Feature Engineering for Improving Financial Derivatives-based Rainfall Prediction

Sam Cramer
School of Computing
University of Kent
Email: sc649@kent.ac.uk

Michael Kampouridis
School of Computing
University of Kent
Email: M.Kampouridis@kent.ac.uk

Alex A. Freitas
School of Computing
University of Kent
Email: A.A.Freitas@kent.ac.uk

***Abstract*—Rainfall is one of the most challenging variables to predict, as it exhibits very unique characteristics that do not exist in other time series data. Moreover, rainfall is a major component and is essential for applications that surround water resource planning. In particular, this paper is interested in extending previous work carried out on the prediction of rainfall using Genetic Programming (GP) for rainfall derivatives. Currently in the rainfall derivatives literature, the process of predicting rainfall is dominated by statistical models, namely using a Markov-chain extended with rainfall prediction (MCRP). In this paper we further extend our new methodology by looking at the effect of feature engineering on the rainfall prediction process. Feature engineering will allow us to extract additional information from the data variables created. By incorporating feature engineering techniques we look to further tailor our GP to the problem domain and we compare the performance of the previous GP, which previously statistically outperformed MCRP, against our new GP using feature engineering on 21 different data sets of cities across Europe and report the results. The goal is to see whether GP can outperform its predecessor without extra features, which acts as a benchmark. Results indicate that in general GP using extra features significantly outperforms a GP without the use of extra features.**

I. Introduction

Predicting rainfall is a major component and is essential for applications that surround water resource planning and management. Over the years numerous attempts have been made at capturing rainfall. One area where it is vital to predict the rainfall amount accurately is within rainfall derivatives. Rainfall derivatives fall under the umbrella concept of weather derivatives, which are similar to regular derivatives defined as contracts between two or more parties, whose value is dependent upon the underlying asset. In the case of weather derivatives, the underlying asset is a weather type, such as temperature or rainfall. The main difference between normal derivatives and weather derivatives is that weather is not tradeable. Hence, typical methods that exist in the literature for other derivatives are not suitable for weather derivatives.

In this problem domain the underlying asset is the accumulated rainfall over a given period, which is why it is crucial to predict rainfall as accurately as possible to reduce potential mispricing. Contracts based on the rainfall index are decisive for farmers and other users whose income is directly or indirectly affected by the rain. A lack or too much rainfall is capable of destroying a farmer's crops, hence their income. Thus, rainfall derivatives are a method for reducing the risk posed by adverse or uncertain weather circumstances. Moreover, they are a better alternative than insurance, because it can be hard to prove that the rainfall has had an impact unless it is destructive, such as severe floods or drought. Similar contracts exist for other weather variables, such as temperature.

Within the literature rainfall derivatives is split into two main parts. Firstly, predicting the level of rainfall over a specified time and secondly, pricing the derivatives based on different contract periods/length. The latter has its own unique problem, as rainfall derivatives constitute an incomplete market[1]. This means the standard pricing models such as the Black-Scholes model are incapable of pricing rainfall derivatives, because of the violation of the assumptions of the model; namely no arbitrage pricing. Thus, a new pricing framework needs to be established. This paper focuses on the first aspect of predicting the level of rainfall. Note it is essential to have a model that can accurately predict the level of rainfall, before pricing derivatives, because the contracts are priced on the predicted accumulated rainfall over a period of time.

In order to predict the level of rainfall for rainfall derivatives, the statistical approaches of Markov-chain extended with rainfall prediction (MCRP) [1] and spatial-temporal rainfall (STR) models [2] is used. By predicting the underlying variable of rainfall, this increases the accuracy of pricing, which is crucial because contracts are priced ahead of time—sometimes this can be up to a year ahead. Please note we are only interested in the approaches that are currently used within the rainfall derivatives literature, because the problem domain of predicting accumulated rainfall amounts is different than applications such as rainfall-runoff or other time-series based applications. Rainfall-runoff are concerned with either short-run predictions, requiring data up to an hour or requiring radar data depending on the application and do not model rainfall directly, but use rainfall indirectly to the problem domain.

The amount of literature surrounding rainfall derivatives is quite light, due to rainfall derivatives being quite a new concept and rainfall being very difficult to accurately measure. Therefore, we focus on the rainfall prediction process by developing a methodology that can predict rainfall as accurately

[1]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.

as possible, noted earlier. The general approach of MCRP is often referred to as a 'chain-dependent process' [3], which splits the model into capturing first the occurrence pattern, and then 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 [1] for a complete description and to [4] where MCRP was most recently applied for rainfall derivatives. The alternative STR methodology is based on the simulation of the underlying physical processes that govern rainfall. The methodology models the storm arrivals and how it develops and decays over time using a poisson jump process. We refer the reader to [2] for a complete description and to [5] where STR was most recently applied for rainfall derivatives.

Even though the above approaches are popular, both face several drawbacks. First of all, MCRP is very simplistic and is heavily reliant on past information being reflective of the future. Additionally, the predicted amount is essentially the average level of rainfall observed across the study period and does not take into account annual deviations in weather patterns. Furthermore, for both approaches the model for each city needs to be specifically tuned as each exhibits different statistical properties, i.e. a new model for each city. Lastly, MCRP produces weak predictive models, as its only focus is on fitting the historical data. Similarly, STR although closer to the meteorological methods and is far more robust than MCRP, does suffer from long range prediction problems. This last point is very important, as one should not only be interested in deriving models that describe past data effectively, as it currently happens; instead, we should also be focusing on producing effective predictive models, which can offer us insights on future long range weather trends.

Due to the disadvantages highlighted above, we divert away from the use of statistical approaches and in this paper we extend our previous work [6] where we proposed using a machine learning technique called Genetic Programming (GP). Rainfall prediction on a daily basis has not been covered in great detail within the machine learning literature due to the complex nature of rainfall and the applications are mainly focussed on either the short term predictions (e.g. rainfall-runoff models up to a few hours [7] or monthly amounts [8] [9]). Little literature exists for the daily predictions, e.g. [10] used a feed-forward back-propagation neural network for rainfall prediction in Sri Lanka, which was inspired by the chain-dependent approach from statistics. [11] applied GP to daily rainfall data, but the GP performed poorly by itself, although when assisted by wavelets the predictive accuracy did improve. [6] applied GP for the first time to modelling the accumulated rainfall amounts using a sliding window within the context of rainfall derivatives. Results showed that GP statistically outperformed the most commonly used approach (MCRP) within rainfall derivatives literature across 21 cities around Europe.

In this paper, we look to predict rainfall amounts more accurately to assist with the pricing step, which as mentioned is the second part of the rainfall derivatives application. We choose to continue with GP for this paper over other machine learning techniques not only because it has outperformed the commonly used approach within the rainfall derivatives literature, but it has the benefit of producing white box (interpretable, as opposed to black box) models, which allows us to probe the models produced. Moreover, we can capture nonlinear patterns in data without any assumptions regarding the data. This should allow us to produce a model that can reflect the ever changing process of rainfall. As a result, we could capture yearly deviations that the current MCRP is unable to replicate and provide longer range predictions that STR is lacking. Additionally, we are able to produce a more general model, which can be applied to a range of cities/climates, without having to build a new model each time.

The extension introduced in this paper is on feature engineering. The idea exploited here is that our original data in its raw values, may have underlying properties that can further assist the prediction process. Although this approach is typically used on high dimensionality problems e.g. 100's or 1000's of variables to help with dimension reduction [12], we experiment to help discover and create more features based on the same variables used within [6]. Moreover, the variables used in [6] are very limited, hence is important to create new features.

Hence, the main contribution of this paper is exploring the use of feature extraction for the problem of rainfall prediction within rainfall derivatives. We will create a comprehensive set of extra features and use a variable selection technique to select only the most significant features, further tailoring a GP to the problem domain. The features themselves will provide additional information to assist with the problem at hand and aim to achieve better predictive accuracy. In order to show the effectiveness of the extra features we will follow and update the methodology used within [6].

The remainder of this paper is organised as follows. Section II will cover the setup of the data including the data sets that will be used. Section III describes in detail our GP for rainfall prediction. Section IV discusses the feature engineering process. Section V will then discuss the experimental setup, and Section VI will discuss the results from feature engineering. Finally, Section VII will conclude findings and suggest future research.

II. Data setup

There are two elements to the setup of the data, first is the number of cities we will test our GP on, including the length of each training set. Second, is how the data will be treated and the number of attributes that will be passed to the algorithms. We follow the same procedure outlined in [6] and have described the process below.

A. Choice of data

The daily rainfall data used is summarised in Table I, which includes a total of 21 cities from around Europe. The cities were chosen based on two aspects, firstly, the availability of data, hence minimising the potential for missing values. The data corresponding to the European cities were provided by the National Centers for Environmental Information[2] (NCEI). Secondly, the climate of each city. In order to get an approach that can be generalised, different climates are present across the selection of cities, ranging from very wet climates to very dry climates. This is an important factor as the climate has an impact upon an algorithm's performance, in the literature individual models are built for each city.

TABLE I
THE LIST OF ALL CITIES WHOSE DAILY RAINFALL AMOUNTS WILL BE USED FOR EXPERIMENTS.

Cities used for daily rainfall			
Amsterdam	(Netherlands)	Ljubljana	(Slovenia)
Arkona	(Germany)	Luxembourg	(Luxembourg)
Basel	(Switzerland)	Marseille	(France)
Bilbao	(Spain)	Oberstdorf	(Germany)
Bourges	(France)	Paris	(France)
Caceres	(Spain)	Perpignan	(France)
Castricum	(Netherlands)	Potsdam	(Germany)
De Kooy	(Netherlands)	Regensburg	(Germany)
Delft	(Netherlands)	Santiago	(Portugal)
Gorlitz	(Germany)	Strijen	(Netherlands)
Hamburg	(Germany)		

The length of data was chosen to be 10 years of daily rainfall for training and 1 year of daily rainfall for testing. We leave it as a future investigation whether different training lengths can impact the results. The length of training data is an important aspect, given climatic shifts can occur across long periods of time. Therefore, by using 10 years allows us to have sufficient observations to build a model on, without having to worry about climatic shifts within the period. Additionally, this will capture the periodic shifts in rainfall that occur each year, not associated with climatic shifts. As rainfall derivative contracts are written several months ahead of time and could span several months at a time, a testing period of 1 year is an appropriate length. Additionally, forecasting one year ahead really tests the robustness and suitability of the algorithm.

B. Treatment of data

The way the data is treated is an additional factor, as it is uncommon that giving raw data values to an algorithm will return anything of use. Therefore, the data should be transformed to better suit our problem domain. The end goal of this work is to price rainfall derivative contracts based on the accumulated amount of rainfall, over the specified contract length. For example, a contract for the month of January would require the summation of daily rainfall over 31 days. An important aspect, which should be taken into account is that contracts must be in the future, usually up to a year ahead of time and the contract period can be of any length. The most common period lengths being monthly or seasonally, but there is nothing stopping having a contract of 37 days or 164 days being specified. In addition, there is an even greater necessity for transforming the data, given the unique aspect of rainfall. Daily rainfall is one of the most volatile and hardest variables to predict, which includes (depending upon climate) long or frequent periods of wet and/or dry spells. Findings from [11] suggest that using daily values for GP is unsuitable given the relative poor performance of their GP. Figure 1 shows the annual rainfall for Luxembourg and just how volatile and unpredictable the rainfall process is over a year.

Therefore, we use a sliding window, which will transform

[2]http://www.ncdc.noaa.gov/

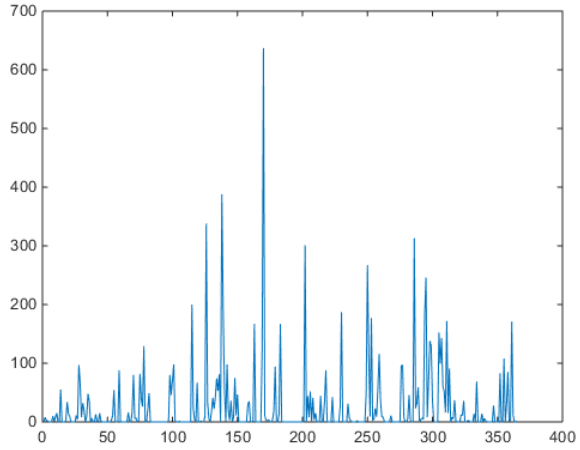

Fig. 1. The daily level of rainfall in tenths of mm of Luxembourg over the period from 01/01/2013 till 31/12/2013.

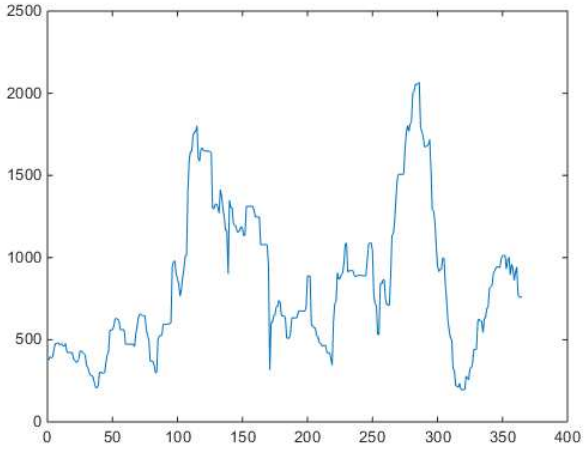

Fig. 2. The daily level of rainfall in tenths of mm of Luxembourg using the sliding window approach over the period from 01/01/2013 till 31/12/2013.

the data to something more manageable and better suited for the problem domain. Figure 2 shows the benefit of applying a sliding window approach to the data. The output appears a lot less random, which was the motivation behind applying the sliding window, i.e., to help smooth out the data. Additionally the day-by-day volatility appears to have decreased and a pattern in rainfall is more easily noticeable. This approach is very flexible to the problem of predicting rainfall and can be modified to any length of interest. We refer the reader to [6] for a more detailed explanation of the process.

C. Data variables

In order to predict the accumulated rainfall amount, historical data from previous periods in the same form is required. If we predict for a 31 day sliding window, then our data variables should be consistent with this. Therefore, constructing the variables in the same way from [6], we generate a set of variables r_t and r_y. 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. We use this latter kind of variable to capture information regarding annual rainfall variations. The variables are shown in Figure 3. For example if our target day was 1^{st} January 2016, then r_{t-1} is 1^{st} December 2015 - 31^{st} December 2015, r_{t-2} is 31^{st} October 2015 - 30^{th} November 2015 and r_{y-1} is 1^{st} January 2015 - 31^{st} January 2016.

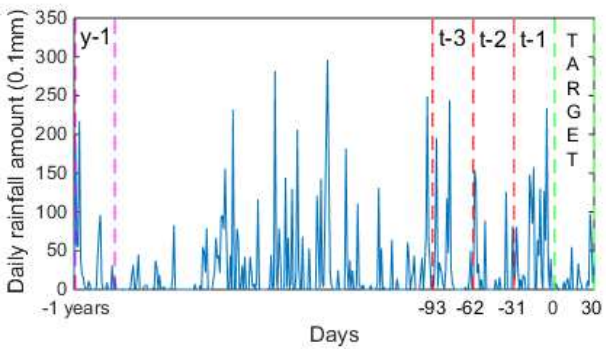

Fig. 3. The sliding window value with the targets day amount with its respective t's and y's. The daily rainfall amounts within each boundary would be accumulated.

To sum up what we have discussed in this section, the data sets that we will use consist of 21 different European cities, from different climate types. In addition, we will use a sliding window approach to summarise the data, instead of daily predictions. Lastly, the attributes we will be using for predicting the rainfall amounts are the previous contract length periods r_t and r_y.

III. Our Genetic Programming methodology

Here we briefly outline the GP used in [6] for the problem of rainfall prediction. To avoid illegal trees being generated we use a Strongly-typed GP (STGP) [13] allowing us to specify different types. Several modifications have been made to the STGP, which will be covered briefly here.

A. Terminal set

There are three types of elements to the terminal set. The first set of elements in the terminal set includes all the variables available within the data. The variables are defined by the original r_y's and r_t's calculated from the original data. The second element is an ephemeral random constant (ERC), which will pick a uniformly distributed random number. We allow our ERC to choose a random number between the limits of -500 to 500. We want to generate a larger spread, due to predicting accumulated rainfall over a contract length, rather than daily amounts. Additionally, we allow for flexibility in our ERC and include a separate range for positive and negative numbers. Therefore, allowing a way to reduce the search space for choosing meaningful random numbers. The ERC requires four parameters to control the range of random numbers. Two parameters to control the positive range and two to control the negative range. Each different range requires a parameter for its upper bound and a parameter for its lower bound.

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 a 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. We opt for choosing from within this range, to avoid excessively large numbers being created, whilst maintaining a reasonable amount of options for our GP to choose from during initialisation and evolution.

TABLE II
GP function and terminal sets.

Set	Value
Functions	ADD, SUB, MUL, DIV, POW, SQRT, LOG
Terminals	11 r_t periods (t-1, t-2, . . . , t-11), 10 r_y periods (y-1,y-2,. . . ,y-10), ERC, Constants in the range [-4,4]

B. 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, because the data includes zeroes and negative numbers. If the input is zero or negative then SQRT and LOG will return zero. If the second argument passed to DIV is zero (denominator), then zero is also returned. Protecting these values will stop NaN's (not a number) and Inf's (infinity) from being generated. The final function that has been modified is POW. It has been forced such that the second argument will be a constant within a specified range as mentioned within the previous discussion regarding the terminals stopping very large values from being generated. Additionally, we allow

TABLE III
A TABLE SHOWING THE COMPLETE LIST OF FEATURES TO BE CALCULATED

Features	
Mean	Standard Deviation
Sum	
Difference	Adjacent Difference
Moving Average	Standard Deviation across MA
Last Maximum	Last Minimum
Time Last Maximum	Time Last Minimum
Magnitude of Maximum	Magnitude of Minimum
Maximum Last Contract Length	Minimum Last Contract Length
Difference of Maximum over Last Contract Length	
Difference of Minimum over Last Contract Length	

for fractional powers, which means there is the potential for rooting negative values and producing NaN. One final check is whether the first argument (number to be raised by a power) is negative, if so then the second argument must be a whole number, which will be rounded to the nearest number if fractional. These adjustments will avoid illegal trees being generated.

All functions and terminals presented in this section are summarised in Table II.

C. Management of trees

Another adjustment made involves dealing with negative number outputs. For this problem domain the values have to be greater than or equal to zero, it is impossible to have negative rainfall amounts. Therefore, we include a wrapper around each individual (candidate solution) to change the prediction to zero if the prediction was less than zero. The final adjustment made was to ensure a good balance between variables and random numbers in an individual. Therefore, when initialising the population using the ramped-half-and-half method, we make sure that the first child is either a function or a variable, whereas the second child can either be a variable, an ERC or another function. This will avoid trees being dominated by random numbers.

D. Fitness (evaluation) function

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

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

where N is the length of the data 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).

IV. ADDITIONAL FEATURES

Here we outline the features that are to be created to extract additional information from the variables outlined in II-C with the given number of r_t's and r_y's as specified from II. The motivation is that the original variables previously used may contain relationships between themselves that provide additional information that can be extracted. Additionally, we can create features that GP does not need to construct by itself or does not have the means to easily do so. Therefore, we are saving precious computational time during the evolution process by giving GP the necessary tools ahead of time, thus we do not need to rely on the features being generated by chance or even not at all. One issue that is raised by such an approach is that we will be increasing the dimensionality of the problem, but we are confident that the extra information is worthy of the increased dimensionality by creating meaningful features in the first place. Given that we are increasing the dimensionality we will also select the best features from those generated to avoid issues caused by high dimensionality.

A. Creation of features

Table III outlines the full list of features that we are to create for the problem at hand. We opt to create them ahead of time, rather than having GP randomly create them during the evolution process as demonstrated by [12]. If left for GP to randomly create the features, we will not know which features were considered other than the best ones in the final individual (if any). Therefore, by creating them ahead of time, we can analyse and compare across multiple data sets which features were actually used from the initial set. Likewise, as previously mentioned we get more choice in deciding what would make a good feature by our intuition of the problem domain. Subsequently, from the list of features created we may want to enforce a structure within the trees on how to handle or which features are allowed to be combined. Thus, avoiding wasted evolution time during the experiments. Additionally, we calculate the features by respecting the natural temporal order of the data, rather than randomly choosing variables ignoring the temporal order.

The Mean, Standard Deviation and Sum are all calculated over various lengths of periods for both r_t and r_y, e.g. Mean_{t-1} would be the mean of the corresponding rainfall amounts at times r_{t-1} and r_{t-2}, Mean_{t-2} would be the mean of the corresponding rainfall amounts at times r_{t-1}, r_{t-2} and r_{t-3}.

The Difference is very similar, however, the relevant period is always taken away from either r_{t-1} or r_{y-1} e.g., Diff_{y-2} would be the difference between rainfall amounts r_{y-1} and r_{y-2}, whereas Diff_{y-3} would be the difference between rainfall amounts r_{y-1} and r_{y-3}. Adjacent difference (AdjDiff) works in a similar way, but is between adjacent pairs e.g., AdjDiff_{t-2} would be the difference between rainfall amounts r_{t-2} and r_{t-3}.

Moving Average (MA) is calculated on each individual r_t and r_y in turn and will go back a predefined number of days rather than across parameters like the Mean previously described. Likewise, Standard Deviation across the Moving Average (StdMA) works in the same way, but calculates the standard deviation instead of the average. We allow for different lengths of moving averages as we make no assumption

what value is the most appropriate. We vary the MA length from the last 10-100 days in increments of 10 days.

Last Maximum (LMax) and Last Minimum (LMin) is the last known local maximum and local minimum point of our data, as we move through our data. The Magnitude of Maximum and Magnitude of Minimum is how far LMax and LMin is from the mean value of our data. Maximum of last Contract Length (MaxCL) and Minimum of Last Contract Length (MinCL) are similar, but are over the predefined contract length, instead of the last known local maxima or minima. Thus, LMax and MaxCL can be different, similarly LMin and MinCL can be different. Difference of Maximum over Last Contract Length and Difference of Minimum over Last Contract Length is the deviance away from the average LMax and LMin values respectively.

The features presented have been chosen to assist GP and are common for statistical analysis or for time-series analysis. Additionally, our GP must operate within a set of constraints whether it is the depth of the tree, the structure of the tree or the terminal/function set (Table II) available. Therefore, we are able to create features that GP would not be able to randomly create if left during the evolution process. Additionally, we have created features that are bounded by temporal constraints, thus reducing the complexity and overhead of checking for feasible feature creation.

B. Selection of features

From the extra features in Table III using the last 11 r_t's and 10 r_y's given in Table II, we are able to create a total of 514 extra features. From the features created, not all of them may be useful and in fact some may not contribute much (or any) in terms of extra information. Therefore, we will use a well known variable selection technique called Correlation based Feature Selection (CFS) [14]. This technique was designed to select a feature subset where each feature has a high predictive ability and the degree of redundancy between features is low. Thus, we only choose variables that have a positive influence to predict.

TABLE IV
A TABLE SHOWING THE TOTAL NUMBER OF FEATURES PER CITY AS SELECTED BY CFS

City	Number of features	City	Number of features
Amsterdam	59	Ljubljana	46
Arkona	56	Luxembourg	40
Basel	48	Marseille	61
Bilbao	57	Oberstdorf	50
Bourges	51	Paris	47
Caceres	47	Perpignan	29
Castricum	55	Potsdam	59
Dekooy	60	Regensburg	52
Delft	43	Santiago	58
Gorlitz	59	Strijen	43
Hamburg	37		

From the 514 features in total we use CFS to select the best features for each data set based on the training data. Table IV shows the full list of cities with their respective number of features after feature selection. As we can observe we have been able to successfully reduce the number of features from 514 to a range between 29 to 61 depending upon the city that will be used for our experimentation.

V. EXPERIMENTAL SETUP

A. Parameter tuning - GP

We used a package called iRace [15] to find the optimal parameters for GP based on the training data, presented in Table V.

TABLE V
THE BEST CONFIGURATION OF GP FROM OPTIMISING THE PARAMETERS USING IRACE.

GP parameters			
Max depth of tree	8	Elitism percentage	0.03
Population size	1400	Number of gens	30
Crossover probability	0.76	ERC negative low	-495.36
Mutation probability	0.69	ERC negative high	-102.56
Primitive probability	0.55	ERC positive low	100.77
Terminal/Node bias	0.2	ERC positive high	438.58

B. Experimental methodology

Using the optimal GP parameters from Table V, we are then ready to move on to the experimental comparison between our GP's, which are tested on all 21 datasets. GP will use the full and most recent training set (01/Jan/2004 - 31/Dec/2013), before testing on the unseen test set (01/Jan/2014 - 31/Dec/2014). As GP is a stochastic algorithm, we run it for 50 times on each city and report the average over those 50 runs.

We will be running three different experiments for GP, the first (GPOF) will be GP using the previous 11 r_t periods and previous 10 r_y periods noted in Table II. The second experiment (GPEF) will be GP using all of the extra features after feature selection given in table IV along with the original 21 features used in the first experiment. Due to an uneven distribution of extra features per data set, we will pick the 21 best ranked features from those generated to match the number from the initial experiment. Hence, the third experiment (GPEF21) will be to use the best ranking 21 features from each data set along with the original 21 features used in the first experiment. For completeness we will include the performance of MCRP, which is the most common method used within rainfall derivatives. We do not include STR mentioned earlier, due to not coping well with long run predictions.

VI. RESULTS

The performance of GPOP, GPEF and GPEF21 is presented in Table VI based on the average RMSE performance from the testing set for each city. For completeness we have also included the results from MCRP. The table has been split

between those data sets seen by iRace (top) and unseen (bottom), arranged by alphabetical order. We have chosen to do this, as the best GP configurations were chosen based on the validation set of the 11 cities shown in the top half. Hence, we would expect GP to perform better. Whereas, the bottom 10 cities by have not influenced the choice of best parameter configuration for GP and help show the ability to generalise. Thus, allowing us to use our best configuration on future data sets that exhibit a similar climate.

TABLE VI
THE AVERAGE RMSE PERFORMANCE IN TENTHS OF MM FOR EACH OF OUR EXPERIMENTS ACROSS EACH CITY.

Data	GPOF [6]	GPEF	GPEF21	MCRP
Amsterdam	412.49	503.93	500.53	**373.42**
Arkona	310.45	**275.64**	278.42	290.72
Bilbao	519.67	437.58	**436.31**	659.03
Bourges	375.46	**335.29**	338.26	382.62
De kooy	343.66	336.08	**329.35**	358.27
Ljubljana	1027.62	**897.10**	917.51	1058.86
Luxembourg	517.34	**477.67**	482.29	585.23
Marseille	505.93	**492.14**	502.01	956.35
Potsdam	320.66	329.83	**317.28**	327.98
Regensburg	330.33	**300.43**	327.47	331.07
Santiago	1062.60	**1043.97**	1067.32	1085.10
Basel	425.13	**403.58**	413.39	467.23
Caceres	438.41	436.25	**434.46**	687.74
Castricum	438.88	409.25	**406.94**	465.77
Delft	**404.47**	447.56	444.09	449.82
Gorlitz	304.38	**282.29**	296.24	304.92
Hamburg	408.25	**374.75**	379.85	409.56
Oberstdorf	677.36	647.27	**647.06**	671.99
Paris	277.79	**271.73**	273.25	280.40
Perpignan	760.73	**760.08**	760.66	968.74
Strijen	298.62	**287.53**	310.30	343.21

GPEF achieved the lowest RMSE on 13 data sets, whilst GPEF21 only 6 times and MCRP and GPOF sharing 1 each. This is a very good result, showing the positives that can be achieved with the use of extra features. Other than Amsterdam, GP once again outperformed MCRP over the testing period for all other cities. Generally speaking, the gains in predictive accuracy from GPEF and GPEF21 over GPOF and is an important step forward for the second part of pricing. From the results the percentage gains in lower RMSE from the use of extra features ranges from 0.1% for Perpignan to 16% for Bilbao. Other noticeable gains include Ljubljana (13%), Bourges (11%) and Arkona(11%), with the overall average increase in performance just over 6%.

To check which of our GP's performed better in terms of victories, we compute the mean rank based on Table VI — the lower the rank, the better the GP's performance. Furthermore, in order to determine whether the above results are statistically significant, we compare the four approaches by using the Friedman test [16]. The Friedman test is a nonparametric test for testing the difference in mean between multiple related samples. The null hypothesis is that there is no significant difference between the average of the four approaches. We apply the test at the 5% significance level.

TABLE VII
THE MEAN RANKINGS OF THE FOUR EXPERIMENTS, AND THE FRIEDMAN'S p VALUE TO TEST WHETHER ONE OR MORE OF THE APPROACHES STATISTICALLY OUTPERFORMED THE OTHERS.

Approach	Ranking
GPOF	2.81
GPEF	**1.62**
GPEF21	1.86
MCRP	3.71
Friedman p-value	$1.36\text{x}10^{-7}$

Table VII shows the mean rank of the four approaches, GP using original features, GP using all extra features, GP using the best ranking 21 features and MCRP, with values of 2.81, 1.62 and 1.86, 3.71 respectively, where a lower rank indicates better performance. Therefore, across all cities on average GP with all extra features outperformed the three other approaches. As we can observe, the Friedman test result has a p value of $1.36\text{x}10^{-7}$, which is less than the 5% significance level. Therefore, there is strong evidence to reject the null hypothesis, and conclude that there is a statistical difference between the four approaches.

Due to having a statistical difference, we will perform the Holm post-hoc test in order to determine which of the four approaches statistically outperformed the other. The results can be found in Table VIII.

TABLE VIII
A TABLE SHOWING THE PAIRED COMPARISONS OF INTEREST FROM THE HOLM POST-HOC TEST FOR OUR FOUR EXPERIMENTS TO DETERMINE WHICH APPROACH OUTPERFORMED ONE ANOTHER

Paired comparisons	p value	Holm
GPEF vs. MCRP	$\mathbf{1.4484\text{x}10^{-7}}$	0.0083
MCRP vs. GPEF21	$\mathbf{3.1408\text{x}10^{-6}}$	0.0100
GPOF vs. GPEF	**0.0028**	0.0125
GPOF vs. GPEF21	0.0168	0.0167
GPOF vs. MCRP	**0.0232**	0.0250
GPEF vs. GPEF21	0.5501	0.0500

The table shows that all of the approaches using GP statistically outperformed MCRP, shown by p values $1.4484\text{x}10^{-7}$, $3.1408\text{x}10^{-6}$ and 0.0232 which are less than the Holm scores of 0.0083, 0.0100 and 0.0250 respectively. This is an important result as it shows that GP once again is able to outperform MCRP, the most common approach that exists within the literature of rainfall derivatives.

GPEF does not statistically outperform GPOF at the 95% confidence level, with a p value of 0.0168 which is marginally greater than the Holm score of 0.0167. However, we obtain

statistical significance at the 90% confidence level where the Holm score would be 0.0333. Despite this, it still shows a positive result as we are statistically outperforming MCRP at the 95% confidence level using GPEF21.

The main result is that GPEF does statistically outperform GPOF, with a p value of 0.0028 which is less than the Holm score of 0.0125. Therefore, these results show that the use of carefully designed extra features is beneficial and does help increase the predictive power of GP.

From the above results, we can conclude that the use of extra features is a beneficial in assisting GP for predicting rainfall in the context of weather derivatives. Shown by outperforming GPOF and also the most common approach in the rainfall derivatives literature (MCRP). This is a very important result, as it shows that there are more potential gains to be made for predicting rainfall and that we can further tune our GP to the problem at hand. Especially as by producing more accurate rainfall predictions, helps to increase the accuracy of pricing rainfall derivatives. As we explained at the beginning of this paper, is another important problem. Lastly, as we are able to give more confidence surrounding the prediction of rainfall, this will help to reduce potential mispricing and attract more investors to the rainfall derivative market.

VII. Conclusion

This paper extends the work by [6] by further looking to improve the predictive accuracy of rainfall within the application of rainfall derivatives. The extension proposed in this paper is the use of carefully designed extra features and to see whether they have a positive effect on Genetic Programming's (GP) ability to predict accumulated rainfall amounts. The motivation for this paper comes from questioning whether the features used within [6] were the most appropriate, which was left for future research. The idea is that the standard features used may contain additional information that might not be utilised if left to GP by itself to construct. Thus, we aimed to construct a set of new features that could extract more information to boost the predictive performance of GP.

Strongly-typed Genetic Programming (STGP) was our chosen methodology, due to producing white box (interpretive) models and to being a technique that can detect and learn from nonlinear data. Furthermore, STGP was chosen over the standard GP, because we can influence types to avoid illegal trees being created. In this paper we compared our STGP across three different experiments, the first was using the original features used in [6], the second was using a set of extra features and the third was to use a smaller subset of extra features. The first experiment and Markov-chain extended with rainfall prediction (MCRP) acted as our benchmark for our later experiments.

Using daily rainfall data across a collection of cities from Europe, we predict accumulated rainfall based on using a sliding window. This approach is more intuitive to the problem at hand. When we compared the performance of our three GP approaches, we found sufficient evidence to suggest that the use of the proposed extra features is beneficial and that GP using extra features statistically outperforms a GP using the original features and the most currently used approach (MCRP).

Future work will include testing other commonly used regression algorithms to compare against GP. Furthermore, we will develop a method to decompose the rainfall prediction problem down for GP to further improve the accuracy. Lastly, since we have obtained further promising rainfall prediction results, we can also move towards the pricing task of rainfall derivatives and investigate if our current results have an overall positive effect in pricing.

References

[1] D. S. Wilks, "Multisite generalization of a daily stochastic precipitation generation model," *Journal of Hydrology*, vol. 210, pp. 178–191, Sep. 1998.

[2] I. Rodriguez-Iturbe, D. R. Cox, and V. Isham, "Some models for rainfall based on stochastic point processes," *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, vol. 410, no. 1839, pp. 269–288, 1987.

[3] R. W. Katz, "Precipitation as a chain-dependent process," *Journal of Applied Meteorology and Climatology*, vol. 16, no. 7, pp. 671–676, 1977.

[4] B. L. Cabrera, M. Odening, and M. Ritter, "Pricing Rainfall Derivatives at the CME," *SFB 649 Discussion Papers*, Jan. 2013.

[5] R. C. Noven, A. E. D. Veraart, and A. Gandy, "A lévy-driven rainfall model with applications to futures pricing," *AStA Advances in Statistical Analysis*, vol. 99, no. 4, pp. 403–432, 2015.

[6] S. Cramer, M. Kampouridis, A. A. Freitas, and 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*, Dec 2015, pp. 711–718.

[7] N. Q. Hung, M. S. Babel, S. Weesakul, and N. K. Tripathi, "An artificial neural network model for rainfall forecasting in bangkok, thailand," *Hydrology and Earth System Sciences*, vol. 13, no. 8, pp. 1413–1425, 2009.

[8] J. Wu, J. Long, and M. Liu, "Evolving {RBF} neural networks for rainfall prediction using hybrid particle swarm optimization and genetic algorithm," *Neurocomputing*, vol. 148, pp. 136 – 142, 2015.

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

[10] H. Weerasinghe, H. Premaratne, and D. Sonnadara, "Performance of neural networks in forecasting daily precipitation using multiple sources," *Journal of the National Science Foundation of Sri Lanka*, vol. 38, no. 3, 2010.

[11] O. Kisi and J. Shiri, "Precipitation forecasting using wavelet-genetic programming and wavelet-neuro-fuzzy conjunction models," *Water Resources Management*, vol. 25, no. 13, pp. 3135–3152, 2011.

[12] A. Kattan, M. Kampouridis, Y.-S. Ong, and K. Mehamdi, "Transformation of input space using statistical moments: Ea-based approach," in *Evolutionary Computation (CEC), 2014 IEEE Congress on*, July 2014, pp. 2499–2506.

[13] D. J. Montana, "Strongly typed genetic programming," *Evolutionary Computation*, vol. 3, no. 2, pp. 199–230, 1995.

[14] M. A. Hall, "Correlation-based feature subset selection for machine learning," Ph.D. dissertation, University of Waikato, Hamilton, New Zealand, 1998.

[15] M. López-Ibáñez, J. Dubois-Lacoste, T. Stützle, and M. Birattari, "The Rpackageirace package, iterated race for automatic algorithm configuration," IRIDIA, Université Libre de Bruxelles, Belgium, Tech. Rep., 2011.

[16] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *J. Mach. Learn. Res.*, vol. 7, pp. 1–30, Dec. 2006.