Udacity: Machine Learning Nanodegree

Capstone Project

Author: Lucas Oliveira Souza

1. Introduction

Choosing to buy assets based on a predictive upward trend is a discipline that dates back to the early days of capitalism. In 1637, in Holland, there was a non expected surge in tulips prices due to speculation, that caused a tulip bubble which today is widely regarded as the first ``stock market crash''. The bubble was caused by investors buying tulips with the expectation that its price would rise in the long term.

These events are more frequently in the last 30 years due to the introduction of computing intelligence in the stock trading game. Technical analysts have tried to predict future trends solely based on price and volume historical data, using esoterica methodologies of charts interpretation that even includes Leonardo da Vinci's infamous golden ratio. Fundamental analysts have manipulated millions of spreadsheets to come upon the perfect expected price of the company and based on it predict whether its price would rise or fall.

In 2013, Eugene Fama won the Nobel Prize for Efficient Market Hypothesis, which states that all relevant information regarding a company is already contained in its stock price, having no possibility therefore for prediction of stock price trends. That assumption has been challenged for many decades, but it has gained new colors with the rise of data science and machine learning as research field.

The introduction of artificial intelligence (AI) methodologies in investment portfolio management and the rise of machine learning have drawn a lot of interest, which promises big rewards for those who can crack the challenge.

Related Work

In [6] we find a multi-agent systems for stock price prediction. The architecture presented by the authors is a four-layered architecture, in which the first layer collects the data from various sources, both qualitative and quantitative, the second layer preprocess the data, the third layer predict the prices using a bat neural network, and a fourth layer that generates scenarios based on the prediction generated and present report to the decision makers. Only the middle two layers were implemented, with the remaining cited as future work.

Similar approaches are found in previous works [10,11]. While [11] is focused on on predicting stock prices trend, [10] is concerned with investment decisions that optimize asset allocation. In [10] we find multi-agent system that uses reinforcement learning for the trading agent, along with traditional strategies for predicting stock price. Author argues that given enough time and data a non-parametric machine learning method can discover complex non-linear relationship that makes the market predictable, and invalidate the Efficient Market Hypothesis. The system is based on previous works on applying Q-learning, recurrent reinforcement learning and adaptive reinforcement learning to stock trading [5,4,13].

The work [3], presents a multi-agent system, composed of coaches and advisors, which cooperate to optimize a portfolio based on an analysis of assets risk. In this work the CAPM theory based on market equilibrium concept defined by [18] is applied by autonomous agents to optimize allocation of assets.

Significant work have been done on the field of agents operating in a simulated stock market environment. In [19], the author implements three different agents competing at Penn Exchange Simulator (PXS) [9], a reinforcement learning, a market making, and a trend following agent. A more recent work on intelligent agents in simulated market model [1] also compares three different strategies implemented by independent autonomous agents, being two fundamental analysis strategies, an averagely informed trader and an insider, and one technical analysis (chartist) strategy.

There is a vast literature on the topic of using non-linear regressors to predict stock prices. There seems to be a concentration in neural networks, considered to be the state of the art in non-linear regressors, as discussed in [14,16,2]. Neural networks is a popular algorithm since the 70s, and since them several algorithms have been designed with the same basis but slight variations, such as convolutional neural networks and deep belief networks.

The designed systems have been applied to a variety of markets, from Romania [14] to Germany [6] and Brazil [8]. This work focuses exclusively on assets traded in Brazilian stock market Bovespa, although there are no limitations to expand to other markets and other types of assets rather than company stocks.

2. Define the Problem

Problem: Predicting stock price trend

The common approach when dealing with stock price trend prediction is using non linear regression models to predict future stock prices. Regression is a complex problem, with unsatisfactory results when dealing with stock prices. Predicting the price 3 days ahead (or 3 minutes ahead, if looking at intraday stock price) is a much harder problem that predicting 1 day ahead.

To avoid this pitfall, we turn predicting stock price trend into a classification problem, contrary to the common standard of using non-linear regressors. The approach is based on a predefined strategy, which the agent

holds as fixed. The strategy will tell which type of asset to buy, when, and when to short the position.

The most common strategy that can be devised is is to buy a stock, hold it for n days, and sell it if reaches an expected valuation. Here onwards it will be referred "swing trading" strategy, in reference to the term "swing trading" which is used to define trades with a span of a few days and which aims to take advantage of market swings due to speculation.

With a strategy defined, the we evaluate the entire training set, and for each observation creates a binary label, which indicates whether the strategy would have worked or not if applied on that day. The success/failure label will be the label learned by the classifier.

The goal of the classification problem, then, is given several variables of the current day, define whether applying this strategy will be successful or not. There is a inherent trade-off, as we are disregarding important extra information (the regression would tell the price, which contains more actionable information) in exchange for a simpler model which can be trained to a higher precision.

The 5 most relevant stocks were selected, most relevant being the 5 stocks with higher share in the Ibovespa index: ABEV3, BBDC4, ITUB4, PETR4, VALE5.

Evaluation

The metric used for evaluation of stock price trend prediction is precision. Precision measures how many of the items classified as positive are actually positive (true positives/positives). Recall is less relevant for the problem, as it is not mandatory the model capture all buy signals - since many assets are being evaluated simultaneously, it is more important to know a buy signal is actually true rather than capturing all buy signals for an asset.

Benchmark

In order for the strategy to be profitable, it is enough if precision is greater than 50% if you the expected profit and acceptable loss are the same. For example, if you define a strategy where you buy an asset and will sell if it reaches +10% valuation or -10% devaluation. In this scenario, if you get a valuation 55% of the time, and a devaluation 45% of the time, you end up with a positive net.

Cost of opportunity should also be taken into consideration in a detailed analysis. It is not enough not to loose money, but the return should be equal or greater than the return you would get from a risk free strategy. As the valuation of a risk free strategy is low (less than 0.5%) for short term trades (within 10 days), we can disconsider for benchmarking purposes.

A higher level approach to benchmarking is to use the buy signals identified by the classifier to simulate actual trades and evaluate its return against a market index. In this project, we use IBOVESPA as the benchmark

index, which is a composition of 100 stocks more commonly traded in BOVESPA, ranked by trade volume.

3. Analyze the problem

Data

The data collected is for the Brazilian stock market (Bovespa). The training period is from 2012 to 2014, and the operation/test period from Jan/2015 to Oct/2016. The market data was collected from quandl.com, which has an easy to operate API. It requires login and password but it takes less than a minute to create a free account.

Data gathering is done using only public available data, to ensure repeatability. There are three types of data relevant to the financial analysis of investment assets, considered for the project:

- Market data: includes general market data, such as interest rate, sector data, such as price of oil gallon, and specific data related to a specific company.
- Stock data: buy, sell, open and low prices, and volume.
- Text data: information that can be mined from text and reflect the behavior of the investor. Includes contextual information such as twitters, headlines, news, and specific information such as summary financial reports.

Of these 3 groups, the following data were collected and deemed relevant, based on previous feature analysis and expert knowledge:

Market Features, used by fundamental analysis, downloaded from Banco Central do Brasil:

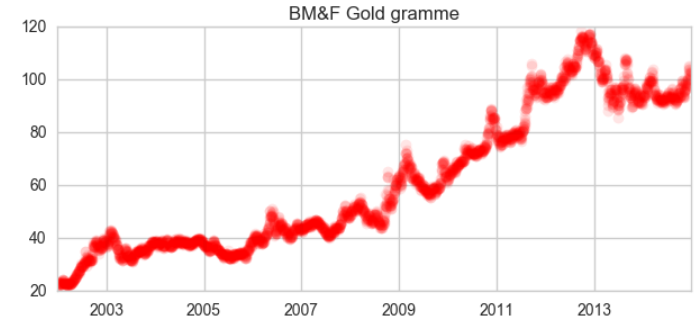
- Selic
- Exchange Rate USD Sell
- BM&F Gold gramme
- Bovespa total volume
- International Reserves
- Bovespa index
- Foreign exchange operations balance
- Nasdaq index
- Dow Jones index

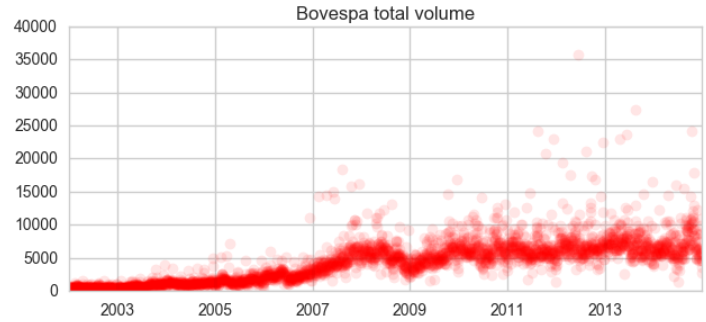
Technical Features, used by technical analysis, calculated

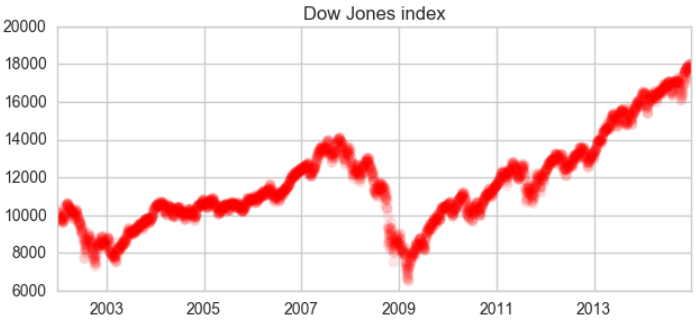
- Moving Averages < 10,20,... 60 > days
- Bollinger Bands < 10,20,... 60 > days

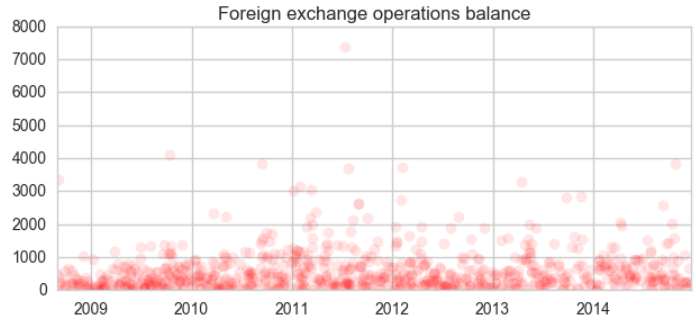
Analysis

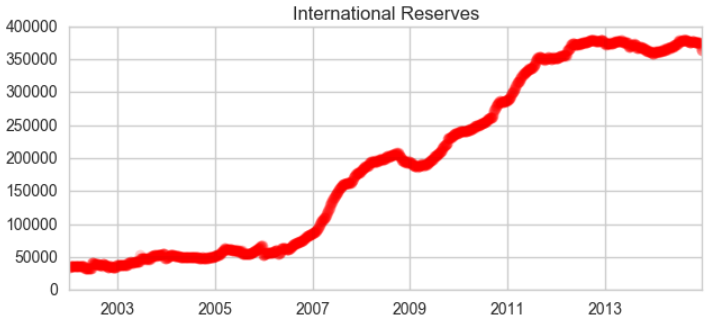
The relevant features analyzed are mainly time series variables, tied to market fluctuation, so no significant outliers are expected. That is confirmed in a short analysis done in the market data, that shows concentrated dots in time series data. In Foreign exchange operations balance and Bovespa Volume, which are not time series variables, a few outliers are identified.

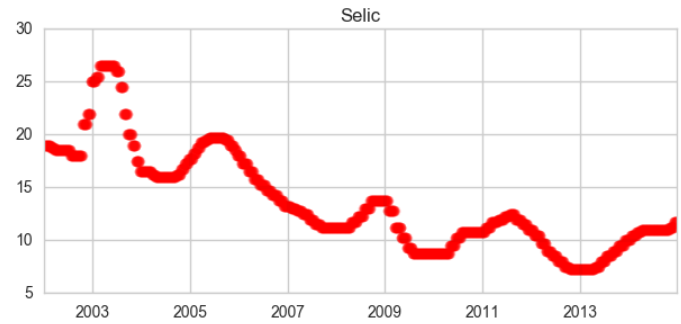












The relevance of each feature in the prediction task will change according to the asset analyzed. Companies

who are tied to dollar are susceptible to exchange rate variations and probably US market indexes such as Dow Jones, while local small caps may be more impacted by Bovespa trading, since their price is highly dependent on volume traded per day.

Besides market data, bollinger bands and moving averages for a range of periods were also added to the data. They are also time series variables tied with the asset price fluctuation, as shown in the examples below:

Transformation

The classifier model infers future stock prices indirectly, by predicting whether or not a given strategy will work or will not work at a given point in time. In order to predict the probability of a positive outcome in the future, the best approach is to look at the past trend and analyze the patterns it holds.

For all the variables in the dataset, we define the number of days in the past which will be considered relevant to predict the future trend, and add the variables of this period to the feature space. So if 10 variables are defined, for a period of 60 days, the total number of variables for each observation is 600.

The relevant pattern is the trend and not the variable itself. So for all features, we take only the ration between the value at N and the value at N-M, being N the day for the current observation and M the number of days

between a past observation and the current. With this transformation we can capture the changes instead of the raw values.

3. Implement solution

Concepts

k-Nearest Neighbors

kNN is a non-parametric model, and hence it inherits the weakness and strenghts of non-parametric classifiers. It does not learn the parameter, rather use the full data to classify. The main weakness is that requires more memory space (N) and search time is long. The strength is that it does not require learning, new observations could be used for model without requiring to re-train the classifier. kNN is widely used in the finance market for stock price prediction.

Gradient Boosting

Boosting works by combining the output of multiple weak learners. A weak learner is defined to be any model that can predict the label better than random chance. There are several variations of boosting, but generally it involves taking a subset of the dataset (subset of samples and/or of features), training a weak learner, select another subset of the data which were badly classified, training another weak learner, and iterate away.

There are several boosting algorithms. Gradient boosting works by building the model "in a stage-wise fashion like other boosting methods do, with several weak learners, and it generalizes them by allowing optimization of an arbitrary differentiable loss function." It is "based on the observation by Leo Breiman (which published the first paper on boosting) that boosting can be interpreted as an optimization algorithm on a suitable cost function" [20].

The difference between AdaBoost and Gradient Boosting is on how to identify shortcomings (subset of the data which were badly classified). AdaBoost identifies it by high-weight data points, while in Gradient Boosting shortcoming are identified by gradients. cross validated An advantage is good performance without overfitting. A disadvantage is having to train several weak learners instead of one strong learner, so it takes a long time to learn compared to other single learner algorithms.

Grid Search

A grid search techinique of trying all possible combinations in a grid. In machine learning, it refers to a technique of attempting to train the model with different sets of parameters, to find the optimal choice of parameters for the algorithm, a practice also referred to as parameter tuning. For each set of parameters, the

model is validated by using a cross-validation strategy. The grid search returns the best choice of parameters for the model.

Cross-Validation

Ideally, you should separate a part of the dataset only to test in the final algorithm, and split the remaining into a learning set and validation set. A validation set is important because there is a risk in overfitting when you use the the test data to choose the most appropriate model or perform hyperparameter tuning, since there is a leakage of test data information into the model.

But training on a smaller set of the data can reduce the algorithm's performance. To avoid discarding relevant observations, and stay clear from overfitting, there are several strategies of cross-validation which dismiss the need for a validation set.

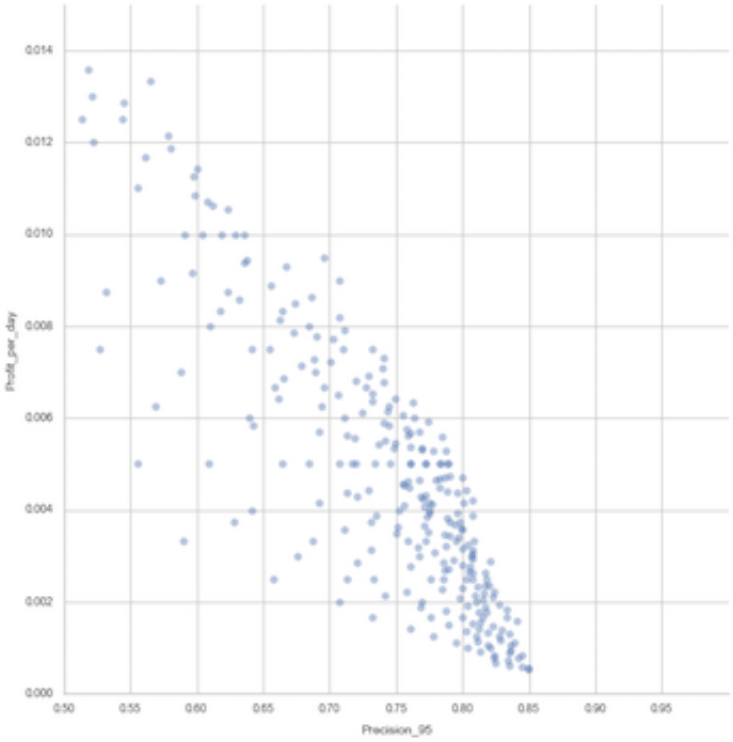
In cross-validation, the training dataset is split into k folds. The prediction function is learned using k - 1 folds, and the fold left out is used for test. This is repeated multiple times, for each fold, and the performance measure reported is the average of the values computed in the loop.

The exact strategy will vary depending on the cross-validation method used. The method described above is the k-fold, which is a basic strategy of cross-validation. Other strategies including using a stratified approach, that preserves the distribution of label classes in each fold, and is ideal for highly unbalanced data, and shuffling the elements before breaking down into folds.

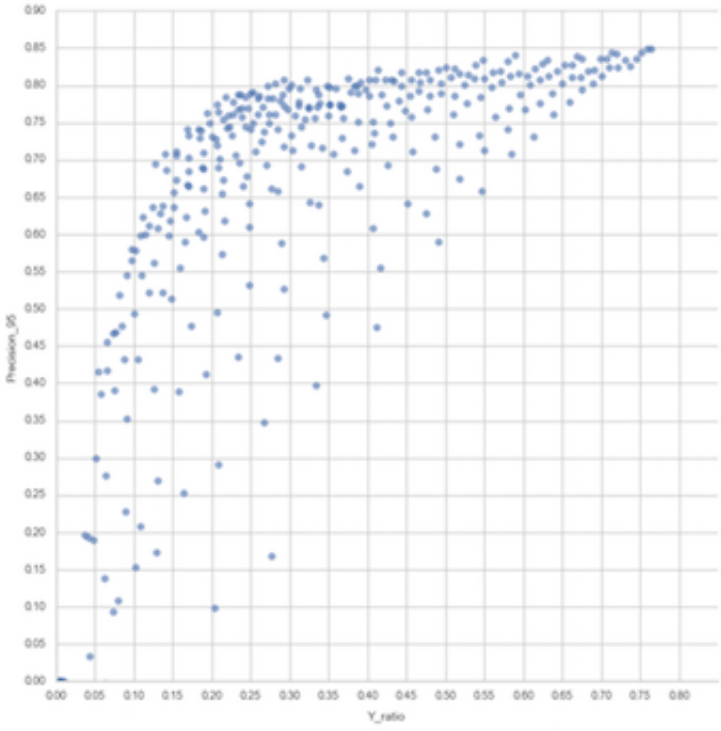
Implementation

One classifier has been trained for each asset. The parameters for the swing trading strategy, the duration of the trade in number of days and expected profit, were optimized through grid search, by maximizing the highest profit per day and classifier precision.

The scatterplot below show how precision varies as we try different rations of profit per day:



The scatter plot below show how yratio varies as we try different rations of profit per day. The yratio is the percentage of positive labels over all labels - a 30% yration means the strategy will be have a positive outcome in 3 out of 10 days.



The best results were achieved by maximizing net results. The optimal span for the trade was 10 days with a

9% expected profit:

```
Span              10.000000
Profit             0.090000
Precision_mean     0.808170
Precision_std      0.050420
Y_ratio            0.139487
Precision_95       0.707331
Profit_per_day     0.009000
Net_profit         0.006366
Net_loss          -0.002634
Net_result         0.003732
Name: 178, dtype: float64
```

Classification

To reduce the number of variables, we first convert the features into principal components which represent the major variability in the features. That is a common practice in machine learning pipeline which is essential to avoid the curse of dimensionality, given the limited dataset when looking at daily operations. From 2002 to 2014, for example, there are only approximately 3276 trading days.

The classifier used for each asset was k-Nearest Neighbors, which achieved high precision score with fast training times. Several other classifiers were tested, with Gradient Boosting achieving the best performance but worst training times.

Tuning

Tuning the parameters is required to optimize each classification model. Another opportunity of optimization applicable to this problem is tuning the parameters of the strategy.

In the swing trade strategy defined, we optimize the number of days to hold the position and the expected profit. By attempting several different parameters it aims to reach the strategy which is easier to predict, meaning, in which the classifier achieves higher precision.

The parameters for the strategy and the machine learning model plus preprocessing were tuned with genetic algorithms at first, and later with a grid search like approach.

Hardware

The infrastructure used was a local Mac 2014 notebook, with 8 cores and 16gb ram. GPU was not used. Future implementations will be based on Amazon Elastic Search for better scalabilit

4. Evaluate results

Evaluation Metrics

There are a myriad of evaluation metrics for classifiers, and the optimal choice of evaluation metric will vary with the problem at hand.

In this work, we attempt to predict whether a strategy will be successful or not. If it predicts a false positive, it can lead to a trade which will result in loss. On the other hand, if we fail to predict a positive label (a false negative), it will only incur in the cost of opportunity of losing a potential trade.

Several assets are evaluated, so a false negative is a lesser problem than a false positive, since we can receive concurrent positive recommendations from different assets.

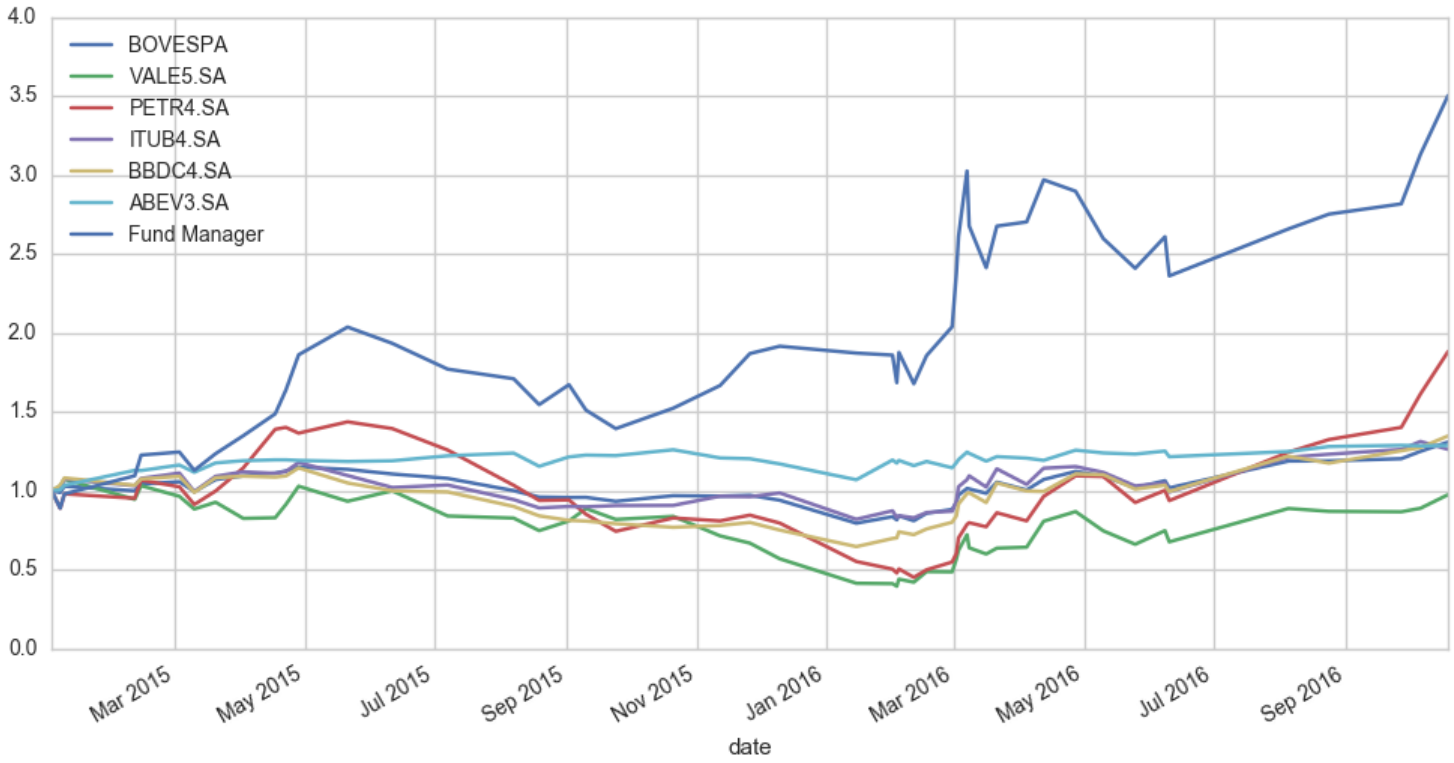
Precision measures the ratio of true positives, divided by all the observations classified as positive, and hence is the metric chosen to evaluate the classifiers.

The trained classifiers achieves precision as high as 80%. Considering the margin of error of 2 standard deviations (approximately 95% confidence interval), the precision is still higher than 60%, which is significantly above the 50% benchmark defined in the problem description.

```
[3105  228]
Precision for ABEV3: 0.87 (+/- 0.14)
[1425  233]
Precision for BBDC4: 0.73 (+/- 0.21)
[2757  426]
Precision for ITUB4: 0.74 (+/- 0.16)
[2661  503]
Precision for PETR4: 0.86 (+/- 0.09)
[2498  460]
Precision for VALE5: 0.80 (+/- 0.11)
```

Backesting

To confirm if the strategy yields positive outcomes, we simulated an actual fund manager trading based on the outputs given by the classifier. The application period is from Jan/2015 to Oct/2016.



The return achieved in the period is 8x higher than the return achieved from IBOVESPA index, or any of the traded stocks treated separately, as shown below. The optimized portfolio is able to properly capture the peaks and valleys of each asset, hence allowing the simulated fund manager to choose the optimal time to buy each asset.

IBOVESPA, defined as benchmark for capital return in the problem definition, is the most important index for the Brazilian Stock Market, and is equivalent to a simulated portfolio composed of the most important 100 stocks traded in Bovespa, weighted by their total trading volume.

5. Conclusion

Considering the described scenario, this work presents an innovative approach to predict stock prices trend and choose optimal investment strategies. Backtesting results, using an environment simulated with actual stock market data of Bovespa from January/2015 to October/2016, shows an investment strategy based on the buy and sell signals predicted by the model net results 8x times higher than the baseline market index Ibovespa.

The model are able to predict success or failure of a strategy with over 80% accuracy for certain assets, successfully replacing traditional approaches that use regression to predict stock price trends.

The classification approach to the stock price prediction, instead of regression yields considerable results and it is worth notation for future works.

Future work

Classification

The results are preliminary and show potential. Improvements can be made by: * Add more assets to the portfolio. As of now, only 5 assets have been considered. * Add different strategies other than swing trade. Including different classes of assets, such as derivatives, can greatly increase the range of available strategies * Use a wider range of classifiers competing for optimal performance. * Optimize infrastructure to run distributed in cloud. Gradient Boosting has shown better results, but with worst performance. Improvements on the infrastructure would allow more complex models which can have a better fit to the data

Reinforcement learning

Portfolio optimization can optimize its policy through a model based reinforcement learning strategy, such as Q-learning. In the Q-Learning parameter space, the state is given by its current capital and risk appetite, and the actions available are which type of strategy to apply and how much capital to allocate.

The portfolio optimization model can be trained on test data not used by the data analysts for classification. Running until convergence can overfit the model to the test data, which can be avoided by setting a high decay exploration rate and settling with an suboptimal maximum.

7. References

[1] D. Bloembergen, D. Hennes, S. Parsons, and K. Tuyls. Survival of the chartist: An evolutionary agent-based analysis of stock market trading. In *Proceedings of the 2015 International Conference on Autonomous Agentsand Multiagent Systems*, AAMAS ’15, pages 1699–1700, Richland, SC, 2015. International Foundation for Autonomous Agents and Multiagent Systems.

[2] T. Chenoweth, Z. Obradovic, and S. S. Lee. Embedding technical analysis into neural networkbased trading systems. *Applied Artificial Intelligence*, 10(6):523–542, 1996.

[3] P. A. L. de Castro and J. S. Sichman. Automated asset management based on partially cooperative agents for a world of risks. *Applied Intelligence*, 38(2):210–225, 2013.

[4] M. A. Dempster and V. Leemans. An automated fxtrading system using adaptive reinforcement learning. *Expert Systems with Applications*, 30(3):543–552, 2006.

[5] X. Gao and L. Chan. An algorithm for trading andportfolio management using q-learning and sharperatio maximization. In *Proceedings of the internationalconference on neural information processing*, pages 832–837. Citeseer, 2000.

[6] R. Hafezi, J. Shahrabi, and E. Hadavandi. A bat-neural network multi-agent system (bnnmas) for stock price prediction: Case study of dax stock price. *Applied Soft Computing*, 29:196–210, 2015.

[7] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing In Science & Engineering*, 9(3):90–95,2007.

[8] E. Jabbur, E. Silva, D. Castilho, A. Pereira, and H. Brandão. Design and evaluation of automaticagents for stock market intraday trading. In *Proceedings of the 2014 IEEE/WIC/ACM International Joint Conferences on Web Intelligence(WI) and Intelligent Agent Technologies(IAT)-Volume 03*, pages 396–403. IEEE Computer Society, 2014.

[9] M. Kearns and L. Ortiz. The penn-lehman automated trading project. *IEEE Intelligent Systems*,18(6):22–31, Nov 2003.

[10] J. W. Lee, J. Park, O. Jangmin, J. Lee, and E. Hong. A multiagent approach to q-learning for daily stock trading. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 37(6):864–877, 2007.

[11] Y. Luo, K. Liu, and D. N. Davis. A multi-agent decision support system for stock trading. *IEEE network*, 16(1):20–27, 2002.

[12] W. McKinney. Data structures for statistical computing in python. In S. van der Walt and J. Millman, editors, *Proceedings of the 9th Python in Science Conference*, pages 51 – 56, 2010.

[13] J. Moody and M. Saffell. Learning to trade via directreinforcement. *IEEE transactions on neural Networks* ,12(4):875–889, 2001.

[14] M. D. Nemes and A. Butoi. Data mining on romanian stock market using neural networks for price prediction. *Informatica Economica*, 17(3):125, 2013.

[15] F. Pedregosa et al. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research* ,12:2825–2830, 2011.

[16] F. Pérez and B. E. Granger. Ipython: a system for interactive scientific computing. *Computing in Science & Engineering*, 9(3):21–29, 2007.

[17] S. Rahimi, R. Tatikunta, R. Ahmad, and B. Gupta. A multi-agent framework for stock trading. *International Journal of Intelligent Information and Database Systems*, 3(2):203–227, 2009.

[18] W. F. Sharpe. Capital asset prices: A theory of market equilibrium under conditions of risk. *The journal of finance*, 19(3):425–442, 1964

[19] A. A. Sherstov and P. Stone. Three Automated Stock-Trading Agents: A Comparative Study, pages173–187. *Springer Berlin Heidelberg*, Berlin, Heidelberg, 2005.

[20] Gradient boosting. *In Wikipedia*, The Free Encyclopedia. Retrieved December 8, 2016, from https://en.wikipedia.org/w/index.php?title=Gradient_boosting&oldid=753674492