Journal of Advanced Research in Applied Sciences and Engineering Technology

Journal homepage:
https://semarakilmu.com.my/journals/index.php/applied_sciences_eng_tech/index
ISSN: 2462-1943

TPOT-MTR: A Multiple Target Regression Based on Genetic Algorithm of Automated Machine Learning Systems

Hanafi Majid[1,2,*], Syahid Anuar[1], Noor Hafizah Hassan[1]

[1] Razak Faculty of Technology and Informatics, Universiti Teknologi Malaysia, 54100 Kuala Lumpur, Malaysia
[2] Malaysia Board of Technologists (MBOT), 62000 Putrajaya, Wilayah Persekutuan Putrajaya, Malaysia

ARTICLE INFO

Article history:
Received 24 November 2022
Received in revised form 11 April 2023
Accepted 18 April 2023
Available online 9 May 2023

Keywords:
Multi-target regression; automated machine learning; genetic algorithm; regression analysis; multi-output regression

ABSTRACT

The concept that a cross correlation might improve prediction error underpins machine learning algorithms for multi-target regression (MTR). Numerous MTR approaches have been created in recent years, however there are still uncertainties concerning how their performances are impacted by dataset properties such as linearity, number of targets, and cross correlational complexity. In order to contribute to a better understanding of the relationship between dataset properties and MTR methods, authors proposed a new model of TPOT-MTR, which its result will be compared to previously generated 33 synthetic datasets with controlled characteristics and tested their performance against other two MTR methods, Random Forest and SVM. The results demonstrated that TPOT-MTR approaches could enhance performance even in datasets with non-linearly correlated targets, although the prediction improvement varies depending on the method and regressor combinations used.

1. Introduction

Multi-target regression (MTR) methods are statistical methods to deal with the prediction of multiple output variables which are related to the same input set. They can be used in a wide range of applications such as predicting multiple characteristics of a product, predicting the risk of disease progression, predicting the effect of a drug on multiple biological targets, and predicting the performance of complex systems. When complexity, linearity, and even target correlations are considered, it is difficult to relate the performance of MTR approaches in datasets to baseline scenario features [1]. There are different types of MTR methods; Supervised learning models can be used to improve the prediction of multiple targets with a single input. These models learn how to predict each target by training on data that corresponds to individual examples of the target. The most widely used supervised learning model for multi-target prediction is the Support Vector Machine (SVM) [2].

* *Corresponding author.*
E-mail address: hanafimajid@graduate.utm.my

https://doi.org/10.37934/araset.30.3.104126

1.1 Prior Works on Multi Target Regression

Both supervised and unsupervised learning models can be used to improve the prediction of multiple targets with a single input. However, this paper will be discussing about multi target regression in supervised learning. Researchers evaluated in Lingitz *et al.,* [3] regarding linear, ridge, and lasso regression, multivariate adaptive regression or multiple target regression, k nearest neighbour, regression tree (bagged, unbagged, and boosted), RF, and ANN. Ultimately, after evaluation researchers found RFs outperformed.

The simulation-based technique of RFs might be used to often retrain models in order to preserve model performance since RFs are sensitive to range changes in the predictor variables [4]. Supervised learning models are often used to improve the prediction of multiple targets with a single input. These models learn how to predict each target by training on data that corresponds to individual examples of the target. In order to recap the level of use, the SVM is the most used supervised learning model for multi-target prediction. The SVM is a supervised learning model that uses a linear regression to predict the target variable from the input variables. The SVM is composed of three parts: a kernel function, a cost function, and an optimization algorithm. The kernel function is used to transform the input data into a feature space. The cost function is used to determine how much weight should be given to each feature in the feature space. The optimization algorithm is used to find the best combination of cost function and kernel function.

The unlimited FIRE algorithm approach, in general, strikes a fair mix between simplicity and precision. Despite tending to produce slightly less accurate models than the most accurate random forests, the size difference is extremely significant. To simulate multi-target regression issues, researchers thus think that the unlimited FIRE with linear terms is a very excellent option. Despite certain difficulties with the training and testing set limitations, the results show that Multi Output Gaussion Process (MOGP) regression has some promise for usage in Residual Storage Life (RSL) prognostic situations [5]. To obtain the optimal forecasting model, MOGP employs the Bayesian algorithm methodology. At the core of the MOGP process lies a Gaussian process as the underlying model. This process is used to model the correlations between the input variables and the output variable. The input variables are then combined with the output variable to form a joint probability distribution. This distribution is used to determine the hyperparameters associated with the model.

The best convolutional and recurrent neural network architectures and hyperparameters are sought using a revolutionary genetic algorithm-based Network Architecture Search (NAS) technique [6]. Real-world traffic speed data from Berlin, Germany is used to evaluate the suggested system. Experimental findings in Li *et al.,* [6] show that the suggested AutoML framework is more effective, efficient, and robust than alternative benchmarking techniques. This study demonstrates the enormous potential of AutoML approaches in applications related to intelligent transportation. Having stated that, this research will also investigate the use of a genetic algorithm to use AutoML in multiple target regression.

A wide range of possible applications for multiple-output regression on data streams exist, including forecasting the weather and predicting air quality, among others. Having said that, this work aims to capitalize on its benefits by enhancing it with the capacity to apply a genetic-based MTR algorithm [7]. Multiple outputs frequently include connected information, but passive-aggressive (PA) simply views these outputs as separate tasks and is unable to detect these relationships [7]. By linking the dot amongst several targets, the suggested TPOT-MTR will be successful in reducing the problem. An Auto-ML system known as TPOT (Tree-Based Pipeline Optimization) employs genetic algorithms in order to improve machine learning pipelines. It does this by making use of another Genetic Algorithm framework known as DEAP (Distributed Evolutionary Algorithms in Python).

However, this paper will be discussing about new proposed method of multi target regression which is named as TPOT-MTR and use TPOT as a core and main engine for regression.

According to a prior study, whereas the LSTM model cannot be acquired with a lower sample size, multiple linear regression models able to do it. In addition, it has been found that when the sample size is large enough, LSTM forecasting models of intensive resources produce predictions that are more accurate. When the resource-intensive nature of apps is considered, this conclusion has more useful applications [8]. In the other hand, A novel strategy for the AutoML domain was introduced in TPOT (2016) by applying genetic algorithm. Although it was only concerned with classification algorithms and classification accuracy, it was effective in breaking down the machine learning process into a series of incremental steps that could be automated one at a time. Three primary jobs or operators are included in TPOT (Tree-based Pipeline Optimization Tool): feature preprocessing operators, feature selection operators, and supervised classification and regression operators. Each of these operators was considered as a genetic algorithm primitive, and genetic algorithm trees were built. Because a structured task-based methodology was adopted, TPOT was particularly adaptable, allowing for easy scaling and the addition or removal of additional pipeline nodes [9].

1.2 Genetic-Based Algorithm of AutoML

Genetic algorithms, which are founded on the ideas of natural selection and evolution, are one of the techniques utilised in AutoML. Natural selection in biology serves as the basis for genetic algorithms (GAs), a category of optimization method. With each member of the population representing a potential solution, GAs search through a wide space of potential solutions using a population-based method. The population's members are exposed to selection, crossover, and mutation processes that mimic natural processes of reproduction and mutation and produce young individuals who are more fit than their parents.

In the context of AutoML, genetic algorithms can be used to look for the best feature engineering methods, model topologies, and hyperparameters. Genetic algorithms can effectively search through a huge number of alternative solutions to discover the best one by modelling each potential solution as an individual in a population and utilizing the selection, crossover, and mutation operations to develop the population.

There have been several studies that have explored the use of genetic algorithms in AutoML. For example, in a paper by LeDell and Poirier [10] presents an automated hyper-parameter optimization method based on the Parallel Genetic Algorithm (PGA) for fast-hyperparameters optimization on the Internet of Things (IoT) field, particularly for analyzing spatiotemporally coherent data collected by wireless sensors. For the efficacy of deep learning models, hyperparameter settings during training are regarded as crucial factors. Automated Machine Optimization (HPO) is primarily utilised for grid search and hyperparameter search based on genetic algorithm, but also contains population initialization, fitness function, tournament selection, crossover operators, mutation operators, subgroup exchange, and evolution. The hyperparameter opponents proposed are discussed and implemented.

Based on GA, PGA is a population-based search algorithm that accelerates the solution by calculating multiple populations in parallel. It has no requirements for the data dimension at runtime and can reduce the number of earlier model solutions. Genetic Algorithms are used to optimize the parameters of models using genetic nano algorithms. These activities have involved HPO implementation. The genetic algorithm is then executed independently in each subgroup, and adjustments are made between subgroups on an individual basis. Eventually, the optimal individual for the paradigm is evolved. However, the PGA model in research by LeDell and Poirier [10] appears

to have limitations in terms of the relationship between outputs, in contrast to multi-target regression, which considers the relationship between variables.

Multi-target regression is a challenging problem in machine learning that aims to simultaneously predict multiple output variables. In recent years, several AutoML solutions for this problem have been devised, including H2O, AutoGluon, and FEDOT. In this response, I will provide a concise overview of these AutoML solutions, and the work analysis associated with them. H2O is an open-source AutoML platform that offers a variety of tools for the development and deployment of machine learning models. It incorporates gradient boosting, random forest, and deep learning as algorithms for multi-target regression. H2O utilizes a combination of grid search and random search to optimize hyperparameters and provides an intuitive interface for model selection and deployment. The article by LeDell and Poirier [10] contains an analysis of related H2O-related work include its ability to handle missing or categorical data natively, it's comprehensive modeling strategy, including powerful stacked ensembles, and the ease in which H2O models can be deployed and used in enterprise production environments.

AutoGluon is an additional AutoML platform that offers a variety of tools for constructing and deploying machine learning models. It incorporates gradient boosting, random forest, and neural networks, among other algorithms for multi-target regression. AutoGluon employs a combination of Bayesian optimization and ensemble learning to optimize hyperparameters and offers an intuitive interface for model selection and deployment. The paper "AutoGluon-Tabular: Robust and Precise AutoML for Structured Data" by Gedam [11]. AutoGluon-Tabular succeeds in contrast to existing AutoML frameworks that concentrate primarily on model/hyperparameter selection by assembling multiple models and stacking them in multiple layers. Experiments indicate that our multi-layered combination of many models is a more efficient use of training time than searching for the best model. Tests on a collection of fifty classification and regression tasks from Kaggle and the OpenML AutoML Benchmark demonstrate that AutoGluon is significantly more accurate, robust, and quick. Authors find that AutoGluon frequently outperforms the finest combination of its competitors' performance in hindsight. Its limitation is enormous datasets, such as those with 400,000 rows, and the inability to complete 10 folds of cross validation.

AutoGluon AutoML can manage multi-target regression situations, wherein many target variables need to be predicted. Several models are combined into one for prediction using an alternate method of "ensembling and stacking" them in many layers [12]. By training several models, one for each target variable, and then aggregating their predictions, it can be utilised to tackle issues involving multiple targets in a regression analysis. AutoGluon is a powerful AutoML system that can handle a wide range of machine learning tasks, including multi-target regression. However, it has some disadvantages compared to other AutoML systems such as TPOT. These include limited customization options, limited interpretability, and longer training time [13]. Additionally, AutoGluon AutoML has limited support for feature engineering, meaning that users need to perform more feature engineering tasks manually before using it. TPOT allows for more customization and interpretability options, and it can be faster to train models, but it requires more effort from the user in terms of specifying custom pipelines and feature engineering. Ultimately, it will depend on your specific needs and requirements.

FEDOT (Federated Optimization) is an AutoML solution based on genetics that employs evolutionary algorithms to optimize machine learning pipelines for multi-target regression problems. It employs a hybrid of genetic algorithms and reinforcement learning for hyperparameter optimization and includes multiple feature selection, preprocessing, and model selection algorithms. FEDOT is intended for distributed and federated learning environments and can manage large-scale datasets. The paper by Olson *et al.,* [14] contains a related work analysis of FEDOT and the obtained

results confirm the correctness and effectiveness of the proposed approach in the comparison with the state-of-the-art competitors and baseline solutions. Due to the limited number of input features, the primary disadvantage is the restricted variability of the obtained decision boundary. The second implementation always passes data to the input of any model (this technique is implemented as a Stacked Ensemble (SE) in the TPOT tool, for instance), but it limits flexibility.

2. Methodology

TPOT-MTR is a new proposed method of multi target regression that uses TPOT as a core and main engine for regression. TPOT-MTR is an improved version of TPOT that use for multiple target regression and considering of multiple targets correlation. The key advantage of TPOT-MTR is that it can consider of multiple targets correlation and improve the prediction accuracy. Another key advantage of TPOT-MTR is that it is a fast algorithm that can be used for multiple target prediction. It is based on the genetic algorithm concept and uses a greedy algorithm to improve the prediction accuracy. TPOT-MTR is also able to identify the optimal parameters for the SVM kernel function and cost function.

2.1 A Proposed Approach

Traditional machine learning methods convert numerous inputs into a single output. For example, you have several images observations of flowers and forecast a single characteristic or label such as length, width, and few more. When numerous outputs are required, it is widespread practice to perform two independent classifications: first predict one variable, then the next. The issue with this method is that it entirely disregards correlations in the results.

TPOT automated machine learning is an open-source Python tool for automating data science tasks such as supervised learning, feature selection and pre-processing. It is built on top of the scikit-learn library and is designed to optimize machine learning pipelines. It is intended to be used as an assistant to data scientists and machine learning engineers, providing them with the potential of ability to optimize their machine learning models quickly and easily. TPOT automates several common data science tasks, such as feature selection and pre-processing. It does this by providing a set of predefined algorithms that can be used to optimize machine learning models.

Integrating that TPOT system, specifically this improved version of that system, which has the potential to manage multiple target regression while still making use of automated machine learning systems to locate the most effective algorithms for each solution. Based on the Figure 1, the raw data is the source data and normally in a format of Comma-separated values (CSV). Common data preprocessing activities like missing value imputation, scaling, and encoding categorical variables are all taken care of automatically by TPOT. TPOT-MTR composing two additional processes which are MTR initiator and MTR check point and between of them, the existing processes are still included.

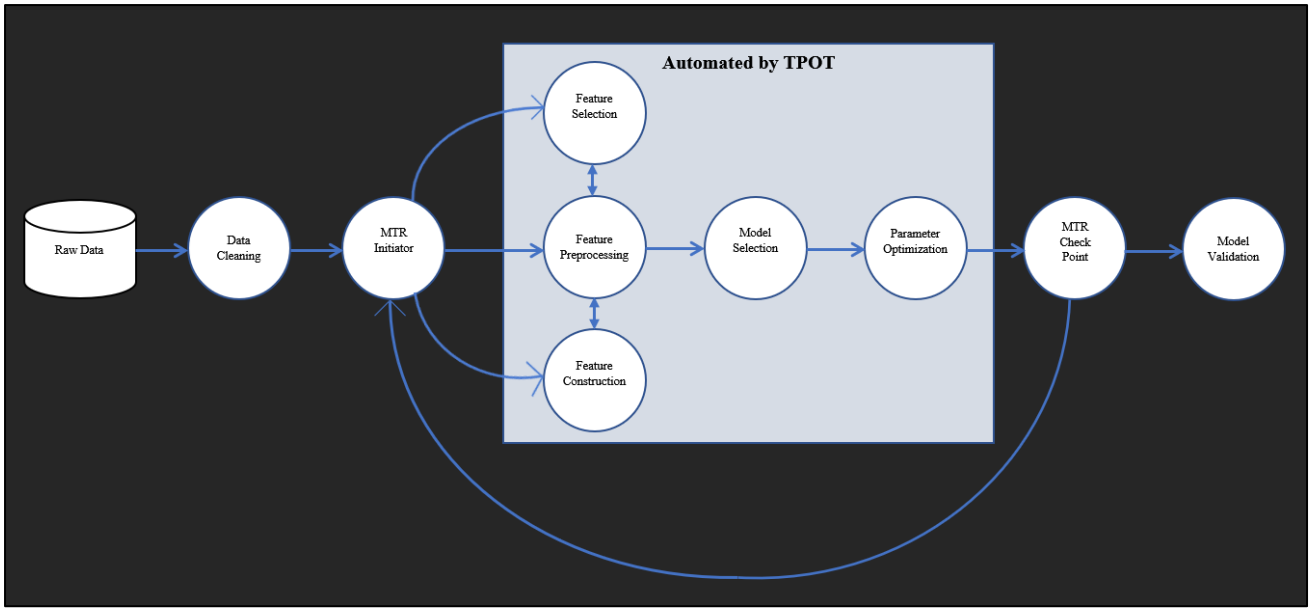

Fig. 1. TPOT-MTR workflow

The MTR initiator function as MTR starter depends on the target variables, for example if there are two number of targets, the concepts of one regressor fit for one target still be fulfilled that the second target will be managed after the first target been served. MTR check point will be validating the multiple target service in the pipeline, and it will halt the operating pipeline after all the number of targets has been served in order to finish the training of the model. If the targets have not yet been met, it will forward the information to the subsequent pipeline and continue doing so until all the targets have been accomplished as shown in Figure 2.

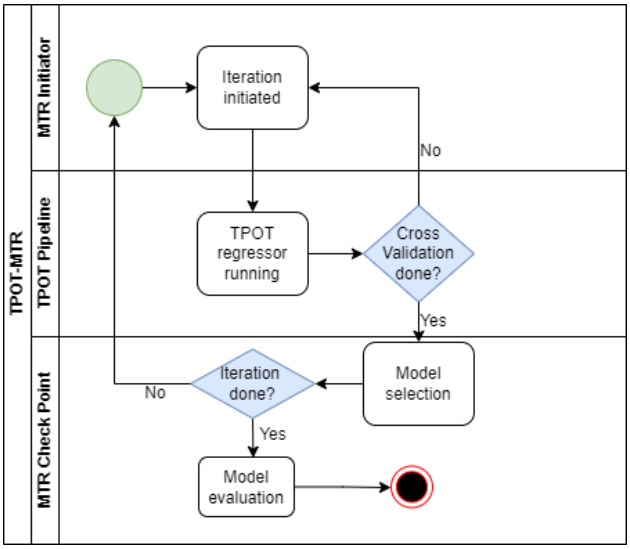

Fig. 2. TPOT-MTR activity diagram

Feature selection is the act of picking out the most outstanding features from a large pool of candidates. This has the potential to enhance the efficiency of the machine learning pipeline by decreasing the data's dimensionality. Transforming the raw features into a format that is more conducive to machine learning algorithms is what feature preprocessing is all about. Scaling, imputation, and one-hot encoding are just a few of the common preprocessing tasks that TPOT automatically handles, but users can also tweak these processes with scikit-learn transformers. The

performance of a machine learning pipeline can be enhanced through the process of feature construction, which involves the generation of new features from preexisting ones. In TPOT, new feature representations are automatically evolved via genetic algorithm.

To create new features, TPOT able perform mathematical operations on existing features or apply feature transformations like log or square root. The goal of model selection is to find the most effective machine learning technique for a specific problem. To build its pipelines, TPOT employs several different algorithms. These include decision trees, support vector machines, and random forest. TPOT uses a genetic algorithm to search through the space of possible pipelines and selects the best-performing model based on cross-validation performance. The term "parameter optimization" refers to the process of fine-tuning the hyperparameters of the chosen machine learning algorithm. To determine the optimal hyperparameters for each machine learning algorithm, TPOT employs both grid search and random search [15].

The aim behind this TPOT-MTR was to first fit for y1, and then include that knowledge into the second fit. That is, instead of fitting a separate regressor for each target of y1, y2, y3, and so forth, this TPOT-MTR employs a daisy chaining strategy that uses information from the prior target of y1 to predict y2. The same is applicable for predicting y3, for which the preceding target of y1 and y2 will be considered, and so for the remainder. The following equations will aid in comprehending the TPOT-MTR methods.

$$\{X, y1\} \rightarrow y2 \tag{1}$$

$$\{X, y1, y2\} \rightarrow y3 \tag{2}$$

2.2 Experiment

The techniques are experimentally examined in this section by comparing their performance to RMSE and MAE. Experiments were carried out on thirty datasets in the realm of multiple target regression. In the first section, the efficiency of the suggested technique was examined by comparing MTR and ST methods using SVM regression based on different feature sizes. The performance of the proposed technique is assessed in the second portion utilizing different regression methods (i.e., ET, DT, and RF) based on the best feature subset determined by feature selection algorithms. The SVM regressor was utilised with the LIBSVM software package's default parameter values and a standard nonlinear kernel [2].

MTR is a subset of the larger subject of predictive problems with multiple structured outputs. MTR, as opposed to multi-label classification (MLC) and hierarchical classification (HC), works with continuous targets. Because MTR is concerned with organized responses, the numerical objectives in these tasks exhibit underlying correlations or dependencies. Formally, an MTR job can be described as the creation of a group of functions or a single function h that models the relationships between a set of input variables X and a target vector Y in a set of N instances. The input (X) and output (Y) sets are made up of m and d variables, accordingly [1]. The following equation describes the previously discussed prediction problem [1]

$$h: X1..m \rightarrow Y1..d \tag{3}$$

To handle this type of difficulty, a basic uncomplicated technique is to treat each target individually, solving multiple single target (ST) problems that share the same input characteristics. This method has been utilised in certain circumstances, although it is generally used as a baseline

method to compare to more advanced methods. Indeed, when correlations between outputs exist, examining them during modelling should result in lower prediction errors.

Table 1 shows an ensemble algorithm utilised in the TPOT and those will be also integrated in new proposed approaches. These algorithms were written in the Python programming language and implemented using specific hyperparameter settings. These regression techniques were implemented into the systems and found widespread use in MTR settings.

Table 1
Twelve algorithms used in TPOT AutoML system

ElasticNetCV	ExtraTreesRegressor	GradientBoostingRegressor	AdaBoostRegressor	DecisionTreeRegressor	KNeighborsRegressor
LassoLarsCV	LinearSVR	RandomForestRegressor	RidgeCV	XGBRegressor	SGDRegressor

In this experiment, two regression strategies were used: one with TPOT-MTR and the other with SK-Learn's multioutput regressor. The algorithms utilised in the multioutput regressor were random forest dubbed MOR-RFR, which is an ensemble of decision trees, and linear SVM named MOR-LSVR. The studies were conducted in conjunction with the local MTR methods: Support Vector Machine (SVM) and Random Forest (RF). Both algorithms were written in Python and ran on the Google Collaboratory (Colab) with the default hyperparameter values. Apart from the suggested TPOT-MTR techniques, these regression algorithms were chosen due to their broad applicability in MTR settings.

2.3 Performance Metrics

An accuracy, R^2 is utilised to transform the correlation coefficient into accuracy measurements since it is a key indicator of how well the technique is working [16]. The root mean square error (RMSE) for n different predictions represented by is one of the most used metrics as an error function [8].

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}{n}} \tag{4}$$

$$MAE = \frac{\sum_{i=1}^{n}|\hat{y}_i - y_i|}{n} \tag{5}$$

where n represents the total number of tests, i represents the i^{th} experiment, y_i is the actual value of the speed prediction, and $\hat{y}_i$ is the predicted value. Average Relative Root Mean Square Error (aRRMSE) was used to assess the MTR techniques. We used a 10-fold cross-validation as our sampling approach. The following equation determines the RRMSE (Relative Root Mean Square Error)

$$RRMSE = \sqrt{\frac{\sum_{k=1}^{N}\left(y^k - \hat{y}^k\right)^2}{\sum_{k=1}^{N}\left(y^k - \bar{y}^k\right)^2}} \tag{6}$$

where y, $\hat{y}$, and $\bar{y}$ indicate the target's real, predicted, and mean values, respectively, and N denotes the number of assessed cases. This metric has been employed as a common way to assess prediction error in a few MTR and ST regression studies [1]. The RRMSE computed for each of the problem's d targets is averaged to provide the average RRMSE (aRRMSE).

2.4 Dataset

Dataset generator was used to generate synthetic datasets based on three hyperparameters: the number of instances (N), features (m), and targets (d). Additionally, a percentage of instances to be degraded by noise () and the number of generation groups (g) are inputs into the algorithm. Random coefficients (X) were generated utilising a uniform distribution. Initially, the desired d targets are generated using unique random linear coefficients for each of the g groups. These groups, termed generation groups here, correspond to randomly defined partitions within the data domain that share the same input-to-output linear mapping properties. Multiple generation groups can add nonlinearity to the generated datasets despite the use of only linear equations to describe the input-output relationships. The instance-wise length of each of these partitions is defined as a random proportion of N. After generating the input data (X) and an initial set of mutually uncorrelated targets (Y), percent of the input cases are degraded with noise to increase the prediction task complexity. 33 datasets were constructed with the following parameters to restrict an initial narrow scope [1]. The instance count, N was set at five hundred. So “N = 500”. "45" and "90" were alternated as the value of m. The “g” switched between the numbers "1" and "2". The d either read "3" or "6." For each target (y1, y2, y3, y4, y5, y6), linear, quadratic, and cubic functions were used alone or in combination. In each created dataset, noise affected 1% of the cases in total [1]. We will investigate more parameter options in future developments. The produced datasets are described in Table 2.

3. Results

3.1 Code Availability

The code and result are publicly available at https://github.com/hanafimajid/tpot-mtr. Based on the Table 2, the mean value of aRRMSE** is 0.065 generated by TPOT-MTR while the mean of aRRMSE in Mastelini *et al.,* [1] cumulating of 0.691. When the regressors are examined separately, the prediction performance of each technique changes. Cases 1, 2, 9, 10, for example, showed R2 improvements of 99% when utilising the TPOT-MTR. The same could not be said for MOR-RFR and MOR-LSVR. Table 4 indicates that for this dataset, the targets are positive linked for all RMSE in Y1, Y2, Y3, Y4, Y5, and Y6. Because of the observed influence of the regression technique, we decided to address them individually. Table 3 shows the best algorithms for each target and mean duration of about 79.49 minutes taken for each case to learn and generate the best model.

The average time taken for three targets was 56 minutes and 101 minutes for six targets. This is showing that the more target that it has, the more time taken to produce the model. Focusing on the algorithms, the ElasticNetCV and LassoLarsCV contributing 18 and 6 for three targets. There are 23 ElasticNetCV used as the best pipeline for six targets. Average R2 for MOR-RFR and MOR-LSVR is 0.368 and 0.508, this is due to powerful of random forest in regressing the training in comparison with linear SVM. Case twenty-five having the highest value of 0.822 and the lowest R2 value is for case 7 which counted at 0.044 as show in Table 4.

Table 2
Performance result for TPOT-MTR. m represents the number of features and g the number of generating groups

Dataset	m	g	y1	y2	y3	y4	y5	y6	aRRMSE*	aRRMSE**	MAE**	R2**
Case 1	90	1	y1	y2	y3	NA	NA	NA	0.446	0.009	0.007	0.997
Case 2	90	1	y1	y1+y2	y1+y3	NA	NA	NA	0.446	0.008	0.006	0.998
Case 3	90	1	y1	$y1^2$	$y1^3$	NA	NA	NA	0.743	0.014	0.010	0.985
Case 4	90	1	y1	$y1^2$+$y2^2$	$y1^2$+$y3^2$	NA	NA	NA	0.84	0.108	0.076	0.473
Case 5	90	2	y1	y2	y3	NA	NA	NA	0.794	0.112	0.078	0.467
Case 6	90	2	y1	y1+y2	y1+y3	NA	NA	NA	0.785	0.115	0.083	0.566
Case 7	90	2	y1	$y1^2$	$y1^3$	NA	NA	NA	0.864	0.103	0.063	0.388
Case 8	90	2	y1	$y1^2$+$y2^3$	$y1^2$+$y3^2$	NA	NA	NA	0.937	0.132	0.082	0.348
Case 9	45	1	y1	y2	y3	NA	NA	NA	0.309	0.005	0.004	0.997
Case 10	45	1	y1	y1+y2	y1+y3	NA	NA	NA	0.305	0.001	0.001	0.998
Case 11	45	1	y1	$y1^2$	$y1^3$	NA	NA	NA	0.637	0.013	0.005	0.971
Case 12	45	1	y1	$y1^2$+$y2^2$	$y1^2$+y^2	NA	NA	NA	0.744	0.020	0.007	0.974
Case 13	45	2	y1	y2	y3	NA	NA	NA	0.786	0.096	0.074	0.512
Case 14	45	2	y1	y1+y2	y1+y3	NA	NA	NA	0.748	0.093	0.070	0.562
Case 15	45	2	y1	$y1^2$	$y1^3$	NA	NA	NA	0.875	0.106	0.067	0.434
Case 16	45	2	y1	$y1^2$+$y2^2$	$y1^2$+$y3^2$	NA	NA	NA	0.92	0.126	0.083	0.064
Case 17	90	1	y1	y2	y3	y4	y5	y6	0.44	0.022	0.010	0.980
Case 18	90	1	y1	y2	y3	y1+y2	y1+y3	y2+y3	0.443	0.020	0.008	0.981
Case 19	90	1	y1	y2	y3	$y1^2$	$y2^2$	$y3^2$	0.733	0.028	0.011	0.957
Case 20	90	1	y1	y2	y3	$y1^2$+$y2^2$	$y1^2$+$y3^2$	$y2^2$+$y3^2$	0.732	0.031	0.015	0.947
Case 21	90	2	y1	y2	y3	y4	y5	y6	0.817	0.112	0.088	0.423
Case 22	90	2	y1	y2	y3	y1+y2	y1+y3	y2+y3	0.801	0.115	0.091	0.524
Case 23	90	2	y1	y2	y3	$y1^2$	$y2^2$	$y3^2$	0.916	0.140	0.091	0.321
Case 24	90	2	y1	y2	y3	$y1^2$+$y2^2$	$y1^2$+$y3^2$	$y2^2$+$y3^2$	0.919	0.165	0.116	0.282
Case 25	45	1	y1	y2	y3	y4	y5	y6	0.313	0.023	0.006	0.980
Case 26	45	1	y1	y2	y3	y1+y2	y1+y3	y2+y3	0.311	0.028	0.007	0.970
Case 27	45	1	y1	y2	y3	$y1^2$	$y2^2$	$y3^2$	0.644	0.026	0.010	0.965
Case 28	45	1	y1	y2	y3	$y1^2$+$y2^2$	$y1^2$+$y3^2$	$y2^2$+$y3^2$	0.633	0.028	0.008	0.953
Case 29	45	2	y1	y2	y3	y4	y5	y6	0.789	0.085	0.052	0.554
Case 30	45	2	y1	y2	y3	y1+y2	y1+y3	y2+y3	0.805	0.078	0.049	0.660
Case 31	45	2	y1	y2	y3	$y1^2$	$y2^2$	$y3^2$	0.924	0.079	0.040	0.516
Case 32	45	2	y1	y2	y3	$y1^2$+$y2^2$	$y1^2$+$y3^2$	$y2^2$+$y3^2$	0.917	0.080	0.047	0.047
Case 33	45	1	y1	y2	y3	$y1^3$	$y2^3$	$y3^3$	0.495	0.017	0.006	0.972

*Metric obtained by SVM regressor with the ST strategy and a 10-fold cross-validation
**Metric obtained by TPOT-MTR method and a 10-fold cross-validation

Table 3
Best pipeline algorithm for each target with duration

Dataset	1st Best Pipeline	2nd Best Pipeline	3rd Best Pipeline	4th Best Pipeline	5th Best Pipeline	6th Best Pipeline	Duration (min)
Case 1	LassoLarsCV	ElasticNetCV	ElasticNetCV	NA	NA	NA	66
Case 2	LassoLarsCV	LinearSVR	ElasticNetCV	NA	NA	NA	72
Case 3	LassoLarsCV	RandomForestRegressor	AdaBoostRegressor	NA	NA	NA	60
Case 4	ElasticNetCV	ElasticNetCV	ElasticNetCV	NA	NA	NA	120
Case 5	ElasticNetCV	ElasticNetCV	ElasticNetCV	NA	NA	NA	57
Case 6	ElasticNetCV	ElasticNetCV	ElasticNetCV	NA	NA	NA	60
Case 7	ElasticNetCV	ExtraTreesRegressor	AdaBoostRegressor	NA	NA	NA	60
Case 8	ElasticNetCV	ExtraTreesRegressor	RandomForestRegressor	NA	NA	NA	78
Case 9	LinearSVR	LinearSVR	ElasticNetCV	NA	NA	NA	35
Case 10	LinearSVR	LassoLarsCV	RidgeCV	NA	NA	NA	31
Case 11	LinearSVR	XGBRegressor	AdaBoostRegressor	NA	NA	NA	48
Case 12	LinearSVR	XGBRegressor	ElasticNetCV	NA	NA	NA	40
Case 13	ElasticNetCV	ElasticNetCV	ExtraTreesRegressor	NA	NA	NA	52
Case 14	ElasticNetCV	ElasticNetCV	LassoLarsCV	NA	NA	NA	55
Case 15	ElasticNetCV	RandomForestRegressor	AdaBoostRegressor	NA	NA	NA	60
Case 16	ElasticNetCV	KNeighborsRegressor	ExtraTreesRegressor	NA	NA	NA	50
Case 17	ElasticNetCV	LassoLarsCV	LassoLarsCV	ElasticNetCV	LassoLarsCV	RidgeCV	120
Case 18	ElasticNetCV	LassoLarsCV	LassoLarsCV	ElasticNetCV	LassoLarsCV	LassoLarsCV	114
Case 19	ElasticNetCV	LassoLarsCV	LassoLarsCV	DecisionTreeRegressor	ExtraTreesRegressor	XGBRegressor	114
Case 20	ElasticNetCV	LassoLarsCV	LassoLarsCV	RandomForestRegressor	LassoLarsCV	ElasticNetCV	114
Case 21	LassoLarsCV	ElasticNetCV	RandomForestRegressor	LassoLarsCV	RidgeCV	RidgeCV	138
Case 22	LassoLarsCV	ElasticNetCV	RandomForestRegressor	LassoLarsCV	LassoLarsCV	LassoLarsCV	144
Case 23	LassoLarsCV	ElasticNetCV	RandomForestRegressor	ExtraTreesRegressor	LassoLarsCV	ElasticNetCV	180
Case 24	LassoLarsCV	ElasticNetCV	RandomForestRegressor	DecisionTreeRegressor	LassoLarsCV	XGBRegressor	138
Case 25	LinearSVR	RidgeCV	RidgeCV	LassoLarsCV	LassoLarsCV	ElasticNetCV	78
Case 26	LinearSVR	RidgeCV	RidgeCV	LassoLarsCV	LassoLarsCV	LassoLarsCV	69
Case 27	LinearSVR	RidgeCV	RidgeCV	XGBRegressor	XGBRegressor	KNeighborsRegressor	56
Case 28	LinearSVR	RidgeCV	RidgeCV	DecisionTreeRegressor	LassoLarsCV	KNeighborsRegressor	84
Case 29	ElasticNetCV	ElasticNetCV	ExtraTreesRegressor	ElasticNetCV	ElasticNetCV	RidgeCV	78
Case 30	ElasticNetCV	ElasticNetCV	ExtraTreesRegressor	LassoLarsCV	LassoLarsCV	LassoLarsCV	78
Case 31	ElasticNetCV	ElasticNetCV	ExtraTreesRegressor	ExtraTreesRegressor	XGBRegressor	DecisionTreeRegressor	96
Case 32	ElasticNetCV	ElasticNetCV	ExtraTreesRegressor	ExtraTreesRegressor	LassoLarsCV	AdaBoostRegressor	66
Case 33	LinearSVR	RidgeCV	RidgeCV	AdaBoostRegressor	DecisionTreeRegressor	KNeighborsRegressor	60

Table 4
Accuracy and error metrics for regular MTR of random forest and linear SVM

Dataset	MOR-RFR(R2)	MOR-RFR(RMSE)	MOR-RFR(MAE)	MOR-LSVR(R2)	MOR-LSVR(RMSE)	MOR-LSVR(MAE)
Case 1	0.709	0.100	0.077	0.988	0.020	0.016
Case 2	0.689	0.104	0.080	0.987	0.021	0.016
Case 3	0.523	0.096	0.066	0.448	0.100	0.050
Case 4	0.312	0.148	0.104	0.206	0.156	0.102
Case 5	0.290	0.131	0.095	0.430	0.116	0.041
Case 6	0.305	0.145	0.116	0.491	0.125	0.044
Case 7	0.044	0.138	0.093	0.210	0.129	0.062
Case 8	0.091	0.157	0.115	0.033	0.161	0.093
Case 9	0.613	0.102	0.081	0.999	0.013	0.010
Case 10	0.633	0.099	0.080	0.991	0.016	0.012
Case 11	0.324	0.108	0.071	0.473	0.092	0.041
Case 12	0.216	0.134	0.097	0.295	0.125	0.067
Case 13	0.362	0.120	0.088	0.480	0.100	0.043
Case 14	0.322	0.120	0.090	0.532	0.096	0.073
Case 15	0.239	0.129	0.089	0.246	0.128	0.064
Case 16	0.077	0.136	0.088	0.113	0.134	0.060
Case 17	0.581	0.110	0.086	0.983	0.023	0.002
Case 18	0.532	0.112	0.090	0.983	0.021	0.002
Case 19	0.226	0.126	0.092	0.388	0.106	0.049
Case 20	0.212	0.128	0.091	0.382	0.108	0.052
Case 21	0.221	0.132	0.099	0.299	0.123	0.094
Case 22	0.271	0.142	0.107	0.463	0.122	0.094
Case 23	0.111	0.164	0.112	0.085	0.167	0.102
Case 24	0.100	0.191	0.136	0.116	0.194	0.126
Case 25	0.822	0.071	0.054	0.980	0.024	0.003
Case 26	0.792	0.077	0.060	0.970	0.029	0.004
Case 27	0.424	0.108	0.082	0.379	0.109	0.054
Case 28	0.399	0.100	0.076	0.410	0.095	0.044
Case 29	0.371	0.109	0.076	0.513	0.091	0.034
Case 30	0.427	0.107	0.079	0.618	0.084	0.031
Case 31	0.148	0.116	0.071	0.233	0.106	0.044
Case 32	0.144	0.126	0.086	0.243	0.117	0.058
Case 33	0.627	0.063	0.043	0.806	0.031	0.013

RMSE value for Y1, Y2, Y3, Y4, Y5 and Y6 showing a strong positive correlation with the range of 0.69 to 0.98. This is due to the significant correlation among of the targets produced by the TPOT-MTR approaches as shown in Table 5. On the other hand, R2 showing negative strongest correlation with the RMSE, and this is caused by the logical relationship in between both that conversely proportionate. The more accurate the model is, the least value of error it has as shown in Table 6. While the Time having positive correlation of 0.4 with the RMSE, which means that if the RMSE increased by 1.0, the Time also will be increased by 0.4 minutes. R2 is having weakest negative correlation with the Time and seems not affected by the value if both ups and downs.

Table 5
Correlation in between error metrics in each target

	RMSE(Y1)	*RMSE(Y2)*	*RMSE(Y3)*	*RMSE(Y4)*	*RMSE(Y5)*	*RMSE(Y6)*
RMSE(Y1)	1					
RMSE(Y2)	0.86	1				
RMSE(Y3)	0.69	0.86	1			
RMSE(Y4)	0.88	0.88	0.87	1		
RMSE(Y5)	0.92	0.90	0.88	0.98	1	
RMSE(Y6)	0.90	0.92	0.92	0.92	0.93	1

Table 6
Correlation in between RMSE, duration and accuracy for TPOT-MTR

	RMSE	*Time(min)*	*R2*
RMSE	1		
Time(min)	0.40	1	
R2 (TPOT-MTR)	-0.87	-0.20	1

Figure 3 below is showing result of the accuracy, R^2 TPOT-MTR, MOR-RFR and MOR-LSVR. It looks like a plot of the performance results for different TPOT-MTR, MOR-RFR, and MOR-LSVR models. The accuracy of each model is displayed along the y-axis, while the numerous iterations of datasets that were utilized throughout the evaluation process are likely represented along the x-axis. According to the narrative, TPOT-MTR achieves the best results in terms of accuracy, with MOR-LSVR coming in a close second. It would indicate that MOR-RFR has a less accurate predictive capability than the other two models. In addition, there is a measurement of R^2 included in the plot. R^2 is a typical metric that is utilised when evaluating the goodness-of-fit of regression models.

Accuracy is an important way to measure how well a model can predict the outcome of an event, but it may not always tell the whole story about how well a model works. In example, a model with a high level of accuracy may not work well on certain subsets of the data or may be more likely to overfit. Also, accuracy might not be the best measure for all kinds of tasks. For example, in some situations, it may be better to judge a model's performance by its precision, recall, or F1 score (as shown in section 3.2). When judging an AutoML model's performance, it is best to use a mix of different performance metrics such as the error rated of RMSE and MAE (as shown in Table 7). This can give a more complete picture of how well the model works and help find possible weaknesses or places to improve.

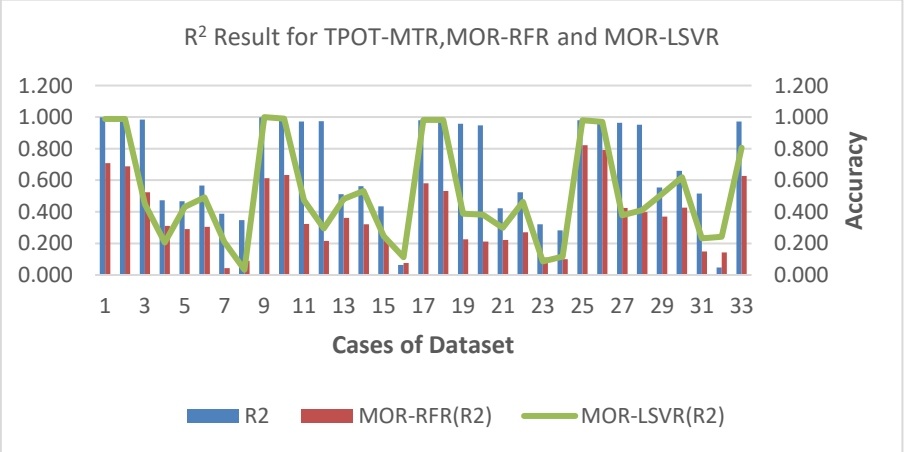

Fig. 3. Accuracy for TPOT-MTR, MOR-RFR and MOR-LSVR

TPOT-MTR presented highest value of R2 for Case 1, Case 2, Case 3, Case 9, Case 10, Case 11, Case 12, Case 17, Case 18, Case 19, Case 20, Case 25, Case 26, Case 27, Case 28, and Case 33. In datasets such as Case 1, Case 2, Case 9, Case 10, Case 17, Case 18, Case 25, case 26 have a good accuracy for MOR-LSVR as shown in Table 2, this is due to linearity of the input data and its better performance for the linear data. For the non-linear input data, the TPOT-MTR gives better accuracy because of the advantages of genetic algorithm concepts which provide better performance for the non-linear data. Figure 4 to Figure 33 showing the actual and predicted value for multiple target regression with the one hundred data in x-axis and result of y with the maximum of 1.0 in y-axis. Having too many outliers as shown in Figure 13, Figure 14, Figure 19, Figure 22, Figure 27, and Figure 28 requiring further investigation on the generated model that causing huge residual value in between actual and predicted. For Case 16, Case 29, and Case 31 having R2 value of 0.06, 0.55 and 0.51 and making it consistent and relevant with the output value.

Hyperparameters for TPOT-MTR is defined as follow

generations=5, population_size=50, verbosity=2, random_state=39, n_jobs=1, max_time_mins=None, max_eval_time_mins=5, cv=10. TPOT will evaluate 250 (5 generations x 50 population_size) pipeline configurations before finishing. In a single grid search, around 2500 models are fitted and assessed on the training data. This corresponds to 250 model configurations to evaluate if 10-fold cross-validation is used. This is a time-consuming process even for relatively straightforward models like decision trees. However, there are datasets where a thorough dive is still needed to investigate why the difference in time between three targets and six targets is so tiny. To fully understand the motivation for this, additional research may be necessary.

The y-axis for Figure 4 to Figure 33 is showing the value of the target y and the x-axis for Figure 4 to Figure 33 is showing the value of testing percentage using the TPOT-MTR.

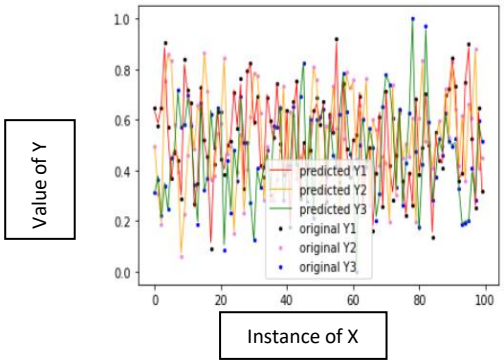

Fig. 4. Actual and predicted value for 500s90f3t1g0.01py1-y1+y2-y1+y3

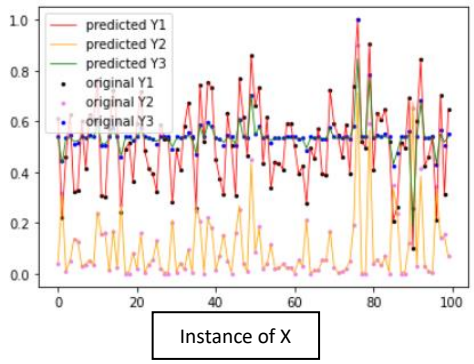

Fig. 5. Actual and predicted value for 500s45f3t1g0.01py1-y1e2-y1e3

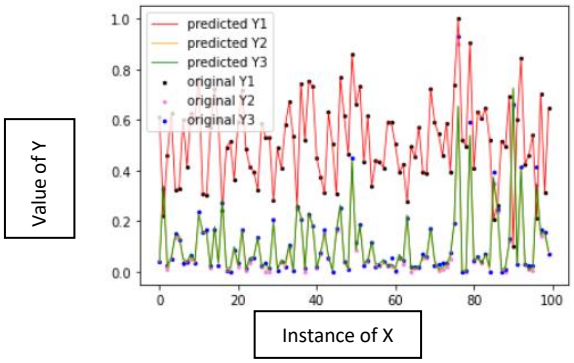

Fig. 6. Actual and predicted value for 500s45f3t1g0.01py1-y1e2+y2e2-y1e2+y3e2

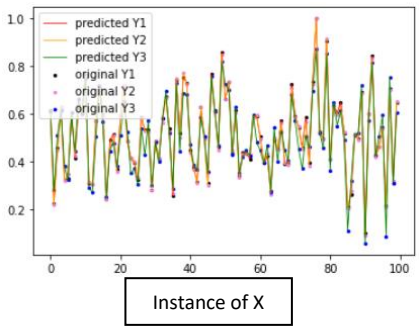

Fig. 7. Actual and predicted value for 500s45f3t1g0.01py1-y1+y2-y1+y3

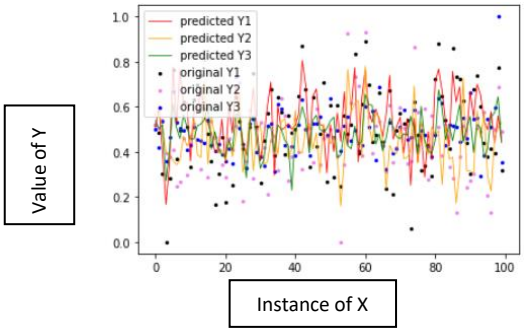

Fig. 8. Actual and predicted value for 500s45f3t2g0.01py1-y1+y2-y1+y3

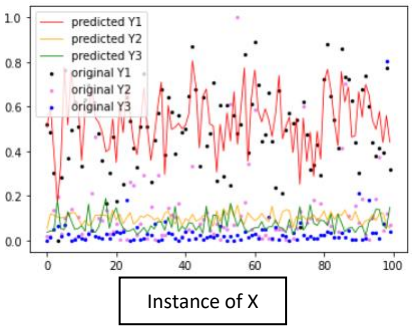

Fig. 9. Actual and predicted value for 500s45f3t2g0.01py1-y1e2+y2e2-y1e2+y3e2

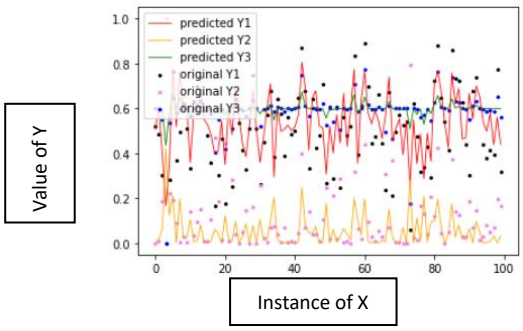

Fig. 10. Actual and predicted value for 500s45f3t2g0.01py1-y1e2-y1e3

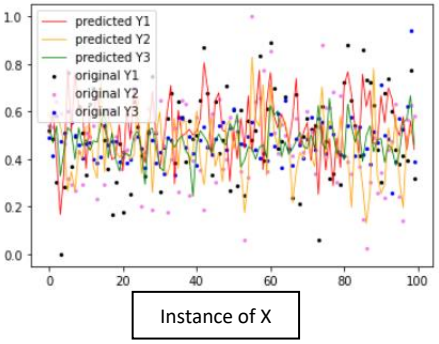

Fig. 11. Actual and predicted value for 500s45f3t2g0.01py1-y1+y2-y1+y3

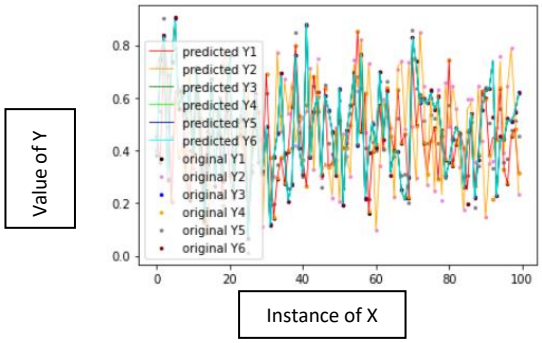

Fig. 12. Actual and predicted value for 500s45f6t1g0.01py1+y2-y1+y3-y2+y3

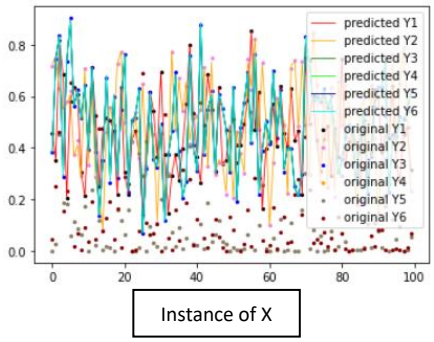

Fig. 13. Actual and predicted value for 500s45f6t1g0.01py1e2+y2e2-y1e2+y3e2-y2e2+y3e2

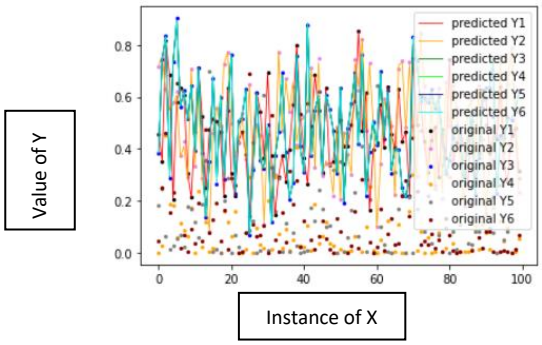

Fig. 14. Actual and predicted value for 500s45f6t1g0.01py1e2-y2e2-y3e2

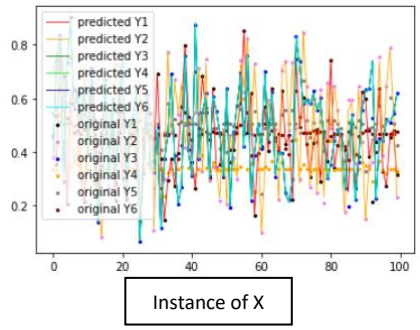

Fig. 15. Actual and predicted value for 500s45f6t1g0.01py1e3-y2e3-y3e3

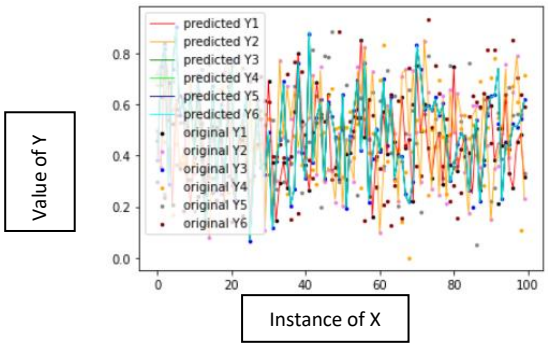

Fig. 16. Actual and predicted value for 500s45f6t1g0.01py1-y2-y3

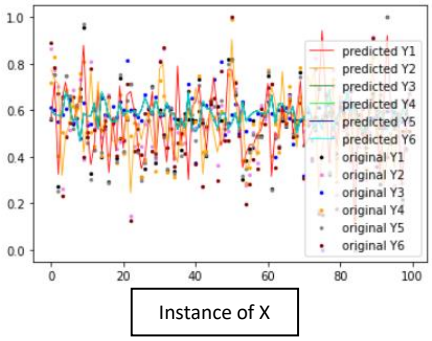

Fig. 17. Actual and predicted value for 500s45f6t1g0.01py1e3-y2e3-y3e3

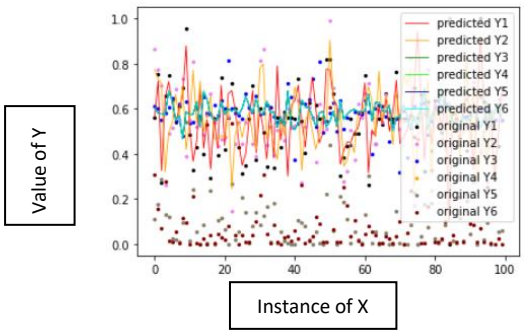

Fig. 18. Actual and predicted value for 500s45f6t2g0.01py1e2+y2e2-y1e2+y3e2-y2e2+y3e2

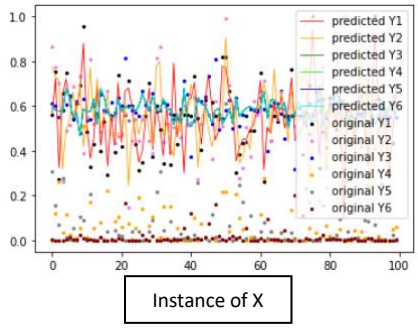

Fig. 19. Actual and predicted value for 500s45f6t2g0.01py1e2-y2e2-y3e2

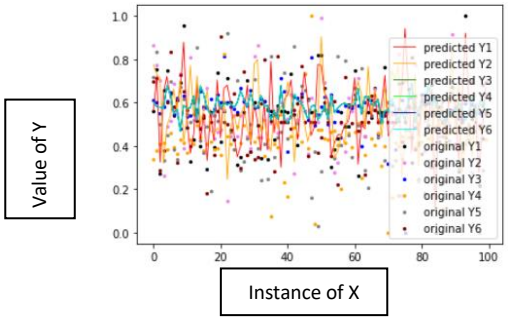

Fig. 20. Actual and predicted value for 500s45f6t2g0.01py1-y2-y3

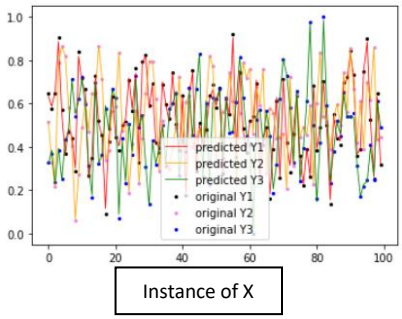

Fig. 21. Actual and predicted value for 500s90f3t1g0.01py1y2y3

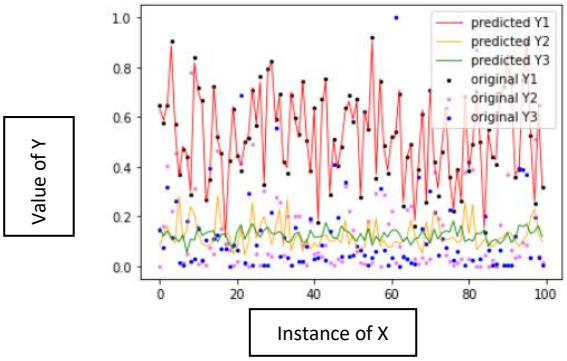

Fig. 22. Actual and predicted value for 500s90f3t1g0.01py1-y1e2+y2e2-y1e2+y3e2

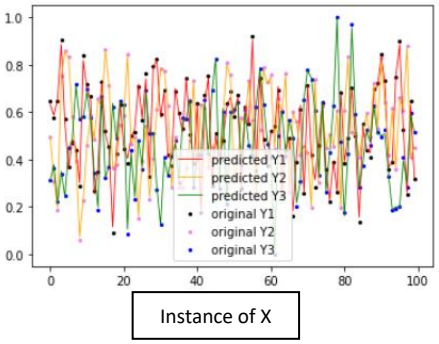

Fig. 23. Actual and predicted value for 500s90f3t1g0.01py1-y1+y2-y1+y3

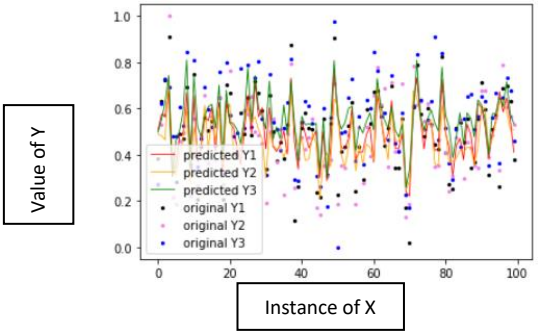

Fig. 24. Actual and predicted value for 500s90f3t2g0.01py1-y1+y2-y1+y3

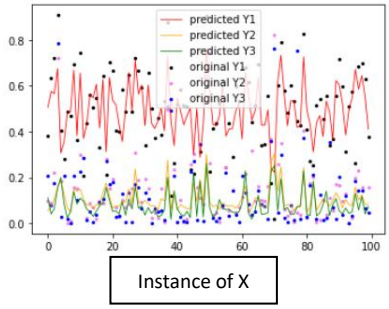

Fig. 25. Actual and predicted value for 500s90f3t1g0.01py1-y1e2+y2e2-y1e2+y3e2

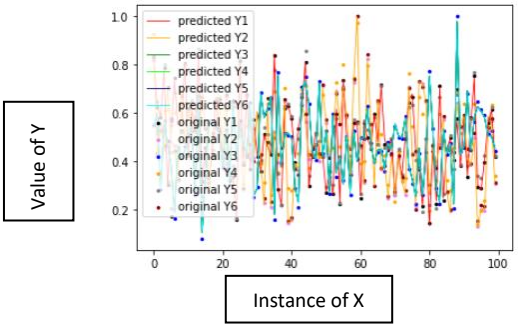

Fig. 26. Actual and predicted value for 500s90f6t1g0.01py1+y2-y1+y3-y2+y3

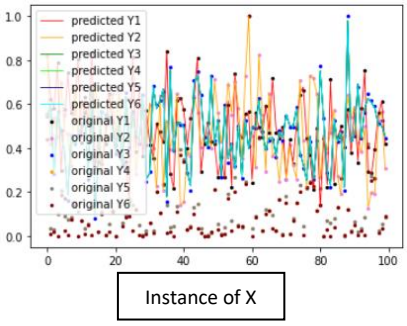

Fig. 27. Actual and predicted value for 500s90f6t1g0.01py1e2+y2e2-y1e2+y3e2-y2e2+y3e2

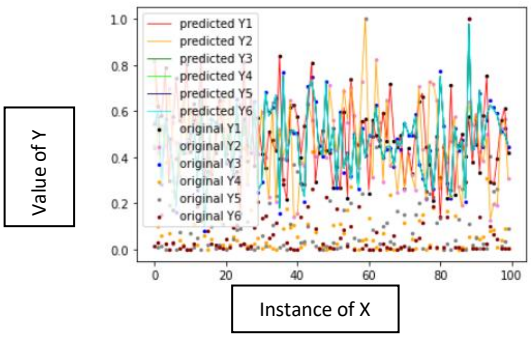

Fig. 28. Actual and predicted value for 500s90f6t1g0.01py1e2-y2e2-y3e2

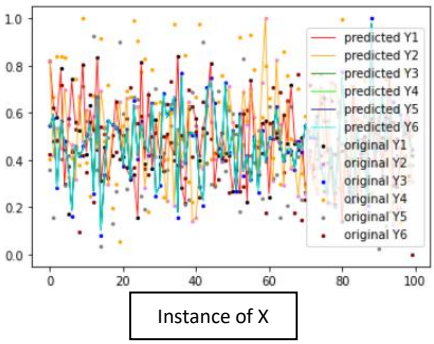

Fig. 29. Actual and predicted value for 500s90f6t1g0.01py1-y2-y3

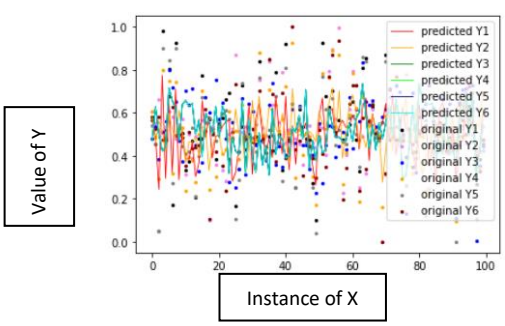

Fig. 30. Actual and predicted value for 500s90f6t2g0.01py1+y2-y1+y3-y2+y3

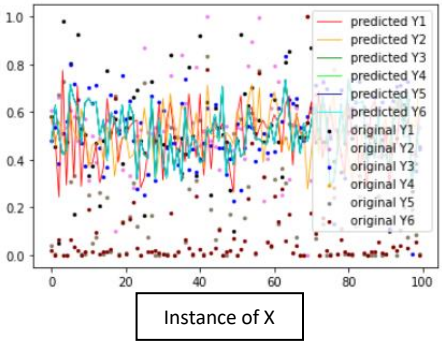

Fig. 31. Actual and predicted value for 500s90f6t2g0.01py1e2-y2e2-y3e2

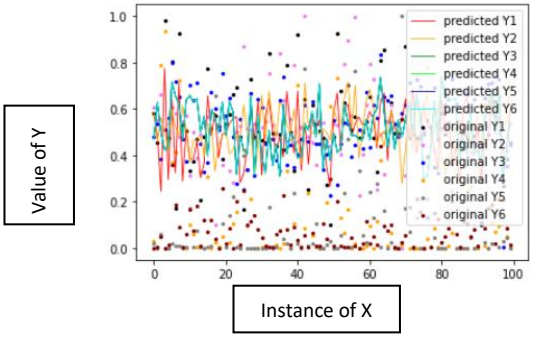

Fig. 32. Actual and predicted value for 500s90f6t2g0.01py1e2+y2e2-y1e2+y3e2-y2e2+y3e2

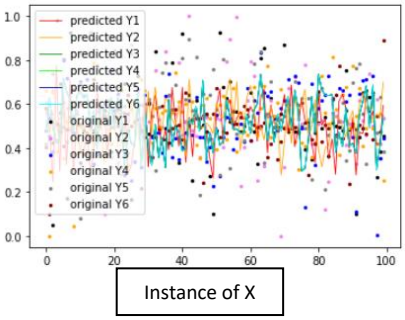

Fig. 33. Actual and predicted value for 500s90f6t2g0.01py1-y2-y3

Figure 34 to Figure 42 displays the Pearson correlation representation graphics for different datasets according to the generating functions used to construct them so that the behaviour of each target can be seen in more detail. In instance eighteen of datasets, most of the targets exhibit high intra-target correlation coefficients. Other instances include examples 3, 20, and 33, which show the presence of both high and inter-target correlation coefficients. Finally, targets 2, and 17 have weakly positive correlations. We included a summary of the correlation coefficient in the heatmaps for each simulated example. Bright colours indicate great correlation, whereas dark hues indicate instances when the targets are uncorrelated.

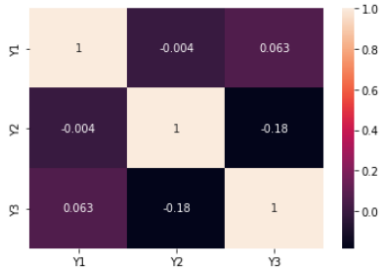

Fig. 34. Pearson correlation heatmap for 500s90f3t1g0.01py1y2y3

Fig. 35. Pearson correlation heatmap for 500s90f3t1g0.01py1-y1+y2-y1+y3

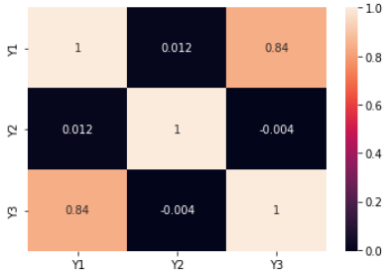

Fig. 36. Pearson correlation heatmap for 500s90f3t1g0.01py1-y1e2-y1e3

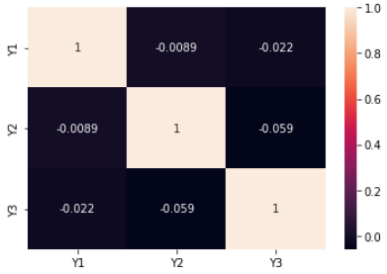

Fig. 37. Pearson correlation heatmap for 500s90f3t1g0.01py1-y1e2+y2e2-y1e2+y3e2

Fig. 38. Pearson correlation heatmap for 500s90f6t1g0.01py1-y2-y3

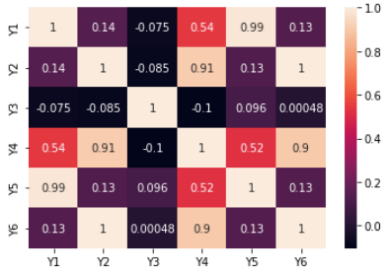

Fig. 39. Pearson correlation heatmap for 500s90f6t1g0.01py1+y2-y1+y3-y2+y3

Fig. 40. Pearson correlation heatmap for 500s90f6t1g0.01py1e2-y2e2-y3e2

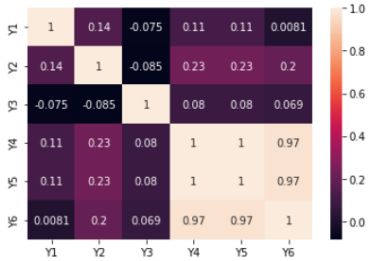

Fig. 41. Pearson correlation heatmap for 500s90f6t1g0.01py1e2+y2e2-y1e2+y3e2-y2e2+y3e2

Fig. 42. Pearson correlation heatmap for 500s45f6t1g0.01py1e3-y2e3-y3e3

3.2 F-Test Result

Table 7 displays the 33 dataset's mean absolute error (MAE) and root mean squared error (RMSE) comparison between TPOT-MTR and TPOT-MO. Data analysis in Microsoft Excel shows the F-Test result for MAE and RMSE between TPOT-MTR and TPOT-MO in Table 8 and Table 9. The F-Test result is interpreted using the research by Xiong *et al.,* [17] and the following hypothesis is made to test their correlation.

H_0: No significant different in between TPOT-MTR and TPOT-MO result
H_1: Significant different in between TPOT-MTR and TPOT-MO result

According to Xiong *et al.,* [17], if F > F Critical, the H_o null hypothesis can be rejected. Both the root-mean-squared error (RMSE) and the mean-squared error (MAE) indicate statistically significant differences between TPOT-MTR and TPOT-MO (p-values for both are less than 0.05). The lower the RMSE and MAE values are, the more accurate the model is, and the better the fit.

Table 7
MAE and RMSE result for TPOT-MTR and TPOT-MO

Case	RMSE(TPOT-MTR)	RMSE (TPOT-MO)	MAE (TPOT-MTR)	MAE (TPOT-MO)
1	0.011	0.099	0.009	0.076
2	0.009	0.103	0.006	0.080
3	0.015	0.095	0.010	0.065
4	0.130	0.148	0.076	0.105
5	0.112	0.131	0.078	0.096
6	0.115	0.144	0.083	0.115
7	0.108	0.136	0.063	0.092
8	0.134	0.157	0.082	0.115
9	0.008	0.100	0.004	0.079
10	0.004	0.100	0.003	0.080
11	0.016	0.109	0.005	0.070
12	0.024	0.135	0.007	0.097
13	0.097	0.120	0.074	0.088
14	0.094	0.120	0.070	0.090
15	0.110	0.129	0.067	0.089
16	0.129	0.136	0.083	0.088
17	0.024	0.110	0.010	0.086
18	0.094	0.111	0.008	0.089
19	0.030	0.128	0.011	0.092
20	0.033	0.129	0.015	0.129
21	0.112	0.132	0.088	0.099
22	0.115	0.142	0.091	0.106
23	0.144	0.165	0.091	0.113
24	0.174	0.190	0.116	0.135
25	0.024	0.072	0.006	0.055
26	0.029	0.076	0.007	0.059
27	0.028	0.109	0.010	0.043
28	0.029	0.099	0.008	0.075
29	0.087	0.109	0.052	0.074
30	0.078	0.107	0.049	0.080
31	0.080	0.115	0.040	0.071
32	0.080	0.130	0.047	0.088
33	0.020	0.063	0.006	0.043

Table 8
F-Test Two-Sample of TPOT-MTR and TPOT-MO for Variances of RMSE

	RMSE(TPOT-MTR)	*RMSE (TPOT-MO)*
Mean	0.070	0.118
Variance	0.002	0.001
Observations	33.000	33.000
df	32.000	32.000
F	3.109	
P(F<=f) one-tail	0.001	
F Critical one-tail	1.804	

Table 9
F-Test Two-Sample of TPOT-MTR and TPOT-MO for Variances of MAE

	MAE (TPOT-MTR)	*MAE (TPOT-MO)*
Mean	0.042	0.087
Variance	0.001	0.000
Observations	33.000	33.000
df	32.000	32.000
F	2.866	
P(F<=f) one-tail	0.002	
F Critical one-tail	1.804	

4. Conclusions

This chapter suggested TPOT-MTR, a new dimension of the daisy-chain technique. In comparison to single target (ST) approaches utilised for multiple regression target work, the novel methodology has developed a more compact and powerful multiple target regression based on genetic algorithm idea. As a result, regression accuracy improved much further. The experimental results validated the TPOT-MTR methods' efficacy. The results revealed that the TPOT-MTR outperformed the ST in most situations. TPOT-MTR successfully improves regression aRRMSE by 91%. TPOT-MTR, on the other hand, was much more successful than ST at reducing dimensionalities across all datasets. More tests are required to validate this complex flow structure.

Moreover, TPOT-MTR was found to be much more effective than the single target approach at reducing dimensionalities across all datasets, which is a significant advantage when working with large datasets. Although the results are promising, more tests are required to validate the efficacy of this complex flow structure. TPOT-MTR has limitations and potential disadvantages, like any other methodology. Here are some of TPOT-MTR's limitations: Computationally Demanding: Multiple phases of optimization utilizing the genetic algorithm are required by the TPOT-MTR methodology, which can be computationally intensive and time-consuming. This can be an issue for large datasets or real-time applications where performance is essential. Since TPOT-MTR employs multiple regression algorithms and hyperparameters, it can be difficult to interpret the final model and comprehend how each input variable affects the output.

This lack of interpretability can be detrimental in situations where explain ability is essential, such as medical or legal applications. Limited Scope: TPOT-MTR is designed exclusively for multiple target regression problems and may not be applicable to classification or clustering problems. Overall, TPOT-MTR represents a significant advancement in the field of multiple target regression, but it is essential to consider its limitations and potential disadvantages when assessing its applicability to a specific problem. In addition, TPOT-MTR represents a significant advancement in the field of multiple target regression and has the potential to revolutionize the way we approach regression problems. Additional research might be conducted in other areas, such as renewable energy, which has shown that Malaysia continues to rely on non-renewable energy [17]. In order to encourage the consumption of renewable energy, integration between AutoML and other systems could be carried out.

Acknowledgement
This research was not funded by any grant.

References

[1] Mastelini, Saulo Martiello, Everton Jose Santana, Victor Guilherme Turrisi da Costa, and Sylvio Barbon. "Benchmarking multi-target regression methods." In *2018 7th Brazilian Conference on Intelligent Systems (BRACIS)*, pp. 396-401. IEEE, 2018.

[2] Nantasenamat, Chanin, Saksiri Jamsak, Likit Preeyanon, Chartchalerm Isarankura-Na-Ayudhya, and Virapong Prachayasittikul. "AutoWeka User Guide (Version 1.0)." *AutoWeka*, 2012. https://www.mt.mahidol.ac.th/autoweka/AutoWeka-1.0-user-guide.pdf.

[3] Lingitz, Lukas, Viola Gallina, Fazel Ansari, Dávid Gyulai, András Pfeiffer, Wilfried Sihn, and László Monostori. "Lead time prediction using machine learning algorithms: A case study by a semiconductor manufacturer." *Procedia CIRP* 72 (2018): 1051-1056. https://doi.org/10.1016/j.procir.2018.03.148

[4] Bender, Janek, Martin Trat, and Jivka Ovtcharova. "Benchmarking automl-supported lead time prediction." *Procedia Computer Science* 200 (2022): 482-494. https://doi.org/10.1016/j.procs.2022.01.246

[5] Harry, Lim Sze Li, Pham Luu Trung Duong, and Nagarajan Raghavan. "Exploration of multi-output Gaussian process regression for residual storage life prediction in lithium ion battery." In *2020 Prognostics and Health Management Conference (PHM-Besançon)*, pp. 263-269. IEEE, 2020. https://doi.org/10.1109/PHM-Besancon49106.2020.00051

[6] Li, Changsheng, Fan Wei, Weishan Dong, Xiangfeng Wang, Qingshan Liu, and Xin Zhang. "Dynamic structure embedded online multiple-output regression for streaming data." *IEEE Transactions on Pattern Analysis and Machine Intelligence* 41, no. 2 (2018): 323-336. https://doi.org/10.1109/TPAMI.2018.2794446

[7] Li, Yang Yuan, Tien Van Do, and Hai T. Nguyen. "A comparison of forecasting models for the resource usage of MapReduce applications." *Neurocomputing* 418 (2020): 36-55. https://doi.org/10.1016/j.neucom.2020.07.059

[8] Nagarajah, Thiloshon, and Guhanathan Poravi. "A review on automated machine learning (AutoML) systems." In *2019 IEEE 5th International Conference for Convergence in Technology (I2CT)*, pp. 1-6. IEEE, 2019. https://doi.org/10.1109/I2CT45611.2019.9033810

[9] Wu, Di, Qinghua Guan, Zhe Fan, Hanhui Deng, and Tao Wu. "AutoML with Parallel Genetic Algorithm for Fast Hyperparameters Optimization in Efficient IoT Time Series Prediction." *IEEE Transactions on Industrial Informatics* (2022). https://doi.org/10.1109/TII.2022.3231419

[10] LeDell, Erin, and Sebastien Poirier. "H2O automl: Scalable automatic machine learning." In *Proceedings of the AutoML Workshop at ICML*, vol. 2020. 2020.

[11] Gedam, P. "Predicting Multiple Columns in a Table (Multi-Label Prediction)." *AutoGluon*. Accessed 2023. https://auto.gluon.ai/stable/tutorials/tabular/advanced/tabular-multilabel.html.

[12] Ferreira, Luís, André Pilastri, Carlos Manuel Martins, Pedro Miguel Pires, and Paulo Cortez. "A comparison of AutoML tools for machine learning, deep learning and XGBoost." In *2021 International Joint Conference on Neural Networks (IJCNN)*, pp. 1-8. IEEE, 2021. https://doi.org/10.1109/IJCNN52387.2021.9534091

[13] Nikitin, Nikolay O., Pavel Vychuzhanin, Mikhail Sarafanov, Iana S. Polonskaia, Ilia Revin, Irina V. Barabanova, Gleb Maximov, Anna V. Kalyuzhnaya, and Alexander Boukhanovsky. "Automated evolutionary approach for the design of composite machine learning pipelines." *Future Generation Computer Systems* 127 (2022): 109-125. https://doi.org/10.1016/j.future.2021.08.022

[14] Olson, Randal S., Nathan Bartley, Ryan J. Urbanowicz, and Jason H. Moore. "Evaluation of a tree-based pipeline optimization tool for automating data science." In *Proceedings of the Genetic and Evolutionary Computation Conference 2016*, pp. 485-492. 2016. https://doi.org/10.1145/2908812.2908918

[15] Nathans, Laura L., Frederick L. Oswald, and Kim Nimon. "Interpreting multiple linear regression: a guidebook of variable importance." *Practical Assessment, Research & Evaluation* 17, no. 9 (2012): n9.

[16] Siegel, Andrew F., and Michael R. Wagner. *Practical Business Statistics (Eighth Edition)*. Academic Press, 2022.

[17] Xiong, Tao, Yukun Bao, and Zhongyi Hu. "Multiple-output support vector regression with a firefly algorithm for interval-valued stock price index forecasting." *Knowledge-Based Systems* 55 (2014): 87-100. https://doi.org/10.1016/j.knosys.2013.10.012