Designing a Novel and High Performance Algorithmic Trading Model using Evolutionary AutoML and Technical Analysis

Abhiram Tirumala
Rishi Bhatnager
Sriram Mudireddy
Pranav Manjunath
Georgia Institute of Technology
Atlanta, Georgia, USA

Jason Zutty
Georgia Tech Research Institute
Atlanta, Georgia, USA

Abstract

Trading point prediction is the action of identifying ideal buy and sell points for stocks to maximize profit. It is a widely studied application of Machine Learning on time series data due to the abundance of available historical data and the challenges presented by stocks' noisy nature. However, we have found that this is not an application that has drawn significant attention from the automated machine learning (AutoML) community, despite its ideal nature, due to the large number of available models and algorithms that can address this problem. In this research, we employ the Evolutionary Multi-objective Algorithm Design Engine (EMADE). This search framework uses genetic programming to automate model creation and hyperparameter optimization. Traditionally, EMADE produced novel algorithms for tabular, time-series, and image-based problems. This research extends EMADE's capabilities for trading algorithms by adding technical indicator (TI) primitives and novel objective functions. We present analyses on objective sets, learners, TI primitives, and their hyperparameters for this problem. To measure the effectiveness of the evolved models, we evaluate profit percentage on historical US stock data. We have found that the models discovered through EMADE AutoML search techniques produce returns up to 36.38% on average, more than two-fold that of state-of-the-art.

CCS Concepts

• **Computing methodologies** → **Machine learning**; **Genetic programming**; • **Mathematics of computing** → **Time series analysis**.

Keywords

Finance, Automated Machine Learning (AutoML), Genetic Programming

ACM Reference Format:
Abhiram Tirumala, Rishi Bhatnager, Sriram Mudireddy, Pranav Manjunath, and Jason Zutty. 2022. Designing a Novel and High Performance Algorithmic Trading Model using Evolutionary AutoML and Technical Analysis. In *Proceedings of The Genetic and Evolutionary Computation Conference 2022 (GECCO '22).* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

GECCO '22, July 9–13, 2022, Boston, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 Introduction

Financial modeling and Stock Market analysis have been studied for decades due to the mass availability of historical data and the unpredictability of securities. Machine Learning (ML) research has brought promising results, demonstrating how regression-based algorithms can be applied to time-series data for both price forecasting and trading signal prediction by using historical prices and volume data. Performance measures for these models include regression-based mean error calculations for price forecasting models, and accuracy and classification objectives for trading signal prediction.

One drawback in these data processing pipelines is that search spaces are limited, often not exploring the extent of different feature extractions, preprocessing hyperparameters, and learner hyperparameters. In the field of data science, novel architectures for hyperparameter tuning expand the model search space drastically, and provide more holistic optimizations of data science pipelines.

Automated Machine Learning (AutoML) has proven to be effective for hyperparameter search and pipeline optimization in ML [8]. One implementation of AutoML is the Evolutionary Multi-objective Algorithm Design Engine (EMADE), initially introduced as the Georgia Tech Multi-Objective Evolutionary Programming (GTMOEP) framework[11–13]. EMADE applies Genetic Programming to evolve ML pipelines from high-level primitives, including data-preprocessing, signal processing, feature extraction, and ML functions in its primitive set. EMADE has been successfully utilized to solve tabular, time-series, and image data problems. We find that a genetic programming-based solution through EMADE allows for a thorough search across the primitive search space and develops a Pareto-optimal front showing the trade-off space of high-performing individuals over a span of objectives. This research uses Technical Indicator (TI) calculation libraries and Sci-kit Learn Regression models [3, 10] to extend EMADE's functionality and apply its time series capabilities to historical stock market data. In this research, we show how applying EMADE to stock prediction methodologies can discover new algorithms that produce better returns than the state-of-the-art (SOTA).

2 Background

One popular application of Artificial Intelligence in the field of Financial modeling is creating and interpreting forecasts of stock

prices to make trading decisions. These solutions ultimately depend on the efficiency of the underlying models in accurately predicting buy/sell/hold decisions for a security. Much research has been conducted in applying technical analysis to train ML models, as most financial analysts use the same information to make trading decisions.

Dash et al. [7] focused on training a Computationally Efficient Function-Link Artificial Neural Network (CEFLANN) with relevant TIs as features to a training set. Dash et al. use six TIs as inputs to their proposed model:

- *Simple Moving Average (SMA).* A continuous average of the most recent stock prices for a specified period. [1]

$$SMA(n) = \frac{1}{n}\sum_{i=0}^{n} P_{t-i} \tag{1}$$

- *Moving Average Convergence Divergence (MACD).* Demonstrates the relationship of the difference between a short-term exponential moving average (EMA) and a long-term EMA of a stock's price. [1]

$$EMA(n) = P_t * \frac{2}{n+1} + EMA_{t-1}(n) * \left(1 - \frac{2}{n+1}\right) \tag{2}$$

$$MACD(n_1, n_2) = EMA(n_1) - EMA(n_2) \tag{3}$$

- *Stochastic Oscillator (STOCH).* Gauges momentum based on a stock's price history by comparing its closing price to a range of its prices over a certain period of time. These measurements are reported as K% and D% values, where D% is the moving average of K%. [1]

$$K\%(n_1) = \left(\frac{P_t - Low(n_1)}{High(n_1) - Low(n_1)}\right) * 100\% \tag{4}$$

$$D\%(n_2) = \frac{1}{n_2}\sum_{i=0}^{n_2} K\%_{t-i} \tag{5}$$

- *Relative Strength Index (RSI).* A momentum oscillator that measures the magnitude of recent price changes of a stock. [1]

$$RSI(n) = 100 - \left(\frac{100}{1 - \frac{\text{Avg. Gain}}{\text{Avg. Loss}}}\right) \tag{6}$$

- *William's %R (WILLR).* A momentum oscillator that measures overbought and oversold levels. [1]

$$WILLR(n) = \frac{High(n) - P_t}{High(n) - Low(n)} \tag{7}$$

While results in this work are promising, the selection of the TIs used as features are arbitrary, and their hyper-parameters are not explored. This research only captures a small set of algorithms out of a larger set of possibilities.

In their 2008 and 2011 work, Chang et al. [4, 5] used a Piecewise Linear Representation (PLR) to develop a continuous trading signal that indicates the degree to which one should buy or sell a stock. An Artificial Neural Network (ANN) is trained on TIs as features to develop a predicted trading signal. The trading signal informs buy and sell decisions by exponentially smoothing the output of the ANN and creating dynamic bounds. When the predicted signal crosses a bound, a buy/sell trading decision is triggered. This research utilizes a process of Stepwise Regression Analysis (SRA) to determine optimal TIs from a predetermined set for each stock to be feature inputs into the model. This work uses a more analytical reasoning for selection of technical indices, but still lacks in searching optimal TI parameters as well as ML model hyperparameters by only including 14 TIs and one learner.

Chang et al. [6] later extended their use of PLR in 2016 by implementing a Takagi-Sugeno Fuzzy (TS Fuzzy) rule-based model in combination with a Support Vector Regression (SVR) model to identify trading points within a time series. This research showed the best results among all three studies, and we will use this as a basis for comparison in subsection 4.3. This research also explores the PLR-SVR algorithm, which applies the PLR post-processing, and uses a fixed threshold bound for trading point decision, which will be discussed more in subsection 3.4.

3 Methods

Our research uses a methodology inspired by that of Chang et al. by implementing the PLR algorithm [4] used to create a continuous trading signal and using the postprocessing methods involving the static bounds for trading point decision used in the PLR-SVR model [6]. This technique forms the basis for our objective functions implemented in EMADE to develop an algorithm that searches for optimal models among a search space comprising:

(1) *Technical Indicator Feature Set.* A subset of various TIs which are to be used by a learner
(2) *Technical Indicator Parameters.* For a selected TI, these are the given parameters such as period or number of standard deviations.
(3) *Learner Algorithm Selection.* These are a set of learner algorithms that can be trained to predict a trading signal.
(4) *Learner Algorithm Hyperparameter Tuning.* The model hyperparameters which are to be used by the selected learner algorithms.

Our research also explores the optimal configuration of objective sets for EMADE that lead to the production of the best individuals. We aim to develop a model that outperforms those of similar works, as well as conduct analysis to understand the primitive combinations in algorithms produced by EMADE. We aim to create a new SOTA model and develop a method of analysis that is transferable to other applications of EMADE and Genetic Programming.

3.1 Data Collection from US Stock Market

In this research, we gathered historical stock market data from six stocks and one index for the purposes of comparing with other trading models. They are Apple Inc. (AAPL), The Boeing Company (BA), Verizon Communications Inc. (VZ), Caterpillar Inc. (CAT), Johnson & Johnson (JNJ), Exxon Mobil Corporation (XOM), and the S&P 500 Index (S&P 500). 253 days were used for training data, ranging from January 2, 2008 to December 31, 2008, and 124 days were used for testing data, ranging from January 2, 2009, to June 30, 2009. Additionally, the previous 40 data points for the training and testing ranges were gathered as inputs to primitives. For each trading day, we collected the Adjusted Close Prices, Trading Volume, Price High, Price Low, and Open Price, all of which are relevant for calculating TIs [9].

3.2 Implementing Technical Indicators Primitives

We extend EMADE's capabilities by including commonly used TI primitives as part of the its primitive set. We implemented the TIs from the TA-Lib library [2]. Table 1 shows the list of implemented TIs. In addition, we introduce a set of dataset normalization functions to the primitive set, including Min-Max Scaling, Standard Scaling, and Robust Scaling, from the Sci-kit Learn API [10]. All implemented TIs have strongly-typed parameters, such as period, that EMADE will include in its search space.

3.3 Data Preprocessing

Building on the 2008 and 2011 work by Chang et al. [4, 5], we adopted the Piecewise Linear Representation (PLR) and Trading Signal calculations performed on historical stock data. In this research, we make the assumption that a model can possess at most one share of an asset at a time, meaning that trading actions alternate between buying and selling. PLR is used to identify the key turning points within a raw financial time series that indicate trading points. Segmentation threshold value is a parameter of the PLR algorithm, which controls the sensitivity of the identified trends. A larger threshold results in longer price trends output by the PLR, whereas a smaller threshold results in shorter price trends. Our study adopts the threshold values reported by Chang et al. in their 2016 work [6], shown below in Table 2.

We convert the turning points output by the PLR algorithm to a continuous trading signal. This trading signal is used in the training data to be predicted by a learner. For each segment in the PLR, the trading signal is calculated as follows [5]. For an uptrend segment:

$$T_i = \begin{cases} 0.5 - \frac{i-1}{L} & \text{if } i \leq \frac{L}{2} \\ \frac{i}{L} - 0.5 & \text{if } i > \frac{L}{2} \end{cases} \tag{8}$$

For a downward segment:

$$T_i = \begin{cases} 0.5 + \frac{i-1}{L} & \text{if } i \leq \frac{L}{2} \\ 1.5 - \frac{i}{L} & \text{if } i > \frac{L}{2} \end{cases} \tag{9}$$

where L represents the length of the uptrend/downtrend segment, T_i represents the trading signal of the i-th day. Figure 1 shows the result of the PLR and its conversion into a trading signal for AAPL on its training period.

3.4 Data Post-processing

We identify buy, sell, and hold actions using a fixed bound, similar to the method described for the PLR-SVR model [6]. A static upper and lower boundary equal to the following:

$$bound = \mu \pm \sigma \tag{10}$$

μ represents the mean of the trading signal produced by the training data, and σ represents one standard deviation (SD) of the trading signal produced by the training data. The first point at which the predicted trading signal is less than the lower bound triggers a buy action, and the next point at which the predicted trading signal is more than the upper bound triggers a sell action. The process alternates between buys and sells in this manner.

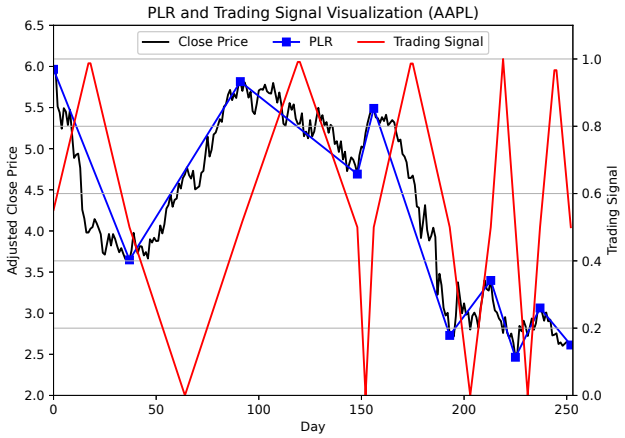


Figure 1: PLR and Trading Signal Visualization on AAPL Stock

We then calculate the profit accumulated over the time period as follows:

$$profit = \sum_{i=1}^{T} \frac{S_i - B_i}{B_i} \tag{11}$$

Where T represents the number of transactions made, S_i represents the sell price of the ith transaction, B_i represents the buy price of the ith transaction.

3.5 Objective Functions

Our study explores the effect of different combinations of objective functions on the best performing individual learners discovered by EMADE's search. We define a better configuration of objectives to be one that results in individuals producing higher profit percentages. We study the impact of four objective functions:

(1) *Overall Profit Percentage (PP).* Using the method described in subsection 3.4 for determining the buy/sell points for a time period, we look to maximize the profit computed in Equation 11 produced for each stock/index.
(2) *Average Profit per Transaction (APT).* For each identified transaction, we average the profit to skew towards individuals that make more profitable trades. This metric is maximized.
(3) *Variance of Profit per Transaction (VPT).* For each identified transaction, we determine the variance of the profit to skew towards individuals that make more consistent returns on trades. This metric is minimized.
(4) *Monte Carlo Distribution Complementary Cumulative Distribution Function (CCDF).* To adjust for natural stock price trends, we developed a novel method where over an arbitrarily large number of simulations, trading points are randomly selected throughout the time period with the same expected number of trades as the model makes. We collect the resulting profit for each simulation and approximate a normal distribution of the Monte Carlo simulations. We compute the z-score of the individual with respect to this distribution, showing to what degree the performance of an individual can be attributed to random chance. We minimize the CCDF (Right-tailed Area) described by the z-score.

Table 1: List of Technical Indicator Primitives

Type	Technical Indicators
Momentum	Williams' %R (WILLR), Stochastic Oscillator (STOCH), Moving Average Convergence/Divergence (MACD), Relative Strength Index (RSI), Stochastic Relative Strength Index (STOCHRSI), Aroon (AROON), Average Directional Movement Index (ADX)
Volume	Ease of Movement (EMV), Chaikin Money Flow (CMF), Volume Weighted Average Price (VWAP), Volume Weighted Moving Average (VWMA), On Balance Volume (OBV)
Misc.	Identity*, Simple Moving Average (SMA), Exponential Moving Average (EMA), Bollinger Bands (BBANDS), Fibonacci Retracement (FIBRET), Beta (BETA), Linear Time Series Forecast (TSF), Rate of Change (ROC), Hilbert Transform Dominant Cycle Period (HTDCP)
Difference	SMA Bias (BIAS), Delta SMA (ΔSMA), Delta SMA Bias (ΔBIAS), EMA Bias (EBIAS), Delta EMA (ΔEMA), Delta EMA Bias (ΔEBIAS), Delta MACD (ΔMACD), Delta STOCH (ΔSTOCH), Delta RSI (ΔRSI), Delta WILLR (ΔWILLR)

** The Identity primitive is a generic primitive that selects a subset of the input data streams (Price, Volume, etc.) as a feature of the training set.*

Table 2: Optimal PLR Threshold Values [6]

AAPL	BA	VZ	CAT	JNJ	XOM	S&P 500
0.5σ	0.7σ	0.8σ	0.5σ	0.7σ	0.9σ	0.7σ

In our implementation, we convert the PP and APT objectives to minimization objectives by negating the output of these functions. This allows us to identify an upper bound for the Area Under Curve (AUC) calculation for Pareto fronts. For a given individual, fitness values are computed by averaging the objective scores across each of assets' test periods, as described in subsection 3.1. These resulting fitness values drive the EMADE search.

3.6 Design of Experiments

From the four objective functions described in subsection 3.5, we developed configurations of objective sets for EMADE to optimize, listed in Table 3. To compare each objective set, we include PP in all configurations. For each set of objectives, we conducted three trials of EMADE with 192 compute hours per trial. Table 4 shows the hyperparameters for EMADE runs that were held constant across all trials and objective configurations. We seeded each run with a constant set of 102 individuals representing an assortment of combinations of TI Primitives paired with varied Learner models. These seeded individuals contain primitive sets of commonly used TIs (SMA, EMA, MACD, etc.) and Learners (MLP, Decision Tree Regressors, SVR, etc.) as well as replicated models like PLR-BPN [4] and PLR-SVR [6]

Table 3: EMADE Objective Set Configurations

2 Objectives	3 Objectives	4 Objectives
PP+APT	PP+APT+VPT	PP+APT+VPT+CCDF
PP+VPT	PP+APT+CCDF	
PP+CCDF	PP+VPT+CCDF	

Table 4: EMADE Hyperparameters

Parameter	Value
Initial Population Size	102
Elite Pool Size	64
Launch Size	36
Crossover Mating Prob.	0.90
Crossover Ephemeral Prob.	0.50
Headless Chicken Prob.	0.20
Headless Chicken Ephemeral Prob.	0.10
Insert Mutation Prob.	0.10
Ephemeral Mutation Prob.	0.25
Node Replace Prob.	0.20
Uniform Mutation Prob.	0.10
Shrink Mutation Prob.	0.10
Selection Algorithm	NSGA2

4 Results

4.1 Analysis of Experiments

To find the objective functions that yield the best individuals, we compared each experiment primarily based on the PP objective, as this is generally the most important metric to a trader, and it was included in all objective sets to facilitate this comparison. We employed a few different ways to compare the profit percentages of experiments: using all individuals, just using the top tier of individuals, or just using the best individual from that experiment.

Figure 2 compares experiments using the profit percentage of each individual in the last 100 generations of each experiment across all trials. This graph compares the performances of the most mature models built by EMADE in each experiment. The graph shows that optimizing for PP+APT yields the greatest median profit, followed by PP+CCDF, and then PP+VPT. However, taking into account the full range from lower to upper extremes (but disregarding outliers), the PP+VPT objective set seems to come out on top: of the three aforementioned objective sets, PP+VPT has the highest lower extreme, a higher upper extreme and lower quartile than PP+CCDF,

and a much smaller range than PP+APT. The relatively tight interquartile range from the PP+VPT set indicates a better chance of generating high-performing individuals.

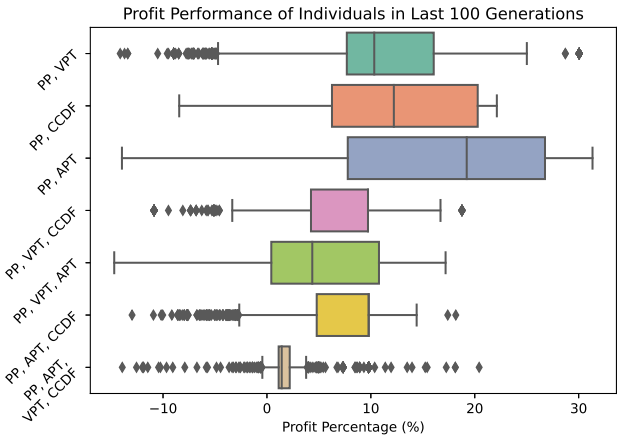


Figure 2: Profit Percentage Comparison Across Objective Set Configurations

Another way to examine which objective set is best is to compare the top tier of individuals from each set, as the most profitable individuals are typically those which hold the greatest significance. For this analysis, we perform the following for each objective set:

(1) We take the average return of the 10 most profitable individuals for each trial i as μ_i. This becomes the random variable we are observing across trials.
(2) We compute the expected average return across the trials as $\bar{\mu} = E[\mu_i]$.
(3) We compute the SD of the average return across the trials as $\hat{\sigma}_{\mu_i} = \sqrt{E[\mu_i^2] - \bar{\mu}^2}$.

Figure 3 displays the expected average return $\bar{\mu}$, with error bars denoting one SD $\hat{\sigma}_{\mu_i}$ of these three average returns. As the graph shows, the PP+VPT objective set has the highest expected average return of just over 27%, closely followed by PP+APT and PP+CCDF at 26.2% and 24.75% profit, respectively. However, all three of these objective sets also have relatively high SDs, meaning their base performance of top individuals approaches those of the other objective sets. In fact, the PP+VPT+APT and PP+VPT+CCDF objective sets have the third and fourth highest base performances despite having relatively low expected average returns.

4.2 Analysis of Individuals

Now that we have identified PP+VPT as our best performing objective set, we can use the individuals from this experiment to analyze how specific learners and TIs correlate to Profit Percentage. Figure 4 shows the average profit percentage of a pairing of a given learner with a given TI, using all individuals from all three trials with the PP+VPT objective set. All individuals containing both the given learner and TI (at least) are considered for analysis (i.e. an individual need not contain just that learner and indicator). Cells with no value indicate there were no individuals with that pairing.

This figure can show us which learners and TIs are best used for constructing trading algorithms. The heat map shows that Argmax

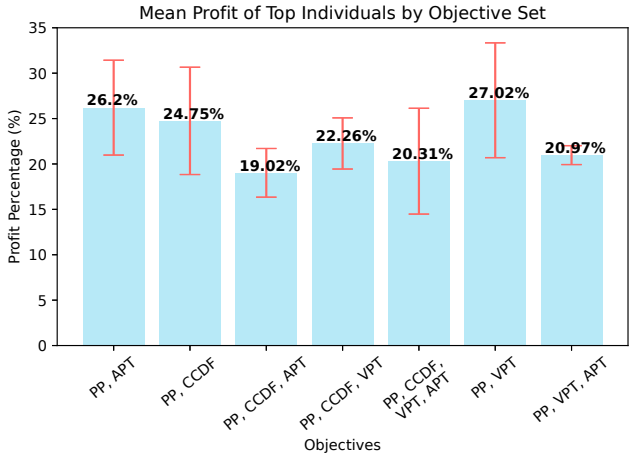


Figure 3: Average Profit Percentage of Top-Tier Individuals within Objective Set Configurations. Error bars denote one SD of mean return across trials.

is generally the strongest learner with an average PP of 7.2%, but it's interesting that Argmin and KNN, especially with certain TIs, can produce higher performing individuals. STOCH, WILLR, and Identity (which simply returns a stream of input data) also perform very well, but as widely used indicators this is expected. However, difference indicators, VWMA, and TSF come out even higher, despite being unconventional and generally unused. Unexpectedly, the top performing pair is an Argmin learner using the VWMA, a relatively unused yet incredibly effective indicator. Ironically, it is these types of unique pairings where AutoML engines such as EMADE excel: they can find efficient models that perform well but are not very obvious, and thus are most likely to be discovered via a comprehensive automated search like EMADE's.

Figure 5 offers insight into which pairings are being most thoroughly explored by EMADE. This figure uses the same experiment as Figure 4. As expected, there is a strong relationship between the most prevalent learner-indicator pairs and those pairs which performed most optimally. Through its evolutionary process, EMADE will reward these individuals through an increased presence in future generations. Additionally, this heatmap shows us that EMADE comprehensively searched each pairing: just about every TI was represented in at least one hundred individuals, and only two learners (AdaBoost and Kernel Ridge regression) were not searched very extensively, but they also did not perform well.

4.3 Performance of Best Individual

Interestingly, our best individual comes from the PP+CCDF objective set. Table 5 outlines this individual. To summarize, the model contains several TI primitives, including multiple repeated indicators with different parameters, and an Argmin learner. The Argmin model returns the index of the feature (in this case indicator) with the lowest value. For instance, if the value returned by the first indicator (the STOCH difference indicator with a K-period of 1 and a D-period of 32) is smaller than those of all other indicators, the Argmin learner returns a 0; if the value returned by the second indicator (the EMV with a period of 8) is smaller than the value returned by all other indicators, the Argmin learner returns a 1, etc.

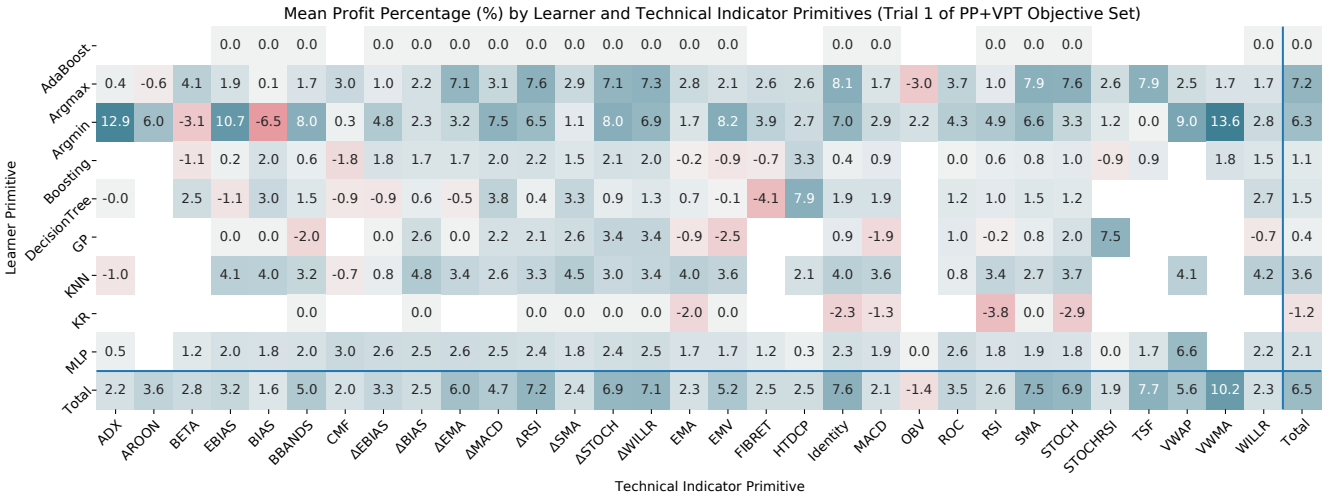


Figure 4: Heatmap of Individuals' Performance by Learner-Indicator Pairings. Box annotations represent the average profit percentage of individuals that contain that pairing.

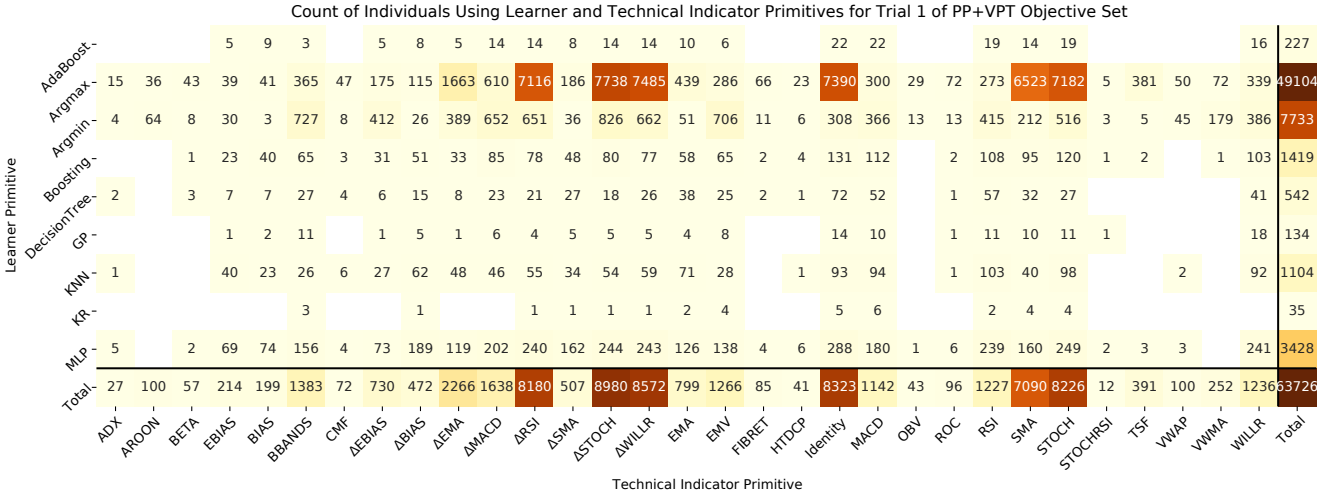


Figure 5: Heatmap of Individuals' Frequencies by Learner-Indicator Pairings. Box annotations represent the quantity of individuals that contain that pairing.

As explained in subsection 3.4, the model alternates between buy and sell signals as the trade signal goes below the lower bound and above the upper bound, respectively. Since the upper and lower bounds are both between zero and one, the model can only signal a buy when the Argmin learner returns a zero, as all other possible return values (indices of features) are above the upper bound. Since the first transaction must be a buy, the first trade, a buy, will occur when the ΔSTOCH(1, 32) is the lowest-valued indicator. The following trade, a sell, will occur at the next point at which the ΔSTOCH(1, 32) is not the lowest-valued indicator. Then, a buy will occur the next time the ΔSTOCH(1, 32) is the lowest-valued indicator, and so on. In short, the individual will alternate between buy and sell transactions when the ΔSTOCH(1, 32) is and is not the lowest-valued indicator.

Given this, it is clear that the ΔSTOCH(1, 32) indicator is the decisive centerpiece of this individual. This is notable due to the uniqueness of this indicator: while the stochastic oscillator is relatively well-known, difference indicators are not, nor is the stochastic oscillator with these parameters (the standard is 14, 3). In this case, the stochastic oscillator is calculated using Equation 4 where $n_1 = 1$ and Equation 5 where $n_2 = 32$. The indicator returns an ordered pair of the difference between the K% and D% of the current day and that of the previous day.

Figure 7 shows how this individual behaves on Apple's stock. In Figure 7a, the predicted signal graphs the output of the Argmin learner. All trading points equal to zero correspond to buys (when the ΔSTOCH(1, 32) is the lowest-valued indicator), and all other

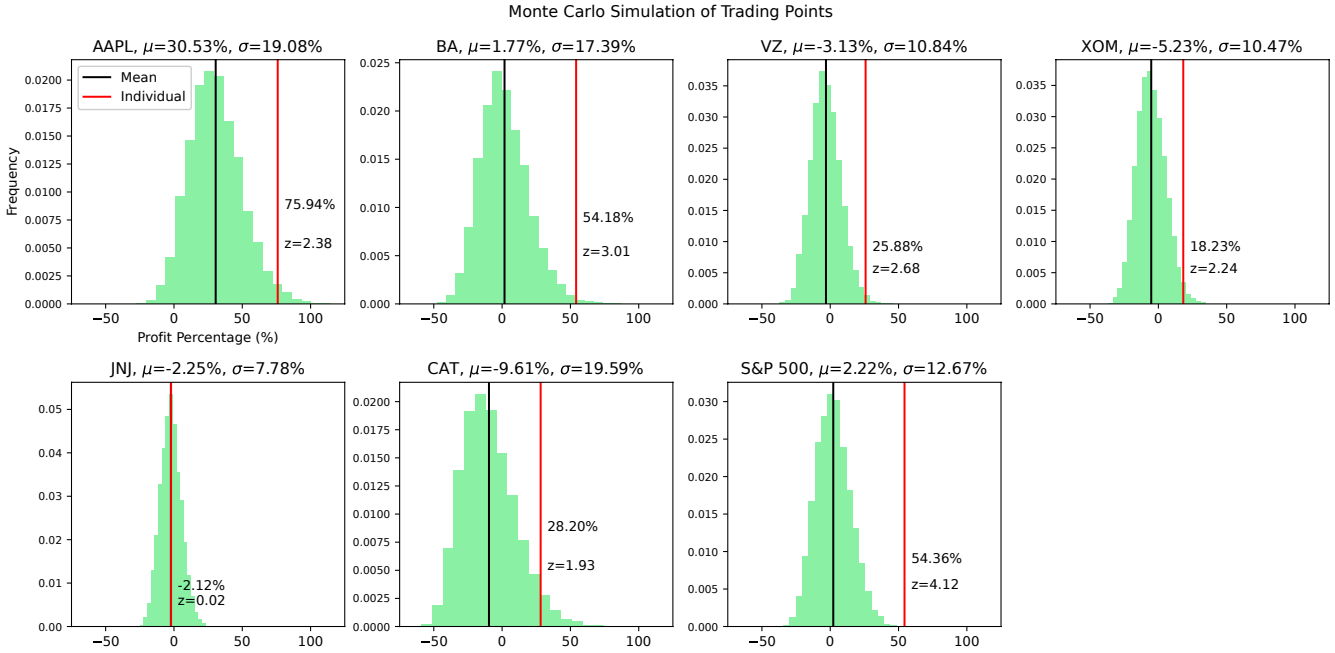


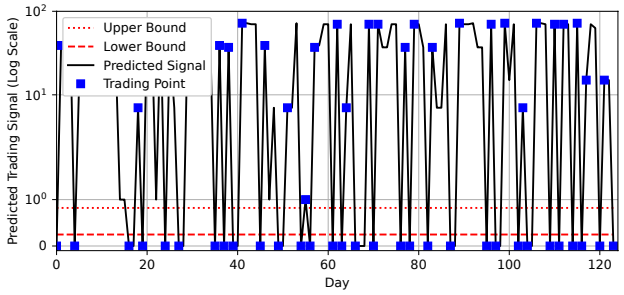
Figure 6: Comparison of Most Profitable Individual to Monte Carlo Simulation. Red line represents the profit percentage of this individual against the distribution of random trades in green. z represents the z-score of the individual's performance.

Table 5: Top Performing EMADE Individual

Type	Primitives
Indicators	ΔSTOCH(1, 32), EMV(8), BBANDS(5, 10), EMV(0), BBANDS(128, 980), ΔSTOCH(1, 14), AROON(8), BBANDS(7, 10), EMV(0), BBANDS(10, 3), ΔSTOCH(1, 14), BBANDS(6207, 32), ΔSTOCH(1, 8),EMV(8), ΔSTOCH(1, 32), ΔSTOCH(1, 10), EMV(12), EMV(0), BBANDS(14, 8), ΔSTOCH(1, 32), EMV(2), BBANDS(6207, 32), ΔSTOCH(1, 8), ΔSTOCH(1, 10), EMV(8), ΔMACD(1, 32), EMV(12), BBANDS(14, 10), ΔRSI(8), ΔSTOCH(1, 4), ΔSTOCH(1, 4), EMV(0)
Model	ARGMIN

ΔSTOCH takes inputs K% period and D% period, EMV takes an input of period, BBANDS takes an input of period and delta, AROON takes in an input of period, ΔMACD takes in an input of fast period and slow period, and ΔRSI takes in an input of period.

(a) Predicted Trading Signal with Fixed Threshold Bound



(b) Interpreted Trading Points (Profit: 75.94%)

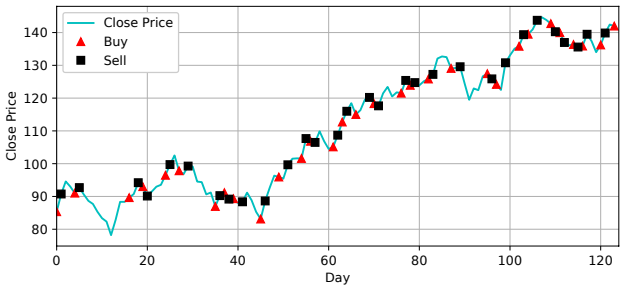
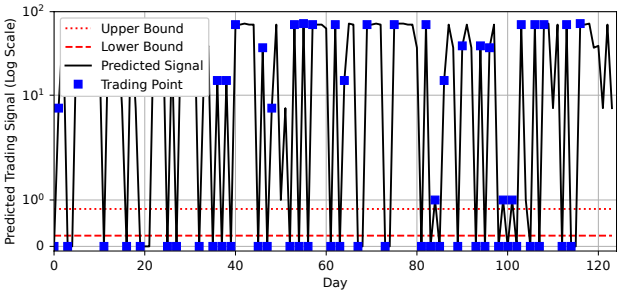


Figure 7: Predicted Trading Actions for AAPL Test Data

trading points are sells. Each trading point in Figure 7a corresponds to a buy or sell in Figure 7b.

Figure 6 illustrates the performance of this individual with respect to the Monte Carlo simulations of a random trading strategy. The distribution shows the proportion of random trading experiments that attained a particular return for a given asset. Above each distribution is the ticker for that stock/index and the mean and SD of the distribution. The red line marks the profit percentage attained by this individual. Notably, the individual achieves an approximate return of at least two SDs above the distribution average for all assets except for JNJ. This approach to visualizing an individual's performance helps determine the relative performance of the individual compared to a naive trading strategy. That is, even

(a) Predicted Trading Signal with Fixed Threshold Bound



(b) Interpreted Trading Points (Profit: 25.88%)

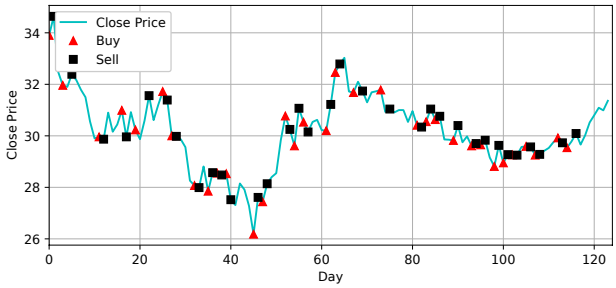


Figure 8: Predicted Trading Actions for VZ Test Data

a bad model or the naive one shown can make a very good return of 30% profit over the six month test period used for Apple because its stock rose sharply during this period. On the other hand, as shown in Figure 8b, Verizon's stock had a downward trend during the test period used for this paper, yet this individual was able to attain a profit of almost 26%. This performance is good for any security over a six month period, let alone one that is generally decreasing in value.

To get a better understanding of this individual's performance, we compared it to SOTA models using similar techniques. As a basis for comparison, we used the TS Fuzzy model [6] discussed in section 2. The paper compared its novel model (the TS Fuzzy model) to various baseline models, all of which utilize the same time period that we used. The baseline models included a PLR-BPN (back propagation network) model that we also use for comparison. Our final baseline is the average profit achieved by the random trading Monte Carlo simulation discussed earlier, as a measure of the profit generated by the natural trend of the stocks/index tested. Our results as compared to these baselines are summarized in Table 6. The PPs presented under our work are those achieved by the best individual described in this section. As the table shows, our research has found a model that outperforms all baselines in six of the seven assets studied, and it only incurs a loss on one stock. On average, we at least double the profits of all baselines.

5 Conclusion

We demonstrated a new AutoML methodology for conducting time series analysis in the field of financial modeling and Stock Market analysis. We create a trading signal using a PLR algorithm approach on U.S. Historical Stock data. EMADE then uses the trading signal, stock/index data, a host of TIs, and novel objective functions to search the algorithmic space and find unique solutions that effectively trade the given stocks/index. The methodology's experimental results display that evolved individuals provide significant trading signals relevant to investors. Additionally, evolved individuals provide much better results than those of conventional ML (PLR-BPN) [5] and more contemporary (TS Fuzzy) [6] SOTA approaches. The models generated are especially impressive given their performance relative to Monte Carlo simulations of random trading patterns. Thus, it is clear that the individuals evolved through EMADE have the capability of producing strong results.

Table 6: Results Comparison of Most Profitable Individual

Stock/ Index	TS Fuzzy[6]	PLR-BPN[6]	Monte Carlo Avg.	Our Work
AAPL	45.16%	12.97%	30.53%	75.94%
BA	32.66%	17.50%	1.77%	54.18%
VZ	-1.24%	9.36%	-3.13%	25.88%
XOM	14.78%	16.80%	-5.23%	18.23%
JNJ	6.31%	-1.99%	-2.25%	-2.12%
CAT	17.01%	27.72%	-9.61%	28.20%
S&P 500	11.90%	3.77%	2.22%	54.36%
Average	**18.08%**	**12.30%**	**2.05%**	**36.38%**

Future work in this context includes introducing held-out validation data sets to assess overfitting of individuals produced by EMADE, since the testing data is driving its search. Additionally, exploring approaches to portfolio optimization outside of stock-price prediction, using AutoML to optimize other time series analysis problems, and improving EMADE's capability to perform time series analysis are possible avenues of research.

It is important to note that the research presented in this paper generalizes beyond trading point detection using EMADE. The novel Monte Carlo CCDF objective function that we introduced is also applicable in other use cases of AutoML, as it effectively compares models to a set of baseline expected outcomes. This comparison allows one to gauge the degree to which a model outperforms random decision-making and determine the model's statistical significance. Another notable contribution is our primitive analysis visualizations. These analyses improve researchers' understanding of EMADE's search by showing both the genetic makeup of the explored algorithms as well as their phenotypic (objective) performance. Chief among these contributions, however, is the relevance of AutoML towards time series analysis: with such a vast search space, AutoML techniques allow researchers to comprehensively explore unconventional, yet effective, solutions.

References

[1] [n.d.]. Fidelity Learning Center. https://www.fidelity.com/learning-center/trading-investing/technical-analysis/technical-indicator-guide
[2] John Benediktsson and Brian Cappello. [n.d.]. https://mrjbq7.github.io/ta-lib/
[3] Lars Buitinck, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, Vlad Niculae, Peter Prettenhofer, Alexandre Gramfort, Jaques Grobler, Robert Layton, Jake VanderPlas, Arnaud Joly, Brian Holt, and Gaël Varoquaux. 2013. API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*. 108–122.

[4] Pei-Chann Chang, Chin-Yuan Fan, and Chen-Hao Liu. 2008. Integrating a piecewise linear representation method and a neural network model for stock trading points prediction. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 39, 1 (2008), 80–92.

[5] Pei-Chann Chang, T. Warren Liao, Jyun-Jie Lin, and Chin-Yuan Fan. 2011. A dynamic threshold decision system for stock trading signal detection. *Applied Soft Computing* 11, 5 (2011), 3998–4010. https://doi.org/10.1016/j.asoc.2011.02.029

[6] Pei-Chann Chang, Jheng-Long Wu, and Jyun-Jie Lin. 2016. A Takagi–Sugeno fuzzy model combined with a support vector regression for stock trading forecasting. *Applied soft computing* 38 (2016), 831–842.

[7] Rajashree Dash and Pradipta Kishore Dash. 2016. A hybrid stock trading framework integrating technical analysis with machine learning techniques. *The Journal of Finance and Data Science* 2, 1 (2016), 42–57. https://doi.org/10.1016/j.jfds.2016.03.002

[8] Xin He, Kaiyong Zhao, and Xiaowen Chu. 2021. AutoML: A Survey of the State-of-the-Art. *Knowledge-Based Systems* 212 (2021), 106622.

[9] Alpha Vantage Inc. [n.d.]. Alpha Vantage. https://www.alphavantage.co

[10] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

[11] Jason Zutty, Domenic Carr, Rodd Talebi, James Rick, Christopher Valenta, and Gregory Rohling. 2019. Reducing bathymetric-lidar algorithm uncertainty with genetic programming and the evolutionary multi-objective algorithm design engine. In *Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications*, Vol. 11006. International Society for Optics and Photonics, 110061U.

[12] Jason Zutty, Daniel Long, Heyward Adams, Gisele Bennett, and Christina Baxter. 2015. Multiple objective vector-based genetic programming using human-derived primitives. In *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*. 1127–1134.

[13] Jason Paul Zutty. 2018. *Automated machine learning: A biologically inspired approach.* Ph.D. Dissertation. Georgia Institute of Technology.