Decomposition Genetic Programming: An Extensive Evaluation on Rainfall Prediction in the Context of Weather Derivatives

Sam Cramer, Michael Kampouridis, Alex A. Freitas[1]

School of Computing, University of Kent

Abstract

Regression problems provide some of the most challenging research opportunities in the area of machine learning, where the predictions of some target variables are critical to a specific application. Rainfall is a prime example, as it exhibits unique characteristics of high volatility and chaotic patterns that do not exist in other time series data. Moreover, rainfall is essential for applications that surround financial securities, such as rainfall derivatives. This paper extensively evaluates a novel algorithm called Decomposition Genetic Programming (DGP), which is an algorithm that decomposes the problem of rainfall into subproblems. Decomposition allows the GP to focus on each subproblem, before combining back into the full problem. The GP does this by having a separate regression equation for each subproblem, based on the level of rainfall. As we turn our attention to subproblems, this reduces the difficulty when dealing with data sets with high volatility and extreme rainfall values, since these values can be focused on independently. We extensively evaluate our algorithm on 42 cities from Europe and the USA, and compare its performance to the current state-of-the-art (Markov chain extended with rainfall prediction), and six other popular machine learning algorithms (Genetic Programming without decomposition, Support Vector Regression, Radial Basis Neural Networks, M5 Rules, M5 Model trees, and k-Nearest Neighbours). Results show that the DGP is able to consistently and significantly outperform all other algorithms. Lastly, another contribution of this work is to discuss the effect that DGP has had on the coverage of the rainfall predictions and whether it shows robust performance across different climates.

Keywords: Weather derivatives, rainfall prediction, problem decomposition, genetic programming, genetic algorithm

1. Introduction

Regression based problems provide a unique challenge for researchers, where the prediction of outputs have a pivotal outcome in real-life problems. The complexity can be overcome through specific domain knowledge, but often this is not the case. Within complex and chaotic time series data, there is a lack of reoccurring patterns and domain knowledge can be scarce. A type of time series, which remains one of the most difficult and crucial to applications, is rainfall. This time series contains high volatility, little to no seasonality and is highly random. The effects of rainfall can lead to devastation, and unfavourable conditions can impact societies' and ecosystems' ability to survive.

The phenomenon of rainfall has a direct impact on various domains such as water resource planning, agriculture and biological systems. Within finance, predicting the level of rainfall is important for protecting an individual's income from the adverse rainfall effects. Over the years people have sought means of protecting their day-to-day income from unfavourable rainfall, but only until more recently has this been possible. Insurance from rain's adverse effects has existed for many years, but often is of little use unless the impact is of high catastrophe, causing destruction. For instance, a farmer would only be able to receive compensation if s/he could demonstrate destruction of their crop, e.g. because of a severe flood. However, such business can also be affected by unfavourable rainfall, which is not

[1] Corresponding author: Michael Kampouridis, School of Computing, Medway, ME4 4AG, UK. Tel: +44 1634 88 8837. Email: M.Kampouridis@kent.ac.uk

necessarily catastrophic. For example, if a certain year is drier than normal, there might be a significant effect in the crop production. In such cases, rainfall derivatives is a new method for reducing the financial risk posed by adverse or uncertain weather circumstances. A rainfall derivative has the advantage that no proof of damages caused by rain is required to exercise protectionism, only the contract purchased.

Rainfall derivatives are part of the concept of weather derivatives, sharing many of the same aspects of normal financial derivatives (e.g., oil and grain). This derivative is an agreed contract between two or more parties and can be written on the level of rainfall expected over a certain period of time. This contract's value is priced according to the level of rainfall predicted over that period in the future. Therefore, the problem of rainfall derivatives can be broken down into two parts. The first problem is predicting the accumulated rainfall over a specified period and the second problem is having a pricing framework. The latter has its own unique problematic features, as rainfall derivatives constitute an incomplete market[2]. To reduce the problem of mispricing, an algorithm that can predict rainfall accurately is key, before assigning a price. In this paper we focus on this first aspect of predicting the rainfall amount.

As the concept of rainfall derivatives is relatively new, there exists little literature on this subject. Moreover, the difficulty in predicting rainfall has deterred the attention of researchers, unlike other weather derivatives such as temperature[3]. To estimate future levels of rainfall, the Markov-chain extended with rainfall prediction (MCRP) [7] method has been commonly applied in a wide range of the literature, including rainfall derivatives [8, 9, 10, 11]. The general MCRP approach is often referred to as a 'chain-dependent process' [12], which splits the model into capturing first the occurrence pattern, and then predicting the rainfall intensities. The occurrence pattern is produced by a Markov-chain, where state 0 is a dry day and state 1 is a wet day. If a wet day is produced then the rainfall intensity is calculated by generating a random number from a given distribution (typically Gamma or Mixed-Exponential distribution), otherwise a value of 0 is assigned (zero rainfall). We refer the reader to [7] for a complete description of MCRP. Despite being a popular approach, MCRP is very simplistic and does not truly capture the irregularities of rainfall. The final result tends to fluctuate around the observable mean of the training data. Moreover, there exists a large number of rainfall pathways that do not reflect future behaviour.

A way of dealing with the difficulty of predicting rainfall and to overcome some of the difficulties in modelling the time series of rainfall, is through change point models. The idea is based on abrupt changes in the time series, those points are considered a change point, with a new model explaining the time series within each segment [13]. They are frequently employed within econometrics [14] [15], climate [16] and hydrology [17], amongst other problem domains. The concept is similar to a decomposition method proposed in [18], but change point models split the time series into a typically larger number of smaller segments on the time axis. In [18], the time series of rainfall is split on the dependent variable according to whether the next day is expected to observe high, medium or low rainfall. The difference being, only three regression equations explain the whole time series of rainfall, instead of a larger number of regression models based on the abrupt changes in the time series.

Machine learning methods can be seen as an alternative and have become more popular over recent years. Typical applications within machine learning revolve around short term predictions (e.g. rainfall-runoff models up to a few hours [19] or monthly amounts [20] [21]). For daily predictions, [22] used a feed-forward back-propagation neural network for daily rainfall prediction in Sri Lanka, which was inspired by the chain-dependent approach from statistics. The work in [23] also applied GP to daily rainfall data, but the GP performed poorly by itself, although when assisted by wavelets the predictive accuracy improved. In the context of rainfall derivatives a selection of machine learning algorithms was explored in detail in [24], which showed that Radial Basis Function (RBF), Support Vector Regression (SVR) and Genetic Programming (GP) outperformed the commonly applied method of MCRP following a transformation of the data. In addition, [25] presented in detail a tailored GP for the problem of rainfall prediction, and [26] extended the above work by exproring the use of feature extraction. Both works showed promising results, where the GP could outperform MCRP, the current-state-of-the art. Furthermore, [18] extended the above GP works, by proposing a new algorithm called Decomposition GP (DGP). This was a novel hybrid algorithm (comprising of a Genetic Algorithm (GA) part, and a Genetic Programming part) that decomposes the problem of rainfall into subprob-

[2] In incomplete markets, the derivative can not be replicated via cash and the underlying asset; this is because one can not store, hold or trade weather variables.

[3] In fact, temperature weather derivatives have attracted a lot of research, both from the statistical and mathematical community [1, 2], as well as the machine learning community[3, 4, 5, 6].

lems. The motivation for doing this was to allow the GP to focus on each subproblem, before combining back into the full problem. The GP did this by having a separate regression equation for each subproblem, determined based on the level of rainfall; in addition, the GA determined which regression equation should be used (solving a classification problem). As we turn our attention to subproblems, this reduces the difficulty when dealing with data sets with high volatility and extreme rainfall values, since these values can be focused on independently.

The main novelty of our paper is to present an in-depth technical and experimental comparative approach of the DGP algorithm, by building on [18]. This algorithm is an important step for time series that exhibit extreme time series behaviour. It is especially important within rainfall derivatives, where the price of a derivative is determined based on the level of rainfall, a prime example of the types of problems that our algorithm is looking to overcome. More specifically, the current study expands our previous work in the following five ways: (i) we present a more in-depth presentation of the DGP algorithm, (ii) we double the number of cities tested to 42, and we include cities not only from Europe, but also from the USA, (iii) we increase the number of algorithms we use as benchmarks from three (GP without decomposition, MCRP, RBF) to seven, as we now also include results for SVR, the M5 algorithm (both model trees M5R, and rules M5P), and k-Nearest Neighbour (KNN), (iv) we provide an extensive analysis on the results in terms of the GA component, which handles a classification task, as we compare it to other well-known classification techniques, such as RBF, SVM, RIPPER, Discriminant Analysis (DA), and Naive Bayes (NB), and (v) we provide an extensive discussion on the effectiveness of the DGP algorithm, by investigating how well its predictions cover the range of all rainfall data, and also by looking into how robustly it performs across different climates.

The remainder of this paper is organised as follows. In Section 2, we outline the data used. In Section 3, we present in detail the decomposition algorithm and its components. In Section 4, we outline the experimental setup for the DGP algorithm, and in Section 5, we discuss the results. In Section 6, we evaluate the effectiveness of DGP and also analyse the algorithm's performance on different climates. Finally in Section 7, we conclude and present future work.

2. The Data Used in the Experiments

The daily rainfall data used is summarised in Table 1, which includes a total of 20 cities from around Europe and 22 from around the United States of America (USA). The data was retrieved from NOAA NCDC[4].

The use of machine learning methods effectively requires a modification to the data to align it with the problem domain of rainfall derivatives. Following [24] we use a sliding window accumulation method, given by:

$$r_{t_s} = \sum_{t=t_s}^{t_e} r_t, \tag{1}$$

where r_t is the accumulated amount of rainfall over a number of days, with the day varying over a contract period from t_s till t_e.

This is consistent with pricing a contract, whereby the price of a contract is the total amount of rainfall within a specified period of time, otherwise known as the contract period. The most common contract traded is monthly and contracts are only available for the months of March through October. Given we are interested in pricing monthly contracts, we use a sliding window length that covers the modal length of contracts, which is 31 days. We do not look for an optimum period to accumulate to help with prediction, because our problem domain is set out as the accumulated rainfall amounts over the contracts that are currently traded — that is, the contract period is chosen by the user, not by the algorithm.

3. Decomposed Genetic Programming

3.1. Overview

Within this section we outline how we achieve the decomposition and how we break the problem down into smaller subproblems.

[4] https://www.ncdc.noaa.gov/

Table 1: The list of all cities whose daily rainfall amounts will be used for experiments.

City	State	City	Country
Akron	Ohio	Amsterdam	Netherlands
Atlanta	Georgia	Arkona	Germany
Boston	Massachusetts	Basel	Switzerland
Cape Hatteras	North Carolina	Bilbao	Spain
Cheyenne	Wyoming	Bourges	Germany
Chicago	Illinois	Caceres	Spain
Cleveland	Ohio	Delft	Netherlands
Dallas	Texas	Gorlitz	Germany
Des Moines	Iowa	Hamburg	Germany
Detroit	Michigan	Ljubljana	Slovenia
Jacksonville	Florida	Luxembourg	Luxembourg
Kansas City	Kansas	Marseille	France
Las Vegas	Nevada	Oberstdorf	Germany
Los Angeles	California	Paris	France
Louisville	Kentucky	Perpignan	France
Nashville	Tennessee	Potsdam	Germany
New York City	New York	Regensburg	Germany
Phoenix	Arizona	Santiago	Portugal
Portland	Oregon	Strijen	Netherlands
Raleigh	North Carolina	Texel	Netherlands
St Louis	Missouri		
Tampa	Florida		

Our DGP consists of a number of individuals split into two separate populations, a GP part and a GA part. The GP part consists of b expression trees, where nodes represent functions or terminals as usual in GP [27]. For our implementation we define b to equal 3, such that we have 3 GP equations to predict low, medium and high rainfall amounts. The GA part consists of a linear chromosome with a string of n rules, each with g genes.

3.1.1. Decomposing Rainfall Amounts

In order to decompose rainfall, we partition the data into three different partitions (low, medium and high rainfall amounts), thus simplifying the prediction process. Partitions are done for each data set separately, thus different data sets may not have the same criterion used for splitting the data. More partitions could be considered, but we anticipate that three partitions is sufficient by analysing previous experimentation, where the low and high levels of rainfall received little coverage by a single regression equation. We discuss the process of splitting the data in Section 3.1.2. Then, in Section 3.1.3, we will discuss how GP was adapted to create multiple regression equations, one for each partition.

3.1.2. Splitting the Data

As we are creating a separate equation for low, medium and high levels of rainfall, we require two constants to split the data into three partitions. We refer to these two constants as a lower criterion LC and upper criterion UC, as shown by Figure 1. Thus, anything below LC is considered low rainfall, anything between LC and UC is considered medium rainfall and above UC is considered high rainfall. We allow for each individual of DGP to have its own LC and UC, instead of having two fixed constants applied to all individuals within the population. By assuming two fixed constants, we would not be able to determine whether the values of LC and UC are optimal and would need a way of estimating them prior to running our DGP. Therefore, we allow the LC and UC to evolve along with the GP and GA part of DGP, by encoding the LC and UC values within the linear representation of a GA individual. The values of LC and UC are considered based on the training data of each individual city. One aspect that is open to future research is considering a dynamically changing LC and UC, taking into account the uncertainty around certain periods of time.

3.1.3. Genetic Programming Trees

Using the information from a given LC and UC the rainfall time series can be split into three partitions. As shown by Figure 1 we require an equation to predict within the boundaries specified, thus we map each partition to a

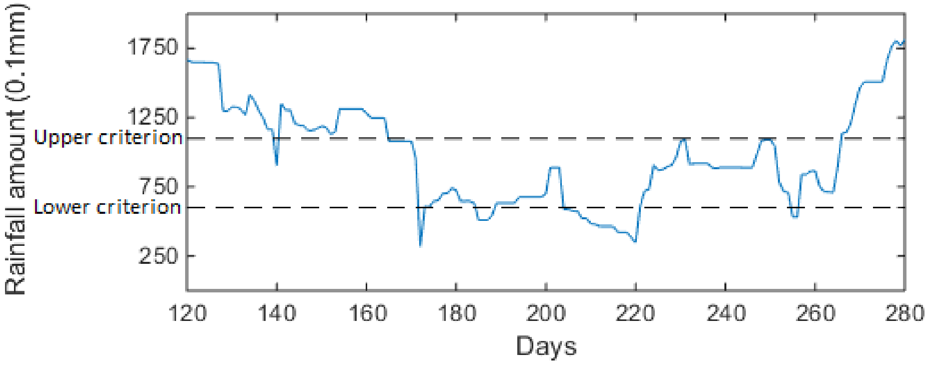


Figure 1: Rainfall data split into three partitions according to a lower criterion and upper criterion.

particular GP branch (b_n), shown by Figure 2. The concept is that a rainfall equation should be capable of predicting all points within its specified range and is evolved based on its ability to do so, whilst restricting behaviour outside of this range. Thus, having independent equations allows the GP to evolve each branch to maximise the predictive performance within each partition. Keeping the branches independent is required given that the patterns of rainfall and available data will differ across partitions. To ensure that b_1 does not consider information from b_2 or b_3, we keep each branch independent and separate throughout the evolutionary process. To achieve this behaviour we create a crossover and a mutation operator that can only act on the same branch amongst individuals. The procedure is similar to the standard genetic operators, but is performed branch-wise, once per branch, rather than once per individual. Using tournament selection to randomly select two parents based on their performance to solve the complete problem, DGP chooses a random node/leaf from one branch and combine it with the same branch from the other parent. This process is repeated for all branches. We choose to keep the same parents for the three crossovers associated with the three branches, rather than select a new parent for each branch, to avoid too much disruption and randomness during the evolutionary process. Mutation follows the same procedure, a parent is chosen and one node/leaf on each branch undergoes single-node mutation.

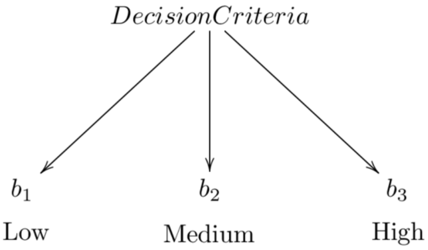


Figure 2: The representation of the decision criteria and the three branches for regression. Upon evaluation of the decision criteria, this leads to one of the three branches; each branch is a different GP tree, representing a different rainfall prediction equation.

Elitism places into the next generation a new individual formed by a combination of branches b_1, b_2 and b_3 based on the predictive performance of each branch. In order to create the elite individual, we merge the best from b_1, b_2 and b_3 across the entire population, creating a new individual consisting of the three best branches from the previous generation. Within this framework we use b_1 to represent low rainfall, b_2 to represent medium rainfall and b_3 to represent high rainfall, as shown by Equation 2.

$$\text{GP individual} \begin{cases} b_1 & \text{if } r_t \leq LC \\ b_3 & \text{if } r_t \geq UC \\ b_2 & \text{otherwise.} \end{cases} \tag{2}$$

The general algorithm of DGP can be found in Algorithm 1. The inputs for the algorithm are the parameters controlling the decomposition of the time series, also the rainfall data, and the final output is the rainfall predictions. One variable that is unknown from Algorithm 1 and Equation 2 is r_t, which is the actual level of rainfall. Within our framework of DGP, this is the crucial variable to compare against LC and UC. To do so we use a classification technique to determine the branch to evaluate, discussed in the next section.

Algorithm 1 Decomposing rainfall amounts

```
$P \leftarrow$ Number of individuals in population
$B \leftarrow$ Number of partitions
$t \leftarrow$ Time period
for Individual $i = 1, \ldots, P$ do
    for Branch $b = 1, \ldots, B$ do
        initialise(branch$^i_b$)
    end for
    Set $LC_i$
    Set $UC_i$
end for
for Generation $g = 1, \ldots, G$ do
    for Individual $i = 1, \ldots, P \; \forall \; t$ do
        if $r_t \leq LC_i$ then
            Evaluate $b_1$
        else if $r_t \geq UC_i$ then
            Evaluate $b_3$
        else
            Evaluate $b_2$
        end if
        Calculate fitness
    end for
    Breed
end for
```

3.2. The GA Component of the DGP

In this section we outline the GA to classify each data point into the correct partition of rainfall amount. First, we introduce the representation of our GA in Section 3.2.1. Then, we discuss the fitness criteria to be used in Section 3.2.2. Finally, the breeding of our GA is described in Section 3.2.3.

3.2.1. Decomposing the Problem with the GA Component

Predicting levels of rainfall requires rebuilding the decomposition back into the original problem. Within our framework, DGP needs to choose which branch to evaluate on a given day. In order to do so, we use a GA with a linear representation, as part of a hybrid DGP individual, to classify. Figure 1 shows the importance of classifying correctly, especially when considering the impact of misclassifying by more than one class. For example, if the actual rainfall amount is within the high rainfall partition (amounts $>$ 110mm) and a classifier predicts low rainfall, then this will point to the wrong branch (tree) in the GP-part representation of the DGP individual, leading to an equation predicting much lower rainfall amounts, possibly in the range of less than 50mm, thus causing an error of at least 50%.

The GA-part of the DGP individual representation consists of 5 genes; predictor, period, lower criterion, upper criterion and order. Our GA linear representation is essentially a rule list for a given period of time within a year. Each rule has the same number of outcomes as the number of specified partitions. Keeping the rules consistent will keep the understanding of the rules very intuitive and comprehensive. The rules will consist of making decisions based on the same attributes used within the GP's terminal set, presented in Section 3.3. The rules will be kept very simple and will be based on a single attribute along with a $>$ or $<$ operator and a constant. For each period of time only one rule will be present with three outcomes. We do not consider chaining rules involving logical operators such as AND, OR and NOT. Based on the outcome of the rule, the GA will decide the respective branch to evaluate.

Table 2: All the possible values for each gene, except for order. As we have a rule for each month, only the total number of days per month is given.

Genes of the GA-part of an individual	
Predictor	$\{r_{t-1}, r_{t-2} \ldots r_{t-11}\}$, $\{r_{y-1}, r_{y-2} \ldots r_{y-10}\}$
Period	31, 30 and 28
Lower Criterion (*predLC*)	0.05 - 0.65
Upper Criterion (*predUC*)	0.35 - 0.95

The predictor refers to one of the attributes used within the GP's terminal set, e.g. r_{t-1}, r_{t-2} and so on. Period refers to the number of days covered by a rule — e.g., a value of 31 would cover the next 31 days. Within our methodology we keep the period consistent and apply a rule for each month of the year, however, variable period lengths could also be considered. The lower and upper criteria are the decision thresholds for choosing which class to predict, *predLC* and *predUC* respectively, based on the predictor's value. For our experimentation we define the *predLC* and *predUC* in terms of percentiles of the training set, but this could be modified accordingly to any real number or function. The complete list (excluding order) of values of the genes in the GA is specified in Table 2. The order is one of the unique permutations of the three branches, given below:

$$\begin{array}{cccccc} \multicolumn{6}{c}{\text{Order reference}} \\ [1] & [2] & [3] & [4] & [5] & [6] \\ \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} & \begin{bmatrix} b_1 \\ b_3 \\ b_2 \end{bmatrix} & \begin{bmatrix} b_2 \\ b_1 \\ b_3 \end{bmatrix} & \begin{bmatrix} b_2 \\ b_3 \\ b_1 \end{bmatrix} & \begin{bmatrix} b_3 \\ b_1 \\ b_2 \end{bmatrix} & \begin{bmatrix} b_3 \\ b_2 \\ b_1 \end{bmatrix} \end{array} \tag{3}$$

where each permutation corresponds to the following criteria:

$$\begin{bmatrix} predictor < predLC \\ predLC < predictor < predUC \\ predictor > predUC \end{bmatrix}.$$

For example order 3, whenever the predictor is less than *predLC* we classify medium rainfall (b_2). If greater than *predUC* we classify high rainfall (b_3), otherwise low rainfall (b_1).

Due to rainfall features exhibiting very complex and chaotic processes, it is highly unlikely that a single predictor can classify accurately. Such low probability in classification motivates us to allow a larger number of rules to be created throughout the year, which is able to reduce complexity in rainfall prediction, hence the period criteria. To best describe the characteristics of each month throughout each year, we set 12 rules, one for each corresponding month. However, the number of rules can be adjusted according to the user's or model's preferences. Furthermore, the order of the three branches is an important aspect within the classification process, because the same predictor could be used in a different month under different criteria. Figure 3, shows a sample representation of the above description, where we demonstrate the rules for January, February and December.

$$\Big[\overbrace{\mathrm{r}_{t-1}, 31, 37, 91, 2,}^{\text{January}} \overbrace{\mathrm{r}_{y-3}, 28, 22, 77, 2}^{\text{February}} \ldots \overbrace{\mathrm{r}_{t-1}, 31, 11, 64, 6}^{\text{December}}\Big]$$

Figure 3: An example of a GA for 3 out of 12 months

For the example in Figure 3, the classification rules for January, February and December are shown in Equation 4, Equation 5 and Equation 6 respectively, showing the impact of a different order (by cross-referencing Equation 3 with Figure 3) and the different criteria to split the predictor. The period refers to the number of days the rules cover and is expressed in each equation as the days covered during a year. Therefore, the rules shown below are the same for every day in the respective months.

$$\text{January (Days 1-31)} \begin{cases} b_1 & \text{if } r_{t-1} \leq 37^{th}\text{percentile} \\ b_2 & \text{if } r_{t-1} \geq 91^{st}\text{percentile} \\ b_3 & \text{otherwise,} \end{cases} \tag{4}$$

$$\text{February (Days 32-60)} \begin{cases} b_1 & \text{if } r_{y-3} \leq 22^{nd}\text{percentile} \\ b_2 & \text{if } r_{y-3} \geq 77^{st}\text{percentile} \\ b_3 & \text{otherwise,} \end{cases} \tag{5}$$

$$\text{December (Days 335-365)} \begin{cases} b_3 & \text{if } r_{t-1} \leq 11^{th}\text{percentile} \\ b_1 & \text{if } r_{t-1} \geq 64^{th}\text{percentile} \\ b_2 & \text{otherwise,} \end{cases} \tag{6}$$

After the inclusion of our GA component into our DGP, we modify our general DGP algorithm as shown in Algorithm 2. The inputs for the algorithm are the parameters controlling the decomposition of the time series and for the GA, also the rainfall data. The output is the rainfall predictions after decomposing the rainfall time series.

Algorithm 2 Adding our decision criteria into DGP

```
$P \leftarrow$ Number of individuals in population
$B \leftarrow$ Number of partitions
$t \leftarrow$ Time period
for Individual $i = 1, \ldots, P$ do
    for Branch $b = 1, \ldots, B$ do
        initialise(branch$^i_b$)
    end for
    Set $LC_i$
    Set $UC_i$
    Initialise GA
end for
for Generation $g = 1, ..., G$ do
    for Individual $i = 1, \ldots, P \; \forall \; t$ do
        Evaluate individual $i$ of the GA
        Choose branch
        Evaluate branch
        Calculate fitness
    end for
    Breed
end for
```

3.2.2. Fitness Criteria

Each individual of the hybrid DGP will have the output of its GP component (which is partly determined by the values of the GA-component genes) evaluated using RMSE (Root Mean Square Error). However, we also need to compute the fitness of the GA-part of an individual separately. To compute the GA-part's fitness we use Kendall's tau (τ) correlation coefficient, which is used to measure the rank correlation between two variables taking into account the natural ordering of our nominal classes (low, medium, high rainfall). This measure will help deter from misclassifying by more than one class. Kendall's tau is given by:

$$\tau_B = \frac{n_c - n_d}{\sqrt{(n_0 - n_1)(n_0 - n_2)}}, \text{where}$$

$$n_0 = \frac{n(n-1)}{2}, n_1 = \sum_i \frac{t_i(t_i-1)}{2}, n_2 = \sum_j \frac{u_j(u_j-1)}{2},$$

where n_c = Number of concordant pairs, n_d = Number of discordant pairs. t_i = Number of tied values in the i^{th} group of ties for the predicted values p and u_j = Number of tied values in the j^{th} group of ties for the actual values a. Let $(p_1, a_1), (p_2, a_2), \ldots, (p_n, a_n)$ be a set of observations, in our case the predicted class and the actual class, where n refers to the number of training instances. A pair of observations is concordant if the ranks for (p_i, a_i) and (p_j, a_j) both agree, such that ($p_i > p_j$ and $a_i > a_j$) or ($p_i < p_j$ and $a_i < a_j$) and vice versa if discordant.

3.2.3. Individual Evaluation and Breeding of the Genetic Algorithm

Each individual of the GA will be evaluated based on the Kendall's correlation mentioned above, which will return a value in the range of $[-1, 1]$. A value of 1 represents a perfect agreement between rankings of predicted and actual classes. Once the population has been evaluated, selected individuals undergo genetic operations. The GA-part of the individuals can undergo point mutation and a variety of crossover techniques. The mutation procedure will choose a random point within the individual and replace it with a random variable or value that is of the same type. Therefore, one can not replace a predictor (e.g. r_{t-4}) with *predLC*, only with another predictor (e.g. r_{y-5}). We will cover the process of elitism in Section 3.4, because it requires the interaction between the GP and GA components of the hybrid DGP. We opt for tournament selection to select the parents for breeding and discuss the variety of crossover methods below. All these methods will be used in our DGP and will be chosen at random to promote a good diverse balance of individuals.

Multiple Split Points. We apply the multiple split point method, similar to the one-point crossover, where we choose a random point and take one section from the first parent and the other section from the second. However, given our chromosome is 60 genes (12 sets of 5 genes) in length and to increase the mixing of individuals, we choose a random number s in the range [1,12] and create s splits in random locations in our chromosome. Therefore, creating individuals with a mix from two parents through random split points.

Multiple Rule Split. We use a crossover technique that swaps entire rules (without breaking a rule) among parents, i.e. choosing a crossover point located at the boundary between two adjacent rules, rather than arbitrary split points (which could be inside rules). One possible advantage is that we keep the rules intact and do not cause too much destruction of each GA individual. Therefore, we consider crossover on our 12 rules. We choose which rules to crossover by assigning a probability to the crossover process. The first step is to choose the number of rules s randomly in the range [1,11] to select from each parent and from that we assign the probability. For example, if s is 6, then the probability is 50% of selecting a rule from either parent and if s is 3 then the probability is 25% of choosing a rule from the first parent. We then sequentially move along each rule and sample a value from the uniform distsribution to decide which parent to choose from, based on the probability identified.

Single Split Within Rule. An alternative is to mix the two crossover methods above. Sequentially moving along each rule, we choose at random a gene in the range [0,5]. A value of 0 means that no split is required and use all of the material from the first parent. A value of 1 would mean that the first 4 genes are from the first parent and the 5th gene would be from the second parent. We repeat this process for all rules.

Uniform Crossover. The final alternative for crossover is adapting a uniform crossover procedure, where we use a probability (0.5) for each gene within each rule. Then, for each gene, we choose at random whether to pick from the first or second parent for the new offspring, when creating each child.

3.3. The GP component of the DGP

In this section we describe the GP-like part of the individual representation, which is based on a Strongly-Typed GP (STGP) [28] with modifications used in [25] for the problem of rainfall prediction. Hereafter we use the terms GP and GA, for short, to refer to the GP and GA components of the hybrid DGP.

3.3.1. Terminals

The variables are defined by the r_t's and r_y's calculated based on the data from Section 2, where r_t is the accumulated rainfall amount in the last known non overlapping sliding window t periods ago. Similarly, r_y is the accumulated rainfall amount in the current sliding window y years ago.

The second element is an ephemeral random constant (ERC), which will pick a uniformly distributed random number. The third element is a set of constants from -4 to 4, at 0.25 intervals, which will take a separate type from the terminals already discussed. These are constants that are specific to the power function. Due to using STGP, we can ensure that the second argument of the power function is always one of these constants and does not create an illegal tree.

3.3.2. Function set

The function set includes: Add (ADD), Subtract (SUB), Multiply (MUL), Divide (DIV), Power (POW), Square root (SQRT), and Log (LOG). The functions LOG, SQRT and DIV are protected. Additionally, the second argument for POW will be a constant in a specified range as mentioned in 3.3.1. Since we allow for fractional powers, we force a whole number for the second argument, if the first argument is negative. The function and terminal sets are summarised in Table 3.

Table 3: GP function an terminal sets

Set	Value
Functions	ADD, SUB, MUL, DIV, POW, SQRT, LOG
Terminals	11 r_t periods $\{r_{t-1}, r_{t-2} \dots r_{t-11}\}$, 10 r_y periods $\{r_{y-1}, r_{y-2} \dots r_{y-10}\}$, ERC, Constants in the range [-4,4]

3.3.3. Management of Trees

Due to rainfall being a strictly non-negative variable, a wrapper around each individual is included to modify the prediction to zero if the tree evaluates to a negative amount. The final adjustment is to ensure a balance between variables and random numbers in an individual. Thus, the first child of each node is either a function or a variable. Whereas, the second child of each node can be a variable, ERC or a function. We initialise the population using the ramped-half-and-half method.

3.3.4. Fitness Function

The fitness function used for evaluation will be the root mean squared error (RMSE), given by:

$$RMSE = \sqrt{\frac{1}{N}\sum_{t=1}^{N}(r_t - \bar{r}_t)^2} \tag{7}$$

where N is the length of the training set, r_t represents the predicted rainfall amount and $\bar{r}_t$ represents the actual rainfall amount for the t^{th} data point (time index).

3.4. Integrating the GP and GA Components

In this section we outline three aspects of the integration of the GP-part and GA-part of the individual representation of the hybrid DGP, namely: penalising the regression trees, elitism, and the evolution of the LC and UC criteria to partition the data for classification.

3.4.1. Penalising GP Regression Trees

Following the decomposition approach, it is key that each regression equation (a GP tree) predicts values within its respective partition. For example, it makes little sense for an equation responsible for the low rainfall class, predicting values in medium and high rainfall class. Therefore, we implement a penalty function based on the distance away from the correct partition, as shown in Figure 4. To integrate the GP and GA components and maximise the usefulness of this idea, we implement a simple check before choosing whether to penalise or not. The GP-related penalty will only apply to situations where the GA has correctly classified. Therefore, we are not penalising GP for making a wrong prediction given that the GA was at fault. This modification should influence GP to predict within a range similar to that of the specified partition. From Figure 4 any deviation denoted by the dashed vertical lines is penalised by Equations 8.

$$
\begin{aligned}
&\text{Actual class is low}\\
&p^{new} = \begin{cases} p^{old} + m(p^{old} - LC) & \text{if } \; c_p = c_a \text{ AND } p^{old} > LC \\ 0 & \text{otherwise.} \end{cases}\\
&\text{Actual class is medium}\\
&p^{new} = \begin{cases} p^{old} - m(UC - p^{old}) & \text{if } \; c_p = c_a \text{ AND } p^{old} < UC \\ p^{old} + m(p^{old} - LC) & \text{if } \; c_p = c_a \text{ AND } p^{old} > LC \\ 0 & \text{otherwise.} \end{cases}\\
&\text{Actual class is high}\\
&p^{new} = \begin{cases} p^{old} - m(UC - p^{old}) & \text{if } \; c_p = c_a \text{ AND } p^{old} < UC \\ 0 & \text{otherwise.} \end{cases}
\end{aligned}
\tag{8}
$$

where p^{new} represents the predicted rainfall amount by GP after penalising and p^{old} represents the rainfall amount originally predicted by GP. m represents a scaling function on the penalty, c_p is the predicted class and c_a is the actual class (i.e. the observed rainfall amount). UC and LC are the upper and lower criteria for splitting the data into its respective classes. For example, let us assume that $c_p = c_a$, if GP predicted 1000 tenths of mm (p^{old}), where the UC is 1100 and m was 2, but the true class is high rainfall. We would then update p^{new} by $1000 - 2 \times (1100 - 1000)$, hence p^{new} is penalised to 800. The idea is for GP to deter from assigning a good fitness to this individual, given the large penalty effect.

An alternative method for handling the case of predicting in the wrong partition is to have a wrapper to round the equation up or down to the nearest partition. However, compared to the idea of penalising, this may encourage poor performers to get selected for future generations by forcefully rounding poor performers. An example is an equation for partition medium, predicting values excessively large or low. By penalising the DGP individual, they are further deterred, but through rounding they are comparable to equations predicting within the same range. Within Algorithm 2, this step would be inserted before calculating the predictive accuracy.

3.4.2. Elitism Merging Different Individuals

The use of elitism in our evolutionary process relies on exchanging information to create the best individual to put into the next generation. Typically, elitism would take the best GP trees and GA genes separately and put them into the next generation. However, due to the close integration between the GP and GA components of an individual, we create our own elitism strategy.

The first consideration was mentioned in Section 3.1.3, where we merge the best performing branches b^{rank}_{number} together in ranking order. The elitism strategy perceives the DGP as combination of three separate populations of individuals and the GA-part as a separate population as well. Each individual in the population of branches gets its fitness evaluated based on how it was able to solve its respective subproblem in terms of RMSE. Additionally, each GA individual has its fitness evaluated based on the Kendall's tau correlation rank. Through this procedure we aim to promote the best branches to create an elite individual. Thus, the best branch b^1_1 will merge with b^1_2 and b^1_3. Note that the GA component and the GP branches are jointly responsible for achieving a better RMSE. For instance, b^1_1, b^1_2 and b^1_3 may not come from the same parent using the same GA-based partition rules. Potentially, we may have 3 different GA-based rule lists influencing the performance. Thus, we need an intermediate step to decide which of

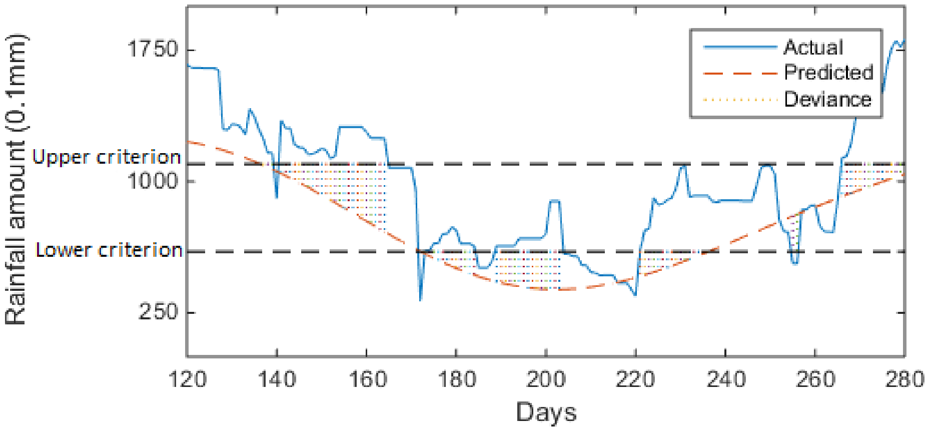


Figure 4: The distance from the predicted amount to either the lower bound or upper bound when GP predicts a rainfall amount in the wrong partition. The deviance is then used to calculate a penalty.

the GA-based rule lists is responsible for the best overall individual using all 3 branches. Therefore, we evaluate in turn each GA-based rule list (i.e., each GA individual) associated with the best branches merged together. We also evaluate the best GA individual overall based on its Kendall's tau correlation rank, which may not be attached to any branch. After re-evaluating the newly merged offspring, the partition rule list that was responsible for returning the best fitness in terms of RMSE is moved into the next generation as part of the offspring.

This helps evolve the partition rules that can perform the best classification across the training period, helping the GP to solve the regression problem.

3.4.3. Evolution of LC and UC

The last aspect of the hybrid DGP is the process of evolving *LC* and *UC* (our decomposition approach). These criteria are required for the GP component to construct regression equations (trees) to predict within each data partition and for the GA-based rule lists to classify into the relevant classes. Recall that each individual consists of three GP regression trees and a GA-based rule list.

The use of *LC* and *UC* is to split the initial data into the three partitions, such that GP creates an equation to predict within each partition and the GA assists by selecting the corresponding branch to evaluate on each day. By evolving the criteria that bind the two hybrid parts together, we hope to find an optimal point where both the GP and GA part can minimise the RMSE on the whole problem. We do not directly influence the behaviour of the *LC* and *UC* and leave it up to the GA through the evolutionary process to modify them as necessary. To ensure the split points for decomposition are evolved, during crossover the two parents' *LC* and *UC* values undergo uniform crossover to create the future offspring. With uniform crossover on two points there is a $\frac{1}{2}$ chance of both *LC* and *UC* coming from the same parent and $\frac{1}{2}$ chance of a mixture, as shown in Figure 5. Moreover, we do allow these points to be mutable as well, but instead of mutating using a uniform selection of values, we opt for the number to be normally distributed around the old value with a variance of 0.1. The motivation is that we want to modify the split point by a small amount, otherwise mutation can be too disruptive by changing a *LC* value from, say, 0.02 to 0.53, which would have a massive effect on our performance. Unlike the previous two aspects, this aspect is more subtle and directly affects the performance of both GP and GA, and helps guide the evolutionary process of both in turn.

3.4.4. Alternative Classification Techniques

An extension to test the effectiveness of the combination of GA and GP is to consider the use of other classification techniques to act as the decision criteria. The GA part is modified to replace the rule list with a different classification method. Therefore, our GA is simplified by containing an *LC* and *UC* and a classification method to perform the selection for which branch to evaluate for our DGP. We use the following classification techniques: Support

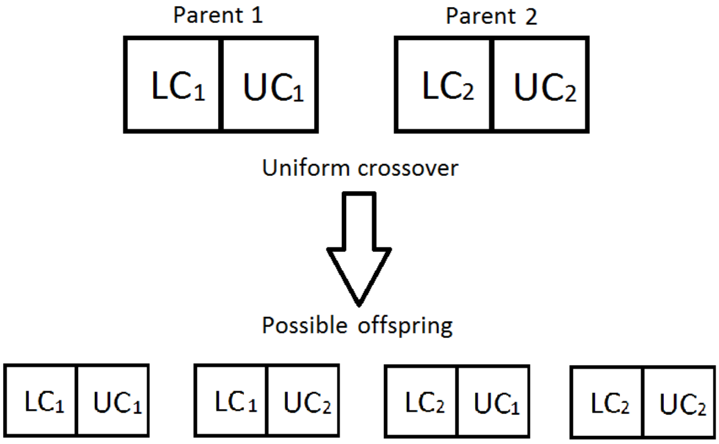


Figure 5: An example showing the breeding of the *LC* and *UC* from two parents using uniform crossover

Vector Machines (SVM), Radial Basis Function (RBF), Repeated Incremental Pruning to Produce Error Reduction (RIPPER), Discriminant Analysis (DA) and Naive Bayes (NB).

We use SVR and RBF as two powerful blackbox techniques that are well regarded for complex applications. Additionally, we use two versions of M5 to have a comparison against a decision tree algorithm (M5P) and decision rules (M5R). This will provide a suitable comparison to GP, as methods that are capable of producing whitebox interpretable models. Finally, we use KNN as a clustering technique for the regression problem.

3.5. Algorithmic Complexity

The computational complexity of DGP on top of a GP is dependent on five main elements: the size of a GA individual n, the length of the training data m, the population size p, the number of generations g and the elitism rate e. As the length of the GA and the population is kept constant throughout the evolution, the best case is equal to the worst case. The complexity can be broken down into the following parts:

(i) Population initialisation
Initialising the population for all decision criteria requires building GA individuals of size n. Thus, a single GA individual has complexity of $O(n)$. This process then needs to be repeated p times, which is the population size. Therefore, the population initialisation complexity is $O(np)$.

(ii) Fitness calculation
First, each GP individual is labelled the partition of rainfall it belongs into, in order to calculate the fitness of the decision criteria. The complexity for a single GP individual is $O(p)$. Given that the fitness calculation has to go through each point of the training data m, the combined complexity of this step is $O(pm)$. In addition, calculating Kendall'ms tau correlation has a complexity of $O(m \log m)$, and occurs for each individual in the population, i.e. $O(pm \log m)$. Thus, the total complexity of calculating the fitness for the decision criteria is $O(pm + pm \log m)$.

(iii) Operators application
Mutation has a complexity of $O(p)$, based on changing a random position for each individual in the population. Crossover has a complexity of $O(np)$, as the whole decision criteria must be visited for each individual. Additionally, the complexity remains the same for all variations specified in Section 3.2.3. Finally for elitism, there is an initial sorting overhead of $O(p \log p)$ to select the best e individuals for elitism at each generation. Furthermore, four evaluations of GP are required with the elitism strategy outlined in Section 3.4.2.[5] As the regression error is calculated for each point in the training data, the complexity is $O(4em)$. Hence, the overall complexity for the operators is $O(p + np + p \log p + 4em)$.

[5] As a reminder, the elitism strategy merges the best three branches, where each branch may have a different decision criterion. Therefore, there are three evaluations of each branch's decision criteria. Moreover, we try the best overall decision criteria, which may be different to the decision criteria corresponding to the best three branches.

The process for steps (ii) and (iii) occurs for a total number of generations g, as per Algorithm 2. As a result, the overall complexity of the DGP algorithm is the combination of the previous three steps, i.e population initialisation, fitness and operators. It is equal to: $O(np + g(pm(1 + \log m) + p + np + p \log p + 4em))$, which can be simplified to: $O(np + g(p(m \log m + n + \log p) + 4em))$.[6]

4. Experimental Setup

The main goal of our experimentation is to establish whether the use of DGP is better than using a standard GP and other well known machine learning methods. As mentioned in the Introduction, producing more accurate rainfall predictions should lead to more accurate pricing.

We have identified three key aspects to investigate for DGP. The first is the performance against the financial state-of-the-art MCRP, as well as several popular machine learning algorithms. The second is the performance of the different classification techniques and the GA, based on how accurately they are able to classify into one of the three classes of rainfall. The third is how each classification algorithm helps the overall problem of rainfall prediction.

4.1. Benchmarks

In order to test the predictive performance, we compare the DGP algorithm against six well known machine learning methods that are capable of performing regression, namely: Radial Basis Function (RBF), Support Vector Regression (SVR), M5 rules, M5 model trees and k-nearest neighbour. Also included is the most commonly used method in rainfall derivatives, Markov chain extended with rainfall prediction (MCRP) and a GP without problem decomposition, which has already been applied to the problem of rainfall derivatives [25].

To determine the classification accuracy we use the following classification techniques to compare against the GA: Support Vector Machines (SVM), Radial Basis Function (RBF), Repeated Incremental Pruning to Produce Error Reduction (RIPPER), Discriminant Analysis (DA) and Naive Bayes (NB). SVM and RBF have been chosen as state-of-the-art classification techniques. Moreover, we use both of these algorithms for regression and have been shown to cope well with the problem landscape. RIPPER is chosen since it generates pruned comprehensive rules, and can thus act as a good comparison against our GA. DA is chosen as it offers different perspectives based on the statistical distribution of our input variables, for this paper we use quadratic DA. Finally, we use NB as a probabilistic approach to the classification problem.

4.2. Parameter Tuning

The general procedure for GP and GA parameters is outlined as follows. Firstly, 10 cities that are not used to evaluate the predictive performance of the methods are used only for the tuning procedure, with 65 years worth of data required for each city. We use the same 10 cities listed in [24] for consistency. Next is to break the data sets into 20 years with 5 years overlap between each one, with the final year being the validation set used for determining the optimal parameter set. The 20 years are then used to construct the data into a training set of 10 years, with the final year being the validation check. 20 years is required, because we allow DGP to observe rainfall values 10 years ago and the final year is always the validation set to preserve the temporal nature of the data.

Using a parameter tuning tool called iRace [29], we iteratively consider all tuning data sets, automatically testing many different parameter setups. Across its many iterations, iRace will resample algorithm configurations that performed well by eliminating poorer configurations via the Friedman test of significance. We need to specify three inputs for iRace, the data sets to calibrate on, the total running budget (number of program calls) and a parameter list. When iRace finishes its execution, the output is the best possible parameter setups, based on all tuning data sets. The optimal set of parameters along with the parameter list specified within iRace for DGP can be found in Table 4.

As we have mentioned, we are also using different classification techniques to act as our decision criteria. We use the same process as tuning DGP for all classification algorithms individually. The only difference is that the classification accuracy is used to determine the best configuration, instead of the combined accuracy using DGP. Table 5 shows the optimal configurations found by iRace for SVM, RBF and RIPPER. NB and DA are not included, as no tuning was required.

[6] Constants have been removed from the simplified equation, as they are irrelevant to big-O notation.

Table 4: The optimal configuration of DGP found by iRace. Parameters with a * are used by both the GP-part and GA-part of DGP. The parameter list passed to iRace is located in the third row. The brackets indicate a range in the given type, either a whole number or a decimal at 2 d.p.

GP Parameters	DGP	iRace parameter setup
Max depth of tree	8	(4, 10)
Population size	1000*	(200, 1000)
Crossover	99%*	(0.40, 1.00)
Mutation	30%*	(0.00, 1.00)
Primitive	32%	(0.30, 0.95)
Terminal/Node bias	64%	(0.30, 0.95)
Elitism	3%*	(0.00, 0.20)
Number of generations	70*	(30, 100)
ERC negative low	-288.42	(-500, 0)
ERC negative high	-224.31	(-500, 0)
ERC positive low	210.43	(0, 500)
ERC positive high	432.23	(0, 500)

Table 5: Optimal parameters using iRace for the three benchmark classification algorithms: SVM, RBF and RIPPER

SVM		RBF		RIPPER	
SVM Type	C-SVC	Minimum SD	28.3	Folds	4
Cost	0.85	Clusters	2	Weight	7.01
Gamma	0.34	Ridge	0.541	Optimisations	3
Kernel Type	RBF			Prune tree	False

4.3. Training/Testing set up

DGP will have its predictive error compared on all 42 different data sets across the USA and Europe against the performance from all methods. DGP will be trained on 10 years of data and be tested on one year of data based on the optimal parameter set found by iRace. The training set is from 01/01/2005 to 31/12/2014 and testing will be compared on the rainfall values from 01/01/2015 to 31/12/2015.

We will then consider the impact of changing the underlying classification technique from GA to one of the techniques given in Section 3.4.4. We will first consider the classification performance and then observe how the DGP performs when the decision process is controlled via a different algorithm. The classification accuracy of our GA and benchmarks will be based on a predefined set of upper and lower criteria. To avoid bias and to have a fair comparison we will use the same set for all classification techniques. Our results will be based on randomly selecting 100 upper and lower criteria to partition our data and we will report the average results. If the algorithm is non-deterministic (which is the case for GA and RBF) then we will run the technique 50 times on the same split points. Following this we will show the performance of DGP with the new decision techniques and compare against DGP with the GA as the decision criteria.

5. Results

Within this section we outline the results for how DGP performs against the benchmarks highlighted earlier. Moreover, we test the classification ability of the original GA against other well known techniques and how this impacts our DGP's predictive accuracy. To compare accuracy we use the Root Mean Squared Error (RMSE), because the data includes large deviations away from the mean of the data set. The idea is to have an algorithm that is able to cope with the extremes, thus analysing which algorithms perform well when large errors is an important part of the analysis. For derivative pricing, mispricing should be minimised and penalising extreme deviations is favourable to encourage this behaviour. In addition, for reference we also provide the Mean Absolute Error (MAE).

5.1. Predictive accuracy of DGP

We present the findings for all algorithms in Tables 6 and 7. Please note that the DGP algorithm displayed is the method using a GA as the decision criteria.

From looking at Tables 6 and 7, we can observe that DGP was able to outperform the original GP from [24] consistently, as shown by the underlined values. The percentage improvement is approximately 8% on average over the 42 cities, which is a positive result. Some noticeable results from the cities are Oberstdorf and Gorlitz, where the predictive error was reduced by 22% when using DGP. We also note that DGP performs better than GP in 33 data sets. Moreover, DGP was able to predict the best out of all 8 methods 16 times (4 times in Table 6 and 12 times in Table 7), which again shows the real performance gains that can be realised by breaking the process of rainfall down and solving subproblems. By comparison, the second best algorithm regarding the number of victories overall was SVR, which achieved the lowest RMSE in 11 cities.

In order to determine the effectiveness of DGP and to test whether the above results are statistically significant, we compare the eight algorithms by using the Friedman test, which is a non-parametric test based on the mean rank of all algorithms across all data sets (cities) [30]. Our null hypothesis is that all algorithms should perform similarly across the testing set at the 95% confidence level. The results of the Friedman hypothesis test can be found in Table 8, where we also include the mean ranks based on the results from Tables 6 and 7. As our Friedman test statistic was significant at the 5% level (p-value was 1.11x10^{-32}), we use the Holm post-hoc test to compare the control (best) algorithm against each of the others.

From looking at the mean rank within Table 8, DGP is ranked top, achieving the lowest RMSE on average against all other algorithms, showing that the use of decomposition has helped to reduce the average predictive error. Table 8 shows DGP as the control method statistically outperforming all algorithms except for SVR, RBF and GP at the 95% confidence level. We can see the effect that DGP has had on the mix of all algorithms, but the original GP was not significantly outperformed, even though DGP predicted more accurately in 33 (out of 42) cities against GP. Here we can see that DGP performed better in terms of mean rank than the top blackbox methods (RBF and SVR) and GP, and statistically outperformed all other algorithms. Therefore, the predictive error has clearly been reduced by the use of decomposition. Additionally, the runtime of DGP is only 4% greater than the original GP used as a benchmark.

Table 6: The average RMSE and MAE (in brackets) for Europe of DGP against the predecessor GP and other methods. Values in bold represent the best algorithm for each city. Underlined values indicate the lowest predictive error between DGP and GP

City	DGP	GP	SVR	RBF	M5R	M5P	KNN	MCRP
Amsterdam	430.28 (340.50)	430.88 (343.32)	432.94 (353.38)	**422.24** (336.41)	492.97 (393.29)	467.45 (367.31)	473.41 (357.30)	625.15 (494.71)
Arkona	296.66 (216.09)	272.16 (209.53)	235.08 (163.82)	**221.70** (169.56)	306.28 (221.57)	319.63 (214.36)	283.35 (211.03)	414.26 (301.75)
Basel	303.90 (233.45)	293.26 (221.75)	**269.35** (198.25)	309.50 (244.34)	387.07 (292.33)	374.88 (273.67)	277.00 (233.30)	373.18 (286.67)
Bilbao	774.16 (555.48)	783.58 (512.18)	787.14 (511.20)	**729.30** (488.26)	878.28 (678.86)	885.94 (621.36)	949.61 (815.05)	1020.70 (732.38)
Bourges	304.95 (255.11)	322.63 (273.81)	295.80 (251.21)	**289.09** (256.27)	397.57 (314.79)	386.09 (298.50)	325.89 (283.55)	425.89 (356.29)
Caceres	357.46 (287.49)	371.71 (293.81)	**318.10** (223.17)	320.09 (282.48)	370.20 (290.24)	439.19 (327.84)	472.00 (434.54)	385.82 (310.30)
Delft	**455.86** (334.30)	476.01 (375.90)	512.31 (361.16)	483.94 (347.20)	562.26 (423.81)	503.30 (355.80)	518.83 (377.32)	732.90 (537.47)
Gorlitz	257.82 (200.82)	330.30 (267.01)	**253.04** (202.46)	329.80 (260.85)	363.04 (269.68)	406.87 (290.34)	272.46 (222.79)	304.21 (236.96)
Hamburg	332.21 (265.53)	330.08 (273.94)	318.09 (248.09)	**298.62** (250.91)	349.39 (287.40)	355.81 (271.52)	325.05 (247.05)	476.22 (380.64)
Ljubljana	483.81 (398.34)	499.10 (401.69)	**455.43** (372.17)	483.10 (380.62)	666.23 (539.67)	689.07 (551.42)	1183.07 (1088.27)	642.49 (528.99)
Luxembourg	331.67 (277.88)	390.91 (333.49)	364.88 (313.02)	370.43 (324.70)	463.58 (389.44)	509.14 (420.01)	**300.47** (232.70)	384.44 (322.10)
Marseille	372.13 (314.53)	395.81 (331.13)	**334.03** (291.22)	337.08 (297.22)	516.69 (408.83)	432.31 (337.86)	718.98 (653.08)	429.98 (363.43)
Oberstdorf	**436.68** (341.36)	563.98 (408.11)	468.04 (363.31)	475.31 (357.64)	561.20 (450.20)	554.66 (460.34)	679.59 (547.66)	682.52 (533.54)
Paris	268.95 (213.43)	287.83 (227.67)	**260.68** (206.25)	265.59 (212.50)	303.47 (234.93)	316.97 (245.95)	260.76 (216.20)	356.38 (282.82)
Perpignan	396.12 (292.45)	407.00 (323.28)	**383.94** (238.95)	398.48 (306.63)	494.05 (314.00)	469.72 (300.25)	1492.69 (1446.05)	445.26 (328.73)
Potsdam	231.30 (188.94)	243.18 (205.30)	**202.30** (164.75)	222.87 (182.18)	291.61 (253.17)	283.94 (224.19)	344.93 (264.73)	362.87 (296.42)
Regensburg	**269.36** (205.25)	277.66 (211.39)	271.41 (203.76)	270.83 (202.40)	335.37 (253.95)	335.78 (252.14)	240.96 (193.83)	334.62 (254.98)
Santiago	**860.67** (672.76)	1034.02 (773.82)	989.13 (697.74)	914.70 (729.68)	1127.44 (910.01)	1268.65 (987.68)	1379.51 (1176.56)	1068.89 (835.52)
Strijen	**458.05** (306.82)	507.86 (362.69)	523.21 (347.94)	520.90 (345.40)	529.29 (357.57)	569.76 (365.57)	548.57 (361.82)	715.82 (479.48)
Texel	399.90 (303.94)	412.91 (322.70)	**393.25** (281.68)	396.05 (289.97)	412.54 (311.50)	423.88 (292.42)	491.18 (434.87)	611.57 (464.82)

Table 7: The average RMSE and MAE (in brackets) for the USA of DGP against the predecessor GP and other methods. Values in bold represent the best algorithm for each city. Underlined values indicate the lowest predictive error between DGP and GP

City	DGP	GP	SVR	RBF	M5R	M5P	KNN	MCRP
Atlanta	**764.76** (560.25)	799.73 (570.53)	857.53 (614.70)	800.03 (567.83)	868.86 (662.97)	851.39 (652.72)	919.44 (667.31)	1159.81 (849.65)
Boston	**380.26** (322.17)	417.14 (352.21)	388.20 (333.17)	400.56 (341.17)	624.63 (490.70)	535.08 (427.68)	966.33 (912.34)	492.04 (416.87)
Cape Hatteras	**866.71** (562.65)	938.51 (635.10)	1023.58 (651.26)	968.16 (650.66)	1062.44 (644.78)	1111.34 (719.07)	1171.89 (768.91)	1304.73 (847.00)
Cheyenne	342.81 (224.44)	339.62 (249.69)	344.60 (207.86)	353.29 (237.02)	353.58 (229.10)	**297.42** (202.53)	443.19 (261.52)	455.40 (298.16)
Chicago	453.91 (368.56)	498.05 (414.11)	443.42 (358.69)	**433.04** (362.29)	550.44 (429.31)	559.64 (452.43)	716.73 (577.96)	655.31 (532.09)
Cleveland	**474.60** (361.12)	534.76 (418.38)	527.59 (383.25)	538.55 (425.09)	562.07 (437.50)	587.37 (459.37)	532.82 (376.60)	676.48 (514.73)
Dallas	**1070.64** (761.83)	1223.09 (862.93)	1283.32 (892.74)	1248.00 (874.51)	1328.54 (950.38)	1273.65 (917.71)	1437.85 (1051.89)	1415.62 (1007.30)
Des Moines	553.35 (450.10)	582.78 (477.09)	564.35 (452.34)	**498.90** (401.67)	678.43 (569.97)	701.12 (568.51)	1020.52 (820.67)	805.94 (655.55)
Detroit	**358.96** (283.93)	387.56 (311.94)	381.69 (303.27)	363.39 (294.43)	429.08 (343.09)	437.55 (352.00)	450.84 (330.07)	486.93 (385.15)
Indianapolis	**834.96** (557.30)	889.52 (589.66)	891.26 (561.02)	898.76 (625.51)	897.16 (630.90)	919.82 (642.73)	948.64 (582.90)	1047.53 (699.19)
Jacksonville	663.42 (501.32)	630.29 (466.20)	573.48 (418.20)	574.81 (447.87)	607.41 (488.89)	**560.05** (435.64)	731.03 (514.07)	793.05 (599.28)
Kansas	**667.69** (493.86)	700.58 (518.51)	691.15 (502.45)	701.07 (529.98)	743.62 (539.36)	835.84 (634.42)	1038.90 (758.93)	917.71 (678.78)
Las Vegas	104.68 (79.20)	97.71 (77.16)	99.57 (73.92)	**86.26** (66.81)	112.69 (85.92)	151.49 (112.73)	101.03 (73.37)	120.97 (91.52)
Los Angeles	323.20 (239.17)	308.52 (221.04)	**213.25** (143.15)	235.13 (185.46)	281.23 (184.75)	315.29 (210.34)	403.75 (379.33)	276.18 (204.38)
Louisville	**784.55** (621.04)	894.04 (710.69)	899.69 (724.15)	895.82 (704.91)	915.28 (734.19)	981.09 (796.21)	1042.56 (781.04)	1172.18 (927.89)
Nashville	431.00 (348.34)	467.94 (383.12)	424.96 (354.67)	**418.49** (342.25)	556.37 (442.33)	533.51 (412.10)	469.66 (380.73)	698.28 (564.36)
New York	454.65 (366.73)	505.07 (404.63)	**390.83** (313.25)	427.85 (342.34)	764.03 (564.82)	531.26 (429)	683.14 (608.15)	551.05 (444.49)
Phoenix	175.79 (139.92)	148.53 (117.98)	160.45 (124.59)	**128.06** (106.22)	217.63 (170.50)	175.75 (141.30)	133.11 (104.90)	182.97 (145.64)
Portland	**661.44** (454.20)	787.17 (508.44)	777.84 (483.17)	729.94 (455.83)	819.96 (606.23)	841.69 (540.42)	1059.11 (897.26)	969.16 (665.51)
Raleigh	485.30 (375.89)	469.13 (361.01)	543.97 (418.54)	**480.19** (371.24)	613.22 (475.03)	626.52 (494.54)	561.72 (433.19)	846.81 (655.89)
St Louis	**838.33** (616.10)	933.92 (713.83)	1010.01 (711.15)	984.12 (719.05)	981.52 (740.37)	881.00 (649.79)	1091.14 (783.97)	1241.65 (912.50)
Tampa	1125.76 (670.43)	1219.45 (702.89)	1184.48 (679.99)	**1112.94** (658.83)	1278.94 (806.02)	1355.24 (803.35)	1408.39 (855.95)	1491.53 (888.26)

Table 8: The mean ranks of all algorithms, and the Friedman test statistic with the best performing algorithm (DGP) being the control method. Values in bold represent a significant difference.

Friedman test *p*-value	**$1.11\text{x}10^{-32}$**		
Algorithm	Mean rank	*p*-value	Critical value
DGP	2.52	-	-
RBF	2.55	0.96	0.050
SVR	2.67	0.79	0.025
GP	3.62	0.04	0.017
M5P	5.81	**$7.90\text{x}10^{-10}$**	**0.013**
M5R	5.83	**$5.96\text{x}10^{-10}$**	**0.010**
KNN	6.00	**$7.85\text{x}10^{-11}$**	**0.008**
MCRP	7.00	**$5.56\text{x}10^{-17}$**	**0.007**

5.2. *Classification Accuracy of the GA*

We will now investigate which classification accuracy provides the best predictive accuracy for the DGP algorithm. In order to determine the GA's effectiveness we will compare it against other well-established techniques as a benchmark. The results can be found in Tables 9 and 10 based on the same randomly chosen set of *LC* and *UC*.

Table 9: Classification accuracy for Europe shown as a percentage of correctness on the test set. Values in bold show the best algorithm for each city.

Data	GA	SVM	RBF	RIPPER	DA	NB
Amsterdam	**51.10**	47.87	48.92	44.01	39.63	48.82
Arkona	46.04	**50.55**	43.81	44.32	39.60	45.35
Basel	53.56	43.45	**63.12**	55.86	36.67	41.65
Bilbao	49.03	51.25	46.59	**52.81**	41.27	51.51
Bourges	45.85	47.17	**51.18**	44.42	23.03	46.53
Caceres	52.27	40.79	63.69	58.40	52.77	**68.54**
Delft	**53.63**	50.33	48.28	45.79	27.63	37.28
Gorlitz	36.02	40.00	46.77	**47.45**	25.58	40.26
Hamburg	46.19	**51.02**	44.45	47.52	32.24	49.18
Ljubljana	45.46	49.51	44.88	**50.87**	41.55	49.19
Luxembourg	**46.27**	35.60	44.43	38.87	29.61	38.86
Marseille	32.25	46.15	**49.33**	48.55	39.10	40.78
Oberstdorf	47.69	55.88	**59.81**	50.42	33.70	41.35
Paris	50.74	**51.41**	49.45	47.90	28.43	40.79
Perpignan	53.35	**57.35**	55.00	50.93	30.41	45.50
Potsdam	57.33	47.99	**60.41**	47.65	53.98	56.26
Regensburg	39.54	47.05	**53.28**	46.24	38.85	49.59
Santiago	46.80	46.47	**52.18**	44.21	39.17	49.11
Strijen	**49.88**	48.36	42.52	47.36	27.46	30.45
Texel	**59.32**	53.85	59.05	53.29	45.41	47.74

In Tables 9 and 10 we can observe that our GA performs well, just behind the best algorithms of RBF and SVM. More precisely, the GA, RBF and SVM were the winners in 10, 11 and 11 cities, respectively. The experimental setup of this was to test the robustness of each algorithm, which is why the average percentage of correctness for most algorithms appears to be near 50% accuracy. The results here are not directly the same as they will be inside the DGP algorithm, as the class boundaries specified by *LC* and *UC* are randomly selected and are not optimised. One issue with choosing random *LC* and *UC* for decomposition is that the chance of it being optimal is slim and does impact performance. Ideally, the algorithm should be able to perform well with a non optimal splitting of data. Considering the range of all classification techniques, in most cases our GA was very competitive, which was a positive sign. The random selection of the criteria was necessary to avoid bias and to allow for a fair comparison across all classification techniques.

In order to determine whether there were any significant differences between classification techniques we perform the Friedman test at the 95% confidence level and show the results in Table 11. We observe a statistical difference,

Table 10: Classification accuracy for the USA shown as a percentage of correctness on the test set. Values in bold show the best algorithm for each city.

Data	GA	SVM	RBF	RIPPER	DA	NB
Atlanta	49.64	**49.68**	46.55	42.21	25.44	27.98
Boston	**45.63**	36.20	42.25	40.43	28.19	41.32
Cape Hatteras	**64.77**	61.06	60.53	47.13	52.12	56.26
Cheyenne	43.46	**66.50**	57.11	57.28	48.10	46.65
Chicago	30.50	43.45	42.64	40.31	32.33	**45.93**
Cleveland	43.45	37.65	**45.90**	44.24	37.38	35.15
Dallas	42.39	**43.21**	41.79	31.70	39.05	29.36
Des Moines	40.95	53.16	56.32	44.84	48.35	**57.21**
Detroit	39.21	37.10	35.76	37.58	39.65	**40.87**
Indianapolis	36.40	39.79	**50.89**	39.71	41.74	40.88
Jacksonville	48.28	56.36	**58.86**	53.13	46.12	45.03
Kansas	45.61	**58.40**	50.76	49.00	48.91	46.83
Las Vegas	**74.71**	68.74	62.50	57.58	54.55	57.90
Los Angeles	75.70	65.08	74.21	72.14	73.54	**77.83**
Louisville	37.18	**43.08**	33.23	32.95	36.65	34.77
Nashville	41.04	**52.28**	49.45	50.56	25.28	38.65
New York	50.05	35.90	**58.08**	52.97	30.81	47.37
Phoenix	**63.03**	60.92	58.47	58.28	48.50	49.38
Portland	59.38	56.87	54.39	53.27	63.89	**72.47**
Raleigh	47.26	**61.10**	56.48	45.44	42.68	54.48
St Louis	46.14	47.04	45.81	40.79	46.76	**48.82**
Tampa	**69.78**	63.80	65.87	53.23	58.90	51.05

Table 11: The Friedman test statistic along with the results of the Holm post-hoc test at the 95% confidence level, with the best performing algorithm (RBF) being the control method. Values shown in bold represent a significant difference in classification accuracy against the control algorithm.

Friedman test p-value	**1.7925x10^{-10}**		
Algorithm	Mean rank	p-value	Critical value
RBF	2.64	-	-
SVM	2.76	0.771	0.050
GA	3.14	0.221	0.025
NB	3.57	0.023	0.017
RIPPER	3.81	**0.004**	**0.013**
DA	5.07	**2.702x10^{-9}**	**0.010**

as can be seen by the Friedman test p-value of 1.7925x10^{-10}, which is much less than the 5% significance level. Therefore, one or more classification algorithms significantly outperformed at least one other algorithm.

From the perspective of our GA, we observe that it is not significantly outperformed by the best performing classification algorithm of RBF. We believe that the better the classification accuracy the better the performance of DGP, given by classifying more data points accurately. Therefore, based on the mean rank, we would expect under this assumption RBF to perform the best when compared to our DGP with GA. However, a key difference is that the GA rules evolve alongside the GP equations, whereas the other classification algorithms are fixed throughout the GP's evolution. We may observe a substantial number of misclassifications throughout the evolutionary process, which may hinder the generalising ability of DGP.

5.3. DGP Performance Under Different Decision Criteria

We now examine the predictive performance of DGP when we use an alternative classification algorithm. We hope to examine two aspects. Firstly, if using a technique that improves the classification accuracy has a greater effect on lowering the RMSE of DGP. Secondly, whether in the final generation of DGP the decision criteria that maximised the classification accuracy was used by the best performing individual (lowest RMSE) of DGP.

Tables 12 and 13 show the average RMSE of DGP averaged over the testing period using each classification algorithm, along with the mean ranks located at the bottom of the tables. Similar to our previous experimentation, we run DGP for 50 times and initialise 1000 randomly generated *LC* and *UC* combinations (population size) pairing them to a DGP individual throughout evolution. We present the order of algorithms according to the classification accuracy from Tables 9 and 10, with RBF performing the best and DA performing the worst. Interestingly, the respective RMSE of each algorithm is not too dissimilar between the first and last place and considering the mean ranks. One aspect we notice is that there does appear to be a negative correlation across the table looking at the mean ranks, where the higher the classification accuracy, the lower the RMSE error, which is exactly as we anticipated. Taking the combined mean rank across both tables, we notice that RBF ranks first (3.23), GA ranks second (3.26), SVM ranks third (3.48) and the remaining algorithms ranked in the same order as per the classification accuracy. GA was the only algorithm to increase its rank on its predictive error relative to its rank on the classification accuracy (from third to second).

In order to determine whether this relationship does exist between the classification accuracy and the predictive error, we calculate the Pearson product-moment linear correlation coefficient to measure the strength of the relationship. We observe based on the results provided in Tables 12 and 13, as well as Tables 9 and 10, that we obtain a coefficient value of -0.8924, indicating a strong negative linear relationship between classification accuracy and predictive error. We obtain a *p*-value of 0.0167, which is less than the 5% significance level and can conclude that a relationship does exist.

We do notice that the use of GA had an irregular effect on the RMSE and is the anomaly that does not fit the trend. The GA's average predicted error was similar to the classification technique ranked first (RBF), despite classifying third.

We perform the Friedman hypothesis test to determine whether there was a significant effect on the RMSE from the use of different decision criteria. We discover the *p*-value is 0.6675, which is greater than the 5% significance level and so we cannot reject the null hypothesis. Although we do observe a trend that is consistent with our previous analysis of the classification accuracy, there is not enough evidence to suggest that one decision criteria leads to a significant change in RMSE.

This shows promise for our algorithm of DGP, indicating that by having a more accurate classification technique does lead to a reduction in RMSE. As further analysis we also consider what effect each classification technique had on the standard deviation of our DGP predictions. We discover that the average standard deviation was 4.83%, 4.91%, 9.12%, 5.10%, 5.37% and 5.25% for RBF, SVM, GA, NB, RIPPER and DA respectively. From this we can identify why the performance generally fitted the negative correlation between RMSE and the classification accuracy. Here we witness that all classification techniques, except GA, tended to increase the robustness of GP, indicated by the lower RMSE. However, we do see that the standard deviation does increase when using our GA respective to the other algorithms. This is quite an interesting discovery for our DGP, where we observe that keeping consistent decision criteria helps to improve the stability of our DGP's performance, since the same model is used for all algorithms except for our GA.

In the special case of our GA, we can have many rules sets explaining the same *LC* and *UC* class threshold combination, which adds more randomness into our model and hence reflects a larger spread of results. On the other hand, under all other classification algorithms the outcome of using a certain *LC* and *UC* combination is fixed across all DGP generations. We discovered that the best *LC* and *UC* combination is evolved much more efficiently with the final generation of DGP having more similar *LC* and *UC*; whereas with the GA we observe a more mixed set of *LC* and *UC* values. In both cases we did not count the effect from mutation in the previous generation.

To further aid the analysis, we also consider whether the *LC* and *UC* that returned the highest classification accuracy from the final generation of DGP were responsible for the lowest RMSE of our final DGP individual. We include in Tables 14 and 15 the best overall classification accuracy on average from the final generation of DGP and, in brackets, the classification accuracy of the individual that minimised the RMSE of DGP. This analysis will help to understand how the classification part of DGP behaves, which may indicate why the individual with the best classification accuracy does not always lead to a lower RMSE.

Tables 14 and 15 show in almost all cases DGP tended to choose the individual with the best classification accuracy, except for our GA. This is interesting as it appears that one of the benefits is the relationship of our GA evolving alongside that of GP. Meaning that there is the potential for the GP part to be overfitting on the incorrect predictions from the classification algorithm, given that there is only a single model for each *LC* and *UC* combination. Alternatively, there may exist a problem of early convergence, as we noticed little diversity in the *LC* and *UC* of each individual in the final generation. On the other hand, in the final generation the GA had many different classification outcomes with more diverse combinations of *LC* and *UC*. This analysis indicates that through the evolution of our

GA-part (outlined earlier in Section 3.4), DGP can learn from more frequently changing information to avoid early convergence and explore different classification rules.

The results show that the GA was competitive with SVM and RBF (Table 11), and we find that the GA was computationally much more efficient than all classification algorithms. Therefore, we continue with this method as our chosen methodology with any future reference to DGP, referring to using the GA as the underlying classification method, to decide which categorical level of rainfall (low, medium, or high) should be predicted by a GP individual.

Table 12: The average RMSE and MAE in brackets for Europe obtained by DGP when applying different classification algorithms. The best results for each city are shown in bold.

Data	GP + RBF	GP + SVM	GP + GA	GP + NB	GP + RIPPER	GP + DA
Amsterdam	458.08 (363.92)	454.12 (351.09)	430.28 (340.50)	454.72 (368.58)	458.38 (384.84)	448.52 (365.57)
Arkona	**274.71** (192.74)	300.34 (228.27)	296.66 (216.09)	310.63 (231.18)	318.23 (224.24)	304.76 (220.96)
Basel	296.91 (241.00)	309.92 (240.38)	303.90 (233.45)	307.70 (236.46)	306.36 (233.93)	**283.30** (206.27)
Bilbao	775.86 (564.78)	765.18 (537.24)	774.16 (555.48)	**716.64** (486.79)	777.88 (552.04)	813.80 (616.32)
Bourges	**297.57** (262.00)	298.79 (262.96)	304.95 (255.11)	324.28 (257.45)	325.02 (266.81)	313.73 (274.24)
Caceres	381.91 (318.71)	366.68 (289.40)	**357.46** (287.49)	380.95 (318.20)	368.83 (293.01)	357.60 (290.26)
Delft	449.07 (324.74)	458.60 (350.50)	455.86 (334.30)	**438.45** (308.56)	455.54 (315.32)	472.91 (356.86)
Gorlitz	258.90 (193.15)	**241.11** (183.38)	257.82 (200.82)	256.12 (209.56)	249.78 (192.24)	254.75 (203.79)
Hamburg	343.14 (257.61)	342.77 (264.99)	**332.21** (265.53)	342.11 (286.05)	344.04 (297.54)	349.72 (276.44)
Ljubljana	480.23 (378.18)	517.53 (430.46)	483.81 (398.34)	483.71 (403.61)	461.51 (369.06)	**454.73** (361.93)
Luxembourg	320.13 (267.88)	319.20 (262.17)	331.67 (277.88)	329.25 (270.18)	**315.32** (270.52)	335.52 (269.08)
Marseille	372.39 (322.55)	349.76 (303.83)	372.13 (314.53)	**344.93** (278.16)	345.45 (295.47)	368.97 (330.24)
Oberstdorf	408.51 (309.82)	456.07 (370.36)	436.68 (341.36)	446.42 (343.24)	439.39 (328.72)	**404.85** (314.24)
Paris	287.88 (241.33)	274.95 (225.53)	**268.95** (213.43)	283.77 (212.51)	288.58 (219.59)	278.23 (218.75)
Perpignan	382.34 (289.00)	373.74 (286.21)	396.12 (292.45)	**366.41** (254.28)	384.24 (251.66)	410.38 (321.54)
Potsdam	240.27 (205.86)	243.10 (200.45)	231.30 (188.94)	232.27 (186.34)	242.19 (190.82)	**228.13** (195.47)
Regensburg	254.41 (181.43)	258.34 (209.56)	269.36 (205.25)	266.96 (209.59)	264.46 (209.37)	**254.09** (197.01)
Santiago	**800.94** (637.49)	823.75 (632.42)	860.67 (672.76)	925.48 (717.34)	880.12 (658.20)	890.02 (715.76)
Strijen	**428.41** (285.01)	440.96 (300.47)	458.05 (306.82)	449.67 (305.82)	439.73 (289.16)	436.80 (272.69)
Texel	**380.10** (276.37)	389.62 (283.19)	399.90 (303.94)	384.54 (307.74)	383.62 (300.52)	428.45 (315.51)
Mean rank	**3.10**	3.40	3.45	3.65	3.90	3.50

Table 13: The average RMSE and MAE in brackets for the USA obtained by DGP when applying the different classification algorithms. The best results for each city are shown in bold.

Data	GP + RBF	GP + SVM	GP + GA	GP + NB	GP + RIPPER	GP + DA
Atlanta	747.63 (538.61)	756.04 (565.76)	764.76 (560.25)	725.30 (544.66)	**711.84** (569.15)	740.13 (566.62)
Boston	373.80 (302.44)	**360.33** (309.08)	380.26 (322.17)	398.70 (333.49)	387.64 (317.97)	390.79 (333.80)
Cape Hatteras	866.19 (576.22)	861.94 (571.66)	866.71 (562.65)	914.99 (573.04)	918.71 (620.37)	**839.84** (569.79)
Cheyenne	**327.90** (210.01)	348.16 (232.54)	342.81 (224.44)	351.11 (228.42)	346.03 (239.98)	357.28 (247.71)
Chicago	475.15 (362.73)	472.52 (383.62)	**453.91** (368.56)	473.70 (390.61)	456.91 (398.91)	482.14 (370.19)
Cleveland	497.05 (391.84)	483.95 (347.46)	**474.60** (361.12)	492.87 (391.53)	485.28 (362.12)	483.90 (357.01)
Dallas	1021.28 (713.67)	**992.06** (658.29)	1070.64 (761.83)	1022.03 (776.14)	1022.25 (762.16)	1048.16 (773.80)
Des Moines	515.94 (427.08)	542.17 (435.97)	553.35 (450.10)	526.90 (413.47)	**512.57** (404.42)	566.19 (467.31)
Detroit	385.02 (322.10)	**348.05** (265.85)	358.96 (283.93)	356.91 (288.56)	359.10 (305.24)	373.28 (277.04)
Indianapolis	783.53 (522.17)	783.61 (547.46)	834.96 (557.30)	**772.34** (529.44)	797.47 (524.29)	887.73 (596.64)
Jacksonville	710.52 (530.10)	668.00 (516.63)	**663.42** (501.32)	702.10 (561.14)	678.75 (561.86)	688.30 (495.67)
Kansas	631.03 (495.21)	685.32 (540.74)	667.69 (493.86)	**622.69** (454.60)	625.69 (459.69)	695.07 (515.44)
Las Vegas	107.07 (84.08)	104.76 (79.4)	**104.68** (79.20)	107.59 (80.94)	106.54 (80.13)	105.84 (82.86)
Los Angeles	339.04 (256.27)	324.53 (244.65)	323.20 (239.17)	313.83 (233.42)	345.95 (247.26)	**300.48** (219.24)
Louisville	790.51 (620.97)	802.83 (634.56)	784.55 (621.04)	793.42 (637.4)	**762.19** (629.99)	811.07 (650.31)
Nashville	426.17 (328.31)	438.93 (361.40)	431.00 (348.34)	427.25 (366.62)	**401.39** (349.57)	436.82 (369.55)
New York	442.15 (367.22)	450.15 (352.04)	454.65 (366.73)	439.65 (367.29)	**421.19** (367.24)	463.42 (369.72)
Phoenix	186.23 (155.18)	182.08 (147.15)	175.79 (139.92)	168.83 (135.21)	165.35 (138.76)	**164.72** (126.56)
Portland	**629.23** (411.17)	705.43 (485.26)	661.44 (454.20)	693.06 (474.28)	691.20 (461.35)	658.93 (424.98)
Raleigh	490.54 (363.38)	**450.50** (352.62)	485.30 (375.89)	491.41 (380.61)	508.69 (406.52)	497.72 (365.20)
St Louis	869.01 (675.11)	891.23 (663.90)	**838.33** (616.1)	845.2 (627.30)	874.63 (648.75)	881.25 (622.07)
Tampa	**1112.70** (618.38)	1151.77 (681.40)	1125.76 (670.43)	1139.83 (681.58)	1153.00 (684.95)	1133.19 (711.47)
Mean rank	3.36	3.55	**3.09**	3.50	3.33	4.18

Table 14: The average classification accuracy in the final generation of DGP for Europe, with the average classification accuracy that provided the lowest RMSE for DGP in brackets.

Data	GP + RBF	GP + SVM	GP + GA	GP + NB	GP + RIPPER	GP + DA
Amsterdam	0.717 (0.713)	0.797 (0.797)	0.756 (0.716)	0.735 (0.731)	0.750 (0.749)	0.831 (0.830)
Arkona	0.807 (0.803)	0.766 (0.765)	0.704 (0.643)	0.715 (0.715)	0.849 (0.845)	0.759 (0.753)
Basel	0.828 (0.821)	0.865 (0.862)	0.707 (0.652)	0.713 (0.713)	0.845 (0.845)	0.775 (0.768)
Bilbao	0.701 (0.700)	0.825 (0.817)	0.781 (0.732)	0.797 (0.796)	0.773 (0.773)	0.678 (0.676)
Bourges	0.761 (0.756)	0.763 (0.762)	0.832 (0.756)	0.836 (0.828)	0.817 (0.816)	0.735 (0.732)
Caceres	0.798 (0.791)	0.843 (0.838)	0.853 (0.806)	0.847 (0.841)	0.800 (0.800)	0.835 (0.833)
Delft	0.818 (0.817)	0.775 (0.775)	0.822 (0.771)	0.792 (0.784)	0.784 (0.784)	0.716 (0.715)
Gorlitz	0.682 (0.680)	0.691 (0.687)	0.679 (0.633)	0.646 (0.644)	0.649 (0.647)	0.672 (0.670)
Hamburg	0.865 (0.858)	0.781 (0.777)	0.846 (0.786)	0.704 (0.700)	0.810 (0.808)	0.794 (0.791)
Ljubljana	0.810 (0.805)	0.834 (0.834)	0.764 (0.701)	0.718 (0.712)	0.733 (0.733)	0.660 (0.654)
Luxembourg	0.845 (0.837)	0.743 (0.743)	0.701 (0.631)	0.848 (0.841)	0.861 (0.854)	0.761 (0.757)
Marseille	0.682 (0.680)	0.703 (0.699)	0.648 (0.592)	0.660 (0.653)	0.610 (0.604)	0.615 (0.614)
Oberstdorf	0.797 (0.792)	0.727 (0.722)	0.788 (0.747)	0.821 (0.817)	0.786 (0.786)	0.733 (0.726)
Paris	0.820 (0.812)	0.835 (0.834)	0.770 (0.693)	0.735 (0.734)	0.745 (0.739)	0.839 (0.838)
Perpignan	0.800 (0.798)	0.674 (0.669)	0.798 (0.750)	0.838 (0.829)	0.774 (0.769)	0.734 (0.730)
Potsdam	0.794 (0.794)	0.712 (0.712)	0.701 (0.666)	0.721 (0.714)	0.845 (0.837)	0.799 (0.798)
Regensburg	0.746 (0.746)	0.652 (0.647)	0.696 (0.651)	0.734 (0.732)	0.660 (0.654)	0.627 (0.621)
Santiago	0.815 (0.809)	0.854 (0.854)	0.742 (0.700)	0.739 (0.735)	0.709 (0.704)	0.807 (0.799)
Strijen	0.724 (0.720)	0.775 (0.774)	0.768 (0.705)	0.799 (0.792)	0.734 (0.734)	0.652 (0.650)
Texel	0.897 (0.892)	0.871 (0.864)	0.811 (0.762)	0.824 (0.822)	0.793 (0.787)	0.703 (0.699)

Table 15: The average classification accuracy in the final generation of DGP for the USA, with the average classification accuracy that provided the lowest RMSE for DGP in brackets.

Data	GP + RBF	GP + SVM	GP + GA	GP + NB	GP + RIPPER	GP + DA
Atlanta	0.752 (0.750)	0.830 (0.824)	0.737 (0.665)	0.734 (0.734)	0.721 (0.720)	0.709 (0.702)
Boston	0.769 (0.763)	0.803 (0.800)	0.808 (0.758)	0.787 (0.784)	0.769 (0.766)	0.724 (0.722)
Cape Hatteras	0.661 (0.661)	0.707 (0.707)	0.785 (0.735)	0.806 (0.804)	0.753 (0.749)	0.692 (0.689)
Cheyenne	0.720 (0.713)	0.745 (0.738)	0.805 (0.750)	0.773 (0.767)	0.761 (0.757)	0.797 (0.793)
Chicago	0.732 (0.730)	0.735 (0.727)	0.672 (0.625)	0.714 (0.707)	0.667 (0.663)	0.616 (0.615)
Cleveland	0.653 (0.648)	0.720 (0.720)	0.771 (0.704)	0.765 (0.764)	0.739 (0.738)	0.711 (0.711)
Dallas	0.806 (0.806)	0.850 (0.842)	0.733 (0.670)	0.703 (0.699)	0.720 (0.715)	0.629 (0.625)
Des Moines	0.776 (0.772)	0.680 (0.674)	0.696 (0.637)	0.720 (0.718)	0.689 (0.688)	0.617 (0.612)
Detroit	0.551 (0.548)	0.553 (0.550)	0.617 (0.577)	0.583 (0.579)	0.604 (0.598)	0.568 (0.567)
Indianapolis	0.699 (0.694)	0.761 (0.757)	0.675 (0.632)	0.703 (0.702)	0.661 (0.655)	0.649 (0.647)
Jacksonville	0.841 (0.836)	0.836 (0.828)	0.726 (0.665)	0.739 (0.739)	0.869 (0.861)	0.824 (0.818)
Kansas	0.790 (0.788)	0.794 (0.787)	0.856 (0.795)	0.845 (0.845)	0.843 (0.838)	0.809 (0.808)
Las Vegas	0.872 (0.867)	0.958 (0.948)	0.827 (0.766)	0.848 (0.841)	0.800 (0.793)	0.796 (0.790)
Los Angeles	0.878 (0.871)	0.997 (0.990)	0.866 (0.796)	0.897 (0.894)	0.847 (0.842)	0.845 (0.837)
Louisville	0.629 (0.627)	0.638 (0.634)	0.641 (0.593)	0.678 (0.674)	0.622 (0.615)	0.639 (0.634)
Nashville	0.727 (0.722)	0.803 (0.802)	0.785 (0.714)	0.799 (0.799)	0.761 (0.760)	0.659 (0.657)
New York	0.820 (0.815)	0.786 (0.782)	0.853 (0.780)	0.818 (0.813)	0.822 (0.815)	0.848 (0.848)
Phoenix	0.702 (0.701)	0.867 (0.862)	0.816 (0.740)	0.773 (0.765)	0.804 (0.801)	0.805 (0.805)
Portland	0.813 (0.807)	0.746 (0.745)	0.843 (0.774)	0.821 (0.815)	0.830 (0.825)	0.835 (0.829)
Raleigh	0.728 (0.721)	0.732 (0.729)	0.819 (0.777)	0.838 (0.832)	0.776 (0.773)	0.700 (0.696)
St Louis	0.795 (0.791)	0.765 (0.760)	0.696 (0.646)	0.712 (0.706)	0.670 (0.668)	0.588 (0.583)
Tampa	0.755 (0.754)	0.804 (0.800)	0.836 (0.760)	0.864 (0.861)	0.786 (0.780)	0.778 (0.772)

Table 16: Average run times of all algorithms presented in the results. The DGP average time represents the average across all types of classification algorithms.

Average regression run times							
DGP	GP	SVR	RBF	M5R	M5P	KNN	MCRP
231.0sec	194.7sec	11.1sec	3.1sec	131.2sec	8.4sec	< 1sec	242.5sec
Average classification run times							
RBF	SVM	GA	NB	RIPPER	DA		
1.2sec	7.3sec	9.4sec	< 1sec	3.8sec	< 1sec		

Lastly, the average computational time per run is provided in Table 16. Note, this is the average run time across all data sets for each run required of the algorithm. As we can observe, the DGP is slower than the other algorithms (with the exception of MCRP). This is expected, as many of the other algorithms (e.g. SVR) follow a deterministic procedure, while GP and DGP create multiple candidate trees before finding the best tree. In addition, computational time has a relatively minor importance in this field, since it represents an off-line application. Hence, the introduced improvements in DGP's performance justify the slower execution speed of the algorithm. Furthermore, GP algorithms can be easily parallelised since each tree builds and evaluates a candidate solution independently from all other trees in the population. Therefore, a large speed up could be obtained by running a parallel version of the DGP algorithm.

6. Effectiveness of the DGP algorithm

Within this section, we consider the effect that the problem decomposition (i.e., evolving a separate equation for each rainfall class) has had on DGP's ability to predict more similarly to the underlying data. [24] noted that GP without decomposition tended to produce equations with flat predictions and was unable to meet the oscillations of the time series. To consider this we analyse the effect that DGP has had on the coverage of the predictions and whether DGP is able to overcome any climatic issues.

6.1. Effect on Increasing the Coverage of Predictions

One of the motivations of DGP was to improve the behaviour of GP by the use of decision criteria to choose an equation that specialises in the wetter or drier periods.

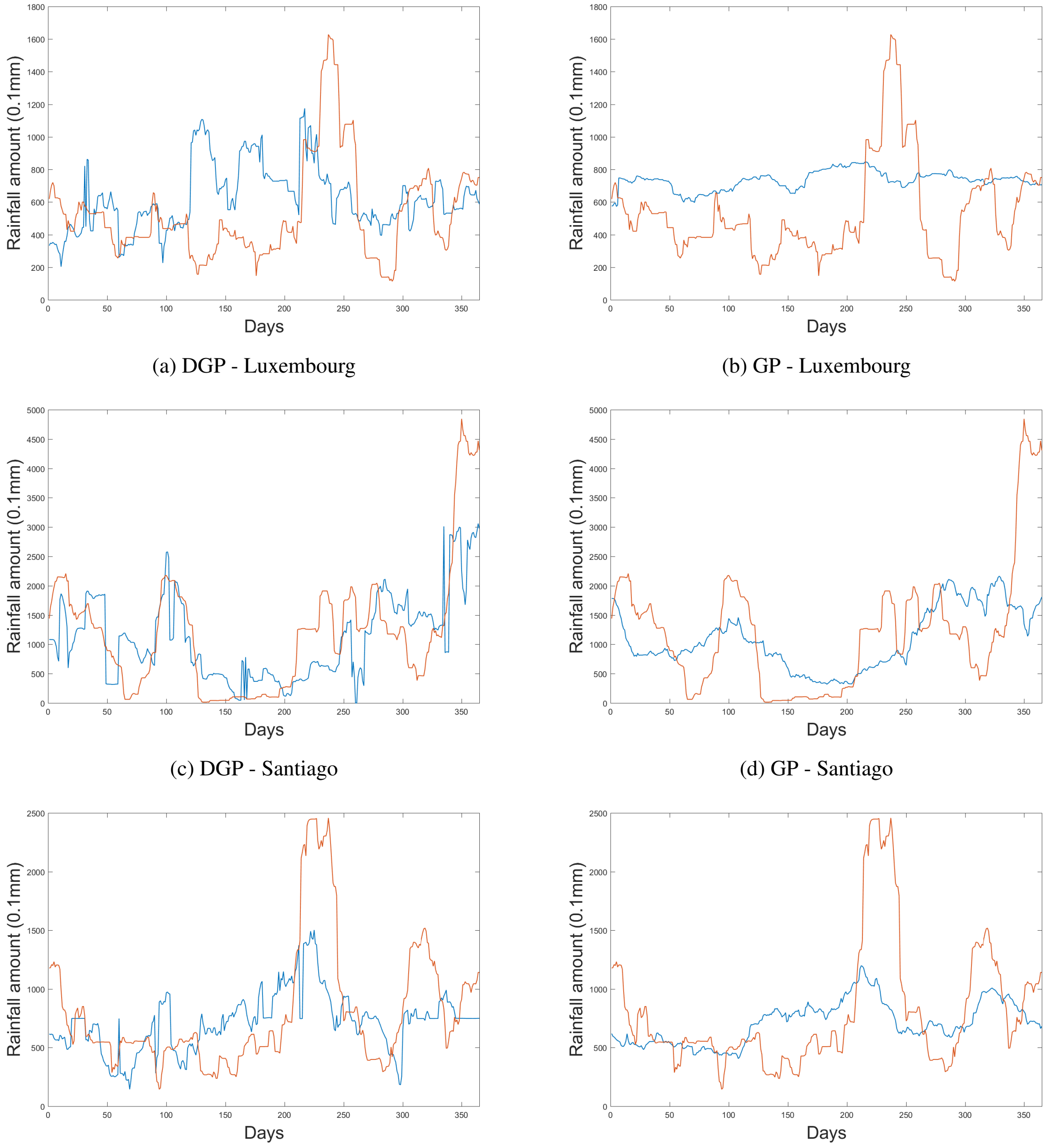
We show in Figure 6 an example comparing the predictions of a DGP individual against the predictions of a GP individual for three cities on the testing set. For each algorithm, we chose the individual that produced the lowest RMSE error on training over all 50 runs. What we observe from this, is that DGP does appear to predict the highs and lows more consistently. Moreover, the predictions are similar to the underlying data where we can observe the more volatile periods. We do generally witness the problem with coverage, where visually it appears that DGP does cover more points, and GP does tend to provide flatter predictions in some examples.

Coverage is formally defined as the percentage between the range of each algorithm's predictions and the range of rainfall in the data set, given by:

$$Coverage = \frac{r_{max} - r_{min}}{\hat{r}_{max} - \hat{r}_{min}} \tag{9}$$

where r represents the predicted rainfall amounts, and $\hat{r}$ represents the rainfall amounts observed in the dataset. If $r_{min} < \hat{r}_{min}$, then we set $r_{min} = \hat{r}_{min}$. Similarly, if $r_{max} > \hat{r}_{max}$, then we set $r_{max} = \hat{r}_{max}$.

We provide the full coverage results in Table 17 to compare DGP and GP across all data sets over 50 runs. From Table 17, we can observe in every city that DGP was able to cover a wider range of rainfall values than GP, when the coverage for GP was less than 100%. *There were no occurrences where DGP covered less rainfall values than GP, and in several cities DGP's coverage was much higher than GP's coverage.* Therefore, we can take away the advantage that DGP has over its predecessor and its ability to increase the coverage and as shown from the figures create equations that predict rainfall amounts more similar to that of the underlying data of accumulated rainfall.



(a) DGP - Luxembourg

(b) GP - Luxembourg

(c) DGP - Santiago

(d) GP - Santiago

(e) DGP - Strijen

(f) GP - Strijen

Figure 6: Rainfall time series for Luxembourg, Santiago and Strijen on the testing set from Jan-01-2015 until Dec-31-2015 for DGP (left) and GP (right). The orange line is the actual accumulated level of rainfall and the blue line is the rainfall level predicted by the best individual from training over 50 runs.

6.2. *Effect on Climate*

Lastly, we discuss how the algorithm has performed considering the same climatic features outlined in [24] and whether the use of decomposition has helped predict the underlying data of rainfall better. We consider the effect of DGP using GA as our underlying classification method. The outlined research questions are as follows:

- Is the predictive error similar between Europe and the USA?
- Are drier or wetter climates associated with a lower predictive error?
- Are more volatile cities associated with higher predictive error?
- Are high rainfall intensities associated with higher predictive error?

By investigating the above research questions, we hope to understand the effect of these issues in the predictive performance of DGP.

Table 17: The coverage (in %) in terms of the range of observed rainfall values covered by the predictions of each algorithm on all cities, for GP and DGP

Data	GP	DGP	Data	GP	DGP
Amsterdam	37%	65%	Boston	31%	44%
Arkona	85%	100%	Capehatteras	23%	61%
Basel	66%	93%	Cheyenne	46%	72%
Bilbao	35%	54%	Chicago	58%	77%
Bourges	35%	61%	Cleveland	16%	57%
Caceres	100%	100%	Dallas	12%	49%
Delft	41%	72%	Des Moines	59%	100%
Gorlitz	84%	99%	Detroit	28%	53%
Hamburg	47%	62%	Indianapolis	13%	71%
Ljubljana	57%	59%	Jacksonville	46%	89%
Luxembourg	34%	62%	Kansas	38%	83%
Marseille	80%	85%	Las Vegas	81%	100%
Oberstdorf	76%	86%	Los Angeles	92%	100%
Paris	71%	83%	Louisville	21%	37%
Perpignan	57%	100%	Nashville	59%	59%
Potsdam	75%	81%	New York	50%	58%
Regensburg	78%	100%	Phoenix	81%	100%
Santiago	36%	66%	Portland	44%	71%
Strijen	22%	61%	Raleigh	37%	49%
Texel	45%	72%	St Louis	26%	64%
Atlanta	16%	41%	Tampa	44%	78%

The first research question is the effect across the two distinct geographic regions of Europe and the USA. We apply the Mann-Whitney test to determine if the predictive error is consistent across both continents. For DGP we obtain a p-value of 0.7721 which is greater than the 5% significance level, thus we can confidently say that the predictive error is similar between Europe and the USA.

In order to investigate the next three research questions we consider the correlation between the descriptive statistical points and the predictive error of our DGP. We present the findings in Table 18, using the Pearson's product-moment linear correlation coefficient (r) to measure the strength of the relationship. Additionally, we include the p-value computed by the Student's t distribution, in order to determine whether there is a statistically significant relationship between the predictive error and the descriptive statistics. The null hypothesis for the test is that $r = 0$. We only include our original GP as a comparison, because we are only considering whether DGP has lead to an improvement over that GP. The values highlighted in bold indicate a statistically significant relationship at the 5% significance level.

Based on the information presented in Table 18, considering the dryness and volatility of cities, these city properties are not significantly correlated with DGP's and GP's predictive error, given the p-value is higher than our significance level in both cases of Europe and the USA, for both algorithms.

Table 18: The linear correlation coefficient (r) and p-value for European and cities from the USA, in order to determine whether there is sufficient evidence that a relationship exists between a data set property and an algorithm's predictive error. The p-value is shown in brackets below the correlation coefficient. Significant relationships (p < 0.05) are shown in bold.

Data set property	DGP		GP	
	USA	Europe	USA	Europe
% of dry days	0.02 (0.9446)	0.27 (0.2571)	-0.30 (0.1683)	0.08 (0.7454)
Average dry spell	0.21 (0.3539)	0.08 (0.7256)	-0.42 (0.0539)	0.23 (0.3315)
Average wet spell	0.24 (0.3410)	-0.23 (0.3577)	-0.08 (0.7362)	0.13 (0.5811)
Annual rainfall	0.06 (0.7940)	0.27 (0.2528)	-0.37 (0.0926)	0.10 (0.6805)
Volatility of annual rainfall	0.28 (0.2140)	0.07 (0.7547)	-0.37 (0.0894)	0.26 (0.2745)
Highest intensity	-0.30 (0.1719)	-0.07 (0.7557)	**0.49** **(0.0199)**	0.42 (0.0641)
Interquartile range of intensity	0.35 (0.1105)	-0.35 (0.1351)	**-0.44** **(0.0396)**	**-0.58** **(0.0071)**

Finally, considering the high rainfall intensities, we observe that this factor was significantly correlated with the GP's predictive error in the USA, with a p-value of 0.038, but not for Europe. DGP's predictive error shows no significant correlation with rainfall intensity within the USA and Europe, in both cases with a p-value greater than 0.05.

To conclude the above analysis, the relationships provided for DGP have shown us that the DGP algorithm is more robust than the GP algorithm against different climates, from across different geographical regions.

7. Conclusion

Within this paper, we presented an extensive evaluation of the Decomposed Genetic Programming (DGP) algorithm for the problem of rainfall within weather derivatives. DGP was proposed as a way to overcome the potential issues highlighted in previous work where we observed that GP was unable to consistently provide equations suitable for the underlying problem of rainfall. Therefore, we aimed to address this issue by thoroughly examining DGP to determine if the correct behaviour exists in our final equations.

DGP is a novel algorithm (recently published in [18]) based on the use of decomposition on the problem of rainfall. The idea revolves around breaking the problem of rainfall into subproblems for our GP to solve, and then recombining the subproblems back into a solution for the original problem. A Genetic Algorithm (GA) was used as a classification technique, because DGP needed to choose which regression equation was evaluated (rebuild back into the whole problem). We additionally evaluated the use of other classification algorithms as the decision process to substitute for the GA. In this work we have extended our previous work on DGP [18] in five different directions, as discussed in the last but one paragraph of the Introduction.

From the results we discussed, we can draw the following conclusions: (i) DGP is an effective regression algorithm for rainfall datasets, as its predictive error ranked the lowest, when compared to the state-of-the-art algorithm of MCRP and other six machine learning algorithms, (ii) GA is an effective algorithm for handling the classification task of DGP, as it demonstrated it is competitive to several other strong classification algorithms, (iii) DGP is able to predict rainfall amounts similar to the amount in the underlying data, as it consistently produces equations that are able to reflect the extreme oscillations that exist within the rainfall time series, and (iv) DGP is not very affected by climatic features, as its predictive error was not significantly correlated with variations in most climatic aspects.

Future research should continue looking into the DGP algorithm by analysing it on other problem domains. Additionally, the GA can be extended further, as it shows great promise in solving the final problem. Extensions can include the creation of multiple rules for dynamically changing time landscapes, in an attempt to improve the problems caused by the irregularities of rainfall. Furthermore, considering change point models for the condition of switching regression models may lead to a more dynamic representation of rainfall and account for sudden irregular patterns.

References

[1] P. Alaton, B. Djehince, D. Stillberg, On modelling and pricing weather derivatives, Applied Mathematical Finance 9 (2002) 1–20.

[2] F. E. Benth, J. Saltyte-Benth, Stochastic modelling of temperature variations with a view towards weather derivatives, Applied Mathematical Finance 12 (1) (2005) 53–85.

[3] A. Agapitos, M. O´ Neill, A. Brabazon, Genetic programming for the induction of seasonal forecasts: A study on weather derivatives, in: Financial Decision Making Using Computational Intelligence, Springer, 2012, pp. 159–188.

[4] A. Agapitos, M. O´ Neill, A. Brabazon, Evolving seasonal forecasting models with genetic programming in the context of pricing weather-derivatives, in: Applications of Evolutionary Computation, Springer, 2012, pp. 135–144.

[5] A. K. Alexandridis, M. Kampouridis, S. Cramer, A comparison of wavelet networks and genetic programming in the context of temperature derivatives, International Journal of Forecasting 33 (1) (2017) 21 – 47.

[6] A. Alexandridis, A. Zapranis, Weather Derivatives: Modeling and Pricing Weather-Related Risk, New York, NY: Springer New York, 2013.

[7] D. S. Wilks, Multisite generalization of a daily stochastic precipitation generation model, Journal of Hydrology 210 (1998) 178–191.

[8] B. L. Cabrera, M. Odening, M. Ritter, Pricing rainfall futures at the CME, Journal of Banking & Finance 37 (11) (2013) 4286 – 4298.

[9] M. Ritter, O. Muhoff, M. Odening, Minimizing geographical basis risk of weather derivatives using a multi-site rainfall model, Computational Economics 44 (1) (2014) 67–86.

[10] M. Cao, A. Li, J. Z. Wei, Precipitation modeling and contract valuation, The Journal of Alternative Investments 7 (2) (2004) 93–99.

[11] M. Odening, O. Musshoff, W. Xu, Analysis of rainfall derivatives using daily precipitation models: opportunities and pitfalls, Agricultural Finance Review 67 (1) (2007) 135–156.

[12] R. W. Katz, Precipitation as a chain-dependent process, Journal of Applied Meteorology and Climatology 16 (7) (1977) 671–676.

[13] E. Carlstein, Nonparametric change-point estimation, The Annals of Statistics 16 (1) (1988) 188–197. doi:10.1214/aos/1176350699.

[14] S. Chib, Estimation and comparison of multiple change-point models, Journal of Econometrics 86 (2) (1998) 221 – 241.

[15] S. I. M. Ko, T. T. L. Chong, P. Ghosh, Dirichlet process hidden markov multiple change-point model, Bayesian Analysis 10 (2) (2015) 275–296. doi:10.1214/14-BA910.

[16] C. Beaulieu, J. Chen, J. L. Sarmiento, Change-point analysis as a tool to detect abrupt climate variations, Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences 370 (1962) (2012) 1228–1249. doi:10.1098/rsta.2011.0383.

[17] D. Cucina, M. Rizzo, E. Ursu, Multiple changepoint detection for periodic autoregressive models with an application to river flow analysis, Preprint, available online: arXiv:1801.01697 (2018).

[18] S. Cramer, M. Kampouridis, A. Freitas, A genetic decomposition algorithm for predicting rainfall within financial weather derivatives, in: Proceedings of the Genetic and Evolutionary Computation Conference 2016, GECCO '16, ACM, New York, NY, USA, 2016, pp. 885–892.

[19] N. Q. Hung, M. S. Babel, S. Weesakul, N. K. Tripathi, An artificial neural network model for rainfall forecasting in bangkok, thailand, Hydrology and Earth System Sciences 13 (8) (2009) 1413–1425.

[20] J. Wu, J. Long, M. Liu, Evolving RBF neural networks for rainfall prediction using hybrid particle swarm optimization and genetic algorithm, Neurocomputing 148 (2015) 136 – 142.

[21] Mislan, Haviluddin, S. Hardwinarto, Sumaryono, M. Aipassa, Rainfall monthly prediction based on artificial neural network: A case study in tenggarong station, east Kalimantan - Indonesia, Procedia Computer Science 59 (2015) 142 – 151, international Conference on Computer Science and Computational Intelligence (ICCSCI 2015).

[22] H. Weerasinghe, H. Premaratne, D. Sonnadara, Performance of neural networks in forecasting daily precipitation using multiple sources, Journal of the National Science Foundation of Sri Lanka 38 (3) (2010) 163–170.

[23] O. Kisi, J. Shiri, Precipitation forecasting using wavelet-genetic programming and wavelet-neuro-fuzzy conjunction models, Water Resources Management 25 (13) (2011) 3135–3152.

[24] S. Cramer, M. Kampouridis, A. A. Freitas, A. K. Alexandridis, An extensive evaluation of seven machine learning methods for rainfall prediction in weather derivatives, Expert Systems with Applications 85 (2017) 169 – 181.

[25] S. Cramer, M. Kampouridis, A. A. Freitas, A. Alexandridis, Predicting rainfall in the context of rainfall derivatives using genetic programming, in: Computational Intelligence for Financial Engineering and Economics, 2015 IEEE Symposium Series on, 2015, pp. 711–718.

[26] S. Cramer, M. Kampouridis, A. A. Freitas, Feature engineering for improving financial derivatives-based rainfall prediction, in: Proceedings of 2016 IEEE Congress on Evolutionary Computation, IEEE Press, Vancouver, 2016.

[27] J. Koza, Genetic Programming: On the programming of computers by means of natural selection, MIT Press, 1992.

[28] D. J. Montana, Strongly typed genetic programming, Evol. Comput. 3 (2) (1995) 199–230.

[29] M. López-Ibáñez, J. Dubois-Lacoste, T. Stützle, M. Birattari, The R package (irace) package, Iterated Race for automatic algorithm configuration, Tech. rep., IRIDIA, Université Libre de Bruxelles, Belgium (2011).

[30] J. Demšar, Statistical comparisons of classifiers over multiple data sets, J. Mach. Learn. Res. 7 (2006) 1–30.