FUNDAÇÃO GETULIO VARGAS
ESCOLA DE ECONOMIA DE SÃO PAULO

DANIEL LINS MATTOS

COMPARING MACHINE LEARNING ALGORITHM PERFORMANCE FOR AUTOMATED TRADING BASED ON FUNDAMENTALS

SÃO PAULO

2019

DANIEL LINS MATTOS

COMPARING MACHINE LEARNING ALGORITHM PERFORMANCE FOR AUTOMATED TRADING BASED ON FUNDAMENTALS

Dissertação apresentada à Escola de Economia de São Paulo da Fundação Getulio Vargas, como requisito para obtenção do título de Mestre Profissional em Economia

Campo de Conhecimento: Macroeconomia Financeira

Orientador: Prof. Dr. Ricardo Ratner Rochman

SÃO PAULO

2019

DANIEL LINS MATTOS

COMPARING MACHINE LEARNING ALGORITHM PERFORMANCE FOR AUTOMATED TRADING BASED ON FUNDAMENTALS

Dissertação apresentada à Escola de Economia de São Paulo da Fundação Getulio Vargas, como requisito para obtenção do título de Mestre Profissional em Economia

Campo de Conhecimento: Macroeconomia Financeira

Orientador: Prof. Dr. Ricardo Ratner Rochman

Data de aprovação: 05/07/2019

Banca Examinadora:

_____

Ricardo Ratner Rochman
(Orientador)
FGV – EESP

_____

Gustavo Mirapalheta
FGV – EESP

_____

Vladimir Pinheiro Ponczek
FGV – EESP

Mattos, Daniel Lins.
Comparing machine learning algorithm performance for automated trading based on fundamentals / Daniel Lins Mattos. - 2019.
33 f.

Orientador: Ricardo Ratner Rochman.
Dissertação (mestrado profissional MPFE) – Fundação Getulio Vargas, Escola de Economia de São Paulo.

1. Aprendizado do computador. 2. Ações (Finanças). 3. Análise de regressão. 4. Modelos lineares (Estatística). 5. Modelos não-lineares (Estatística). I. Rochman, Ricardo Ratner. II. Dissertação (mestrado profissional MPFE) – Escola de Economia de São Paulo. III. Fundação Getulio Vargas. IV. Título.

CDU 336.763.2

Ficha Catalográfica elaborada por: Cristiane de Oliveira CRB SP-008061/O
Biblioteca Karl A. Boedecker da Fundação Getulio Vargas - SP

ABSTRACT

Recent applications of machine learning in finance have highlighted the capacity of these techniques to predict asset returns. In this article, we compare different machine learning methodologies in forecasting stock returns over several one-month periods, based on fundamentals and price data. For this purpose, the fitted models are applied in a simulated trading strategy (“backtest”). Preliminary results suggest that nonlinear algorithms such as Random Forests, Extreme Gradient Boosting and Support Vector Machine may be superior to linear algorithms such as Linear Regression and Lasso in predicting stock returns. However, for the set of factors studied in this article and for the assets analyzed, these algorithms do not perform better than a buy-and-hold strategy for most assets.

Keywords: machine learning, Extreme Gradient Boosting, Support Vector Machine, Random Forests, Linear Regression, Lasso, backtesting, stocks, fundamentals

RESUMO

Aplicações recentes de *machine learning* em finanças têm destacado a capacidade dessas técnicas em prever retornos de ativos. Neste artigo, comparamos diferentes metodologias de *machine learning* na previsão de retornos de ações durante vários períodos de um mês, com base em fundamentos e dados de preço. Para isso, os modelos ajustados são aplicados em uma estratégia de negociação simulada ("*backtest*"). Resultados preliminares sugerem que algoritmos não lineares como *Random Forests*, *Extreme Gradient Boosting* e *Support Vector Machine* podem ser superiores a algoritmos lineares como Regressão Linear e Lasso ao prever retornos de ações. Contudo, para o conjunto de fatores estudado neste artigo e para os ativos analisados, estes algoritmos não apresentam performance superior a uma estratégia de *buy-and-hold* para a maior parte dos ativos.

Palavras-chave: *machine learning*, *Extreme Gradient Boosting*, *Support Vector Machine*, *Random Forests,* Regressão Linear, Lasso, *backtesting*, ações, fundamentos

LIST OF FIGURES

LIST OF TABLES

TABLE OF CONTENTS

1. Introduction

Investing in stocks has always been an inherently complex activity, which has attracted the attention of a great number of practitioners and academia over the years. Much of that complexity has been attributed to that fact that the prices of such instruments are determined by the decisions of many individuals with different goals and personalities, who themselves are influenced by and react to different outside factors.

There are century-old techniques that try to predict asset price movements, including stocks, based historical prices alone. This is broadly defined as “Technical Analysis”. The ensemble of these techniques attempts to forecast future prices by studying the behavior of past prices and other related summary statistics about securities trading. According to Murphy (1999), technicians believe that the price of a company’s stock already reflects all the information that could possibly be extracted from accounting figures as well as news sources. In this sense, technicians usually model the historical behavior of a financial asset as a time series, believing that the history tends to repeat itself.

In the 1930s, Benjamin Graham and David Dodd were the first notable authors to build a solid foundation of security analysis based on a combination of accounting and market data, which became known as “Fundamental Analysis”. They advised that, before buying or selling a stock, investors analyze the company’s assets, growth, and ratios that indicate the company’s price relative to accounting figures, such as price to earnings, price to book value, among other metrics. They claimed that stocks that are cheap according to these metrics have a higher chance of outperforming more expensive stocks. This investment style became known as “Value Investing”.

Value Investing is not the only style that makes use of Fundamental Analysis tools. Certain investors argue that cheap stocks are cheap for a good reason, and their prices may not revert to fair value. According to Lowenstein (2008), Warren Buffet, who was a student of Benjamin Graham and worked with him early in his career, adapted the Value Investing approach to accommodate more growth and quality of earnings, while being more flexible with regards to price he would pay for a company’s stock. He argues that companies which deliver stable growth, while keeping profitability margins high, will keep driving their stock prices higher. Even though he is still considered a

value investor by the investment management community, “Growth” and “Quality” have become investment styles, with a number of investors chasing these factors exclusively.

These investment frameworks were created by practitioners rather than academia, and therefore lack the economic rigor of scientific work. The Efficient Market Hypothesis (“EMH”), conceived by Fama (1963, 1965a, 1965b, 1970), despite its scientific foundations, is built on the assumption that investors are rational and that all information is available to them. EMH, in its semi-strong form, claims that no agent possessing any publicly available historical information would do better than the random walk in trying to predict security price movement. Even though it is a purely theoretical concept, EMH is one of the cornerstones of modern financial theory.

Despite EMH’s success, several articles that try to look for evidence against it have been published. And some of these articles do show evidence of consistent outperformance by techniques that combine analysis of past returns and selected fundamentals to predict stock returns. Most research aimed at finding such models relied heavily on linear regressions until the early 90s, potentially influenced by traditional econometrics and limited by the processing capacity of computers at the time, which would hinder the use of more computationally intensive, non-linear models.

In recent years, automated trading has taken over an important chunk of the volume in various exchanges, as evidenced by the surge in algorithmic and high frequency trading (“HFT”), which, as of 2009, accounted for more than 60% of all US equity trading volume according to The New York Times (2012). HFT and algorithmic trading usually focus on profiting from small price discrepancies in a split-second timeframe, but among the algorithms driving investment management decisions, there are some that have a longer investment horizon. In either case, automation has become the norm in today’s market.

Machine learning (“ML”) is a method of data analysis that automates analytical model building. It is comprised of algorithms and mathematical models that computer systems use to progressively improve their performance on a specific task. It is not a new concept in mathematics and computing. Linear regression itself can be considered an

ML technique. However, most ML models could not be widely used until recently due to their heavy computational requirements.

Today, with much higher processing power and specialized software, the application of machine learning models in trying to predict asset returns has become widespread. Several authors argue that some ML techniques can be used to predict security price movement more accurately than a random walk or a regression.

In this analysis we'll compare different machine learning models in selecting stocks listed on the São Paulo Stock Exchange ("B3") as part of a simple investment strategy. The main objective of the analysis contained herein is to compare the performance of traditional algorithms, such as linear regression and penalized regression (Lasso), to more modern techniques such as Random Forests, Extreme Gradient Boosting, and Support Vector Machine in making investment decisions for this particular strategy.

2. Related Literature

Haugen and Baker (1996) find evidence that the return of stock prices can be predicted to some degree by cross-sectional data on risk, liquidity, price-level, growth potential, and price history. They also find that stocks with higher expected and realized rates of return are of lower risk than stocks with lower return, suggesting a potential failure in EMH.

Alberg and Lipton (2017) apply a machine learning approach which uses multi-layer perceptron ("MLP") and recursive neural network ("RNN"), to predict stock fundamentals based on their historical values and past stock returns. They find this approach to consistently yield better results than a linear regression and naïve prediction, which basically replicates last period's figures.

According to Vanstone and Finnie (2009), even though several researchers have investigated how to use rules-based and statistical learning systems to predict financial market movements, there is no well-established and tested methodology which describes how to create an autonomous trading system. Once more, Vanstone and

Finnie (2010) find that the main reason why successful autonomous trading methods are not communicated to the scientific community is that investors try to keep their intellectual capital saved.

R.C. Cavalcante et al. (2016) conducted an extensive survey of papers in computational investing and presented a complete framework to build and test algorithmic trading strategies, presented in Figure 1.

Kolanovic et al. (2017) compare the performance of different machine learning approaches to predict next day returns of 9 US sector ETFs based on 8 macro factors. The authors found that XGBoost had the best performance among the models.

We will draw upon some of the fundamental and technical factors used in the work by Haugen and Baker (1996) and Alberg and Lipton (2017) to derive our experiment. The framework adopted in the analysis will be that proposed by R.C. Cavalcante et al. (2016) and the models applied – Linear Regression, Lasso, Random Forests, XGBoost, and Support Vector Machine – are based on the work of Kolanovic et al. (2017).

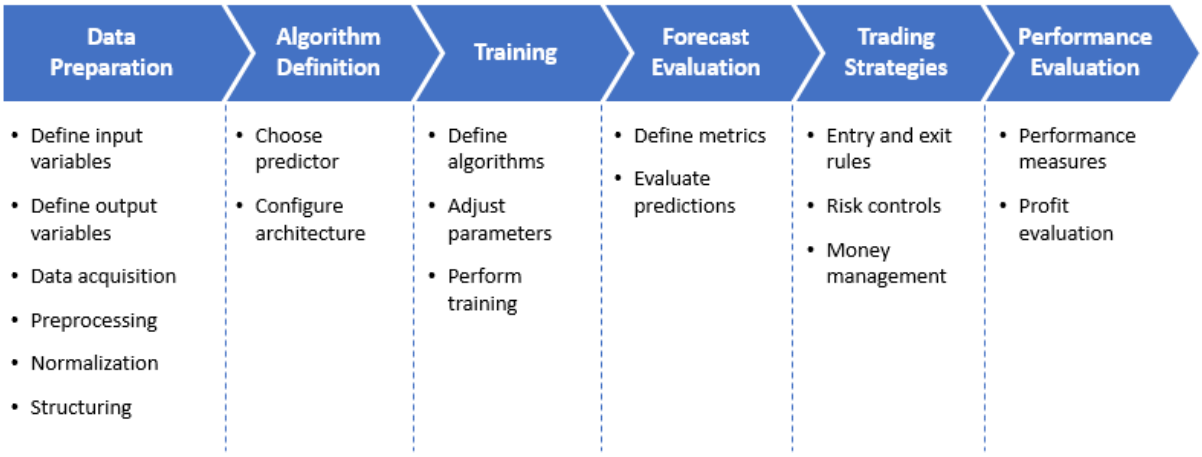

Figure 1. High-level Framework to Build and Test Algorithmic Trading Strategies

2.1 Algorithms

In the experiment, we test a total of five algorithms, namely Ordinary Least Squares ("OLS") Linear Regression, Lasso Regression ("Lasso"), Random Forests ("RF"), Extreme Gradient Boosting ("XGBoost"), and Support Vector Machine ("SVM").

The choice of these algorithms reflects the major classes of algorithms studied by Kolanovic et al. (2017), with the exception of neural networks methods, due to their higher complexity, computational requirements, and no proven superiority to other non-linear methods such as RF, XGBoost, and SVM for structured/tabular data.

The general principle behind each of the algorithms, which we present in the following sections, is also largely based on the work of Kolanovic et al. (2017).

2.1.1 Ordinary Least Squares Regression (OLS)

In an OLS regression, we forecast the value of y in Equation 1 to be a linear combination of the predicting variables, x_1, x_2,…, x_n. In short we assume that variable y has 'Betas' to a number of variables x (plus some random noise).

Eq. 1. $y = \beta_0 + \sum_{i=1}^{n} \beta_i x_i + \epsilon_i$

To find the 'betas' β_0, β_1,…, β_n, OLS minimizes the historical error (square of error) between actual observations of variable 'y', and predicted (or model) values of the variable. This is the reason the method is also called least-squares (since it minimizes the square of errors):

OLS: Minimize Historical Sum of

Eq. 2. $[y - (\beta_0 + \sum_{i=1}^{n} \beta_i x_i)]^2$

2.1.2 Lasso Regression

Despite its simplicity, the OLS minimization is not stable and can yield spurious and/or large values of betas. One way to prevent that from occurring is to change the objective function in the minimization above. Instead of minimizing the least-squares objective, we can modify it by adding a penalty term that reflects our aversion towards complex models with large “betas”. If we change the objective to include a penalty term equal to the absolute value of the beta coefficients, i.e.

Lasso: Minimize Historical Sum of

Eq. 3. $[y - (\beta_0 + \sum_{i=1}^{n} \beta_i x_i)]^2 + \alpha \sum_{i=1}^{n} |\beta_i|$

then the optimizer will set “unnecessary” and very large betas to zero. The addition of a penalty term equal to the absolute value of the coefficients is called L1 regularization and the modified linear regression procedure is called Lasso. By concentrating only on the most relevant predictors, Lasso performs an implicit feature selection. If we set α = 0, then we recover the coefficients of ordinary linear regression. As α increases, we choose a smaller and smaller set of predictors, concentrating only on the most important ones.

2.1.3 Random Forests

Decision tree models are essentially flow charts used commonly in business management and financial analysis. To arrive at a "decision", the analyst asks a series of questions. At each step, a decision tree branches based on the answer. The questions are asked in order of importance; we ask a question on the most important factor first, and then subsequently refine the question. In the end, the combination of answers to different questions enables one to make a decision.

Decision trees are one of the simplest non-linear models that can be used to classify outcomes. For example, we can classify whether a stock has strong or weak returns by looking at a few factors such as momentum, volatility, and price-to-book ratio. An example of such a decision tree is shown in Figure 2 below. Decision trees are able to consider interactions between variables, e.g. stock returns are strong if momentum is low, volatility is high, and price-to-book is high.

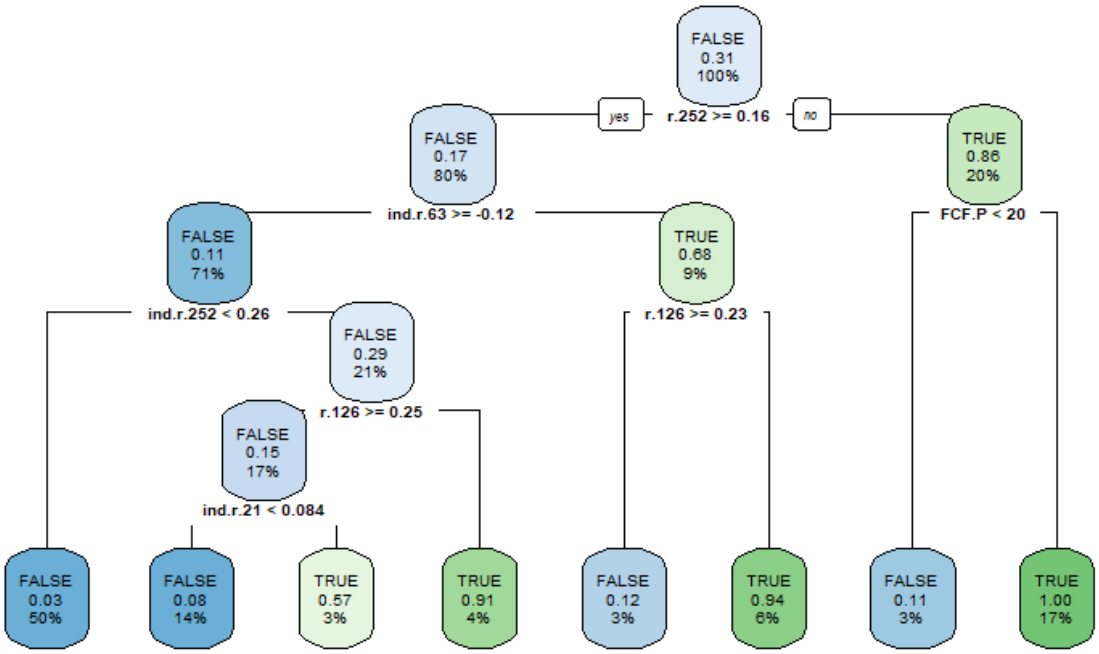

Figure 2. Decision Tree Example

The example above, shown in Figure 2, answers whether we should buy the preferred stock of Gerdau S.A. between October 6 2017 and November 27 2018. Each node shows an answer – true or false – based on the frequency of true values, as well as the percentage of training samples it represents. Even though this is a merely illustrative example, the division of the nodes looks arbitrary. Indeed, the main problem with decision tree is they are prone to overfitting, fitting the data very well during training, but failing to predict well while testing the model.

Random forests improve upon this idea by averaging the output of many decision trees. This process is known as “bagging”. Each decision tree is then fit on a small subset of training examples or is constrained to use only a small subset of input features. Averaging the output of these trees reduces variance of the overall estimator. Alongside support vector machines, random forests were the most popular amongst machine learning methods until the recent invention of XGBoost and rise of Deep Learning.

2.1.4 Extreme Gradient Boosting (XGBoost)

The Extreme Gradient Boosting algorithm relies on classification and regression trees (“CART”) to build a more robust model. This type of model is built sequentially by minimizing errors from previous models while increasing (boosting) the influence of high performing models. CARTs are similar to decision trees, except that at each leaf, we get a continuous score instead of a class label. This is necessary in order to perform gradient descent to minimize errors in the sequential model.

Back to the more formal definition provided by Kolanovic et al. (2017), if a regression tree has T leaves and accepts a vector of size m as input, then we can define a function q: $\mathbb{R}^m \rightarrow$ {1,…, T} that maps an input to a leaf index. If we denote the score at a leaf by the function w, then we can define the k-th tree (within the ensemble of trees considered) as a function $(x) = (\underline{x})$, where $w \in \mathbb{R}^T$. For a training set of size n with samples given by $(\underline{x}_i, y_i)$, $x_i \in \mathbb{R}^m$, $y_i \in \mathbb{R}$, a tree ensemble model will use K additive functions to predict the output as follows: $\widehat{y_i} = \phi(\underline{x}_i) = \sum_{k=1}^{K} f_k(\underline{x}_i)$.

To learn the set of functions in the model, the regularized objective is defined as follows:

Eq. 4. $\mathcal{L}(\phi) = \sum_i l(\hat{y}_i, y_i) + \sum_k \Omega(f_k)$

where $\Omega(f) = yT + \frac{1}{2}\lambda||w||^2$.

The tree ensemble model is optimized in an additive manner. If $\hat{y}_i^{(t)}$ is the prediction of the i-th training example at the t-th stage of boosting iteration, then we seek to augment our ensemble collection of trees by a function f_t that maximizes the following objective:

Eq. 5. $\mathcal{L}^{(t)} = \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t-1)} + f_t(\underline{x}_i)) + \Omega(f_t)$

The objective is approximated by a second-order Taylor expansion and then optimized. For calculation steps, we refer the reader to Chen and Guestrin (2016). To prevent overfitting, XGBoost uses shrinkage (to allow future trees to influence the final model) and feature sub-sampling, as in Random Forest.

2.1.5 Support Vector Machine (SVM)

According to Bambrick (2016), SVMs are based on the idea of finding a hyperplane that best divides a dataset into two or more classes. As a simple example, for a classification task with only two features, one can think of a hyperplane as a line that linearly separates and classifies a set of data.

The distance between the hyperplane and the nearest data point from either set is known as the margin. The goal is to choose a hyperplane with the greatest possible

margin between the hyperplane and any point within the training set, giving a greater chance of new data being classified correctly.

But what happens when there is no clear hyperplane? One can apply a transformation to the data so that it can be classified more effectively in a different space (of higher dimension). This technique is known as kernelling. According to Kolanovic et al. (2017), it enables linear learning algorithms to learn a non-linear boundary without specifying an explicit mapping. One need not stop at a finite-dimensional mapping. The most commonly used kernel function is the Gaussian Radial Basis Function, $k(\vec{x} - \vec{y}) = e^{\frac{\|\vec{x}-\vec{y}\|^2}{2}}$, which corresponds to an infinite-dimensional mapping function φ and yet is computable in just O(n) time.

3. Data and Methodology

The main objective of the experiment is to compare the performance of different machine learning algorithms for trading individual stocks over different periods. Secondly, the analysis will also show the potential of each algorithm to outperform a passive strategy over time. In order to accomplish this, we will simply analyze, for time horizons ranging from one to ten years, the percentage of stocks for which a certain algorithm (strategy) performed better.

In order to assess the performance of different machine learning methods in predicting stock price movements based on technical and fundamental indicators, we chose as our sample space the companies comprising the Ibovespa index of the São Paulo stock exchange between early 2005 and early 2019 and ran a backtest following trading recommendations of the algorithms. A detailed description of the process will be provided below.

We use R version 3.5.1 as the programming language of choice for this experiment, as it offer libraries which cover most of the machine learning algorithms used by the scientific community.

3.1 Selected Assets

The sample used in the analysis is that of the stocks comprising the Ibovespa between January 3rd, 2005 and February 14th 2019. Delisted stocks, as well as those that were initially part of, but were eventually excluded from the index at some point, were also considered in the analysis. Due to the nature of the metrics used, which will be further detailed below, we excluded from the sample stocks of companies in the financial sector as well as stocks with missing or inconsistent data. A full list of the selected stocks is provided on Table 1 in the Appendix.

3.2 Data Acquisition and Pre-Processing

We obtained stock closing prices and most of the fundamentals directly from a Bloomberg terminal, using Bloomberg's Excel plugin. The remaining fundamentals are a combination of the prices and fundamentals downloaded from Bloomberg. All formulas and parameters used available on the project repository on Gitlab[1].

The data is loaded into R as a list of data frames. Each list element corresponds to a stock, and each data frame contains all of the data available for each stock.

The closing prices used in the analysis are adjusted for splits, dividends, and share buybacks. With the daily closing prices, we calculated 1-month (21 business days), 3-month (63 business days), 6-month (126 business days), and 1-year (252 business days) log returns for individual stocks and for the Ibovespa index (using the same timeframes). These returns comprise the technical input variables which we used in the analysis.

[1] Access to the repository is available upon request. Please contact the author.

The fundamental input variables, which we divided into separate categories are described here. For indication of profitability and returns: ROIC (return on invested capital), ROE (return on equity), and profit margin (net income as a percentage of revenue), which are all available directly from Bloomberg on a quarterly basis. For indication of the level of growth we used: annual sales growth and EBITDA (earnings before interest, taxes, depreciation, and amortization for the last twelve months) growth, which are also available directly from Bloomberg on a quarterly basis. In order to assess value, we used the following ratios: to last twelve months earnings to price (also known as earnings yield), book value of equity (of the latest quarter) to price, trailing twelve months dividend yield (based on the last closing price), free cash flow (last twelve months) to price, and EBITDA yield (EBITDA to enterprise value). In order to calculate these ratios, we calculated the enterprise value as the market capitalization plus net debt and minority interest, which are all available from Bloomberg. This was done in order to have daily values for the ratios.

The data obtained using the Excel plugin was then loaded into the main program, which reads the spreadsheets and calculates the custom features which are fed to the machine learning algorithms.

The chosen output variable for the analysis was the forward 21 business days (approximately one month, on average) log return, which was also calculated based on historical prices from Bloomberg.

The resulting number of input variables (features) is 17, of which 8 are technical (based on past returns), and 9 are fundamental (based on actual company financial performance).

As the algorithms used do not require normalization, except SVM, and assuming that the features used should be stationary over a very long period of time (at least a full economic cycle), we have assumed, conceptually, that there is no need to differentiate the data.

It is important to note that, as of the date of the experiment, Bloomberg provided all fundamentals with the corresponding closing dates for each quarter (e.g. June 30th for the last twelve months EBITDA of a second quarter report), not the filing date. During

the data pre-processing stage, we made sure that all fundamentals were only available for the algorithms to process as of the filing date, thus avoiding look ahead bias.

3.3 Training and Testing Split

For each stock, the algorithms are trained for an incremental (initially 300-day) rolling window, followed by a 30 day test period. At each turn we add 30 days to the training window, which is equivalent to the size of the prior testing period. The actual number of data points for each training window is in practice 21 days less than the size of the window, as the forward returns (21 business days) cannot stay outside of the training period in order to avoid look-ahead bias. We also need to account for the input variables of the returns, which require a minimum of 252 closing price data points on top of the training and testing periods. For instance, if a given stock has 612 closing price data points, 252 will be used for past returns calculation. The remaining 360 will allow for 2 train-test windows, the first with 300 + 30 days and the second with 330 + 30 days. First, the algorithm is trained for 279 days out of the first 300 day window (which includes the inputs and output variable). It is then used for 30 days to predict future returns and make buy, hold, or sell recommendations for the stock. The performance for this 30 day period is recorded. Then, a new 330 day training window is established, incorporating the prior 30-day test period and the process is repeated.

The trading strategies applied to each stock is a straightforward one: if the expected return for the next 21 business days is greater than zero, a long signal is recorded, otherwise the investor should refrain from buying the stock and invest in interbank deposit certificates, which returns the average overnight CDI rate. The algorithm converts the signal into to an actual position with a one-day lag, in order to avoid look-ahead bias.

We have refrained from shorting stocks in this analysis due to the operational complexity and costs involved in this type of transaction.

3.4 Technical Specifications of Machine Learning Libraries in R

For linear regression, we used a readily available R function “lm” available in version 3.5.1 of the “stats” package to fit the models. The function does not require additional parameters other than the input variables and the output variable.

We used library “glmnet” 2.0-16 to implement Lasso. The function used has the same name as that of the library. This function can implement both Lasso and Ridge regressions. In order to obtain the former, we chose parameter alpha = 1 (this is not equivalent to the coefficient shown on Eq. 3). The other parameter, which is the α penalty coefficient shown on Eq. 3, is called lambda in the glmnet function, and is obtained through an optimization of the penalty coefficient between, testing 100 values between 10,000 and 0.0001, choosing the one that minimizes the error during the training period. Besides the parameters, the input and output variables used were the same as the ones used in the linear model. The glmnet function requires that the data frame be converted into a matrix before it can be used. In order to do that we use the function as.matrix from R’s base library.

To implement the random forest model in R, we used the “randomForest” function from the library with the same name, version 4.6-14. The “ntree” parameter of the function, which defines the number of trees, which will be averaged to get to a more robust tree, is set to 200. Besides the parameters, the input and output variables used were the same as the ones used in the linear model.

We trained the XGBoost model in R using the “xgboost” function of the “xgboost” package, version 0.82.1. This function requires the original data frames to be converted into a sparse matrix. We chose to convert the data frame into a sparse dgCMatrix[2] using the Matrix function, from the Matrix library version 1.2-14, with “sparse” parameter set to “TRUE”. For the “xgboost” function we set the parameter “booster” to “gbtree”, which means the algorithm will use classification and regression trees as the functions which the algorithm will optimize in additive manner. The “max_depth” parameter, which is maximum depth of a tree used in the model, was set

[2] dgCMatrix class is a class of sparse numeric matrices in the compressed, sparse, column-oriented format.

to 10. Finally, the “nrounds” parameter, which indicates the number of iterations during the optimization process, was set to 40.

The SVM model was fit in R using the “svm” function of the “e1071” package, version 1.6-8. It uses Gaussian Radial Basis Function by default, thus potentially capturing the non-linearity between the features and stock returns. The main difference with this algorithm, when compared to the prior ones, is that it requires data normalization. Due to the calculation of distances between the hyperplane and the data, in case the data is not normalized, distortions in the results may arise.

3.5 Putting Everything Together

In the main program, we run all the strategies for each stock, under the same period and conditions, storing the performance of each asset for every strategy in a data frame. Figures 3 and 4 are graphic representations of the data stored for each asset.

4. Results

4.1 Performance of Tested Algorithms

Preliminary results indicate that non-linear algorithms XGBoost, SVM, and Random Forests perform better at predicting stock returns when compared to the other algorithms tested.

Table 2 indicates this group of non-linear algorithms performed better more often than linear algorithms – Linear Regression and Lasso.

However, XGBoost, SVM, and Random Forests do not perform better than a Buy-and-hold strategy for picking individual stocks over the periods analyzed. For the combined full test period, a buy-and-hold strategy would have had the highest returns for 21 out

of the 96 stock sample. This number is 18 for Random Forest, 17 for SVM, 14 for XGBoost, 14 for Lasso, 12 for linear regression.

Table 3 indicates that the total number of trades per stock for the non-linear algorithms is larger than the linear ones, which indicates that their superior performance comes at the cost of more frequent trading.

We also looked at maximum drawdown for all strategies. Buy-and-hold has the highest number of maximum drawdowns for the period, with 74, while Linear Regression, Lasso, Random Forest, XGBoost, and SVM had 3, 2, 5, 2, and 10, respectively.

Figure 3 shows a chart containing the performance of each strategy with Souza Cruz S.A. (CRUZ3). XGBoost and SVM both performed well for this stock until its tender offer in 2015, although the first stayed ahead for most of the period.

Another example is given in Figure 4. In this chart we can see that a Buy-and-hold strategy would have performed best. Among the algorithms, XGBoost was still the leading performer throughout most of the period.

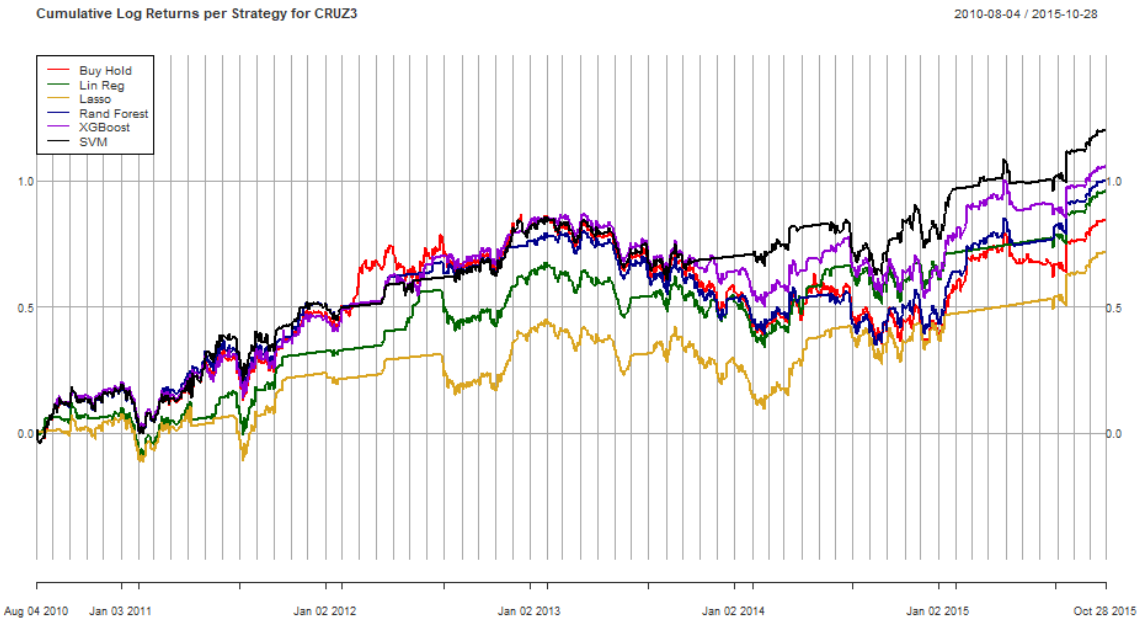

Figure 3. Performance Analysis for CRUZ3

5. Discussion

The results suggest that machine learning offers good alternatives to classic models in explaining stock returns to some extent. Non-linear models such as XGBoost, SVM, and Random Forests seem to offer a superior alternative to linear algorithms OLS and Lasso in explaining future returns based on fundamentals and past returns. However, no model offers perfect predictive capacity, nor works well for all stocks.

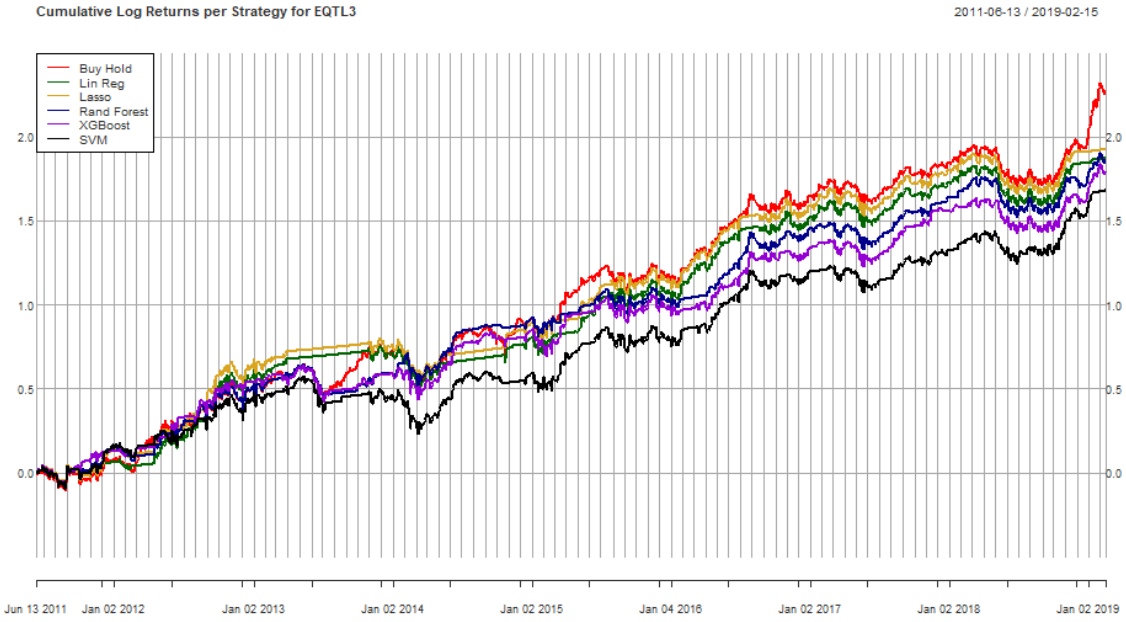

Figure 4. Performance Analysis for EQTL3

In the long run, a buy-and-hold strategy appears to work better for stocks, when compared to the machine learning algorithms. However, there could be several reasons behind this.

The first is the explanatory variables used in this analysis. A more comprehensive, or better choice of variables, could lead to superior predictive capacity for the algorithms. This could be very challenging for Brazilian stocks, as it is very hard to obtain consistent, reliable data for stocks dating back to the early 2000s, which leaves us with little more than 10 to 15 years of data for most stocks. To illustrate this point, for many of the stocks we studied, there was missing data for recent years, or certain Bloomberg

fields were available for one stock and not for another, without a clear reason. This led us to let go of many of the factors which were included in the studies of other authors.

A considerable number of stocks analyzed herein are highly speculative, and their price behavior could have little connection with fundamentals. Having specific variables for industry groups, level of maturity of companies, etc. could be helpful in determining the returns for a given cluster of stocks. In other words, it is not ideal to use a “one size fits all” approach when modeling returns.

The method (and set of variables) used here could be better suited for a mean-reverting regime. When there is a transition from one phase to another in the economic cycle, causing changes to country risk, cost of capital, etc., multiples tend to expand or compress, leading to a trend with no near-term reversion. Having the ability to perceive these changes is more akin to an asset manager (a human) than a computer algorithm. So, managers buy more, or sell more than they usually do, in anticipation of key events in the market cycle.

Lastly, we have trained the algorithms and performed analysis on individual stocks. If algorithms are trained on data from all stocks, it could lead models that are less susceptible to noise.

These tests could be the subject of future work by the authors or by readers who are also interested in studying the application of machine learning algorithms in modeling asset returns. We also suggest as follow-up to this dissertation, the incorporation of portfolio and risk analysis for a more complete perspective on the potential of the techniques that were covered.

References

Kolanovic, Marko (2017), “Big Data and AI Strategies”, J.P. Morgan Research

Haugen, Robert A. and Baker, Nardin L. (1996), “Commonality in the Determinants of Expected Stock Returns”, Journal of Financial Economics

Alberg, John and Zachary C., Lipton (2017) “Improving Factor-Based Quantitative Investing by Forecasting Company Fundamentals”, arXiv

Fischer, Thomas and Krauss, Christopher (2017), “Deep Learning With Long Short-term Memory Networks for Financial Market Predictions”, European Journal of Operational Research

C. Cavalcante, Rodolfo et al. (2016), “Computational Intelligence and Financial Markets: A Survey and Future Directions”, Expert Systems With Applications

Gu, Shihao, et al. (2018), “Empirical Asset Pricing via Machine Learning”

Bambrick, Noel (2016), “Support Vector Machines: A Simple Explanation”, (https://www.kdnuggets.com/2016/07/support-vector-machines-simple-explanation.html)

Murphy, John J. (1999), “Technical Analysis of the Financial Markets: A Comprehensive Guide to Trading Methods and Applications”, New York Institute of Finance

Times Topics: High-Frequency Trading (2012), “Declining U.S. High-Frequency Trading”, New York Times

Lowenstein, Roger (2008), “Buffett: The Making of an American Capitalist”, Random House

Fama, Eugene (1963), "Mandelbrot and the stable Paretian hypothesis”, Journal of Business 36, 420–29

Fama, Eugene (1965a), “The behavior of stock market prices”, Journal of Business 38, 34–105

Fama, Eugene (1965b), “Random walks in stock market prices”, Financial Analysts Journal 21, 55–9

Fama, Eugene (1970), “Efficient capital markets: a review of theory and empirical work”, Journal of Finance 25, 383–417

Morde, Vishal (2019), “XGBoost Algorithm: Long May She Reign!”, https://towardsdatascience.com/https-medium-com-vishalmorde-xgboost-algorithm-long-she-may-rein-edd9f99be63d

Appendix

Tickers of Selected Stocks				
ABEV3	CMIG4	ESTC3	MRVE3	TAMM4
AEDU3	CPFE3	EVEN3	MULT3	TCSL4
ARCZ6	CPLE6	FIBR3	NATU3	TIMP3
B3SA3	CRUZ3	FLRY3	OGXP3	TMCP4
BISA3	CSAN3	GFSA3	OIBR3	TNLP3
BRFS3	CSNA3	GGBR4	OIBR4	TNLP4
BRKM5	CTAX4	GOAU4	PCAR4	TRPL4
BRML3	CTIP3	GOLL4	PDGR3	UGPA3
BRPR3	CYRE3	HGTX3	PETR3	USIM3
BRTP3	DASA3	HYPE3	PETR4	USIM5
BRTP4	DTEX3	IGTA3	POMO4	VALE3
BTOW3	EBTP4	JBSS3	QUAL3	VALE5
CCPR3	ECOR3	KLBN4	RADL3	VCPA4
CCRO3	EGIE3	KROT3	RDCD3	VIVT4
CESP5	ELET3	LAME4	RENT3	VVAR3
CESP6	ELET6	LIGT3	RLOG3	WEGE3
CGAS5	ELPL4	LIQO3	RSID3	
CIEL3	EMBR3	LREN3	SMLS3	
CLSC4	ENBR3	MGLU3	SUZB5	
CMIG3	EQTL3	MRFG3	TAEE11	

Table 1. Tickers of Stocks in the Experiment

Beg. Date	End Date	Buy and Hold	Linear Reg.	Lasso	Rand. Forest	XGBoost	SVM
Jan-08	Jan-09	4.8%	26.2%	9.5%	23.8%	14.3%	21.4%
Jan-09	Jan-10	30.2%	11.3%	7.5%	11.3%	18.9%	20.8%
Jan-10	Jan-11	4.8%	14.5%	12.9%	35.5%	19.4%	12.9%
Jan-11	Jan-12	8.1%	20.3%	14.9%	21.6%	23.0%	12.2%
Jan-12	Jan-13	29.3%	14.6%	11.0%	15.9%	17.1%	12.2%
Jan-13	Jan-14	8.8%	16.3%	11.3%	16.3%	23.8%	23.8%
Jan-14	Jan-15	13.8%	15.0%	10.0%	22.5%	17.5%	21.3%
Jan-15	Jan-16	3.8%	18.8%	12.5%	31.3%	13.8%	20.0%
Jan-16	Jan-17	51.2%	12.2%	4.9%	11.0%	13.4%	7.3%
Jan-17	Jan-18	37.0%	12.3%	7.4%	13.6%	11.1%	18.5%
Jan-18	Jan-19	24.7%	15.6%	9.1%	18.2%	16.9%	15.6%

Table 2. Percentage of Stocks With Best Performance for Each Strategy for a Given Period

Ticker	Linear Regression	Lasso	Random Forest	XGBoost	SVM
ABEV3	98	96	236	293	188
ARCZ6	25	17	33	67	35
BRKM5	106	86	192	298	164
BRTP3	27	27	74	94	44
BRTP4	31	25	52	82	59
CESP5	21	19	64	62	80
CGAS5	89	97	125	197	111
CLSC4	101	95	174	224	181
CMIG3	164	160	201	285	201
CMIG4	122	130	227	311	205
CPLE6	153	147	231	352	208
CRUZ3	73	47	59	115	63
CSNA3	64	66	140	220	138
EBTP4	91	85	108	92	85
EGIE3	126	144	204	368	227
ELET3	78	80	176	270	130
ELET6	104	104	182	232	138
EMBR3	117	123	263	349	197
GGBR4	128	130	245	312	184
KLBN4	89	92	248	320	178
OIBR4	110	114	192	286	180
PETR3	78	74	204	214	168
PETR4	86	74	178	242	162
TCSL4	43	51	73	99	51
TIMP3	113	119	231	353	239
TMCP4	66	56	34	84	64
TNLP3	50	54	108	107	88
TNLP4	32	44	96	158	54
TRPL4	98	120	231	271	187
USIM5	85	71	232	301	190
VALE3	107	109	157	281	155
VALE5	64	74	127	201	131
VCPA4	21	16	47	43	41
VIVT4	163	164	213	209	187
CTAX4	64	64	88	114	96
GOAU4	92	102	204	261	174
LIQO3	40	36	52	144	50
BRFS3	110	104	228	316	198
CCRO3	79	77	256	392	198
CESP6	112	113	163	201	127
ELPL4	87	75	146	204	113

LIGT3	126	120	164	250	134
NATU3	143	157	261	319	265
PCAR4	117	114	273	306	162
TAMM4	90	84	74	86	83
BTOW3	160	180	258	299	211
CCPR3	60	62	73	98	56
CPFE3	110	122	280	382	206
CSAN3	164	152	186	280	164
CYRE3	64	74	223	275	159
GFSA3	87	85	117	216	113
GOLL4	133	143	232	296	171
LAME4	116	116	190	230	166
LREN3	113	105	246	288	197
B3SA3	116	114	138	262	172
JBSS3	130	112	193	245	153
RDCD3	44	42	63	85	53
RSID3	84	76	142	206	110
USIM3	135	137	215	377	195
DTEX3	124	142	209	197	150
FIBR3	93	107	129	169	99
OIBR3	72	88	190	263	174
BISA3	66	51	61	99	61
CIEL3	72	76	120	196	116
MRFG3	84	92	168	297	170
MRVE3	93	136	215	250	134
OGXP3	77	73	57	135	98
PDGR3	58	56	135	201	133
BRML3	75	79	179	209	118
HGTX3	127	131	159	211	154
HYPE3	107	95	143	167	131
UGPA3	189	157	175	236	189
CTIP3	48	48	102	170	157
DASA3	105	101	115	215	133
RENT3	80	90	258	400	214
SUZB5	76	74	202	208	153
AEDU3	34	50	38	54	37
BRPR3	109	109	150	161	100
ENBR3	158	160	236	356	205
KROT3	92	92	60	108	90
ECOR3	108	102	132	130	131
ESTC3	136	142	137	205	137
EVEN3	77	75	185	289	173
POMO4	136	130	213	329	159
QUAL3	34	40	42	92	32

RLOG3	18	22	29	49	22
EQTL3	78	84	184	245	120
MULT3	96	90	137	143	133
RADL3	49	81	152	195	114
SMLS3	80	78	98	130	58
WEGE3	142	138	274	402	176
FLRY3	104	112	136	231	134
IGTA3	116	116	154	222	126
MGLU3	29	27	49	81	29
TAEE11	78	78	109	189	147
VVAR3	108	114	94	132	114
Mean	92	93	157	218	136

Table 3. Number of trades per stock for each strategy