Cooperative PSO with Spatially Meaningful Neighbors

Peter Wilson

Supervised by Beatrice M. Ombuki-Berman

Submitted in partial fulfillment
of the requirements for COSC 4F90

Department of Computer Science

Brock University
St. Catharines, Ontario

Abstract

Genetic programming (GP) has been shown to be a strong candidate when performing symbolic regression. In doing so, mathematical functions can be approximated using an evolutionary "survival of the fittest" technique, based on a provided language of functions and terminals. This methodology is applied throughout this research in an attempt to use symbolic regression as a tool for forecasting economic welfare. Various languages are considered in an effort to analyse performance in price forecasting. Additionally, the size ratio of training data to testing data is examined in hopes of finding value that yields comparatively better results.

Considered languages include a simple mathematical operator based language, as well a more advanced language that includes time-lagged statistical operators, as well as relevant financial data from the past. It was found that, while languages consisting solely of mathematical operators are able plot general patterns of growth, in order to truly capture the volatility seen day to day in the stock market, more advanced statistical operators and financial data must be considered. Through an ensemble learning approach, forecasted data never before seen by the GP system was accurately predicted within $< x >$ standard deviations from the target function, and is shown to yield most profitable investments $< y >$ days after the initial training period. Additionally, it was found that a training data to testing data ratio of $< a : b >$ yielded the highest long term forecasting results. This is attributed to the relationship between $< variable1 >$ and $< variable2 >$, specifically their correlation over time.

Contents

List of Tables

List of Figures

1 Introduction

The use of artificial intelligence to forecast stock prices is a widely studied field, with many major contributions from a variety of perspectives. Much research has applied genetic programming (GP) to this domain, and a many varying methodologies have been studied. This provides very contrasting results among literature. In general, GP approaches appear to yield improved results over traditional investment strategies, such as the common buy and hold, or day trading.

GP as it applies to financial forecasting is hindered by the fact that it is common for researchers in this field to not disclose information regarding GP systems or language definitions due to personal benefit from their findings, or corporate investments backing their studies. Due to the large number of factors at play when studying GP, particularly in financial applications, it is very important to consider exactly how parameter decisions will affect your results. Important parameters in the financial field include timing and outcome of quarterly reports, analysis of industry average indicators, changes relative to similar markets, and volume of trade. Important parameters in GP include the GP language, population size, number of iterations, mutation and crossover rates, mutation and crossover techniques, fitness algorithms, and the given language.

As GP is a probabilistic system, a single run cannot be considered statistically significant in its approximations of a given function. Instead, a stacking, or "Ensemble Learning" method will be used as a way to aggregate average predictions of the future. This will be used to provide a range of predictions of future welfare, including minimum, maximum, and average approximations across a number of individuals. A programmatic approach to the detection of outliers in ensemble methods will be discussed, as well as a comparison of parameters, squared errors in fitness calculations, and language design will be discussed.

This research aims to consider GP as a technique for financial forecasting while attempting to address certain gaps in the field. The goal is not to actualize a trading strategy and earn a positive return on investment, rather it is to forecast stock prices with accuracy and robustness in uncertain economic conditions. Additionally, new statistical methods that may prove beneficial in using GP as a tool for price forecasting will be explored. Multiple language variations will be considered and tested for accuracy in both short and long run economic forecasting. First, a purely mathematical language will be considered and applied to aggregate indexes, such as S&P 500 and Dow Jones. Comparatively, an enhanced language consisting of informative financial and statistical functions will be contrasted in performance with the purely mathematical language. Both languages will also be analyzed for quality of predictions based on the length of training data compared to the length of testing data.

2 Background

2.1 Genetic Programming

2.1.1 Overview

Inspired by nature, GP takes an evolutionary perspective on machine intelligence. It is a variation on Genetic Algorithms, that derives from patterns commonly seen in wild animals. In this scenario, small deviations from normality can yield significant improvements in a population of a particular species over a long enough time. These differences allow a species adapt to new environments and challenges, and as it applies to computer programming, approximate a solution to a given problem without an exhaustive search of all potential solutions. Problem solutions are represented as a by means of a defined language.

Similar to general genetic algorithms, wherein solutions are chromosomes, a language defines functions and terminals as attributes of an individual that are combined and altered throughout the course of evolution to eventually lead towards some ideal solution based on a measure of strength. This measure of strength is referred to as fitness, and is denoted by some heuristic function $f(x)$. There is no best general heuristic function, as a heuristic must be problem specific. In the case of path finding GP systems, the heuristic could be represented as absolute distance to a goal. The general algorithm for GP is shown in detail in table 1.

Table 1: GP Algorithm Pseudo Code

```
$pop \leftarrow randomPopulation(popSize)$;
**for** $gen \leftarrow 0, maxGen$ **do**
    $evaluateFitness(pop)$;
    $parents \leftarrow selectBest(pop)$;
    $children \leftarrow applyCrossover(parents)$;
    $applyMutation(children)$;
    $gen \leftarrow gen + 1$;
**end for**
**return** $bestIndividual$;
```

2.1.2 Parameters

System parameters decide entirely the ability of a GP. Crossover and mutation are two standard operations among GP that behave analogously to their real world counter parts. Between generations, candidate solutions will breed, sharing features of their chromosomes with another and creating a new individual from the combined result. This is what is referred to as crossover. Similarly, mutation modifies the chromosome of an individual however this function is asexual, in that it does not require two parent candidates to be performed.

Talk about Adjusted and Standardizzed fitness (also hits, and about human interpretation)

2.1.3 Symbolic Regression

It is common that GP systems applied to stock selection will either generate rules to follow given circumstances or generate perform a regression to create a function that will accurately predict previously unknown inputs. Common econometric techniques are able to more closely be compared to a given mathematical function as a result of a GP run to quantify levels of efficiency and errors. The general approach in either case, however, is similar. A system must be trained on selected data before being applied to unknown data and verified. It is hoped that the training period provides accurate example behaviour of how the economy generally behaves, so that the returned function can remain robust given unknown data. It is for this reason that the choice of training data is very important, as it must be representative of typical economic behaviour. Many econometric techniques that approach price forecasting use regression, and many of these techniques also use mean squared errors as an indicator of strength in a specified model. For symbolic regression problems in GP then, the heuristic function will be defined as the squared residuals between the target data and the models specification.

2.1.4 Defining a Language

Using this approach, programs that adhere to a specified language can evolve. More specifically, a language is defined as a Function Set and Terminal Set. Functions are n-ary operations defined as taking parameters of a certain type. Terminals are parameters to a function, or constant value themselves. For example, the + function is a binary function, so can be written as $+(a, b)$. In this case, a and b could themselves be functions or terminals. A terminal could be an initially randomly generated value, referred to as ephemeral random constants (ERC). This exists throughout an entire GP run without variation. Terminals can also be represented as a stream of values. As an example, a terminal x could simply exist in the range of integers $[0, 100]$, but could also be represented as a much more complicated series like real GDP between 2001-2004. In this case, variables can be time dependent. This means that all functions and terminals that exist as a time-series must be sampled within the same time period by the GP system each generation. In doing this, significant improvements can be made to a language as GP is now considering data across time.

2.1.5 Data Representation

Functions can use a tree-based representation and any given individual will exist initially as a random combination of these functions and terminals. Over time, they are ranked for their performance against a specific problem and set of training data. Much like in nature, the individuals breed and mutate, sharing genetic attributes and creating entirely new ones probabilistically. Given enough generations, strong attributes will shine through and can represent significant features of the problem itself. Crossover and mutations take place at leaf nodes in the tree-based representation, ensuring syntactic correctness of candidate solutions after a transformation.

2.1.6 Tree Generation

The choice of constraining tree size is something that is discussed by John R. Koza [1] extensively. It is said that neither the *full* or *grow* approach to tree generation will provide performance on their own, and instead Koza proposed what is referred to as *ramped half-and-half.* This method creates half of the initial population using the full method, where all leaf nodes are at the same depth, and half using the grow method, where functions and terminals are selected randomly (similar to full) however leaves are not forced to be at any particular depth, allowing for a variety of shapes and sizes in trees that could be very unbalanced. Notice that both of these methods build trees to a user-specified depth. This choice seems to provide performance in generic GP systems, however given that Koza's commentary on the performance of constrained trees is ignorant of the domain specific problem of symbolic regression applied to economic data, appropriate tree size is only another parameter that will require fine tuning to understand more thoroughly in economic analysis.

2.2 Ensemble Learning

A common approach in machine learning is to allow for multiple opinions when making a decision. This allows for safer approximations of problems, particularly those that are probabilistic such as GP. Ensemble learning can be implemented quite simply in GP by running the system multiple times concurrently, and aggregating the results. Kalyan *et al.* [2] have chosen to do this, while simultaneously altering parameters for different runs. This allows a diverse set of models that will have noticeably different outcomes, and thus a strong, safe prediction across the ensemble.

Some ensemble learning techniques in GP include average ensemble prediction (AVE), median average model(MAD), and adaptive regression mixing (ARM) [2]. AVE and MAD are similar in nature, finding the average or the median among the discrete set of models respectively. The ARM ensemble approach is more complex, assigning weights to each model to represent how important its prediction should be in the aggregated result. AVE and MAD approaches follow logically, given the problem domain of finance. They will be used throughout the experiments in this paper, labeled appropriately when either, or both, are shown in a figure.

It is not uncommon in ensemble learning to use entirely different machine learning techniques to provide the highest level of diversity of models for a single problem. Symbolic regression is a particularly flexible problem, and as such is open to many such techniques. GP is the only tool that will be used in this research, using 20 separate models executed concurrently on the same dataset as the basis of experimentation. Kalyan *et al.* [3] use many cloud-based platforms for comparison to their own system, which also makes use of feed forward neural networks, multi-objective genetic programming, and optimized multi-objective genetic programming. These systems are all run in parallel, and Kalyan goes on to discuss different ensemble fusing techniques as well as best approaches for concurrent division of datasets.

¡section on outliers¿

2.3 Financial Forecasting

¡include forecasting without mention of GP here. Use paper by Nalia¿

GP will have a math based language that will try to best recreate a sequence of data, minimizing the distance between the curves until a strong fit has been achieved. In econometrics, there are many different approaches to regression problems, and many statistical tools that can be used for informative analysis. Many approaches require specifying an underlying static structure for the model, but this will not be the approach taken as GP is inherently dependant on diversity in structure and form of its individuals.

To aid the GP system in forecasting financial data, it is wise to add financial and statistical language elements to the function and terminal sets. This follows in the logic any person could take in trying to forecast the stock market independently. Rather than attempt to do so using only math, one would also incorporate historical data and relevant statistical information in their estimates. This ensures that GP's information set at as knowledgeable as possible.

3 Literature Review

$$addgeneralGPsymbolicregressionhere(nofinance)$$

Symbolic regression problems are sensitive to the role that a large training dataset will play on a system, as well as how known features of a dataset will manipulate results. For their research, Kim *et al.* [4] chose the S&P 500 index from January 1990 to December 2006. This period of 16 years provides a surge of growth in multiple markets, as the S&P 500 is known to be a strong reflection of real economic welfare. The choice to end in 2006 likely is to avoid modeling the undesirable economic state of 2007-2009. It is argued that severely abnormal economic behaviour is not ideal for building a robust system. Recessions are seen as uninformative in that they model unusual behaviour, which in practice tends to skew econometric models.

Having a model overfit training data is a common occurrence among symbolic regression problems, and machine learning in general. This is done by too closely fitting a particular data set that will map that exact behaviour onto the testing data. This ignores potentially informative attributes in the data, and instead memorizes a pattern seen in training. Kim *et al.* [4] have approached this problem by creating an objective function, dividing the generation of a system into three steps. First is the training step, where the selected period of training data is divided into two subsections. Here, each half of the data is given to two separate runs of the GP system and are compared and merged upon completion. An interesting variation of this part of the process would be running two independent test GP systems on different data sets, perhaps one containing a recession and another with more usual economic behaviour. This could possibly point towards some merged behaviour that, on average, can perform well even under poorer circumstances. After the runs have complete, the best candidates in each GP are then run again on the remaining half of the data, along with new random candidates. Finally, in the validation step, the best models performance are evaluated and analysed on entirely new data.

It is uncommon for financial researchers to reveal specifics regarding their GP implementation, or even more so their specific fitness functions. However, Kim [4] has provided this detailed and problem specific guide to structure the creation of a strong GP that is resilient to overfitting of data. The key to the robustness against overfitting in their experience is that the chosen tree depth is constrained to substantially beyond what is considered typical. Many other researchers such as John Koza [5] have spoken on this topic, using values larger than Kim has chosen. This is an interesting choice, as a large tree can allow a GP to be "lazy" and select less promising or factually representative trees that just yield a high fitness, while small trees can force more representative functions that do not allow too much detail, ignoring noise in data. Conversely, small tree sizes can restrict possibly good solutions, while large tree sizes can leave large portions of the tree completely useless, a common occurrence normally referred to as bloat. An example of this is if the given language contains a boolean function, but GP has used it in such a way that it is always true, or always false. Anything beneath this node in the tree is dead weight, but still renders a marginally higher fitness, acting as a safety net, and is kept.

In the work of Korns [6] he focuses on constrained symbolic regression problems and aims to challenge commonly held beliefs that are considered the norm in academia, as well as aggregate multiple ideas he believes to be more accurate than others. The particular focus involves symbolic regression on a large scale wherein economic applications, it can be expected that there will be many large datasets. This depends more so on the chosen granularity and length of training and verification data, as in data-rich environments Korns discusses the different possible sampling methodologies that can be taken in opposition to sampling an entire dataset. Korns chooses to use 1250 days of data for 800 common stocks as training data. This resulted in a total of 1,000,000 data entries. A large factor regarding his decision is the time it takes to process the information and retrain his system regularly, as the expected time to finish training periods is 50 hours. In doing this, his goal is to analyze how well the standard population size, crossover rate, etc. will work in large scale problems, and consider alternate values for these parameters.

The training process described is simpler than what is discussed previously by Kim [4], in that the process is less rigorous, given the quantity of data. Rather, the system is run on 9 test case formulas ranging in difficulty, and even going to the extent of adding random noise. Economic data is already very noisy, and it's hard to pinpoint whether or not the data may have underlying factors attributing to its behavior that may not have been accounted for. Random noise will likely not be necessary to add to a model focused on econometric applications, even if it adds a layer of complexity to training. The intentional addition of noise to data is not standard practice among researchers who have focused on econometric applications

The idea of a hybrid grammar and tree-based GP system is discussed as well. Korns' [6] work experiments on a variety of test case formulas, ranging from simple linear equations to complex ellipses, as well as mixed formulas that use boolean operators. This shows some promising results on the simpler functions, yet it thought possible that the addition of noise made it difficult to come up with such results in more complex scenarios. To account for the magnitude of the problem, he also doubles the number of generations traditionally used in GP systems. It is hard to say whether or not a specific number of generations is cutting a problem short of finding a good solution, or simply wasting CPU time. This is a possible indicator that a large number of generations should, at the very least, be investigated, as improvement may still be possible at this point in the execution.

Becker *et al.* [**?**] take a stronger economics perspective in their work than in many of the previously mentioned papers. Despite the problem of symbolic regression being purely mathematical, it is important to remember that any additional, problem specific, information that can guide a GP to a more adequate and robust solution is ideal. Becker chooses to analyze the S&P 500 index, as it is an aggregate of information on the overall economic well being at any time for 500 large companies traded publicly on the NASDAQ. Attributes associated with an aggregate index in this case are argued as being representative of generic economic welfare, and in this case bring substantial value and robustness to a symbolic regression model.

Korns [6] mentions the simpler data points such as open, high, low, close, volume, earnings per share (EPS), and analyst rating as being valuable in price forecasting. Despite the simplicity of these indicators, Korns argues that they are essential, as they act as a baseline of data describing market behaviour. This contrasts the opinions of Kim's [4] work in discussing the overfitting of GP systems, mentioned previously. The nature of simplicity in these indicators is a double edged sword, as they are also the most widely available. Even the simplest stock portfolio application will track this information. Although specific relationships are not shown, to maintain trade secrets, descriptions are associated with each variable in Kim's [4] description of their GP system. The chosen indicators are noticeably more sophisticated, using variables such as dividend yield, profit and growth margins, and net debt ratios. Additionally, analyst long-term estimates have been chosen despite a discussed apparent variability in these ratings.

Kaboudan [7] explores the topic of actualizing a GP system to make real trades thoroughly. Very strong econometric methods are used to analyze the performance of the given GP. It is hypothesized that a single day trading strategy (SDTS) is likely the most profitable, however specifics regarding his language or parameters were not mentioned. This study finds that by analyzing previous data up to the present day it is possible to make accurate predictions regarding tomorrow's performance. Any efforts exceeding a day seemed to yield next to no return, or at the very least no statistically significant advantage of regular trading strategies, such as buy and hold.

Kaboudan [7] notes that this method comes with a set of drawbacks. If the forecasted increase is not enough to justify the risk, as well as the cost of performing the trade, then of course a trade will not be made. He also adds that if the actual prices fall below the forecasted low, or above the forecasted high, then again a trade will not be made. It is understood that this is a safety precaution because otherwise our forecasted profit cannot be relied upon. However it is not necessarily true that an opening price lower than the forecasted opening price is necessarily bad. It can be a sign of a failing stock, but also a sign of an undervalued stock. The distinction between the two, while difficult to make, can be very important. Kaboudan [7] presents all of the logistics regarding actualizing a GP that forecasts stock prices well, but does not provide information regarding implementation of the system itself. Despite this, econometric analysis in this work will provide useful techniques towards evaluating performance of a GP.

Similar to the work of Kaboudan [7], Mallick [8] aims to implement a stock trading system based purely on trading rules created in a GP based system. Trading rules create an environment of binary yes/no questions that lead to a buy or sell decision on the chosen stock. He does not include details on the chosen language, but does provide details regarding parameters. With population size he does not deviate from what is considered standard, however the system parameters fall slightly lower than what is seen in alternate literature. This decision is not specifically justified, but it is most likely due to the differing nature of a rule based system over symbolic regression. Similar to Kaboudan [7] comparisons are drawn from the implemented trading strategy to the common buy and hold technique, as well as the MACD (Moving Average Conversion/Diversion) technical indicator. This indicator is a highly praised accurate forecasting method of the general direction a stock is likely to drift towards.

Contrary to many of the previous works, an alternate perspective of approaching this problem is focus on the model itself. The above use a time-period based model. Some used daily data, while others may have used longer time periods in an attempt to aggregate general behaviour and remove granularity from the problem. Gypteau [9] argues that using physical time intervals is a poor structure for the problem at hand, and instead proposes event-driven models. Models that are event-driven, from a high level perspective, require a large amount of analysis in order to automate the process of deciding what is and is not a significant social event that will impact a particular market. This may seem easy in the case of analyzing quarterly reports, but an important aspect to the problem is general market mood. Bollen [10] argues that Twitter is able to accurately predict the stock market based on overall moood, as human emotions profoundly drive our decision making process. In his specific work, he forecasts the mood of Twitter in an aggregate manner daily and compares it to the public's response to the presidential election, and finds very positive results.

Autocorrelation and business cycles are two important concepts that should be recognized. In an economics-oriented GP survey, Navet [11] discusses the statistics behind use a time lag and analyzes the entropy of given stocks to find out if low entropy stocks are more profitable than those with higher entropy. There is little mention of the GP system used, besides a brief introduction to GP as a whole, but the use of time lagged data comparisons is interesting as dependence of today's market values on yesterday's is a highly debated topic among economists.

Business cycles provide a way to analyze short bursts of volatility in an economy, and are notably separate from the general direction of growth. Sudden drops in market valuations do not necessarily imply negative growth, as the following time period may be matched with an equally as substantial boom in the economy. First differences (time lagged data) is one of the few ways we can analyze business cycles in an attempt to differentiate them from real growth in GDP.

The field of genetic programming as it pertains to financial forecasting is rich with ideas of how to create a profitable system that may accurately predict stock prices to some degree. It is hoped that some of the more promising ideas, discussed throughout this paper, will be implemented in the Genetic Programming Portfolio (GPP) system. Different methodologies for both GP and the economic analysis of the problem will be discussed in an attempt to further knowledge of the specific problem of symbolic regression with financial applications.

4 System Design

4.1 ECJ & GP Language

Developed at George Mason University's Evolution Computation Lab, ECJ is a diverse library providing access to classes and interfaces that promote efficient programming. All system parameters are determined at runtime by a user-defined file, which can be read from and altered without the need to rebuild the system. To create an instance of a GP system using ECJ, the user need only provide an implementation file, data type file, and classes representing function and terminal sets.

All evolutionary operations, such as crossover, mutation, elitism, and handling generations are handled by ECJ. All methodologies follow those discussed by John R. Koza [12] by default, but the user is free to override this behaviour, providing a flexible framework. This allows the user to focus entirely on the problem definition and higher level analysis, such as the GP language.

A GP language definition is generally considered stronger when it uses as much problem-specific data as necessary. In the case of symbolic regression, it is required to have a number of mathematical operators however when used for financial forecasting it is also extremely beneficial to provide financial data and statistical analysis functions as well. Throughout the experiments covered in this research, the language will vary.

4.2 Data Processing

To aid in processing important data of system executions, a Python interface has been included in the project implementation. This includes a variety of Python scripts that serve varying purposes, such as graphing data or aggregating it into a format that is easier to manipulate. The choice of Python as a processing language was based in ease of use, as there exists many statistical analysis and graphing libraries.

From the main ECJ execution file, a proxy Java class is called that is responsible for executing the Python scripts and handling their output. At this point, a variety of text files are created that include data on the best of run individuals for every execution, as well as other important statistics like standardized and adjusted fitness. This allows a high level of automation for repetitive analysis tasks as well as aids in formatting for Microsoft Excel spreadsheets.

Consider Figure 1 for further details on system design, particularly the flow of data as it is created and processed by Java, and further processed by the Python interface. Information on the fitness of every generation in every run is stored as intermediate output periodically as raw text files. The Python interface then reads this intermediate output, aggregating it to provide an average look across runs, so that no individual, coincidentally strong run skews perceptions regarding system performance. Additionally, once at the end

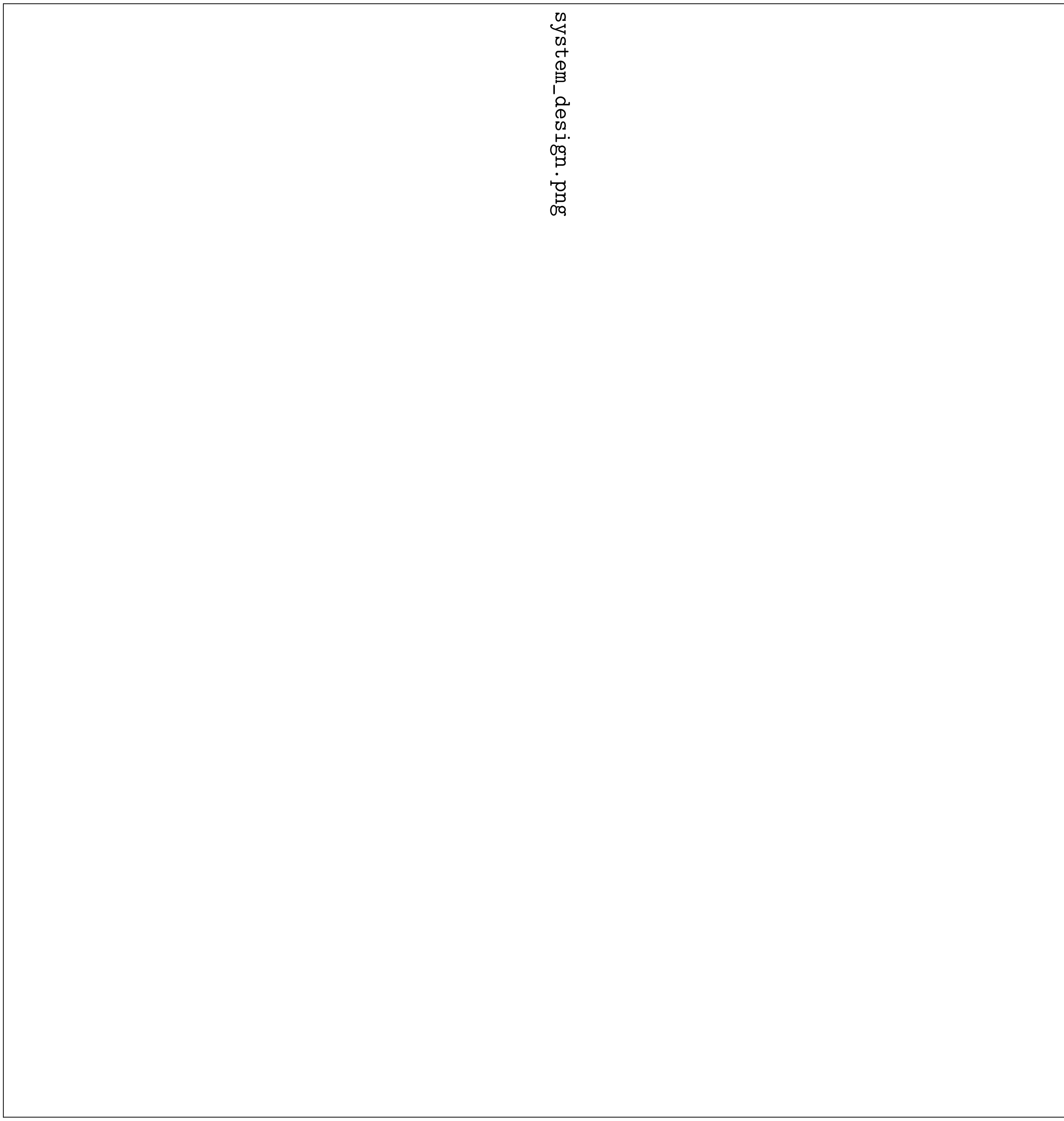

Figure 1: Design of the GPP System.

of each run Java will output extra details regarding the best of run individual into a separate directory. Here, Python sorts them and applies ensemble learning methodologies, such as detection of outliers in the ensemble model, as well as simple average, median, minimum, and maximum information.

5 Experiments

5.1 Preliminary Trials & Data Selection

A preliminary experiment that focused on parameter decisions found noticeably superior results given the parameters shown in Table 2. During the experiment conducted by Kim *et al.* [4] on overfit symbolic regression models, parameter variations were also discussed. Similar results were found as being particularly strong, and so will be used in the following experiments.

Parameter	**Value**
Crossover	90%
Mutation	10%
Elitism	0
Tournament Size	2
Max Tree Depth	17
Pop. Size	1000
Generations	100

Table 2: Ideal Parameters.

Although forecasting very brief, volatile moments in time presents a significant challenge for GP when using symbolic regression, long term trends are where it excels. Different time ranges are considered in the following experiments, in particular the ratio of training data to testing data is explored. Dow Jones data is used between January of 2013 to January of 2016 to provide 3 fiscal years of data to explore. The data provides a wide range of economic behaviour that allows for a diverse GP training set. Additionally, data has been normalized to fall in the range $[0, 1]$ for the purposes of human interpretation as well as simplicity in fitness calculations and defining a numeric range for other variables of significance.

5.2 Experiment 1: Mathematical Language

5.2.1 Setup

In this experiment, the use of simple mathematical operators is explored and contrasted with the use of an additional boolean function, namely *if less than*, or $IFLT$. This additional function is defined in table 3, and the terminal and function sets in the first experiment are defined in Table 4.

Table 3: IFLT Algorithm

```
$IFLT(a, b, x, y)$ :
if $a < b$ then
    return $x$
else
    return $y$
end if
```

Terminal Set	x, ERC
Function Set	$+$, $-$, $\times$, $\div$, log, cos, sin

Table 4: Math Language Definition

Here x represents day number in the data, and ERC represents floating point random values. The ERC value is set at the beginning of each run and is invariant between generations, while x exists in the range $[0, 283]$, given by the number of data points being sampled. Additionally, a fixed ratio of 90% training data to 10% testing data is used in this experiment. An analysis of differing ratios for training and testing data ranges is considered in the experiments that follow.

The boolean function added to the language for these additional experiments is of arity 4, defined as seen in Table 3. The choice of only having an *if less than* function, and not *if less than or equal to* came from the precision required to compare floating point values. The difference in function behaviour is negligible in nearly all circumstances. Additionally, there is no greater-than function because of course $a < b \Rightarrow b > a$, and so any similar additional boolean functions would be redundant.

5.2.2 Results

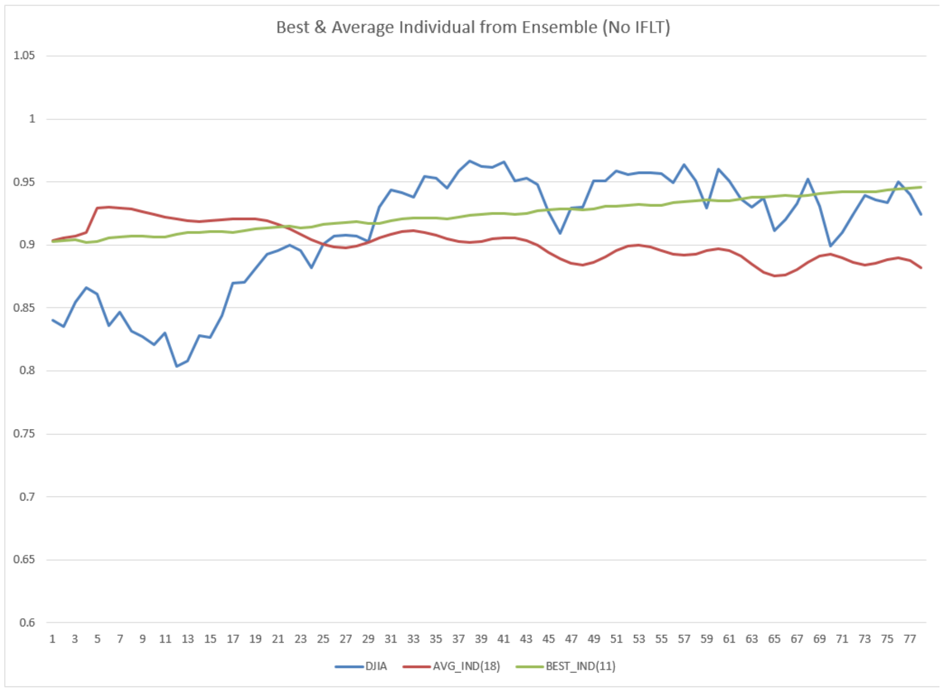

Figure 2: Without IFLT: Best Fit Individual vs. Average Fit Individual.

Figure 2 shows the target data plotted against the strongest individual model in the ensemble, as well as an average model. Figure 3 shows the same, only with the additional boolean function $IFLT$. Both are shown in the testing data range, and highlight interesting features in the target data considering the use of a language based only in math functions. Both individuals have relatively smooth curves with no sudden breaks or highly volatile moments in time, while both individuals using $IFLT$ are nearly perfectly flat. It is also interesting to note that the most fit individual in the entire ensemble without use of $IFLT$ seems to be increasing, relatively linearly, while not the case with the average individual that

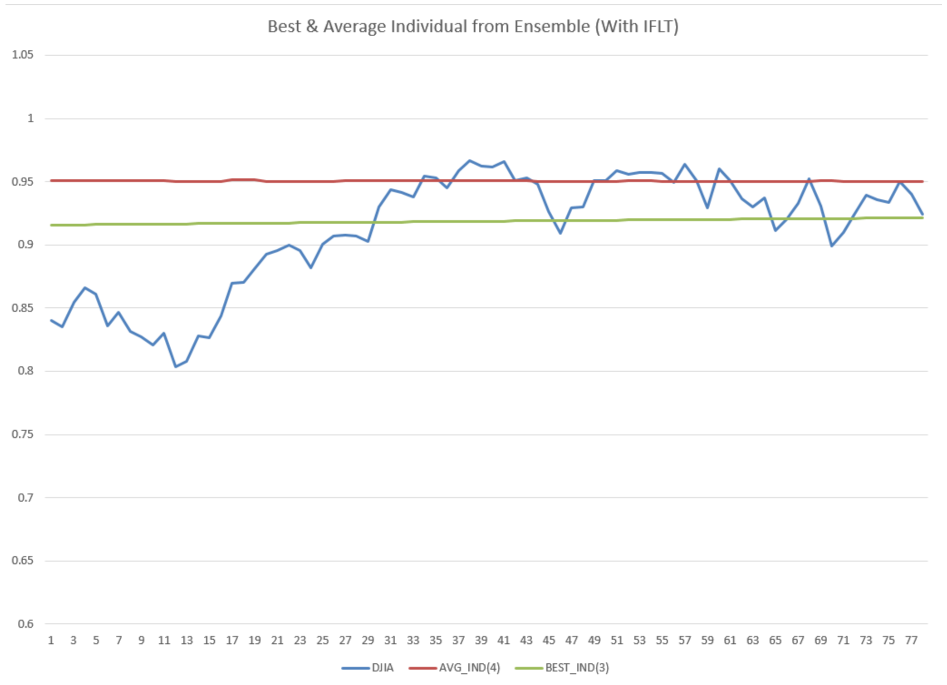

Figure 3: With IFLT: Best Fit Individual vs. Average Fit Individual.

has more fluctuations in a cyclic, downward sloping fashion.

Consider the performance statistics shown for Figures 2 and 3 in Table 5. It is interesting to note that making use of the additional function $IFLT$ did not immediately yield better individuals, and in fact they are marginally worse. This describes the performance of the language in such a way that multiple diverse individual models are generated that may not always agree. This seems like an ideal situation for ensemble learning, as differing opinions among models can improve strength of the overall ensemble due to being more diverse.

Category	**Average Ind. Fitness**	**Best Ind. Fitness**
No $IFLT$	0.251398	0.120458
With $IFLT$	0.279529	0.158227

Table 5: Average Individual & Best Individual Performance

Figures 4 and 5 shows the ensemble using both average and median approaches to fuse individual models, with and without the use of $IFLT$, respectively. Additionally, minimum

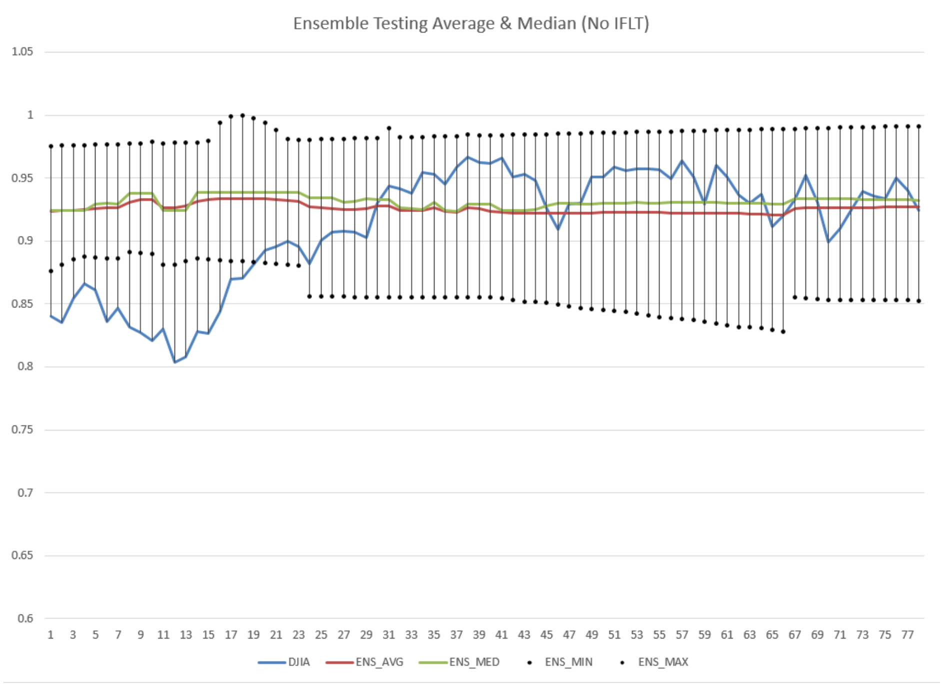

Figure 4: Without IFLT: Average Ensemble vs. Median Ensemble.

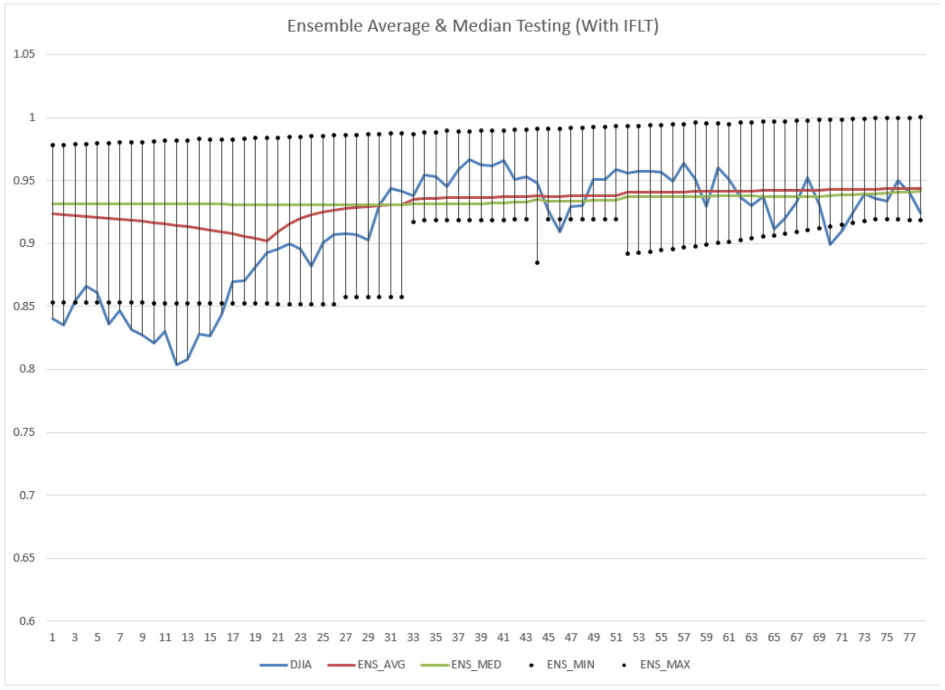

Figure 5: With IFLT: Average Ensemble vs. Median Ensemble.

and maximum data points from individual models are kept and plotted to create high-low lines representative of a forecasting range. It can be seen that both average and median ensemble techniques perform relatively the same, with some initial volatility then tapering off as time goes on. This applies to both ensembles, with and without $IFLT$. Automatic outlier detection is put in place to avoid skewed ensembles. This is very beneficial, particularly for the fusion technique of averaging models. Ideally, fewer outliers will point towards a more intelligent system, so the number of outliers removed is also tracked when calculating the performance fitness of an ensemble in testing.

Table 6 shows the standardized fitness of both the average and median techniques, as well as the number of outliers removed from the ensemble. As both ensembles use the same 2σ outlier detection rule, the number of outliers removed in each ensemble will always be the same. Comparing the language that made use of $IFLT$, it is clear to see that it is superior when using either ensemble fusion technique. This difference is most pronounced when considering ensemble using averages, with the most fit ensemble scoring substantially below all other variants.

Category	**Ensemble (Avg) Fit**	**Ensemble (Med) Fit**	**Outliers Removed**
No $IFLT$	0.194444	0.200982	35
With $IFLT$	0.135833	0.190525	96

Table 6: Average & Median Ensemble Performance

The ensemble performance plot can be seen in Figure 6. This plot shows the standardized fitness improving over all 100 generations, averaged across all 20 individual models. This provides a very good idea of how, on average, the entire ensemble will improve in this time span. A steady, consistent convergence can be seen approaching zero as time goes on. This is expected, and is indicative of a system that is generally converging well over time.

Additional work surrounding the mathematics based language includes the use of a simple automated outlier detection system. In Figures 7 and 8, a comparison between no outlier prevention and an automated method is shown. On the left, it is clear to see

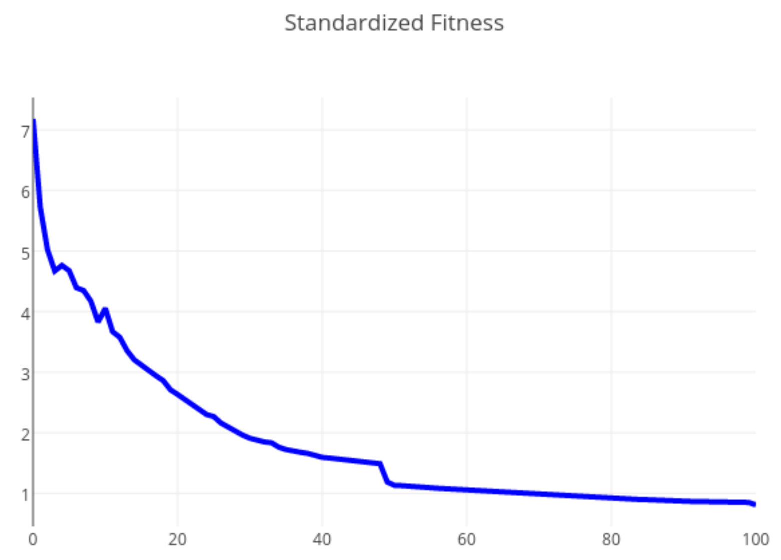

Figure 6: Average ensemble standardized fitness over time.

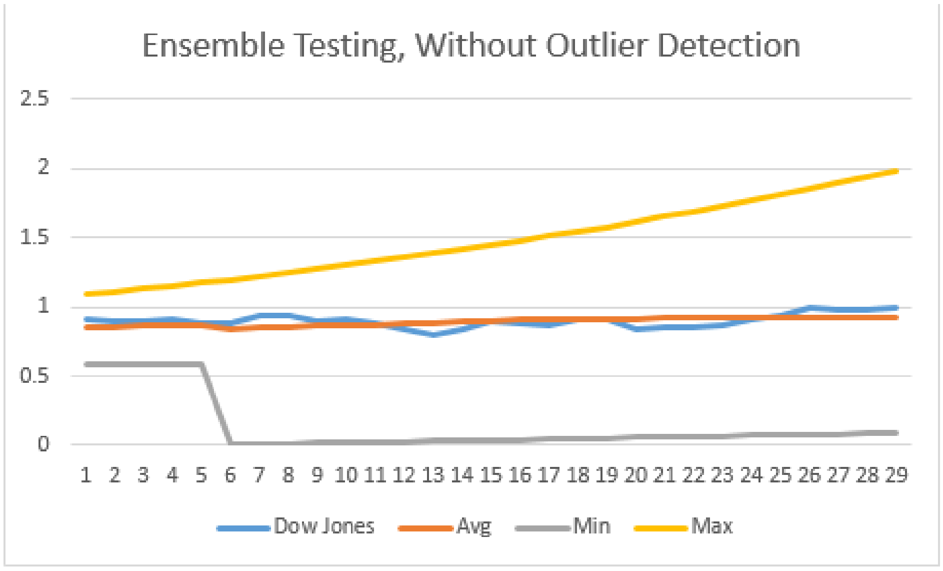

Figure 7: Ensemble testing without outlier detection.

that outliers have skewed the results. A single model in the ensemble has a dramatically lower value over any other model, and thus drags the *min* plot down. This in turn has the possibility of severely effecting the average ensemble prediction, and so needs to be addressed. On the right, a preventative method has been put in place. When aggregating

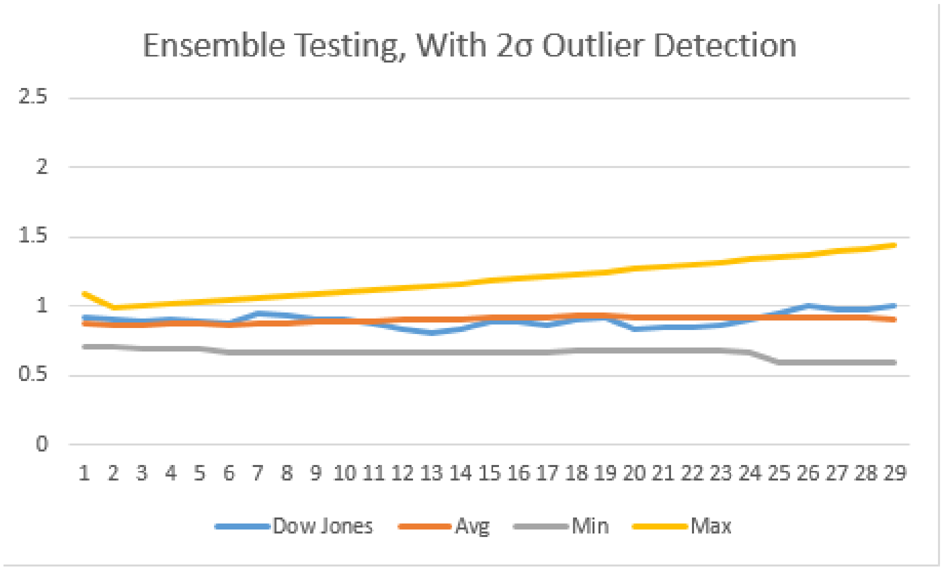

Figure 8: Ensemble testing with 2σ outlier detection.

GP results and building the ensemble, the system will always check if any data point exceeds ± 2 standard deviations, or 2σ, from the average. This 2σ rule provides a range of maximum and minimum values that can models may stray from the average. It can be seen that the upper and lower bound functions surrounding the target data and average ensemble have been greatly improved by implementing this rule. This provides a somewhat statistically sound methodology to a complex problem that protects the ensemble from biased estimates. This is, however, dependent on the chosen data being normally distributed.

5.2.3 Discussion

In the shown results, the comparison between ensembles with and without $IFLT$ is clear. As shown in Table 6, the use of $IFLT$ provides favourable ensemble results regardless of the chosen fusion technique. This is likely because the use of this function provides each individual an alternate outlet to achieve increased volatility, and therefore closer approximate highly volatile stock data. Normally the ensemble must rely on use of cos and sin to achieve this. This third resource allows individuals to create non-differentiable functions that can have sudden breaks, suiting of fitting very jagged information.

A major flaw to using $IFLT$ is bloat. This can behave as a safety net of sorts that allows GP to protect fitness evaluations, even if the parameters to the $IFLT$ are constant. Despite this being a possibility, the ensemble still clearly benefits from the use of this additional function. This is particularly noticeable when using averages across models as the fusion technique to build the ensemble. Consider Figures 4 and 5. The most pronounced difference between the use of $IFLT$ is shown here, particularly when the function is in use in Figure 5. The use of average versus median ensemble fusion techniques perform nearly identically without the use of $IFLT$, which can likely be attributed to the average and median being so closely related when the language doesn't allow non-differentiable functions. This contrasts with using the additional function greatly. The strongest sign of differing features between average and median fusion techniques, when using $IFLT$, is when there is a sudden, positive directional change in the ensemble. This takes place when using average fusion. This is likely due to a model, or models, making use of the $IFLT$ function and creating a threshold that will change the slope of the generated function. This was prominent enough to become a distinct feature in the ensemble, and implies that the use of $IFLT$ is, on average, being picked up by GP rather than being ignored, as though it were not providing valuable.

Although the use of 2σ as a rule to protect against outliers in ensemble learning yields improved results in Figure 8 when compared to no outlier detection in Figure 7. There are drawbacks, however, such as the assumption of data being normally distributed, or the disadvantage of ignoring all outliers without further consideration of what an individual data point may tell you about features of the data. With very specified outlier detection techniques, resilience of a model or ensemble is able to be maximized. This follows with the logic of defining a GP language, in that problem specific modifications will improve a model. Despite this, the 2σ rule is not far from the estimation techniques of Veeramachaneni [2] *et al.* In their work, a method is put in place where the minimum and maximum values will be estimated, and any model that exceeds these bounds is removed before being merged into the ensemble.

A useful aspect of ensemble learning is the ability to analyze how GP is converging to a solution. This can be shown by averaging how the fitness of every model at every generation. Performing the same analysis on a single model would not be as informative, as GP is probabilistic in nature and you could not derive any conclusions from the performance of a single model. This avoids any bias regarding single models that are better by chance. Consider the graphs in Figure 6. Hits, as discussed previously, is a human readable approximation of how many times a candidate has correctly approximated the target data, within some radius. For this reason, it should be taken lightly as an indicator. Adjusted and standardized fitness play the largest role in showing the convergence pattern. A very clear increase over time is visible in the adjusted fitness plot. From here, this shows that the entire ensemble, on average, will have an adjusted fitness of ≈ 0.35. This shows a steady convergence to a solution that clearly yields a strong ensemble fitness using a simplistic language.

Through the results in this experiment, it is clear that the additional boolean function $IFLT$ proves useful. The added flexibility in the language allows individual models to achieve a higher degree of volatility. Although over time the ensemble tapers off to relatively linear estimates, this can be attributed to having heteroscedasticity in residuals, resulting in a decrease in estimate accuracy as time goes on. The increase in performance associated with $IFLT$ creates a substantial addition to the simple math based language, defined in Table 4. Further, the implementation of a programmatic outlier detection system creates a simple, robust way to avoid skewed ensemble estimates. As such, both of these additional features will be included in all remaining experiments.

5.3 Experiment 2: Statistics Language

5.3.1 Setup

With the inclusion of an outlier detection system, and the additional boolean function $IFLT$, there is additional room to continue expanding the language to consider more advanced functions that may create increasingly accurate and robust systems for price forecasting using symbolic regression. Consider Table 7, where additional statistics based functions have been included in the language.

Terminal Set	x, ERC
Function Set	$+$, $-$, $\times$, $\div$, log, cos, sin, $IFLT$, sum, $stdev$, $skew$

Table 7: Math Language Definition

5.3.2 Results

5.3.3 Discussion

5.4 Experiment 3: Financial Language

5.4.1 Setup

5.4.2 Results

5.4.3 Discussion

5.4.4 OLD OLD OLD Results

In Figure 9 the target data is plotted against the strongest individual model in the ensemble, as well as an average model. Both are shown in training and testing data ranges, and highlight interesting features in the target data considering the use of a language based only in math functions. Most noteworthy is the high volatility common between both models early in the training data. Although this doesn't persist as time goes on, it is indicative of common features seen when using cos and sin functions.

1-best-avg-compare.PNG

Figure 9: Best Fit of Ensemble vs. Average Fit Individuals.

Two common approaches to building ensembles is to average the individual models, or use the median at each time interval. Figure 10 shows a comparison using both of these methodologies, plotted against the target data. This is shown only in the testing range, to contrast performance of each method. In using median values, an ensemble becomes very resilient to the harsh effects of outliers skewing data, but this comes at a cost. The ensemble

`1-testing-high-low-median.PNG`

Figure 10: Average Ensemble vs. Median Ensemble.

is inherently ignoring a large number of the models it is comprised of, as it is most likely that only a few select models fall around the median for each time period. This effect can be seen in Figure 10. Although the average and median based ensembles perform similarly, as the GP language becomes more complex it is likely that the median ignoring important attributes of models in the ensemble will become a more severe hindrance in accurate forecasting.

Outliers pose serious problems for ensemble models. In an effort to build a truly robust system, a programmatic attempt at outlier detection and removal must be taken. In Figure 11, a comparison between no outlier prevention and an automated method is shown. On the left, it is clear to see that outliers have skewed the results. A single model in the ensemble has a dramatically lower value over any other model, and thus drags the min plot down. This in turn has the possibility of severely effecting the average ensemble prediction, and so needs to be addressed. On the right, a preventative method has been put in place. When aggregating GP results and building the ensemble, the system will always check if any data point exceeds ± 2 standard deviations, or 2σ, from the average. This 2σ rule provides a range of maximum and minimum values that can models may stray from the average. It can be seen that the upper and lower bound functions surrounding the target data and average ensemble have been greatly improved by implementing this rule. This provides a somewhat statistically sound methodology to a complex problem that protects the ensemble from biased estimates. This is, however, dependent on the chosen data being normally distributed.

Although the use of 2σ as a rule to protect against outliers in ensemble learning yields improved results in Figure 11, it does have drawbacks, such as the assumption of data being normally distributed, or the disadvantage of ignoring all outliers without further consideration of what an individual data point may imply. Many researchers have implemented problem specific outlier detection methods. With very specified outlier detection techniques, resilience of a model or ensemble is able to be maximized. This follows with the logic of defining a GP language, in that problem specific modifications will improve a model. Despite this, the 2σ rule is not far from the estimation techniques of Veeramachaneni [2] *et al.* In their work, an estimation method is put in place where the minimum and maximum values will be guessed. Any model that exceeds these bounds is removed before being merged into the ensemble.

A steady convergence of the ensemble is shown using standardized fitness in Figure 12. This represents the average fitness at the end of each generation, averaged across all

`1-outlier-detection.PNG`

Figure 11: A Comparison of Programmatic Outlier Removal.

Figure 12: Average standardized fitness over time.

20 models created. A smooth convergence plot is indicative of intelligent evolution in the ensemble and in individual models. In this case, it tells us that GP is using the provided functions to fit the curve provided in the target function well.

5.4.5 Discussion

An interesting phenomena commonly seen in symbolic regression occurs when *cos* and *sin* are included in the function set. As a result of having very volatile, noisy data, these functions tend to create many sudden peaks and troughs. This is thought to be an effort to replicate patterns in the target data, despite the generated peaks and troughs regularly exceeding the bounds of the target data.

The problem of appropriate sampling techniques for is raised given the behaviour of these functions. Excessively granular data, despite being more informative in theory, will skew the results of GP. What is common among use of *cos* and *sin* becomes more extreme in the presence of highly granular and noisy data. Additionally, when mean squared errors are used to define the fitness function GP can also take shortcuts and cut directly through the center of the of the target data. This essentially is creating a line of best fit over the data as, rather than take the loss in fitness from attempting to match the seen volatility, it is easier for GP to ignore it. This makes the value of *cos* and *sin* a somewhat debatable topic.

Both of the functions seen in Figure 9 make use of *cos* and *sin*, and show the common associated features of increased volatility. This is particularly true for the stronger individual. It can be seen that neither individual is severely hindered by the use of these functions. This points towards interesting behaviour in how GP is calculating strength of an individual, especially in the absence of more advanced functions or terminals. The functions *cos* and *sin* present the only way that GP is able to approximate any reasonable level of volatility in the target data. It can be seen that as the target data increases in volatility over time, this behaviour becomes less beneficial, as the model resorts to creating a line of best fit instead. An additional thought to keep in mind is how versatile GP can be with the use of a relatively simple language. Although the exaggerated volatility of these functions exists in the first month of data, they still must be the most fit of the candidates within the population. This points in a positive direction for the use of a more advanced language.

A useful aspect of ensemble learning is the ability to analyze how GP is converging to a solution. This can be shown by averaging how the fitness of every model at every generation. Performing the same analysis on a single model would not be as informative, as GP is probabilistic in nature and you could not derive any conclusions from the performance of a single model. This avoids any bias regarding single models that are better by chance. Consider the graphs in Figure 12. Hits, as discussed previously, is a human readable approximation of how many times a candidate has correctly approximated the target data, within some radius. For this reason, it should be taken lightly as an indicator. Adjusted and standardized fitness play the largest role in showing the convergence pattern. A very clear increase over time is visible in the adjusted fitness plot. From here, this shows that the entire ensemble, on average, will have an adjusted fitness of ≈ 0.35. This shows a steady convergence to a solution that clearly yields a strong ensemble fitness using a simplistic language.

5.4.6 A Boolean Function

Table 8: IFLT Algorithm

```
$IFLT(a, b, x, y)$ :
if $a < b$ then
    return $x$
else
    return $y$
end if
```

The boolean function added to the language for these additional experiments is of arity 4, defined as seen in table 8. The choice of only having an *if less than* function, and not *if less than or equal to* came from the precision required to compare floating point values. The difference in function behaviour is negligible in nearly all circumstances. Additionally, there is no greater-than function because of course $a < b \Rightarrow b > a$, and so any similar additional boolean functions would be redundant.

5.5 Statistics-based Language

A reference to figure 13

5.5.1 Use of Boolean Functions

5.6 Training to Testing Data Size Ratios

5.6.1 1-to-1: Keeping Things Even

5.6.2 3-to-1: The Rule of Thirds

5.6.3 95-to-5: Resourceful Training

5.7 Financial Functions

Table 9: Example table.

1	First entry.
2	Second entry.
3	Third and final entry.

2-single-outlier-detection.PNG

Figure 13: Outlier Detection and Removal With Statistics Language.

6 Conclusion

A Preliminary Trials

References

[1] John R. Koza. *Genetic Programming: On the Programming Computers by Means of Natural Selection*. MIT Press, 1992.

[2] Kalyan Veeramachaneni, Owen Derby, Una-May O'Reilly, and Dylan Sherry. Learning regression ensembles with genetic programming at scale. *Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation*, pages 1117–1124, 2013.

[3] Kalyan Veeramachaneni, Ignacio Arnaldo, Owen Derby, and Una-May O'Reilly. Flexgp: Cloud-based ensemble learning with genetic programming for large regression problems. *Journal of Grid Computing*, 13:391–407, 2015.

[4] Minkyu Kim, Ying L. Becker, Peng Fei, and Una-May O'Reilly. Constrained genetic programming to minimize overfitting in stock selection. *Genetic Programming Theory and Practice*, 6:178–194, 2009.

[5] Ellery Fussell Crane and Nicholas Freitag McPhee. The effects of size and depth limits on tree based genetic programming. *Genetic Programming Theory and Practice*, 3:223–240, 2006.

[6] Michael F. Korns. Large-scale time-constrained symbolic regression. *Genetic Programming Theory and Practice*, 4:299–314, 2009.

[7] M.A. Kaboudan. Genetic programming prediction of stock prices. *Computational Economics*, 16:207–236, 2000.

[8] Devayan Mallick, Vincent C. S. Lee, and Yew Soon Ong. An empirical study of genetic programming generated trading rules in computerized stock trading service system. *International Conference on Service Systems and Service Management*, pages 1–6, 2008.

[9] Jeremie Gypteau, Fernando E. B. Otero, and Michael Kampouridis. Generating directional change based trading strategies with genetic programming. *Applications of Evolutionary Computation*, pages 267–278, 2015.

[10] Johan Bollen, Huina Mao, and Xiao-Jun Zeng. Twitter mood predicts the stock market. *Journal of Computational Science*, 2:1–8, 2010.

[11] Nicolas Navet and Shu-Heng Chen. Financial Data Mining with Genetic Programming: A Survey and Look Forward. In *56th Session of the International Statistical Institute (ISI 2007)*, Lisboa, Portugal, July 2007.

[12] Riccardo Poli, William B. Langdon, and Nicholas Freitag McPhee. *A field guide to genetic programming.* Published via `http://lulu.com` and freely available at `http://www.gp-field-guide.org.uk`, 2008.