Natural Computing
https://doi.org/10.1007/s11047-020-09829-9

Stock market trend detection and automatic decision-making through a network-based classification model

Tiago Colliri[1] · Liang Zhao[2]

Accepted: 26 November 2020

Abstract
Many complex systems observed in nature and society can be described in terms of network. A salient feature of networks is the presence of community patterns. Network-based models have already been applied in the analysis of data from very diverse areas, from epidemics modeling to periodicity detection in meteorological data. In this paper, inspired by the formation of community structures, such as the metabolic networks and the anatomical and functional connectome observed in biological neural networks, we present a model which makes use of connector hubs to detect price trend reversals and to automatize decision-making processes in stock market trading operations for selecting a good investment strategy and improve the returns. It starts by mapping the historical stock price time series as a network, where each node represents a price variation range and the edges are generated according to the time sequential order in which these ranges occur. Afterwards, communities of the constructed network so far are detected, which represent the *up* and *down* trends of the stock prices. The model has two phases: (1) Trend detection phase, where the price trend communities are detected and trend labels are generated; and (2) Operating phase. In this phase, the proposed technique predicts trend labels to future stock prices, in such a way that these trends can be used as triggers to perform buying and selling operations of the stock. We evaluate the model by applying it on historical data from 10 of the most traded stocks from both NYSE and the Brazilian Stock Exchange (Bovespa). The obtained results are promising, with the model's best returns being able to outperform the stock price returns for the same period in 15 out of the 20 cases under consideration.

Keywords Complex networks · Machine learning · Classification · Trend detection · Decision making · Stock market

1 Introduction

Many *complex systems* in nature and society can be described in terms of networks to capture the intricate web of connections among the units they are made of Palla et al. (2005), Watts and Strogatz (1998), Barabási and Albert (1999), Dorogovtsev and Mendes (2013). A network (or *graph*) consists of a collection of *nodes* (or vertices) joined by *links* (or edges). A salient feature of networks is the presence of community patterns, one of the examples is the biological neural networks. Data on both anatomical and functional connectome of human (animal) brain has shown the small-world structure with highly clustered modules at different scales (Akiki and Abdallah 2019; Gleiser and Spoormaker 2010; Hagmann et al. 2008). These communities are known to represent subsystems of neurophysiological functions, e.g., visual cortex. Besides the brain (Sporns 2002), other examples of real world networks include the internet (Faloutsos et al. 1999), food chains (Montoya and Solé 2002), blood distribution networks (West et al. 2009) and power grid distribution networks (Albert et al. 2004). Mathematically, a network can be defined as a graph $G = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is a set of nodes and $\mathcal{E}$ is a set of tuples representing the edges between pairs of nodes $(i,j) : i,j \in \mathcal{V}$. Although complex network is a relative new field of study, there are currently already several network-based models developed for *machine learning* tasks, such as *clustering* (Silva et al. 2013; Liu et al. 2018), *classification* (Silva and Zhao 2012; Colliri

✉ Tiago Colliri
tcolliri@usp.br

[1] Institute of Mathematics and Computer Science, University of Sao Paulo, São Carlos, Brazil

[2] Faculty of Philosophy, Science, and Letters, University of Sao Paulo, Ribeirão Preto, Brazil

Published online: 03 January 2021

et al. 2018; Silva and Zhao 2015; Carneiro and Zhao 2017) and *regression* (Gao et al. 2014; Loglisci and Malerba 2017; Casiraghi 2017).

Stock market prediction assumes that it is possible to determine the future value of a company stock or other financial asset traded on an exchange. Although there are numerous works published on this subject, the existence of such possibility in the stock market is still a controversial matter. In economics, the origin of the *random walk hypothesis* can be traced back to the works of Regnault (1863) and Bachelier (1900). These works argue that stock market prices evolve according to a random walk and thus cannot be predicted. This idea, along with concepts proposed by Hayek (1945), has been later incorporated in the *efficient market hypothesis* (Malkiel and Fama 1970), which states, broadly speaking, that the price of an asset fully reflects all available information in the market and, consequently, predicting market movements would be an impossible task to be accomplished. *Technical analysis*, on the other hand, is known as the study of financial market's historical data with the purpose of forecasting future price trends (Wang and Chan 2007). It, hence, contradicts the efficient market hypothesis (Roberts 1959) by believing that it is indeed possible to identify trending patterns within the historical prices of an asset in short-term and even in long-term periods.

The use of "network thinking" (Watts 2004; Barabási 2003) when dealing with complex systems in the real world can help to better understand the phenomenon, by analyzing its topological features (Mitchell 2006). In vaccination strategies, for instance, the immunization of hubs in the network is more likely to slow the spread of a disease than choosing random individuals to vaccinate (Pastor-Satorras and Vespignani 2002). A similar type of reasoning can also be applied to other important tasks, such as managing public policies for controlling epidemics (Pastor-Satorras and Vespignani 2001), mitigating the effects of power failures (Motter and Lai 2002) and viruses spread in computers (Wang and Chen 2003). Additionally, there is the concept of *functional cartography*, introduced by Guimera and Amaral (2005), based on observations that metabolic networks in organisms may also present community structure, in which each node plays a different "role" or "sub-role", according to their pattern of intra- and inter-module connections. In this manner, according to this concept, a hub could be classified into three sub-roles: *provincial* (the vast majority of the node's links are within the node's module), *connector* (the node is both a hub in its module and has many links to most other modules) or *kinless* (the node's links are homogeneously distributed among all modules).

Inspired by the concept of functional cartography and the formation of community structures, observed in natural and man-made systems, we present a model which makes use of connector hubs to detect *price trend reversals* in the market, thus allowing the model to detect the starting point of *up* or *down* trends for a stock, as well as triggering a buying or a selling operation accordingly. It starts by, in the *trend detection phase*, mapping the stock price variation ranges into a network and then classifying theses nodes as being more or less likely to indicate an *up* or *down* trend in the stock price, according to the network's community structure. Afterwards, in the *operating phase*, the model propagates these labels to future prices, also triggering buying or selling operations for the stock, accordingly. Through the identification of hubs connecting the network's communities, the model detects a pattern formation representing what is known in the stock market as a trend reversal, such that the return obtained from each trade can be improved. For evaluating its efficiency, the model is applied to the historical prices of 10 of the most traded stocks from NYSE and from Bovespa Stock Exchange, and the obtained results are encouraging, with the best returns of the proposed model being able to outperform the stock price returns for the same period in 15 out of the 20 considered cases.

Data classification is a common task, which can be performed by both computers and human beings. However, a fundamental difference between them can be observed: computer-based classification considers only physical features (e.g., similarity, distance, or distribution) of input data; on the other hand, brain-based classification takes into account not only physical features, but also the semantic meaning or the pattern formation of data. Therefore, the proposed model is nature inspired in two aspects: (1) in the data representation, by mapping the stock prices as a network and making use of the functional cartography concept, and (2) in the data processing, inasmuch as the decision making technique proposed in this work is also inspired by human (animal brain) in such a way that the classification is made according to the pattern formation of the data beyond the physical features. In this case, the community structure of the price variations' network is used to detect the price trend patterns for the stock.

This model was first introduced in Colliri and Zhao (2019) and, for this extended version, we update the historical prices of the original database obtained from Bovespa, used in the model's evaluation. Moreover, we introduce the historical prices of 10 of the most traded stocks from NYSE (New York Stock Exchange). The Methodology and Experimental Results sections have also been improved, with new figures and more detailed descriptions, in order to enhance their overall content. The extensive experiments presented in this paper generate results which confirm the superior performance of the proposed model.

Regarding the organization of this paper, besides this introduction, we have, in Sect. 2, a discussion about other works related to this study. In Sect. 3, we provide a detailed description of the model, showing how it generates the network and the labels from the input data and also the algorithms used in its tasks, both for the trend detection and the operating phases. In Sect. 4, we explain the methodology used for obtaining the experimental results. In Sect. 5, we present the results obtained by applying the model to real financial time series from NYSE and Bovespa Stock Exchange. In Sect. 6, we conclude the study with some final remarks.

2 Related works

The origin of the *graph theory* traces back to 1736, when the Swiss mathematician Leonhard Euler published the solution to the Königsberg bridge problem (Boccaletti et al. 2006). This initial work led to the development of other important works, such as the ones made by Vandermonde (1771), Cauchy (1813) and Huillier (1861), which ended up resulting in the creation of a new branch of discrete mathematics, known as *topology*. Later, Erdös and Rényi (1960) introduced probabilistic methods in graph theory, while Cartwright and Harary (1956) helped to standardize the terminology of graphs, as well as to broaden its reach by including and demonstrating possibilities of applications in several branches of science. In 1995, the field of complex networks was included as one of the main topics of the area of complex systems, i.e., dynamical systems that present properties such as adaptation, emergence and transition. In 1998 and 1999, there were two papers responsible for great advances in this field, published by Watts and Strogatz (1998), on small-world networks, and by Barabási and Albert (1999), on scale-free networks. Currently, new studies in this field have the potential to offer novel tools and perspectives for a wide range of scientific problems, from social networking to drug design (Barabási 2016).

In recent years, the continuous advances in the area of *artificial intelligence* contributed to increasing the interest in using different approaches, such as *machine learning*, to treat the challenging problem of predicting financial data. In Huang et al. (2005), the authors have compared the performances of Support Vector Machines (Vapnik 2000) and other traditional classifiers on predicting the tendency of the NIKKEI index, obtaining favorable results by SVM. In Adebiyi et al. (2014), the authors have compared the performance of the statistical model ARIMA and artificial neural networks on forecasting the future direction of stock prices of NYSE, obtaining better results by the latter. Another interesting approach was made by Lee and Jo (1999), in which an expert system based on a knowledge base comprising candlestick's technical analyses was developed to predict future price movements for stocks, with the experimental results achieving an average hit ratio of 72% tested with Korean stock market data.

A network-based machine learning technique offers the advantage to not only analyze physical features of the input data (e.g., distance or distribution), as in traditional machine learning techniques, but to also consider the pattern formation in their topological properties (Colliri et al. 2018; Silva and Zhao 2012). Besides, as this is intrinsically a graphical approach, it has the convenience of being *interpretable*, which is not the case with other machine learning approaches, such as neural networks (Guresen et al. 2011). In Cao et al. (2019), a network was built from S&P 500, NASDAQ and DJIA indexes data and the topological measures are extracted and used as attributes for predicting the next-day patterns, obtaining an accuracy rate of over 70%. Another application in this sense was made by Anghinoni et al. (2019), which forecasts trends in stochastic time series from the Bovespa index through *unsupervised learning* techniques, based on community detection and network walk observations, and obtained an accuracy rate of over 90% in the predictions, outperforming traditional classifiers such as naive Bayes (Rish 2001), Decision Tree (Safavin and Landgrebe 1991) and Multilayer Perceptron (Hinton 1989).

The main motivation behind the development of such methods and techniques, aiming to predict future prices in the stock market, lies in the possibility of using them to correctly determine the exact time for buying and selling a stock. Although the determination of these "timings", when it comes to stock market decisions, may be subject to cultural differences among the investors (Ji et al. 2008), it is possible to affirm that they all share the same goal, which is to increase the return obtained from each trading operation. Within this context, the investor who makes use of the model presented in this paper has the benefit of not only being able to detect price trends, but also of being advised about when will be the most adequate moment to buy or to sell a stock, in order to improve his returns.

3 Model description

The model proposed in this study is summarized below.

3.1 Model overview

In *supervised learning* models, initially we have an input dataset comprising an array of attributes $X_{training}$ and an array of labels $Y_{training}$, and the model then has to learn

from these two arrays to predict (or classify) the instances from the array X_{test} in the testing phase. In our case, $X_{training}$ is provided and it comprises the price history of a stock, while $Y_{training}$, containing the labels, is not provided and must comprise the respective stock price trends for each closing price in $X_{training}$. So the first task of the model is to generate the labels $Y_{training}$ based on the data provided in $X_{training}$, i.e., the model must detect the price trends for the stock according to its historical prices. This is achieved in the *trend detection phase*, by first adding two columns (or attributes) in $X_{training}$: the price variation within a v-days sliding window and its respective discretized version, consisting of n price variation ranges. Each of these ranges is then mapped as a node in a network, and edges are created between each pair of variation ranges if they ever appeared consecutively in $X_{training}$. Next, a community detection algorithm is ran, and the labels $Y_{training}$, i.e., the stock price trends, will be the respective community to which each node belongs. Nodes from the lower ranges' community represent a *down* trend, while nodes from the higher ranges' community represent an *up* trend. Later, in the testing phase (here named as the *operating phase*), these labels are used to predict the future stock prices, triggering possible *buying* or *selling* operations for the stock, whenever the current price is represented by a connector hub from an upper or lower community in the network, respectively.

Note that it is possible to exist more than two communities in the network and, in this case, the model takes into consideration only the one with lower ranges and the one with upper ranges, for triggering buying or selling operations. A summary of the model's processing is illustrated in Fig. 1. It is also worth noting that the model only triggers a buying or selling operation when it detects what is known as a *trend reversal* pattern for the stock price, i.e., when the prices movement changes from a *down* trend to an *up* trend (triggering a *buying* operation) or, alternately, when the prices movement changes from an *up* trend to a *down* trend (hence triggering a *selling* operation).

In technical analysis, there is the known concept among investors of *support* and *resistance*, when analyzing chart patterns. The main idea is that these two indicators tend to act as "barriers", preventing the price of an asset from going up or down beyond these levels. Additionally, given the market dynamics, in the medium and long terms, a previous resistance level may become a support level, and vice-versa, when the price goes up or down beyond that level. The rationale behind the proposed model has some similarities with this type of analysis, with the main difference that, instead of measuring the support and resistance levels in terms of prices, the model determines them in terms of a sliding-window price variation, with the connector hubs of the network acting as some sort of support and resistance indicators, in this case. During the operating phase, the model assigns a label to the asset's current price according to the respective node that its variation range occupies in the network, which is built during the trend detection phase. Since the trend reversal patterns identified and learned by the model from this stock's historical price data in the training phase continues to reoccur in the stock's future prices, therefore, it is used for predicting purposes in the operating phase. For this reason, the model can get good results.

3.2 Trend detection phase

For conducting our experiments, we split each stock price history dataset X into two subdatasets: $X_{training}$ and X_{test}, according to a time range parameter *tdays*. The other two parameters required by the model are: v (measured in days, used as a sliding window for generating the price variations) and l (also measured in terms of days, is the length of the ranges for the discretization of the price variations). The trend detection phase can be summarized in the following steps below, in that order:

1. start with a dataset X, containing attributes *date* and *closing price* for the stock, and insert column *var-v* showing the discretized closing price variations within a v-days sliding window;
2. split dataset X into two parts according to the previously stipulated number of training days *tdays*, such that we have $X_{training} = X_{[:tdays]}$ and $X_{test} = X_{[tdays:]}$;
3. sort $X_{training}$ in ascending order by attribute *var-v*;
4. stipulate n variation ranges r of the same length l and categorize each variation value in $X_{training}^{var-v}$ into its respective range r, generating attribute R in $X_{training}$;
5. revert $X_{training}$ to its original order by *date* attribute, and generate adjacency matrix A, representing the constructed network G, such that each range r in X^R may become a node in a network (Algorithm 1);
6. detect communities in G.

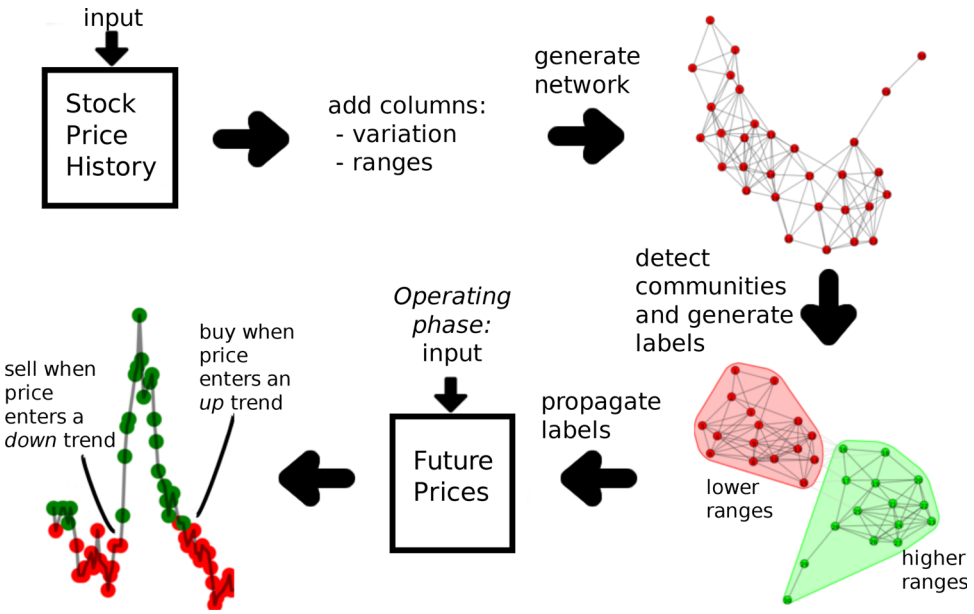


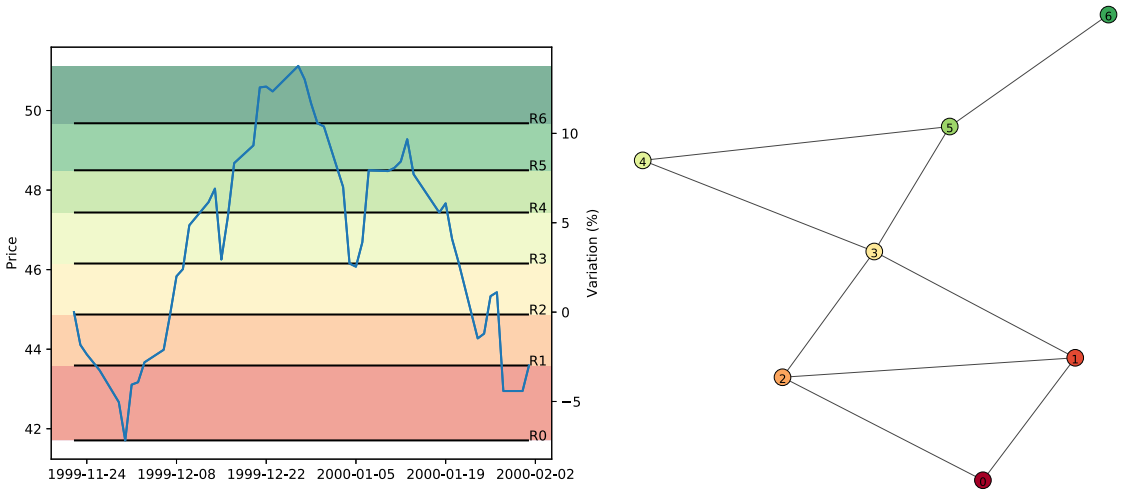
Fig. 1 Overview of the model. From the input stock price history, two columns are added: the price variation within a v-days sliding window and its respective discretized values, in the form of n variation ranges. Then, a network is generated, where each node represents a variation range and the edges denote whether the ranges ever appeared consecutively in the stock price history, pairwise. The community detection algorithm has the role of labeling the ranges into an *up*, *down* or one of the possible in-between trends for the stock price (in this example there are only two possibilities: *up* or *down*, because there are only two communities). In the operating phase, the labels are propagated to future prices, possibly triggering a buying or a selling operation accordingly, when it identifies a *trend reversal* pattern in the price series

Algorithm 1 Adjacency matrix A generation.

Input: price history with variation ranges X
Output: adjacency matrix A

procedure GET ADJACENCY MATRIX A(X)
  $n \leftarrow$ number of ranges stipulated for the variations in X
  $A \leftarrow$ zero matrix (n x n)
  $t \leftarrow$ length of X
  for $i = 1$ to t **do**
    $r1 \leftarrow X^R_{i-1}$
    $r2 \leftarrow X^R_i$
    if $r1 \neq r2$ **then**
      $A_{r1,r2} = 1$
      $A_{r2,r1} = 1$
  return A

The rules for generating the adjacency matrix A are depicted in Algorithm 1, in which X, in this case, stands for $X_{training}$. According to these rules, two vertices v_1 and v_2 in the network G, which represent ranges r_1 and r_2 in X^R, will be connected by a link only if they are immediately next to each other at any point in X^R. Since the values in X^R are ordered by dates in ascending order and given that they indicate how distant the current price is from the price of v days ago, then the position of the nodes in the network will be affected by the overall stock price oscillation, with subsequent and similar ranges tending to be connected and closer to each other (Fig. 2). This pattern will later be recognized by the community detection algorithm, which will group the nodes according to the "momentum" they represent in the current stock price oscillation trend (so to speak). It is worth noting that the Algorithm 1 was originally applied as the first step in a network-based model for detecting periodicity in time series, such as the ones from meteorological data (Ferreira et al. 2014), and here we are taking advantage of its rationale by applying it in the stock market context. During the operating phase, depending on the community to which the node of the current price variation range belongs, the model then infers whether it is an indication that the stock price is currently in an *up*, *down* or maybe in one of its *in-between* trends, when compared to its price from v days ago. If the community of the current price is different from the community of the previous day price, then a trend reversal may be detected, and a buying or selling operation is triggered for the stock.



(a) Price variation ranges (b) Network resulted

Fig. 2 Illustration showing how the price variation ranges are defined and later mapped as nodes in the network. For this example, we use the first 50 days of stock GE, from NYSE, and, for the sake of simplification, the daily price variations are calculated based on the initial price, instead of using a sliding window. **a** After being sorted in ascending order, the price variations are split into 7 equal parts (the square root of 50), thus delimiting the ranges R0 to R6. **b** Each range becomes a node in the network, and two nodes are connected if they ever appeared consecutively in the stock price time series. Note that node 2 is connected to node 0 due to the price drop happened on 2000-01-28

date	ticker	close	var-30	R	cm
1998-10-02	EXMP3	1.46	-35	1	0
1998-10-05	EXMP3	1.39	-34	1	0
1998-10-06	EXMP3	1.46	-27	1	0
1998-10-07	EXMP3	1.44	-26	1	0
1998-10-08	EXMP3	1.45	-20	2	0
1998-10-09	EXMP3	1.55	-2	12	0
1998-10-13	EXMP3	1.64	5	17	1
1998-10-14	EXMP3	1.63	9	19	1
1998-10-15	EXMP3	1.73	5	17	1
1998-10-16	EXMP3	1.74	-1	13	0
1998-10-19	EXMP3	1.78	7	18	1
1998-10-20	EXMP3	1.91	20	25	1
1998-10-21	EXMP3	2.09	38	29	1
1998-10-22	EXMP3	2.10	42	29	1
1998-10-23	EXMP3	1.96	60	30	1
1998-10-26	EXMP3	1.83	29	28	1
1998-10-27	EXMP3	1.80	13	21	1
1998-10-28	EXMP3	1.83	-3	12	0
1998-10-29	EXMP3	1.74	3	16	1
1998-10-30	EXMP3	1.88	30	28	1

(a) $X_{training}$ (b) Network (c) Communities

Fig. 3 An example of the trend detection phase for the stock EXMP3. **a** Fragment from the subdataset $X_{training}$. The columns *var-30*, *R* and *cm* represent the discretized 30-days closing price variations, the ranges used for categorizing the variations and the community to which each range *r* belongs (added later from *G*), respectively. **b** The network *G* resulted from the application of Algorithm 1, to generate the adjacency matrix *A*. **c** The communities identified in the network *G*. Note that, in this case, the green community represents the higher ranges (from 15 to 30) and hence indicates an *up* trend, while the red community contains the lower ranges (from 0 to 14) and hence indicates a *down* trend. (Color figure online)

The trend detection phase is illustrated in Fig. 3, where we have a stock EXMP3 with a sliding window v of 30 days for the price variation attribute. In this example, the training process resulted in 31 ranges r (labels 0 until 30) for categorizing the 30-days price variations, and the community detection algorithm returned 2 communities (or *clusters*) to be used as labels for the price trends. It is also important to observe in this figure that, here, the indexes and range values start at 0 instead of at 1, and so we proceed when writing the algorithms. Therefore, in case one wants to replicate our model and uses different indexing values, then he has to adapt the code indexes correspondingly.

Regarding the processing time, all processes required by the model in the trend detection phase have a linear time, hence with a time complexity of $O(n)$, with the only

exceptions being the sorting operations, which are of type $O(n \log n)$, where n may represent the size of the input dataset X, of subdataset $X_{training}$ or of the network G, depending on the task.

3.3 Operating phase

After having detected the trending patterns for the stock and properly assigned them as labels for the subdataset $X_{training}$, we then proceed to the testing phase, in which we propagate these labels to the future prices of the stock in X_{test}. Here, this phase is named as the *operating phase*, since depending on the labels assigned to the current prices, an operation of buying or selling may also be triggered for the stock. We stipulate some rules for buying or selling the stock according to which node, in the previously generated network G, the current stock price variation range is. There is only one parameter required by the model in this phase – the number of connector hubs h taken into account to identify possible reversals in the price trend. The operating phase of the model can be summarized in the following steps below, in that order:

1. add columns R and cm in X_{test} dataset by using the same conversion values between attributes *var-v*, R and cm (community) obtained from the trend detection phase;
   - in case a value from $var{-}\mathrm{v}_i^{test}$ is higher than $max(var{-}\mathrm{v}^{training})$, then its r_i value should be $max(R^{training})$;
   - in case a value from $var{-}\mathrm{v}_i^{test}$ is lower than $min(var{-}\mathrm{v}^{training})$, then its r_i value should be 0;
2. perform buying and selling operations for the stock according to the rules described in Algorithm 2; and
3. in case there is still an open position after processing the last row of X_{test} with Algorithm 2, this position then must be "closed" (sold) and its respective return should also be computed in the overall model's return for the stock.

Algorithm 2 Perform buy and sell operations.

Input: variation ranges network G, number of hubs considered h, prices list to operate X
Output: list with trading operations performed in X

```
procedure GET HUBS(G, h, cm1)
    if cm1 = 0 then
        cm2 ← 1
    else
        cm2 ← cm1 − 1
    nodes ← [ ]
    for v in G^vertices do
        if v in G^clusters_cm1 then
            nodes ← append v
            v^k ← 0
            for j in v^neighbors do
                if j in G^clusters_cm2 then
                    v^k += 1
    nodes ← sort descending by v^k
    return nodes_[:h]
procedure TRADE(X, G, h)
    n ← number of communities in G
    downhubs ← GetHubs(G, h, 0)
    uphubs ← GetHubs(G, h, n − 1)
    t ← length of X
    list ← [ ]
    flat ← True
    for i = 0 to t do
        if flat then
            if X_i^R in uphubs then
                list ← add dict with buying info (price, date)
                flat ← False
        else
            if X_i^R in downhubs then
                list ← add dict with selling info (price, date, return, length)
                flat ← True
    return list
```

The rules for buying and selling the stock during the operating phase are described in Algorithm 2, in which X, in this case, stands for X_{test}. The first step of the *Trade* procedure, in Algorithm 2, is to identify the h *downhubs* and the h *uphubs* in the network G, through the *GetHubs* procedure. The *downhubs* are the h vertices within the lowest community (label 0) with most links to vertices from its immediately higher community (label 1). These connector hubs thus indicate the beginning of a *down* trend for the stock price, according to our model. On the other hand, the *uphubs* are the h vertices within the highest community $(n-1)$ with most links to vertices from its immediately lower community $(n-2)$, where n is the number of communities in G. These connector hubs thus indicate the beginning of an *up* trend for the stock price, according to our model.

In the second step of the *Trade* procedure, in Algorithm 2, the model may execute a buying or a selling order according to the price's variation range (represented by a node in the trade data network), as well as its current position in the stock. The model performs a buying operation when it is currently *flat*, i.e. it is not already positioned in the stock, and the variation range (X_i^R) of the current price (X_i^{close}) is among one of the *uphubs*. Conversely, it performs a selling operation for the stock when it is currently positioned in the stock and the variation range (X_i^R) of the current price (X_i^{close}) is among one of the *downhubs*. When the current price variation range reaches one of the h "entry" connector hubs from the higher ranges' community, then it is a sign of *trend reversal* with the stock price entering in an *up* trend, so the model advises to buy the stock at that moment. On the other hand, when it reaches one of the h "entry" connector hubs from the lower ranges' community, then it is also a sign of *trend reversal*, this time with the stock price entering in a *down* trend, so the model advises to sell the stock at that moment. Taking Fig. 3c as an example, in this case, when $h = 3$, the *downhubs* in the network would be the connector hubs 11, 13 and 14, while the *uphubs* would be the connector hubs 15, 16 and 17.

Note that, by proceeding this way, we are here also assuming that the price oscillation patterns identified for the stock during the trend detection phase will still remain during the operating phase, i.e., after the *tdays* days previously stipulated for splitting the dataset into $X_{training}$ and X_{test}. It is also important to emphasize that the model only buys the stock when it is "flat", i.e., it is not currently already positioned in the stock (hence there are no "averaging down" strategies here) and, conversely, it only sells the stock when it is currently positioned in it (hence "short selling" strategies are not allowed here either).

Table 1 Time series used as database, obtained from NYSE and Bovespa Stock Exchange

Ticker	Company	d_0	d_f	Days
NYSE				
BAC	Bank of America	1999-11-22	2019-11-20	7303
EFX	Equifax	1999-11-22	2019-11-20	7303
F	Ford Motor	1999-11-22	2019-11-20	7303
GE	General Electric	1999-11-22	2019-11-20	7303
JPM	JPMorgan	1999-11-22	2019-11-20	7303
M	Macy's	1999-11-22	2019-11-21	7304
SCHW	Charles Schwab	1999-11-22	2019-11-21	7304
PFE	Pfizer	1999-11-22	2019-11-20	7303
T	AT&T	1999-11-22	2019-11-20	7303
X	United States Steel	1999-11-22	2019-11-20	7303
Bovespa				
ABEV3	Ambev	1999-09-17	2019-11-19	7368
BBAS3	Banco do Brasil	1998-03-16	2019-11-19	7918
BRFS3	Brasil Foods	2009-12-10	2019-11-19	3631
BRKM5	Braskem	2002-09-02	2019-11-19	6287
CIEL3	Cielo	2009-12-18	2019-11-19	3623
CSNA3	CSN	1998-03-16	2019-11-19	7918
ITSA4	Itaúsa	1998-03-16	2019-11-19	7918
JBSS3	JBS	2007-03-29	2019-11-19	4618
PETR4	Petrobras	1998-03-16	2019-11-19	7918
VALE3	Vale	1998-03-16	2019-11-19	7918

Regarding the time complexity of the model during the operating phase, there are two tasks with linear time ($O(n)$), where n represents the size of subdataset X_{test}, and one task with logarithmic time ($O(n \log n)$), which is the *GetHubs* procedure from Algorithm 2, where n then represents the size of network G.

4 Materials and methods

In the stock market, the returns of trading operations depend on specific timing strategies for buying and selling the stock. The proposed model defines these timings strictly according to the trending patterns detected through a network-based technique, for each stock. Hence, in order to evaluate the model, we compare its performance, in terms of financial return, with the one provided by the stock price during the same period of the tests, i.e., with the returns obtained by the "buy and hold" strategy. The database we build for the tests comprises the daily closing price histories of 10 of the most traded stocks from NYSE and 10 of the most traded stocks from the Brazilian Stock Exchange (Bovespa). These stocks are listed in Table 1,

along with their respective time ranges considered. Eventual stock splits and consolidations are also taken into account and appropriately reflected into the price histories for generating the model's input data.

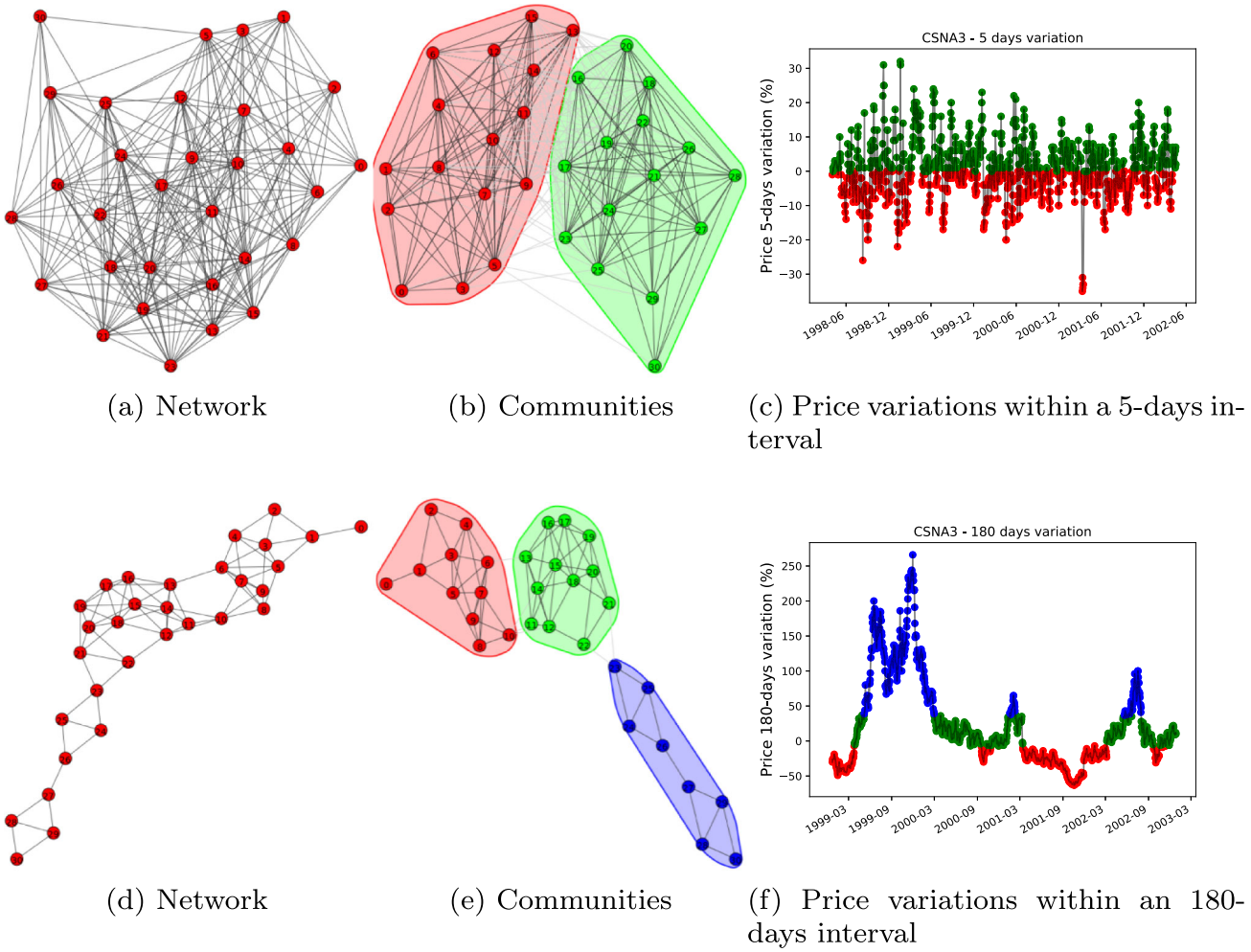
In order to obtain the results, we run the model using different values for the sliding window parameter v and for the number of hubs parameter h, when processing all stocks in the database, such that $v \in (5, 10, 30, 60, 120, 180)$, and $h \in [1, 10]$. The number of days *tdays* for splitting X into $X_{training}$ and X_{test} is fixed as 1000, which represents almost 4 years of trading days. The values of *var-v* are discretized through a simple rounding function, for the sake of converting them into integers, and the length l of each range r is defined as the square root of the length of $X_{training}$, which results in $n = l$, for all cases. For detecting communities in the network, we make use of the *fast greedy* algorithm (Newman 2004).

5 Experimental results

In this section, we present the obtained results when applying the proposed model on historical data from NYSE and Bovespa stocks.

5.1 Generated networks

We start by showing, in Fig. 4, the networks, detected communities and the evolution of the values in $X^{var-v}_{training}$ obtained from the trend detection phase for the stock CSNA3, for $v = 5$ and $v = 180$, respectively. We present the plots for the same stock with these specific values of v with the aim of demonstrating how the parameter v may affect the training process' output. For lower values of v, the network G tends to have a higher average degree (Fig. 4a) and the price variations tend to present a lower



(a) Network (b) Communities (c) Price variations within a 5-days interval

(d) Network (e) Communities (f) Price variations within an 180-days interval

Fig. 4 Output from the trend detection phase for stock CSNA3, when the sliding window $v = 5$ days (above) and $v = 180$ days (below). Left column: Network G. Middle column: Communities detected. Right column: Price variation ranges evolution. The colors in **c** and **f** denote the communities to which each variation range belongs. Note that, when comparing the price variations, in the third column, the ranges for $v = 180$ follow much longer trends and also reach higher values, both on the positive and on the negative sides. (Color figure online)

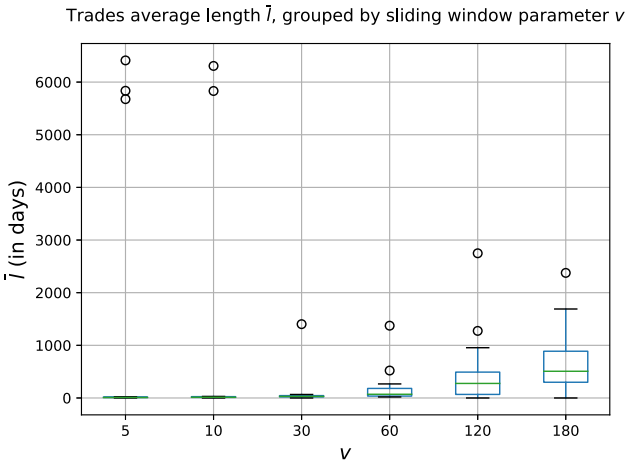


Fig. 5 Box plots of the average length of trades $\bar{l}$ (in days) according to sliding window parameter v (also in days) in the operating phase for both NYSE and Bovespa databases, considering all values of parameter h. Usually, as lower the value chosen for v, the shorter is the expected length of the trading operations performed by the model. The *outliers* shown in this figure occur for small values of h, such as 1 or 2

autocorrelation (Fig. 4c). In opposition, higher values of v tend to result in a network G with lower average degree (Fig. 4d) and to incur a higher autocorrelation for the price variations (Fig. 4f). A lower v also tends to increase the number of trades performed by the model during the operating phase, as the nodes get more interconnected and so there are more chances of the price variation ranges to reach the *uphubs* and the *downhubs* in the network. Another consequence of having lower values for the sliding window v is that it makes the average length of the trades shorter (Fig. 5), for the same reason explained above. Therefore, we can say that if one is planning to make use of the model with the goal of optimizing short-term trading operations, then he is advised to choose lower values of v. While if one plans to optimize long-term trading operations, then a higher value of v is indicated. The parameter h may also affect the number of operations performed by the model, since lower values of h, such as 1 or 2, tend to lessen its probability of triggering new operations, both for *buying* and for *selling*.

5.2 Obtained returns

The returns achieved by the model in the operating phase, considering all values of parameters v and h, both for NYSE and Bovespa databases, are displayed in Fig. 6 and summarized in Table 2, grouped by stock. The first column in Table 2 , r_{stock}, shows the stock return in terms of its price variation within the period of the operating phase, i.e., the time range provided in X_{test}. The column $\overline{r}_{model}$ shows the average return achieved by the model, considering all values of v and h. The columns d_0^* and d_f^* display the initial and final dates taken into consideration by the model for achieving the best return r^*_{model} for each stock. The value of d_0^* may some times differ from the operating phase's initial date due to the sliding window parameter v^*.

For the NYSE database, the model's average return $\overline{r}_{model}$ is able to surpass the stock price return r_{stock} in the period for 6 out of the 10 cases considered, with the exceptions being EFX, JPM, SCHW and T. As for the model's best return r^*_{model}, it is able to outperform the stock price return in 8 out of the 10 cases considered, with the 2 exceptions being EFX and SCHW. The best case, in terms of return optimization, is achieved for stock F, in which the stock price return for the period is -33.46% while the model's best return is 673.3% in the same period. This return rate for F is achieved through 131 trading operations, with an average length of 26 days and a geometric mean return of 1.57% per operation. For the stock JPM, the highest return is obtained for stock F, of 673%. For Bovespa, the returns obtained for stock CSNA3 really stand out from the others, reaching more than 8000%

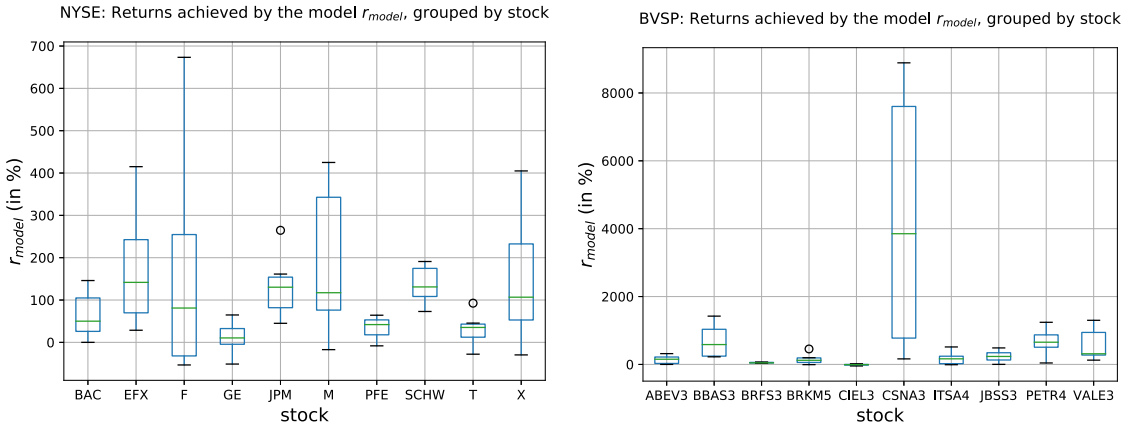


Fig. 6 Box plots of the returns achieved by the model r_{model} (in %) in the operating phase on the NYSE and Bovespa databases, grouped by stock, considering all values of parameters v and h. For NYSE, the

Table 2 Main results obtained from the tests: the returns r are expressed in percentages (%)

Stock	Model's average results		Model's best results							
								Trades info		
	r_{stock}	$\overline{r}_{model}$	r^*_{model}	d^*_0	d^*_f	v^*	h^*	n^*	$\overline{l}^*$	$\overline{r}^*$
NYSE										
BAC	− 56.03	− 5.46	146.04	2003-11-21	2019-11-20	5	6	193	21	0.47
EFX	478.34	152.10	415.04	2003-12-30	2019-11-20	30	2	4	1402	50.65
F	− 33.46	3.55	673.30	2003-12-01	2019-11-20	10	3	131	26	1.57
GE	− 60.55	− 15.60	64.72	2004-05-10	2019-11-20	120	2	14	275	3.63
JPM	263.52	54.19	264.65	2003-12-01	2019-11-20	10	1	1	5831	264.65
M	− 35.74	135.43	424.95	2004-08-05	2019-11-21	180	2	3	1191	73.80
SCHW	308.77	67.70	183.73	2003-12-30	2019-11-21	30	1	61	66	1.72
PFE	17.06	18.13	64.12	2004-08-05	2019-11-20	180	2	12	253	4.21
T	47.07	4.44	92.59	2004-08-05	2019-11-20	180	10	4	1222	17.80
X	− 64.00	57.32	404.96	2003-12-30	2019-11-20	30	3	155	19	1.05
Bovespa										
ABEV3	360.16	71.14	318.10	2004-04-20	2019-11-19	5	2	1	5676	318.10
BBAS3	906.37	394.20	1423.18	2002-04-29	2019-11-19	10	1	1	6307	1423.18
BRFS3	− 24.48	19.85	71.10	2014-01-13	2019-11-19	10	1	44	25	1.23
BRKM5	87.82	84.67	455.75	2006-09-13	2019-11-19	5	2	185	15	0.93
CIEL3	− 77.06	− 32.67	19.47	2014-01-14	2019-11-19	5	8	78	4	0.23
CSNA3	451.51	2541.69	8887.88	2002-05-24	2019-11-19	30	10	120	28	3.82
ITSA4	602.62	133.33	514.82	2002-12-20	2019-11-19	180	2	4	837	57.47
JBSS3	390.74	179.46	354.17	2011-07-14	2019-11-19	60	1	5	521	35.35
PETR4	313.28	382.56	1241.27	2002-04-15	2019-11-19	5	10	397	9	0.66
VALE3	660.68	399.17	1300.08	2002-06-03	2019-11-19	10	2	151	28	1.76

The mark * indicates the best result for each stock. The columns n^*, $\overline{l}^*$ and $\overline{r}^*$ show the total number of trades, average length per trade and the geometric mean return per trade, respectively

model's best return is only 1% higher than the stock price's return for the period, and this return is achieved through 1 operation alone. Looking at the trades list of that stock, for these parameters, we noted that the model in fact just performed the equivalent to a "buy and hold" strategy in this case, buying the stock on 2003-12-02 and having being forced to sell it just at the end of the simulation's period in order to compute its return.

The most surprising item in the results for Bovespa database is the high rates of return obtained for the stock CSNA3. Since these values really stand out from the others, we have double checked these numbers in order to be sure that the model's calculations and the input data are both correct. The average return from the model $\overline{r}_{model}$ for CSNA3 (2,541%)—which is the arithmetic mean of the returns obtained for all values of parameters v and h—is already much higher than the stock return r_{stock} for the same period (451%). The best return obtained by the model r^*_{model} (8,887%) though, when $v = 30$ and $h = 10$, is even higher. This high return is achieved from a total of 120 trades, with an average length of 28 days and a geometric mean return of 3.82% per trade. Overall, the average return from the model is able to outperform the stock return in 4 out of 10 cases: BRFS3, CIEL3, CSNA3 and PETR4. As for the model's best return, it is able to outperform the stock price return in 7 out of 10 cases, with the 3 exceptions being ABEV3, ITSA4 and JBSS3. For ABEV3 and BBAS3, the model achieved its best return with only 1 trade, being that in the case of BBAS3 it was nevertheless able to outperform the stock price return in the same period by a high margin (1,423% versus 906%). Looking at this individual trade in the stock's trades list for these parameters ($v = 10$ and $h = 1$), we observe that the model waited for the stock initial price to decrease before opting for buying it, on 2002-08-12.

To help us understand how the model is able to achieve such higher returns for stocks F, from NYSE, and CSNA3, from Bovespa, we generate Fig. 7, in which it is possible to

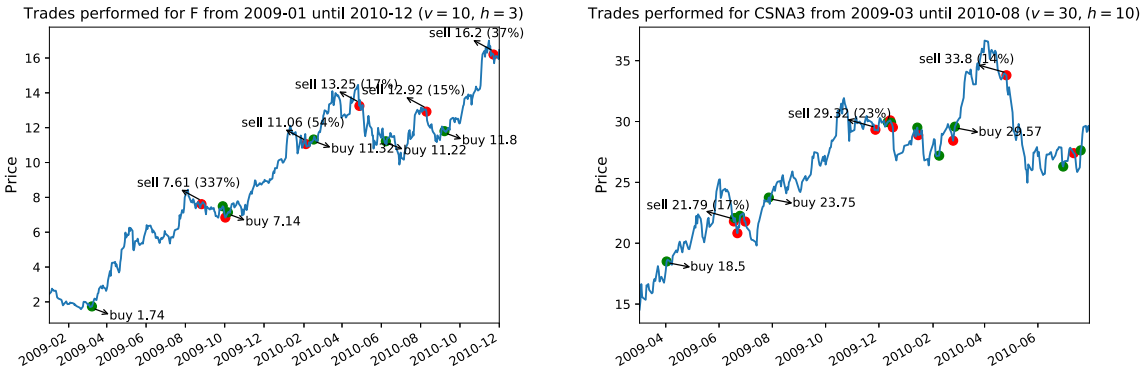


Fig. 7 Examples of trades performed by the model for stocks F and CSNA3 during the operating phase, using their respective optimal parameters v and h. *Buying* operations are indicated by the green color and *selling* operations by the red color. The values in parentheses show the return achieved in each trade. In these two cases, the model is more able to correctly detect a future *up* or *down* trend for the price, hence the higher overall returns obtained for these stocks when compared to others. (Color figure online)

Table 3 Annualized Sharpe Ratio (SR) obtained by a "buy and hold" strategy and the one from the proposed model's best returns, for the NYSE and Bovespa stocks in the database

	Buy and hold	Model	
	SR	SR	t-stats
NYSE	0.17	2.14	8.46
Bovespa	0.56	1.99	7.33

see some examples of the operations performed by the model when using the respective optimal values of the parameters h and v for these stocks. The first trading operation shown in this figure, for stock F, is the most profitable of all, achieving a return of 337% (by buying it at 1.74 on 2009-03-09 and selling it at 7.61 on 2009-08-26). As it can be noted from the remaining operations shown for this stock in the figure, the model is very successful at detecting the price trending patterns during this period, both when buying and selling the stock. It is worth remembering that, evidently, during the operating phase only the historical and current prices are acknowledged by the model, and not the future prices. Hence, we can also say that the model is accurately predicting the future prices for the stock during this period, thus the high return rates achieved in these trading operations. The same observations can also be made for the stock CSNA3 during the period shown in Fig. 7.

In Table 3, we make a comparison between the annualized Sharpe ratio of a "buy and hold" strategy and the one from the proposed model's best returns, for the NYSE and Bovespa stocks in the database. The Sharpe ratio (Sharpe 1966) is the average return earned in excess of the risk-free rate over the strategy volatility or standard deviation, and it is widely used in finance. Generally, the greater the value of the Sharpe ratio, the more attractive the risk-adjusted return. In this case, we assume that the risk-free rate is equal to zero. The statistical significance of the mean obtained returns is given by t-statistic = Sharpe Ratio $\times\sqrt{\text{number of years}}$. As it is shown in Table 3, the strategy from the proposed model's best returns outperforms the buy-and-hold one for both NYSE and Bovespa stocks in the considered periods.

6 Final remarks

In this work, we introduced a network-based model to detect price trends and also to automatize decision-making processes in stock market trading operations in order to optimize the returns. The model starts by mapping the stock's historical price variation ranges into a network and then generates the trend labels based on its topological structure. These labels, which represent the price's *up* and *down* trending patterns identified in this phase, are later propagated to the future prices, and the network's connector hubs are used as indicators of price trend reversals, triggering operations for buying or selling the stock accordingly. The results obtained through the application of the model to real financial time series, both from NYSE and the Brazilian Stock Exchange, are promising, and future studies should be made with the aim of applying it to larger databases as well as of possibly improving its efficiency, in terms of return rates optimization.

One remaining open question is to discover the main factors that contribute to a better performance of the model for a specific stock. The answer for it could be, for instance, in the topological properties of the network resulted from the trend detection phase, with some specific network structural patterns being more suitable to the model than others. In case there is such a correlation, then a

preliminary stock filtering process, based on their resulting network topological properties, would help on determining the ones which are more likely to have their returns optimized by the model.

Another possible improvement in the model, which would be interesting to test and check the results, is in turning its trend detection phase more dynamic. At the way it is now, it makes use of the first *tdays* days of the stock historical prices to generate the variation ranges network *G*, which remains unchanged during the whole operating phase. Additionally, updating the network regularly could lead to higher overall returns at the end of the processing. This could be done as newer price variations are known during the operating phase. Preferably, the network should be updated daily, but for testing purposes one can make use of a monthly update, i.e., around every 21 trading days. The parameter *tdays*, in this case, could also be used as a sliding window, for saving processing time in the tests as well as for keeping the trend detection process concise and always up to date.

Acknowledgements This work is supported in part by the Sao Paulo State Research Foundation (FAPESP) under Grant Nos. 2015/50122-0 and 2013/07375-0, the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001, the Pro-Rectory of Research (PRP) of University of Sao Paulo under grant number 2018.1.1702.59.8, the C4AI under grant number FAPESP/IBM/USP: 19/07665-4 and the Brazilian National Council for Scientific and Technological Development (CNPq) under Grant No. 303199/2019-9.

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest.

References

Adebiyi AA, Adewumi AO, Ayo CK (2014) Comparison of ARIMA and artificial neural networks models for stock price prediction. J Appl Math. https://doi.org/10.1155/2014/614342

Akiki TJ, Abdallah CG (2019) Determining the hierarchical architecture of the human brain using subject-level clustering of functional networks. Sci Rep 9(1):1–15

Albert R, Albert I, Nakarado GL (2004) Structural vulnerability of the north American power grid. Phys Rev 69(2):025103

Anghinoni L, Zhao L, Ji D, Pan H (2019) Time series trend detection and forecasting using complex network topology analysis. Neural Netw 117:295–306

Bachelier L (1900) Théorie de la spéculation. Gauthier-Villars, Paris

Barabási AL (2003) Linked: the new science of networks. Perseus Books Group

Barabási AL, Albert R (1999) Emergence of scaling in random networks. Science 286(5439):509–512

Barabási AL et al (2016) Network Science. Cambridge University Press, Cambridge

Boccaletti S, Latora V, Moreno Y, Chavez M, Hwang DU (2006) Complex networks: structure and dynamics. Phys Rep 424(4):175–308

Cao H, Lin T, Li Y, Zhang H (2019) Stock price pattern prediction based on complex network and machine learning. Complexity 2019:1–12. https://doi.org/10.1155/2019/4132485

Carneiro MG, Zhao L (2017) Organizational data classification based on the importance concept of complex networks. IEEE Trans Neural Netw Learn Syst 29(8):3361–3373

Cartwright D, Harary F (1956) Structural balance: a generalization of Heider's theory. Psychol Rev 63(5):277

Casiraghi G (2017) Multiplex network regression: how do relations drive interactions? ArXiv preprint arXiv:1702.02048

Cauchy AL (1813) Sur les polygones et polyedres, second mémoire. J Ecole Polytechnique 9:87–98

Colliri T, Zhao L (2019) A network-based model for optimizing returns in the stock market. In: 2019 8th Brazilian conference on intelligent systems (BRACIS), pp 645–650. https://doi.org/10.1109/BRACIS.2019.00118

Colliri T, Ji D, Pan H, Zhao L (2018) A network-based high level data classification technique. In: 2018 International joint conference on neural networks (IJCNN). IEEE, pp 1–8

Dorogovtsev SN, Mendes JF (2013) Evolution of networks: from biological nets to the Internet and WWW. OUP, Oxford

Erdös P, Rényi A (1960) On the evolution of random graphs. Publ Math Inst Hung Acad Sci 5(1):17–60

Faloutsos M, Faloutsos P, Faloutsos C (1999) On power–law relationships of the internet topology. ACM SIGCOMM Comput Commun Rev 29(4):251–262

Ferreira LN, Zhao L (2014) Detecting time series periodicity using complex networks. In: 2014 Brazilian conference on intelligent systems. IEEE, pp 402–407

Gao X, An H, Fang W, Huang X, Li H, Zhong W, Ding Y (2014) Transmission of linear regression patterns between time series: from relationship in time series to complex networks. Phys Rev E 90(1):012818

Gleiser PM, Spoormaker VI (2010) Modelling hierarchical structure in functional brain networks. Philos Trans R Soc A Math Phys Eng Sci 368(1933):5633–5644

Guimera R, Amaral LAN (2005) Functional cartography of complex metabolic networks. Nature 433(7028):895

Guresen E, Kayakutlu G, Daim TU (2011) Using artificial neural network models in stock market index prediction. Expert Syst Appl 38(8):10389–10397. https://doi.org/10.1016/j.eswa.2011.02.068

Hagmann P, Cammoun L, Gigandet X, Meuli R, Honey CJ, Wedeen VJ, Sporns O (2008) Mapping the structural core of human cerebral cortex. PLoS Biol 6(7):e159

Hayek FA (1945) The use of knowledge in society. Am Econ Rev 35:519–530

Hinton GE (1989) Connectionist learning procedures. Artif Intell 40(1–3):185–234

Huang W, Nakamori Y, Wang SY (2005) Forecasting stock market movement direction with support vector machine. Comput Oper Res 32(10):2513–2522

Huillier S (1861) Mémoire sur la polyèdrométrie. Annales de Mathématiques 3:169–189

Ji LJ, Zhang Z, Guo T (2008) To buy or to sell: cultural differences in stock market decisions based on price trends. J Behav Decis Mak 21(4):399–413

Lee K, Jo G (1999) Expert system for predicting stock market timing using a candlestick chart. Expert Syst Appl 16(4):357–364. https://doi.org/10.1016/S0957-4174(99)00011-1

Liu W, Suzumura T, Ji H, Hu G (2018) Finding overlapping communities in multilayer networks. PLoS ONE 13(4):e0188747

Loglisci C, Malerba D (2017) Leveraging temporal autocorrelation of historical data for improving accuracy in network regression. Stat Anal Data Min ASA Data Sci J 10(1):40–53

Malkiel BG, Fama EF (1970) Efficient capital markets: a review of theory and empirical work. J Finance 25(2):383–417
Mitchell M (2006) Complex systems: network thinking. Artif Intell 170(18):1194–1212
Montoya JM, Solé RV (2002) Small world patterns in food webs. J Theor Biol 214(3):405–412
Motter AE, Lai YC (2002) Cascade-based attacks on complex networks. Phys Rev E 66(6):065102
Newman ME (2004) Fast algorithm for detecting community structure in networks. Phys Rev E 69(6):066133
Palla G, Derényi I, Farkas I, Vicsek T (2005) Uncovering the overlapping community structure of complex networks in nature and society. Nature 435(7043):814–818
Pastor-Satorras R, Vespignani A (2001) Epidemic spreading in scale-free networks. Phys Rev Lett 86(14):3200
Pastor-Satorras R, Vespignani A (2002) Immunization of complex networks. Phys Rev E 65(3):036104
Regnault J (1863) Calcul des chances et philosophie de la bourse. Mallet-Bachelier, Paris
Rish I (2001) An empirical study of the naive Bayes classifier. In: IJCAI 2001 workshop on empirical methods in artificial intelligence 3(22). IBM, New York
Roberts HV (1959) Stock-market "patterns" and financial analysis: methodological suggestions. J Finance 14(1):1–10
Safavin SR, Landgrebe D (1991) A survey of decision tree classifier methodology. IEEE Trans Syst Man Cybern 21(3):660–674
Sharpe WF (1966) Mutual fund performance. J Bus 39(1):119–138
Silva TC, Zhao L (2012) Network-based high level data classification. IEEE Trans Neural Netw Learn Syst 23(6):954–970
Silva TC, Zhao L (2015) High-level pattern-based classification via tourist walks in networks. Inf Sci 294:109–126
Silva TC, Zhao L, Cupertino TH (2013) Handwritten data clustering using agents competition in networks. J Math Imaging Vis 45(3):264–276
Sporns O (2002) Network analysis, complexity, and brain function. Complexity 8(1):56–60
Vandermonde AT (1771) Remarques sur les problèmes de situation. Mémoires de l'Académie Royale des Sciences (Paris) 2:566–574
Vapnik VN (2000) The nature of statistical learning theory. Springer, New York
Wang JL, Chan SH (2007) Stock market trading rule discovery using pattern recognition and technical analysis. Expert Syst Appl 33(2):304–315
Wang XF, Chen G (2003) Complex networks: small-world, scale-free and beyond. IEEE Circuits Syst Mag 3(1):6–20
Watts DJ (2004) Six degrees: the science of a connected age. WW Norton & Company, New York
Watts DJ, Strogatz SH (1998) Collective dynamics of "small-world" networks. Nature 393(6684):440
West GB, Brown JH, Enquist BJ (2009) A general model for the structure, and allometry of plant vascular systems. Nature 400:125–126