Towards Explainable AutoML Using Error Decomposition

Caitlin A. Owen(✉), Grant Dick, and Peter A. Whigham

Department of Information Science, University of Otago, Dunedin, New Zealand
{caitlin.owen,grant.dick,peter.whigham}@otago.ac.nz

Abstract. The important process of choosing between algorithms and their many module choices is difficult, even for experts. Automated machine learning allows users at all skill levels to perform this process. It is currently performed using aggregated total error, which does not indicate whether a stochastic algorithm or module is stable enough to consistently perform better than other candidates. It also does not provide an understanding of how the modules contribute to total error. This paper explores the decomposition of error for the refinement of genetic programming. Automated algorithm refinement is examined through choosing a pool of candidate modules and swapping pairs of modules to reduce the largest component of decomposed error. It is shown that a pool of candidates that are not examined for diversity in targeting different components of error can provide inconsistent module preferences. Manual algorithm refinement is also examined by choosing refinements based on their well-understood behaviour in reducing a particular error component. The results show that an effective process should exploit both the advantages of targeted improvements identified using a manual process and the simplicity of an automated process by choosing a hierarchy of the most important modules for reducing error components.

Keywords: Genetic programming · Automated machine learning · Algorithm refinement · Symbolic regression

1 Introduction

With the successful application of machine learning algorithms to many problem domains [1–3], there is increasing interest from end users who are not experts in machine learning. These applications of machine learning typically involve a wide range of parameters or algorithm module choices [4,5]. This wide range of choices provide a greater opportunity to produce a good predictive model, given that a particular algorithm will not provide consistently better predictive performance than all other candidates [6]. However, even for data scientists, the process of choosing between a large number of algorithms and choosing appropriate parameters/modules is difficult [7,8]. This is the motivation for automated

H. Aziz et al. (Eds.): AI 2022, LNAI 13728, pp. 177–190, 2022.
https://doi.org/10.1007/978-3-031-22695-3_13

machine learning (AutoML), which involves the data-driven algorithmic selection, composition and parameterisation of machine learning methods, usually in order to minimise prediction error [9,10].

AutoML methods in the literature use only an aggregate measure of prediction error for a continuous response variable (referred to in this paper as total error) to examine combinations of algorithm modules. Performing AutoML using only total error does not provide an understanding of how the modules of the algorithm interact or the role each module plays in reducing total error. Instead, decomposed error will allow an AutoML process to make more accurate and informed decisions, in terms of the compatibility of algorithm module combinations and their parameterisation, by targeting a reduction in the largest component of error.

For a deterministic algorithm, using the same training data for each run produces the same prediction model. These algorithms can be characterised by decomposing error into bias and variance, where the single source of error due to variance is associated with sampling of finite training data [11]. In contrast, a stochastic algorithm involves multiple sources of error due to variance. Therefore, the algorithm can be more fully characterised by splitting error due to variance into error due to external variance (variance due to the training data) and error due to internal variance (variance due to the algorithm itself) using an extended error decomposition [12]. The extended decomposition of error and AutoML are expected to be strongly compatible as they are both empirical processes.

Performing AutoML using "black-box" total error creates ambiguity about which sources of error are being reduced and why a particular combination of algorithm modules has been chosen. In contrast, decomposed error provides an explanation of which error components are being minimised when choosing modules. Therefore, AutoML driven by decomposed error would provide more transparency and an explanation of why a particular combination of modules has been chosen, which are both important for AutoML [13]. This includes an understanding of how the chosen algorithm is appropriate for a given problem. Explainability is important for AutoML because end users need to have confidence and trust in the performance/behaviour of an algorithm [14]; this confidence is gained by understanding why the algorithm modules have been chosen.

In this paper, AutoML driven by an extended decomposition of error is explored by performing algorithm refinement that targets the largest component of error. This process is applied to the refinement of genetic programming (GP) for symbolic regression, although any machine learning algorithm could be refined using this process. Algorithm refinement is first performed using an automated process to determine whether it can be reliable for stochastic algorithms. This method focuses on refining the selection and variation operators of GP, although the method could be used to refine all parts of the GP algorithm. It is shown that error due to internal variance is not sufficiently stable to provide consistent decisions about which module reduces prediction error. This highlights the importance of choosing a diverse set of candidate modules that provide targeted reductions in all components of error, particularly error due

to internal variance. To confirm this, algorithm refinement is performed using a manual process. In each iteration, new algorithm components are hand-picked for their well-understood behaviour in terms of reducing the largest component of error, which is shown to successfully reduce total error.

The remainder of this paper is structured as follows: a brief overview of algorithm refinement methods are discussed in Sect. 2; a description of how to decompose error for the refinement of GP is outlined in Sect. 3; results for an automated algorithm refinement process using decomposed error are discussed in Sect. 3.1 and critiqued in Sect. 3.2; results for a manual algorithm refinement process using decomposed error are discussed in Sect. 3.3 and critiqued in Sect. 3.4; finally, conclusions and future work are discussed in Sect. 4.

2 Algorithm Refinement

A number of different methods have been used for AutoML. The combined algorithm selection and hyperparameter optimisation (CASH) problem can be viewed as a "single hierarchical hyperparameter optimisation problem", with the chosen type of algorithm being considered as a hyperparameter [9, p. 847]. Grid search examines all possible combinations of hyperparameters [15]. While this is a simple method, it is computationally expensive and potentially intractable if the number of hyperparameters is large. Random search improves on grid search by not examining the full distributions of hyperparameter values [15]. However, random search is still potentially computationally expensive. Bayesian optimisation, involving a probability surrogate model of objectives, is more computationally efficient and is applicable to any type of objective function. Auto-WEKA [16] involves Bayesian optimisation using tree-based models. Auto-SKLearn [17] also uses Bayesian optimisation, extending Auto-WEKA in order to provide an initial meta-learning step as well as automated ensemble construction. Evolutionary computation has also been used for AutoML. RECIPE (REsilient ClassifIcation Pipeline Evolution) uses grammar-based GP, with a grammar representing an algorithm pipeline, i.e., a combination of algorithm modules [18]. RECIPE provides a larger number of algorithm modules than both Auto-SKLearn and Auto-WEKA. TPOT (Tree-Based Pipeline Optimization Tool) also uses a variant of GP to represent algorithm pipelines, allowing parallel processing by using multiple copies of a data set [19].

When applied to regression problems, these AutoML methods involve the same basic process. A portfolio of candidate algorithm modules and parameters is chosen (although the reasons for including the selected module options are not usually explained). During the hyperparameter optimisation process, combinations of these candidates are examined. All of the methods in the literature use total prediction error (for test observations) to guide the improvement of the combination of algorithm modules.

The goal of AutoML, for supervised machine learning, is to minimise total prediction error. However, performing AutoML using total error can lead to a lack of model parsimony and does not focus on an algorithm's sensitivity to

noise [20]. Also, AutoML provides no insight as to why a particular combination of algorithm modules should be chosen. Total error does not explain the behaviour of the algorithm, providing a lack of confidence in the chosen algorithm and a discrepancy between AutoML and the growing need for explainable artificial intelligence [21]. It also does not illustrate the stability of the algorithm and therefore the reliability of the AutoML process. Instead, decomposed error should be used in AutoML to examine algorithm behaviour as it is important to provide end users with greater insights into an algorithm's behaviour and therefore enhanced confidence in its performance [22,23]. Therefore, it needs to be explored how decomposed error can be used to guide algorithm refinement.

3 Algorithm Refinement of Genetic Programming Using Decomposed Error

Estimating decomposed prediction error involves splitting up total error into components based on the potential sources of error. In a typical bias-variance error decomposition, a total error measure (e.g., mean squared error) is decomposed into two primary components: a *bias* component quantifies the ability of the method to learn the underlying generating function of a problem, while a *variance* component quantifies the learning method's sensitivity to stochastic effects encountered during the learning process (e.g., the sampling of data) [24]. To enable variance due to the sampling of training data (external variance) as well as variance due to the algorithm (internal variance) to be represented, the standard bias-variance decomposition can be further expanded, as shown in [12]. Error due to internal variance captures the changes in model predictions observed over multiple algorithm runs using the same training data. When further decomposing the error, Tukey's outlier removal is performed in order to provide more stable decomposed error estimates [12]. The error is decomposed using a set of 100 runs (10 training sets and 10 runs per training set). In order to explore how decomposed error can be used to reliably and effectively refine the GP algorithm, this is performed using both automated and manual processes.

3.1 Automated Algorithm Refinement

A desirable option for algorithm refinement is to use an automated process. Starting with traditional GP, alternative modules can be examined, choosing one to minimise the largest component of error (reducing error due to bias, internal variance or external variance). The process can be repeated a specific number of times or until the largest component of error cannot be reduced. The process has been illustrated by focusing on particular key parts of an evolutionary algorithm, rather than examining all possible alternative modules.

The typical framework of an evolutionary algorithm is shown in Fig. 1. A number of design choices present themselves at each block of the diagram, meaning that there are potentially many choices to be made at each module. The automated algorithm refinement process examined in this paper focuses on the

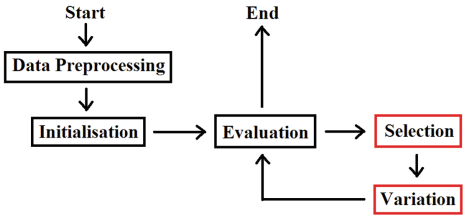

Fig. 1. Modules involved in an evolutionary algorithm process for symbolic regression.

selection and variation (crossover and mutation) parts of the algorithm (shown in red). However, in principle, any part of the algorithm could be examined using this process. The sequential process of targeting the reduction of the largest component of error was performed using these steps:

1. Run GP using an initial configuration of modules.
2. Decompose the error associated with the initial GP configuration.
3. Determine the largest component of error (i.e., error due to bias, internal variance or external variance).
4. Run GP using the current combination of modules except for swapping in each alternative module individually, calculating the decomposed error associated with each combination of modules.
5. Determine which new combination of modules reduces the largest component of error.
6. If the largest component of error cannot be reduced, stop the process and return the current combination of modules. Otherwise, determine the new largest component of error (i.e., repeating Step 3).
7. Repeat Steps 4 to 6 for n time steps (if the process has not already been terminated).

Steps 1 and 4 involve multiple complete runs of GP in order to decompose the error associated with each combination of modules that are examined. Modules from traditional GP, Angle-Driven Geometric Semantic GP ($ADGSGP$) [25] and GP using semantic similarity [26] have been selected to refine GP. The following individual modules were examined:

Selection Operators:

- Tournament selection (TS)
- TS and angle-driven selection (ADS) for crossover [25]
- Double tournament selection (DTS) [27]
- DTS and ADS for crossover

Crossover Operators:

- One point crossover (OPX)

– Perpendicular crossover (PC) [25]
– Semantic similarity-based crossover (SSC) [26]

Mutation Operators:

– Uniform subtree mutation (UM) using full growth
– Random segment mutation (RSM) [25]
– Semantic similarity-based mutation (SSM) based on SSC [26]

The modules of $ADGSGP$ were performed using the implementation of [25]. An initial GP configuration of OPX, UM and TS was used as the starting point for the refinement process, with a maximum of five steps. The refinement of the GP algorithm is explored using a variant of the Keijzer-5 function [28]:

$$f(x, y, z) = \frac{30(x - a)(z - a)}{((x - a) - 10)(y - a)^2} \tag{1}$$

where $a = 10$ for $x, z \in U[9, 11)$ and $y \in U[11, 12)$. A similar adaptation of the Keijzer-5 function is used by [29]. Equation (1) was used to generate 10 training folds of 100 observations and a test fold of 1000 observations. Algorithm refinement driven by decomposed error will generalise to other problems because decomposed error characterises the algorithm for the given problem and therefore will guide the choice of modules for that problem. GP is performed using the Distributed Evolutionary Algorithms in Python (DEAP) framework [30] and the parameters that are not considered for refinement are shown in Table 1, which are typical of those in recent work [31,32].

Table 1. Fixed parameters for GP

Parameter	Value
Population size	100
Number of generations	100
Probability of crossover	0.3
Probability of subtree mutation	0.7
Maximum depth	17
Initial minimum depth	2
Initial maximum depth	6
Minimum depth of subtree mutation	2
Maximum depth of subtree mutation	6
Elitism	Yes (1 individual)
Size of tournament	3
Function set	$\{+, -, \times, \div\}$

The decomposed error values associated with the steps taken to improve the algorithm are shown in Fig. 2. After initially running GP (using OPX, UM and

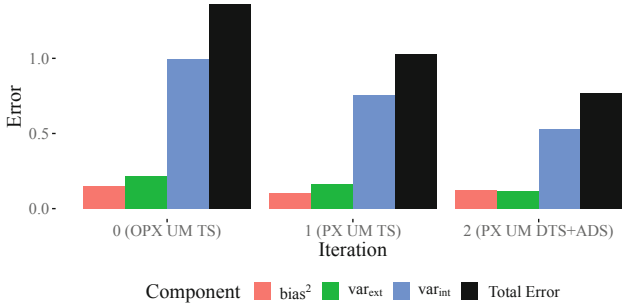

Fig. 2. Decomposed error for data generated by Eq. (1), using the automated algorithm refinement process to change the combination of GP modules implemented.

TS), it was determined that OPX should be swapped with PX in order to reduce the largest component of error (internal variance). Despite this configuration change, internal variance remains the largest error component. In the second iteration, TS was swapped with $DTS + ADS$ in order to continue to reduce error due to internal variance. Subsequent iterations were unable to find combinations of modules that reduced error due to internal variance. Therefore, the process terminated and the mutation operator remained unchanged. However, it is interesting to see that two modules from $ADGSGP$ (PX and ADS for crossover) have been selected using this process. While it is preferable to simultaneously minimise all components of error, the primary goal of each module swap is to reduce the largest component of error, and therefore total error. The process was effective in reducing error primarily due to internal variance but also external variance. However, there was an apparent trade-off between error due to variance and error due to bias. Swapping TS for $DTS + ADS$ provided a reduction in both types of error due to variance but with a slight increase in error due to bias.

3.2 Critique of Automated Algorithm Refinement

The motivations for using this automated algorithm refinement process are clear. First, by automating the process, the only human involvement required is in determining the initial combination of modules examined, the candidate modules and the parameters associated with these modules. Second, the process of changing only one module at each time step allows all candidate modules to be applied without the computational expense of trying all possible combinations. Finally, by choosing a combination of modules that appears to reduce the largest component of error, total error can be reduced (see Fig. 2). However, it needs to be determined whether the estimated decomposed error of GP provides stable enough estimates in order to assess whether the inclusion of a particular module reduces the largest component of error.

For the initial GP configuration of modules $\{OPX, UM, TS\}$ and the subsequent chosen combination of modules $\{PX, UM, TS\}$, the 100 runs ($M = 10$ and $R = 10$) used to estimate the decomposed error were repeated 50 times. The mean decomposed error values (and associated error bars representing one standard deviation) from the 50 repetitions are shown in Fig. 3. The identification of error due to internal variance as the largest component of error for both combinations of modules is consistent across the 50 repetitions. However, the mean internal variance value for the initial combination of $\{OPX, UM, TS\}$ is very similar to that for $\{PX, UM, TS\}$. Also, the error bars have significant overlap, with the inclusion of OPX sometimes providing lower error values compared to PX. While the magnitudes of error due to bias and error due to external variance for OPX and PX are relatively stable across repetitions, this is not the case for error due to internal variance. Across the 50 repetitions, some of them determined that swapping OPX for PX reduces error due to internal variance while other repetitions determined the opposite result (selecting a different module or terminating the process). OPX provided lower error 22 times, larger error 16 times and similar error 12 times (an absolute difference of less than 0.1). As it is not known which error component is targeted by choosing either OPX or PX, it is possible that they both target a reduction in error due to bias and therefore a reduction in error due to variance could not be achieved.

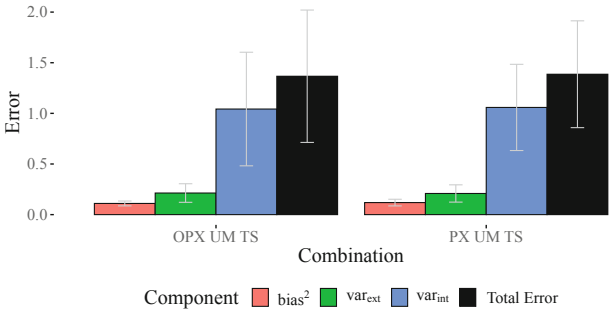

Fig. 3. Mean decomposed error (and associated error bars representing one standard deviation) for data generated by Eq. (1), for 50 repetitions of the first two combinations of GP modules involved in the automated refinement process (see Fig. 2).

These results show that automatic algorithm refinement is associated with difficulties in swapping algorithm modules in order to reduce the largest component of error. In particular, a comparison of modules that exhibit similar decomposed error does not help to substantially reduce the largest component of error. This can also provide inconsistent results when performed for multiple repetitions due to unstable estimates of error due to internal variance. Performing a manual algorithm refinement process might allow for more meaningful comparisons of modules, as we can choose to examine modules with a prior

understanding that they are expected or known to target the largest component of error. Such a manual process is investigated in the next section.

3.3 Manual Algorithm Refinement

The motivation for a manual algorithm refinement process is to identify and then target a reduction in the largest component of error using a module known to successfully reduce that component of error. The automated algorithm refinement process used in Sect. 3.1 was able to consistently determine the largest component of error. However, individual runs gave inconsistent results as to whether changing an operator reduced the largest component of error. Therefore, it is plausible that the largest error component can be determined but used within a manual algorithm refinement process as a heuristic for reducing the largest component of error. This was performed by examining a single well-understood adaptation to an algorithm that targets the largest component of error (or is hypothesised to reduce the largest component of error), rather than blindly comparing alternative algorithm module combinations that involve uncertainty and may be too similar in terms of their error reducing behaviour.

Starting with the GP results for the initial combination of $\{OPX, UM, TS\}$ (see Fig. 3), the largest component of error is consistently error due to internal variance. Therefore, an adaptation to the algorithm needs to be applied in order to reduce this component of error. Bagging is well understood to reduce both error due to internal and external variance. However, there is evidence that bootstrapping of the training data is unnecessary when the error due to external variance component is much smaller than the error due to internal variance component [33]. Averaging the predictions from an ensemble of models without bootstrapping will, like bagging, reduce error due to internal variance. Therefore, an ensemble of 25 models (using the operator combination of $\{OPX, UM, TS\}$ and calculating the median value) was used to predict the test observations, with each set of 100 runs ($M = 10$ and $R = 10$) being performed 30 times. The mean and standard deviation of decomposed error for the ensemble algorithm is compared to that of the initial combination of operators in the first two rows of Table 2. An ensemble of models (without bootstrapping) provided a large reduction in error due to internal variance, which is statistically significant ($p < 0.0001$) using the Wilcoxon signed-rank test for the difference between the error due to internal variance components. The additional runs involved in model averaging provided more accurate predictions with a reduction in error due to bias as well as a reduction in error due to external variance.

The new largest error component associated with the operator combination of $\{OPX, UM, TS\}$ in an ensemble is error due to bias. This is consistent across all 30 repetitions. Therefore, a different type of adaptation needs to be applied in order to reduce error due to bias. As both GP with Z-Score standardisation (of explanatory and response variables) and GP with linear scaling have been shown to reduce error due to bias [31,34], GP was performed using a combination of both standardisation and linear scaling (GP_{Z+LS}). The decomposed error of the ensemble algorithm without feature scaling is compared to the ensemble

Table 2. Variants of GP examined using manual algorithm refinement

Variant	$bias^2$	var_{ext}	var_{int}	Total error
OPX UM TS	0.1138 ±0.0005	0.2289 ±0.0951	1.0826 ±0.5985	1.4253 ±0.6959
OPX UM TS Ens	0.0441 ±0.0005	0.0263 ±0.0012	0.0054 ±0.0008	0.0759 ±0.0018
GP_{Z+LS} OPX UM TS Ens	**0.0029 ±0.0000**	**0.0025 ±0.0001**	**0.0002 ±0.0000**	**0.0057 ±0.0001**
GP_{Z+LS} OPX UM TS	0.0031 ±0.0004	0.0055 ±0.0024	0.0232 ±0.0147	0.0318 ±0.0174

algorithm using GP_{Z+LS} in the second and third rows of Table 2 (with the lowest error component values bolded in the third row). Using GP_{Z+LS} provided a large reduction in error due to bias, which is statistically significant ($p < 0.0001$) using the Wilcoxon signed-rank test for the difference between the error due to bias components. It also reduced error due to both external and internal variance (with both differences statistically significant, both $p < 0.0001$). As exhibited when creating an ensemble of models without bootstrapping, all error components were reduced. Therefore, these wrapper methods have reduced total error without exhibiting a trade-off between error due to bias and error due to variance.

The new largest error component associated with an ensemble of models (without bootstrapping) using GP_{Z+LS} was still error due to bias (across all 30 repetitions). However, this component was only slightly larger than error due to external variance, with both components being significantly reduced by performing standardisation and linear scaling. Therefore, this manual refinement process reached an appropriate stopping point for the examined data set.

3.4 Critique of Manual Algorithm Refinement

Estimating which component of error is the largest component gave consistent results over multiple repetitions of the runs required to decompose error. This provides confidence in determining what type of adaptation to the algorithm needs to be made in order to reduce the largest component of error and therefore total error. While this process requires human involvement after performing multiple runs of each adaptation of the algorithm, it allows domain knowledge and targeted decision making to be exploited. It also provides an explainable refinement process by understanding the purpose of selecting particular modules in terms of decomposed error. The two adaptations to the algorithm (see Table 2) were successful in targeting the largest component of error, and therefore total error. However, it needs to be confirmed whether both adaptations were necessary to reduce total error and error due to internal variance in particular.

The final combination of modules before stopping the manual refinement process (GP_{Z+LS} using an ensemble of 25 models) is compared to GP_{Z+LS} without ensembling in the third and fourth rows of Table 2. The results show that a single model provides a statistically significant increase in error due to internal variance ($p < 0.0001$) compared to the ensemble model, using the Wilcoxon signed-rank test for the difference between the error due to internal variance components. This makes it clear that while standardisation and linear scaling provide lower error due to internal variance than the initial set of modules (without standardisation and linear scaling), they do not sufficiently target the component. An ensemble model is needed to target error due to internal variance as standardisation and linear scaling specifically target error due to bias. The standard deviation is also larger for error due to bias and error due to external variance for a single model.

By considering a small set of candidate algorithm adaptations that are known or expected to target the reduction of a specific error component, this manual process requires fewer runs of GP. Instead of performing many runs across many different algorithm adaptations, the focus can be on multiple repetitions of the same algorithm to determine the consistency of both the overall predictive performance and the magnitude of the error components. A small set of candidate algorithms is sufficient if, between them, they capture a reduction in all error components. This algorithm refinement process is not trying to find the algorithm with the best possible predictive performance but instead find an algorithm that provides reasonable performance as well as stable and well-understood behaviour. While only wrapper adaptations to the algorithm have been examined (feature scaling and ensemble models), adaptations internal to the algorithm can also be examined using this manual process. *A set of candidate algorithm adaptations or modules with known or expected behaviour, in terms of targeting a reduction of the largest component of decomposed error*, is a desirable characteristic of an AutoML process. By understanding the behaviour of algorithm adaptations or modules, the set of candidate options can be chosen more carefully in order to provide a diverse range of behaviour, leading to a more effective and explainable reduction of error.

4 Conclusion

This paper introduces the use of decomposed error for performing algorithm refinement. It has been applied to the refinement of GP using both automatic and manual processes. The results for the automatic algorithm refinement process show that comparing algorithm modules with similar decomposed error values makes it difficult to target a reduction of the largest error component. This is particularly the case for algorithms like GP than can exhibit a large and/or unstable error due to internal variance component and therefore can provide inconsistent conclusions about candidate algorithm adaptations. In order to make more meaningful comparisons, the manual refinement process focuses on choosing a candidate algorithm adaptation or module that is known to reduce

the largest component of error, determined by manually examining the estimated decomposed error of the algorithm at each step in the process.

For the manual algorithm refinement process, the sequence of algorithm adaptations was successful in reducing the largest component of error, with the type of the largest error component changing throughout the process. Therefore, a set of candidate algorithm adaptations or modules need to provide diversity in reducing different components of decomposed error. Many traditional AutoML processes choose a set of candidate algorithm adaptations or modules without prior examination of their diversity in terms of reducing different components of error. Therefore, choosing candidate modules that coincidentally target a reduction of the same component of error will significantly limit the ability to improve the predictive performance of an algorithm. A greater understanding of how an algorithm module reduces prediction error, and the module's interaction with other modules, can be provided using the extended error decomposition. A more strategically chosen set of candidate algorithm modules, in terms of providing diverse behaviour in reducing different components of error, can then be applied to an automated algorithm refinement process. It is particularly important for the set of candidates to include a module that reduces error due to internal variance. This allows for the prediction error associated with an algorithm to be stabilised, if required, before being able to make reliable further refinements that target other components of error.

Although the manual algorithm refinement process was more successful than the automated process in reducing the largest component of error (and therefore total error), the motivations for automating algorithm refinement are still clear and important. Therefore, mapping the successful elements of the manual algorithm refinement process into an AutoML framework should be explored in future work. This would involve choosing a hierarchy of modules that are most important for reducing a diverse range of error components. This provides efficiency in reducing the search space of hyperparameters while providing confidence in the behaviour of the module combinations. The examination of algorithm refinement in this paper focused on the overall algorithm module structure rather than the tuning of parameters; this should be examined in future work.

Acknowledgment. Thank you to Dr Qi Chen for kindly allowing your ADGSGP code to be used as part of this paper.

References

1. Erickson, B.J., Korfiatis, P., Akkus, Z., Kline, T.L.: Machine learning for medical imaging. Radiographics **37**(2), 505–515 (2017)
2. Tuggener, L., et al.: Automated machine learning in practice: state of the art and recent results. In: 2019 6th Swiss Conference on Data Science (SDS), pp. 31–36. IEEE, New Jersey (2019)
3. Carleo, G.: Machine learning and the physical sciences. Rev. Mod. Phys. **91**(4), 045002 (2019)

4. Mitchell, T.: Machine Learning, ser. McGraw-Hill International Editions. McGraw-Hill, New York (1997). https://books.google.co.nz/books?id=EoYBngEACAAJ
5. Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer, Berlin (2009). https://doi.org/10.1007/978-0-387-21606-5
6. Wolpert, D.H., Macready, W.G.: No free lunch theorems for optimization. IEEE Trans. Evol. Comput. **1**(1), 67–82 (1997)
7. Elshawi, R., Maher, M., Sakr, S.: Automated machine learning: state-of-the-art and open challenges, pp. 1–23. CoRR, vol. abs/1906.02287 (2019). http://arxiv.org/abs/1906.02287
8. Olson, R.S., Cava, W.L., Mustahsan, Z., Varik, A., Moore, J.H.: Data-driven advice for applying machine learning to bioinformatics problems. In: Pacific Symposium on Biocomputing 2018: Proceedings of the Pacific Symposium, pp. 192–203. World Scientific, Singapore (2018)
9. Thornton, C., Hutter, F., Hoos, H.H., Leyton-Brown, K.: Auto-weka: combined selection and hyperparameter optimization of classification algorithms. In: Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 847–855. ACM, New York (2013)
10. Mohr, F., Wever, M., Hüllermeier, E.: Ml-plan: automated machine learning via hierarchical planning. Mach. Learn. **107**(8), 1495–1515 (2018)
11. James, G., Witten, D., Hastie, T., Tibshirani, R.: An Introduction to Statistical Learning: with Applications in R. STS, vol. 103. Springer, New York (2013). https://doi.org/10.1007/978-1-4614-7138-7
12. Owen, C.A., Dick, G., Whigham, P.A.: Characterising genetic programming error through extended bias and variance decomposition. IEEE Trans. Evol. Comput. **24**(6), 1164–1176 (2020)
13. Drozdal, J., et al.: Trust in automl: exploring information needs for establishing trust in automated machine learning systems. In: Proceedings of the 25th International Conference on Intelligent User Interfaces, pp. 297–307 (2020)
14. Adadi, A., Berrada, M.: Peeking inside the black-box: a survey on explainable artificial intelligence (xai). IEEE Access **6**, 52 138–52 160 (2018)
15. Bergstra, J., Bengio, Y.: Random search for hyper-parameter optimization. J. Mach. Learn. Res. **13**(2), 281–305 (2012)
16. Kotthoff, L., Thornton, C., Hoos, H.H., Hutter, F., Leyton-Brown, K.: Auto-WEKA: automatic model selection and hyperparameter optimization in WEKA. In: Hutter, F., Kotthoff, L., Vanschoren, J. (eds.) Automated Machine Learning. TSSCML, pp. 81–95. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-05318-5_4
17. Feurer, M., Klein, A., Eggensperger, K., Springenberg, J.T., Blum, M., Hutter, F.: Auto-sklearn: efficient and robust automated machine learning. In: Hutter, F., Kotthoff, L., Vanschoren, J. (eds.) Automated Machine Learning. TSSCML, pp. 113–134. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-05318-5_6
18. de Sá, A.G.C., Pinto, W.J.G.S., Oliveira, L.O.V.B., Pappa, G.L.: RECIPE: a grammar-based framework for automatically evolving classification pipelines. In: McDermott, J., Castelli, M., Sekanina, L., Haasdijk, E., García-Sánchez, P. (eds.) EuroGP 2017. LNCS, vol. 10196, pp. 246–261. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-55696-3_16
19. Olson, R.S., Bartley, N., Urbanowicz, R.J., Moore, J.H.: Evaluation of a tree-based pipeline optimization tool for automating data science. In: Proceedings of the Genetic and Evolutionary Computation Conference, ser. GECCO 2016, pp. 485–492. ACM, New York (2016)

20. Brighton, H., Gigerenzer, G.: The bias bias. J. Bus. Res. **68**(8), 1772–1784 (2015)
21. Krawiec, K.: Behavioral Program Synthesis with Genetic Programming, vol. 618. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-27565-9
22. Lipton, Z.C.: The mythos of model interpretability. Commun. ACM **61**(10), 36–43 (2018)
23. Arrieta, A.B., et al.: Explainable artificial intelligence (XAI): concepts, taxonomies, opportunities and challenges toward responsible AI. Inf. Fusion **58**, 82–115 (2020)
24. Bishop, C.M.: Pattern Recognition and Machine Learning (Information Science and Statistics). Springer-Verlag, Secaucus (2006)
25. Chen, Q., Xue, B., Zhang, M.: Improving generalization of genetic programming for symbolic regression with angle-driven geometric semantic operators. IEEE Trans. Evol. Comput. **23**(3), 488–502 (2019)
26. Uy, N.Q., Hoai, N.X., O'Neill, M., McKay, R.I., Galván-López, E.: Semantically-based crossover in genetic programming: application to real-valued symbolic regression. Genetic Program. Evol. Mach. **12**(2), 91–119 (2011)
27. Luke, S., Panait, L.: Fighting bloat with nonparametric parsimony pressure. In: Guervós, J.J.M., Adamidis, P., Beyer, H.-G., Schwefel, H.-P., Fernández-Villacañas, J.-L. (eds.) PPSN 2002. LNCS, vol. 2439, pp. 411–421. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-45712-7_40
28. Keijzer, M.: Improving symbolic regression with interval arithmetic and linear scaling. In: Ryan, C., Soule, T., Keijzer, M., Tsang, E., Poli, R., Costa, E. (eds.) EuroGP 2003. LNCS, vol. 2610, pp. 70–82. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-36599-0_7
29. Vladislavleva, E.J., Smits, G.F., Den Hertog, D.: Order of nonlinearity as a complexity measure for models generated by symbolic regression via pareto genetic programming. IEEE Trans. Evol. Comput. **13**(2), 333–349 (2009)
30. Fortin, F.-A., De Rainville, F.-M., Gardner, M.-A.G., Parizeau, M., Gagné, C.: Deap: evolutionary algorithms made easy. J. Mach. Learn. Res. **13**(1), 2171–2175 (2012)
31. Owen, C.A., Dick, G., Whigham, P.A.: Standardisation and data augmentation in genetic programming. IEEE Trans. Evol. Comput. (2022)
32. Dick, G., Owen, C.A., Whigham, P.A.: Evolving bagging ensembles using a spatially-structured niching method. In: Proceedings of the Genetic and Evolutionary Computation Conference, ser. GECCO 2018, pp. 418–425. ACM, New York (2018). http://doi.acm.org/10.1145/3205455.3205642
33. Owen, C.A.: Error decomposition of evolutionary machine learning (Thesis, Doctor of Philosophy). University of Otago (2021). http://hdl.handle.net/10523/12234
34. Owen, C.A., Dick, G., Whigham, P.A.: Feature standardisation in symbolic regression. In: Mitrovic, T., Xue, B., Li, X. (eds.) AI 2018. LNCS (LNAI), vol. 11320, pp. 565–576. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-03991-2_52