innovative Technology (IJEIT)* 2.1 (2012), pp. 189–194. URL: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.1082.1323&rep=rep1&type=pdf.

[4] Christopher J. Neely et al. “Forecasting the Equity Risk Premium: The Role of Technical Indicators”. In: *Management Science* 60.7 (2014), pp. 1772–1791. DOI: https://doi.org/10.1287/mnsc.2013.1838. URL: https://pubsonline.informs.org/doi/abs/10.1287/mnsc.2013.1838.

[5] Jigar Patel et al. “Predicting stock market index using fusion of machine learning techniques”. In: *Expert Systems with Applications* 42.4 (2015), pp. 2162–2172. ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2014.10.031. URL: https://www.sciencedirect.com/science/article/pii/S0957417414006551.

[6] Rodolfo Cavalcante et al. “Computational Intelligence and Financial Markets: A Survey and Future Directions”. In: *Expert Systems with Applications* 55 (Feb. 2016). DOI: 10.1016/j.eswa.2016.02.006.

[7] Mark Flood, H. V. Jagadish, and L. Raschid. “Big data challenges and opportunities in financial stability monitoring”. In: *Financial Stability Review* 20 (2016), pp. 129–142. URL: https://EconPapers.repec.org/RePEc:bfr:fisrev:2016:20:14.

[8] Saad Albawi, Tareq Abed Mohammed, and Saad Al-Zawi. “Understanding of a convolutional neural network”. In: *2017 International Conference on Engineering and Technology (ICET)*. 2017, pp. 1–6. DOI: 10.1109/ICEngTechnol.2017.8308186.

[9] Jou Fan Chen et al. “Financial time-series data analysis using deep convolutional neural networks”. English. In: Proceedings - 2016 7th International Conference on Cloud Computing and Big Data, CCBD 2016 (July 2017). null ; Conference date: 16-11-2016 Through 18-11-2016, pp. 87–92. DOI: 10.1109/CCBD.2016.027.

[10] Thomas Fischer and Christopher Krauss. “Deep learning with long short-term memory networks for financial market predictions”. In: 11/2017 (2017). URL: https://ideas.repec.org/p/zbw/iwqwdp/112017.html.

[11] Hakan Gunduz, Yusuf Yaslan, and Zehra Cataltepe. “Intraday prediction of Borsa Istanbul using convolutional neural networks and feature correlations”. In: *Knowledge-Based Systems* 137 (2017), pp. 138–148. ISSN: 0950-7051. DOI: https://doi.org/10.1016/j.knosys.2017.09.023. URL: https://www.sciencedirect.com/science/article/pii/S0950705117304252.

[12] Diederik P. Kingma and Jimmy Ba. “Adam: A Method for Stochastic Optimization”. In: (2017). arXiv: 1412.6980 [cs.LG].

[13] Christopher Krauss, Xuan Anh Do, and Nicolas Huck. “Deep neural networks, gradient-boosted trees, random forests: Statistical arbitrage on the SP500”. In: *European Journal of Operational Research* 259.2 (2017), pp. 689–702. ISSN: 0377-2217. DOI: https://doi.org/10.1016/j.ejor.2016.10.031. URL: https://www.sciencedirect.com/science/article/pii/S0377221716308657.

[14] Sneh Jain, Roopam Gupta, and Asmita A. Moghe. “Stock Price Prediction on Daily Stock Data using Deep Neural Networks”. In: (2018), pp. 1–13. DOI: 10.1109/ICACAT.2018.8933791.

[15] Hiransha M et al. “NSE Stock Market Prediction Using Deep-Learning Models”. In: *Procedia Computer Science* 132 (2018). International Conference on Computational Intelligence and Data Science, pp. 1351–1362. ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2018.05.050. URL: https://www.sciencedirect.com/science/article/pii/S1877050918307828.

[16] Hui Yang, Yingying Zhu, and Qiang Huang. “A Multi-indicator Feature Selection for CNN-Driven Stock Index Prediction”. In: *Neural Information Processing*. Ed. by Long Cheng, Andrew Chi Sing Leung, and Seiichi Ozawa. Cham: Springer International Publishing, 2018, pp. 35–46. ISBN: 978-3-030-04221-9.

[17] Jiasheng Cao and Jinghan Wang. “Stock price forecasting model based on modified convolution neural network and financial time series analysis”. In: *International Journal of Communication Systems* 32 (May 2019), e3987. DOI: 10.1002/dac.3987.

[18] Yingxuan Chen, Weiwei Lin, and James Z. Wang. “A Dual-Attention-Based Stock Price Trend Prediction Model With Dual Features”. In: *IEEE Access* 7 (2019), pp. 148047–148058. DOI: 10.1109/ACCESS.2019.2946223.

[19] Ehsan Hoseinzade and Saman Haratizadeh. “CNNpred: CNN-based stock market prediction using a diverse set of variables”. In: *Expert Systems with Applications* 129 (2019), pp. 273–285. ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2019.03.029. URL: https://www.sciencedirect.com/science/article/pii/S0957417419301915.

[20] Guihua Tao Can Yang Junjie Zhai. “Deep Learning for Price Movement Prediction Using Convolutional Neural Network and Long Short-Term Memory”. In: *Mathematical Problems in Engineering* 2020.Article ID 2746845 (2020), pp. 1–13. DOI: 10.1155/2020/2746845.

[21] Haoyue Yan and Chenwei Wu. “Analysis of A Mixed Neural Network Based on CNN and RNN for Computational Model of Sensory Cortex”. In: *2021 International Conference on Electronics, Circuits and Information Engineering (ECIE)*. 2021, pp. 245–249. DOI: 10.1109/ECIE52353.2021.00059.

[22] JAMES CHEN. *Technical Indicator*. URL: https://www.investopedia.com/terms/t/technicalindicator.asp.

[23] MK Gurucharan. *Basic CNN Architecture: Explaining 5 Layers of Convolutional Neural Network*. URL: https://www.upgrad.com/blog/basic-cnn-architecture/.

[24] ADAM HAYES. *Correlation*. URL: https://www.investopedia.com/terms/c/correlation.asp.

[25] ADAM HAYES. *What Is Technical Analysis?* URL: https://www.investopedia.com/terms/t/technicalanalysis.asp.

[26] Gil Press. *Cleaning Big Data: Most Time-Consuming, Least Enjoyable Data Science Task, Survey Says*. URL: https://www.forbes.com/sites/gilpress/2016/03/23/data-preparation-most-time-consuming-least-enjoyable-data-science-task-survey-says/?sh=370ecfa56f63.

[27] Peter Salvage. *Artificial Intelligence Sweeps Hedge Funds*. URL: https://www.bnymellon.com/us/en/insights/all-insights/artificial-intelligence-sweeps-hedge-funds.html.

[28] Troy Segal. *Fundamental Analysis*. URL: https://www.investopedia.com/terms/f/fundamentalanalysis.asp.

[29] Marcos López de Prado. “Estimation of Theory-Implied Correlation Matrices”. In: *SSRN* (November 9, 2019). DOI: http://dx.doi.org/10.2139/ssrn.3484152.