Machine Learning and the Stock Market

Jonathan Brogaard[*], Abalfazl Zareei[+]

Abstract

Practitioners allocate substantial resources to technical analysis whereas academic theories of market efficiency rule out technical trading profitability. We study this long-standing puzzle by applying a diverse set of machine learning algorithms. The results show that an investor can find profitable technical trading rules using past prices, and that this out-of-sample profitability decreases through time, showing that markets have become more efficient over time. In addition, we find that the evolutionary genetic algorithm's attitude in not shying away from erroneous predictions gives it an edge in building profitable strategies compared to the strict loss-minimization-focused machine learning algorithms.

JEL classification: B26, G12, G14, C58, N20

Keywords: Technical trading, Machine learning, Big data analysis

---

This paper was previously circulated under the title "Machine Learning, Anomalies and Market Efficiency." We thank Juhani T. Linnainmaa, Lars Norden, Björn Hagstromer, Jose Marin, and Pedro Serrano, Deniz Anginer (discussant), Walter Pohl (discussant), Federico Maglione (discussant) as well as seminar participants at the Stockholm Business School, 2017 Paris Financial Management Conference, NFN Young Scholar 2018, 2nd Annual European Quantitative and Macro Investment Conference, Paris December Finance Meeting 2020, Man Investments Inc., Menta Capital, and Lynx Asset Management. [*]Jonathan Brogaard, David Eccles School of Business, University of Utah, (Email) brogaardj@eccles.utah.edu; [+]Abalfazl Zareei, Stockholm Business School, Stockholm University, (Email) abalfazl.zareei@sbs.su.se.

Technical trading rules use past prices and volume data to infer future prices. These rules are used widely by investment professionals, and advertised regularly by popular retail brokers and finance websites. In academia, there is an ongoing debate over the profitability of technical trading rules. Academics' semi-strong theory of market efficiency states that share prices reflect all publicly available information and thus trading rules based on prices and volume should not be profitable. Even the weak-form theory of market efficiency says that prices and volume information cannot be used to predict future returns.

The evidence gathered by researchers on the profitability of technical trading rules is contradictory. Some studies find technical trading profitability (Neftci, 1991; Brock, Lakonishok, and LeBaron, 1992; Neely, Weller, and Dittmar, 1997; Sullivan, Timmermann, and White, 1999; Lo, Mamaysky, and Wang, 2000; Kavajecz and Odders-White, 2004), and others conclude the opposite (Fama and Blume, 1966; Bessembinder and Chan, 1998; Allen and Karjalainen, 1999; Ready, 2002; Bajgrowicz and Scaillet, 2012). Bajgrowicz and Scaillet (2012) argue that the studies concluding in favor of technical trading profitability are suffering from two main issues: (1) they do not take the transaction costs into account, and (2) data-snooping, that is whether the investors could have chosen the best performing trading rules ex ante. Examining a set of 7,846 trading rules, Bajgrowicz and Scaillet (2012) show that an investor would have never been able to select the best performing strategies with profitable out-of-sample performance.

In this paper we exploit machine learning technology to search for profitable trading rules. We do so by utilizing a diverse set of methods in the machine learning repertoire, including evolutionary genetic algorithm besides standard loss-minimization algorithms i.e. support vector machine, decision tree, random forest and ensemble learning. In addition, our experiments are accompanied by rigorous controls for data-snooping and transaction costs. Our results show that an investor would have been able to find profitable technical trading rules ex

ante, but that this out-of-sample profitability decreases through time. Moreover, our findings advocate using evolutionary genetic algorithm over loss-minimization-based machine learning algorithms (such as random forest and decision trees).

Developments in machine learning gave rise to many different methodologies that can be divided in two parts based on the optimization objectives: (i) the standard machine learning algorithms that are concerned with minimizing mean-square errors of the model fit; (ii) evolutionary algorithms that are suitable to optimize any economic objective. The lack of restriction in setting the search direction result in the evolutionary algorithms not shying away from erroneous predictions for the purpose of achieving more accurate overall fit to the data. This paper examines the profitability of technical trading rules via the lens of machine learning algorithms and in doing so, it provides a comprehensive comparison of the evolutionary relative to the standard machine learning algorithms.

We start by applying the evolutionary genetic algorithm to the technical trading profitability puzzle. The genetic algorithm views the search for profitable trading rules from an optimization perspective. The optimization problem is defined by three ingredients: a search space, an objective function, and a set of constraints. The search space is a large set of trading strategies, the objective is to find strategies with high risk-adjusted returns, and the constraints are whether the risk-adjusted returns are statistically significant and whether the strategies incur low transaction costs. The optimization problem has several local optima given that several trading rules can yield high and statistically significant risk-adjusted returns.

The genetic algorithm machine learning technique adopts the principles of natural evolution and is founded upon the notion that "the strongest survive." It starts with a randomly generated population (a set of trading rules) and applies the concept of evolution to generate stronger population sets. It repeats this routine to find the strongest members that survive until the final generation. The innovative feature of the genetic algorithm approach is the way in

which the notion of evolution is applied to generate stronger population sets and to find several local optima.

We control for data-snooping in the genetic algorithm's search for profitable rules. If one searches long enough one can always find a profitable rule; however, this may be just a fortunate coincidence. We take several steps to combat the data-snooping issue in the genetic algorithm process. First, in the optimization procedure we separate the performance of trading rules into two distinct periods: a training period and a selection period. The selection period acts as a validation filter to make sure that the trading rules found by the algorithm in the training period are not simply a fortunate aberration. Second, for those trading rules that perform well in the training and selection periods we evaluate the performance in a third period, the out-of-sample period. Consistent out-of-sample performance indicates that machine learning can be used to detect profitable trading rules. Third, we evaluate the performance of strategies in other datasets as an additional out-of-sample robustness.

For our baseline results, we apply the algorithm to find profitable technical trading rules in the different NYSE/AMEX volatility-decile portfolios between January 1, 1965 to December 31, 2014. With the volatility-decile portfolios, we can see whether variation in information uncertainty (noise-to-signal ratio or volatility) influences the profitability of trading rules. This is because with higher information uncertainty, investors tend to underreact to public information (Zhang, 2006) and rely more on technical signals, compared to the fundamental signals (Han, Yang, and Zhou, 2013).

With the economic objective of maximizing the out-of-sample abnormal returns, we find that the algorithm consistently selects technical trading rules that perform well out-of-sample. The set of optimized technical trading rules uncovered by the machine learning algorithm are able to generate an average out-of-sample four-factor alpha of 25.8% annually for a diversified portfolio of volatility deciles. We compare this paper's approach to an alternative set of

technical trading strategies based on moving averages examined by Han, Yang, and Zhou (2013). Our approach outperforms their set of moving average strategies on average by 21.6% annually. In addition, out-of-sample profitability decreases through time. On average, for each year since 1975, the abnormal return of optimized rules decreases by 0.63%. The decrease in abnormal returns is consistent with markets becoming more efficient over time.

We take several steps to address the data-snooping concerns in the genetic algorithm's search. First, we conduct a placebo test to show that the results are not driven by some mechanical regression process or a fortunate adaptation of rules to the data. We scramble the dataset along the time dimension and apply our algorithm to the scrambled data. Any predictability from prices should be broken in the dataset, and so applying the optimization procedure should fail to identify profitable technical trading rules. We apply this placebo test to the NYSE/AMEX volatility-decile portfolios and find that the optimized set of trading rules consistently produces negative alphas out-of-sample.

In addition, we examine the performance of the genetic algorithm on other datasets (i.e. NYSE/NASDAQ/AMEX size-decile portfolios and large stocks) and also with other objective functions (i.e. Sharpe ratio). When testing the NYSE/NASDAQ/AMEX size-decile portfolios from January 1, 1965 to December 31, 2014, the results show the same positive out-of-sample performance as in volatility-decile portfolios. We also apply our algorithm to individual stocks. We use a rolling set of 100 highly capitalized stocks in each out-of-sample year from 1975 to 2014, apply our algorithm to each individual stock, and build a diversified portfolio across the optimized trading rules and stocks. This portfolio, after including transaction costs, earns consistent positive excess returns and generates high Sharpe ratios out-of-sample. This profitability is orthogonal to market, size, value and momentum factors, and it decreases through time with a substantial drop after 2005.

Next, we investigate the profitability of technical trading using a set of standard loss-

minimization machine learning algorithms i.e. linear classification models (regularized SVM + logistic regression), decision tree, KNN, random forest and an ensemble of machine learners (that include boosting besides bagging of classification trees). We use the same investment procedure as the genetic algorithm that is in each out-of-sample year, we use the last 10 years of price data to train the machine learning algorithm for the purpose of predicting buy and sell signals. To overcome possible overfitting problems, we use a ten-fold cross-validation procedure to train the machine learning models. Applying the standard machine learning algorithms to the 10 NYSE/AMEX volatility decile portfolios, we observe similar pattern as the genetic algorithm. The out-of-sample alphas are promising in the early years (the 1970s) with alphas ranging up to 60% annually net-of-cost. However, the performance depreciates post 2000 that matches nicely with the findings with the genetic algorithm.

We compare the performance of the genetic algorithm to the standard machine learning algorithms by regressing the genetic algorithm's out-of-sample returns on the standard machine learning algorithms' returns when solving the same problem: investing on the 10 NYSE/AMEX volatility decile portfolios from 1975 to 2014. The results show a positive and significant intercept (generalized alpha) for each of the standard algorithms that translates into better performance of the genetic algorithm relative to the standard machine learning algorithm. For example, the genetic algorithm earns a generalized alpha of 9.32% (t=8.99) relative to the Random Forest machine learning algorithm. Overall, the results show that the genetic algorithm's flexibility in setting up the search space, the economic objective and the add-in constraints gives it an edge relative to the standard machine learning algorithms such as decision trees and random forest.

This paper contributes to several strands of literature. First, we add to the studies that examine the profitability of technical trading rules. One main challenge in these studies is data-snooping. Researchers tackle this issue in different ways. For example, Sullivan, Timmermann,

and White (1999) locate technical trading profitability by employing the data-snooping bootstrap reality check used by White (2000). However, this data-snooping test is flawed because we cannot know if the investors could have chosen those technical trading rules without the benefit of foresight. Bajgrowicz and Scaillet (2012) show that, for a set of 7,846 trading rules, and using false discovery rate methodology to account for data-snooping, an investor would have never been able to select the best performing strategies with profitable out-of-sample performance. Our paper addresses the data-snooping concern by using a machine learning algorithm that constructs technical trading rules for ex ante trading over the previous ten years. We show that an investor could have profited from technical trading rules ex ante, although this profitability has dropped substantially in recent years.

The paper also contributes to the literature on asset pricing anomalies. Anomalies in asset pricing are cross-sectional or time-series empirical patterns that are inconsistent with a central asset pricing model. Systematically searching for anomalies is a slow process; there are voluminous amounts of data that can be combined and conditioned, and, even if an anomaly is identified, it may be due to overfitting. Yan and Zheng (2017) searched for cross-sectional anomalies among 18,000 trading signals constructed from accounting information. Our paper, in contrast, uses machine learning technology to hasten and regulate the search for *time-series* anomalies. We use the recent developments in machine learning that combine data-mining and optimization techniques while controlling for data-snooping. We set the fitness function in the genetic algorithm to find out-of-sample four-factor alphas, and this fitness function can be modified to account for other factor exposures.

In addition, this paper contributes to ongoing research on using machine learning in empirical asset pricing (see, e.g., Gu, Kelly, Zhou, 2020; Rossi, 2018; Chen, Pelger and Zhu, 2019). These papers apply machine learning to the current known predictors in asset pricing and show that machine learning adds to the out-of-sample performance by capitalizing on non-

linearities among already known predictors. Our paper departs from these studies. Our machine learning algorithm does not have prior knowledge on the shape of the viable predictors. We feed price information into the algorithm, which extracts predictors (features) from the prices. In short, our algorithm can be looked at as a mix of data-mining and machine learning technology.

Our findings provide empirical support for the studies documenting anomaly and technical trading performance deterioration through time (e.g., Linnainmaa and Roberts, 2018; McLean and Pontiff, 2016). The idea is that, because of increased computing power, "arbitrageurs' ability to attack mispricing has improved over time" (Linnainmaa and Roberts, 2018). On this note, Dugast and Foucault (2020) provide a theoretical study showing that "greater computing power raises the average quality of the predictors used in equilibrium and therefore price informativeness. The first effect raises speculators' expected trading profit while the second reduces it." This theoretical finding fits nicely with the results in our paper. Our algorithm requires a vast amount of computational power.[1] Nordhaus (2001) shows that computing power has increased by an average of 55% per year since 1940, with growth post-1980 at around 80% per year. The phenomenal increase in computational power is necessary for our paper's analysis. The profitability of our algorithm in earlier years (e.g., the 1970s), when current computing power was barely even imaginable, matches the first effect in Dugast and Foucault (2020) — the increase in the quality of predictors (and subsequently, trading profits). Later on, as the computing power grows even more (post-2000), we observe the increase in price informativeness that leads to a decrease in trading profit.

[1] In our study, the average time needed to find the optimum trading rules for a diversified portfolio of ten NYSE/AMEX volatility deciles for the 40-year sample using a computer with an Intel® Core (TM) CPU i7-2600 and 16 GM RAM is 459.29 days (11,022.97 hours). We run our analysis on a supercomputer with an Intel Xeon E5-2690v4 CPU and 128 GB memory per node via parallel computing procedures.

1. Searching for Profitable Trading Rules

This section describes the machine learning algorithm. Our search is focused on finding profitable technical trading rules based off of past prices. An example of a technical trading rule is a moving-average strategy (Han, Yang and Zhou, 2013; Han, Zhou, and Zhu, 2016), which functions by producing buy and sell signals based on price patterns. Han, Yang and Zhou (2013) finds that moving average strategies generate positive and statistically significant abnormal returns through time.

There are three benefits in concentrating the search process on technical trading rules. First, modeling the technical trading rules is straightforward and gives us a computationally inexpensive opportunity to perform a data-mining procedure. Second, technical analysis encompasses prediction rules with unknown statistical properties using past information. The rules are often developed ad hoc by practitioners. Perhaps the most well-known technical trading rule that has penetrated academia is the momentum strategy. The Jegadeesh and Titman (1993) momentum strategy has become a core asset pricing factor, starting with the four-factor Fama and French model (Carhart, 1997). Third, because trading rules operate by generating buy and sell signals, they can be directly applied as a trading strategy. The next subsections elaborate on the apparatus of the machine learning algorithm.

1.1. Searching Mechanism

Our objective is to find strategies with the highest risk-adjusted returns (alphas). The constraints are (i) for the risk-adjusted returns to be positive and statistically significant and (ii) for the incurred transaction costs to be lower than a specific level. We formulate the search criteria as an optimization problem:

$$maximize\ alpha(strategy)$$

$$subject\ to: \tag{1}$$

$$p - value\ (strategy) < level_{p-value}$$
$$transaction\ cost\ (strategy) < level_{transaction\ cost}$$

where $alpha(strategy)$ is the risk-adjusted return of the trading rule. We require the alpha to be statistically significant by requiring the p-value to be lower than a specific value, $level_{p-value}$.[2] In addition, $level_{transaction\ cost}$ is the maximum level of transaction costs that we want the strategy to incur. The above optimization function can be adopted to include other objective functions (e.g., minimizing variance, maximizing the Sharpe ratio) and other constraints (e.g., drawdown).

Solving the optimization problem in equation (1) involves several obstacles. First, the objective function is non-differentiable. The abnormal return of a technical trading rule is a discrete number and depends on the form of the rule under review, so we cannot employ the conventional gradient-based methods to find optimal solutions. Second, the search space may have several local optima, and we may get trapped in a non-profitable trading rule in the search life-span. Third, the size of the search space is large because of the multitude of possible technical trading rules. To overcome these challenges, we employ a genetic algorithm approach. The genetic algorithm is ideal for solving optimization problems with non-differentiable objective functions. In addition, the stochastic nature of a genetic algorithm decreases the chances of getting stuck in a local optimum, so such an algorithm is suitable for finding optimum solutions in large search spaces. These characteristics make genetic algorithm approach preferable for our analysis (Allen and Karjalainen, 1999; Neely, Weller, and Dittmar,

[2] The variable, $level_{p-value}$, can also function as a control mechanism for data-snooping. Mclean and Pontiff (2016) and Harvey, Liu, and Zhu (2016) have pointed out the concerns of data mining due to the large number of anomalies being discovered in the literature and in response they propose increasing the t-statistic threshold for accepting anomalies.

1997; Potvin, Soriano and Vallee, 2004).

Genetic algorithms adopt the principles of natural evolution (schema theorem: best observed building block or schema survives) in searching for an optimal solution. The approach was developed by Holland (1962, 1975) and since then has been applied in various fields such as economics, management science, engineering, and cognitive science. The genetic algorithm generates a sample population of solution candidates. The solution candidates are ranked according to a specific fitness function (objective function). A new population is generated by combining solution candidates according to their relative fitness using crossover and mutation operators. The crossover operator generates new solution candidates by inserting partial characteristics of fitted candidates into newly born solution candidates. The mutation operator inserts random changes in the structure of solution candidates to generate new solutions. The algorithm keeps generating new fitted populations until a stop criterion is reached.

1.2. Trading Rule Representation

Any trading rule representation is set to have three main characteristics. First, as we want all available possibilities to be reachable before starting the search process, it should cover a large set of viable technical trading rules. Second, the trading rules should be feasible. A random integration of functions and operators may result in meaningless rules that only increase the computational expensiveness of the search process. Thus, the trading rules should follow a predefined structure that guarantees the generation of reasonable trading rules. Third, any large set of rules should encompass well-known technical trading rules.

Our trading rule representation insures these characteristics. We adopt a tree-format representation of the solution candidates. However, their representation suffers from the absence of a definitive imposed structure on trading rules, resulting in the generation of meaningless trading rules that make the algorithm's job of finding optimum solutions

cumbersome. Improving upon their representation, we ensure that the generated solution candidates are sensible. Instead of blindly combining functions and operators to construct a trading rule, we confirm that the generated trading rules are valid and acceptable (improving the closure property of the solution candidates where all trees are to be synthetically valid composite functions). We ensure validity by imposing a level-based structure on the trading rules whereby in each level a definitive set of functions and operators are included. A description of trading rules encoding and lists of functions and operators in each level is described in Table 1.

[Table 1 About Here]

The tree-structure has four levels. In the root node, level 1, we use Boolean operators and functions (If-then-else, and, or) that establish the buy or sell signal. In the second level, we incorporate relational operators ($>, <$) that return zero or one values. In the third level, we include real functions (Average, Maximum, Minimum, Median, Lag, Volatility, RSI and Filter). Selecting Average, Maximum, Minimum and Lag functions are supported by Neftci (1991), who shows that many trading rules rely on specific patterns of local extremes of past data.

The Volatility function compares the volatility of the underlying variable in the selected days to the volatility across the entire input information length. RSI, the relative strength index, is a technical indicator determining whether a stock is over-bought or over-sold. It is measured by $RSI = 100 - 100/(1 + RS^*)$, where RS^* is the average number of days that the underlying variable is above its average divided by the average number of days that it is lower than its average. Assuming the underlying variable is the asset's price, an RSI above 70 regards the asset being over-bought, and a value lower than 30 suggests it is over-sold.

The Filter operator is responsible for generating trading signals similar to filter rules in the context of technical analysis. Assuming price is the underlying variable for the operator, the Filter operator takes in two parameters, Pr: a value between -1 and 1 and *Days*: number of days, and produces $P_{t-Days} + Pr * P_{t-Days}$, where P_{t-Days} is the price value number of $Days$ before today, t. Comparing the price today with the calculated price, the rule generates a trading signal.

Level 4 in the trading rules' tree structure includes the input variables. We use price and return as the input market information that is inserted in the signal generation process.[3] The number of observations inserted into the trading rules depends on the terminal, $Days$, that specifies the number of days before the current day. Moreover, if the real function Filter is chosen in level 3, we also include Pr, a random number between -1 and 1, as an input to the trading rule.

Figure 1 presents an example of a possible trading rule. This rule consists of two branches. The left one generates a short-selling signal. According to the rule, we short the asset if the price is lower than the average of prices in the last 80 days. The right branch is responsible for the buy signal. We buy the underlying asset if the price today is higher than the average price in the last 20 days. In the case when the branches generate opposite signals (buy and sell), we hold the risk-free asset. The example in Figure 1 presents a case in which the root node includes an "If-Then-Else" operator. A more complicated case of trading rules can employ "and" and "or" operators in the root node.

[Figure 1 About Here]

[3] Trading volume is an additional variable that can provide useful information into the trading rule mechanism (Blume, Easley and O'Hara, 1994; Grundy and McNichols, 1989). We use volume as input information when we apply our algorithm to individual stocks in Section 4.3.2.

The four-level representation of trading rules can generate up to 130 million possible rules using data only from the last 100 days. In addition, the mechanism is able to produce well-known technical trading rules, such as filter, moving average, support and resistance, and breakout rules.

1.3. Fitness Value

The fitness value sets the search direction in the genetic algorithm. This function combines the objective function and the constraints in the optimization problem in expression (1). In this subsection, we explain how the fitness value is computed.

For a trading rule, a zero-cost portfolio is calculated by subtracting the trading rule's return ($\tilde{R}_t$) from the return of the buy-and-hold strategy (R_t). The portfolio's return is $PR_t = \tilde{R}_t - R_t$. Next, we regress the zero-cost portfolio return on the Fama and French four-factor portfolio returns [4]:

$$PR_t = \alpha + \beta_{MKT} r_{MKT} + \beta_{SMB} r_{SMB} + \beta_{HML} r_{HML} + \beta_{MOM} r_{MOM} + \epsilon_t, \qquad (2)$$

where r_{MKT} , r_{SMB}, r_{HML}, and r_{MOM} are the returns on market, size, value and momentum portfolios. We apply Newey and West (1987) standard errors. α measures the abnormal risk-adjusted return and is the variable that our algorithm aims to maximize. From equation (2) we obtain the value of α and its corresponding *p*-value.

We also take the transaction costs into account. Following Balduzzi and Lynch (1999), Lynch and Balduzzi (2000), Han (2006) and Han, Yang and Zhou (2013), transaction costs are assumed to be incurred when trading the underlying asset (long or short). We assume no costs when trading the 30-day T-bill. We calculate the breakeven transaction cost (BETC) value that

[4] We consider alternative asset pricing models, CAPM, three-factor and five-factor models, in the robustness tests.

makes the average return of the trading strategy's return ($\tilde{R}_t$) equal to zero. Balduzzi and Lynch (1999) use one basis point (bps) and 50 basis points (bps) as the lower and upper bounds for transaction costs. In order to show the algorithm's ability to identify profitable tradeable rules, we require the search to find trading rules with a BETC higher than 25 bps in our baseline analysis.[5]

The fitness value for trading rule i, after accounting for p-value and transaction costs, is computed as:

$$fitness_i = \begin{cases} \alpha_i, & if\ p-value_i \leq 0.1\ and\ BETC_i \geq 25\ bp \\ -M, & Otherwise \end{cases}, \quad (3)$$

where α_i is the alpha value computed for trading rule i, and M is an arbitrarily large number. This fitness function in our baseline analysis ensures the survival of trading rules with abnormal returns that are statistically significant at least at the 10% level while the BETC is higher than 25 bps.

1.4. Optimization Procedure

Having designed the search mechanism, the next step is to optimize the testing procedure. We start by creating a set of randomly generated trading rules from the universe of possible trading rules called the *primary set*. The primary set represents the raw trading rules before any optimization procedure takes place. The population evolves via genetic algorithm by producing new populations while ensuring the survival of the fittest.

To create the new evolved populations of trading rules, we employ crossover and

[5] Lynch and Balduzzi (2000) consider the same level of transaction cost. In addition, Sadka and Scherbina (2007) investigate the TAQ dataset and estimate 25 bps as the average effective spread for a typical stock and a typical trade. Besides, in section 4.3.2 when trading individual stocks, we use the yearly fixed transaction cost estimates from Jones (2000) before 2000 and the actual daily quoted bid-ask spreads for each individual stock from TAQ dataset after 2000.

mutation operators. The crossover operator aims to produce new trading rules by using the characteristics of the existing population members. In crossover, we randomly select two rules in the existing population ("Parents" in the genetic algorithm context) and switch one branch from the buy-side of one rule with a branch from a sell-side of another trading rule. The branches are chosen randomly in both rules. Two new rules are generated. In the genetic algorithm literature, the new rules are referred to as offsprings. Appendix A demonstrates a crossover operation.

The mutation operator aims to preserve the diversity in the population. We apply two different mutation operation techniques. First, we follow Allen and Karjalainen (1999) and implement mutations by doing a crossover between a randomly chosen rule in the existing population and a newly generated trading rule. Using the newly generated rule introduces additional diversity in the population. Second, we apply another form of mutation operation by simply inserting newly generated rules into the population. The two mutation operations increase genetic diversity in the optimization process.

We apply the crossover and mutation operations to the existing population in order to generate new sets of trading rules. We generate one new population using the crossover operator. Each crossover operation generates two new rules. We apply the crossover operator until we generate the same number of rules as in the population set. In addition, we use the mutation operator to generate two new populations. The three new populations, generated by cross-over and mutation operations, have the characteristics of the existing (survived) population as well as random characteristics to preserve the genetic diversity. The newly generated populations are merged with the existing one, and the fitness values are evaluated. Next, we sort the rules based on the fitness values and select a population of rules with the highest fitness values.

Applying the algorithm to a specific time-period potentially introduces a data-snooping

problem. Any profitable rule that we discover may only be so for those data points used in the optimization procedure. To address this data-snooping issue, we divide the time span of the algorithm's information input into two periods: a training period and a selection period. The selection period acts as a validation period to ensure that the profitable rules found by the genetic algorithm in the training period are not simply an example of data-snooping. We start by searching along the training period to construct new populations; after selecting the fittest trading rules in the training period, we re-evaluate them in the selection period. The successful rules from the selection period are added to a final dataset, the *Final Set*. In the next generations, the members in the Final Set are updated until the stopping criterion (in our case, a maximum number of generations) is reached. Table 2 presents the genetic algorithm procedure in detail.

[Table 2 About Here]

Two parameters that influence the performance of the optimization process are population size, |POP| (number of rules in the population set) and number of generations, |GEN| (the number of times that the population is reproducing using cross-over and mutation operators). A larger population size results in a higher number of optimized trading rules after the completion of optimization procedure. However, a larger population size makes the optimization more computationally expensive. The choice of population size thus involves a trade-off between a greater number of optimized solution candidates and computational cost. We follow Allen and Karjalainen (1999) and choose a population size of 500, a number large enough to result in a meaningful evaluation of the algorithm performance while still computationally manageable. In robustness tests we run a limited number of experiments with other population sizes (100, 1000, 2000), and the results are economically similar.

The choice of the number of generations, |GEN|, also carries a trade-off between

computational expensiveness of the procedure and optimized characteristics of the solution candidates. From the Darwinism perspective, more generations mean more competition, survival, and reproduction in the evolution process. In the optimization procedure, a higher |GEN| results in better-performing rules but also means more computational expense. We report the results using 20 generations. We use 50 and 100 generations for a limited number of robustness tests, and the results are qualitatively similar.

Allen and Karjalainen (1999) also apply a genetic algorithm to search for profitable trading rules. Their results show that the genetic algorithm cannot find profitable rules; However, we show that it has been possible to create profit out-of-sample. Apart from benefiting from much higher computational power, our algorithm is different in two aspects. First, we provide a standardized mechanism to generate technical trading rules. In particular, a trading rule has four levels with a predetermined set of variables or functions in each level. Because the machine learning algorithm's objective is to find local optima, by imposing a structure on the search space we make sure that the rules follow a valid structure, which substantially reduces the computational cost. Second, the fitness function that determines the appropriateness of trading rules is changed from excess returns to risk-adjusted return filtered by significance and transaction costs level. This leads to automatic removal of trading rules with statistically insignificant risk-adjusted returns and high transaction costs. One downside is that this form of setting the fitness function increases the computational cost of the optimization process. In the next sections, we evaluate the performance of our machine learning algorithm.

2. Can Machine Learning Find Profitable Rules ex ante?

In this section we examine the algorithm's performance. We begin by describing the data and a number of implementation choices, and, later, we report the results.

2.1. Data and Implementation

For the baseline analysis, we use ten NYSE/AMEX volatility-decile portfolios from July 1, 1965 to December 31, 2014 as the test assets.[6] Volatility-decile portfolios capture variation in information uncertainty across stocks. When there is higher information uncertainty, investors tend to underreact to public information (Zhang, 2006) and rely more on technical signals, compared to the fundamental signals (Han, Yang, and Zhou, 2013). Therefore, using volatility-decile portfolios gives us a laboratory to see whether higher uncertainty (or noise-to-signal ratio) leads to more profitable trading rules.

We assume a five-year training period and a five-year selection period.[7] The rolling analysis is as follows: we start from 1965 and use the five-year training period (1965-1969) and five-year selection period (1970-1974) as the input to the algorithm. After optimization, we test the performance of trading rules in the one year out-of-sample (1975). Next, we roll the input data by one year and use the five-year training period (1966-1970) and five-year selection period (1971- 1975) as input to the algorithm and evaluate the performance of the algorithm's output rules in the one year out-of-sample period (1976). We repeat the procedure until 2014. In addition, in each rolling window we perform the analysis 20 times (i.e., 20 simulations) to make sure that the results are not driven by an auspicious one-time run of the algorithm.

Our main interest is in the performance of the Final Set, which contains the set of optimized technical trading rules. As an alternative benchmark, we use the set of moving-average strategies in Han, Yang and Zhou (2013) who show that moving average (MA) strategies generate positive and economically and statistically significant CAPM, Fama and French three-factor, and Fama and French four-factor alphas with low transaction costs.[8] They

[6] Later as a robustness check we consider NYSE/AMEX/NASDAQ size decile portfolios and also individual stocks.

[7] We also study different duration of training and selection periods (3, 8, and 10 year) and the results are qualitatively similar. Overall, for longer duration of training and selection periods, we get better results; however, longer duration leads to higher computational cost and less out-of-sample data.

[8] Han, Yang and Zhou (2013) use 10 NYSE/AMEX volatility decile portfolios between July 1, 1963 to December

use a list of moving-average strategies with lag lengths from 3 to 200. We follow the same procedure and generate a set of moving average strategies with lags between 3 to 100, which we call the Moving-Average Set.[9]

We also consider the Primary Set as a benchmark. Recall that the Primary Set is the initial randomly drawn trading rules over which we optimize to produce the Final Set. If the optimization adds no value, then the Final Set and the Primary Set should perform similarly.

2.2. Asset-by-Asset Performance

Table 3 reports the summary statistics of annualized out-of-sample four-factor alphas for the three sets of trading rules across volatility-decile portfolios.

[Table 3 About Here]

Table 3 has four key insights. First, the Final Set generates higher average alphas than the other trading rule sets across volatility decile portfolios. For example, in the lowest volatility-decile portfolio, the Final Set generates a 14.27% annualized abnormal return, while the Moving-Average Set and Primary Set generate abnormal returns of 4.76% and -1.26%, respectively. Second, as the portfolio volatility increases, average alpha values increase for the rules in the Final Set and the Moving-Average Set. For example, the Final Set in the lowest volatility-decile portfolio generates 14.27% and for the highest volatility decile portfolio, it generates 22.98%.

Third, the difference between the average alphas generated by the Primary Set and the

31, 2009 to show the well-performance of moving average (MA) strategies. In an untabulated analysis, we compare our optimized rules with MA strategies in the same exact dataset and the same periods as in Han, Yang and Zhou (2013). The relative performance of strategies is economically similar to the baseline results in our paper.

[9] We choose 100 because in the baseline optimization we use the information up to 100 days before. The results are economically similar when using 200 days.

Final Set confirms that the algorithm adds value. The Final Set contains strategies that generate positive abnormal returns whereas the Primary Set does not. For instance, in the lowest volatility decile portfolio, the Primary Set generates an average of -1.26%, and the Final Set generates an average of 14.27%, showing that the algorithm successfully optimizes. Fourth, the algorithm finds fewer number of rules than the maximum number of possible rules (that is, the size of the population set, or 500 rules). For the lowest volatility portfolio, the algorithm finds only 332 unique trading rules that satisfy the search criteria.

2.3. Portfolio Performance

Instead of investing in each asset separately, we can more realistically create a portfolio. Here we consider a case in which we diversify across all of the volatility-decile portfolios. The analysis assumes an investor uses the algorithm, finds optimized rules specific for each asset, and invests equally across the rules. The summary statistic for the out-of-sample results is presented in Table 4.

[Table 4 About Here]

A diversified portfolio across the rules in the Final Set results in an average annualized alpha of 25.8%, higher than the Moving Average Set's alpha of 4.2% and the Primary Set's alpha of -1.3%.

Although the overall average statistics are useful to determine the overall performance of the different Sets, the time series dynamics is also of interest. Figure 2 presents the average annualized out-of-sample alphas for the different sets of trading rules in a diversified portfolio across volatility-decile assets through time.

[Figure 2 About Here]

The alpha values for the Final Set in Figure 2 show that the algorithm produces positive alphas most years. In addition, the Moving-Average Set usually generates positive returns through time, albeit at a lower level than the Final Set, and with more frequent and larger negative alpha years. The Primary Set consistently generates near zero or negative abnormal returns. Not surprisingly, blindly investing using randomly generated technical trading rules does not outperform the Fama and French three-factor benchmark.

The Final Set in Figure 2 appears to have a downward slope to its alpha generation over time. To test for a statistically significant negative time trend, we perform a simple OLS regression. We construct a time trend variable, $year_t$, which starts from one in 1975, the first year of out-of-sample results, and increases by one unit every year until 2014. We compute the average out-of-sample alpha values in the Final Set for each out-of-sample year. We regress the average yearly alpha values on the time trend variable. The results are presented in Table 5.

[Table 5 About Here]

The first row in Table 5 tests a portfolio of trading rules across all the volatility assets, and the next ten rows test each of the volatility assets separately. We focus on the overall result in the first row. The estimated coefficient on the time trend is negative and statistically significant. The negative coefficient of -0.63 implies that, as each year goes by, the performance of a set of optimized trading rules decreases by 0.63%. As the sufficient techniques and the computational power has been available since the early 2000s, it is feasible that the decreasing time trend is due to a one-time shift, not a gradual slope. It is beyond the scope of this paper to

attempt to disentangle the functional form of the time trend[10].

The decrease in out-of-sample alphas is consistent with the theoretical findings in Dugast and Foucault (2020). They show that the increase in computing power raises (i) average quality of predictors and (ii) price informativeness. In our setting, we can observe these two effects separately. In the early years (e.g., 1970s), implementing our algorithm was impossible because the current computational power was not available. Therefore, applying our algorithm led to the identification of high-quality predictors, and trading rules in the early years gain high average abnormal returns. Later in time, when the computing power and also the machine learning techniques become available, we see a reduction in trading rules' performance. This pattern is in line with an increase in price informativeness due to increased computational power that has led to lower expected trading profit[11].

The negative trend in the average alphas is also consistent with studies on anomalies' performance deterioration through time (e.g., Linnainmaa and Roberts, 2018). Our algorithm's job is to find an optimized set of time-series anomalies with an imposed trading rule structure (functions and input variables). We show that out-of-sample abnormal returns have declined, suggesting that markets have become more efficient. That is, fewer mispricings exist that violate the weak-form of market efficiency. Furthermore, our results do not depend on knowing the anomalies ex-ante. The conventional approach in investigating the performance of anomalies (Chordia, Subrahmanyam and Tong, 2013; McLean and Pontiff, 2016; Linnainmaa and Roberts, 2018) is constrained by the researcher knowing the anomalies ex ante, which creates a bias with the ad-hoc specification of anomalies. Using this paper's approach, we do

[10] In Appendix E, we examine the relationship between a measure of computational power — Number of transistors on integrated circuits (Transistor Count) — and the out-of-sample alphas. We show that there is a clear negative relation between the computing power and the out-of-sample alphas that is especially evident after the year 2000 with the sharp rise of computing power.

[11] We also investigate the performance of other machine learning algorithms in profiting out-of-sample from detecting patterns in past prices. Appendix D shows the out-of-sample results for five other machine learning algorithms: support vector machine, decision tree, KNN, random forest, an ensemble of machine learners. The results in Appendix D also shows that computational power improves market efficiency through time.

not need to know the anomalies; we systematically uncover them[12].

Next, we examine the risk-return trade-off by calculating the Sharpe ratio on the raw excess returns (not alphas). We compute the average value of the out-of-sample Sharpe ratios of rules across the simulations for a diversified portfolio of volatility deciles each year. Figure 3 reports the results.

[Figure 3 About Here]

The graph shows that using the machine learning algorithm can lead to sizeable positive Sharpe ratios through time that are consistently greater than one and that consistently decline through time. The average Sharpe ratio in 1975 is about 4.2 and it reduces to about 1 in 2014.

Next, we test whether the trading strategy is implementable or whether transaction costs nullify the alpha. We consider a one-sided transaction cost of 5 basis points. For each transaction that occurs we subtract 5 basis points from the return. As we move between a long position in the asset, holding the risk-free asset, and a short position in the asset we subtract a reasonable transaction cost. Figure 4 presents the results for abnormal returns after transaction costs for a diversified portfolio across volatility deciles. The figure shows that even after accounting for transaction costs the profitability of the optimized trading rules remains.

[Figure 4 About Here]

3. Characterizing the Trading Signals

This section examines the characteristics of optimized trading rules in the Final Set. First,

[12] We also examine the average BETC values across volatility decile portfolios through time. We find that the average transaction cost level needed to nullify the profitability of trading rules also decreases over time (with a sharp decline after 2005). This shows that over time, it has become harder to trade weak-form inefficiencies into positive returns.

we look at the direction of the positions that the rules hold over time. Next, we evaluate whether the buy or sell side signals drive performance. We also look into the different types of underlying functions used in the optimized rules. Before examining the portfolio allocation, we characterize the diversity of the selected trading strategies. We conclude by observing how trading rules evolve across the volatility deciles over time.

3.1. Signal Direction

Our search algorithm enjoys flexibility in selecting its investment direction. In this subsection, we study the frequency of buy, sell and hold risk-free signals generated by the optimized rules. As an example of what we count as a buy/sell signal, suppose that a rule generates signals as follows in five consecutive days: [buy-buy-buy-sell-sell]. Following the signals, we buy the underlying asset in day 1, hold it for three days, and then in day 4, change the position to a short-sell one. In the example we count one buy signal and one sell signal. We only count the signals that require an *action* from the investor. For each rule in the out-of-sample period we count the number of buys, sells, and out-of-market signals averaged across all of the simulations. The results are presented in Figure 5.

[Figure 5 About Here]

Figure 5 produces two key takeaways. First, the frequency of buy, sell, and neutral signals are relatively stable over the sample period. Second, the buy signals make up roughly 40%, and the sell and hold risk-free signals each make up about 30% of the actionable signals.

We next examine whether the performance of the trading rules comes from both the long and short positions or if one dominates the other. In Table 6, we look at the return summary statistics across the volatility portfolios generated from the buy, sell and hold-risk free signals.

[Table 6 About Here]

The values in Table 6 are the $(t+1)^{th}$ day returns conditioned on the type of signals (buy, hold, sell) at day t averaged over the out-of-sample years and the simulations. For deciles 1–8 long signals make up over 50%, and short signals make up less than 25%. In deciles 9 and 10 the long signal is less than 50%, and the short signal occurs more than 25% of the time.

Table 6 shows that both the short and long signals predict future returns and contribute to the algorithm's success. Optimized rules for the lowest and highest volatility-decile portfolio generate on average a mean return of 0.113% daily (28.48% annually) and 0.245% (61.74% annually), respectively. The corresponding value for a simple buy and hold strategy is 0.049% and 0.183% for the lowest and highest volatility-decile portfolios, respectively. The average next day return across all the strategies for the buy signals is 0.168% and 0.245% daily for the lowest and highest volatility-decile portfolios, respectively. The average next day return conditioned on the short-selling signals is 0.130% and 0.197% daily for the lowest and highest volatility decile portfolios, respectively.

3.2. Signal Functions

We further explore the types of functions and variables employed in the trading rules. Do the optimized rules tend to capitalize on a particular function or variable in their structure? Table 7 summarizes the characteristics of the trading rules in the Final Set when the underlying asset is the lowest and highest NYSE/AMEX volatility--decile portfolios.[13] All the values are averaged over the rolling out-of-sample windows over the 20 simulations.

[13] Because of the large number of optimized rules, we are unable to list them. For example, for the Final Set in the lowest volatility decile portfolio in Table 3, there are 332 rules on average across out-of-sample years. There will be $332 \times 20\ (simulations) \times 40(years) = 265{,}600$ rules in the overall analysis assuming the rules are unique through time. We address the diversity of trading rules in section 3.3.

[Table 7 About Here]

We focus on the results for the lowest volatility-decile portfolio (the highest volatility-decile portfolios follow a similar pattern). The maximum number of times a function or real variable (Price or Return) can be used is 1884 (when each side of trading rules is a two-branch tree). However, it is possible for the rules to have only one branch in the buy and sell side. The functions Minimum and Median are used the most. Price is used more than Return as an input to the optimized trading rules. The results also suggest that no single function or variable dominates. We explore the diversity of trading rules in the next subsection.

3.3. Strategy Diversity

The previous subsection suggests that the trading rules depend on a variety of signals. Here we directly test the diversity of the trading rules. Specifically, we examine whether all the strategies in the Final Set generate the same set of signals. Given that similar signals result in similar returns across the strategies, we focus the analysis on the returns. Table 8 reports the summary statistics for the pair-wise correlation values of the out-of-sample returns for the optimized trading rules in the Final Set.

[Table 8 About Here]

A correlation close to one implies the same signal-generation procedure along the optimized trading rules and that the algorithm uses the same signal generation procedure. Lower correlation values suggest that the optimization procedure finds dissimilar trading rules. On average, the correlation among strategies' returns is approximately 0.65. The moderate average

pairwise correlation reveals the usefulness of the optimization process in finding a diversified portfolio of technical trading rules.

Figure 6 presents the boxplot for correlation values through time. The correlation values across trading rules are computed for each out-of-sample year from 1975 and pulled together across simulations and assets. We draw the boxplot for each out-of-sample year. Figure 6 shows that the algorithm finds a variety of trading rules throughout the sample period.

[Figure 6 About Here]

3.4. Portfolio Allocation

Before testing the robustness of the machine learning algorithm, we consider how the algorithm allocates resources across the ten volatility deciles. Recall that the portfolio allocation depends on the number of trading rules that survive in the Final Set for each volatility decile. Thus, asking how the algorithm allocates resources across the volatility deciles is the same as asking how many trading rules in each volatility decile make it into the Final Set. Figure 7 presents the weight allocation across volatility-decile portfolios through time.

[Figure 7 About Here]

Figure 7 shows that through most of the sample we allocate weights equally across the volatility deciles. Each asset gets about 10% each year. However, starting in 2005, the weight allocation across decile portfolios changes. After 2005, the lowest and highest volatility-decile portfolios are the main investment vehicles. For example, in 2014, 35% of trading rules are from the lowest volatility-decile portfolio and another 30% are from the highest volatility decile. Although explaining the stark shift in portfolio allocation after 2005 is beyond the scope

of this paper, we find this a fascinating event[14].

In addition, in order to examine whether the same predictability mechanism drives the trading rules of different volatility decile portfolios, we apply the final rules of each volatility decile portfolio to other volatility decile portfolios. The results, reported in Appendix D, show that the trading rules found in the higher volatility-decile portfolios result in higher average out-of-sample alpha when applied to other decile portfolios. Thus, we conclude that different predictability mechanisms drive the set of rules found in each of the decile portfolios.[15]

4. Robustness

This section addresses three possible concerns. First, we conduct a placebo test to show that the results are not driven by some mechanical regression process or a fortunate adaptation of rules to the data. Second, we consider other asset pricing models to test whether the trading rule anomalies we find can be explained by other factor models. Third, as an additional out-of-sample test, and to ensure that the results do not merely hold for one specific set of assets, we repeat the analysis on different datasets, i.e., NYSE/NASDAQ/AMEX size-decile portfolios and individual stocks.

4.1. Placebo test

The results may be driven by some mechanical regression process or simply a fortunate adaptation of rules to the data. To address this concern, we run the following experiment. We scramble the dataset along the time dimension and apply the optimization procedure to the

[14] There can be several explanations for this stark shift in portfolio allocation after 2005. The possible explanations are the rise in computational power (See Appendix E), the increase in the utilization of machine learning models in the investment industry; and also changes in the regulation that advocate more competition and hence, more price informativeness (e.g. Reg NMS).
[15] We thank the reviewer for this constructive comment.

scrambled data. Any predictability from prices should be broken in the dataset, and so applying the optimization procedure should be unsuccessful in identifying trading rules. We use the ten NYSE/AMEX volatility-decile portfolios from July 1, 1965 to December 31, 2014. We assign a random number to each time period and sort the data by the random number. We use the same setting for the algorithm as in our baseline analysis in Section 2, i.e., five-year training periods, five-year selection periods, one-year out-of-sample period, 20 simulations, population size of 500, and generation number of 20. The results are presented in Table 9.

[Table 9 About Here]

The average out-of-sample alphas are negative consistent with expectations. If the performance of the optimization procedure is a result of detecting trends, as we argue, scrambling the data causes the optimization procedure to be unsuccessful. As expected, we find that the Final Set consistently produces negative alphas.

4.2. Choice of Asset Pricing Model

We use the Fama and French four-factor asset pricing model as the benchmark to calculate alphas. It is possible that the results are driven by some unmodeled risk factor. To address this, we reevaluate the out-of-sample performance of the optimized trading rules in the Final Set in Section 2 using CAPM (with market portfolio), three-factor (with market, size, and value portfolios, Fama and French, 1993) and five-factor (with market, size, value, profitability, and investment portfolios, Fama and French, 2015) asset pricing models. We obtain the data on the factor portfolios from Kenneth French's website. The results with the alternative asset pricing models are reported in Table 10.

[Table 10 About Here]

The optimized rules in the Final Set continue to consistently generate abnormal returns and to perform better than the Primary and Moving Average Sets.

4.3. Choice of underlying asset

The search procedure may possibly perform well because of the choice of the underlying assets (ten volatility-decile portfolios). To eliminate this possibility, and as additional out-of-sample tests, we repeat the analysis on the NYSE/NASDAQ/AMEX size-decile portfolios and also on individual stocks.

4.3.1. Size-decile portfolios

We first report the Kendall rank correlation coefficients between size-decile and volatility-decile portfolios to confirm that the size and volatility portfolios are sufficiently unrelated. Panel A in Table 11 reports the correlation values.

[Table 11 About Here]

The correlations between the portfolios are scattered between 0.31 and 0.78. Overall, the rank correlation between the size-decile and volatility-decile portfolios are low to moderate, so using the size deciles gives us a new setting for testing the algorithm's ability to recognize patterns.

We repeat the machine learning analysis on the size decile portfolios as the alternative dataset. We use the time period January 1, 1965 to December 31, 2014, and deploy the same search procedure (five years of training period, five years of selection period, and one year of

out-of-sample period). The results are presented in Panel B of Table 11 for the out-of-sample years from January 1, 1975 to December 31, 2014.

Panel B in Table 11 provides the summary statistics for annualized alpha values for the Primary Set, Final Set, and Moving Average Set. The Primary Set again performs poorly, consistently generating negative alphas across the size-decile portfolios. The moving average strategy performance improves as we move towards smaller decile portfolios, consistent with the results of Han, Yang and Zhou (2013). The Final Set performs better than the moving average strategies across the size deciles. The last row in Panel B of Table 11 shows the results for a diversified portfolio of rules across size decile portfolios. The average alpha for the Final Set is higher than the benchmark values of Moving Average Set and the Primary Set. While the Moving Average Set gains an average annualized alpha of 9.97%, the improved rules in the Final Set generate an alpha of 32.32% on average out-of-sample.

4.3.2. Individual Stocks

We also apply our algorithm to individual stocks. Because running our algorithm is computationally expensive we focus on the largest 100 stocks each year in the NYSE/AMEX/NYSE from 1965 to 2014 with prices higher than $5 (and non-missing prices over the last ten years). Beside prices, we also use daily volume as an input to the algorithm.

For individual stocks we use two separate fitness functions: (i) average excess return and (ii) Sharpe ratio. For average excess return, we set the fitness function equal to average return in excess of risk-free rate if the *t*-statistics of the null hypotheses with mean equal to zero is higher than 2; otherwise, the fitness function is equal to -M (M being a large value). For the Sharpe ratio, we set the fitness function equal to this ratio if the *t*-statistics of a null hypothesis of the average Sharpe ratio is higher than 2 (we do this test by bootstrapping each rule's daily returns); otherwise, the fitness value is a large negative number (-M).

In order to take the transaction costs into account, we use two datasets. First, for the transaction costs from 2000 to 2014, we take the daily quoted spread from the TAQ dataset as the actual transaction cost for each stock. For transaction costs prior to 2000, we use the fixed yearly average estimated bid-ask spreads on Dow Jones stocks from Jones (2000). The transaction cost values for each year is reported in Appendix B.[16]

For each stock, we apply our algorithm to search for the optimized rules in the last ten years (five-year training and five-year selection period) and then compute the one-year out-of-sample returns. We do ten simulations for each of fitness function, i.e., Average excess return and Sharpe ratio.

Figure 8 shows the out-of-sample results when maximizing the Sharpe ratio.[17] Figure 8 (a) reports the average excess return and Figure 8 (b) shows the Sharpe ratio of optimized trading rules in the Final Set, averaged over rules, stocks, and simulations, after transaction cost. The average excess returns after transaction costs prior to 1990 is consistently positive, while after 1990, we see a decline in out-of-sample performance. This decline is more apparent post-2000. Furthermore, the out-of-sample Sharpe ratios after transaction costs are consistently positive and higher than 0.5 close to 1990 and higher than zero up to 2005. Overall, the decline in performance in both performance measures is evident post-2000.

[Figure 8 About Here]

[16] In our optimization procedure to search for optimized trading rules, for all the years before 2000, we use a fixed transaction cost of 27 *bps* (we choose 27 bps because it is the average transaction cost for the sample of top 20% of stocks in the year 2000). For each stock from 2000 to 2014, we use a fixed transaction cost equal to annual average quoted spread from TAQ dataset. We fixed the transaction costs in order to reduce the computational expensiveness of the optimization process. In our out-of-sample results, presented in the paper, we use the actual transaction costs as described above in the text. If from 2000 to 2014, the quoted spread is missing from the TAQ dataset for a stock in a particular day, we use the average quoted spread in that year for that stock.

[17] We report the results for average-return maximization in Appendix C. The results follow the same overall pattern through time as in maximizing Sharpe ratio.

In Figure 9, we report the CAPM and Fama and French four-factor alphas (the point estimate and the 95% confidence intervals) in each five-year out-of-sample interval from 1975. There are three deductions. First, our optimization process results in consistent positive alphas prior to 2005. We observe that the optimized rules result in risk-adjusted returns orthogonal to what we could have achieved from following market, size, value, and momentum portfolios. Second, alphas clearly decline over time. For example, following our strategy between 1975 to 1979 would have earned an annualized four-factor alpha of around 18% out-of-sample after transaction costs, significant at 5% level. This reduces to about 1% between 2010 to 2014. These results contradict Bajgrowicz and Scaillet (2012) by showing that in the past an investor could have enjoyed the post-transaction-cost profitability of technical trading rules through time without prior knowledge of rewarding rules.

[Figure 9 About Here]

5. Standard machine learning algorithms

The genetic algorithm is flexible in adopting different objective functions (e.g., in our paper, we use genetic algorithm to maximize out-of-sample alpha and also Sharpe ratio). In addition, the genetic algorithm does a simultaneous search of model structure and model parameters in non-linear settings. Its stochastic nature propels the search for a global optimum, avoiding convergence to local optima. On the other hand, the standard machine learning models, such as KNN or Random Forest, are directed by a loss-minimization objective function (minimizing the prediction error or mean squared error) that limits the creative aspect of the search process. Overall, the accommodating and the stochastic nature of the genetic algorithm in finding optimum solutions gives it an edge in the optimization process. This section compares the

genetic algorithm to the standard machine learning models in the search for profitable trading rules.

We use several machine learning algorithms for detecting past price trends for future trading. These standard machine learning algorithms are the SVM + Logistic (Regularized SVM + Logistic Regression), Decision Tree, KNN, Random Forest, an ensemble of machine learners (Ensemble) and Long Short-Term Memory Network (LSTM Network). Moreover, as a robustness check, we also include a random investment strategy in which we randomly create signals each day out-of-sample.

As the dependent variable, for each day t, we create a zero-one variable that is equal to one if that day's return is positive; and zero otherwise. As the set of features for the Regularized SVM + Logistic Regression, Decision Tree, KNN, Random Forest, and Ensemble methods, we use functions of prices and returns. This feature set includes average return, volatility, skewness, kurtosis and autocorrelation in the last 10, 25, 50 and 100 days (in total, 20 features)[18]. As input to the LSTM network, we use the last 100 days of prices that is similar to the input to the genetic algorithm.

For each out-of-sample year, we use the previous ten years for training the machine learning algorithm (with 10-fold cross-validation). We use 10 NYSE/AMEX volatility decile portfolios from 1965 to 2017. In each year from 1975, we use the previous ten years to train the machine learning algorithms. Using the trained algorithm, we predict the buy and sell signals in the out-of-sample period. This is implemented for each volatility decile portfolio. In computing the out-of-sample returns, we assume a fixed 5 bps transaction cost. We calculate

[18] We also test two other individual feature sets. One is similar to the feature set in the genetic algorithm that is the last 100 days of prices. Another feature set includes the last 100 days of prices besides the functions of prices (average return, volatility, skewness, kurtosis, and autocorrelation in the last 10, 25, 50 and 100 days). The best out-of-sample performance relative to the genetic algorithm is obtained when using only the functions of prices as the feature set.

out-of-sample alpha for each year and each volatility decile portfolio and report the equal-weighted alpha across the portfolios.

We briefly describe the machine learning algorithms. *Linear classification models (SVM + Logistic):* In this method, the algorithm chooses between various linear classification models, i.e., regularized support vector machines (SVM) and logistic regression models. We optimize hyperparameters using ten-fold cross-validation and select the model with the lowest classification error. *Decision tree:* The algorithm fit a binary classification decision tree (a tree-like model of decisions) to the training data. The optimal leaf size for the decision trees is estimated using ten-fold cross-validation. *KNN:* This algorithm fits a k-nearest neighbor classification model to the training data. The number of nearest neighbors is optimized using ten-fold cross-validation. *Random forest:* This algorithm trains decision trees to bootstrapped samples of training data and forecast using a forest of decision trees. The hyper-parameters are optimized using ten-fold cross-validation. *An ensemble of machine learners:* In this machine learning approach, we use the ensemble of classification methods that include boosting besides bagging of classification trees (i.e., AdaBoost, Random Forest). We choose the model with the lowest classification error. The hyper-parameters are optimized using ten-fold cross-validation. *Long Short-Term Memory (LSTM) network:* We use input sequence of size 100 (last 100 days of prices), an LTSM layer with 100 hidden units, a fully connected layer of size 2, followed by a softmax layer and a classification layer (buy and short-selling).

Figure 10 presents the out-of-sample four-factor abnormal returns for the returns net-of-cost. The results follow similar pattern as the genetic algorithm over time. For all of the standard machine learning algorithms, the out-of-sample alphas show promising performance in the early years (the 1970s), and the performance decreases through time. Among the algorithms, the LSTM Network and also SVM + Logistic linear algorithm shows promising performance before 2000 with alphas ranging up to 60% annually net-of-cost. However, the

out-of-sample performance depreciates post 2000 that matches nicely with the findings with the genetic algorithm. Overall, the results across all of the machine learning algorithms point to the increase in market efficiency over time.

[Figure 10 About Here]

Furthermore, the placebo test of random investing result in consistent negative out-of-sample alphas through time. This shows that investing via the machine learning algorithm is not the same as blindly investing following buy and sell signals and furthermore, it indicates that the machine learning algorithms create value over time.

Next, we compare the performance of the standard machine learning algorithms to the genetic algorithm. To do so, we regress the genetic algorithm's out-of-sample returns on the standard machine learning algorithms' returns when solving the same problem: investing on the 10 NYSE/AMEX volatility decile portfolios from 1975 to 2014.

Since we regress the genetic algorithm's out-of-sample return on different standard machine learning algorithm's returns, a positive and significant intercept (generalized alpha) translates into better performance of the genetic algorithm to the standard machine learning algorithm. The results in Table 12 clearly shows the better performance of the genetic algorithm to most of the standard machine learning algorithms. For example, the genetic algorithm earns a generalized alpha of 2.62% (t=3.24) relative to the SVM + Logistic algorithm. Moreover, among the standard machine learning algorithms, LSTM network shows net-of-cost

performance that is comparable to the genetic algorithm (with a generalized alpha of 1.56% (t=1.17))[19].

[Table 12 About Here]

6. Conclusion

Recent literature provides contradictory evidence on the profitability of technical trading rules. Some identify technical trading profitability (e.g., Sullivan, Timmermann, and White, 1999) while others do not (e.g., Bajgrowicz and Scaillet, 2012). A recent study by Bajgrowicz and Scaillet (2012) argues that an investor could have never been able to pick the technical trading rules ex ante that are profitable out-of-sample after transaction costs. In this paper, we challenge this argument with machine learning algorithms that aim to find technical trading rules that are profitable out-of-sample after transaction costs. First, our results show that the technical trading could have been profitable given the availability of computational power. This is evident in our results for early years of applying our machine learning algorithm, and it is in line with Dugast and Foucault (2020)'s theoretical finding that better computing power raises the average quality of predictors and hence of profitability.

Second, we show that the technical trading was profitable. This is demonstrated in our results by positive out-of-sample abnormal returns between 1995 to 2005. Third, we show that the technical trading found by our machine learning algorithm is much less profitable in recent years. This is apparent in our results from the algorithms' poor performance after 2005. This is

[19] One key factor in determining the performance of the standard machine learning methods is the input feature set. The genetic algorithm has the feature selection procedure built-in in its process; while, for the standard machine learning models, we should make sure the right features are chosen. We try to address this concern by selecting functions of prices and returns for the standard machine learning models that are comparable to the input features of the genetic algorithm; however, we should note that there is a chance that we missed on selecting the appropriate features.

also in line with Dugast and Foucault (2020)'s conclusion that higher computing power increases price informativeness and results in lower profitability of predictors.

Our paper also provides a comparison of evolutionary genetic algorithm to the standard loss-minimization machine learning algorithms in finding profitable technical trading signals. The out-of-sample results show that the genetic algorithm's flexibility in setting up the search space, the economic objective and the add-in constraints gives it an edge relative to the standard machine learning algorithms such as decision trees and random forest.

Our paper also adds to the literature on uncovering anomalies. We show that the evolutionary genetic algorithm is useful in searching for time series anomalies. This complements the study by Yan and Zheng (2017), which searches for cross-sectional anomalies based on accounting information. Applying our algorithm to search for time-series four-factor alphas in the NYSE/AMEX volatility-decile portfolios, we show that it is possible to find time-series anomalies that perform better than a benchmark set of moving-average strategies. Our results are also in line with recent evidence on the deterioration of anomalies' performance through time (e.g., Linnainmaa and Roberts, 2018). More generally, this paper concerns the application of big data exploration in financial economics. Big data analysis comes with the potential for data-snooping, *p*-hacking, and data-dredging (See, e.g., Harvey, 2017). We show that big data paired with sophisticated computational techniques can be informative in financial economics.

References

Allen, F., Karjalainen, R., 1999. Using genetic algorithms to find technical trading rules. Journal of Financial Economics 51, 245–271.

Bajgrowicz, P., Scaillet, O., 2012. Technical trading revisited: False discoveries, persistence tests, and transaction costs. Journal of Financial Economics 106, 473–491.

Balduzzi, P., Lynch, A. W., 1999. Transaction costs and predictability: Some utility cost calculations. Journal of Financial Economics 52, 47–78.

Bessembinder, H., Chan K., 1998. Market efficiency and the returns to technical analysis. Financial Management 27, 5–17.

Blume, L., Easley D., O'Hara, M., 1994. Market statistics and technical analysis: The role of volume. Journal of Finance 49, 153–181.

Brock, W., Lakonishok, J., LeBaron, B., 1992. Simple technical trading rules and the stochastic properties of stock returns. Journal of Finance 47, 1731–1764.

Carhart, M. M., 1997. On persistence in mutual fund performance. Journal of Finance 52, 57–82.

Chen, L., Pelger, M, Zhu, J., 2019. Deep learning in asset pricing. Working Paper.

Chordia, T., Subrahmanyam, A., Tong, Q., 2014. Have capital market anomalies attenuated in the recent era of high liquidity and trading activity? Journal of Accounting and Economics 58, 41–58.

Dugast, J., and Foucault, T., 2020. Equilibrium data mining and data abundance. Working Paper.

Fama, E., Blume, M., 1966. Filter rules and stock-market trading. Journal of Business 39, 226–241.

Fama, Eugene F, French, K. R., 1993. Common risk factors in the returns on stocks and bonds. Journal of Financial Economics 33, 3–56.

Fama, E. F., French, K. R., 2015. A five-factor asset pricing model. Journal of Financial Economics 116, 1-22.

Grundy, B. D., McNichols, M., 1989. Trade and the revelation of information through prices and direct disclosure. Review of Financial Studies 2, 495–526.

Gu, S., Kelly, B., Xiu, D., 2020. Empirical asset pricing via machine learning. Review of Financial Studies 33, 2223–2273.

Han, Y., 2006. Asset allocation with a high dimensional latent factor stochastic volatility model. Review of Financial Studies 19, 237–271.

Han, Y., Yang, K., Zhou, G. 2013. A new anomaly: The cross-sectional profitability of

technical analysis. Journal of Financial and Quantitative Analysis 48, 1433–1461.

Han, Y., Zhou, G., and Zhu, Y., 2016. A trend factor: Any economic gains from using information over investment horizons? Journal of Financial Economics 122, 352–375.

Harvey, C. R., 2017. Presidential address: The scientific outlook in financial economics, Journal of Finance 72, 1399–1440.

Harvey, C. R, Liu, Y., Zhu, H., 2016. … and the cross-section of expected returns, Review of Financial Studies 29, 5–68.

Holland, J. H., 1962. Outline for a logical theory of adaptive systems. Journal of the ACM 9, 297–314.

Holland, J. H., 1975. Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Arbor.

Jegadeesh, N., Titman, S., 1993. Returns to buying winners and selling losers: Implications for stock market efficiency. Journal of Finance 48, 65–91.

Jones, C. M., 2002,.A century of stock market liquidity and trading costs. Working Paper.

Kavajecz, K., Odders-White, E., 2004. Technical analysis and liquidity provision. Review of Financial Studies 17, 1043–1071.

Linnainmaa, J. T., and Roberts, M. R., 2018. The history of the cross section of stock returns. Review of Financial Studies 31, 2606–2649.

Lo, A. W., Mamaysky, H., Wang, J., 2000. Foundations of technical analysis: Computational algorithms, statistical inference, and empirical implementation. Journal of Finance 55, 1705–1765.

Lynch, A. W, Balduzzi, P., 2000. Predictability and transaction costs: The impact on rebalancing rules and behavior. Journal of Finance 55, 2285–2309.

Markoff, John, 29 August 1990. Business technology; What's the Best Answer? It's Survival of the Fittest. New York Times.

McLean, R. D. Pontiff, J., 2016. Does academic research destroy stock return predictability? Journal of Finance 71, 5–32.

Neely, C., Weller, P., Dittmar, R., 1997. Is technical analysis in the foreign exchange market profitable? A genetic programming approach. Journal of Financial and Quantitative Analysis 32, 405–426.

Neftci, S. N, 1991. Naive trading rules in financial markets and wiener-kolmogorov prediction theory: A study of “technical analysis.” Journal of Business 64, 549–571.

Newey, W. K., and West, K. D., 1987. Hypothesis testing with efficient method of moments estimation. International Economic Review 28, 777–787.

Nordhaus, W., 2015. Are we approaching an economic singularity? Information technology

and the future of economic growth. Working Paper.

Potvin, J.,, Soriano, P., and Vallee, M., 2004. Generating trading rules on the stock markets with genetic programming, Computers & Operations Research 31, 1033–1047.

Ready, M., 2002. Profits from technical trading rules. Financial Management 31, 43–61.

Rossi, A. G., 2018. Predicting stock market returns with machine learning. Working Paper.

Sadka, R., Scherbina A.,, 2007. Analyst disagreement, mispricing, and liquidity. Journal of Finance 62, 2367–2403.

Sullivan, R., Timmermann, A., White, H., 1999. Data-snooping, technical trading rule performance, and the bootstrap. Journal of Finance 54, 1647–1691.

Yan, X., and Zheng, L., 2017. Fundamental analysis and the cross-section of stock returns: A data-mining approach. Review of Financial Studies 30, 1382–1423.

White, H., 2000. A reality check for data snooping. Econometrica 68, 1097–1126.

Zhang, X. F., 2006. Information Uncertainty and Stock Returns. Journal of Finance, 61, 105–137.

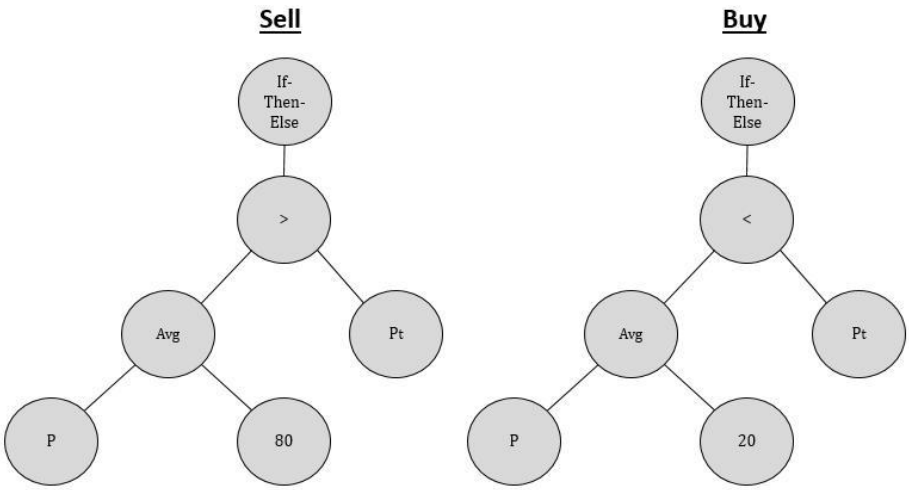


Figure 1. Trading rule example
This figure presents an example of a trading rule. In this case, we buy the asset when the price today is higher than the average of prices over the last 20 days (right branch: *Buy*) and short the asset when the price is lower than the average of prices over the last 80 days (left branch: *Sell*). We hold the risk-free asset when both or none of the conditions are satisfied.

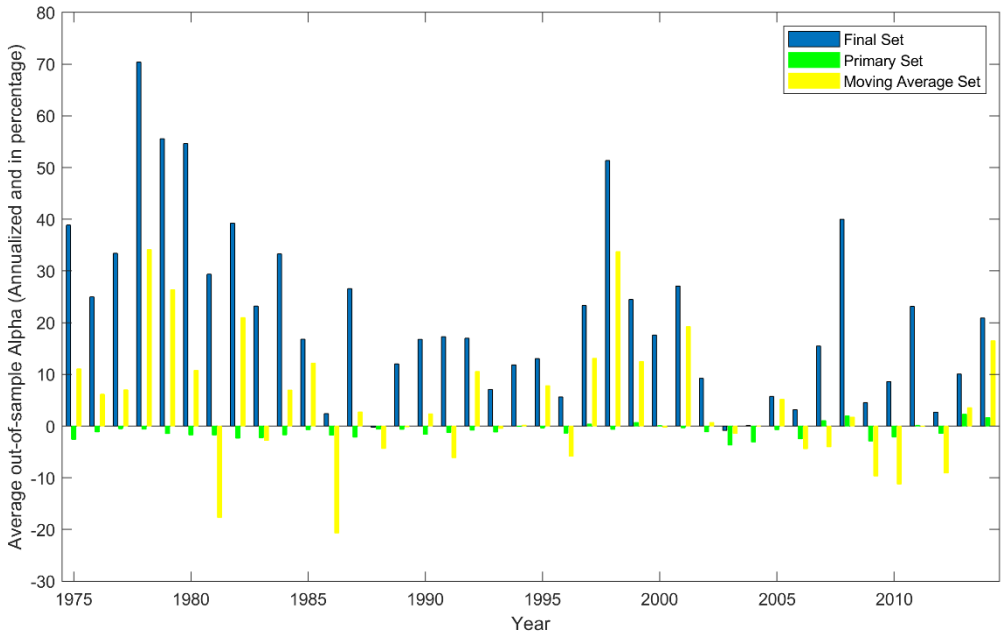


Figure 2. Performance through time

This figure reports the out-of-sample Fama and French four-factor annualized alphas in percentages through time. Primary Set is a randomly generated set of technical trading rules. Final Set is the set of optimized trading rules after applying the optimization procedure. Moving Average Set is the set of moving average strategies using lags of 3 to 100 days. The alpha values are computed on all of the NYSE/AMEX volatility-decile portfolios between January 1, 1965 and December 31, 2014. We use a rolling procedure with five-year training period, five-year selection period, and one-year out-of-sample period. We run the optimization procedure for 20 simulations. In each out-of-sample year, we compute the average alphas over the rules, volatility deciles, and simulations.

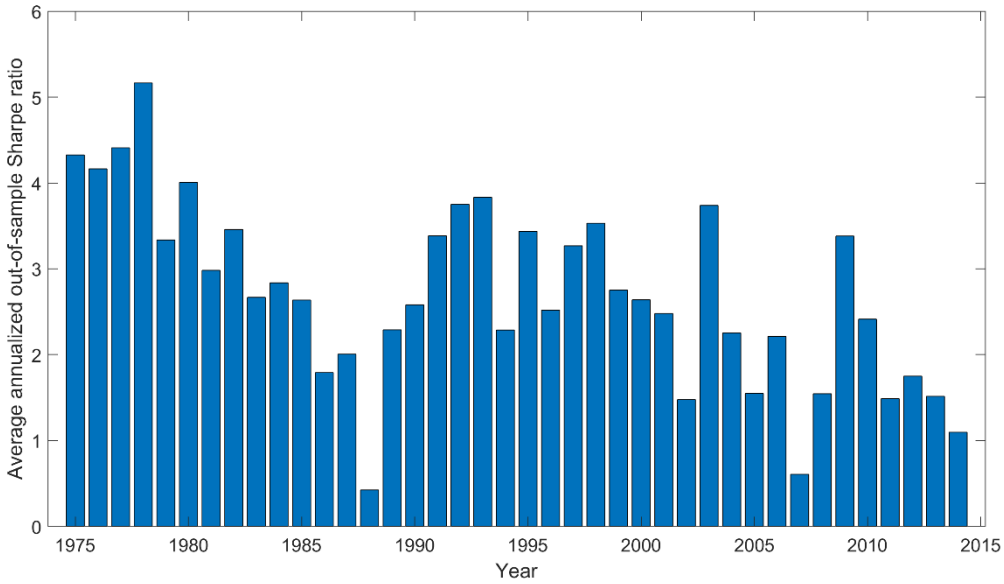


Figure 3. Annualized out-of-sample Sharpe ratio through time

This figure reports the out-of-sample average Sharpe ratios through time. Final Set is the set of optimized trading rules after applying optimization procedure to each of the NYSE/AMEX volatility-decile portfolios between January 1, 1965 to December 31, 2014. We use a rolling procedure with five-year training period, five-year selection period, and one-year out-of-sample period. We run the optimization procedure for 20 simulations. In each out-of-sample year, we compute an average of Sharpe ratio values over the rules, volatility deciles, and simulations.

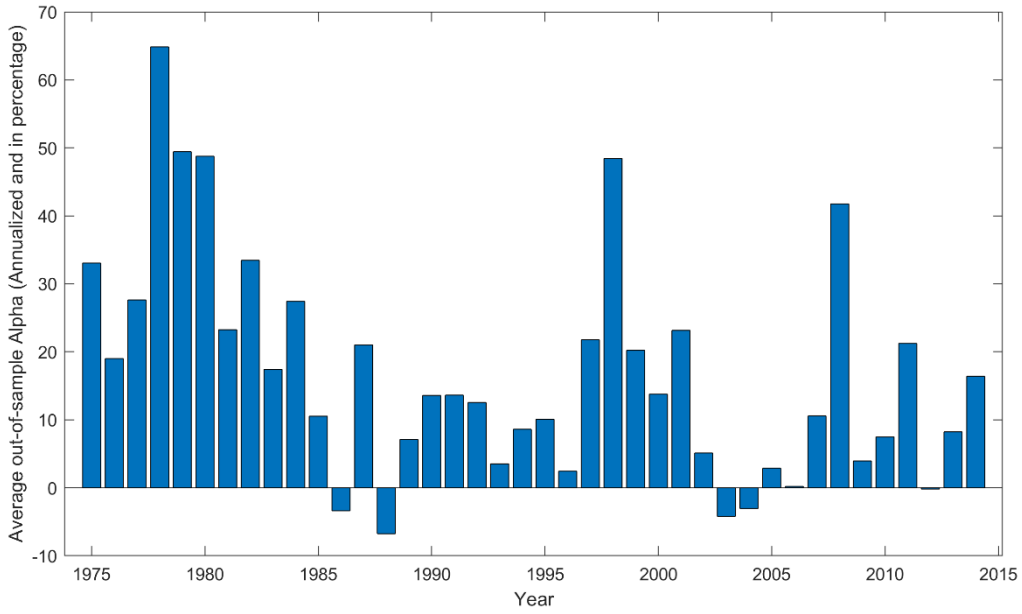


Figure 4. Performance and transaction cost

This Figure presents four-factor Fama-French annualized alphas through time with a 5 *bps* transaction cost. In each out-of-sample year, we search for a set of optimized trading rules by applying our algorithm to each of the NYSE/AMEX volatility--decile portfolios between January 1, 1965 and December 31, 2014. We use a rolling procedure with five-year training period, five-year selection period, and one-year out-of-sample period. In applying each trading rule, as we move between a long position in the asset, holding the risk-free asset, and a short position in the asset, we subtract a transaction cost of 5 *bps*. We run the optimization procedure for 20 simulations. In each out-of-sample year, we compute the average alphas over the rules, volatility deciles, and simulations.

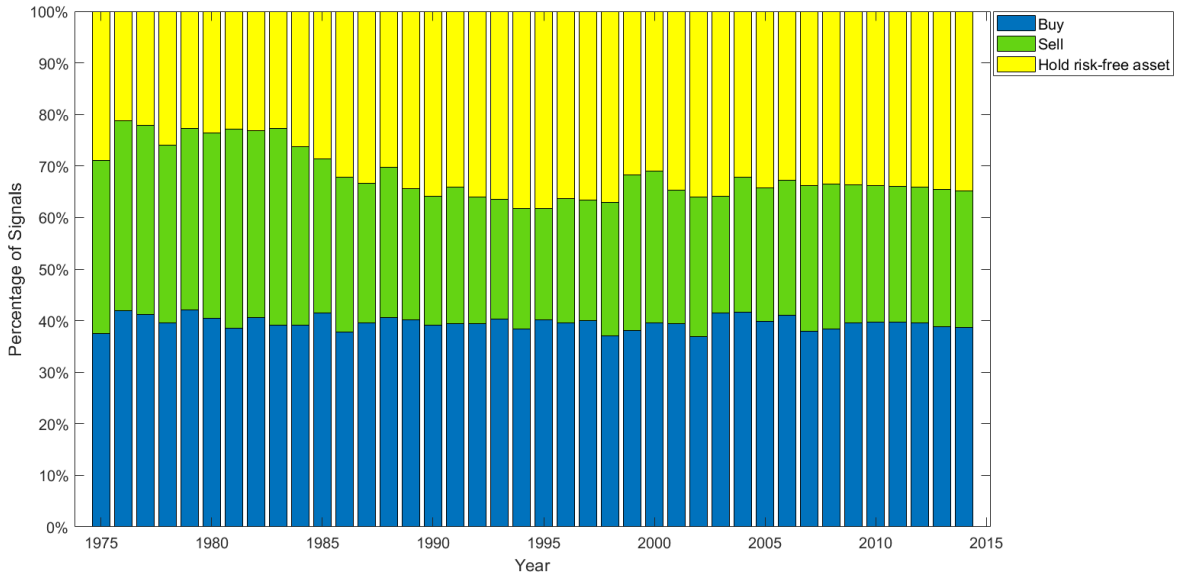


Figure 5. Trading signals

This graph shows the percentage of buy, sell, and hold risk-free signals across the optimized trading rules in the Final Set. In each out-of-sample year, we search for a set of optimized trading rules, Final Set, by applying our algorithm to each of the NYSE/AMEX volatility decile portfolios between January 1, 1965 and December 31, 2014. We use a rolling procedure with five-year training period, five-year selection period, and one-year out-of-sample period. We run the optimization procedure for 20 simulations. In each out-of-sample year, we compute the percentage of buy, sell, and hold risk-free signals averaged over the rules, assets, and simulations.

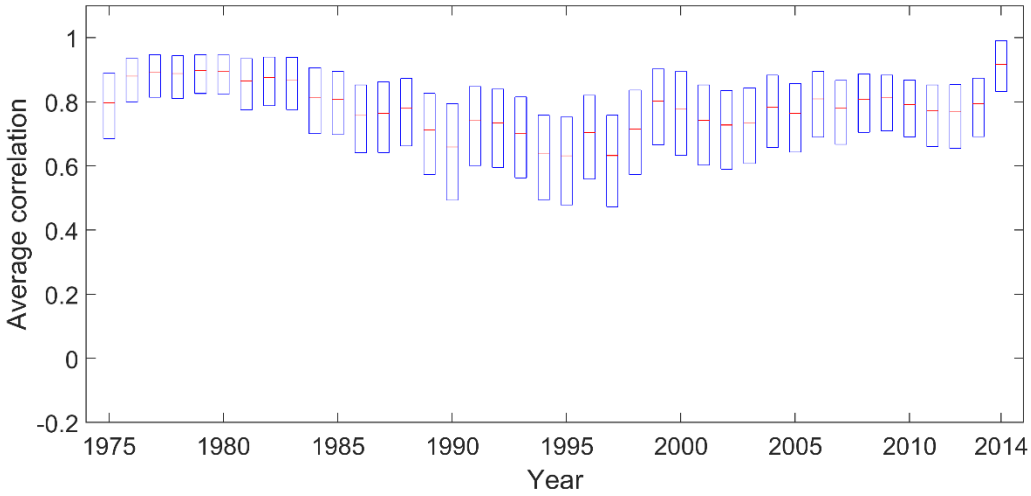


Figure 6. Average correlation across trading rules through time

This figure shows the average out-of-sample return correlations among trading rules in the Final Set in each out-of-sample year from 1975 to 2014. The Final Set is the set of optimized trading rules after applying optimization procedure to each of the NYSE/AMEX volatility-decile portfolios between January 1, 1965 and December 31, 2014. We use a rolling procedure with five-year training period, five-year selection period, and one-year out-of-sample period. We run the optimization procedures for 20 simulations. The correlation values between trading rule returns are pooled together over assets and simulations in each out-of-sample year. The box plot contains the average value in red line, first quartile as lower tail of box, and third quartile as upper tail of the box.

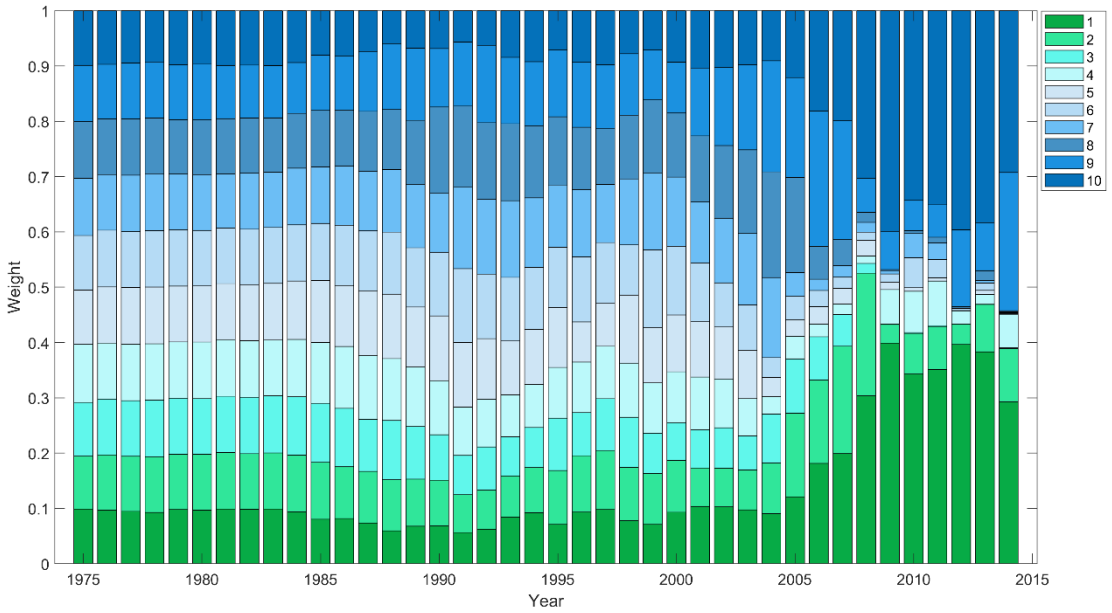
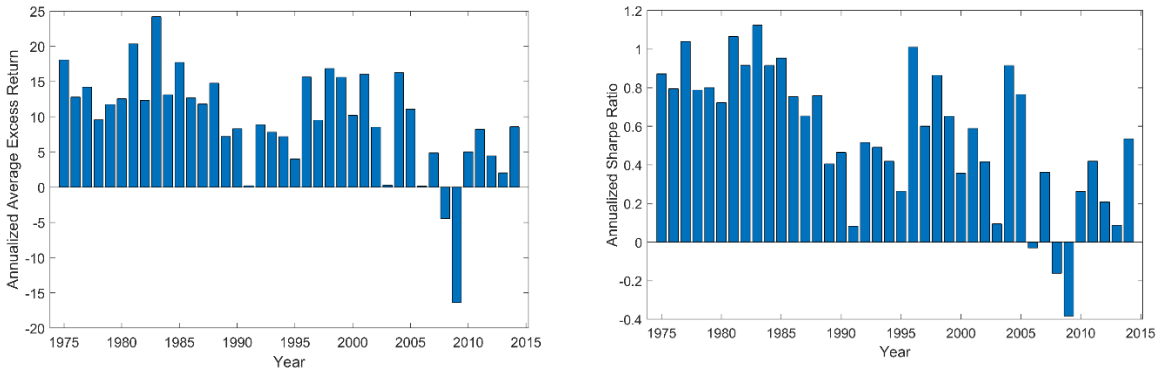


Figure 7. Weight allocation of volatility deciles through time

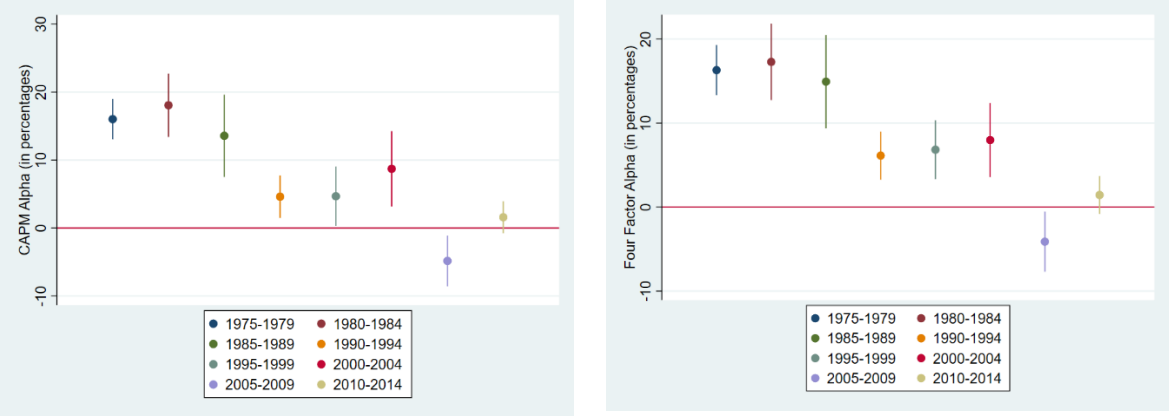
This figure reports the weights invested in each decile in a diversified portfolio of volatility deciles through time. We compute the weights as the fraction of the number of rules in the Final Set for each volatility decile to the total number of optimized rules across the volatility deciles in each out-of-sample year. The Final Set is the set of optimized trading rules after applying optimization procedure to each of the NYSE/AMEX volatility-decile portfolios between January 1, 1965 and December 31, 2014. We use a rolling procedure with five-year training period, five-year selection period, and one-year out-of-sample period. We run the optimization procedures for 20 simulations. The portfolio allocation weights are averaged over the rules and simulations.



(a) Average Excess Return (b) Sharpe Ratio

Figure 8. Performance through time: individual stocks

This figure reports the (a) average excess return and (b) Sharpe ratio in one-year out-of-sample periods for the set of optimized trading rules across 100 highly capitalized stocks from NYSE/AMEX/NYSE from 1975 to 2014. For each out-of-sample year and each stock, we use the last ten years (five-year training period and five-year selection period) to search for optimized trading rules. The fitness value is set as the Sharpe ratio if the *t*-statistics of a null hypothesis of the average Sharpe ratio is higher than 2 (we do this test by bootstrapping each rule's daily returns); otherwise, the fitness value is a large negative number (-M). We repeat the optimization procedure for each stock for 10 simulations. The results are averaged over stocks, trading rules, and simulations. We take the transaction costs into account by using the yearly fixed estimates from Jones (2000) prior to 2000 and actual daily quoted bid-ask spread from the TAQ dataset from 2000 to 2014.



(a) CAPM Alpha

(b) FF4 Alpha

Figure 9. Alphas through time: individual stocks

This figure reports the (a) CAPM alpha and (b) four-factor alpha for five-year out-of-sample periods for the set of optimized trading rules across 100 highly capitalized stocks from NYSE/AMEX/NYSE from 1975 to 2014. For each out-of-sample year and each stock, we use the last ten years (five-year training period and five-year selection period) to search for optimized trading rules. The fitness value is set as the Sharpe ratio if the *t*-statistic of a null hypothesis of the average Sharpe ratio is higher than 2 (we do this test by bootstrapping each rule's daily returns); otherwise, the fitness value is -M (M being a large value). We take the transaction costs into account by using the yearly fixed estimates from Jones (2000) prior to 2000 and actual daily quoted spread from the TAQ dataset from 2000 to 2014. We repeat the optimization procedure for each stock for 10 simulations. For each of the five-year out-of-sample periods, we compute the daily post–transaction cost average excess return across stocks, rules, and simulations, and compute the annual CAPM and Four-factor alphas. The plot shows the point estimates and the 95% confidence intervals.

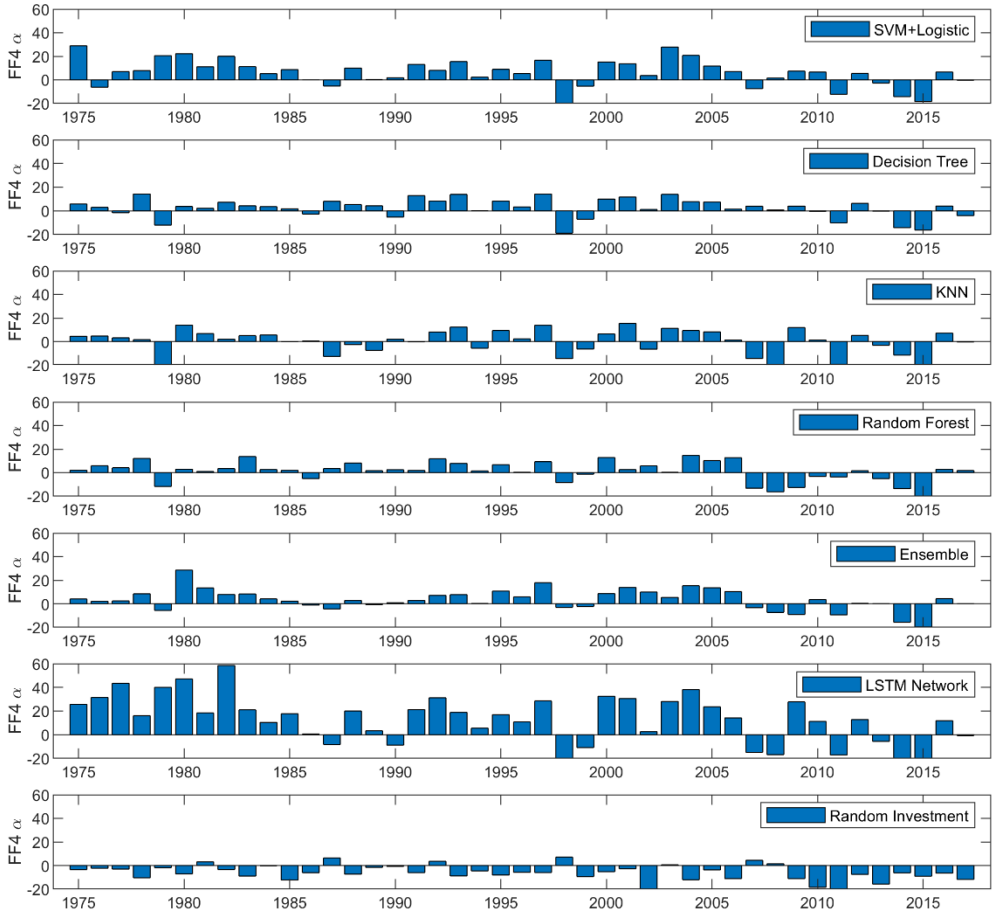


Figure 10. Out-of-sample Fama and French four-factor alpha

This figure reports the Fama and French four-factor alphas for five machine learning algorithms and a random investment strategy. The machine learning algorithms are SVM+ Logistic, Decision Tree, KNN, Random Forest, Ensemble Methods and and LSTM Network. In the Random Investment strategy, we randomly assign a signal in each out-of-sample year. We use a 10-year training period and a one-year out-of-sample period. The results correspond to the out-of-sample period. As the set of features for the Regularized SVM + Logistic Regression, Decision Tree, KNN, Random Forest, and Ensemble, we use functions of prices and returns. This feature set includes average return, volatility, skewness, kurtosis and autocorrelation in the last 10, 25, 50 and 100 days (in total, 20 additional features). As input to the LSTM network, we use the last 100 days of prices, similar to the input to the genetic algorithm. We use 10 NYSE/AMEX volatility decile portfolios from 1965 to 2017. We use 10-fold cross-validation to train the machine learning algorithms, assume a fixed transaction cost of 5 *bps, and* use our algorithm output signals to engage in buying and short-selling the underlying assets. We apply each machine learning algorithm to each volatility decile portfolio and compute the out-of-sample alphas as an equal-weighted portfolio of alphas across volatility decile portfolios.

Table 1. Trading rule representation

This table presents the variables and functions employed in each of the four levels of a trading-rule tree. In level 1, we consider Boolean operator and functions and, in level 2, the relational operators. Real functions are assigned to level 3, where s stands for input variables. In level 4, we have two sets of inputs: the real variables and the terminals that indicate the type of input variables.

Levels	Operator
Level 1:	
Boolean operator and functions	If-Then-Else
	and
	or
Level 2:	
Relational operator	$<$
	$>$
Level 3:	
Real Functions	$Avg\ (s, days)$
	$Max\ (s, days)$
	$Min\ (s, days)$
	$Median\ (s, days)$
	$Lag\ (s, days)$
	$Volatility\ (s, days)$
	$RSI\ (s, days)$
	$Filter(s, Pr, days)$
Level 4:	
Real variables	$P: Price$
	$R: Return$
Terminals	Pr: Random number in [-1, 1]
	$Days$: Number of days before today
	P_t: Price of the current day
	R_t: Return of the current day

Table 2. The optimization procedure

This table presents the algorithm designed to search for profitable technical trading rules. |POP| is the size of the population, and |GEN| is the number of generations that the algorithm is run.

Step 1
Generate a random population with |POP| individual trading rules (**Primary Set**)
Compute the fitness value of rules in the *training period*
Save the set of rules as the *Current Population*

Step 2
Compute the fitness value of the rules in the *Current Population* in the *selection period*
Save them as the initial best rules in the **Final Set**

Step 3
Crossover
Select two random rules from the *Current Population* at random and apply crossover operator
(Generate |POP| new rules)
Mutation
Select one random rule from the *Current Population* and also generate a new rule; apply crossover operator
(Generate |POP|/2 rules)

Generate random individual rules (Generate |POP|/2 rules)
Calculate the fitness of the *new rules* in the *training period* and add them to the *Current Population*.
Sort all the trading rules based on the fitness values in the *training period* and select the |POP| fittest rules as the *Current Population*

Step 4
Compute the fitness value of the new rules in the *Current Population* in Step 3 in the *selection period*
Add them to the best rules in the Final Set
Sort all the trading rules in the Final Set based on the fitness values in the *selection period* and select the |POP| fittest rules as the new best rules in the Final Set

Stop if |GEN| generations has passed
Back to step 3

Table 3. Summary statistics

This table reports the summary statistics for out-of-sample Fama and French four-factor annualized alphas in percentages. Primary Set is a randomly generated set of technical trading rules. Final Set is the set of optimized trading rules after applying optimization procedure. Moving Average Set is the set of moving average strategies using lags of 3 to 100 days. The alpha values are computed on all of the NYSE/AMEX volatility-decile portfolios between January 1, 1965 and December 31, 2014. We use a rolling procedure with five-year training period, five-year selection period, and one-year out-of-sample period. We run the optimization procedure for 20 simulations. A time-series of alpha values is calculated by averaging alphas across the rules and simulations in each set. We report N (average number of unique trading rules in each rolling period and simulation), Mean (average value), SD (standard deviation), Skew (skewness), Kurt (kurtosis), 25% (first quartile), and Median, 75% (third quartile).

	Sets	N	Mean	SD	Skew	Kurt	25%	Median	75%
Low Volatility	Moving Average	98	4.76	14.12	1.44	5.82	-3.08	2.15	12.53
	Primary	500	-1.26	13.97	0.43	4.86	-13.18	-5.32	2.35
	Final	332	14.27	15.21	1.14	4.27	2.28	9.26	23.17
2	Moving Average	98	2.49	13.34	0.67	3.67	-5.17	2.22	12.08
	Primary	500	-1.34	13.68	0.08	4.83	-12.95	-4.77	2.13
	Final	304	16.52	15.39	0.91	3.17	4.23	11.33	22.36
3	Moving Average	98	1.09	15.51	0.73	3.88	-7.59	0.68	11.32
	Primary	500	-1.25	13.79	-0.11	4.73	-12.62	-4.43	2.65
	Final	271	20.27	18.31	0.93	3.50	5.39	14.36	29.06
4	Moving Average	98	0.33	18.83	0.10	5.60	-8.85	0.67	12.89
	Primary	500	-1.33	14.99	-0.26	5.08	-13.31	-4.62	3.07
	Final	285	23.86	19.64	0.65	3.13	7.20	19.71	34.90
5	Moving Average	98	0.70	21.95	0.26	4.16	-9.94	0.81	14.97
	Primary	500	-1.12	16.06	-0.09	5.00	-13.33	-4.06	4.34
	Final	292	29.35	21.57	0.48	2.85	11.63	25.02	40.79
6	Moving Average	98	1.86	23.79	0.27	3.56	-10.91	2.40	17.69
	Primary	500	-0.95	16.87	-0.11	5.14	-12.86	-3.42	5.42
	Final	302	29.82	20.92	0.33	3.10	14.10	27.38	41.11
7	Moving Average	98	3.26	24.57	0.52	4.00	-11.39	2.70	19.63
	Primary	500	-0.82	18.19	-0.09	5.46	-12.88	-3.03	6.61
	Final	323	33.54	25.01	0.58	3.84	14.01	29.78	47.20
8	Moving Average	98	7.79	25.79	0.54	3.76	-7.44	6.56	25.89
	Primary	500	-0.07	19.27	-0.15	4.95	-10.94	-0.01	10.94
	Final	338	33.33	22.95	0.59	3.85	16.85	30.91	43.69
9	Moving Average	98	11.3	27.7	0.5	3.7	-4.7	9.5	30.4
	Primary	500	-0.3	22.3	0.1	4.7	-13.6	-0.9	11.6
	Final	361	32.4	27.4	0.4	3.0	10.7	30.7	47.5
High Volatility	Moving Average	98	8.35	32.88	0.22	3.13	-10.08	8.65	30.61
	Primary	500	-4.92	37.68	-0.48	4.36	-37.95	-15.56	4.50
	Final	304	22.98	36.95	0.29	3.13	-3.22	20.46	44.51

Table 4. Summary statistics — Diversifying across assets

This table reports the summary statistics of Fama and French four-factor annualized alphas for a diversified portfolio of technical trading rules. Primary Set contains a randomly generated set of technical trading rules. Final Set is the set of optimized trading rules after applying optimization procedure. Moving Average Set is the set of moving average strategies using lags of 3 to 100 days. The alpha values are computed on all of the NYSE/AMEX volatility-decile portfolios between January 1, 1965 and December 31, 2014. We use a rolling procedure with five-year training period, five-year selection period, and one-year out-of-sample period. We run the optimization procedure for 20 simulations. The time-series of alpha values is calculated by averaging alphas across the rules, volatility deciles, and simulations in each out-of-sample year. We report Mean (average value), SD (standard deviation), Skew (skewness), Kurt (kurtosis), 25% (first quartile), and Median, 75% (third quartile).

Sets	Mean	SD	Skew	Kurt	25%	Median	75%
Moving-Average	4.2	23.1	0.6	4.7	-7.5	3.3	17.6
Primary	-1.3	20.4	-0.8	9.0	-14.3	-4.2	5.2
Final	25.8	24.5	0.6	4.1	7.1	22.1	38.8

Table 5. Regression of alphas and time trend

This table reports the estimation results for an OLS regression of average four-factor annualized alphas of trading strategies in out-of-sample years on a time trend variable, $year_t$. The time trend variable starts from one in 1975 and increases by one unit each year until 2014. The Final Set is a set of optimized trading rules that comes from applying the optimization procedure to each of the NYSE/AMEX volatility-decile portfolios between January 1, 1965 and December 31, 2014. We use a rolling procedure with five-year training period, five-year selection period, and one-year out-of-sample period. We run the optimization procedure for 20 simulations. For the diversified portfolio across assets, the time-series of alpha values is calculated by averaging alphas across the rules, volatility deciles, and simulations in each out-of-sample year. For each individual volatility decile, the time-series of alpha values is calculated by averaging alphas across the rules and simulations in each out-of-sample year. The values in parentheses are *t*-statistics.

	Intercept	$year_t$	R^2
Diversifying across assets	36.15	-0.63	0.177
	(6.98)	(-2.86)	
Low Volatility	15.64	-0.16	0.016
	(3.35)	(-0.79)	
2	27.09	-0.70	0.300
	(6.64)	(-4.03)	
3	35.43	-1.11	0.531
	(8.90)	(-6.56)	
4	41.59	-1.27	0.550
	(9.49)	(-6.81)	
5	48.46	-1.41	0.530
	(9.59)	(-6.55)	
6	48.26	-1.31	0.480
	(9.26)	(-5.92)	
7	54.21	-1.43	0.408
	(8.21)	(-5.11)	
8	53.68	-1.35	0.275
	(9.09)	(-5.38)	
9	53.30	-1.18	0.275
	(7.30)	(-3.79)	
High Volatility	12.21	0.20	0.005
	(1.06)	(-0.42)	

Table 6. Summary statistics — Trading signals

This table reports the summary statistics for returns conditioned on the type of trading signals (Buy, Hold risk-free, Sell) in the Final set. In each out-of-sample year, we search for a set of optimized trading rules, Final Set, by applying our algorithm to each of the NYSE/AMEX volatility deciles between January 1, 1965 and December 31, 2014. We use a rolling procedure with five-year training period, five-year selection period, and one-year out-of-sample period. We run the optimization procedure for 20 simulations. *N* for Long (Hold risk-free, Short) stands for the number of times we are either buying (holding risk-free asset, short-selling) the asset or already bought (held risk-free asset, short-sold) the asset. We also report % of N (percentage of each signal), Avg Ret (Average return), Avg Std (Average standard deviation), and Avg Skew (Average skewness).

		N	% of N	Avg Ret	Avg Std	Avg Skew
		101.998		0.113	0.003	0.325
Low Volatility	Long	53.412	52%	0.168	0.003	-0.084
	Hold risk-free	27.215	27%	0.023	0.000	-0.129
	Short	21.371	21%	0.130	0.004	0.249
	All	103.132		0.123	0.004	0.293
2	Long	54.351	53%	0.174	0.004	-0.030
	Hold risk-free	25.474	25%	0.023	0.000	-0.151
	Short	23.307	23%	0.155	0.005	0.200
	All	102.884		0.123	0.004	0.295
3	Long	54.192	53%	0.174	0.004	-0.028
	Hold risk-free	25.402	25%	0.023	0.000	-0.152
	Short	23.290	23%	0.156	0.005	0.201
	All	102.679		0.124	0.004	0.267
4	Long	51.927	51%	0.176	0.004	-0.019
	Hold risk-free	26.251	26%	0.023	0.000	-0.183
	Short	24.501	24%	0.155	0.005	0.149
	All	102.583		0.125	0.004	0.270
5	Long	51.952	51%	0.177	0.004	-0.018
	Hold risk-free	26.177	26%	0.023	0.000	-0.181
	Short	24.454	24%	0.156	0.005	0.152
	All	102.814		0.126	0.004	0.271
6	Long	52.009	51%	0.178	0.004	-0.016
	Hold risk-free	26.143	26%	0.023	0.000	-0.185
	Short	24.663	24%	0.158	0.005	0.151
	All	102.799		0.127	0.004	0.269
7	Long	52.050	51%	0.179	0.004	-0.016
	Hold risk-free	26.085	26%	0.023	0.000	-0.181
	Short	24.664	24%	0.158	0.005	0.151
	All	102.762		0.126	0.004	0.266
8	Long	51.980	51%	0.177	0.004	-0.012
	Hold risk-free	26.169	26%	0.023	0.000	-0.182
	Short	24.613	24%	0.158	0.005	0.139
	All	110.459		0.235	0.008	0.750
9	Long	45.627	45%	0.316	0.009	1.113
	Hold risk-free	26.802	26%	0.023	0.000	-0.377
	Short	38.029	37%	0.315	0.010	-0.271
	All	93.472		0.245	0.009	2.548
High Volatility	Long	36.852	36%	0.485	0.012	2.105
	Hold risk-free	28.651	28%	0.023	0.000	-0.159
	Short	27.968	27%	0.197	0.008	0.295
	All	102.558		0.147	0.005	0.539
Diversifying across assets	Long	50.435	49%	0.219	0.005	0.290
	Hold risk-free	26.437	26%	0.023	0.000	-0.190
	Short	25.686	25%	0.176	0.006	0.132

Table 7. Characteristics of trading rules

This table reports the number of real functions and types of input variables in the Final Set. Final Set is the set of optimized trading rules after applying optimization procedure with a rolling procedure of five-year training period and five-year selection period. We run the optimization procedure for 20 simulations on the highest and lowest NYSE/AMEX volatility decile portfolios from January 1, 1965 to December 31, 2014. All the values are averaged over the out-of-sample years and simulations.

Panel A: Real Functions

		Lowest Volatility Decile			Highest Volatility Decile		
	Set	Buy	Sell	Total	Buy	Sell	Total
Average	Primary	93.10	93.65	186.75	93.33	94.43	187.75
	Final	139.03	164.23	303.25	147.38	129.58	276.95
Minimum	Primary	95.75	93.65	189.40	94.63	93.10	187.73
	Final	201.08	155.18	356.25	140.48	132.98	273.45
Maximum	Primary	92.45	95.18	187.63	92.55	92.03	184.58
	Final	78.40	53.05	131.45	84.95	55.70	140.65
Median	Primary	94.05	93.88	187.93	91.15	92.80	183.95
	Final	135.70	156.40	292.10	137.83	156.28	294.10
Lag	Primary	94.28	92.78	187.05	91.75	94.15	185.90
	Final	54.05	44.45	98.50	87.13	65.93	153.05
Volatility	Primary	94.23	93.50	187.73	95.60	92.43	188.03
	Final	66.00	34.78	100.78	73.73	39.00	112.73
RSI	Primary	92.65	95.73	188.38	96.88	93.83	190.70
	Final	153.10	132.35	285.45	148.85	136.50	285.35
Filter	Primary	94.33	93.93	188.25	94.70	92.90	187.60
	Final	130.10	104.03	234.13	120.23	98.98	219.20

Panel B: Real Variables

		Lowest Volatility Decile			Highest Volatility Decile		
	Set	Buy	Sell	Total	Buy	Sell	Total
Price	Primary	500.18	501.63	1001.80	503.83	498.08	1001.90
	Final	1192.73	1119.43	1310.35	1167.58	1046.73	1212.40
Return	Primary	250.65	250.65	501.30	246.75	247.58	494.33
	Final	515.55	477.30	491.55	523.55	513.85	543.08

Table 8. Return correlation

This table reports the summary statistics for the out-of-sample return correlations among trading rules in the Final set. The Final Set is the set of optimized trading rules after applying optimization procedure to each of the NYSE/AMEX volatility-decile portfolios between January 1, 1965 and December 31, 2014. We use a rolling procedure with five-year training period, five-year selection period, and one-year out-of-sample period. We run the optimization procedures for 20 simulations. The correlation values are pooled together over years and simulations. We report Mean (average value), SD (standard deviation), Skew (skewness), Kurt (kurtosis), 25% (first quartile), and Median, 75% (third quartile).

	Mean	SD	Skew	Kurt	25%	Median	75%
Low Volatility	0.73	0.14	-0.72	3.88	0.65	0.75	0.83
2	0.73	0.15	-0.84	4.23	0.65	0.75	0.84
3	0.69	0.18	-0.72	3.76	0.58	0.71	0.81
4	0.61	0.21	-0.52	3.87	0.49	0.60	0.75
5	0.65	0.21	-0.62	3.45	0.52	0.67	0.80
6	0.64	0.20	-0.54	3.43	0.52	0.65	0.78
7	0.66	0.19	-0.62	3.69	0.55	0.68	0.79
8	0.65	0.19	-0.71	3.90	0.54	0.66	0.78
9	0.69	0.18	-0.84	4.04	0.57	0.71	0.83
High Volatility	0.71	0.18	-1.03	4.32	0.60	0.74	0.84
Diversifying across assets	0.77	0.19	-1.17	4.87	0.61	0.75	0.86

Table 9. Summary statistics — Placebo test

This table reports the out-of-sample Fama and French four-factor annualized alphas in percentages for optimized trading rules in the randomly scrambled data. The datasets are the NYSE/AMEX volatility-decile portfolios from July 1, 1965 to December 31, 2014. In each volatility decile, we assign a random number to each time period and sort the data by the random number. Our optimization algorithm is applied to the scrambled data using a five-year training, a five-year selection period and one-year out-of-sample period. We run the optimization procedures for 20 simulations. Alpha values are averaged over the simulations and also the out-of-sample years. *N* is the average number of unique trading rules found in each rolling period and each simulation. We also report Mean (average value), SD (standard deviation), Skew (skewness), Kurt (kurtosis), 25% (first quartile), and Median, 75% (third quartile).

	N	Mean	SD	Skew	Kurt	25%	Median	75%
Low Volatility	333	-3.45	9.70	0.25	5.48	-9.28	-3.32	2.20
2	304	-6.34	17.57	0.45	4.08	-17.46	-8.19	4.67
3	271	-7.90	26.15	-0.13	3.20	-24.90	-8.09	9.83
4	285	-8.48	23.10	-0.53	3.44	-23.46	-6.02	7.87
5	292	-11.38	27.15	0.36	3.10	-30.12	-13.88	3.95
6	303	-10.16	34.01	0.83	5.01	-31.61	-11.82	6.64
7	323	-26.63	30.50	-0.28	3.27	-45.98	-26.52	-3.97
8	338	-22.66	41.98	-1.44	8.72	-39.61	-19.30	0.23
9	361	-9.96	34.50	0.38	3.27	-33.87	-11.41	11.64
High Volatility	305	-47.43	43.20	-0.54	3.93	-71.45	-43.97	-18.98

Table 10. Other factor models

This table reports the summary statistics of out-of-sample annualized alphas in percentage for three other asset pricing models: CAPM (with market portfolio) in Panel A, Fama and French three-factor model (with market, size, and value) in Panel B and Fama and French five-factor model (with market, size, value, investment, and profitability) in Panel C. The set of trading rules are the same as in Section 2. Primary Set is a randomly generated set of technical trading rules. Final Set is the set of optimized trading rules after applying optimization procedure. Moving-Average Set is the set of moving average strategies using lags of 3 to 100 days. The alpha values are computed on all of the NYSE/AMEX volatility-decile portfolios between January 1, 1965 and December 31, 2014. We use a rolling procedure with five-year training period, five-year selection period, and one-year out-of-sample period. We run the optimization procedure for 20 simulations. A time-series of alpha values is calculated by averaging alphas across the rules and the simulations in each set. We report Mean (average value), SD (standard deviation), Skew (skewness), Kurt (kurtosis), 25% (first quartile), and Median, 75% (third quartile).

Panel A: CAPM							
Sets	Mean	SD	Skew	Kurt	25%	Median	75%
Moving-Average	7.28	23.96	1.25	6.72	-5.75	4.23	19.42
Primary	-1.19	22.57	-0.32	7.40	-15.71	-4.00	7.24
Final	23.59	25.46	1.07	5.60	6.24	20.23	35.15
Panel B: Three Factor							
Sets	Mean	SD	Skew	Kurt	25%	Median	75%
Moving-Average	4.96	22.43	0.58	4.74	-7.11	3.94	18.47
Primary	-1.08	20.17	-0.74	8.95	-13.47	-3.39	6.07
Final	25.09	23.43	0.54	4.14	7.14	21.84	37.94
Panel C: Five Factor							
Sets	Mean	SD	Skew	Kurt	25%	Median	75%
Moving-Average	7.33	33.67	1.50	7.06	-10.93	-0.16	19.74
Primary	-1.41	13.64	0.24	4.24	-13.41	-5.49	2.35
Final	13.08	13.51	0.89	3.52	1.89	9.19	20.32

Table 11. Size decile portfolios

This table reports (Panel A) the Kendall rank correlation coefficients between the NYSE/AMEX/NASDAQ size-decile portfolios and the NYSE/AMEX volatility-decile portfolios from January 1965 to December 2014 and (Panel B) the summary statistics for out-of-sample Fama and French four-factor annualized alphas for the trading rules across size-decile portfolios. Primary Set is a randomly generated set of technical trading rules. Final Set is the set of optimized trading rules after applying optimization procedure. Moving-Average Set is the set of moving average strategies using lags of 3 to 100 days. The alpha values are computed on all of the NYSE/AMEX volatility-decile portfolios between January 1, 1965 and December 31, 2014. We use a rolling procedure with five-year training period, five-year selection period, and one-year out-of-sample period. We run the optimization procedure for 20 simulations. A time-series of alpha values is calculated by averaging alphas across the rules and the simulations in each set. We report *N* (average number of unique trading rules in each rolling period and simulation), Mean (average value), SD (standard deviation), Skew (skewness), Kurt (kurtosis), 25% (first quartile), and Median, 75% (third quartile).

Panel A: Correlation matrix

		Size decile portfolios									
		Large	2	3	4	5	6	7	8	9	Small
Volatility decile portfolios	Low Volatility	0.31	0.38	0.41	0.43	0.44	0.46	0.47	0.48	0.50	0.49
	2	0.38	0.46	0.51	0.56	0.59	0.62	0.63	0.66	0.68	0.65
	3	0.40	0.48	0.54	0.59	0.63	0.66	0.69	0.71	0.74	0.70
	4	0.41	0.50	0.56	0.61	0.66	0.69	0.72	0.74	0.77	0.70
	5	0.42	0.51	0.57	0.63	0.68	0.72	0.74	0.77	0.78	0.70
	6	0.43	0.52	0.59	0.65	0.70	0.74	0.76	0.78	0.78	0.68
	7	0.45	0.54	0.61	0.67	0.72	0.75	0.77	0.78	0.78	0.66
	8	0.46	0.56	0.62	0.69	0.73	0.75	0.76	0.77	0.76	0.63
	9	0.49	0.58	0.64	0.69	0.72	0.73	0.73	0.72	0.70	0.58
	High Volatility	0.53	0.59	0.61	0.61	0.60	0.58	0.56	0.55	0.53	0.43

Table 11 Continued

Panel B: Summary statistics									
	Sets	N	Mean	SD	Skew	Kurt	25%	Median	75%
	Moving-Average	98	-6.21	19.98	-0.64	6.74	-17.64	-5.05	6.39
Large	Primary	500	-0.45	13.57	0.07	8.52	-8.70	-1.50	4.72
	Final	162	19.60	16.25	0.24	2.48	4.77	17.53	32.05
	Moving-Average	98	1.49	21.19	0.50	4.00	-10.80	1.30	15.26
2	Primary	500	-0.46	16.85	-0.17	5.11	-11.04	-1.66	7.06
	Final	325	37.38	21.20	0.28	3.90	22.91	33.60	46.51
	Moving-Average	98	1.94	23.32	0.33	3.91	-10.76	1.39	16.52
3	Primary	500	-0.21	16.75	-0.12	4.80	-9.96	-0.68	8.14
	Final	335	35.95	21.92	-0.05	3.41	21.75	35.92	48.79
	Moving-Average	98	3.68	24.03	0.38	3.80	-8.84	3.72	19.95
4	Primary	500	-0.04	16.71	-0.05	5.20	-9.42	-0.11	8.63
	Final	328	34.82	23.11	0.15	3.31	17.91	32.62	47.02
	Moving-Average	98	5.77	23.50	0.48	4.42	-6.95	6.82	20.34
5	Primary	500	0.05	16.57	-0.02	5.08	-8.74	0.39	9.20
	Final	333	31.58	23.43	0.31	3.61	15.57	29.93	42.87
	Moving-Average	98	9.27	23.72	0.61	4.31	-5.45	8.97	21.91
6	Primary	500	0.27	17.82	0.14	5.00	-8.75	1.00	11.03
	Final	326	28.90	24.88	0.46	3.00	7.39	25.87	42.26
	Moving-Average	98	14.43	25.16	0.67	3.84	-2.07	12.32	30.26
7	Primary	500	0.53	20.80	0.19	5.17	-9.21	2.21	13.76
	Final	373	30.63	24.96	0.21	2.71	10.72	28.28	45.98
	Moving-Average	98	19.01	26.40	1.48	6.75	2.50	15.43	31.37
8	Primary	500	0.79	23.61	0.32	6.95	-8.76	3.18	15.71
	Final	407	32.12	27.21	1.48	6.00	12.74	24.83	44.43
	Moving-Average	98	24.6	28.3	1.7	8.9	5.5	21.9	38.4
9	Primary	500	1.1	27.7	0.4	7.7	-9.0	4.4	19.3
	Final	412	34.6	28.5	1.6	7.5	14.7	30.1	47.0
	Moving-Average	98	25.78	33.44	0.62	5.65	6.20	19.04	38.58
Small	Primary	500	1.12	30.12	0.20	5.72	-11.36	5.79	20.70
	Final	370	31.03	30.07	0.97	5.80	12.38	27.14	46.02
	Moving-Average		9.97	26.73	0.87	6.26	-6.03	7.81	24.52
Diversifying across assets	Primary		0.27	20.80	0.39	8.05	-9.43	0.70	11.51
	Final		32.32	25.44	0.83	5.35	13.81	29.44	45.25

Table 12. Genetic algorithm and standard machine learning algorithms

This table reports the intercepts (generalized alpha) and slopes from regressing the net-of-cost out-of-sample daily portfolio return of genetic algorithm versus common machine learning algorithms (SVM + Logistic, Decision Tree, KNN, Random Forest, Ensemble of machine learners and, LSTM network). We use 10 NYSE/AMEX volatility decile portfolios from 1965 to 2014. We use 10-fold cross-validation to train the standard machine learning algorithms. We assume a fixed transaction cost of 5 *bps*, and use the algorithms' output signals to engage in buying and short-selling the underlying assets. The out-of-sample return is computed as the equal-weighted average of out-of-sample returns across the volatility decile portfolios. The intercept (generalized alpha) is annualized and in percentages. The slope is in percentages. The values in parentheses are *t*-statistics.

	Intercept (generalized alpha)	**Slope**
SVM + Logistic	2.62	-0.04
	(3.24)	(-0.09)
Decision Tree	2.53	-0.36
	(2.26)	(-0.60)
KNN	2.42	0.12
	(3.24)	(0.11)
Random Forest	2.54	-0.29
	(3.18)	(-0.42)
Ensemble	2.51	-0.72
	(4.18)	(-0.95)
LSTM Network	1.56	0.19
	(1.17)	(0.92)

Appendix

A. Crossover Operation

The goal of the crossover operation in genetic algorithm is to generate two new offsprings with shared characteristics of their parents. We explain the operation with an example. Suppose that we randomly select two trading rules from the existing population, labeled Parent 1 and Parent 2 (Figure A.1). In the cross-over operation, we substitute the left branch in the Buy-side of Parent 1 with the left branch of Sell-side of Parent 2. The substitution generates two new offsprings added to the existing population. The offsprings are illustrated in Figure A.2.

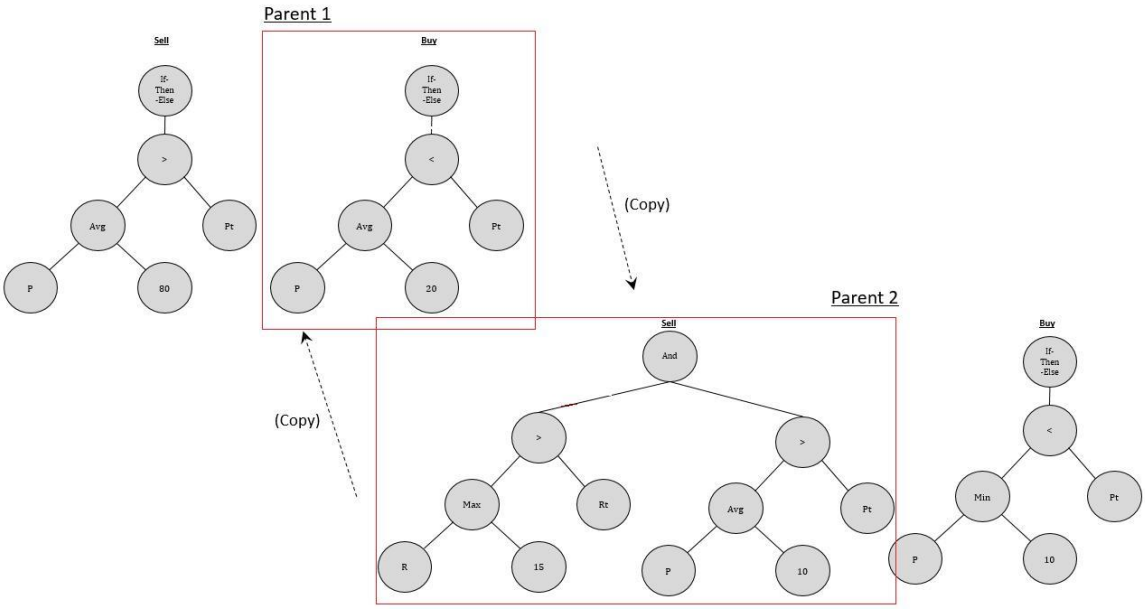


Figure. A.1. Sample trading rules
These Trading rules are chosen randomly from the existing population for the purpose of cross-over operation. We substitute the buy-side branch of Parent 1 to the sell-side branch of Parent 2. The resulting trading results, offsprings, are shown in Figure A.2.

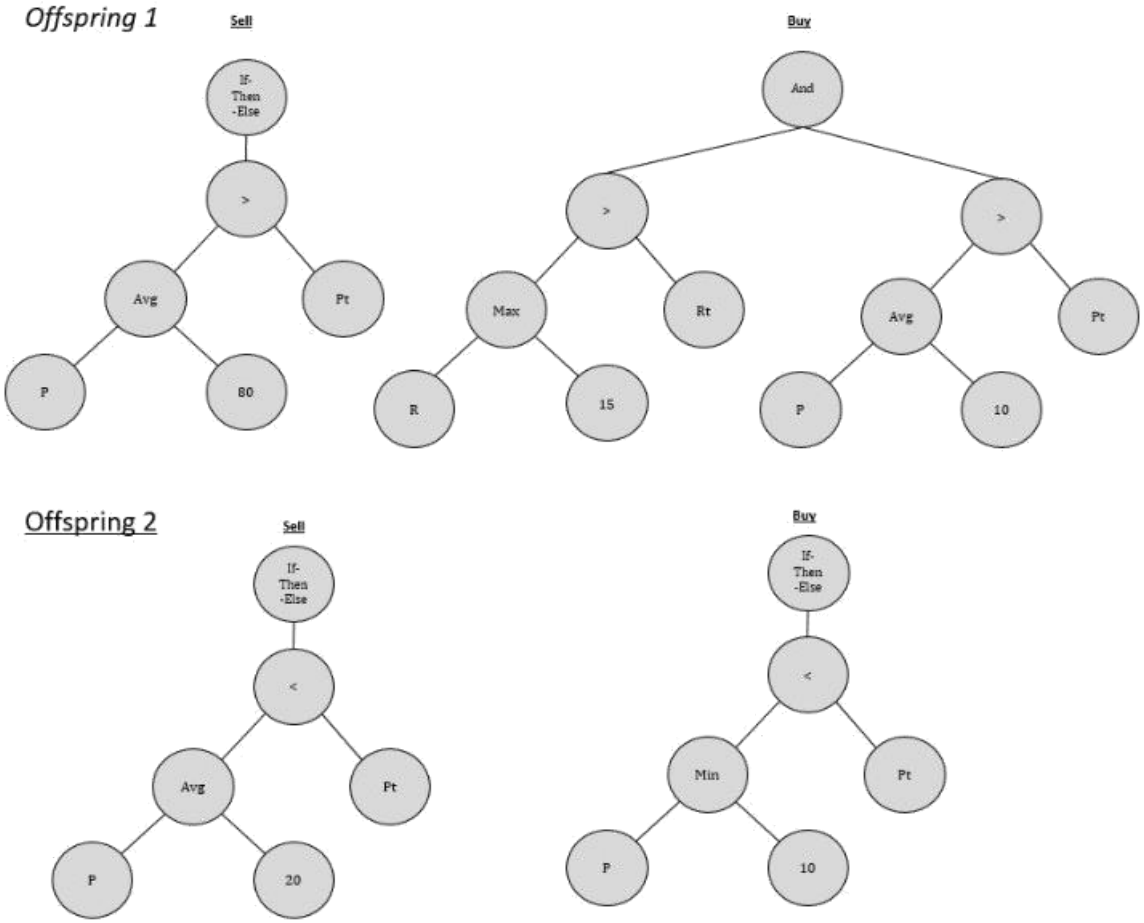


Figure. A.2. Offsprings from the crossover operation

This figure presents the resulting trading rules (labeled as offsprings) from crossover operation on the trading rules (Parents) in Figure A.2.

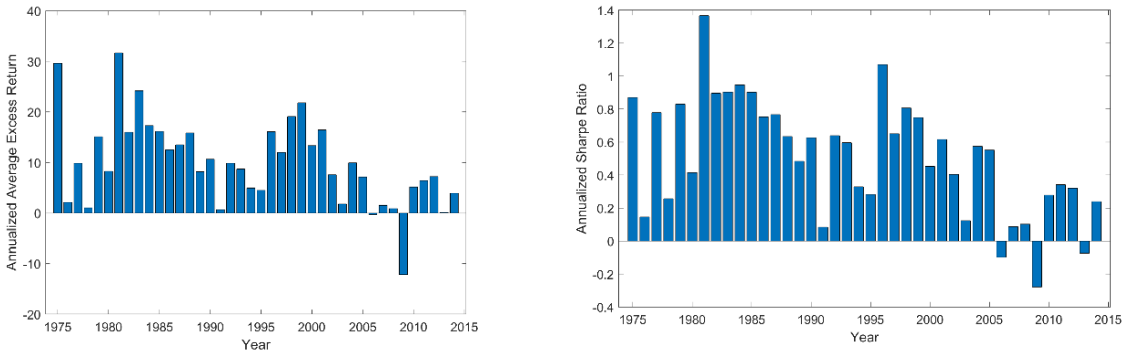
B. Average estimated bid-ask spreads on Dow stocks

Table B1. Bid-ask spread

This Table reports the yearly bid-ask spread on Dow Jones Industrial stocks from Jones (2000) between 1975 to 1999. The values are in basis points (*bps*).

Year	Bid-ask spread (bps)
1975	70
1976	60
1977	53
1978	50
1979	52
1980	70
1981	55
1982	50
1983	45
1984	42
1985	62
1986	70
1987	58
1988	50
1989	55
1990	65
1991	50
1992	42
1993	40
1994	38
1995	36
1996	30
1997	25
1998	20
1999	21

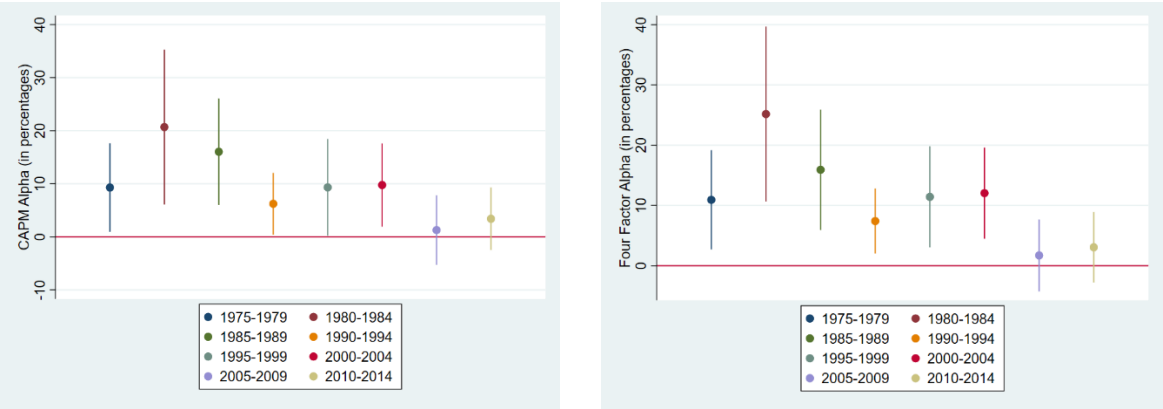
C. Individual stocks — averages excess return as objective function



(a) Average Excess Return (b) Sharpe Ratio

Figure C1. Performance through time: individual stocks

This figure reports the (a) average excess return and (b) Sharpe ratio for one-year out-of-sample periods for the set of optimized trading rules across 100 highly capitalized stocks from NYSE/AMEX/NYSE from 1965 to 2014. For each out-of-sample year and each stock, we use the last ten years (five-year training period and five-year selection period) to search for optimized trading rules. We set the fitness value equal to average return in excess of risk-free rate if the *t*-statistic of the null hypotheses with mean equal to zero is higher than 2; otherwise, the fitness function is equal to -M (M being a large value). The results are averaged over stocks and trading rules. We take the transaction costs into account by using the yearly fixed estimates from Jones (2000) prior to 2000 and the actual daily quoted spread from the TAQ dataset from 2000 to 2014.



(a) CAPM Alpha (b) FF4 Alpha

Figure C2. Alphas through time: individual stocks

This figure reports the (a) CAPM alpha and (b) four-factor alpha for five-year out-of-sample periods for the set of optimized trading rules across 100 highly capitalized stocks from NYSE/AMEX/NYSE from 1965 to 2014. For each out-of-sample year and each stock, we use the last ten years (five-year training period and five-year selection period) to search for optimized trading rules. We set the fitness value equal to average return in excess of risk-free rate if the *t*-statistic of the null hypotheses with mean equal to zero is higher than 2; otherwise, the fitness function is equal to -M (M being a large value). We take the transaction costs into account by using the yearly fixed estimates from Jones (2000) prior to 2000 and actual daily quoted spread from the TAQ dataset from 2000 to 2014. For each of the five-year out-of-sample periods, we compute the average excess return across stocks for each day (post–transaction cost), and compute the annual CAPM and Four-factor alphas. The plot shows the point estimates and the 95% confidence intervals.

D. Performance in population and data deciles

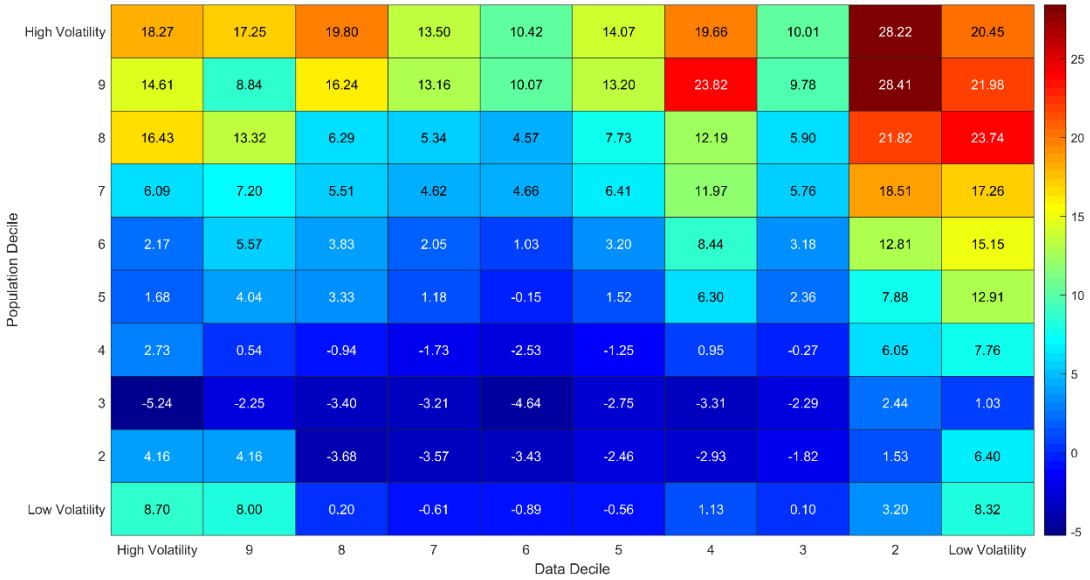


Figure D1. Annual average alphas in population and data deciles

This Figure presents the average four-factor Fama-French annualized alphas with a 5 *bp* transaction cost adjustment. We define "Population Decile" as the volatility decile used by the genetic algorithm to search for final rules and also "Data Deciles" as the volatility decile in which we evaluate the out-of-sample performance of final rules. In each out-of-sample year, we search for a set of optimized trading rules by applying our algorithm to each of the NYSE/AMEX volatility--decile portfolios (the Population Deciles) between January 1, 1965 and December 31, 2014. We use a rolling procedure with five-year training period, five-year selection period, and one-year out-of-sample period. In applying each trading rule, as we move between a long position in the asset, holding the risk-free asset, and a short position in the asset, we subtract a transaction cost of 5 *bps*. We run the optimization procedure for 20 simulations. For the set of rules in each population decile, we compute the average annualized alpha net of cost in different Data Deciles.

Appendix E. Computational power and the out-of-sample alphas

To further examine the extent to which the increase in computing power is related to the out-of-sample performance of our machine learning algorithm, we examine the relationship between a the Number of Transistors on Integrated Circuits (Transistor Count) --- a measure of computational power --- to the out-of-sample alphas.

Number of Transistors on Integrated circuits (Transistor Count)

According to the Moore's law, the number of transistors on integrated circuits doubles approximately every two years. This measure captures the rise of computational power through time with observations start from 1959 (Moore's original publication was in 1965).

Figure E.1 shows the Transistor Count in relation to the out-of-sample alphas earned by our machine learning algorithm. Clearly, as we move forward in time, the computational power, measured by the number of transistors per microprocessor increases while the out-of-sample alphas decreases. The correlation between the our-of-sample alphas and the transistor count is -0.39 (-0.04).

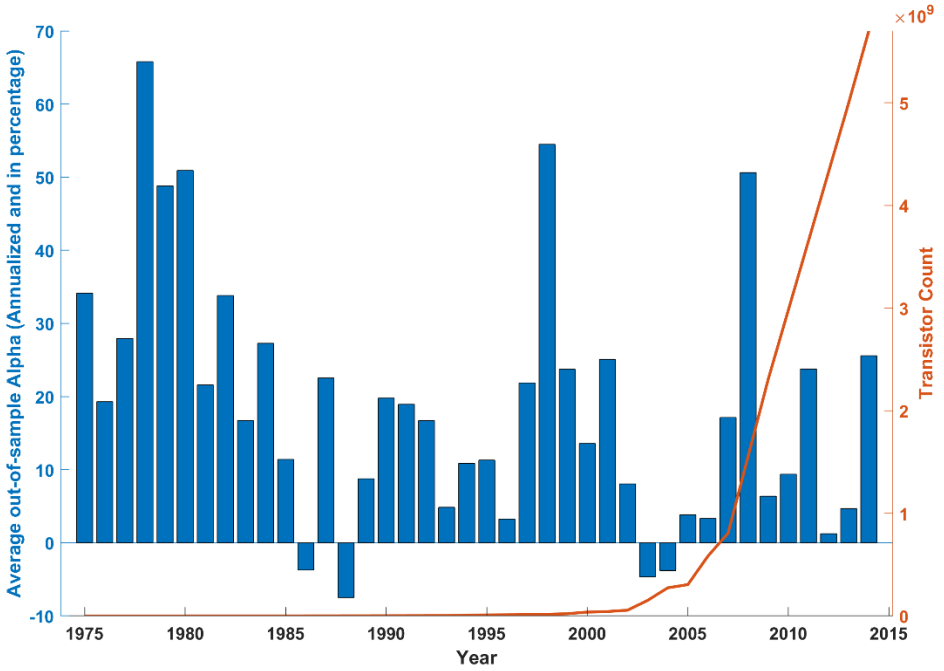


Figure E.1. Performance and transaction cost

This Figure presents four-factor Fama-French annualized alphas through time with a 5 *bps* transaction cost. In each out-of-sample year, we search for a set of optimized trading rules by applying our algorithm to each of the NYSE/AMEX volatility--decile portfolios between January 1, 1965 and December 31, 2014. We use a rolling procedure with five-year training period, five-year selection period, and one-year out-of-sample period. In applying each trading rule, as we move between a long position in the asset, holding the risk-free asset, and a short position in the asset, we subtract a transaction cost of 5 *bps*. We run the optimization procedure for 20 simulations. In each out-of-sample year, we compute an average of Sharpe ratio values over the rules, volatility decile portfolios, and simulations. The red line shows the Transistor Count, the number of transistors per microprocessors in each year.