# Evaluating Nonlinear Classification Methods for Valuation Changes in Financial Time-Series Data and Portfolio Optimization

Atticus Rex
GTID: 903913962

Instructor:
Nisha Chandramoorthy, Ph.D.

**Abstract**

This paper investigates Adaptive Boosting (AdaBoost), Kernel Ridge Regression, and Kernel Support Vector Machines and their ability to predict valuation changes in Exon Mobil stock augmented with exogenous regressors of oil price per barrel, S&P 500 data, google search data, and electric car demand. Further, this report also investigates applying the AdaBoost algorithm to high-frequency trading, optimized and diversified by applying Markowitz Portfolio theory to form a convex minimization problem. The report found that AdaBoost outperformed KRR, and KSVM, especially when the number of rounds of boosting was increased. Further, the portfolio optimization algorithm converged on six companies to apply the algorithm to and outputted the optimal allocation of capital to each strategy. This has strong implications in the field of quantitative finance and was demonstrated to be profitable in this paper.

A report presented as the term project for
**CSE 6740** - Computational Data Analytics

**Georgia Institute of Technology**
College of Computing, Atlanta, GA
December 19, 2023

# Introduction

The oil industry is a multi-trillion dollar sector that significantly affects the operations of many firms worldwide. Approximately two thirds (66.6%) of world oil usage is toward transportation [1]. With the advent of electric vehicles, the demand for fossil-fuel based transportation is evolving. The valuations of oil giants such as Exxon, Chevron, and Shell Oil are highly reactive to this economic shift. Hence, using established statistical learning techniques to classify valuation moves is desirable to inform investors and provide further into some of the most important factors in industry.

This project will be a computationally heavy implementation and evaluation of AdaBoost with various weak learners, Regularized Logistic Regression and Kernel-based SVM classification on classifying Oil Company Valuation (Such as Exxon, Chevron, etc.) moves regressed on various feature sets such as Cushing, OK WTI oil Prices, S&P 500 Index Measurements, and electric vehicle demand.

Further, this paper will investigate the use of high-frequency portfolio optimization using Markowitz Portfolio Theory, which provides a conceptual model for how to allocate capital based on different strategies.

# Current Methods

A large literature exists in financial time-series regression, with the most state-of-the-art methods centering around deep learning and recurrent deep learning methods such as Deep Neural Networks (DNNs or ANNs) and Long-Short-Term Memory Neural Networks (LSTMs) [2], [3]. Many algorithms seek to use Data Mining algorithms to extract concise, low-dimensional information from high-dimensional data [4]. Clustering Algorithms such as FCM Clustering and Classification using ANNs have been widely applied to various time-series data within the market and have shown varying degrees of accuracy [5]. Further, many have successfully applied theories of Markowitz Portfolio Optimization which maximizes expected returns at a desired risk tolerance to varying financial situations [6], [7]. There are a handful of concerns I had when reviewing the literature on this work. These are the following:

- **Overparametrization of Neural Networks**: Deep learning scales extremely well with large amounts of data, and indeed, financial markets are a real-world source of large amounts of data. However, there were approximately 3,250 trading days since the year 2010. I argue that this number is certainly high enough to capture general relationships in the data, but not sufficiently high to justify the training of a neural network with thousands of parameters. The functional complexity of these ANNs is so high that overfitting is constantly a worry when dealing with a sample of this order of magnitude. We see in other disciplines such as Epidemiology that time-series predictions often fall-short when the model is overparameterized. In other words, there

may be multiple combinations of parameters that match the training data well, but only one will generalize to unseen data. The MDL Infer framework attempts to regularize machine learning models with the "Minimum Descriptive Length" of the model which tends to work well (Deep Learning networks tend to be some of the most penalized models within this framework) [8].

- **Lack of Model Uncertainty**: Oftentimes, the models trained in the kind of classification algorithms decsribed previously are attempting to classify valuation moves into one of two categories: up or down. While this fits in nicely with established binary classification theory, it also disables the models from expressing any sort of uncertainty. In a financial model, I would much rather my model spit out "I don't know" rather than "Put all your money into this security" just because it wasn't trained in this manner. An important aspect to keep in mind here is that financial markets have a fundamentally random component to them; they are the remnant of the behavior of millions of human and nonhuman agents all working in real-time to generate a particular valuation. No model, human or not, armed with solely public information should theoretically ever be able to say with certainty about the direction of a particular company's valuation. Hence, it is useful to work on a more probabilistic, distribution-based method rather than a simple "yes/no" scheme; we seek to increase the mean returns of a strategy by selectively picking what to invest in.
- **Limitations of Markowitz Optimization**: Markowitz Portfolio Optimization has primarily been used as a conceptual model to describe an **efficient frontier** when comparing how to divide money between two companies. Many firms have used it to weight the returns of long-term investments, but very few have combined it with machine learning to optimize short-term gains.

# Algorithms and Techniques

## AdaBoost

The AdaBoost algorithm developed by Freund and Shaipre gives an effective framework for classification problems that relies on an ensemble of "weak learners" that can classify the training data slightly better than random chance [9]. AdaBoost relies on a series of "rounds" of boosting. In each round, a new weak learner is instantiated and is trained to focus on the observations that the previous weak learner got wrong. An ensemble of all the weak learners is created and the entire model is an average of all the learners put together. As the number of rounds of boosting increases, the ability of the AdaBoost model to predict more complex relationships increases (or the functional complexity increases). Mathematically, with more rounds of boosting, the generalization risk also increases [10]. However, in empirical studies it appears that in many cases, models with more rounds of boosting actually perform *better* on test sets as shown in Mohri et al. [11].

## Kernel Ridge Regression

Kernel Ridge Regression (KRR) is a powerful regression technique used widely throughout statistical machine learning [12]. It goes beyond traditional ridge regression by introducing a kernel function, $k(x)$, which allows it to predict nonlinear relationships. The overarching concept is to map input data into a high-dimensional feature space, where a linear regression model is applied. KRR seeks to encounter a linear combination of these transformed features that minimizes the mean squared error between the predicted and target values while simultaneously penalizing the magnitude of the regression coefficients, which prevents overfitting. The kernel trick enables KRR to operate in this high-dimensional space without actually computing the feature mappings, making it very computationally efficient.

## Kernel Support Vector Machines

Kernel Support Vector Machines (KSVMs) are a powerful class of supervised machine learning algorithms used for classification and regression tasks [13]. At their core, SVMs seek to find an optimal hyperplane that best separates data points belonging to different classes while maximizing the margin between them. What distinguishes Kernel SVMs is their use of kernel functions, which allow them to operate effectively in high-dimensional and nonlinear feature spaces. These kernels transform the input data into a higher-dimensional space, making it possible to find a linear decision boundary that can accurately classify or predict new data points. Popular kernel functions include the radial basis function (RBF) kernel and polynomial kernel. Kernel SVMs excel when dealing with complex and nonlinear relationships within the data, making them valuable tools for various machine learning applications. The choice of kernel and tuning of hyperparameters are crucial in achieving optimal SVM performance.

## High-Frequency Markowitz Optimization

This algorithm I put together myself using the literature, so I'm going to go into considerably more detail. Consider we have $n$ strategies, $f_i$, each of which we assume to have normally distributed returns with mean $\mu_i$ and standard deviation $\sigma_i$. So the return on a given day, $f_i(k) \in \mathcal{N}(\mu_i, \sigma_i)$. $\alpha_i$ represents the proportion of the portfolio's total capital $P$, we allocate to strategy $f_i$. We want to maximize the expected returns of the entire portfolio which can be expressed as the following:

$$F(\alpha) = \sum_{i=1}^{n} \mu_i \alpha_i$$

More interestingly, the variance of the entire portfolio can be expressed as the following:

$$\text{Var}(F(\alpha)) = \alpha^T \Sigma \alpha$$

where $\Sigma$ is the covariance matrix of the returns of all strategies $f_i, i \in [n]$. Because $F(\alpha)$ is a linear function and linear functions are both concave and convex, we can express this problem as a convex minimization problem. I found better convergence when I used the exponential function to make the linear function more convex:

$$\min \left( \exp[-\sum_{i=1}^{n} \mu_i \alpha_i] \right)$$
$$\text{such that: } \alpha_i \geq 0$$
$$\sum_{i=1}^{n} \alpha_i = 1$$
$$\alpha^T \Sigma \alpha \leq \sigma_P^2$$

The first constraint is that all strategy weights must be positive, the second constraint states that they all must add up to one and the last constraint says that the variance of the portfolio must be less than some variance set by the user, $\sigma_p^2$, which indicates a desired risk-tolerance of the overall portfolio. This can be solved via the Lagrangian method:

$$\mathcal{L}(\alpha, \lambda, \gamma, \nu) = \left( \exp[-\sum_{i=1}^{n} \mu_i \alpha_i] \right) + \sum_{i=1}^{n} \alpha_i \lambda_i + \gamma \left( \sum_{i=1}^{n} \alpha_i - 1 \right) + \nu \left( \alpha^T \Sigma \alpha - \sigma_P^2 \right)$$

I derived the KKT conditions, set up the systems of equations of size $2n + 2$ and solved for the $2n + 2$ variables by computing the Jacobian and using Newton's method. However, I will be honest, using the "minimize" function from the scipy library converged way faster than my Newton's Method implementation in python.

# Classification Results

To evaluate the classification models, I trained each of the models on Exxon Mobil with a looking forward period of 7 days. In other words, the model was trying to classify the stock price change 7 days in the future. I implemented exogenous regressors which were oil price per barrel, S&P 500 prices, and electric car valuations.

## Validation Strategy

I decided to make this validation as close to real-world application as possible so I used an autoregressive validation strategy that iterated through the time-series step by step. Figure 1 is a visualization of the results of the AdaBoost Classification results attempting to predict Exxon Stock 7 days in the future.

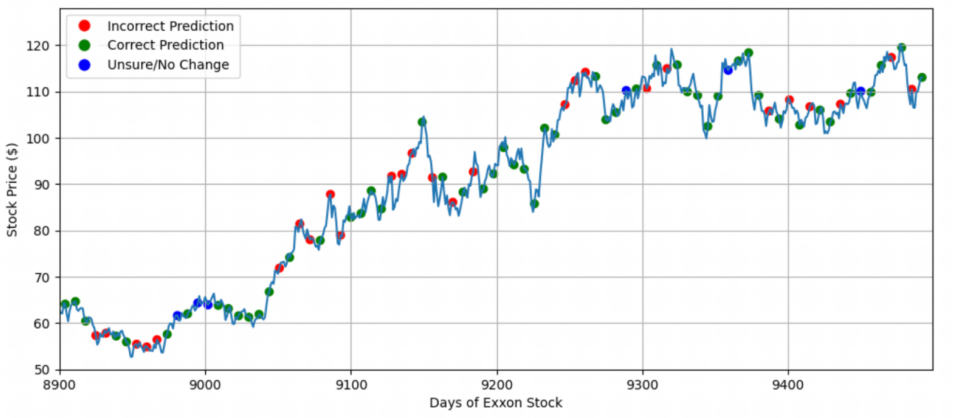


Figure 1: Detail View of AdaBoost Autoregression Results on Exxon Stock.

## Algorithm Validation Accuracies

The following table depicts the results of applying each algorithm to this validation strategy. Each of the accuracies outlined in the table below represent how the algorithm performed on unseen data seven days in the future. The second row is the percentage of the algorithm's predictions that were either correct or a 0, which examines the percentage of predictions which were breaking even or profitable.

**Table 1.** Accuracies and Breakeven Percentages of Classification Algorithms

| | AdaBoost (100 Rounds) | AdaBoost (1000 Rounds) | **AdaBoost (5000 Rounds)** | KRR | SVM |
|---|---|---|---|---|---|
| Accuracy (%) | 58.98 | 65.32 | **69.20** | 57.94 | 59.89 |
| Breakeven (%) | 67.38 | 72.03 | **79.16** | 63.48 | 62.16 |

As shown in the table, the more rounds of AdaBoost, the more accurately the algorithm performed on validation sets. This conflits with the mathematical assertion that the higher the functional complexity, the higher the generalization risk, but it does coincide with what Mohri observed when applying AdaBoost to natural language processing datasets [CITE]. Figure 2 shows how the distribution of returns experienced from the most successful AdaBoost

is shifted into the profitable region when compared to the distribution of the changes in the stock price.

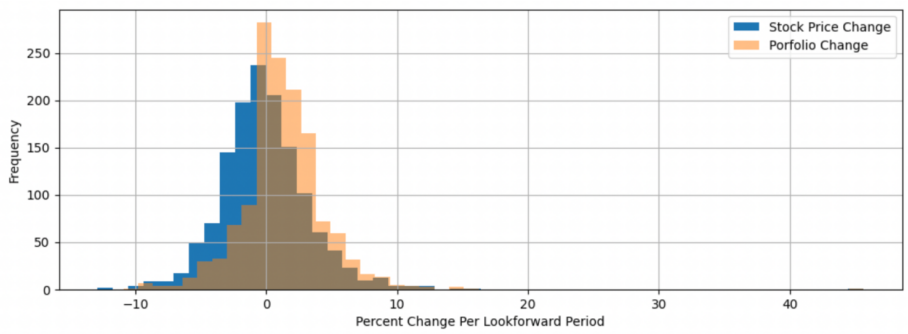


Figure 2: Distribution of Exxon Stock Price Changes (Blue) vs. The Outcome of Buying/Selling according to the AdaBoost strategy (Orange)

## Portfolio Optimization Results

The portfolio optimization section of this investigation uses the AdaBoost algorithm because it had the highest testing accuracy. The goal here is to use this strategy on many different companies to create a sort of "superportfolio" based on empirical Markowitz Portfolio Theory. The following list outlines the specifications for this evaluation:

- **Number of Companies Evaluated**: This investigation combined data from 492 of the S&P 500 companies for the last 10 years. It selected companies with at least five years of trading data available and a median trading volume of 100,000 shares per day.
- **Validation Strategy**: Used AdaBoost strategy with 100 rounds of boosting to approximate mean and covariance of returns of applying this strategy to each company.
- **Lookback and Lookforward**: The algorithm had 4 previous days of stock data as features and attempted to predict just 1 day in the future.
- **Real-Life Parameters**: I introduced a spread of \$0.05 into the model, which indicates the difference between the price that's listed for a stock and what it can actually be bought and sold for. This was a pretty conservative estimate.
- **Hardware**: The algorithm was run in parallel on a 16-thread CPU with clock speeds of up to 4.2 GHz and 16gb of RAM.

The algorithm took approximately three hours to run and produced the portfolio shown in **Table 2.** I set the desired risk tolerance to 1.00% anad the algorithm returned six out of the 492 total companies to apply the AdaBoost strategy to in conjunction. The algorithm

had a mean expected return *per day* of 0.42%. This means that, on average, the algorithm expects to make 0.42% per year. To put this into context, this is an expected return of +187.5% annually. This shows just how profitable this strategy can be and this was only with 100 rounds of boosting. The previous data shows more rounds of boosting can produce even more profitable results.

**Table 2.** Optimal Portfolio with 1.00% Risk Tolerance (0.42% daily expected returns).

| **Stock Symbol** | **Portfolio Makeup** |
|---|---|
| AVGO | 17.86% |
| EQIX | 17.64% |
| HUM | 18.76% |
| PANE | 15.61% |
| NOW | 15.32% |
| ULTA | 14.81% |

# Conclusions and Discussion

To summarize, these contemporary classification methods have shown compelling accuracy in predicting price moves in a highly volatile market. Trained on the Exxon Mobil dataset, the AdaBoost algorithm outperformed Kernel Ridge Regression and Support Vector Machines. Further, the higher the rounds of boosting, the more accurate the algorithm became. This is due to the margin-increasing phenomena first described by Bartlett+ et al. [10]. However, AdaBoost is an inherently sequential processing algorithm; the next round of boosting fundamentally depends on the previous round. Hence, this algorithm does not lend itself to parallel computing like like Neural Networks do. The autoregression validation we discussed, however, can be run in parallel. The total number of AdaBoost models that need to be trained if we wish to evaluate $n$ companies on $m$ datapoints is $n \times m$. This means that for the Portfolio Optimization section of this paper, we trained approximately 250,000 different AdaBoost models, which means that the algorithm underwent approximately 2,500,000 rounds of boosting. The computational complexity of this problem makes this difficult to validate at scale using a sufficiently high number of rounds of boosting. This is still desirable to implement given its level of profitability.

## Author's Note

I thoroughly enjoyed working on this investigation and was happy to see some promising results. I have limited experience with financial time series analysis so it was fun to dabble in a new discipline. I implemented a version of this algorithm on a Raspberry Pi 4 that uses AdaBoost to trade with fake money in real-time and we'll see how well it performs!

# References

[1] *Use of oil - U.S. Energy Information Administration (EIA)*. [Online]. Available: https://www.eia.gov/energyexplained/oil-and-petroleum-products/use-of-oil.php (visited on 12/06/2023).

[2] S. Borovkova and I. Tsiamas, "An ensemble of LSTM neural networks for high-frequency stock market classification," en, *Journal of Forecasting*, vol. 38, no. 6, pp. 600–619, 2019, _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/for.2585, ISSN: 1099-131X. DOI: 10.1002/for.2585. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/for.2585 (visited on 12/06/2023).

[3] X. Zhong and D. Enke, "A comprehensive cluster and classification mining procedure for daily stock market return forecasting," *Neurocomputing*, vol. 267, pp. 152–168, Dec. 2017, ISSN: 0925-2312. DOI: 10.1016/j.neucom.2017.06.010. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231217310652 (visited on 12/06/2023).

[4] V. Kotu and B. Deshpande, *Data Mining Process - an overview — ScienceDirect Topics*. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B978012801460800001X (visited on 12/06/2023).

[5] K. Schierholt and C. Dagli, "Stock market prediction using different neural network classification architectures," in *IEEE/IAFE 1996 Conference on Computational Intelligence for Financial Engineering (CIFEr)*, Mar. 1996, pp. 72–78. DOI: 10.1109/CIFER.1996.501826. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/501826 (visited on 12/06/2023).

[6] M. Ivanova and L. Dospatliev, "APPLICATION OF MARKOWITZ PORTFOLIO OPTIMIZATION ON BULGARIAN STOCK MARKET FROM 2013 TO 2016," *International Journal of Pure and Applied Mathematics*, vol. 117, no. 2, pp. 291–307, Dec. 2017, Publisher: Academic Publications, Ltd., ISSN: 1311-8080. [Online]. Available: https://ijpam.eu/contents/2017-117-2/5/index.html (visited on 12/06/2023).

[7] J. Xia and J.-A. Yan, "Markowitz's Portfolio Optimization in an Incomplete Market," en, *Mathematical Finance*, vol. 16, no. 1, pp. 203–216, 2006, _eprint: https://onlinelibrary.wiley.com/do 9965.2006.00268.x, ISSN: 1467-9965. DOI: 10.1111/j.1467-9965.2006.00268.x. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-9965.2006.00268.x (visited on 12/06/2023).

[8] K. Budhathoki and J. Vreeken, "MDL for Causal Inference on Discrete Data," in *2017 IEEE International Conference on Data Mining (ICDM)*, ISSN: 2374-8486, Nov. 2017, pp. 751–756. DOI: 10.1109/ICDM.2017.87. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8215550 (visited on 12/06/2023).

[9] Y. Freund and R. Schapire, "A Short Introduction to Boosting," 1999. [Online]. Available: https://www.semanticscholar.org/paper/A-Short-Introduction-to-Boosting-Freund-Schapire/c834bddd5e75a64ca9bb80c195cf84345c38bb9b (visited on 12/06/2023).

[10] P. Bartlett, Y. Freund, W. S. Lee, and R. E. Schapire, "Boosting the margin: A new explanation for the effectiveness of voting methods," *The Annals of Statistics*, vol. 26, no. 5, pp. 1651–1686, Oct. 1998, Publisher: Institute of Mathematical Statistics, ISSN: 0090-5364, 2168-8966. DOI: 10.1214/aos/1024691352. [Online]. Available: https://projecteuclid.org/journals/annals-of-statistics/volume-26/issue-5/Boosting-the-margin--a-new-explanation-for-the-effectiveness/10.1214/aos/1024691352.full (visited on 12/06/2023).

[11] M. Mohri, A. Rostamizadeh, and A. Talwalkar, *Foundations of Machine Learning, second edition*, en. MIT Press, Dec. 2018, Google-Books-ID: dWB9DwAAQBAJ, ISBN: 978-0-262-35136-2.

[12] V. Vovk, "Kernel Ridge Regression," en, in *Empirical Inference: Festschrift in Honor of Vladimir N. Vapnik*, B. Schölkopf, Z. Luo, and V. Vovk, Eds., Berlin, Heidelberg: Springer, 2013, pp. 105–116, ISBN: 978-3-642-41136-6. DOI: 10.1007/978-3-642-41136-6_11. [Online]. Available: https://doi.org/10.1007/978-3-642-41136-6_11 (visited on 12/06/2023).

[13] N. Cristianini and J. Shawe-Taylor, *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*, en. Cambridge University Press, Mar. 2000, Google-Books-ID: _PXJn_cxv0AC, ISBN: 978-0-521-78019-3.