# Modeling and Predicting the Lima Stock Exchange General Index with Bayesian Networks and Information from Foreign Markets

Daniel Chapi, Soledad Espezua, Julio Villavicencio, Oscar Miranda, and Edwin Villanueva(✉)

Pontifical Catholic University of Peru, Lima, Peru
ervillanueva@pucp.edu.pe

**Abstract.** This paper presents a Bayesian Network approach to model and forecast the daily return direction of the Lima stock Exchange general index using foreign market's information. Thirteen worldwide stock market indices were used along with the copper future that is negotiated in New York.

The proposed approach was compared against popular machine learning methods, including decision tree, SVM, Multilayer Perceptron and Long short-term memory networks. The results showed competitive results at classifying both positive and negative classes. The approach allows graphical representation of the relationships between the markets, which facilitate the understanding on the target market in the global context. A web application was developed to demonstrate the advantages of the proposed approach. To the best of our knowledge, this is the first effort to model the influences of the main stock markets around the world on the Lima Stock Exchange general index.

**Keywords:** Stock market index prediction · Bayesian networks · S&P/BVL

## 1 Introduction

Predicting the closing direction of stock market indices is an important task, since investors could benefit from it to devise strategies for trading the stocks comprising the index, thereby increasing their potential for future profit. However, predicting the stock index direction is a challenging problem due to the complex and stochastic nature of the markets. Several machine learning methods have been proposed to address this task, including: Support Vector Machines [13,14,17,22,29], Tree-based classifiers [2], Fuzzy Inference systems [4–6,15,25], Artificial Neural Networks (ANN) [1,11,12,26], Neuro-Fuzzy systems [3,24],

Supported by Pontifical Catholic University of Peru.

Bayesian networks [18,26], Hidden Markov models [30] and more recently, Deep Learning models [7,8,10,23,28].

Despite this considerable amount of research, the focus has been primarily on improving predictive performance by devising new model designs or by searching for new informative variables to incorporate into the models. Regarding the latter, a large proportion of published articles proposes to use predictor variables derived from the same index or from sources of information generated in the same target market, like economic or sentiment indicators. Few works have described models with predictor variables from other markets. One of which came from Sung and So [20], which analyzed various interrelated world stock market indices to extract association rules for predicting daily changes of the Korea Composite Stock Price Index (KOSPI). Their methodology was able to find some unexpected association patterns among global stock market indices that were useful to forecast the target market and understand its behavior in the global context. In addition, following on from this, a recent article from Malagrino et al. [18] described a Bayesian Network (BN) approach to model the conditional dependencies between iBOVESPA index (the main index in the Sao Paulo Stock Exchange) and different stock market indices from around the globe. The accuracy results were comparable to the obtained by some popular machine learning methods using single market data, but its simplicity was remarkable, since it only used closing directions values from foreign stock market indices.

According to the findings of Malagrino's work, BNs can be a useful tool not just to forecast stock indices, but also to model the interrelationships among the markets. However, the simplifications made in that work by grouping the indices by their home continent limited the understanding whether modeling the indices as individual variables is a worthwhile path. This paper describes a Bayesian Network approach to model and forecast the daily closing direction of the Lima Stock Exchange General Index (S&P/BVL) based on information from 13 foreign market indices. Separately to the work of Malagrino et al. each individual index is modelled as a random variable, thus representing the full joint probability distribution among the different markets. An exhaustive investigation evaluates the appropriate time period for the forecasting task and subsequently compare the results against several popular machine learning models. To the best of our knowledge, this is the first effort trying to model and forecast the S&P/BVL index with information from global markets.

## 2 Materials and Methods

We collected stock index daily closing prices from Yahoo! Finance's website from 13 representative markets: Dow Jones (USA), S&P 500 (USA), IBEX 35 (Spain), CAC 40 (France), FTSE 100 (UK), DAX (Germany), BSE Sensex (India), Hang Seng (China), Nikkei 225 (Japan), S&P/ASX 200 (Australia), IPC Mexico (Mexico), iBOVESPA (Brasil). We also collected historical Copper Futures prices (that is negotiated in New York Stock Exchange) due to the importance of this commodity in the target market. The period of the collected data was from

03/30/2000 to 03/30/2020, corresponding to a 7306 d. Additionaly, the historical opening and closing prices from the target index S&P/BVL Peru General (Peru) were also collected.

Figure 1 shows the kernel density distribution of each collected index. No index is stationary or follows normal distribution.

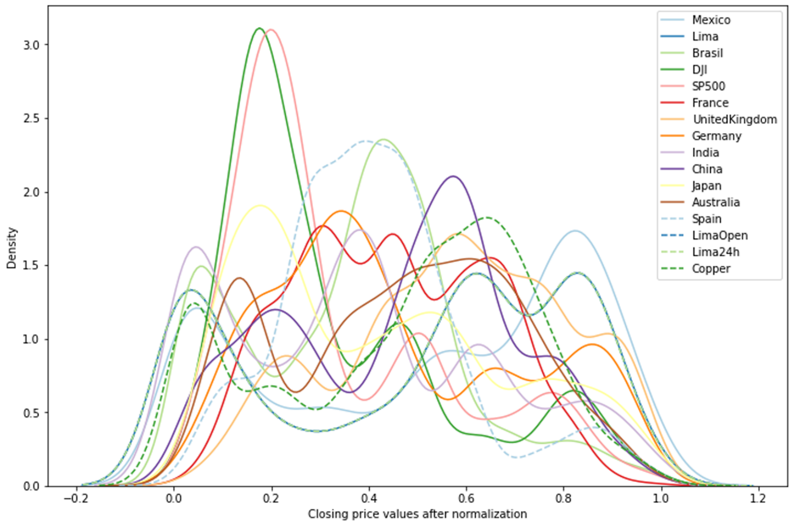

**Fig. 1.** Kernel density estimate of the different index values

In order to facilitate the modeling with BNs we transformed each index series into a discrete one: values greater than or equal to the previous day are assigned the discrete value "1", otherwise is assigned "0". The resulting discrete dataset was composed by 5205 days (excluding weekends and days where all the markets were closed).

The indices used in this study are from different time zones, which means that the closing time of each stock market may not be the same (Table 1). As our intention is to capture potential relationships from the foreign stock markets with respect to the Lima Stock market in order to predict it, we have constraints in the search of such relationships, which is dictated by the order in which the markets close. Thus, the variables representing the indices are defined and ordered by their closing time relative to the closing time of Lima Stock Exchange. Figure 2 illustrates this, where the longitudinal distance to Lima indicates the difference of a market closing time with respect to that of Lima. Variables in the same longitudinal position represent markets that have the same closing time. For the markets that close at the same time or after Lima, the corresponding variables represent values from the previous day (eg. SP500, DJI, Mexico and Copper). No left-to-right relationships are allowed in the BN structure learning phase in order to respect the flow of information produced by the market closings.

**Table 1.** Closing times of the modeled stock markets

| Index | Stock market | Closing time (GMT+00) |
|---|---|---|
| Copper Future | New York Stock Exchange | 22:00 h |
| IPC Mexico | Mexico Stock Exchange | 22:00 h |
| S&P500 | New York Stock Exchange | 21:00 h |
| Dow Jones | New York Stock Exchange | 21:00 h |
| iBOVESPA | Brazil Stock Exchange | 21:00 h |
| S&P/BVL Peru General | Lima Stock Exchange | 21:00 h |
| CAC 40 | Euronext Paris | 16:30 h |
| DAX | Frankfurt Stock Exchange | 16:30 h |
| IBEX35 | Madrid Stock Exchange | 16:30 h |
| FTSE 100 | London Stock Exchange | 16:30 h |
| BSE Sensex | Bombay Stock Exchange | 10:00 h |
| Hang Seng | Hong Kong Stock Exchange | 08:00 h |
| Nikkei 225 | Tokyo Stock Exchange | 06:00 h |
| S&P/ASX 200 | Australian Securities Exchange | 06:00 h |

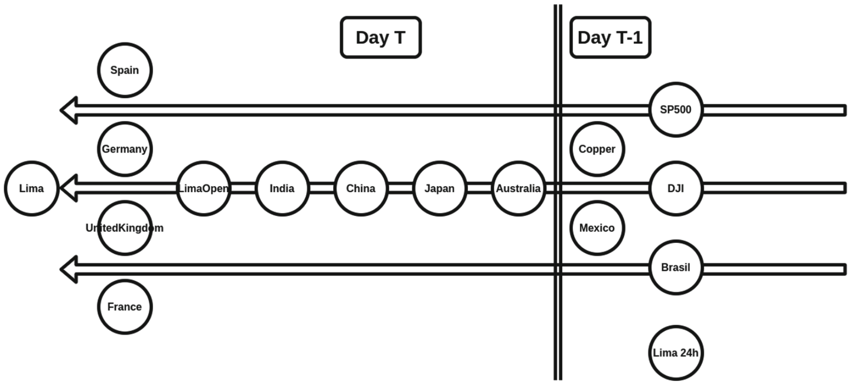

**Fig. 2.** Temporal order of the indices stock markets closing times

We call the set of variables defined according to Fig. 2 as 24 h set because they all represent indices closed up to 24 h before the Lima stock market. In addition, we generated a 48 h set that include all the variables of 24 h set plus those that represent the closing information of the indices on the previous day (this results in a set of 30 variables and the Lima target variable). In a similar way we generated a 72 h set comprising of 45 variables and the target variable. In Sect. 3 we discuss the results obtained with each set of variables.

For each set of variables we arranged its corresponding dataset. Missing values due to holidays were treated following two different methods. The first corresponds to the removal of all data rows corresponding to the days where some market is on holiday, this resulted in a reduction of the data to a total of 3567 d where all the markets were open. The second method involves imputing the missing value by copying the value from the previous working day. The results with these two methods will later be discussed in Sect. 3.

To learn the structure (acyclic directed graph - DAG) of the BN models we experimented with two algorithms: Hill Climbing and Max-min Hill Climbing [27].

As mentioned before, the learning was constrained to follow the temporal order of the closing times. For this we constructed lists of forbidden edges (blacklists) so the learning algorithms do not consider adding such edges in the structure discovery process. The size of the blacklist was 266 for the 24 h dataset, 1044 for the 48 h dataset and 2332 for the 72 h dataset. After learning the structure of the models, their parameters (conditional probability tables of the variables) are computed by using Maximum Likelihood Estimations from the corresponding dataset.

## 3 Experiments and Results

Here we describe experiments and results obtained with our BN approach and alternative methods.

As alternative methods we tested the following algorithms: decision trees (DT), support vector machine (SVM) with radial and polynomial kernels, Multilayer Perceptron (MLP) and Long short-term memory (LSTM) neural architectures. These methods were chosen for their popularity and good results reported in the field [9,19].

Prior to performing the experiments we first split the data using a 80:20 ratio (the first 80% for training and the remaining 20% for testing). The 20% set was reserved for comparisons between the final optimized models for each model type. First, we describe the experiments with our approach and then proceed to describe the experiments with alternative models.

**Bayesian Networks (BN).** With the 80% training data we follow a walk-forward validation strategy in order to evaluate the effectiveness of each combination of imputation method, set of variables (time window) and BN structure learning algorithm (12 combinations in total) and identify the optimal one. This consisted of splitting the data in n temporal blocks and iterating from the second one, using all the previous blocks as a training set to induce the BN model and testing it on the iterating block to obtain performance metrics. After doing all n-1 iterations we obtain averaged scores. As performance metrics we register: Accuracy, Precision, Recall, F1-score and G-mean.

Table 2 shows the average G-mean scores obtained in the walk-forward validation of each combination. G-mean is worth observing here since it penalize models that present poor results in any of the classes (we consider that false positives and false negatives are equally important). All configurations presented close scores, ranging from 0.6160 to 0.6344. The best result was obtained with the Hill Climbing learning algorithm, with the dropping-rows-with-nulls as treatment of missing values and using the time window of 24 h. In all time windows the Hill Climbing algorithm presented better results than Max-min Hill Climbing. In relation to the missing values treatment, the dropping-rows-with-nulls strategy tends to present slightly better results than the imputation by duplicating the previous value. This might indicate that adding artificial data adds noise to the model or that the imputation method applied may not be the most appropriate for this problem. The small differences in scores between the configurations (lower than 0.02) suggests that there is not a superior better treatment, learning algorithm or time window for our approach, which can also suggest that the approach is robust to these parameters, according to the G-mean metric.

**Table 2.** G-mean scores of the bayesian network models

| Time window | Missing values treatment | Structure learning algorithm | G-mean |
|---|---|---|---|
| 24 h | Duplicating last value | Max-min Hill Climbing | 0.6191 |
| 24 h | Duplicating last value | Hill Climbing | 0.6220 |
| 24 h | Dropping rows with null values | Max-min Hill Climbing | 0.6246 |
| 24 h | Dropping rows with null values | Hill Climbing | **0.6344** |
| 48 h | Duplicating last value | Max-min Hill Climbing | 0.6160 |
| 48 h | Duplicating last value | Hill Climbing | 0.6224 |
| 48 h | Dropping rows with null values | Max-min Hill Climbing | 0.6279 |
| 48 h | Dropping rows with null values | Hill Climbing | 0.6309 |
| 72 h | Duplicating last value | Max-min Hill Climbing | 0.6194 |
| 72 h | Duplicating last value | Hill Climbing | 0.6268 |
| 72 h | Dropping rows with null values | Max-min Hill Climbing | 0.6258 |
| 72 h | Dropping rows with null values | Hill Climbing | 0.6324 |

**Decision Trees (DT).** Decision Tree models were induced with the same 80% training datasets used in the BN modeling. In addition, we obtained continuous versions of the datasets to induce DT models with corresponding continuous variables. Also, based on the results obtained in the BN experiments, we hereinafter adopt the dropping-rows-with-nulls strategy to treat the missing values, since it tended to present superior results. The DT learner has a set of hyperparameters that can affect the quality of the resulting models: the max depth of the tree, the minimum samples split and the minimum samples per leaf. We optimized these parameters by using a Bayesian optimization method in order to allow a wide and smart search of the hyperparameter space. The search ranges were: [1, training_data_size] for max depth; [1, 2000] for the minimum samples to split; and [1, 100] for the minimum samples per leaf. The evaluation of each hyperparameter combination followed the walk-forward strategy in the training data with the G-mean as scoring metric (as in BN models). Table 3 shows the G-mean score for the best hyperparameter configuration found in each combination of data type and time window. The results obtained with discrete data are noticeably more accurate than those obtained with continuous data, with almost no difference throughout the different time windows.

**Table 3.** G-mean results for the best hyperparameters configurations of DT model found by Bayesian optimization in each combination of data type and time window

| Data type | Time window | Max depth | Min samples split | Min samples leaf | G-mean |
|---|---|---|---|---|---|
| Discrete data | 24 h | 1 | 229 | 1 | 0.6263 |
| Discrete data | 48 h | 1 | 462 | 1 | 0.6263 |
| Discrete data | 72 h | 1 | 2 | 1 | **0.6265** |
| Continuous data | 24 h | 2853 | 2 | 1 | 0.3397 |
| Continuous data | 48 h | 2852 | 2 | 1 | 0.4084 |
| Continuous data | 72 h | 2852 | 2 | 1 | 0.3645 |

**Support Vector Machines (SVM).** For this kind of model we experimented two common kernels: the radial basis function kernel (RBF) (as in [13]) and the polynomial kernel, as in [19]. Similar to DT models, main SVM hyperparameters were optimized with Bayesian optimization: for RBF kernel models the gamma parameter was optimized in the range [0.01, 100]; for polynomial-kernel models the degree hyperparameter was optimized in the range [1, 5]. In all models the regularization hyperparameter (C) was optimized in the interval [0.01, 100]. Table 4 shows results for the best hyperparameter configuration of RBF-kernel models in each combination of data type and time window. Likewise, Table 5 shows results for SVM model with polynomial kernels. Similar to the DT results, the superiority of the scores obtained with discrete data is noticeable, but minor differences exist along the different time windows in that data type.

**Table 4.** Results for the best hyperparameter configurations of SVM models with RBF kernels found by Bayesian optimization in each combination of data type and time window.

| Data type | Time window | Regularization parameter (C) | Gamma ($\gamma$) | G-mean |
|---|---|---|---|---|
| Discrete data | 24 h | 13.86 | 0.01 | 0.6217 |
| Discrete data | 48 h | 43.37 | 0.01 | 0.6303 |
| Discrete data | 72 h | 2.01 | 0.01 | **0.6411** |
| Continuous data | 24 h | 77.45 | 31.68 | 0.4081 |
| Continuous data | 48 h | 9.52 | 64.70 | 0.3860 |
| Continuous data | 72 h | 41.35 | 82.31 | 0.3577 |

**Multilayer Perceptrons (MLP).** For this kind of model we experimented 3-layer topologies, as in [16, 19, 21]. The number of neurons in the input layer is fixed to the size of the set of variables. The number of neurons in the second layer

**Table 5.** Results for the best hyperparameter configurations of SVM models with polynomial kernels found by Bayesian optimization in each combination of data type and time window.

| Data type | Time window | Regularization parameter (C) | Degree (D) | G-mean |
|---|---|---|---|---|
| Discrete data | 24 h | 0.03 | 2 | 0.6432 |
| Discrete data | 48 h | 0.12 | 1 | 0.6300 |
| Discrete data | 72 h | 2.52 | 1 | **0.6434** |
| Continuous data | 24 h | 83.12 | 2 | 0.3132 |
| Continuous data | 48 h | 100.00 | 2 | 0.3622 |
| Continuous data | 72 h | 43.85 | 3 | 0.3273 |

(hidden layer) is treated as a hyperparameter to be optimized in the range [10, 400]. In the third layer (output layer) there is only one neuron, which delivers the output of the model (the support for positive classification). All neuron units used hyperbolic tangent activation functions (tanh). To adjust weights we used Adam method, which updated the net weights using adaptive momentum in the backpropagation step to avoid local minima. The decay rates for the moments of the exponential moving average of the gradient were set to 0.9 (and 0.999 for the squared gradient) and a penalty term $\alpha = 0.0001$. The maximum number of epochs was set to 1000. The learning rate was treated as another hyperparameter to optimize in the range [0.00001, 0.5]. Table 6 shows results for the best hyperparameter configuration of MLP models in each combination of data type and time window. As with DT and SVM models, the discrete data generated superior Gmean scores but with small differences between the different time windows in that data type.

**Table 6.** Results for the best hyperparameter configurations of MLP models found by Bayesian optimization in each combination of data type and time window

| Data type | Time window | Learning rate | Hidden layer size | G-mean |
|---|---|---|---|---|
| Discrete data | 24 h | 0.00291 | 258 | **0.6479** |
| Discrete data | 48 h | 0.00020 | 290 | 0.6449 |
| Discrete data | 72 h | 0.00012 | 400 | 0.6409 |
| Continuous data | 24 h | 0.06273 | 399 | 0.1459 |
| Continuous data | 48 h | 0.00001 | 10 | 0.2063 |
| Continuous data | 72 h | 0.00001 | 14 | 0.1422 |

**Long Short-Term Memories (LSTM).** For this kind of model we used a topology of one input layer, two hidden LSTM layers and a dense output layer with one neuron unit. The two LSTM layers have a dropout rate of 0.2 and

0.1 respectively in order to prevent overfitting in training. As in MLP models, Adam algorithm was used to update the weights in the backpropagation steps. Binary cross entropy was the loss function used for computing error gradients. The maximum number of training epochs was set to 5 due to the large amount of time that this type of model demands on training.

The number of units in the LSTM layers was treated as hyperparameters to be optimized, with the range of [10, 400] for the first layer and [10, 200] for the second layer. The learning rate was also considered as hyperparameter to be optimized, being between 0.00001 and 0.1. Table 7 shows results for the best hyperparameter configuration of LSTM models in each combination of data type and time window. Different from the other kind of models, LSTM models do not show large differences between the results with discrete and continuous data.

**Table 7.** Results for the best hyperparameter configurations of LSTM models found by Bayesian optimization in each combination of data type and time window

| Data type | Time window | Learning rate | Layer 1 size | Layer 2 size | G-mean |
|---|---|---|---|---|---|
| Discrete data | 24 h | 0.09435 | 10 | 10 | 0.6251 |
| Discrete data | 48 h | 0.04406 | 10 | 176 | 0.6366 |
| Discrete data | 72 h | 0.08404 | 10 | 10 | 0.6251 |
| Continuous data | 24 h | 0.03914 | 89 | 10 | 0.6301 |
| Continuous data | 48 h | 0.03527 | 58 | 21 | 0.6299 |
| Continuous data | 72 h | 0.04423 | 10 | 156 | **0.6368** |

## Model Comparison and Discussion

The previous experiments allowed us to identify the best set of hyperparameters for each combination of model type, data type, and time window. Here we present another set of experiments with hyperparameter-optimized models. The aim with these experiments is to compare the forecasting abilities of the different model-generating methods. The 20% test set is used to perform the model comparison. We performed a 1-day walk forward validation in the test set. This means that we iterate over the days in the test set, each time selecting a new day for testing and using all previous historical data for training the model (with optimized hyperparameters) and asking it to predict the closing price movement of the selected day. After repeating this on every test day we compute the confusion matrix and all associated metrics. For the case of BN model, we used Hill Climbing as a learning algorithm in these experiments. BN models were only evaluated on discrete data.

Table 8 shows the results of all metrics obtained by the different model types and time windows on discrete data. Table 9, shows equivalent results with continuous data.

**Table 8.** Models accuracy, precision, recall, F1-score and G-mean score with the discrete dataset

| Time window | Model | Accuracy | Precision | Recall | F1-score | G-mean |
|---|---|---|---|---|---|---|
| 24 h | Decision Tree | 0.6275 | 0.6632 | 0.6531 | 0.6581 | 0.6240 |
| 24 h | SVM - RBF kernel | 0.6275 | 0.6632 | 0.6531 | 0.6581 | 0.6240 |
| 24 h | SVM - Polynomial kernel | 0.6162 | 0.6705 | 0.5918 | 0.6287 | 0.6183 |
| 24 h | Multilayer Perceptron | 0.6303 | 0.6448 | 0.7270 | 0.6835 | 0.6104 |
| 24 h | Long short-term memory | 0.6176 | 0.6374 | 0.7041 | 0.6691 | 0.6007 |
| 24 h | Bayesian network | 0.6232 | 0.6511 | 0.6760 | 0.6633 | 0.6147 |
| 48 h | Decision Tree | 0.6275 | 0.6632 | 0.6531 | 0.6581 | 0.6240 |
| 48 h | SVM - RBF kernel | 0.6218 | 0.6412 | 0.7066 | 0.6723 | 0.6054 |
| 48 h | SVM - Polynomial kernel | 0.6190 | 0.6546 | 0.6480 | 0.6513 | 0.6151 |
| 48 h | Multilayer Perceptron | 0.6275 | 0.6452 | 0.7143 | 0.6780 | 0.6105 |
| 48 h | Long short-term memory | 0.5560 | 0.5532 | 0.9949 | 0.7110 | 0.1471 |
| 48 h | Bayesian network | 0.6261 | 0.6528 | 0.6811 | 0.6667 | 0.6171 |
| 72 h | Decision Tree | 0.6269 | 0.6623 | 0.6522 | 0.6572 | 0.6236 |
| 72 h | SVM - RBF kernel | 0.6227 | 0.6533 | 0.6650 | 0.6591 | 0.6164 |
| 72 h | SVM - Polynomial kernel | 0.6269 | 0.6623 | 0.6522 | 0.6572 | 0.6236 |
| 72 h | Multilayer Perceptron | 0.6339 | 0.6533 | 0.7084 | 0.6798 | 0.6205 |
| 72 h | Long short-term memory | 0.5372 | 0.5864 | 0.5294 | 0.5565 | 0.5379 |
| 72 h | Bayesian network | 0.6255 | 0.6520 | 0.6803 | 0.6658 | 0.6167 |

**Table 9.** Models accuracy, precision, recall, F1-score and G-mean score with the continuous dataset

| Time window | Model | Accuracy | Precision | Recall | F1-score | G-mean |
|---|---|---|---|---|---|---|
| 24 h | Decision Tree | 0.5056 | 0.5506 | 0.5408 | 0.5457 | 0.5003 |
| 24 h | SVM - RBF kernel | 0.5224 | 0.5482 | 0.7398 | 0.6298 | 0.4367 |
| 24 h | SVM - Polynomial kernel | 0.5392 | 0.5580 | 0.7730 | 0.6481 | 0.4437 |
| 24 h | Multilayer Perceptron | 0.5266 | 0.5482 | 0.7832 | 0.6450 | 0.4097 |
| 24 h | Long short-term memory | 0.5490 | 0.5490 | 1.0000 | 0.7089 | 0.0000 |
| 48 h | Decision Tree | 0.4776 | 0.5241 | 0.5281 | 0.5260 | 0.4688 |
| 48 h | SVM - RBF kernel | 0.5224 | 0.5458 | 0.7755 | 0.6407 | 0.4077 |
| 48 h | SVM - Polynomial kernel | 0.5378 | 0.5587 | 0.7526 | 0.6413 | 0.4561 |
| 48 h | Multilayer Perceptron | 0.5224 | 0.5636 | 0.5765 | 0.5700 | 0.5130 |
| 48 h | Long short-term memory | 0.5490 | 0.5490 | 1.0000 | 0.7089 | 0.0000 |
| 72 h | Decision Tree | 0.5133 | 0.5561 | 0.5575 | 0.5568 | 0.5062 |
| 72 h | SVM - RBF kernel | 0.5456 | 0.5666 | 0.7289 | 0.6376 | 0.4852 |
| 72 h | SVM - Polynomial kernel | 0.5288 | 0.5528 | 0.7366 | 0.6316 | 0.4512 |
| 72 h | Multilayer Perceptron | 0.5049 | 0.5403 | 0.6522 | 0.5910 | 0.4612 |
| 72 h | Long short-term memory | 0.5484 | 0.5484 | 1.0000 | 0.7083 | 0.0000 |

From the above results it is clear that the models trained with discrete data tend to achieve better scores in Accuracy, Precision and G-mean. Recall results of LSTM models learnt with continuous data were perfect (value 1), but when inspecting the predictions it was found that such models only predict positive labels (they are totally incapable of predicting the negative class). In the same models the G-mean values were 0, suggesting that this is a better metric to assess the present prediction task.

Models trained with discrete data show close G-mean scores (with the exception of the LSTM models that showed lower results). The scores ranged from 0.6054 (SVM - RBF kernel - 48 h time window) to 0.6240 (Decision Tree - 24 h and 48 h time windows, SVM - RBF kernel - 24 h time window). Figure 3 shows these results in graphical form, where there is no clear difference between models along different time windows. From this graph we can confirm the large advantage of the models induced with discrete data.

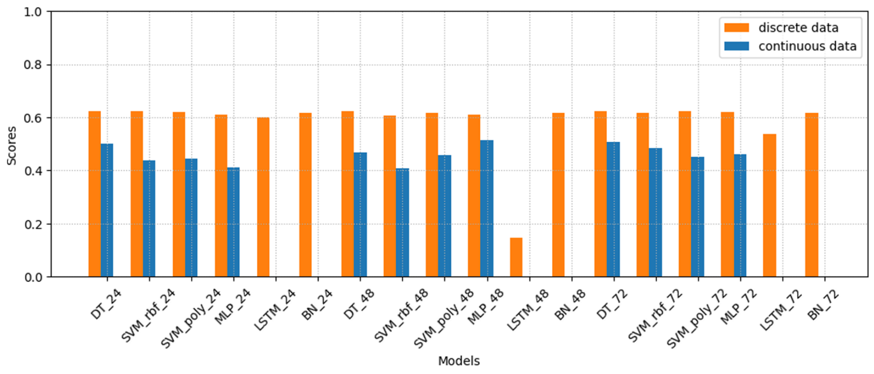

**Fig. 3.** G-mean scores for the proposed and alternative models experiments using both discrete and continuous datasets and the three time windows

To better understand how the predictive capabilities of the models are balanced in each of the classes, we plot the models in a ROC-fashion plot (True positive rate - TPR vs True negative rate - TNR).

Figures 4 and 5 show such plots for the discrete and continuous cases respectively. Off-diagonal line represents the results of a random predictor and the diagonal line represents the results of a balanced predictor. The perfect predictor is in the top right corner of the plots. With respect to the discrete case (Fig. 4) we note that, even though all the models have close Gmean scores (except the LSTM in 48 h and 72 h that are near to random predictors), the models present some variability in the balance between TPR vs TNR, ranging from 0.65 to 0.73 in the TPR axis and from 0.5 to 0.6 in the TNR axis. It is interesting to note that BN models show values closer to the center in both ranges and a small variability to the time window when compared to the other model types. The increased predictability of the TPR and TNR values in the BN models along with their

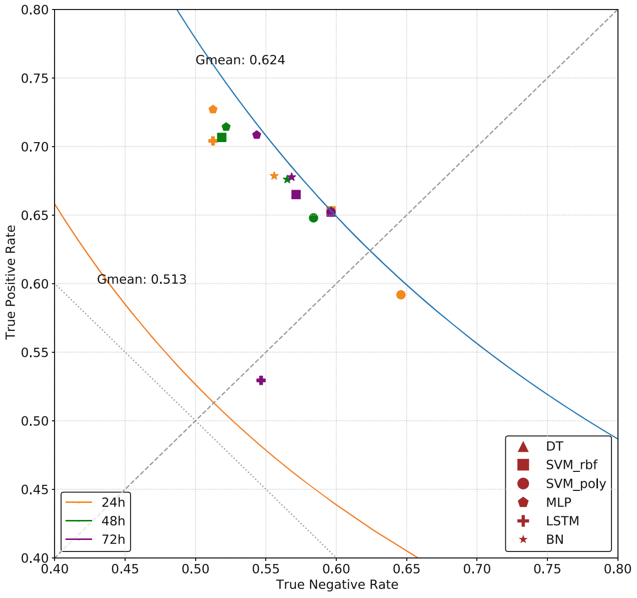

**Fig. 4.** TPR vs TNR graph for the proposed and alternative models experiments using discrete data and the three time windows

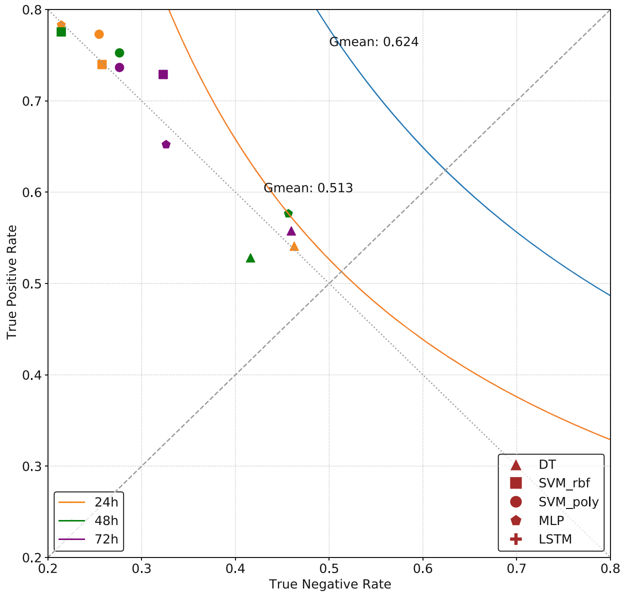

**Fig. 5.** TPR vs TNR graph for the proposed and alternative models experiments using continuous data and the three time windows

greater robustness to the time windows makes this approach attractive for the closing direction prediction task.

With respect to the continuous case (Fig. 5) the models are mostly near to the random predictions (represented by the off-diagonal). This means that the same quantity of predictability gained in one class is lost in the other.

It is apparent that the raw continuous data pose difficulties for the studied models to learn useful patterns and some data transformation is needed to facilitate this task, as demonstrated with the results in discretized data.

Finally, a web application was developed to show the capabilities of the proposed approach in predicting the closing direction of the S&P/BVL Peru General Index. Each hour the application connects to Yahoo Finance api and retrieves the closing index values of the markets closed at that time. With such information the model makes the prediction and shows it in the web application. The model structure and conditional probability distribution are estimated every 24 h. The model structure can be visualised in the web application, which is hosted at https://chapi-tesis.shinyapps.io/code2/.

## 4 Conclusion

A Bayesian network approach was proposed to model and forecast the S&P/BVL Peru General index, based on representative stock market indices from four continents. The predictive capabilities of the proposed approach were compared against popular machine learning methods, showing competitive results at classifying both the positive and negative classes using different time windows. One of the advantages of our approach is that it doesn't require all the variables to be known at the prediction time, which is common in stocks markets (some indices may not be available or are still open and therefore their closing direction is unknown). This property allows us to develop the web application that can predict the target market at any time. The approach allows us to specify the temporal flow of information between stock markets, which gives the possibility of generating models with possible causal interpretation. Unlike many existing models that are considered black boxes, our approach graphically represents the relationships between the dependent variables and the target variable, facilitating the understanding of the domain.

As future works we would like to extend the approach to incorporate into the model certain economic variables that investors usually consider in their decisions, such as interest rates, dollar prices, GDP, etc. In the same way, we are planning to extract sentiment indices of market news and tweets to incorporate into our approach.

**Acknowledgment.** The authors gratefully acknowledge financial support by Pontifical Catholic University of Peru (CAP program, project ID 735).

## References

1. Asadi, S., Hadavandi, E., Mehmanpazir, F., Nakhostin, M.M.: Hybridization of evolutionary levenberg-marquardt neural networks and data pre-processing for stock market prediction. Knowl.-Based Syst. **35**, 245–258 (2012)
2. Basak, S., Kar, S., Saha, S., Khaidem, L., Dey, S.R.: Predicting the direction of stock market prices using tree-based classifiers. North Am. J. Econ. Financ. **47**, 552–567 (2019)
3. Boyacioglu, M.A., Avci, D.: An adaptive network-based fuzzy inference system (anfis) for the prediction of stock market return: the case of the istanbul stock exchange. Expert Syst. Appl. **37**(12), 7908–7912 (2010)
4. Chakravarty, S., Dash, P.: A pso based integrated functional link net and interval type-2 fuzzy logic system for predicting stock market indices. Appl. Soft Comput. **12**(2), 931–941 (2012)
5. Chen, M.-Y., Chen, D.-R., Fan, M.-H., Huang, T.-Y.: International transmission of stock market movements: an adaptive neuro-fuzzy inference system for analysis of TAIEX forecasting. Neural Comput. Appl. **23**(1), 369–378 (2013). https://doi.org/10.1007/s00521-013-1461-4
6. Chen, S.M., Chang, Y.C.: Multi-variable fuzzy forecasting based on fuzzy clustering and fuzzy rule interpolation techniques. Inf. Sci. **180**(24), 4772–4783 (2010)
7. Chong, E., Han, C., Park, F.C.: Deep learning networks for stock market analysis and prediction: methodology, data representations, and case studies. Expert Syst. Appl. **83**, 187–205 (2017)
8. Ding, X., Zhang, Y., Liu, T., Duan, J.: Deep learning for event-driven stock prediction. In: Proceedings of the 24th International Conference on Artificial Intelligence. pp. 2327–2333. IJCAI$^{TM}$15, AAAI Press (2015)
9. Fischer, T., Krauss, C.: Deep learning with long short-term memory networks for financial market predictions. Eur. J. Oper. Res. **270**(2), 654–669 (2018)
10. Gunduz, H., Yaslan, Y., Cataltepe, Z.: Intraday prediction of borsa istanbul using convolutional neural networks and feature correlations. Knowl.-Based Syst. **137**, 138–148 (2017)
11. Hadavandi, E., Shavandi, H., Ghanbari, A.: Integration of genetic fuzzy systems and artificial neural networks for stock price forecasting. Knowl.-Based Syst. **23**(8), 800–808 (2010)
12. Hsieh, T.J., Hsiao, H.F., Yeh, W.C.: Forecasting stock markets using wavelet transforms and recurrent neural networks: an integrated system based on artificial bee colony algorithm. Appl. Soft Comput. **11**(2), 2510–2525 (2011)
13. Huang, W., Nakamori, Y., Wang, S.Y.: Forecasting stock market movement direction with support vector machine. Comput. Oper. Res. **32**(10), 2513–2522 (2005). applications of Neural Networks
14. Ince, H., Trafalis, T.B.: A hybrid forecasting model for stock market prediction. Econom. Comput. Econom. Cybernet. Stud. Res. **51**(3), 263–280 (2017)
15. Jia, J., Zhao, A., Guan, S.: Forecasting based on high-order fuzzy-fluctuation trends and particle swarm optimization machine learning. Symmetry-Basel **9**(7) (2017)
16. Kara, Y., Acar, M., Kaan, Ö.: Expert Systems with applications predicting direction of stock price index movement using artificial neural networks and support vector machines: The sample of the Istanbul Stock Exchange. Expert Syst. Appl. **38**(5), 5311–5319 (2011)
17. Kumar, D., Meghwani, S.S., Thakur, M.: Proximal support vector machine based hybrid prediction models for trend forecasting in financial markets. J. Comput. Sci. **17**, 1–13 (2016)

18. Malagrino, L.S., Roman, N.T., Monteiro, A.M.: Forecasting stock market index daily direction: a Bayesian Network approach. Expert Syst. Appl. **105**, 11–22 (2018)
19. Misra, P., Chaurasia, S.: Data-driven trend forecasting in stock market using machine learning techniques. J. Inform. Technol. Res. **13**(1), 130–149 (2020)
20. Na, S.H., Sohn, S.Y.: Short communication: Forecasting changes in korea composite stock price index (kospi) using association rules. Expert Syst. Appl. **38**(7), 9046–9049 (2011)
21. Patel, J., Shah, S., Thakkar, P., Kotecha, K.: Predicting stock and stock price index movement using trend deterministic data preparation and machine learning techniques. Expert Syst. Appl. **42**(1), 259–268 (2015)
22. Ren, R., Wu, D.D., Liu, T.: Forecasting stock market movement direction using sentiment analysis and support vector machine. IEEE Syst. J. **13**(1), 760–770 (2019)
23. Sezer, O.B., Ozbayoglu, A.M.: Algorithmic financial trading with deep convolutional neural networks: time series to image conversion approach. Appl. Soft Comput. **70**, 525–538 (2018)
24. Singh, P., Borah, B.: High-order fuzzy-neuro expert system for time series forecasting. Know.-Based Syst. **46**, 12–21 (2013)
25. Singh, P., Borah, B.: Forecasting stock index price based on m-factors fuzzy time series and particle swarm optimization. Int. J. Approx. Reasoning **55**(3), 812–833 (2014)
26. Ticknor, J.L.: A bayesian regularized artificial neural network for stock market forecasting. Expert Syst. Appl. **40**(14), 5501–5506 (2013)
27. Tsamardinos, I., Brown, L.E., Aliferis, C.F.: The max-min hill-climbing bayesian network structure learning algorithm. Mach. Learn. **65**(1), 31–78 (2006)
28. Vargas, M.R., de Lima, B.S.L.P., Evsukoff, A.G.: Deep learning for stock market prediction from financial news articles. In: 2017 IEEE International Conference on Computational Intelligence and Virtual Environments for Measurement Systems and Applications (CIVEMSA), pp. 60–65 (2017)
29. Yu, L., Chen, H., Wang, S., Lai, K.K.: Evolving least squares support vector machines for stock market trend mining. IEEE Trans. Evol. Comput. **13**(1), 87–102 (2009)
30. Zhang, M., Jiang, X., Fang, Z., Zeng, Y., Xu, K.: High-order Hidden Markov Model for trend prediction in financial time series. Physica Stat. Mech. Appl. **517**, 1–12 (2019)