Financial Time Series Analysis and Prediction With Feature Engineering and Support Vector Machines

Newton Linchen[1],

[1] Department of Computer Science, Federal University of Rio Grande do Sul (UFRGS)

`contato@linchen.com.br`

Abstract - Time series analysis and prediction has been treated as a regression problem, specially in the field of Econometrics. In order to fit a regression model, the financial time series must be converted from a non-stationary process into a stationary process. We believe this method removes important data relationships and we propose a method for time series analysis and prediction using a classification algorithm (Support Vector Machine) and a new set of time series features. We evaluate this model using the Bovespa Index Futures Contract (Ibovespa Futuro), by making short-term predictions in a simulated environment.

Keywords - Algorithmic trading (algo-trading), Support Vector Machines; classification; prediction; stock index futures; time series.

I. INTRODUCTION

Time series, including those of financial data, are regarded to have as features: *Trend, Seasonality, Cycles and Serial Dependence*. In order to make predictions about time series in the future, practitioners rely mostly on regression models, and it's a requirement that some features must be transformed to fit those models.

In particular, the *Trend* feature must be removed, and the "de-trended" time series is now converted from a non-stationary process into a stationary one. This method is applied because regression modeling would lead to spurious correlations and predictions, if the time series would have *Trends*.

However, we believe that time series contains numerous intrinsic and non-linear relationships, and to remove any of it's features *a priori*, in order to fit the model requirements, would lead to a data loss which would make difficult if not impossible to make accurate and useful predictions.

In this work, we propose a different way to model, analyze and predict financial time series, using it's raw inputs and not removing any features *a priori*. To achieve this prediction task, we propose new features which capture *spatial price relationships*, and use a classification algorithm, Support Vector Machine, to learn non-linear relationships in the data and to make predictions.

The remainder of this paper is structured as follows: Section II reviews related work, i.e., cases in which the Support Vector Machine algorithm was applied to classify and predict financial time series. Section III deals with the implementation of our intuition in a classification model. Section IV evaluates the findings and results. Section V is dedicated to Conclusion and future work.

II. RELATED WORK

We have encountered few reports of Support Vector Machines in the field of financial trading. The granularity of the datasets were daily data points (daily close prices) in every case. The markets covered by these studies were stocks and index futures contracts from the U.S. market, the German market and the French market.

III. IMPLEMENTATION

In order to assess the meaning and the merit of our model, we should understand the intuition behind it. After that, we will demonstrate the feature engineering and the classification algorithm.

A. Intuition

The financial time series of securities or futures contract prices are an output (or result) of a decision-making process carried out by investors, which make the decisions to buy, sell or do nothing, during a continuous auction at the Stock or Futures Exchange trading session.

So, the first element of our intuition is that prices are not "entities" or "elements", nor some "random phenomena", disconnected from real world decision making and financial consequences. Prices are real representations of decision processes that account for massive capital allocation.

In a financial times series, prices are a human-driven phenomena, and therefore, any analysis of price series should account for the behavior (or psychology) of the decision-makers (investors). How do we do that?

When deciding whether to buy, sell or do nothing when it comes to a security or futures contract, investors usually rely on the prices themselves and apply to those prices whatever proprietary modeling they have at their disposal, in order to assess the viability of a trading decision to buy or sell.

In other hand, the practitioners of time series modeling and prediction (usually Econometricians) argue that time series as they are (with *Trend, Seasonality,* and other inherent features) are unfeasible to model. They propose various transformations, in order to fit the data into their regression-driven models.

The investor, however, rarely (if ever) *de-trends* a price series, and therefore doesn't transform it in a *stationary process* before analyzing it and taking action (to buy or sell). They take price at face value, and analyze it by measuring it's path (uptrend, downtrend), and it's relative position in an specific time window.

As every investor has it's preferences, biases and trading models, it's not in the scope of this work to generalize a prediction model that would explain all price behavior. Instead, we are trying to model the investor's framework into a classification problem., in which we try to emulate their reasoning in order to find hidden properties in the price data. The first step to do that is by feature engineering.

B. Dataset

We used the traded prices for the Bovespa Stock Index Futures Contract ("mini-índice") from 2011 to 2015. The granularity was the 15 minute price intervals, each data point with the values of OHLC (Open, High, Low, Close) for the time interval.

C. Feature Engineering

We have created five features that would represent price relationships and insert the current price , at any time *t,* into a *spatial price framework*. The features are:

"F.H.4" (From High.4) - The linear (%) difference between the current price and the highest price recorded in a four-day time window. (A negative value).

"F.L.4" (From Low.4) - The linear (%) difference between the current price and the lowest price recorded in a four-day time window. (A positive value).

"F.H.0" (From High.0) - The linear (%) difference between the current price and the highest price recorded in the current trading session (intraday). (A negative value).

"F.L.0" (From Low.0) - The linear (%) difference between the current price and the lowest price recorded in the current trading session (intraday). (A positive value).

"F.OPEN.0" (From Open.0) - The linear (%) difference between the current price and the opening price of the current trading session (intraday). (A positive or negative value, depending on the current price).

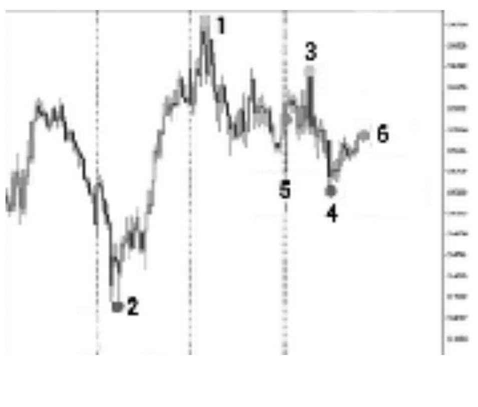

Figure 1. Significant prices of the time series used in the features: 1. Highest price of the four-day time window, 2. the lowest price of the same window, 3. the highest price of the trading session, 4. the lowest price of the session, 5. the opening price for the session and 6. the current price.

The features we propose try to represent some of the reasoning investors use in order to decide whether to buy, sell, or do nothing at any time, given the price relationships. They can be seen in Figure 2.

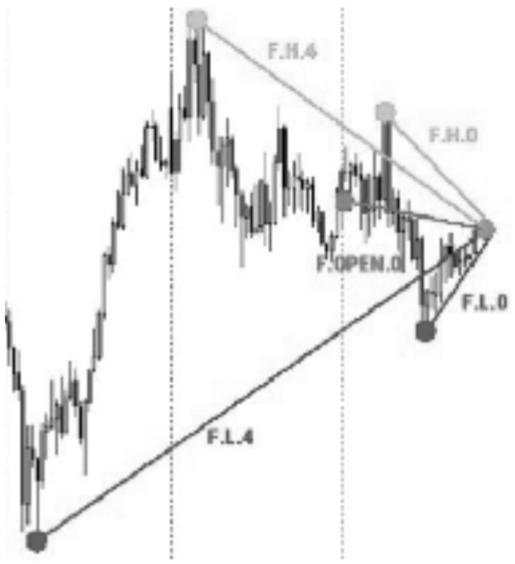

Figure 2. Transforming the significant prices into features.

D. Support Vector Machine

Support Vector Machine (SVM) is a very well-stablished learning algorithm that has been successfully applied to a number of fields. It can be used both for regression and classification, however, it's mostly used as a classification algorithm.

When modeling financial data, SVM is an interesting choice due to the following reasons: it doesn't make strong assumptions on the data, it doesn't require a de-trended time series, (to convert it into a stationary process), and since it's not an empirical error minimization method, it should be robust enough to not overfit the data.

In SVM, each data item is plotted as a point in an n-dimensional space. (The n corresponds to the number of features we are using in the model). The values of the features are the values of a particular coordinate. The classification is made by finding the decision boundary (hyperplane) that best differentiate the two (or more) classes.

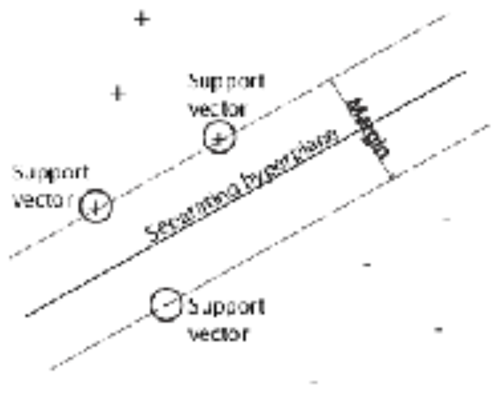

Figure 3. The linear SVM hyperplane separating the two classes of data points.

On finding the best hyper-parameters among C, Gamma and Kernel, for the SVM algorithm, the hyper-parameters were iterated over to arrive at the best combination for the given training data. We chose 4 test values for 'c' and 3 test values for 'g'.

c = [10,100,1000,10000]
g = [1e-2,1e-1,1e0]

The Kernel function chosen was the RBF Kernel (Radial Basis, or Gaussian), for the dataset would present non-linear relationships.

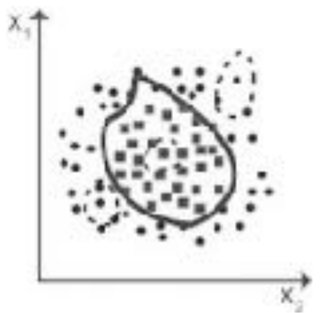

Figure 4. The RBF (Gaussian) Kernel representation.

E. Return Function and Output Signal

The return function was set as the *linear return (%)* of the price from the actual time *t* (which is the time for the "current" price at the series, in each data point) until 60 minutes into the "future". In other words, if we bought the futures contract at the *time stamp*, for example, of 09:45, our "investment" would last until 10:45, with a 60 minutes duration. Our return would be the % difference between the end price and the start price. This is a very short-term approach for investing and it's called *day trading*, by the investors.

The return was calculated for the whole series, and in the training process, the algorithm would classify the returns into three categories: equal or above the 66 quantile, between the 33 and 66 quantile, and, equal or below the 33 quantile.

The algorithm was trained to generate the following output signals:

+1, or a "buy" signal (in trading parlance, to go "long"), if the classification indicated the return would be *equal or above the 66 quantile*.

0, or a "do nothing" signal, if the classification indicated the return would be in *between the 33 quantile and the 66 quantile*.

-1, or a "sell" signal (in trading parlance, to go "short"), if the classification indicated the return would be *equal or below the 33 quantile*.

F. Data Split and Cross-Validation

The dataset was split in two blocks: 80 percent for training (from 2011 to 2014) and 20 percent for testing (from 2014 to 2015, comprising eight and a half months).

A k-fold Cross-Validation technique was used in the training set, with k=7.

IV. EVALUATION

The model generated 56% prediction success, and the simulated *financial return (minus transaction costs)* amounted to a 170% gain, during the testing period.

For the same period, the *Trivial Strategy* (to buy if the last data point was also classified as a buy signal, to sell if the last data point was classified as a sell signal) generated a *-39%* return, and the *Buy-and-hold* strategy (in which we simply buy at the start of the test period and sell at the end of it) had a *-9%* return.

	Model	*Trivial Strategy*	*Buy-and-hold*
Accuracy	56%	48%	N.A.
Return	170%	-39%	-9%

Figure 5. Results.

The *return* as an *equity curve* can be compared to the actual price series:

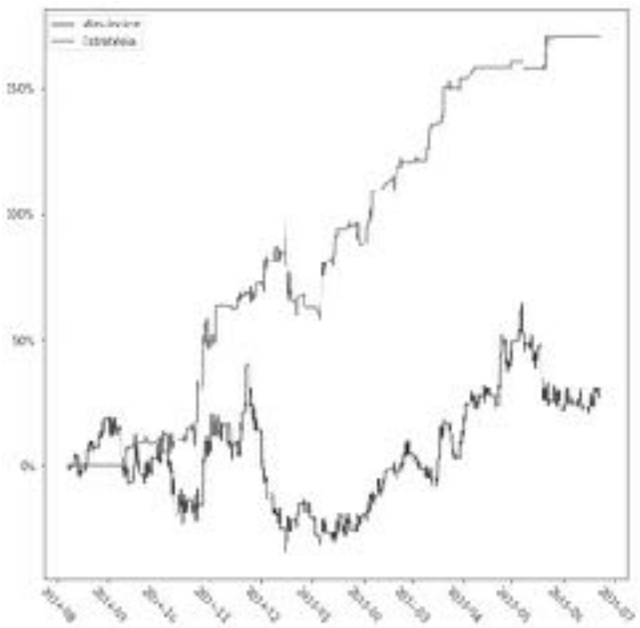

Figure 6. Return plotted against the time series.

V. CONCLUSION AND FUTURE WORK

Despite the non-stationarity of the time series, the model performed a successful classification of buying and selling opportunities, which were applied to the test data with acceptable accuracy (from the investment practitioner's point of view) and excellent *% financial return*,

(+170%), that was discounted for *slippage and trading costs* in each "transaction".

The *Feature Engineering* was key in this process, as the feature were able:

a. To model price behavior according to an investment practitioner's point of view;

b. To maintain the time series properties while allowing for Cross-Validation;

c. To capture significant price relationships that were instrument for the SVM classification and predictions.

The Support Vector Machine algorithm was key in the way of learning the non-linear relationships of the dataset and accurately classifying and predicting.

Future work will evolve this model with the addition of new features, hyper-parameter tuning and the addition of an exogenous time series in order to turn the model into a multivariate analysis model.

REFERENCES

Cao, L., and Tay, F. (2001) "Financial Forecasting Using Support Vector Machines", Neural Computing & Applications. Springer-Verlag London Limited.

Cao, L. and Tay, F. (2002) "Modified support vector machines in financial time series forecasting". Neurocomputing. Elsevier Science B.V.

Cao, L. and Tay, F. (2003) "Support Vector Machine With Adaptive Parameters in Financial Time Series Forecasting". IEEE Transactions on Neural Networks, Vol.14, N.º 6.

King, C., Vandrot, C. and Weng, J. (2009) "A SVM Approach To Stock Trading".

Madge, S. (2015) "Predicting Stock Price Direction using Support Vector Machines". Independent Work Report Spring 2015.