de

MDSAA

Master Degree Program in

Data Science and Advanced Analytics

A study on variations of Genetic Programming applied to time series forecasting

Machine Learning for *Energy Consumption Forecasting*

Davide Farinati

Dissertation

presented as partial requirement for obtaining the Master Degree Program in Data Science and Advanced Analytics

NOVA Information Management School
Instituto Superior de Estatística e Gestão de Informação
Universidade Nova de Lisboa

NOVA Information Management School
Instituto Superior de Estatística e Gestão de Informação
Universidade Nova de Lisboa

A STUDY ON VARIATIONS OF GENETIC PROGRAMMING APPLIED TO TIME SERIES FORECASTING

by

Davide Farinati

Dissertation presented as partial requirement for obtaining the Master's degree in Advanced Analytics, with a Specialization in Data Science

Supervisor : Leonardo Vanneschi

June 2022

STATEMENT OF INTEGRITY

I hereby declare having conducted this academic work with integrity. I confirm that I have not used plagiarism or any form of undue use of information or falsification of results along the process leading to its elaboration. I further declare that I have fully acknowledge the Rules of Conduct and Code of Honor from the NOVA Information Management School.

Milano, 01/06/2022

Se vuoi puoi, se puoi devi.

ABSTRACT

Evolutionary Computation is a sub-field of Machine Learning algorithms based on *Darwin's theory of Evolution*. Individuals are evolved using the principles of mutation, crossover and natural selection. One of the most known Evolutionary Algorithms is Genetic Programming (GP), that evolves as individuals computer programs in order to solve regression problems. In this thesis two variations of GP, namely Geometric Semantic Genetic Programming(GSGP) and Tree-based Pipeline Optimization Tool(TPOT), are applied to two energy consumption time series regression problems. Their performance are then compared to state-of-the-art models, LSTM and SVR optimized with DE, and to standard GP. It is showed that the variations of GP outperform standard GP and SVR optimized with DE, while also having comparable performance to LSTM. Additionally a study on the feature selection ability of GSGP is proposed, showing that the algorithm is not actually able to perform feature selection.

KEYWORDS

Machine Learning; Genetic Programming; Geometric Semantic Genetic Programming; Time Series

INDEX

Contents

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

Energy is a fundamental part of our everyday life, and forecasting its consumption is crucial. Energy consumption data is often presented as time series, that are sequences of equally distant and ordered data points. Time series forecasting has been studied thoroughly, since it is fundamental in many aspects of our world, and it gives us the opportunity to “look into the future". This thesis explores the performance of new methods based on Genetic Programming(GP), namely Geometric Semantic Genetic Programming(GSGP) and Tree-based Pipeline Optimization Tool(TPOT) applied to time series forecasting problem. Two datasets were used for this experiment, one is a energy consumption time series provided by America Electric Power (AEP), and made of 2500 hourly observations from 2004 to 2005, the other is the China Energy Consumption(CEC) dataset, that is made up of China’s yearly energy consumption from 1965 to 2017. The performance of the two afore mentioned algorithms will be compared to state-of-the-art models, LSTM and SVR optimized with DE, and to standard GP. In order to prove statistical significance the Wilcoxon rank-sum test for pairwise data comparison will be used. To maximize the performance of all the algorithms, a greed search to find the best combination of hyperparameters will be performed. For this experiment, it is expected for the variations of GP to outperform the standard version of the algorithm, and to have comparable performance with the state-of-the-art algorithms. Additionally a study on the feature selection ability of GSGP will be performed. The performance of the algorithm on the AEP dataset will be compared with the performance on modified versions of the dataset, where either random features are added, or less important features are modified.

2. PREVIOUS WORK

In the past many approaches have been experimented in order to find a state-of-the-art forecasting model for energy consumption time series. Many studies on energy consumption forecasting have been done, such as in the Papers [1] and [2], where all the recently developed models for solving energy performance in building are reviewed. In the Paper [3] different time series analysis methods, such as Holt-Winters (HW), centered moving average (CMA) and others, are tested in order to find the best performing one, and it is shown that HW gives the smallest mean absolute error (MAE) and mean absolute percentage error (MAPE), while CMA produces the lowest mean square error (MSE) and root mean square error (RMSE). Being artificial intelligence on the rise, many authors proposed approaches that exploit its upsides applied to energy consumption forecasting. For example in the Paper [4] where the results of forecasting of the gas demand obtained with the use of artificial neural networks are presented. The training data for this experiment was taken from the actual natural gas consumption in Szczecin (Poland). In the model, calendar (month, day of month, day of week, hour) and weather (temperature) factors were considered. And it was obtained a multilayer perceptron (MLP) model capable of successfully predicting gas consumption. A similar approach was presented in Book [5], where different Deep Learning and Artificial Neural Network algorithms, such as Deep Belief Networks, AutoEncoder, and LSTM, were introduced to the field of renewable energy power forecasting. And, compared to a standard MLP and a physical forecasting model, showed superior forecasting performance. Then in Paper [6] energy consumption time series are transformed in a regression data-set and neural networks, in particular long-short term memory (LSTM), are used to forecast future values. It is proved that regression from time series is more efficient and easier to use than variable regression, that was based on data such as GDP, population, the primary sector of the economy, the secondary sector of the economy, and tertiary sector of the economy. In the Book [7] long short term memory (LSTM), a deep learning technique, is compared with other learning techniques, such as the back propagation algorithm and the more recently proposed online sequential learning algorithm, in the context of time-series prediction. It is demonstrated that the online sequential learning algorithm is more reliable and provides faster convergence resulting in better prediction performance. In Paper [8] a hybrid model for building energy consumption forecasting is proposed, the parameters of weighted support vector regression(SVR) models with nu-SVR and epsilon-SVR are optimized with a differential evolution (DE). A detailed comparison with other evolutionary algorithms show that the proposed model yields higher accuracy for forecasting. This thesis focuses on studying the performance of evolutionary algorithms in regard of forecasting time series, but previous work already exists, for example [9] and [10]. In the former, genetic and evolutionary algorithms (GEAs) with arithmetic crossover and Gaussian perturbation are used to optimize the parameters of linear combination and ARMA models while forecasting different time series. The handicap of the evolutionary approach is then compared with conventional forecasting methods, being competitive. While in the latter a variation of Genetic Programming (GP), Geometric Semantic Genetic Programming (GSGP) with a deterministic-crossover operator (D-GSGP) is adopted. The experimental results indicate that D-GSGP works effectively and the acquired programs are useful for knowledge acquisition of the application domain. Another study on evolutionary algorithms applied to time-series is showed in Paper [11], where Postfix-GP, a postfix notation based GP, is used. The Postfix-GP uses linear genome representation and stack based evaluation to reduce space-time complexity of GP. Its performance

indicate that the discussed model offers a new possibility for solving time series modeling and prediction problems. The focus of this thesis is to explore the performance of different evolutionary algorithms on energy consumption time series, and compare them to the ones of other machine learning methods, some of which are mentioned above.

3. THEORETICAL BACKGROUND

This chapter is meant to provide a better and more complete understanding of what discussed further. It starts with an introduction of general machine learning concepts in Section 1.1. Then in Section 1.2 it proceeds with a more in depth description of the models used later in the thesis, and it concludes in Section 1.3 with a discussion of evolutionary algorithms, with a focus on Geometric Semantic Genetic Programming and Tree-based Pipeline Optimization Tool in Sections 1.3.2.1 and 1.3.2.2.

3.1. MACHINE LEARNING

Machine Learning is the science that studies how computers can learn through data and without being provided specific instructions. Machine Learning is commonly divided into two sub fields: supervised learning and unsupervised learning. In this paper only examples of supervised learning are shown. The process of Machine Learning is made up of different steps:

- Data Cleaning; the practice of transforming raw data taken from various sources into useful input for modeling.
- Feature Selection, Preprocessing and Construction; applying changes to the existing features in order to extrapolate information. It can include creating new features or dropping existing ones.
- Model Selection; applying different Machine Learning models and comparing their performance and generalization capability.
- Parameter Optimization; tuning the parameters of the chosen model in order to maximize its performance on the data.
- Model Validation; evaluating the chosen model with the fitted parameters on unseen data and evaluating its performance.
- All the steps mentioned above can be automated in Pipelines, that are end-to-end constructs that orchestrates the flow of data into an output produced by one or more models.

3.1.1. Supervised Learning

The main characteristic of supervised learning is that the data (called training data) used by computers for their learning process is composed of features and labels (also known as targets), and the goal is to find a function that maps the former into the latter ones. Depending on whether the targets are continuous or discrete values, we talk about regression problems (for the first ones) or classification problems (for the second ones) . In supervised learning we are looking for the function that generalizes the best, that meaning the function that performs the best not only on training data but also on new data (called test data or unseen data).

3.2. Used Machine Learning Models

3.2.1. Neural Networks

Neural Networks(NN) are a technology that mimics the function of the human brain, made of neurons (also called units) connected by set of weights. Neurons are grouped in layers, all Neural Networks must have one input layer, one output layer and at least one hidden layer. The most common type of NN are feed forward NN trained with back propagation, but many different types exists. In feed forward NN the input layer is fed with data (training data) that is then passed to the hidden layers, where each neuron is activated by a activation function that determines whether the unit modifies the data provided to it, then an understandable output is produced via the output layer (continuous or discrete values depending on the kind of problem we are working on). The output produced by the Neural Network is then compared with the label of the training data and a loss is calculated according to a predetermined function (called loss function), and thanks to the back propagation of error the weights of the Neural Network are adjusted in order to produce an output closer to the label of the training data. This process is done for all the data points present in the training data set and the iteration of passing all the data points through the Neural Network is called epoch. The number of epochs is set by the user before starting to train the Neural Network.

Examples of possible activation functions of the neurons are:

- Hyperbolic Tangent function $f(x) = \frac{(e^{x} - e^{-x})}{(e^{x} + e^{-x})}$
- Sigmoid / Logistic function $f(x) = \frac{(-1)}{(1 + \mathrm{e}^{-x})}$
- ReLU (Rectified Linear Unit) function $f(x) = \max(0, x)$

Where x is the input of the singular neuron, that corresponds to the sum of all the values of the neurons present in the layer before, multiplied by the weights connecting them to the current neuron, except for the input layer, where the input is the raw data. The formula that regulates how the weights are updated is the following:

$$w_{ij+1} = w_{ij} + \alpha(target - loss)x_j$$

Where w_{ij+1} is the updated value of the weight connecting neuron i to neuron j, w_{ij} is the current value of the weight connecting neuron i to neuron j , α is the learning rate and x_j is the value of the neuron j. This formula is also known as the back propagation of error. The learning rate is an hyper parameter whose value ranges between 0 and 1, set by the user before starting to train the Neural Network. It regulates how much the NN should adjust to the error. A high learning rate provides a faster convergence(a consistent error through generations) but it can lead to overfitting, while a lower

one would provide a slower convergence, but also a more generalized model . Common practice is to set a learning rate decay, meaning that the learning rate decreases after each epoch.

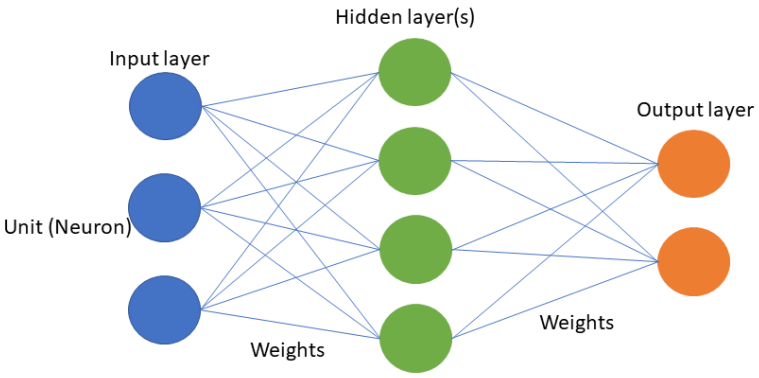


Figure 3.1 Structure of a Neural Network.

3.2.1.1. Recurrent Neural Networks

Standard Neural Networks don't have memory of the data points that they have already seen, while recurrent Recurrent Neural Networks (RNN) are able to remember thanks to an internal loop. Thanks to their ability to remember RNN are mainly used for data where the order matters, such as time series and text. Information about previous data is stored in an hidden state that is fed to the network together with the subsequent input.

3.2.1.2. Long-Short Term Memory

Long-Short Term Memory (LSTM) are a special type of Neural Networks, mainly used for predictions on ordered data(such as time series). They are a variation of RNN. In LSTM the output is calculated taking in consideration both the input and the current state, that is an output of the previous state. This type of Neural Network has the capability to forget certain parts of the current state and to add new information to it.[6]

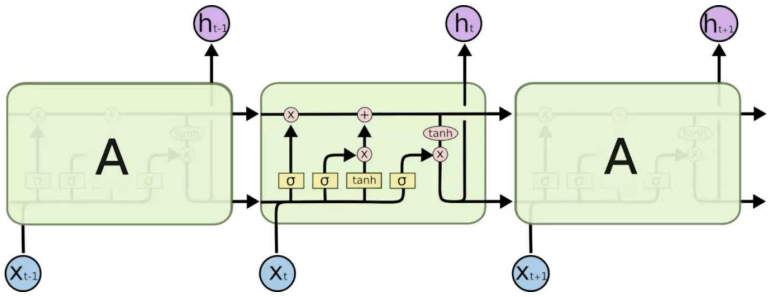


Figure 3.2 Structure of LSTM.

3.2.2. Support Vector Machines

Support Vector Machines (SVM) are a machine learning model based on two components, a kernel and an optimizer algorithm. The first component is used to map non-linear data into a high-dimensional space, where the transformed data is then linearly separable. SVM are normally used for classification, and the main idea behind the optimizer algorithm is to find a line or hyper plane (depending on the number of dimensions) that separates the classes, and that it maximizes the margin between the margin points (also known as support vectors). When used for regression SVM, in this case called Support Vector Regression(SVR), uses the support vectors to define a linear regression. SVR instead of looking for a boundary is looking for a curve that maps vectors into targets.

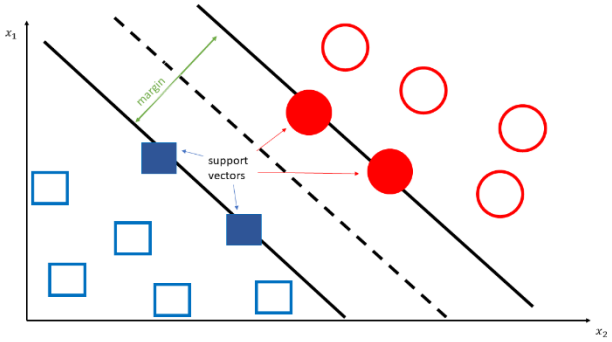


Figure 3.3 Structure of SVM.

Two version of the SVR algorithm exist, the first one, called eps-SVR, proposed in the Book [12], and it uses the parameter ϵ that specifies the epsilon-tube within which no penalty is associated in the training loss function with points predicted within a distance epsilon from the actual value. While nuSVR, proposed in the Book [13], substitutes the parameter ϵ with ν, that controls the number of support vectors and training errors. Common to both the algorithms are the parameters C and γ, the first one is inversely proportional to the model's regularization strength and it specifies the trade-off between the empirical risk and the model smoothness. While the second one controls the Gaussian function width of the kernel function. [8]

3.3. Evolutionary Computation

Evolutionary Computation is a family of algorithms inspired by biological evolution. They are based on C. Darwin's Theory of Evolution presented in [14]. Darwin's theory can be summarized in 5 steps, show in Figure 1.4:

1. Reproduction
2. Ability of adaptation to environment
3. Inheritage
4. Variations
5. Competition

The evolutionary process of EC starts with an initial population P (usually randomly created) of size N, then via a selection process N individuals (N ≤ N) of P are selected and a new population P' is created

(called population of parents). From the individuals in P', thanks to the genetic operators, a new population P" of size N (same as P) is created, and then process is repeated all over until a certain stopping criteria is met. When the iterations are over the best individual of the final population is considered the winner, the best solution to the optimization problem. The genetic operators are crossover and mutation, the first one is the random combination of two individuals(called parents) and corresponds in nature to sexual reproduction, while the second is a random change of the single individual, and it corresponds to asexual reproduction.

3.3.1. Differential Evolution

Differential Evolution (DE) is a sub-field of Evolutionary Computation where the optimization problem is solved by trying to improve the candidate solution iteration after iteration.

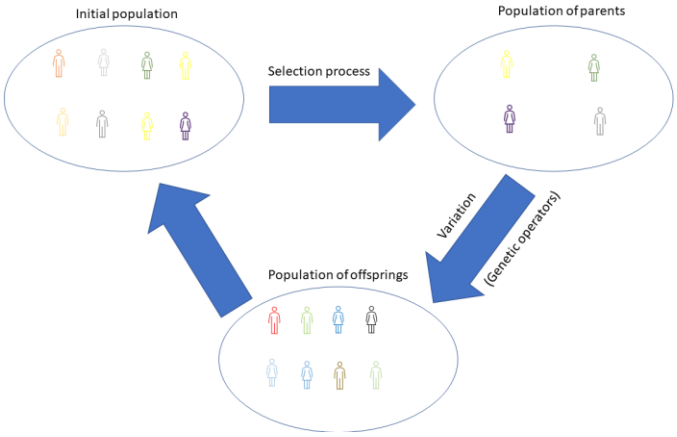


Figure 3.4 Illustration of the evolutionary process that characterizes Evolutionary Computation algorithms, inspired by Darwin's Theory of Evolution.

In DE a random population of N individuals is created. Then, after setting a crossover probability Pc and a mutation scale factor F, each individual Ig in the population is mutated according to the formula:

$$I_{g+1} = \begin{cases} R1_g + F(R2_g - R3_g) \text{ with probability } P_c \\ I_g \text{ with probability } 1 - P_c \end{cases} \quad (3.1)$$

Where g indicates the generation of the individual and R1, R2 and R3 are three random individuals of the population of the current generation such that R1≠ R2≠ R3≠ I. This process is then repeated until a certain stopping criteria is met, usually a certain number of generations. DE is used in [8] to optimize the parameters ν, γ and C of nu-SVR, and ϵ, γ and C of eps-SVR, together with the weights $\{w_e, w_v\}$ of the Equation (3.1).

3.3.2 Genetic Programming

Genetic Programming (GP) is another sub-field on Evolutionary Computation where individuals are computer programs and usually are represented as trees, as shown in Figure 1.5, and

are defined by two sets, one of primitive functions and one of terminals, both set by the user. In the tree shown in Figure 1.5 sets are $\text{terminals} = \{\text{x1}, \text{x2}, \text{x3}\}$ and $\text{functions} = \{\div, \times, +, -\}$ and the tree results in the mathematical function $\text{f}(\text{x1}, \text{x2}, \text{x3}) = \text{x1} \div (\text{x1} \times \text{x2})$.

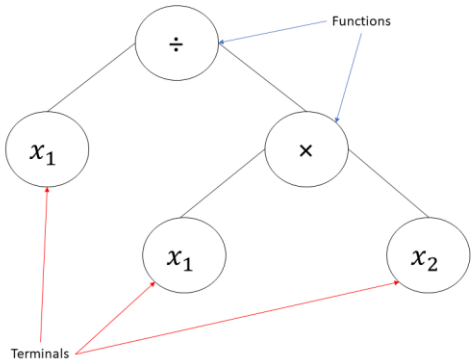


Figure 3.5 Example of a representation of an individual as a tree.

3.3.2.1 Geometric Semantic Genetic Programming

In GP the individuals can be seen in two spaces, in genotypic or syntactic space as trees, or in a semantic space as vectors that represent the output. Geometric Semantic Genetic Programming (GSGP) is a variation of GP where Geometric Semantic (GS) operators are used. These GS operators modify the genotype of the individual in such a way that the changes are reflected in the semantic space. As shown in the Paper [6] this operators also induce a uni-modal error surface on Supervised Learning optimization problems, meaning that the optimization algorithm will always reach the global optimum. The GS crossover produces an offspring individual To starting form the two parent trees T1 and T2 according to the formula:

$$T_o = (T_1 \times T_r) + ((1 - T_r) \times T_2$$

where T_r is a random function whose output values lie in the range [0,1]. While GS mutation produces an offspring T_0 starting from a parent T_1 according to the formula:

$$T_o = T_1 + m_s \times (T_{r1} - T_{r2})$$

where T_{r1} and T_{r2} are random functions whose output values lie in the range [0,1] and ms is a parameter called mutation step. GSGP has two major drawbacks, the first one being that the GS operators increase the size of the offsprings rapidly, making the algorithm very slow and difficult to use. In [16] a solution is proposed, but the final tree produced by the algorithm is too big to be visualized. Another drawback of GSGP is that for the GS crossover to lead to the global optimum its semantics need to be surrounded by the semantics of the individuals in the population, and this makes the initialization of the algorithm extremely important. [15] [17]

3.3.2.2 Tree-based Pipeline Optimization Tool

Tree-based Pipeline Optimization Tool (TPOT) is a GP technique that automates the optimization of Machine Learning pipelines. TPOT represents pipelines as binary expression trees with ML operators as terminals, and optimize them according to both performance and complexity. [18]

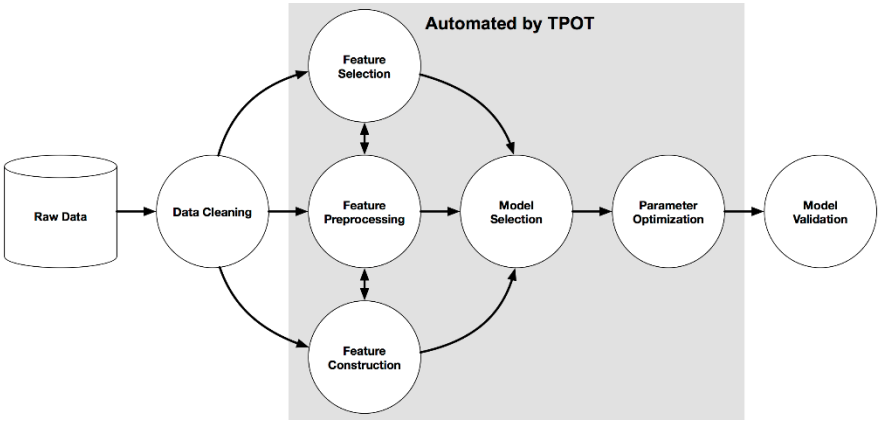


Figure 3.6 Example of a machine learning pipeline.[18]

4. EXPERIMENTAL STUDY

4.1. DATA USED

Two datasets were used used for this experiment, one[19] is a energy consumption time series provided by America Electric Power (AEP), and made of 2500 hourly observations from 2004 to 2005. In order to use it with regression models the data needed to be transformed from simple time series to a regression data set. Each observation was used together with the previous 25 as features to predict the value two time step ahead, as shown in Figure 4.1.

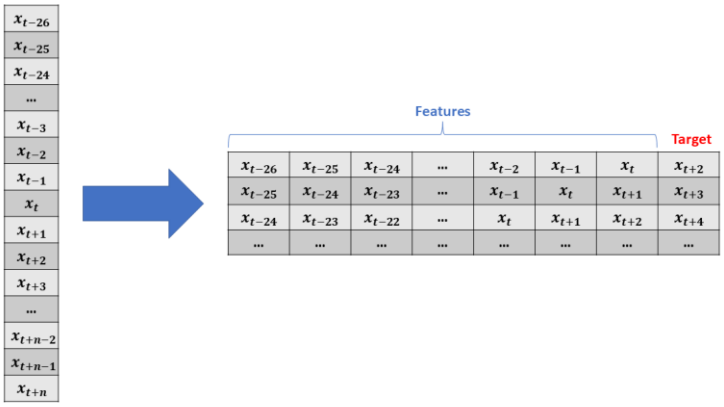


Figure 4.1 Transformation of the AEP data in order to use it for regression.

The China Energy Consumption(CEC) dataset, taken from [6], is a time series made up of China's yearly energy consumption from 1965 to 2017. In order to use it with regression models the same transformations applied to the first dataset were repeated, although this time, following the instruction from [6], 5-year data made up the features to predict the 6th year, as shown Figure 4.2.

$$\begin{bmatrix} \text{data}_1 & \cdots\cdots & \text{data}_m \\ \text{data}_2 & \cdots\cdots & \text{data}_{m+1} \\ \vdots & & \vdots \\ \text{data}_{n-m} & \cdots\cdots & \text{data}_{n-1} \end{bmatrix} \Rightarrow \begin{bmatrix} \text{data}_{m+1} \\ \text{data}_{m+2} \\ \vdots \\ \text{data}_n \end{bmatrix}$$

Figure 4.2 Transformation of the CEC data in order to use it for regression.

4.2. EXPERIMENTAL SETTINGS

In this paper the performance of EC models on time series are compared to the performance of other machine learning models used in the literature. In particular LSTM[6] and a combination of nu-SVM and epsilon-SVR with parameters and weights optimized with DE[8]. As proved in the literature [17], GSGP performs better when trained with a small population, so for this experiment it was trained with 50 individuals over 20000 generations, while GP with 500 individuals over 2000 generations. In this way both algorithms evaluate the same number of total individuals, exactly 1 million each. Instead,

being computational power a big constraint, TPOT evaluated 25 individuals over 30 generations, in this way the computational times of the three algorithms(GSGP, GP and TPOT) are similar. The metric used to evaluate the performance of every regression model was the root mean squared error (RMSE) between predicted and actual output. 30 independent runs were made for each evaluated method, in order to provide statistical significance. Every run was performed with a different train/test split, consistent across different methods. The TPOT and the SVR optimized with DE algorithms, after a learning/test split that is consistent with the train/test split performed with the other algorithms, were trained using a validation set obtained from the split of the learning dataset into train and validation. The results presented in Table 4.3 are all taken from the last iteration of the respective algorithms. In order to find the best possible combination of parameters a grid search was performed for all the models, except for the parameters mentioned above(number of individuals and generations for GSGP, GP and TPOT) that were handpicked. Another exception were also the parameters of LSTM applied to the CEC dataset, since they were provided in [6]. During the grid search, all the combinations of parameters were tested on three different train/test data splits, and the ones that had the best overall performance were chosen. The search space for the grid search of both datasets are shown in Table 4.1 and in Table 4.2.

Table 4.1 Search space and final parameters of the grid search for the AEP dataset[19].

Model	Parameter	Search space	Final Choice
GSGP	Function set	$\{+, \times, \div, -\}$ $\{+, \times, \div, -, log, 1f, tanh, sin, cos\}$ $\{+, \times, \div, -, log, 1f, tanh, sin, cos\}$ $\{+, \times, \div, -, log, 1f, tanh, sin, cos, mean, max, min\}$	$\{+, \times, \div, -\}$
	Constant set	[-1,1] [-5, 5] [-100, 100]	[-1,1]
	Probability of constant	0.1 0.7	0.1
	Mutation step	[0, 1] [1, 20] [5, 100]	[5, 100]
	Tournament selection pressure	0.04 0.1 0.7	0.04
	Mutation probability	0.1 0.7	0.7
	Crossover probability	0.1 0.7	0.7
GP	Function set	$\{+, \times, \div, -\}$ $\{+, \times, \div, -, log, 1f, tanh, sin, cos\}$ $\{+, \times, \div, -, log, 1f, tanh, sin, cos\}$ $\{+, \times, \div, -, log, 1f, tanh, sin, cos, mean, max, min\}$	$\{+, \times, \div, -\}$
	Constant set	[-1,1] [-5, 5] [-100, 100]	[-1,1]
	Probability of constant	0.1 0.7	0.1
	Tournament selection pressure	0.1 0.9	0.9
	Mutation probability	0.1 0.7	0.7
	Crossover probability	0.1 0.7	0.1
LSTM	LSTM layers	1 5	5
	LSTM units	10 20	10
	Dense layers	0 2	2
	Epochs	30 100	100
	Batch size	1 16 64	64
SVR optimized with DE	Strategy	best1bin randtobestexp randtobest1binn	best1bin

Table 4.2 Search space and final parameters of the grid search for the CEC dataset[6]

Model	Parameter	Search space	Final Choice
GSGP	Function set	$\{+, \times, \div, -\}$ $\{+, \times, \div, -, log, lf, tanh, sin, cos\}$ $\{+, \times, \div, -, log, lf, tanh, sin, cos\}$ $\{+, \times, \div, -, log, lf, tanh, sin, cos, mean, max, min\}$	$\{+, \times, \div, -\}$
	Constant set	[-1,1] [-5, 5] [-100, 100]	[-1,1]
	Probability of constant	0.1 0.7	0.7
	Mutation step	[0, 1] [1, 20] [5, 100]	[1, 20]
	Tournament selection pressure	0.04 0.1 0.7	0.04
	Mutation probability	0.1 0.7	0.7
	Crossover probability	0.1 0.7	0.7
GP	Function set	$\{+, \times, \div, -\}$ $\{+, \times, \div, -, log, lf, tanh, sin, cos\}$ $\{+, \times, \div, -, log, lf, tanh, sin, cos\}$ $\{+, \times, \div, -, log, lf, tanh, sin, cos, mean, max, min\}$	$\{+, \times, \div, -\}$
	Constant set	[-1,1] [-5, 5] [-100, 100]	[-1,1]
	Probability of constant	0.1 0.7	0.7
	Tournament selection pressure	0.1 0.9	0.1
	Mutation probability	0.1 0.7	0.1
	Crossover probability	0.1 0.7	0.1
SVR optimized with DE	Strategy	best1bin randtobestexp randtobest1bin	randtobestexp

Table 4.3 Parameters taken from [6] for the LSTM algorithm when applied to the CEC dataset.

Units	10
Loss	Mean Squared Error
Optimizer	Adam
Epochs	30
Batch Size	1

4.3. EXPERIMENTAL RESULTS

The results of this study are shown in Tables 4.4 and 4.5, respectively using the AEP dataset [19] and the CEC dataset[6]. In order to prove the statistical significance of the experiment the Wilcoxon rank-sum test for pairwise data comparison was performed, and the p-values are presented in the Tables 4.6 and 4.7. The Figures 4.3, 4.6, 4.7, 4.10 and 4.13 show the progress of fitness through generations for the GSGP, GP and TPOT algorithms. For a better understanding the logarithmic scale of the results of the GP algorithm are also presented. The results clearly show that in both cases the variation of GP, TPOT and GSGP, have increased performance compared to GP itself and state-of-the-art algorithm SVR optimized with DE[8]. While using the the CEC dataset the performance of GSGP and TPOT are comparable to each other, being the p-value of the Wilcoxon test greater than 0.05(as shown in Table 4.7), while with the AEP dataset, the best performing algorithm is LSTM, with comparable performance to GSGP, being the p-value of the Wilcoxon test greater than 0.05(as shown in Table 4.6).

Table 4.4 Performance of the algorithms on the AEP dataset [19]

Algorithm	RMSE on train	RMSE on test
GSGP	327.9414	451.2126
GP	265.1971	580.9472
SVR optimized with DE	543.5433	549.5942
LSTM	**238.3935**	**294.1798**
TPOT	246.3488	294.5654

Table 4.5 Performance of the algorithms on the CEC dataset.

Algorithm	RMSE on train	RMSE on test
GSGP	**1.6611**	**40.2433**
GP	9.7530	76.5929
SVR optimized with DE	135.6913	130.42633
LSTM	67.613	65.9228
TPOT	31.333	47.6787

Table 4.6 Wilcoxon p-test values for each algorithm tested against LSTM, all applied to the AEP dataset[19]. Presented with an asterisk(*) if smaller than 0.05.

Algorithm	Wilcoxon p-value
GSGP	1.7344e-6*
GP	1.9209e-06*
SVR optimized with DE	1.7344e-6*
TPOT	0.1414

Table 4.7 Wilcoxon p-test values for each algorithm tested against GSGP, all applied to the CEC dataset[6]. Presented with an asterisk(*) if smaller than 0.05.

Algorithm	Wilcoxon p-value
TPOT	0.7189
GP	0.01175*
SVR optimized with DE	1.7344e-06*
LSTM	1.7344e-06*

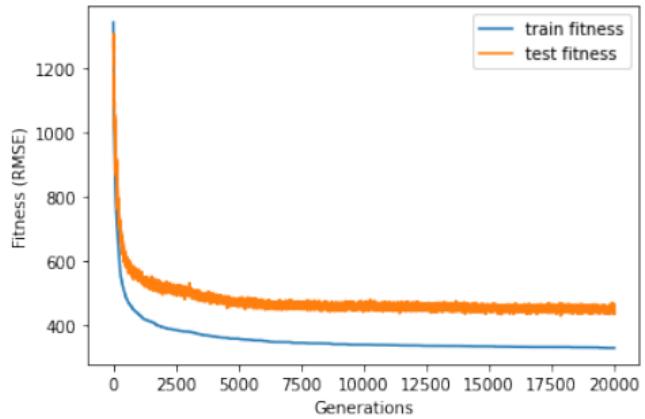
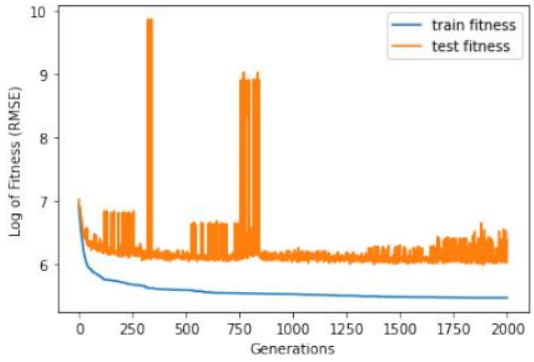
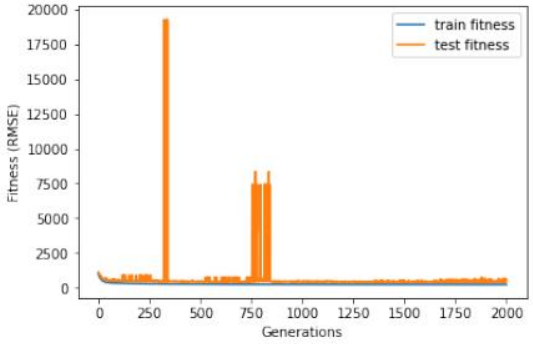


Figure 4.3 Average fitness of the GSGP algorithm over the generations with the AEP dataset [19].



(a) Logarithmic scale.



(b) Not scaled.

Figure 4.4 Figure 3.4: Average fitness of the GP algorithm over the generations with the AEP dataset [19].

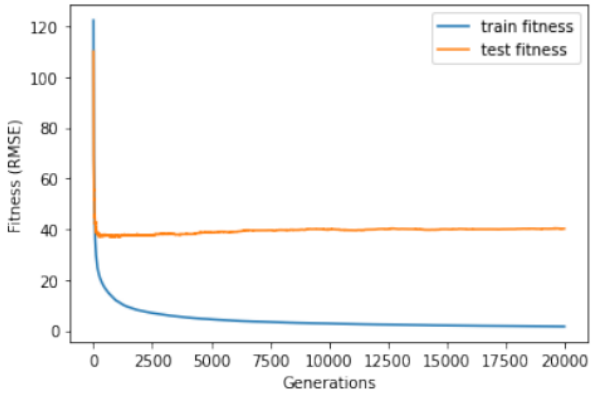
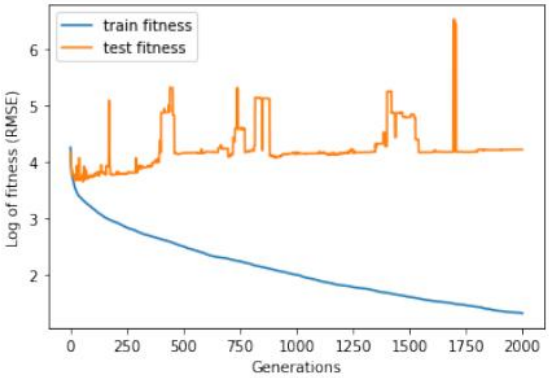
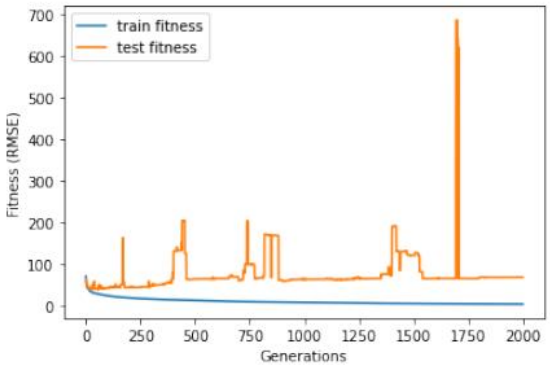


Figure 4.5 Average fitness of the GSGP algorithm over the generations with the CEC dataset.

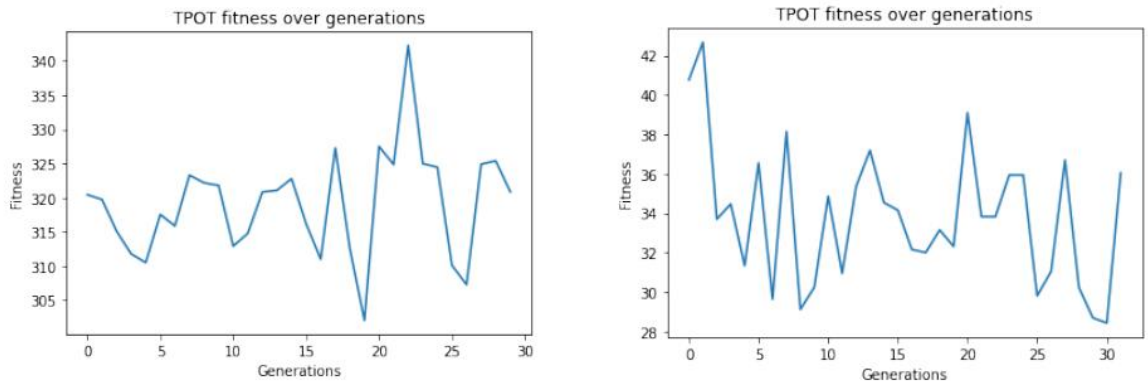


(a) Logarithmic scale.



(b) Not scaled.

Figure 4.6 Average fitness of the GP algorithm over the generations with the CEC dataset.



(a) AEP dataset [19] (b) CEC Dataset [6]

Figure 4.7 TPOT's fitness through generations for both datasets on the learning partition.

4.3.1. TPOT models

Of all the 30 generated TPOT models applied to the AEP dataset [19], two were selected for a deeper analysis, the best performing one, shown in Figure 4.8, and the smallest one, shown in Figure 4.9. The first one passes the features through two stacked estimators, ExtraTreeRegressor and LassoLarsCV, then it scales the output with a MaxAbsScaler and PolynomialFeatures and finally produces the estimation by passing the scaled values through another LassoLarsCV and an XGBRegressor. The second model is fairly more simple, it passes the data through a SDGRegressor, it scales the results with PolynomialFeatures and produces an estimation by passing the scaled values trough a LassoLarsCV.

```
import numpy as np
import pandas as pd
from sklearn.ensemble import ExtraTreesRegressor
from sklearn.linear_model import LassoLarsCV
from sklearn.model_selection import train_test_split
from sklearn.pipeline import make_pipeline, make_union
from sklearn.preprocessing import MaxAbsScaler, PolynomialFeatures
from tpot.builtins import StackingEstimator
from xgboost import XGBRegressor
from tpot.export_utils import set_param_recursive

# NOTE: Make sure that the outcome column is labeled 'target' in the data file
tpot_data = pd.read_csv('PATH/TO/DATA/FILE', sep='COLUMN_SEPARATOR', dtype=np.float64)
features = tpot_data.drop('target', axis=1)
training_features, testing_features, training_target, testing_target = \
            train_test_split(features, tpot_data['target'], random_state=11)

# Average CV score on the training set was: -290.4296021986596
exported_pipeline = make_pipeline(
    StackingEstimator(estimator=ExtraTreesRegressor(bootstrap=True, max_features=0.4, min_samples_leaf=17, min_samples_split=6,
n_estimators=100)),
    StackingEstimator(estimator=LassoLarsCV(normalize=False)),
    MaxAbsScaler(),
    PolynomialFeatures(degree=2, include_bias=False, interaction_only=False),
    StackingEstimator(estimator=LassoLarsCV(normalize=False)),
    XGBRegressor(learning_rate=0.1, max_depth=7, min_child_weight=2, n_estimators=100, n_jobs=1, objective="reg:squarederror",
subsample=0.8500000000000001, verbosity=0)
)
# Fix random state for all the steps in exported pipeline
set_param_recursive(exported_pipeline.steps, 'random_state', 11)

exported_pipeline.fit(training_features, training_target)
results = exported_pipeline.predict(testing_features)
```

Figure 4.8 Code of the best performing TPOT model.

```python
import numpy as np
import pandas as pd
from sklearn.linear_model import LassoLarsCV, SGDRegressor
from sklearn.model_selection import train_test_split
from sklearn.pipeline import make_pipeline, make_union
from sklearn.preprocessing import PolynomialFeatures
from tpot.builtins import StackingEstimator
from tpot.export_utils import set_param_recursive

# NOTE: Make sure that the outcome column is labeled 'target' in the data file
tpot_data = pd.read_csv('PATH/TO/DATA/FILE', sep='COLUMN_SEPARATOR', dtype=np.float64)
features = tpot_data.drop('target', axis=1)
training_features, testing_features, training_target, testing_target = \
            train_test_split(features, tpot_data['target'], random_state=20)

# Average CV score on the training set was: -307.9147020549357
exported_pipeline = make_pipeline(
    StackingEstimator(estimator=SGDRegressor(alpha=0.0, eta0=0.1, fit_intercept=False, l1_ratio=0.75, learning_rate="invscaling",
loss="epsilon_insensitive", penalty="elasticnet", power_t=0.5)),
    PolynomialFeatures(degree=2, include_bias=False, interaction_only=False),
    LassoLarsCV(normalize=True)
)
# Fix random state for all the steps in exported pipeline
set_param_recursive(exported_pipeline.steps, 'random_state', 20)

exported_pipeline.fit(training_features, training_target)
results = exported_pipeline.predict(testing_features)

```

Figure 4.9 Code of the smallest TPOT model.

4.4. STUDY ON GSGP FEATURE SELECTION ABILITY

As explained in Section 3.3.2.1 the final individuals produced by GSGP are trees, that take as input the features and, by applying mathematical functions, produce an output. Since the size of the individuals increases rapidly throughout generations, is not possible to visualize the one produced at the last step, and therefore we cannot understand what happens to each feature from the dataset and if there is any feature selection performed by GSGP. An experiment was set in order to test this, starting from the AEP dataset[19] the 7 least relevant features were recognized using to two methods, correlation to the target and the presence or not in the best individual at the last generation of the GP algorithm. Then random noise was applied to those features and also 7 other random features were created. Then the performance of GSGP on normal data were compared to the ones of the same algorithm applied to 5 modified datasets:

- one with random noise applied to the 7 least important features according to correlation
- one with the addition of 7 new randomly created features
- one with both of the previous methods
- one with random noise applied to the 7 least important features according to both GP and correlation
- and finally one with random noise applied to the 7 least important features according to both GP and correlation and 7 new random features.

In order to have statistical significance the algorithm was tested on the data for 30 generations, and the average performance are shown in Table 4.8. Additionally the p-values of the Wilcoxon rank-sum test for pairwise data comparison between GSGP with the standard dataset and the same algorithm with the modified datasets presented above are shown in Table 4.9, and the graphical representation of the performance over the generations are shown in the images 4.10, 4.11, 4.12, 4.13 and 4.14. As shown GSGP performs worse on the modified datasets and we can assume that it does not perform feature selection, but it uses all the information from all the features.

Table 4.8 performance of GSGP on modified versions of the AEP dataset [19]

Dataset	RMSE on train	RMSE on test
Standard AEP dataset	327.9414	451.2126
Noise applied to least important features according to correlation	371.374	612.763
Noise applied to least important features according to both GP and correlation	344.822	571.910
Randomly created features	396.812	654.146
Noise applied to least important features according to correlation plus randomly created features	366.854	611.273
Noise applied to least important features according to correlation and GP plus randomly created features	358.547	608.753

Table 4.9 P-value of GSGP on modified versions of the AEP dataset compared to the ones on the standars dataset [19]

Dataset	p-value
Noise applied to least important features according to correlation	6.983e-06
Noise applied to least important features according to both GP and correlation	5.306e-05
Randomly created features	2.163e-05
Noise applied to least important features according to correlation plus randomly created features	4.286e-06
Noise applied to least important features according to correlation and GP plus randomly created features	6.984e-06

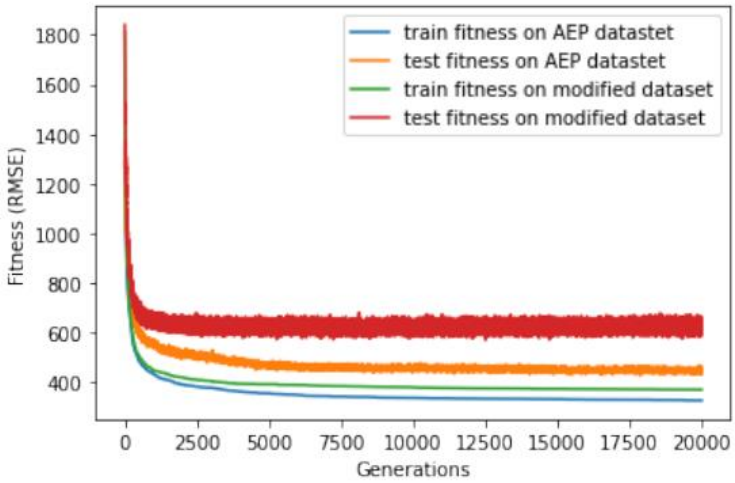


Figure 4.10 Fitness of GSGP over generations on AEP dataset and dataset with random noise added on least important features according to correlation.

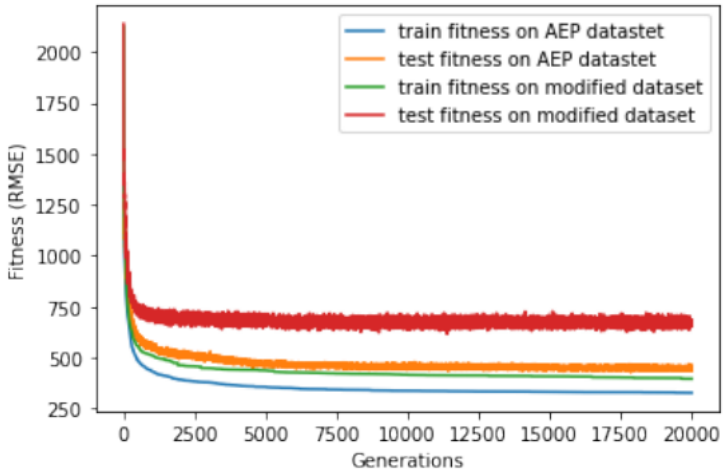


Figure 4.11 Fitness of GSGP over generations on AEP dataset and dataset with added random features.

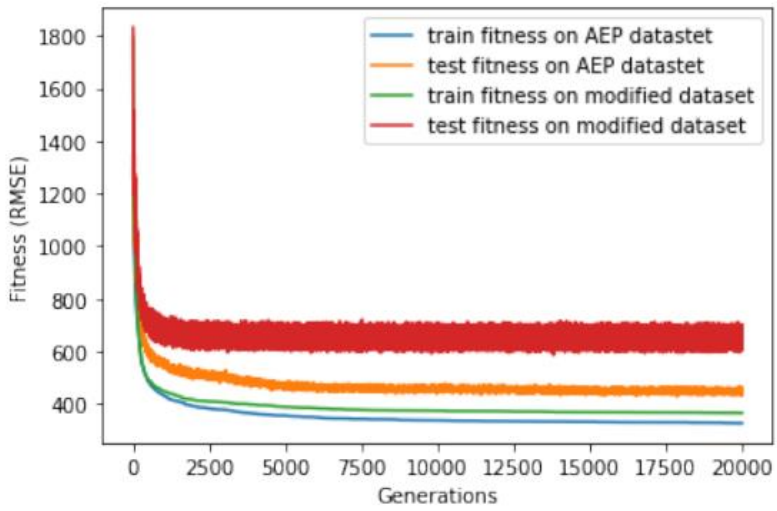


Figure 4.12 Fitness of GSGP over generations on AEP dataset and dataset with random noise added on least important features according to correlation and added random features.

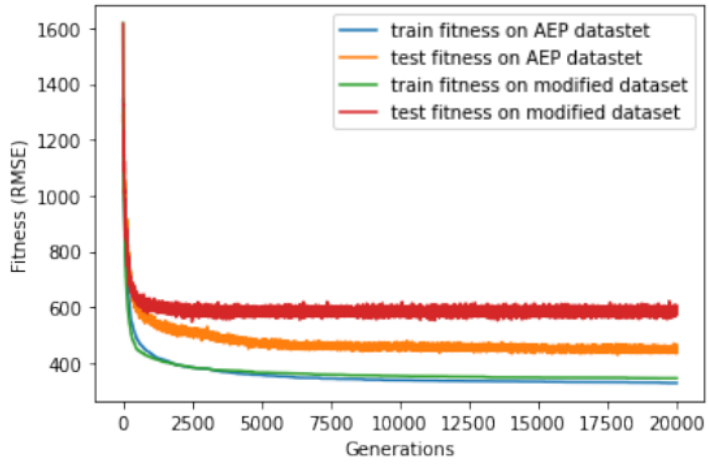


Figure 4.13 Fitness of GSGP over generations on AEP dataset and dataset with random noise added on least important features according to both correlation and GP.

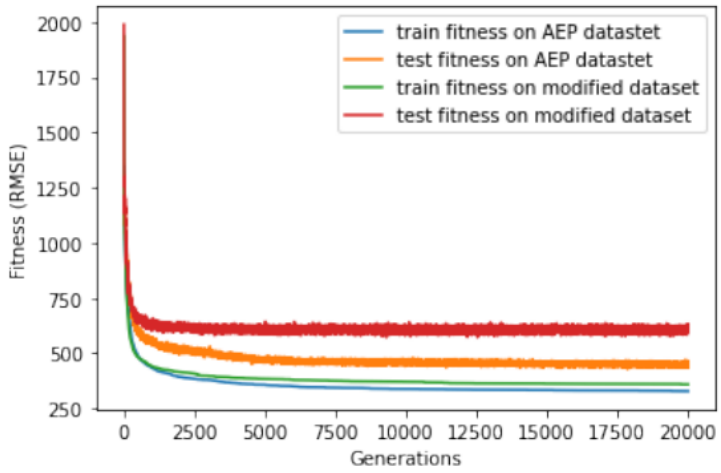


Figure 4.14 Fitness of GSGP over generations on AEP dataset and dataset with random noise added on least important features according to both correlation and GP plus added random features.

4.5. FINAL REMARKS

In this Chapter it was shown how the variations of GP exceed standard GP and have comparable performance to state-of-the-art algorithms. In particular we looked at GSGP and TPOT, and compared them to GP, LSTM and SVR optimized by DE. Then it was performed a study about the capability of GSGP to achieve feature selection. It was done by comparing the performance of GSGP on the standard dataset, to modified versions of it, where noise was added to the less important features and new random features were created. The results on the latter were statistically worse, and therefore it was shown that GSGP is not actually able to perform feature selection.

5. CONCLUSION AND FUTURE WORKS

In this thesis two variations of GP, namely GSGP and TPOT, are applied to two energy consumption time series regression problems. Their performance is then compared to state-of-the-art models, LSTM and SVR optimized with DE, and to standard GP. All the algorithms are applied to two time series forecasting problems. The first one is an energy consumption time series dataset provided by America Electric Power (AEP), and made of 2500 hourly observations from 2004 to 2005, the second one is the China Energy Consumption(CEC) dataset, that is a made up of China's yearly energy consumption from 1965 to 2017. In order to prove statistical significance, the Wilcoxon rank-sum test for pairwise data comparison is used and, to maximize the performance of all the algorithms, a greed search to find the best combination of hyperparameters is performed. It is shown that GSGP and TPOT outperform standard GP in both the scenarios and have also a better performance than SVR optimized with DE. It is presented that for the CEC dataset GSGP and TPOT have the best performance. While for the AEP dataset LSTM is the best performing algorithm, with performance comparable to TPOT. Additionally a study on the feature selection ability of GSGP is performed. The performance of the algorithm on the AEP dataset is compared with the performance on modified versions of the dataset, where either random features were added, or less important features were modified. Being the performance on the standard dataset statistically better than the one on the modified versions, it is shown that GSGP is not actually able to perform feature selection.

In the future the performance of GSGP and TPOT on other time series regression problems should be explored, in order to understand better their capabilities.

6. REFERENCES

[1] Hai xiang Zhao and Fr´ed´eric Magoul`es. “A review on the prediction of building energy consumption”. In: Renewable and Sustainable Energy Reviews 16.6 (2012), pp. 3586– 3592. issn: 1364-0321. doi: https://doi.org/10.1016/j.rser.2012.02.049. url: https://www.sciencedirect.com/science/article/pii/S1364032112001438.

[2] Aur´elie Foucquier et al. “State of the art in building modelling and energy performance prediction: A review”. In: Renewable and Sustainable Energy Reviews 23 (2013), pp. 272–288. issn: 1364-0321. doi: https://doi.org/10.1016/j.rser. 2013.03.004. url: https://www.sciencedirect.com/science/article/pii/ S1364032113001536.

[3] Y.W. Lee, Tay Gaik, and Choy Yaan Yee. “Forecasting Electricity Consumption Using Time Series Model”. In: International Journal of Engineering and Technology(UAE) 7 (Nov. 2018), pp. 218–223. doi: 10.14419/ijet.v7i4.30.22124.

[4] Jolanta Szoplik. “Forecasting of natural gas consumption with artificial neural networks”. In: Energy 85 (2015), pp. 208–220. issn: 0360-5442. doi: https : / / doi . org/10.1016/j.energy.2015.03.084. url: https://www.sciencedirect.com/ science/article/pii/S036054421500393X.

[5] Andr´e Gensler et al. “Deep Learning for solar power forecasting — An approach using AutoEncoder and LSTM Neural Networks”. In: 2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC). 2016, pp. 002858–002865. doi: 10.1109/ SMC.2016.7844673.

[6] Yan Li. “Prediction of energy consumption: Variable regression or time series? A case in China”. In: Energy Science Engineering 7 (Aug. 2019). doi: 10.1002/ese3.439.

[7] Koshy George et al. “Comparison of neural-network learning algorithms for time-series prediction”. In: 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI). 2017, pp. 7–13. doi: 10.1109/ICACCI.2017. 8125808.

[8] Fan Zhang et al. “Time series forecasting for building energy consumption using weighted Support Vector Regression with differential evolution optimization technique”. In: Energy and Buildings 126 (2016), pp. 94–103. issn: 0378-7788. doi: https: //doi.org/10.1016/j.enbuild.2016.05.028. url: https://www.sciencedirect. com/science/article/pii/S0378778816303899.

[9] Paulo Cortez, Miguel Rocha, and Jose Neves. “Genetic and Evolutionary Algorithms for Time Series Forecasting”. In: June 2001, pp. 393–402. doi: 10.1007/3-540-455175_44.

[10] Akira Hara, Jun-ichi Kushida, and Tetsuyuki Takahama. “Time Series Prediction Using Deterministic Geometric Semantic Genetic Programming”. In: 2019 IEEE International Conference on Systems, Man and Cybernetics (SMC). 2019, pp. 1945–1949. doi: 10.1109/SMC.2019.8914562.

[11] Vipul K. Dabhi and Sanjay Chaudhary. “Time Series Modeling and Prediction Using Postfix Genetic Programming”. In: 2014 Fourth International Conference on Advanced Computing Communication Technologies. 2014, pp. 307–314. doi: 10.1109/ACCT. 2014.33.

[12] Vladimir Vapnik. The nature of statistical learning theory. Springer science & business media, 1999.

[13] Bernhard Sch¨olkopf et al. “New Support Vector Algorithms”. In: Neural Computation 12.5 (2000), pp. 1207–1245. doi: 10.1162/089976600300015565.

[14] Charles Darwin. Origin of species. New York, Boston: H.M. Caldwell Co., 1900. doi: 10.5962/bhl.title.959.

[15] Alberto Moraglio, Krzysztof Krawiec, and Colin Johnson. “Geometric Semantic Genetic Programming”. In: vol. 7491. Sept. 2012, pp. 21–31. isbn: 9783642329364. doi: 10.1007/978-3-642-32937-1_3.

[16] Mauro Castelli, Sara Silva, and Leonardo Vanneschi. “A C++ framework for geometric semantic genetic programming”. English. In: Genetic Programming And Evolvable Machines 16.1 (Jan. 2015), pp. 73–81. issn: 1389-2576. doi: 10.1007/s10710-0149218-0.

[17] Leonardo Vanneschi. “An Introduction to Geometric Semantic Genetic Programming”. In: Studies in Computational Intelligence NEO 2015 (2016), 3–42. doi: 10.1007/9783-319-44003-3_1.

[18] Trang T Le, Weixuan Fu, and Jason H Moore. “Scaling tree-based automated machine learning to biomedical big data with a feature set selector”. In: Bioinformatics 36.1 (2020), pp. 250–256.

[19] Rob Mulla. Hourly Energy Consumption. https://www.kaggle.com/robikscube/ hourly-energy-consumption/version/3. Accessed: 2021-12-08.

]

NOVA Information Management School
Instituto Superior de Estatística e Gestão de Informação

Universidade Nova de Lisboa