EasyChair Preprint

№ 15909

Building Cross-Sectional Trading Strategies via Geometric Semantic Genetic Programming

Kritpol Bunjerdtaweeporn and Alberto Moraglio

March 14, 2025

Building Cross-Sectional Trading Strategies via Geometric Semantic Genetic Programming

Kritpol Bunjerdtaweeporn and Alberto Moraglio

Department of Computer Science, University of Exeter, Exeter, United Kingdom
{kb801,a.moraglio}@exeter.ac.uk

Abstract. Cross-sectional trading strategies involves constructing portfolios by comparing expected performance of assets within a group, typically using predicted returns. In this study, we frame the estimation of cross-sectional expected returns as a symbolic regression problem, and investigate the predictive capabilities of geometric semantic genetic programming in developing cross-sectional trading strategies in the U.S. stock market. We employ standard genetic programming and other common methods used for studying cross-sectional returns as baselines for comparison. Our findings indicate that geometric semantic genetic programming provides better forecast accuracy, portfolio performance, and ranking accuracy than standard genetic programming. Furthermore, we show the limitations of errors-based metrics as performance measurement in cross-sectional trading strategies.

Keywords: Geometric Semantic Genetic Programming · Portfolio Construction · Stock Returns Prediction · Stock Selection

1 Introduction

Cross-sectional trading (CST) strategies are a widely used investment approach in asset management. The goal of CST strategies is to build a portfolio by buying assets expected to outperform and selling those expected to underperform within an asset group at a specific point in time. Assets are compared using various criteria, ranging from a single characteristic[1] to output of models that combine multiple characteristics. A common criterion for evaluating assets is their predicted returns. Forming an effective portfolio in CST strategies requires understanding why certain assets are expected to give higher or lower returns than others, rather than focusing on how each asset returns will change over time. The models are found by estimating the cross-sectional expected asset returns, which can be formulated as a symbolic regression problem. The portfolio formed by CST strategies profits from relative differences in asset returns with minimal influenced by overall market movement [15]. Linear models estimated via ordinary least squares (OLS) have long been the preferred choice in both research and practice for cross-sectional returns forecasting due to its simplicity

[1] In the context of computer science, a characteristic of an asset is analogous to a predictor or feature, terms which will be used interchangeably.

and effectiveness. In recent years, numerous machine learning (ML) applications [13,16,17] have been proposed to improve predictions over OLS. The improved predictive performance of these ML models is largely attributed to their ability to capture non-linear signals [6]. Geometric semantic genetic programming (GSGP) is a variant of genetic programming (GP) that uses geometric semantic operators (GSOs) to replace the standard genetic operators. GSOs apply precise syntax modification to individuals, resulting in predictable and well-known geometric properties in their semantics. Moreover, the fitness landscape seen by GSOs is unimodal error surface for any supervised ML problem. Given the previous success of ML models in improving prediction accuracy, GSGP presents a promising, yet unexplored, approach for building CST strategies. In this paper, we examine predictive power of GSGP for constructing CST strategies in the U.S. stock market, exploring its potential to enhance forecasting accuracy and portfolio performance. The main contributions of this study are threefold. First, we introduce a novel application of GSGP. Second, we present a comparative analysis of GSGP and the variant in which the optimal mutation step is calculated for geometric semantic mutation operator ($GSGP_o$) [14,29] against standard GP and other common methods used for studying the cross-sectional stock returns, including OLS, LASSO, gradient boosted regression trees (GBRT), random forest (RF), and neural networks (NNs) [7,11,16]. Our results show that both GSGP and $GSGP_o$ offer additional value gains over standard GP, providing better forecasting accuracy, portfolio performance and ranking performance. Comparing to the remaining methods, $GSGP_o$ portfolio tends to show competitive performance across overall metrics, ranking near the top-performing models, although it may not always be the absolute best. Third, we empirically show the limitations of error-based metrics when used for evaluating CST strategies or as an optimisation objective. The rest of this paper is organised as follows: Section 2 introduces CST strategies. Section 3 outlines methodology. Section 4 describes experimental setup. Section 5 presents results and discussion. Section 6 concludes and suggests future work.

2 Cross-Sectional Trading Strategies

Cross-sectional trading (CST) strategies involve selecting assets by comparing their relative performance at a specific point in time. In this paper, we consider a set of stocks as the primary asset class for our CST strategy. Essentially, CST strategies focus on buying stocks that are expected to outperform in a long portfolio and selling those that are expected to underperform in a short portfolio. Forming a portfolio in CST strategies typically involves four main steps [37]: *score calculation*, *score ranking*, *stock selection*, and *portfolio construction*.

Score Calculation The score for each individual stock is computed based on its corresponding characteristics. This process produces a vector of scores for all stocks as the final output. For example, an individual stock score could be determined by a single characteristic such as its earnings-to-price ratio [3] or through more sophisticated models that combine several characteristics [16,24,37].

Score Ranking Score ranking involves sorting the score vector, where stocks with the highest scores are considered the best. The sorting procedure produces a predicted rank list.

Stock Selection The selection step involves retaining some groups of the stocks based on their rank to form a long-short portfolio. Stocks expected to outperform are included in a long portfolio, those expected to underperform in a short portfolio, while stocks ranked in the middle are excluded.

Portfolio Construction Finally, weights are assigned to the selected stocks such that the total in both long and short portfolio sums to zero. Some common weight allocation schemes include equal-weighting, where weights are distributed equally to each stock, and characteristic-weighting, which allocates weights based on stock-specific characteristics, such as market capitalisation [11,16] or in proportion to the inverse of their historical volatility [37].

CST strategies contrast with time-series trading (TST) strategies, where trading decisions for each asset are based on expected individual performance. The key difference lies in the threshold for buying or selling assets: TST strategies use a zero-return threshold, while CST strategies employ average return as the benchmark. For example, consider a scenario with 100 stocks, each predicted to yield a positive return. CST strategies would buy a group of stocks with the highest expected return and sell those with the lowest, while TST strategies would buy all 100 stocks.

3 Methodology

This section introduces a general framework for stock returns prediction. We then describe our forecasting models and provide a brief explanation of their implementation in each subsection.

3.1 General Framework

The score for each stock is calculated based on its predicted future returns corresponding to its characteristics. Suppose $\mathbf{S} = \{s_1, \ldots, s_N\}$ represents the set of N stocks, where s_i denotes an individual stock. For simplicity, we assume that the number of stocks remains constant over time $t = 1, \ldots, T$. We consider the relationship between stock returns and their characteristics as a general additive prediction model, as in Gu et al. [16]:

$$r_{i,t+1} = \mathbb{E}_t(r_{i,t+1}) + \epsilon_{i,t+1} \tag{1}$$

where $r_{i,t+1}$ is the return on stock s_i at time $t+1$. The cross-sectional expected return $\mathbb{E}_t(r_{i,t+1})$ is assumed to be represented by some *true* underlying function g^*, which takes stock characteristics as an input:

$$\mathbb{E}_t(r_{i,t+1}) = g^*(p_{i,t}) \tag{2}$$

where $p_{i,t} = (p_{i,t}^{(1)}, ..., p_{i,t}^{(K)}) \in \mathbb{R}^K$ denotes a vector of K stock characteristics of stock s_i at time t. The objective is to approximate a function g^* by estimating a function g that maps $p_{i,t}$ into a single real value, $g : \mathbb{R}^K \to \mathbb{R}$, a task that can be

viewed as a symbolic regression problem. The form of an estimated function g is left unspecified, allowing it to be either linear or non-linear, as well as parametric $g(p_{i,t};\theta)$ or non-parametric $g(p_{i,t})$. Although the algorithms used to approximate g^* vary, the output $g(p_{i,t})$ generally aims to predict the *true* stock returns by minimising the mean squared forecast errors (MSFE), defined as:

$$MSFE(g) = \frac{1}{N(T-1)} \sum_{t=1}^{T-1} \sum_{i=1}^{N} \Big(r_{i,t+1} - g(p_{i,t}) \Big)^2 \tag{3}$$

We use $g(p_{i,t})$ instead of $\hat{r}_{i,t+1}$ to emphasise that scores do not necessarily need to be predicted returns but rather the output of a function. By assuming g^* to be time-invariant, the MSFE function utilises pooled data across the entire panel (see Figure 1) to estimate the model once, ignoring the time dimension rather than estimating it across each time as in the Fama-MacBeth framework [17]. This approach reduces the computational time required by ML algorithms.

3.2 Forecast Models

Our selection of algorithms to be compared with GSGP are those commonly used in the cross-sectional studies for estimating expected stock returns including nature-inspired algorithms, penalised linear models, and tree-based approaches [7,11,16,24].

Genetic Programming (GP) GP is an evolutionary computation method based on the principle of Darwinian evolution. The population contains the candidate solutions of function g, represented in the form of tree structure. While the advantage of using GP lies in its flexibility to choose fitness function(s) specifically tailored to the nature of the problem [4,5,21,24], we use MSFE as our fitness function to ensure direct comparability with other algorithms and to serve as a baseline for GSGP. Our function set consists of both linear and non-linear operators: $\{+, -, *, aq, sin, tanh\}$. The combination of linear function and function such as *sin* or *tanh* in our function set enables GP to approximate any arbitrary function similar to the universal approximation capability of neural network [33,40]. Analytic quotient (aq) [32] is used to remove discontinuities that can often arise from using unprotected and protected division. Replacing protected division with aq in the function set improves the generalisation ability of the evolved model in symbolic regression, an issue that cannot be resolved through model selection using a validation dataset [33]. Our terminal set consists of stock characteristics and an ephemeral constant that return a value in range $[-1, 1]$. While Liu et al. [24] consider the population size ($npop$) and the maximum number of generations ($ngen$) to be the two most important hyperparameters for a similar problem, they found that $ngen$ plays a more critical role. In our preliminary runs, the training fitness did not vary significantly with different values of $npop$ and when $ngen$ exceeded 40. Therefore, we consider $ngen$ and maximum tree depth ($maxdepth$) to be the two most important hyperparameters, keeping $npop$ fixed. When recombination operators result in an individual that exceed the tree depth limit, we randomly return one of its parents. To

further improve the model robustness, we adopt an ensemble approach in our model training [22,24,41]. Specifically, we select the five best unique individuals from the final population and form a prediction by taking the arithmetic mean of each individual forecast.

Geometric Semantic Genetic Programming (GSGP) GSGP uses geometric semantic operators (GSOs) to replace the standard syntax-based genetic operators. Given N inputs corresponding to the characteristics of each stock, the semantic of a function g is defined as $s(g) = \big(g(p_1), \ldots, g(p_N)\big)$. This vector can be represented as a point in N-dimensional space, called semantic space. Note that the target vector $\vec{r} = \big(r_1, \ldots, r_N\big)$ is also a point in the semantic space. The offspring's semantic produced by GSOs is predictable with well-known geometric properties. Moreover, the fitness landscape seen by GSOs is unimodal for any supervised ML problem. The distance between the target semantic and individual semantic can be used as the fitness function in GSGP, which is MSFE for our problem. Geometric semantic crossover (GSCX) and geometric semantic mutation (GSM) are originally defined in Moraglio et al. [31] as:

Definition 1. ***Geometric Semantic Crossover (GSCX)*** *Given two parent functions $g_1, g_2 : \mathbb{R}^K \to \mathbb{R}$, $GSCX(g_1, g_2) = g_r \cdot g_1 + (1 - g_r) \cdot g_2$, where g_r is a random function whose codomain$(g_r) \in [0, 1]$*

Definition 2. ***Geometric Semantic Mutation (GSM)*** *Given a parent function $g : \mathbb{R}^K \to \mathbb{R}$, $GSM(g) = g + ms \cdot (g_{r_1} - g_{r_2})$, where ms is a mutation step and g_{r_1}, g_{r_2} are random functions*

GSCX has a nice property that the fitness of an offspring is guaranteed not to be worse than the worst of its parents. However, the offspring semantic can reach the target only if the target semantic lie within the convex hull of its parents [35]. Moreover, applying GSCX increases the node size exponentially, making the operator inapplicable in some cases [39]. While applying simplification helps reduce node size, the evolved function often remains large and computationally intensive [27]. In contrast, GSM increases node size linearly and using only GSM can yield comparable or better results than using both GSOs in the evolutionary process [31,39]. Although the semantic of random functions generated by the original GSM operator are unbounded, later studies showed that limiting the codomain of random functions to a pre-defined interval using a sigmoid function improves generalisation [14,39]. However, Nicolau and McDermott [34] found that offspring semantic from unbounded GSM have long-tailed distributions in each dimension due to the lack of a bounded radius, with variance differing according to the distribution of training cases. Thus, applying a sigmoid function blindly may result in poor semantic and unnecessary complexity. For these reasons, we adopt a semantic stochastic hill climber (GSGP with population of size 1) that relies solely on GSM in our evolutionary process. We use a variant of GSM defined in [1] as $GSM(g) = g + ms \cdot \mathcal{N}(g_r)$, where $\mathcal{N}(g_r) = 2 * \frac{g_r - min(s(g_r))}{max(s(g_r)) - min(s(g_r))} - 1$. This variant stabilises the output distribution of random function without requiring a sigmoid function and reduces node size by generating a single random

function. A random function is generated using either *grow* or *full* method, randomly chosen with equal probability, with the same primitive set used in GP. To ensure the offspring is at least as good as its parent, if the normalised random function $\mathcal{N}(g_r)$ worsen the offspring, then $-\mathcal{N}(g_r)$ is considered. The new solution is accepted only if it improves upon its parent. Additionally, we consider GSGP with an optimal mutation step ($GSGP_o$) [14,29], which uses the GSM operator that selects the mutation step to minimise MSFE of the mutated function. We select the best individual (i.e., the one with the lowest training error) from 10 independent evolutionary processes to account for potential poor choice of initial individual [29] and variations in convergence speed. Our GSGP implementation also incorporates higher-order functions and memoization techniques for improved efficiency [30]. A pseudocode for our GSGP evolutionary process is provided in Algorithm 1.

Algorithm 1 Pseudocode for GSGP Evolutionary Process (GSGP)

function GSGP($\mathcal{F}, \mathcal{P}, \vec{r}$) ▷ $\mathcal{F}$: Fitness function
  $ngen \leftarrow 0$ ▷ $\mathcal{P}$: Set of stock characteristics
  Initialise random function g_0 ▷ $\vec{r}$: A vector of stock returns (target)
  while termination criteria not satisfied **do**
    $ngen \leftarrow ngen + 1$
    Generate random function g_r ▷ $\mathcal{N}(g_r) = 2 * \frac{g_r - min(s(g_r))}{max(s(g_r)) - min(s(g_r))} - 1$
    Normalise g_r to obtain $\mathcal{N}(g_r)$
    Calculate ms (pre-determined or based on $\mathcal{N}(g_r)$)
    $g_{temp} \leftarrow g_{ngen-1} + ms \cdot \mathcal{N}(g_r)$
    if $\mathcal{F}(g_{temp}; \mathcal{P}, \vec{r})$ is better than $\mathcal{F}(g_{ngen-1}; \mathcal{P}, \vec{r})$ **then**
      $g_{ngen} \leftarrow g_{temp}$
    else
      $g_{temp} \leftarrow g_{ngen-1} - ms \cdot \mathcal{N}(g_r)$
      if $\mathcal{F}(g_{temp}; \mathcal{P}, \vec{r})$ is better than $\mathcal{F}(g_{ngen-1}; \mathcal{P}, \vec{r})$ **then**
        $g_{ngen} \leftarrow g_{temp}$
      else
        $g_{ngen} \leftarrow g_{ngen-1}$
  return g_{ngen}

Ordinary Least Squares (OLS) OLS is the least complex method in our analysis. It serves as a baseline to assess the additional value gained from using more sophisticated algorithms. Despite its simplicity, OLS is widely used in studies of cross-sectional stock returns [2,3,9,20,23,25].

LASSO LASSO adds an l_1-penalty term to the optimisation problem, encouraging parsimonious models by shrinking some coefficients to zero. This helps mitigate overfitting caused by multicollinearity among predictors, as OLS may potentially captures noise instead signal. We use LASSO since it is analogous to the support vector machine algorithm [19].

Gradient Boosted Regression Trees (GBRT) & Random Forest (RF) Both GBRT and RF are built on regression tree (RT) using the CART algorithm as it is one of the most widely used algorithm among the existing options [26]. Ensemble methods like GBRT and RF aim to tackle overfitting by combining predictions from multiple simple trees into one consensus forecast, rather than building a single complex tree. GBRT is constructed via stagewise additive expansions. The process starts by fitting a simple RT. Next, a second RT is added

by fitting prediction errors of the previous RT. The forecast of the second tree is shrunken by some small positive value, called learning rate, to help prevent the model from overfitting the residuals. The process is repeated for a predetermined number of iterations. The final output is an additive model of simple RTs. RF is built based on the idea of bootstrap aggregation or bagging. Bagging involves randomly selecting samples from the training data with replacement. These bootstrapped samples are then used to construct simple RTs. This process is repeated multiple times. RF extend bagged trees by randomly selecting a subset of features at each split, which helps reducing the correlation among trees across different bootstrap samples.

Neural Networks (NNs) We consider the feedforward neural networks architecture, where each node is connected to all nodes in the previous layer and the connections follow a one-way direction, from the input to output layer. Each node in the input layer represents a predictor and the output layer contains a single node producing the score output. We use Rectified Linear Units (ReLU) as our activation function for its computational efficiency and universal approximation capability. We avoid overfitting by adopting multiple regularisation strategy. First, we use stochastic gradient descent to train our NNs. Second, we apply learning rate shrinkage, where the learning rate is divided by 5 each time MSFE failed to decrease training loss or failed to improve the validation samples, retained from the training samples by some pre-determined threshold. Third, we train NNs with multiple random seeds to form an ensemble to improve generalisation ability [18]. Specifically, we train NNs with five independent random seeds. The predictions are then averaged as the final output. Lastly, we use l_2-penalty term to the weight parameters, shrinking the weights toward zero. We consider NNs with up to three hidden layers, denoted as NN_1, NN_2, and NN_3, selected according to the pyramid rule [28], as using more hidden layers could potentially lead to performance deterioration [16].

The implementation of our models utilises well-established frameworks. We employ Scikit-learn [36] for OLS, LASSO, GBRT, RF, and NNs. For GP, GSGP, and $GSGP_o$, the implementations are based on DEAP framework [12]. The hyperparameters are selected from a comprehensive set of parameter specifications (see Table A1), following commonly used choices in the literature [7,11,16,24].

4 Experimental Setup

We begin with a description of the dataset and preprocessing steps. Next, we explain the sample splitting and hyperparameter tuning process. Lastly, we detail the performance evaluation methods used to assess each algorithm.

4.1 Data

The data are sourced from *FactSet financial data and analytics*, obtained with a permission of *Jupiter Asset Management Systematic Equities* team. We consider the latest available values of the data at the end of each month, represented as a panel, shown in Figure 1. Our stock universe consists of all U.S. firms listed in MSCI North America. The sample spans from January 1990 to December

Month	Predictor 1	...	Predictor K	Return
1	$p_{1,1}^{(1)}$	...	$p_{1,1}^{(K)}$	$r_{1,2}$
	$\vdots$	$\vdots$	$\vdots$	$\vdots$
	$p_{N,1}^{(1)}$	...	$p_{N,1}^{(K)}$	$r_{N,2}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
T-1	$p_{1,T-1}^{(1)}$	...	$p_{1,T-1}^{(K)}$	$r_{1,T}$
	$\vdots$	$\vdots$	$\vdots$	$\vdots$
	$p_{N,T-1}^{(1)}$	...	$p_{N,T-1}^{(K)}$	$r_{N,T}$

Fig. 1: An example of dataset presented in a panel format

2022, encompassing approximately 3000 stocks in total, with an average of 650 stocks per month. We consider 12 stock characteristics, as described in Figure 2. Since each stock characteristic differs in their range of values and unit, it is not straightforward to intuitively compare them. Therefore, we employ a typical transformation to each stock characteristic by converting them into ranks and map into an interval $[-1, 1]$ across each time [7,11,16]. Specifically, we rank each characteristic in ascending order then divided by the number of stocks at each month. The transformed values in the range $[0, 1]$ are multiplied by 2 then subtracted by -1. We use stock returns instead of their excess returns of treasury-bill rate on our analysis to represent the actual performance without any adjustments. The returns are winsorized each month at 1% and 99% during model training to handle outliers, while retaining their original value during portfolio formation. We only include stock observations with complete data on returns and all stock characteristics, i.e., observations with no missing values.

#	**Predictor**	**Definition**
1	Short-term reversal [25]	Returns of the prior month
2	Momentum [20]	Returns from 12 months prior to 2 months prior
3	Long-term reversal [9]	Returns from 36 months prior to 13 months prior
4	Size [2]	Market value of an equity at the end of prior month
5	Earnings-to-price [3]	Net income in the prior fiscal year divided by market capitalisation at the end of the prior month
6	Asset turnover	Net sales and revenues in the prior year divided by market capitalisation at the end of prior month
7	Beta	Market beta estimated from daily returns against MSCI North America from the prior 12 months
8	Book-to-price	Book value of equity divided by market value of equity
9	Debt-to-assets	Short term plus long term debts to total assets
10	Volatility	Daily returns standard deviation from the prior 252 days
11	Dividend yield	Dividends per share over the prior 12 months divided by price at the end of the prior month
12	Return on assets	Income before extraordinary items divided by average total assets in the prior year

Fig. 2: Description of all stock characteristics

4.2 Sample Splitting & Hyperparameter Tuning via Validation

The data are divided into three disjoint parts: *training*, *validation*, and *out-of-sample*, while keeping the temporal order of the dataset, following common practices [7,11,16]. The training data are used to estimate the model for each combination in the hyperparameter set. The validation data are used for hyperparameter tuning. The out-of-sample data, which have neither been used for training nor hyperparameter tuning, are used to evaluate the method's predictive performance. Specifically, forecasts are generated based on the model trained with training samples. MSFE is then calculated using the validation samples for each combination of hyperparameters. This hyperparameter tuning step, which preserves the temporal ordering of the data, has been shown to be preferable to the standard cross-validation scheme for data with a time dimension [8,38]. The model with the hyperparameter combination that best minimises MSFE is selected to forecast during out-of-sample periods. The data from the first 12 months are initially used as a training sample and the subsequent 12 months as validation sample, making the out-of-sample period span from January 1992 to December 2022, for a total of 372 months. The models are retrained on an annual basis. The chosen hyperparameter combination of the trained model is fixed, making out-of-sample predictions over the next 12 months. The training sample increases by 12 months while retaining the entire history. The validation sample remains a fixed size and is rolled forward by 12 months. Figure 3 illustrates the sample splitting and model retraining process.

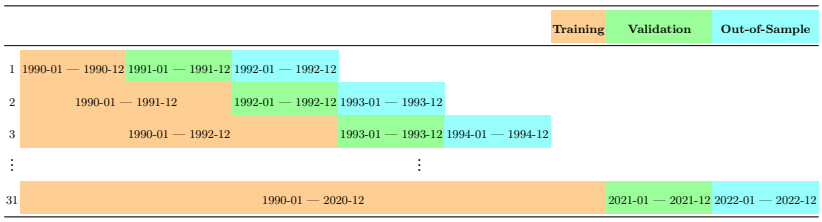

Fig. 3: Sample Splitting and Model Retraining

4.3 Performance Evaluation

The performance of each method is evaluated using various type of metrics, including prediction accuracy, portfolio performance, and ranking accuracy. All calculations are made during out-of-sample period.

We assess the statistical significance of the differences in forecast errors among models using a modified Diebold-Mariano test [10], as defined in Gu et al. [16]. The Diebold-Mariano test is adapted by comparing average prediction errors over time rather than comparing the errors of each individual forecast. Specifically, to test for the forecast performance of method (1) against method (2), the test statistic DM_{12} is defined as:

$$DM_{12} = \frac{\bar{d}_{12}}{\hat{\sigma}_{\bar{d}_{12}}} \tag{4}$$

where,

$$d_{12,t} = \frac{1}{N}\sum_{i=1}^{N}\Big(\big(r_{i,t} - g_1(p_{i,t})\big)^2 - \big(r_{i,t} - g_2(p_{i,t})\big)^2\Big) \tag{5}$$

$g_1(p_{i,t})$ and $g_2(p_{i,t})$ denote the output score for stock s_i at time t using each method. $\bar{d}_{12}$ and $\hat{\sigma}_{\bar{d}_{12}}$ denote the mean and Newey-West standard error of $d_{12,t}$ over the out-of-sample period. We compare the predictive performance against a naive prediction using R^2_{oos}, defined in Gu et al. [16] as:

$$R^2_{oos} = 1 - \frac{\sum_{(i,t)\in\mathcal{T}_{oos}}(r_{i,t} - g(p_{i,t}))^2}{\sum_{(i,t)\in\mathcal{T}_{oos}} r^2_{i,t}} \tag{6}$$

$\mathcal{T}_{oos}$ indicates that the predictions are only evaluated during the out-of-sample period. R^2_{oos} pools prediction errors across stocks and over time into one panel-level assessment of each model.

We form a long-short portfolio at each point in time during the out-of-sample period, following the procedures described in Section 2. Specifically, we divide stocks into ten groups according to their predicted return (i.e., their score). Next, we buy stocks in the top decile to include them in a long portfolio and sell those in the bottom decile in a short portfolio. The selected stocks are weighted equally, ensuring that the total weight sums to 1 in a long portfolio and -1 in a short portfolio. We report the long-short portfolio monthly return mean and its standard deviation, annualised Sharpe ratio, final cumulative returns, maximum drawdown, and turnover. The portfolio turnover is defined following Gu et al. [16] as:

$$Turnover = \frac{1}{|\mathcal{T}_{oos}|}\sum_{t\in\mathcal{T}_{oos}}\left(\sum_{i\in\mathbf{S}}\left|w_{i,t} - \frac{w_{i,t-1}(1+r_{i,t})}{1+\sum_j w_{j,t-1}r_{j,t}}\right|\right) \tag{7}$$

We also assess the correctness of the predicted rank list by computing the average of Spearman's rank correlation coefficient across each time, denoted as ρ_{oos}.

5 Results and Discussion

We report the run closest to the median Sharpe ratio out of 30 simulation runs, as it aligns towards the investment objective and likely presents how each method will perform in practice. All calculations are made based on out-of-sample period.

5.1 Results

Table 1 reports pairwise comparison of the test statistics of modified Diebold-Mariano test. Negative value indicates that the row method outperform the column method by having lower average forecast errors and vice versa. Bold font indicates that the difference is significant at 5% level. Asterisk sign indicates that the difference is significant after conservative Bonferroni adjustment. The forecast errors of GSGP and $GSGP_o$ are significantly less than those of standard GP, with GSGP considered best. However, after applying the conservative Bonferroni adjustment, there is no significant difference between GSGP

Table 1: This table reports pairwise modified Diebold-Mariano test statistics comparing the average forecast errors during the out-of-sample period. Negative value indicates that the row method outperforms the column method, i.e., it has lower average forecast errors and vice versa. Bold font indicates that the difference is significant at the 5% level. Asterisk sign indicates that the difference is significant after Bonferroni adjustment.

	OLS	LASSO	GBRT	RF	NN_1	NN_2	NN_3	GP	GSGP	$GSGP_o$
OLS	–	**5.81***	0.78	**3.79***	**2.38**	0.60	1.06	**-10.15***	**-4.11***	**-5.92***
LASSO		–	-0.98	1.04	**-5.28***	**-3.07**	**-3.06**	**-13.23***	**-7.42***	**-9.07***
GBRT			–	2.06	-1.78	-0.51	-0.32	**-7.94***	**-2.51**	**-3.71***
RF				–	**-4.98***	**-3.24***	**-3.28***	**-11.81***	**-5.84***	**-7.37***
NN_1					–	**2.91**	**2.92**	**-8.26***	-1.59	**-3.47***
NN_2						–	0.45	**-10.17***	**-3.86***	**-5.52***
NN_3							–	**-11.41***	**-4.29***	**-6.07***
GP								–	**8.28***	**6.91***
GSGP									–	**-2.32**
$GSGP_o$										–

and $GSGP_o$. Nonetheless, their forecast errors remain relatively high compared to the remaining methods.

Figure 4 shows the cumulative returns of the decile long-short portfolio as selected by the models during out-of-sample period. The portfolio performance and R^2_{oos} for each method are summarised in Table 2. The numbers are presented without transaction cost, highlighting the raw predictive ability. The results show that both GSGP and $GSGP_o$ achieve higher R^2_{oos} than GP, with GSGP slightly higher than $GSGP_o$. Nonetheless, their R^2_{oos} are still relatively lower than the rest of the methods. $GSGP_o$ achieves the best portfolio performance among the GP-based approaches. Although GP portfolio has lower volatility, its risk-adjusted return, Sharpe ratio, is significantly lower at 0.06, which is more than four times lower. The final cumulative return of GP portfolio is the only one that is less than 1, indicating losses at the end of the period. $GSGP_o$ ranked behind OLS, NN_1, and NN_2 in terms of return mean, Sharpe ratio, and final cumulative return but had a lower maximum drawdown compared to those methods, except for NN_1.

Figure 5 illustrates stock characteristics importance across each method. We calculate the importance of each stock characteristic following the approach in [7,16]. Specifically, the importance of a stock characteristic $p^{(k)}_{i,t}$ is measured by the reduction in R^2_{oos} when it is set to 0 for all predictions at each time period, while other characteristics remain unchanged. The value for each characteristic are normalised to sum to 1 for relative interpretation. The colour gradient within each column indicates the importance of the characteristics for a particular method, with darker colours representing greater importance. The figure

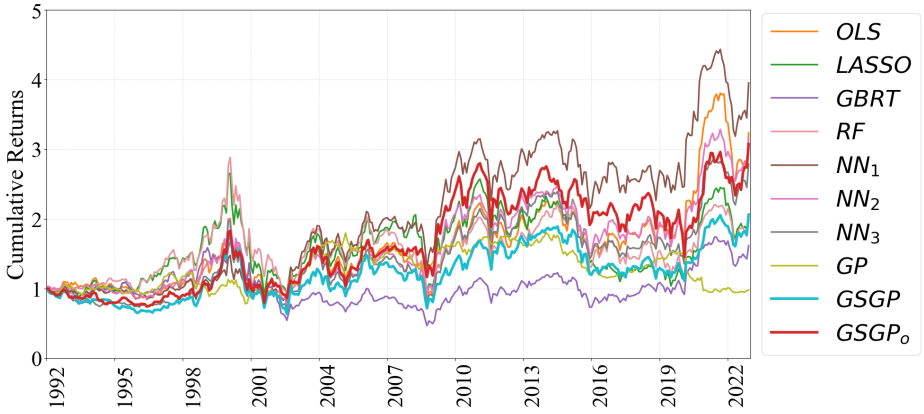

Fig. 4: This figure shows the cumulative returns of decile long-short portfolio, as selected by the models, during out-of-sample period of a run closest to the median Sharpe ratio out of 30 simulation runs for each method

Table 2: This table compares the performance of decile long-short portfolio during out-of-sample periods across methods. The numbers are presented for equal-weighting scheme with monthly rebalance excluding transaction cost, highlighting the raw predictive ability of the models.

	OLS	LASSO	GBRT	RF	NN_1	NN_2	NN_3	GP	GSGP	$GSGP_o$
Return Mean (%)	0.55	0.43	0.33	0.41	0.56	0.50	0.46	0.06	0.39	0.49
Return Std (%)	6.86	7.11	6.39	6.88	6.20	6.29	6.15	3.47	6.37	6.26
Sharpe Ratio	0.28	0.21	0.18	0.21	0.31	0.28	0.26	0.06	0.21	0.27
Final CumRet	3.24	1.97	1.62	1.98	3.95	3.14	2.79	0.97	2.06	3.08
MaxDD (%)	59.83	66.10	72.13	68.12	49.75	62.15	63.42	47.68	64.17	56.37
Turnover (%)	103.12	82.84	96.55	89.44	103.81	105.69	109.76	82.24	101.09	98.25
R^2_{oos} (%)	0.71	0.75	0.73	0.77	0.68	0.71	0.72	0.44	0.65	0.61
ρ_{oos} (%)	0.59	0.41	0.30	0.01	0.69	0.30	0.27	0.02	0.19	0.57

suggests that linear models like OLS and LASSO consider *Beta* to be the most important characteristic, while GBRT, NN_1, and $GSGP_o$ prioritise *Volatility*; other methods prioritise different characteristics.

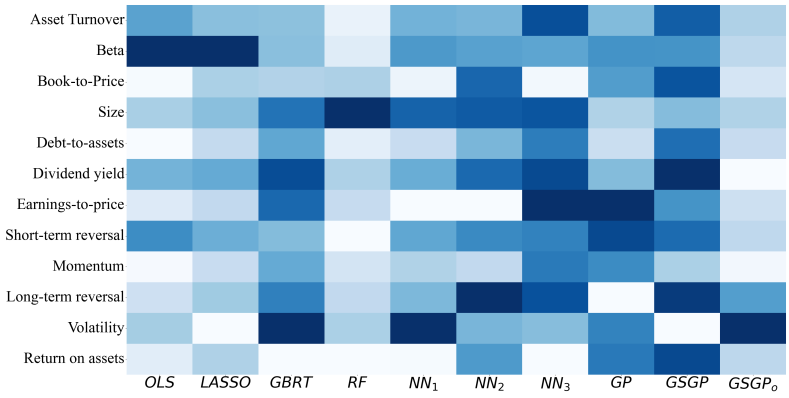

Fig. 5: This figure shows stock characteristics importance across methods. The importance of a characteristic is calculated as the reduction in R^2_{oos} when its values are set to 0, while other characteristics remain fixed. The values are normalised to sum to 1 for relative interpretation. The colour gradient within each column indicates the importance of the characteristics of a particular method, with darker colours representing greater importance.

5.2 Discussion

Our results show that both GSGP and $GSGP_o$ offer advantages over standard GP in terms of forecasting accuracy, with improvements in both forecast errors (see Table 1) and R^2_{oos} (see Table 2), contrary to previous findings suggesting GSGP limited benefits in financial applications [29]. These improvements likely result from periodic model retraining and hyperparameter updates with new data applied here, a strategy not fully explored in prior studies. Additionally, GSGP and $GSGP_o$ achieve better portfolio performance and higher ρ_{oos} than standard GP, with $GSGP_o$ generally outperforming GSGP. The differences between their portfolios performance are likely due to the distinct signals each method capture (see Figure 5). When compared to the remaining methods, the forecast errors of GSGP and $GSGP_o$ remains relatively high (see Table 1) and their values of R^2_{oos} are somewhat lower (see Table 2). Nonetheless, $GSGP_o$ portfolio tends to show competitive performance across non-error-based metrics, ranking near the top-performing models in each aspect, though it might not always come out on top.

Interestingly, OLS portfolio performs well and is difficult to beat, ranking third in Sharpe ratio, with final cumulative return and ρ_{oos} just behind NN_1. This could be partly explained by the smaller predictor set used in this study, as

previous studies has shown that the additional values provided by ML algorithms often comes from larger predictor sets, e.g., 920 predictors in Gu et al. [16] and over 100 predictors in Cakici et al. [7]. However, only 12 predictors, which likely contain fewer non-linear signals, are considered here. Nonetheless, this explanation does not account for all observed outcomes, particularly the inconsistencies in portfolio performance and forecasting errors across many method pairs e.g., $GSGP_o$ against the remaining methods, except GP and LASSO against NNs.

A more plausible explanation for these discrepancies lies in the limitation of MSFE as an optimisation objective. Although perfectly minimised MSFE would ideally capture the correct relative order among stock returns, even low but non-zero MSFE values can still lead to large ranking errors. Because CST strategies prioritise the relative performance of assets over absolute accuracy, MSFE fails to account for important aspect like the order of asset returns. As a result, the portfolio performances likely depends more on the signals each method capture rather than their forecast errors. Ranking metrics such as ρ_{oos} better capture the effectiveness of CST strategies compared to traditional error-based metrics like the modified Diebold-Mariano test and R^2_{oos}. These findings suggest that the current definition of semantics in semantics GP framework overlooks cases where the relative order of outputs is more critical than their precise values.

6 Conclusion and Future Work

We introduce an application of GSGP in developing CST strategies in the U.S. stock market. The predictive power of GSGP and its variant in which the optimal mutation step is used in GSM operator ($GSGP_o$) are compared against standard GP and other common methods used for studying cross-sectional stock returns including OLS, LASSO, GBRT, RF, and NNs. The results show that both GSGP and $GSGP_o$ offer additional value gains over standard GP, providing better forecasting errors, R^2_{oos}, ρ_{oos}, and portfolio performance. Part of the success is attributed to periodic model retraining and hyperparameter updates with new data, previously overlooked. GSGP and $GSGP_o$ forecast errors remain high compared to the remaining methods. However, the portfolio performance of $GSGP_o$ often ranked among the top across metrics, though it may not always be the winner. The findings further suggest that MSFE, commonly used in GSGP and other regression models, may be suboptimal as an optimisation objective for building CST strategies as it overlooks the relative order of asset returns. In future work, we aim to explore the incorporation of semantics that consider order structure among outputs within GSGP framework, potentially leading to more accurate portfolio formation and improved performance in CST strategies.

Acknowledgments. We thank Dr. Linquan Chen, Matus Mrazik, and Jupiter Systematic Equities team for their helpful comments.

Disclosure of Interests. The first author is a PhD student partially supported by Jupiter Asset Management. The views and findings expressed in this paper are solely those of the authors and do not necessarily reflect the opinions or positions of Jupiter Asset Management or its subsidiaries.

A Hyperparameter Tuning

Table A1: This table describes hyperparameter combinations for tuning with validation samples, with all possible combination are formed from a Cartesian product. For example, for an algorithm with hyperparameters $A = \{a_1, a_2\}$ and $B = \{b_1, b_2\}$, the pairs (a_1, b_1), (a_1, b_2), (a_2, b_1), (a_2, b_2) are considered.

	Hyperparameter	Specification	Definition
OLS	none		
LASSO	*alpha*	$\{10^{-5}, 10^{-4}, 10^{-3}\}$	Constant that multiplies the l_1 penalty term
	tol	$\{10^{-5}, 10^{-4}, 10^{-3}\}$	If the updates are smaller than *tol*, checks the dual gap for optimality and continues until it is smaller than *tol*
GBRT	*n_estimators*	{50, 100}	Number of boosting stages to perform
	learning_rate	$\{10^{-2}, 10^{-1}\}$	The contribution of each tree
	maxdepth	{1, 2, 3, 4, 5}	Maximum depth of each individual tree
RF	*n_estimators*	{50, 100, 200}	Number of trees in the forest
	maxdepth	{1, 2, 3, 4, 5}	Maximum depth of each individual tree
	max_samples	0.75	Proportion of samples to draw to train each base estimator
	max_features	sqrt	Number of features to consider when looking for the best split
NN_1-NN_3[a]	*solver*	sgd	Solver for weight optimisation
	learning_rate_init	$\{10^{-2}, 10^{-1}\}$	Initial learning rate
	batch_size	500	Batch size
	validation_fraction	0.2	Proportion of training data to set aside as validation set for early stopping
	tol	$\{10^{-6}, 10^{-5}, 10^{-4}\}$	When the loss or validation score is not improving by at least *tol*, the current learning rate is divided by 5
	alpha	$\{10^{-5}, 10^{-4}, 10^{-3}\}$	Strength of l_2 regularisation term
	max_iter	400	Number of epochs
	ensemble	5	Number of independent random seeds used for model training
GP	*npop*	200	Number of population
	ngen	{10, 20, 40}	Number of maximum generation
	maxdepth	{10, 20, 30}	Maximum depth of an individual after applying genetic operators
GSGP	*npop*	1	Number of population
	ngen	{20, 50, 100}	Number of maximum generation
	max_gen_depth	{3, 5, 7}	Maximum depth for each tree generation
	ms	$\{10^{-3}, 10^{-2}, 10^{-1}\}$	Mutation step
	nrun	10	Number of independent runs
$GSGP_o$	*npop*	1	Number of population
	ngen	{20, 50, 100}	Number of maximum generation
	max_gen_depth	{3, 5, 7}	Maximum depth for each tree generation
	ms	optimal	Mutation step
	nrun	10	Number of independent runs

[a] The hidden layers for neural network with 1, 2 and 3 hidden layers are (32), (32, 16), and (32, 16, 8) respectively

References

1. Bakurov, I., Muñoz Contreras, J.M., Castelli, M., Rodrigues, N., Silva, S., Trujillo, L., Vanneschi, L.: Geometric semantic genetic programming with normalized and standardized random programs. Genetic Programming and Evolvable Machines **25**(1), 6 (2024)
2. Banz, R.W.: The relationship between return and market value of common stocks. Journal of financial economics **9**(1), 3–18 (1981)
3. Basu, S.: Investment performance of common stocks in relation to their price-earnings ratios: A test of the efficient market hypothesis. The journal of Finance **32**(3), 663–682 (1977)
4. Becker, Y.L., Fei, P., Lester, A.M.: Stock selection: An innovative application of genetic programming methodology. Genetic Programming Theory and Practice IV pp. 315–334 (2007)
5. Becker, Y.L., Fox, H., Fei, P.: An empirical study of multi-objective algorithms for stock ranking. Springer (2008)
6. Bonne, G., Wang, J., Zhang, H.: Machine learning factors: Capturing nonlinearities in linear factor models. MSCI Research (2021)
7. Cakici, N., Fieberg, C., Metko, D., Zaremba, A.: Machine learning goes global: Cross-sectional return predictability in international stock markets. Journal of Economic Dynamics and Control **155**, 104725 (2023)
8. Cerqueira, V., Torgo, L., Smailović, J., Mozetič, I.: A comparative study of performance estimation methods for time series forecasting. In: 2017 IEEE international conference on data science and advanced analytics (DSAA). pp. 529–538. IEEE (2017)

9. De Bondt, W.F., Thaler, R.: Does the stock market overreact? The Journal of finance **40**(3), 793–805 (1985)
10. Diebold, F.X., Mariano, R.S.: Comparing predictive accuracy. Journal of Business & Economic Statistics **13**(3), 253–263 (July 1995), https://ideas.repec.org/a/bes/jnlbes/v13y1995i3p253-63.html
11. Drobetz, W., Otto, T.: Empirical asset pricing via machine learning: evidence from the european stock market. Journal of Asset Management **22**, 507–538 (2021)
12. Fortin, F.A., De Rainville, F.M., Gardner, M.A.G., Parizeau, M., Gagné, C.: Deap: Evolutionary algorithms made easy. The Journal of Machine Learning Research **13**(1), 2171–2175 (2012)
13. Giglio, S., Kelly, B., Xiu, D.: Factor models, machine learning, and asset pricing. Annual Review of Financial Economics **14**(1), 337–368 (2022)
14. Gonçalves, I., Silva, S., Fonseca, C.M.: On the generalization ability of geometric semantic genetic programming. In: Genetic Programming: 18th European Conference, EuroGP 2015, Copenhagen, Denmark, April 8-10, 2015, Proceedings 18. pp. 41–52. Springer (2015)
15. Goyal, A., Jegadeesh, N.: Cross-sectional and time-series tests of return predictability: What is the difference? The Review of Financial Studies **31**(5), 1784–1824 (2018)
16. Gu, S., Kelly, B., Xiu, D.: Empirical asset pricing via machine learning. The Review of Financial Studies **33**(5), 2223–2273 (2020)
17. Han, Y., He, A., Rapach, D.E., Zhou, G.: Cross-sectional expected returns: New fama–macbeth regressions in the era of machine learning. Review of Finance p. rfae027 (2024)
18. Hansen, L.K., Salamon, P.: Neural network ensembles. IEEE transactions on pattern analysis and machine intelligence **12**(10), 993–1001 (1990)
19. Jaggi, M.: An equivalence between the lasso and support vector machines. Regularization, optimization, kernels, and support vector machines pp. 1–26 (2013)
20. Jegadeesh, N., Titman, S.: Returns to buying winners and selling losers: Implications for stock market efficiency. The Journal of finance **48**(1), 65–91 (1993)
21. Kim, M., Becker, Y.L., Fei, P., O'Reilly, U.M.: Constrained genetic programming to minimize overfitting in stock selection. Genetic Programming Theory and Practice VI, Genetic and Evolutionary Computation pp. 179–195 (2008)
22. Kotanchek, M., Smits, G., Vladislavleva, E.: Trustable symbolic regression models: using ensembles, interval arithmetic and pareto fronts to develop robust and trust-aware models. Genetic programming theory and practice V pp. 201–220 (2008)
23. Lewellen, J.: The cross-section of expected stock returns. Critical Finance Review **4**(1), 1–44 (2015)
24. Liu, Y., Zhou, G., Zhu, Y.: Maximizing the sharpe ratio: A genetic programming approach. SSRN Electronic Journal. https://doi.org/10.2139/ssrn **3726609** (2020)
25. Lo, A.W., MacKinlay, A.C.: When are contrarian profits due to stock market overreaction? The review of financial studies **3**(2), 175–205 (1990)
26. Loh, W.Y.: Classification and regression trees. Wiley interdisciplinary reviews: data mining and knowledge discovery **1**(1), 14–23 (2011)
27. Martins, J.F.B., Oliveira, L.O.V., Miranda, L.F., Casadei, F., Pappa, G.L.: Solving the exponential growth of symbolic regression trees in geometric semantic genetic programming. In: Proceedings of the genetic and evolutionary computation conference. pp. 1151–1158 (2018)
28. Masters, T.: Practical neural network recipes in C++. Morgan Kaufmann (1993)

29. McDermott, J., Agapitos, A., Brabazon, A., O'Neill, M.: Geometric semantic genetic programming for financial data. In: Applications of Evolutionary Computation: 17th European Conference, EvoApplications 2014, Granada, Spain, April 23-25, 2014, Revised Selected Papers 17. pp. 215–226. Springer (2014)
30. Moraglio, A.: An efficient implementation of gsgp using higher-order functions and memoization. In: Semantic Methods in Genetic Programming, Workshop at Parallel Problem Solving from Nature (2014)
31. Moraglio, A., Krawiec, K., Johnson, C.G.: Geometric semantic genetic programming. In: Parallel Problem Solving from Nature-PPSN XII: 12th International Conference, Taormina, Italy, September 1-5, 2012, Proceedings, Part I 12. pp. 21–31. Springer (2012)
32. Ni, J., Drieberg, R.H., Rockett, P.I.: The use of an analytic quotient operator in genetic programming. IEEE Transactions on Evolutionary Computation **17**(1), 146–152 (2012)
33. Nicolau, M., Agapitos, A.: Choosing function sets with better generalisation performance for symbolic regression models. Genetic programming and evolvable machines **22**(1), 73–100 (2021)
34. Nicolau, M., McDermott, J.: Genetic programming symbolic regression: What is the prior on the prediction? Genetic Programming Theory and Practice XVII pp. 201–225 (2020)
35. Pawlak, T.P., Krawiec, K.: Competent geometric semantic genetic programming for symbolic regression and boolean function synthesis. Evolutionary computation **26**(2), 177–212 (2018)
36. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al.: Scikit-learn: Machine learning in python. the Journal of machine Learning research **12**, 2825–2830 (2011)
37. Poh, D., Lim, B., Zohren, S., Roberts, S.: Building cross-sectional systematic strategies by learning to rank. The Journal of Financial Data Science **3**(2), 70–86 (2021)
38. Schnaubelt, M.: A comparison of machine learning model validation schemes for non-stationary time series data. Tech. rep., FAU Discussion Papers in Economics (2019)
39. Vanneschi, L., Silva, S., Castelli, M., Manzoni, L.: Geometric semantic genetic programming for real life applications. Genetic programming theory and practice xi pp. 191–209 (2014)
40. Yao, X.: Universal approximation by genetic programming. Foundations of Genetic Programming pp. 66–67 (1999)
41. Zhang, Y., Bhattacharyya, S.: Genetic programming in classifying large-scale data: an ensemble method. Information Sciences **163**(1-3), 85–101 (2004)