A stock trading algorithm based on trend forecasting and time series classification

Matheus Rosisca Padovani [1], **João Roberto Bertini Junior** [1]

[1]Faculdade de Tecnologia – Universidade Estadual de Campinas
Limeira – SP – Brasil

m222360@dac.unicamp.br, bertini@ft.unicamp.br

Abstract. *Algorithm trading relies on the automatic identification of buying and selling points of a given asset to maximize profit. In this paper, we propose the Trend Classification Trading Algorithm (TCTA) which is based on time series classification and trend forecasting to perform trade. TCTA first employs the K-means to cluster 5-days closing price segments and label them according to its trend. A deep learning classification model is then trained with these label sequences to estimate the next trend. Trading points are given by the alternation on trend estimates. Results considering 20 shares from Ibovespa show TCTA present higher profit than buy-and-hold and trading schemes based on Moving Average Converge Divergence (MACD) or Bollinger bands.*

1. Introduction

Stock trading is the process of negotiating shares to make a profit in the short term. A trader can make a profit by operating with long positions and short positions. In the former, the trader buys assets expecting their value will increase to sell. In the latter, the trader sells an asset without owning it and purchases them as the price drops. Trades can be done within a day or in a period of a few days or weeks, namely day trade and swing trade, respectively [Brogaard et al. 2015]. A trading strategy supports trader decisions by signaling entries, exits, and size positioning. Such strategies rely on financial indicators and forecasting models to estimate market movements [Gandhmal and Kumar 2019].

Forecasting models can be used to estimate both price and trend. Trend forecasting aims to estimate the price direction within a given period. It is an alternative to stock price prediction stated as a classification task. Such models can be built with a time series of historical data. Time series are categorized as stationary or non-stationary regarding their underlying distribution [Box et al. 2016]. If the time series has a constant mean and standard deviation, it is characterized as a stationary time series. In this case, statistical models like the auto-regressive (AR), the moving average (MA), or the mixed auto-regressive moving average (ARMA) can be successfully applied to perform forecasting. Non-stationary time series, on the other hand, do not present constancy to data over time. In such cases, transformations can be applied to match the stationary conditions, thus the transformed time series can be modeled with Auto-regressive integrated moving average (ARIMA). Still, auto-regressive models implicitly assume that the future will resemble the past, and may not be the best model for non-stationary data [Rachev et al. 2007].

Given the restriction of statistical models non-stationary time series, machine learning algorithms, and most recently, deep learning methods, have been proposed to address forecasts [Fawaz et al. 2018]. A branch of these algorithms transforms the task

of trend forecasting on a time series classification. The classifying technique chosen depends on the type of problem [Bagnall et al. 2016].

Time series Classification has been a challenging problem of data mining, one of the challenges is deal with noise on series, which can lead to not consistent results. Even noise removing using filters does not resolve, instead introduces a problem of lag in the data [Yang and Wu 2005].

In this paper, we propose a Trend Classification Trading Algorithm (TCTA) that employs machine learning forecasting methods to signal entrance and exit points. Specifically, we use K-means to group 5-days closing prices into three clusters, which we interpret as up, down, and constant trends, respectively. Then, two neural networks are trained with sequences of those cluster labels, to predict the next. Trade signals are then emitted when the time series trend label alternate.

The remainder of the paper is organized as follows. Section 2 describes in a brief some related works. Section 3 details the proposed methodology employed in the experiments and presents the proposed trading algorithm. Section 4 presents the conducted experiments and discusses the obtained results. At last, Section 5 concludes the paper the suggests some future works.

2. Related works

This section presents some recent reviews and related works to the concepts used in the proposed TCTA algorithm. Regarding input data, TCTA expects the closing price processed by the Rate of Change (ROC) indicator. Stock closing price and volume are the most frequent entry for forecasting algorithms, although unstructured data such as news and texts from social media has also been employed [Bustos and Pomares-Quimbaya 2020].

TCTA uses the K-means clustering algorithm to help identify and label the trends as upward, downward, and constant, before train a forecasting model. Clustering algorithms have been used in market analysis. Most frequently, fuzzy clustering approaches are used together with neural networks as forecasting method [Rajab and Sharma 2019]. Authors in [Shao et al. 2017] have used K-means to cluster the stock price sequences. Then a Long short-term memory (LSTM) neural network was trained for each obtained cluster. Their model first predicts the next day's closing price, then a refined prediction is made by the LSTM corresponding to the shortest distance clustering from the input sequence. The proposed model showed higher prediction accuracy when compared to the traditional feed-forward neural network and single LSTM neural network prediction. A two-stage fusion model based on clustering has been proposed in [Xu et al. 2020]. The authors used K-means to cluster several technical indicators, then ensemble learning has also been applied to improve the prediction accuracy. Their experiment with daily stock data of Bombay Stock Exchange, CNX Nifty, and S&P 500 stock indices showed the proposed model obtains a better balance between accuracy and interpretability than the other four prediction methods.

Once the price sequences have been grouped and labeled, we have a time series of trend labels. Thus, trend forecasting is addressed as a time series classification task. Fawaz et al. [Fawaz et al. 2018] and Bagnall et al. [Bagnall et al. 2016] present recent

surveys on time series classification. The former focuses on deep learning methods for time series classification while the latter presents an extensive comparison with time series classification algorithms proposed up to five years before 2017.

The proposed algorithm uses deep learning methods to learn the label sequence patterns and estimate the next trend. Deep learning is a subset of neural networks that uses multiple layers and provides a high-level abstraction for data modeling. In [Ozbayoglu et al. 2020], the authors present a review of deep learning models in financial applications, such as algorithm trading, risk assessment, portfolio management, and related tasks. From their study, they concluded that most of the trading algorithms concentrate on stock price prediction and that LSTM is the most preferred deep learning model. Thakkar and Chaudhari [Thakkar and Chaudhari 2021] presented a review of deep learning models for the stock price and trend prediction from 2017 to 2020. They provide an experimental evaluation with nine deep learning-based models. According to their comparisons, the deep Q-network model obtained the best result in five days of trend forecasting.

3. Methodology

This section presents the methods employed to design the TCTA algorithm. Section 3.1 presents how the stock prices are pre-processed and transformed in a sequence of trend labels. Then, Section 3.2 presents the methods used to learn from the trend label time series to predict the next trend. Section 3.3, then details the proposed TCTA algorithm to perform trade through trend classification.

3.1. Data pre-processing

We collected the closing prices of each stock by a determinate period and made some transformations. We separated closing prices using a sliding window method dividing them into five days of lag variables. These parts have a four-day overlap since the window moves just one day to the future per iteration. Each row of the data set ends up with the date of the first-day value, and five days of values.

To limit the variation and to help to elucidate the trends, a transformation was made on data replacing closing price values by the Rate of Change (ROC). ROC is a momentum indicator that compares current and past prices and shows the percentage change between them. If this value is positive, then the value has increased by that percentage and, if it is negative, the value had decreased. ROC is calculated by comparing the interval between the prices. For example, ROC 200, compares today's price with the price of 200 days ago [Achelis 2001]. In this work, we use ROC values instead of closing prices to better capture changes in prices. The base of comparison is always the first day of the series, so if we use a five sized interval, we will have the base, noted as ROC 0 that always will be zero so we took it off, followed by ROC 1, ROC 2, ROC 3 and ROC 4, as shown in Table 1.

Table 1. Example of ROC transformation of price values

PRICES	VAL1	VAL2	VAL3	VAL4	VAL5	Transform	ROC	ROC1	ROC2	ROC3	ROC4
02/01/2019	87.90	88.00	88.50	88.48	88.83	$\rightarrow$	02/01/2019	0.11	0.68	0.66	1.06
03/01/2019	88.00	88.50	88.48	88.83	90.32	$\rightarrow$	03/01/2019	0.57	0.55	0.94	2.64
04/01/2019	88.50	88.48	88.83	90.32	90.42	$\rightarrow$	04/01/2019	-0.02	0.37	2.06	2.17

At the end of this process, we have a data set where each row represents one day and the percentage change from 4 subsequent days. At this point, the data is 5-dimensional and represents the percentage variation of 5 days period of one stock. To classify future values, the data must be divided into classes. Thus, the next step is labeling each data instance with the trend it represents, i.e., upward (U), downward (D) and, constant (C). To automate labeling and make it an unbiased process we employed the K-means algorithm and have abstracted the group is found to the corresponding class.

The batches of percentage change were fed into K-means and the labels were created. The values necessary to be classified as upward and downward vary along with all stocks, depending on their history.

The K-means clustering algorithm is unsupervised, meaning that it does not require data with labels. Its objective is grouping elements that are like each other in K clusters, the K value is predefined. At each iteration, a given distance metric between an element and the centroid of each cluster is measured. Each element is assigned to cluster with the nearest centroid. At the end of each iteration, all clusters' centroids are recalculated as the mean position between all elements of the cluster.

Table 2 shows the inputs as percentage changes or ROC and the output of K-means as the classes. The visualization of these classes is depicted in Fig. 1 and 2. Values were taken from a run of the 'BOVA11', an Exchange-traded fund (ETF) that follows the index Ibovespa.

Table 2. Table Roc into Classes

ROC 1	ROC 2	ROC 3	ROC 4	OUTPUT
0.24	2.63	4.17	4.06	Upward
-1.95	0.79	0.13	0.04	Constant
-0.57	-0.09	-1.88	-3.17	Downward

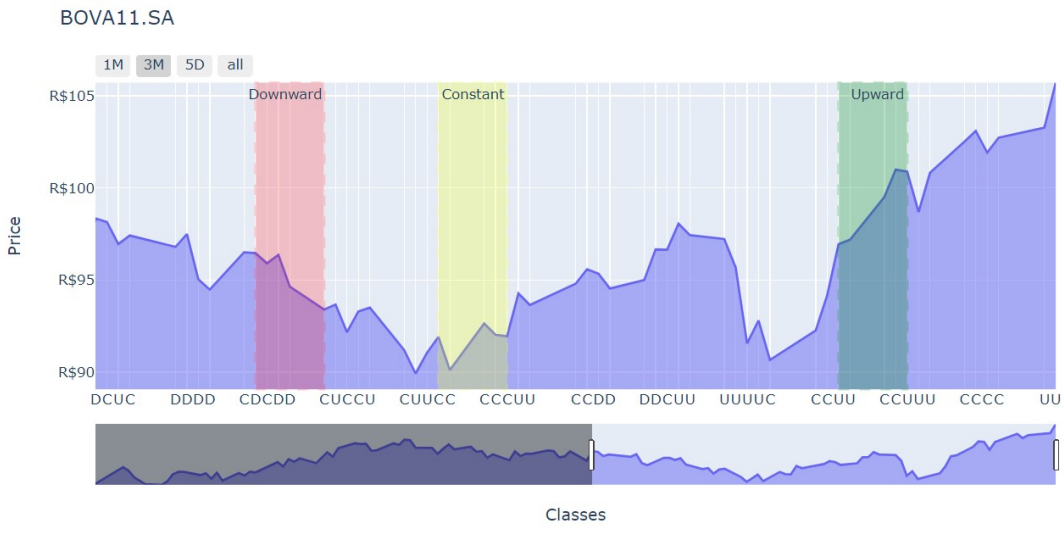

Figure 1. Closing price chart classified into Upward (U), Downward (D), or Constant (C) trends

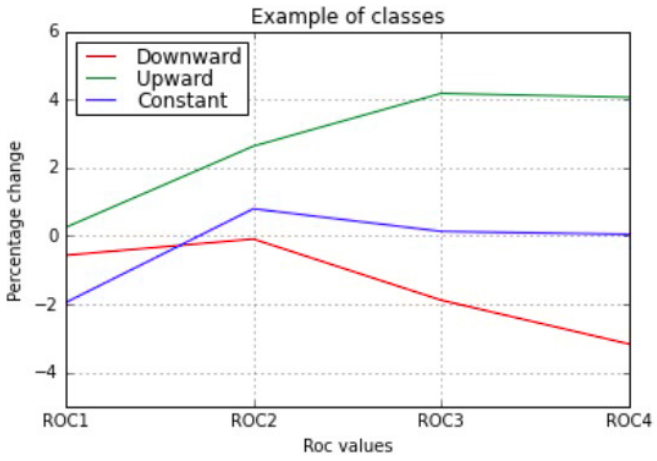

Figure 2. Examples of the three classes found in the ROC transformed price data

In Figure 1 we can see the division with the prices that each day represents. The figure shows closing price data from 2020-09-01 to 2020-11-24 of 'BOVA11' share. Axis x shows the classes that each day represents, Upward (U), Downward (D), or Constant (C), Axis y shows closing price at day. There is an example of each class in the colored boxes. The first day on the red box is a D, meaning that from this day to the next 4 days the class is Downward according to K-means. The same is applied to the other boxes. Figure 2 plots the same values presented in Table 2 and Fig. 1 with the same starting point.

3.2. Trending forecasting methods

As described in the previous section, we transform the stock closing price data into a sequence of trending labels, denoted as U (upward), D (downward), and C (constant). Now, given a sequence of five such labels, we want to estimate the next. i.e. $C, C, C, U, U \rightarrow U$. For doing so, we considered two deep learning models, the Deep Neural Network model (DNN) and Long Short-Term Memory model (LSTM), as described next.

Before input the label sequence to the deep learning models, however, the categorical variables need to be transformed into one-hot variables. That transformation is done to remove the possibility of a model creating ordinal relations between categorical data. Our model has categorical values representing classes of each batch of days, so we transform all classes using one-hot encoding. Creating a data set capable of being fed into the neural networks.

3.2.1. Deep Neural Network

Deep Neural Networks (DNN) are artificial neural networks composed of several layers. Among the many DNN possible architectures, Feed Forward DNN are, in essence, multilayer Perceptrons with a larger number of layers [Aggarwal 2018].

The DNN architecture used in this paper was created with the try and error method. The idea was to create larger layers at the beginning and shrink them at the end, we prioritize layers with multiples of the input. The network topology consists of

seven layers. The first is an input layer with 15 neurons, which receives a sequence of five trend labels (either up, down, or constant) transformed into one-hot encoding. The first layer is followed by two dense layers of 300 neurons, two dense layers of 150 neurons, two dense layers of 50 neurons, a layer of 15 neurons, and the final layer with three neurons. All neurons of the dense layers have Relu as an activating function, and the output layer has a softmax activation. The weights were initialized by the glorot-normal method, Adam optimizer was used with a learning rate of 0.1 and the categorical cross-entropy was set as the loss function. We used a method of training that is called early stopping, it stops the train after 7 generations without improvement on the loss function. We made a train/test division of $0.85/0.15$, and trained with mini-batches of size 30.

3.2.2. Long Short-Term Memory

Long Short-Term Memory (LSTM) is a Recurrent Neural Network (RNN) that introduces a special neuron called long short-term memory. A recurrent neural network has connections between hidden units associated with a delay, which allows to retain information of past and discover correlations of distant data. The information remains in the internal memory of the neuron because the output is transmitted to the next layer and the neuron itself. This model of network (RNN) is hard to train because of the vanishing gradient problem when the gradient used to update weights become so small that changes in weights don't reflect on the next layers of the network. This problem occurs often with temporarily distant events and may stop a network further training [Pascanu et al. 2013].

The LSTM was proposed to address the vanishing gradient problem. The LSTM units are linear and contain a set of cells inside it, whose activation is controlled by three gates; 1) input gate, whose role is deciding which data of short term memory should be activated, 2) forget gate, employed to decide which parts of the cell state remains relevant, and 3) output gate, which tells what part of the cell state is relevant for the current data, producing an output [Graves et al. 2009].

For creating the LSTM architecture we used the method try and error, the logic was the same as described in the DNN section 3.2.1. The network has seven layers; the input layer has 15 neurons, which receives a sequence of 5 trend labels transformed into one-hot encoding. The hidden layers are composed of two LSTM layers of 60 neurons, two dense layers of 30 neurons, and a layer of 15 neurons. All neurons of the dense layers have Relu as an activating function. The output layer has three neurons with softmax activation. For training the LSTM, we employed Adam optimizer with a learning rate of 0.08, the categorical cross-entropy as loss function, and glorot-normal as weight initialization method. We used a method of training that is called early stopping, it stops the train after 7 generations without improvement on the loss function, we made a train/test division of $0.85/0.15$ and trained with mini-batches of size 30.

3.3. The Trend Classification Trading Algorithm

This section presents a proposed algorithm named Trend Classification Trading Algorithm (TCTA). After creating both networks and train with the pre-processed data of 3.1 transformed into one hot encoding classes, as explained in Section 3.2. We use each network to generate a list of predictions for all days of the testing data set. We join that predictions list with a prices list and feed into TCTA.

TCTA works running through all days analyzing predictions and values, using them to trade. The output of TCTA is the final Balance after all trades, and a data set containing all trades made. The current Status of TCTA is important to the run because it tells if the model is bought, sold, or out of the trades. All trades occur with the entire balance, so the model can buy just once until it sells. The logic used is described in the Algorithm 1.

Algorithm 1 Trend Classification Trading algorithm

Require: $Price$ {Price}; $Prediction$ {Prediction of the Neural Network}; $Status$ {Status indicating if the model has a short position, a long position or no position} {Status 0 indicates no position, status 1 indicates long position and -1 indicates short position}
if $Predicion = Uptrend$ and $Status = 0$ or $Status = -1$ **then**
$Buy(Price)$ {Buy at current Price}
$Status \leftarrow Status + 1$ {update status indicating the buy}
else if $Predicion = Downtrend$ and $Status = 0$ or $Status = 1$ **then**
$Sell(Price)$ {Sell at current Price}
$Status \leftarrow Status - 1$ {update status indicating the sell}
end if

The strategy used by the neural consists of buying when the prediction is an upward trend and selling when the prevision is a downward trend, if it is a stable prevision, keeps its position like before. This algorithm repeats every day until the end of the training data. On the last day, the model makes a last transaction leaving the trades. The final balance is saved in a data set and compared to others afterward. The TCTA strategy works with both DNN and LSTM predictions, one at a time. We train and test the networks with TCTA 3 times for each stock and selected the best run to use as a result.

4. Experiments

The stocks used in the trading experiment were selected from Ibovespa, the main index tracking stocks performance in Brazil's stock exchange (B3). We collected closing values of 19 stocks and one ETF (Exchange-traded fund) 'BOVA11', that mirrors Ibovespa index performance. All companies were chosen based on their relevance in the index at the start and end dates of the time series, for diversification proposes there are no more than two stocks per Market sector. We collected 10 years of daily closing values from begin of 2010 to the end of 2020.

Table 3. List of shares and label distribution

Stock	Upward	Constant	Downward	Stock	Upward	Constant	Downward
BOVA11	703	1397	620	ITUB4	570	1377	773
ABEV3	537	1573	610	JBSS3	602	1402	716
BBDC4	505	1525	690	LAME4	706	1398	616
BRFS3	507	1630	583	LREN3	600	1415	705
CCRO3	400	1572	748	KLBN4	506	1754	460
CIEL3	581	1625	514	MRVE3	571	1431	718
CMIG4	580	1677	463	PETR4	498	1678	544
CYRE3	571	1542	607	TIMS3	358	1582	780
ELET6	371	1510	839	VIVT3	668	1430	622
GOLL4	194	1566	960	VALE3	464	1559	697

Table 3 shows all stocks used for testing and distribution of classes in each training data set. The label Constant is by far the most common.

Tests of trade started on 02/01/2020, the first market day of that year. The Moving Average Convergence Divergence (MACD) Model needed 10 days of market to build the exponential mean, so all the trades started on day Jan 16, 2020, and go on until June 21, 2021. In total, they traded for 350 market days.

All trading algorithms follow the same pattern, starting with R$\$10,000$ for trade, on the same dates. They have functions to buy or sell a stock and each Model has its logic. At each transaction of a model, the balance, quantity of shares owned, number of total trades, and actual status of transactions are updated, but this part was taken off the example's code to make evident just the logic. All trades have a fee of 0.03%, this value is the fee, for all transactions made by investors in general at B3 (http://www.b3.com.br). All models trade with the entire amount of money at each transaction. All project code was made in Python, neural networks were made using Keras API, share values were collected using Yahoo Finance API, only the daily closing value was used.

4.1. Comparative Algorithms

We choose three well-stated trading methods to compare the proposed algorithm. A brief explanation of them and their trading rule is described ahead. They are Buy and Hold method, MACD trading pattern, and Bollinger Bands trading pattern. Both trading patterns used 5 days moving average, so all models have access to the same data.

4.1.1. Buy and Hold

Buy and hold is a common name for a passive investment strategy, it implies that in the long run that stock will raise its prices and you should hold it and wait for the gains. This strategy reduces costs of fees and taxes, which are paid when trading and take a small portion of the investments. For testing this strategy, the model buys on the first day of tests and sells the stock on the last day, making only these two transactions.

4.1.2. Moving Average Convergence Divergence

Moving Average Convergence Divergence (MACD) is a trend indicator that uses the relation between two exponential moving averages (EMA) of different sizes. An exponential moving average differs from the simple moving average because the exponential gives more relevance to ending days, unlike the simple that gives all days the same weight. The values chosen for EMA are typically two values indicating the same strategy, like two short-term indicators or two long-term indicators.

When the model shows an average divergence also known as a gap between two EMA values, it is an entry point for trading, because as time passes the means should converge again. The trading strategy consists in buying when the nearest moving average $(EMA5)$ is lower than the farthest one $(EMA10)$ expecting the rise in prices returning to mean. When the nearest average is bigger than the farthest is time to sell, expecting falling prices returning to mean.

You can use MACD in multiple ways, long-only, short-only, or long and short strategies. In this project, we decided to use long and short. The EMA values chosen were $EMA5$ and $EMA10$. In this strategy either you have a long or short position all the time [Huang and Kim 2006].

- Buy Rule
  - If EMA5< EMA10
- Sell Rule
  - If EMA5>EMA10

4.1.3. Bollinger Bands

Bollinger Bands are a type of price envelope that uses moving average(MA) and standard deviation(std), this is the band limit formula $(MA + -n * std)$. It plots the middle line being the simple moving average, an upper limit n deviations up, creating an upper band, and a lower band with n deviations down. It's a parameter to known when the market is devalued or overvalued, using standard deviation in the formula makes volatility part of the equation, so if the market is too volatile, the bands will be widen indicating a new range of prices for the market.

Trading using Bollinger Bands consists in selling when the values are high, above the upper limit, and buying when the prices are low, below the lower limit. Since a value tends to return to the mean, typically in some time the values converge to the mean and the strategy made money with this volatility [Williams 2006].

In this project we are using 5 days as Moving Average and 1.5 standard deviations, representing a short-term strategy. We trade using long and short to use Bollinger's full potential.

- Buy Rule
  - If price $<$ Bollinger Lower Band
- Sell Rule
  - If price $>$ Bollinger Upper Band

4.2. Results and Discussion

The results of trading models are shown as Percentage over initial capital on Figure 3 and Table 4.

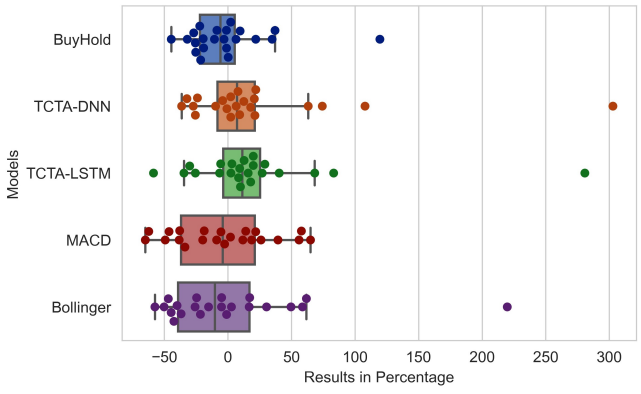

Figure 3. Boxplot distribution of trading results (percentage)

Figure 3 shows the distribution of results, each ball represents a stock return and each row represents a model. TCTA models were more consistent than others and had more positive values overall.

Table 4. Percentage over initial capital for trading models under comparison

Stock	BuyHold	TCTA-DNN	TCTA-LSTM	MACD	Bollinger
BOVA11	9.6	**107.87**	83.2	55.97	58.66
ABEV3	-1.06	**6.53**	-34.56	2.04	-44.62
BBDC4	-8.64	2.22	8.4	21.9	**49.64**
BRFS3	-19.42	17.93	17.93	26.05	**219.77**
CCRO3	-26.87	21.94	**29.08**	-62.24	-36.81
CIEL3	-44.53	-32.24	-30.05	-49.28	**-4.92**
CMIG4	6.39	-9.69	-6.52	**64.98**	16.76
CYRE3	-21.43	**21.27**	2.38	14.03	-46.86
ELET6	37.07	7.93	**40.35**	11.75	-21.61
GOLL4	-31.94	-0.78	26.99	-19.81	**30.41**
ITUB4	-1.15	**63.19**	3.16	-2.58	-5.02
JBSS3	**0.24**	-36.37	-58.51	-37.92	-39.99
LAME4	-25.58	-25.58	-25.58	-38.3	**-24.61**
LREN3	-22.13	-27.23	**19.97**	-33.8	-25.82
KLBN4	**21.92**	-23.97	15.94	-18.61	-15.21
MRVE3	-19.07	2.26	**12.74**	-8.84	-50.17
PETR4	2.19	**302.87**	280.79	18.73	2.8
TIMS3	-25.3	**-4.12**	-5.68	-46.3	-57.41
VIVT3	-10.57	**12.53**	9.88	-64.94	-42.46
VALE3	**119.57**	9.27	9.27	57.83	17.14
Mean	-3.03	**20.79**	19.95	-5.46	-1.01
Std	**34.79**	74.25	68.3	39.39	61.75
Results removing outliers (two best and two worst runs of each model)					
Mean	-4.88	4.60	**8.01**	-6.56	-11.94
Std	**17.41**	21.43	18.04	29.29	28.11

Table 4 shows the best result of each share in bold, we can see the victories lead by, the TCTA-DNN with 7 wins, Bollinger Bands with 5, TCTA-LSTM with 4, Buy and Hold Strategy with 3, and MACD with just one.

Talking more deeply about the TCTA, they had the biggest deviation of results. Otherwise, they had more constant profits. Both TCTA-DNN and TCTA-LSTM had near results, they diverge in few cases, surprisingly their mean was almost the same, overall TCTA-DNN, showed more constant wins and had a higher deviation. Even if we remove the two biggest results of both, it is still better than the other models, even though the difference would be smaller.

The TCTA works predicting the next day trend and trading, if it succeeds in buying a stock before a raise or selling before a drop, it increases profits over time. However, this guess is not always right and besides that, sudden variations in prices caused by changes

in a company or the economy cannot be predicted, these variations lead to movements out of trends provoking losses for TCTA. MACD and Bollinger Bands react to sudden price changes in less time, a cross in the means of MACD, and prices hitting bands in Bollinger bands are faster than a change on the 5-day trend, used by TCTA, sometimes leading to minor losses to them, yet these short-term fast trades can become noise on the long run.

The common statistical trading methods in general create a somewhat bigger return than Buy and Hold, but trading fees decrease their return over time, besides that they come with much higher risk and volatility.

MACD was the least volatile model, and had the least wins, this method can start a short position when prices increasing suddenly, expecting a return to the means and if prices stay a long time far from the mean, it loses money.

Bollinger Bands method showed variated results, it was slightly better than Buy and Hold in the mean, but worse in volatility. The logic of the method can lead to early trade exits not benefiting from long-term appreciations.

If we remove both the two best and two worst stocks of each model we will end up with just the two TCTA algorithms with positive mean results, TCTA-LSTM being the best and Bollinger Bands being the worst. The standard deviation of TCTAs would be near the deviation of buy and hold but with a mean 8% bigger for TCTA-DNN and 12% bigger for TCTA-LSTM. The difference between LSTM and DNN shows that a specific neural network can surpass a general network with proper data and training. Removing these outliers brings the models into an average field, that may be expected for other stocks or periods.

5. Conclusions

In this paper, we proposed the Trend Classification Trading Algorithm which uses time series classification and trend forecasting to identify entry and exit points. TCTA uses K-means to group price data and to label the trends. Then we tested two deep learning neural networks models, DNN and LTSM, to estimate the next trend. We have conducted trading experiments with 20 shares from Ibovespa comparing the two variants of TCTA, TCTA-DNN, and TCTA-LTSM, with buy-and-hold and two trading schemes based on the MACD and Bollinger Bands indicators, respectively. The trading schemes were evaluated according to the percentage of earns/loses over an initial capital and within the period Jan. 16, 2020, to Jun. 21, 2021. The TCTA variants presented the best results among the trading strategies. Confirming that neural network models learned a pattern on the prices. TCTA-DNN presented a slightly better average of 20.8 while TCTA-LTSM of 19.9 percent. However, the TCTA-DNN presented a higher standard deviation than its counterpart, 74.5 and 68.3, respectively. The other trading strategies had negative performance in the period, being MACD the worst. Removing the outliers would make TCTA-LTSM the best one. Future works include considering different labeling strategies, more deep learning models, and comparing TCTA against other deep learning-based trading models.

References

Achelis, S. B. (2001). *Technical analysis from A to Z.* McGraw Hill, 2nd edition.

Aggarwal, C. C. (2018). *Neural Networks and Deep Learning*. Springer, 1st edition.

Bagnall, A., Lines, J., Bostrom, A., Large, J., and Keogh, E. (2016). The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances. *Data Mining and Knowledge Discovery*.

Box, G. E. P., Jenkins, G. M., Reinsel, G. C., and Ljung, G. M. (2016). *Time Series Analysis Forecasting and Control*, volume 5. John Wiley and Sons, Inc.

Brogaard, J., Hagströmer, B., and Lars Nordén, R. R. (2015). Trading fast and slow: Colocation and liquidity. *The Review of Financial Studies*, 28:3407–3443.

Bustos, O. and Pomares-Quimbaya, A. (2020). Stock market movement forecast: A systematic review. *Expert Systems with Applications*, 156:113464.

Fawaz, H. I., Forestier, G., Weber, J., Idoumghar, L., and Muller, P.-A. (2018). Deep learning for time series classification: a review. *Data Mining and Knowledge Discovery*, pages "917–963".

Gandhmal, D. P. and Kumar, K. (2019). Systematic analysis and review of stock market prediction techniques. *Computer Science Review*, 34:100190.

Graves, A., Liwicki, M., Fernández, S., Bertolami, R., Bunke, H., and Schmidhuber, J. (2009). A novel connectionist system for unconstrained handwriting recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(5):855–868.

Huang, B. and Kim, Y. S. (2006). A test of MACD trading strategy. Master's thesis, Faculty of Business Administration - Simon Fraser University.

Ozbayoglu, A. M., Gudelek, M. U., and Sezer, O. B. (2020). Deep learning for financial applications : A survey. *Applied Soft Computing*, 93:106384.

Pascanu, R., Mikolov, T., and Bengio, Y. (2013). On the difficulty of training recurrent neural networks. In *Proceedings of the 30th International Conference on Machine Learning*, volume 28, pages 1310–1318. PMLR.

Rachev, S. T., Mittinik, S., Fabozzi, F. J., Focardi, S. M., and Jašić, T. (2007). *Financial Econometrics: From Basics to Advanced Modeling Techniques*. John Wiley & Sons, Inc, 1st edition.

Rajab, S. and Sharma, V. (2019). An interpretable neuro-fuzzy approach to stock price forecasting. *Soft Computing*, 23:921–936.

Shao, X., Ma, D., Liu, Y., and Yin, Q. (2017). Short-term forecast of stock price of multi-branch lstm based on k-means. In *The 2017 4th International Conference on Systems and Informatics (ICSAI 2017)*, pages 1546–1551.

Thakkar, A. and Chaudhari, K. (2021). A comprehensive survey on deep neural networks for stock market: The need, challenges, and future direction. *Expert Systems with Applications*, 177:114800.

Williams, O. (2006). Empirical optimization of bollinger bands for profitability. Master's thesis, Faculty of Business Administration - Simon Fraser University.

Xu, Y., Yang, C., Peng, S., and Nojima, Y. (2020). A hybrid two-stage financial stock forecasting algorithm based on clustering and ensemble learning. *Applied Intelligence*.

Yang, Q. and Wu, X. (2005). 10 challenging problems in data mining research. *International Journal of Information Technology and Decision Making*, 5:"597–604".