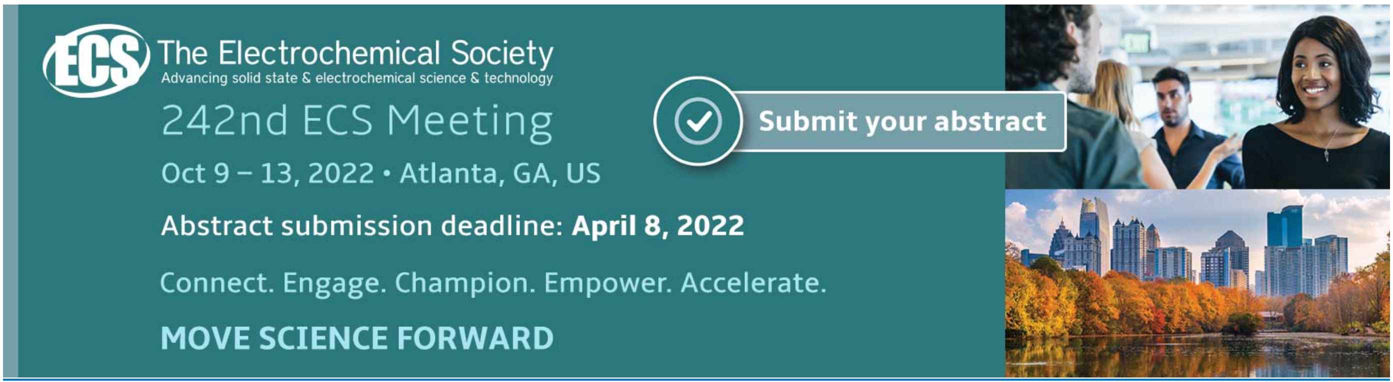
IOP Conference Series: Earth and Environmental Science

PAPER • OPEN ACCESS

A Reinforcement-Algorithm Framework for Automatic Model Selection

To cite this article: Diya Shang *et al* 2020 *IOP Conf. Ser.: Earth Environ. Sci.* **440** 022060

View the article online for updates and enhancements.


ECS
The Electrochemical Society
Advancing solid state & electrochemical science & technology
242nd ECS Meeting
Oct 9 – 13, 2022 • Atlanta, GA, US
Abstract submission deadline: April 8, 2022
Connect. Engage. Champion. Empower. Accelerate.
MOVE SCIENCE FORWARD
Submit your abstract

A Reinforcement-Algorithm Framework for Automatic Model Selection

Diya Shang[a], Hua Sun[b] and Qingliang Zeng[c]

School of Xinjiang University, Urumqi 830000, China

[a]shangdiya1024@gmail.com, [b]xj_sh@163.com, [c]13579980465@163.com

Abstract. In recent years, with the development of artificial intelligence, how to realize the automation of machine learning has become a very popular research item. Among them, automatic model selection is an important part. For traditional machine learning, the part of the feature preprocessing, feature selection, model selection and so on are all needed manual processing, and they will take too much time and human resource, so automatic model selection is proposed to reduce resource waste. The automatic model selectionis based on the feature distribution of the data, and the model can be trained to obtain a model with better performance. This paper proposes an automatic model selection framework based on reinforcement learning. The framework can be divided into two phases, one is data preprocessing stage, which uses meta-learning to process; the other is model se lection framework, including feature preprocessing, feature selection and model selection. These three aspects will be the environment of reinforcement learning, through which a model with the highest accuracy can be chosen as the predictive model output. It has been proved by experiments that the method is effective.

1. Introduction

The automatic machine learning [1, 2] is the trend of future machine learning development. It is a way to apply the traditional machine learning to practical problems in an end-to-end form. In the process of traditional machine learning, data scientist should know how to perform data preprocessing [3, 4], implement feature engineering [5], complete feature selection [6, 7] and model selection [8]. Therefore, in order to practice or develop that according to these processing steps, we must know how to perform algorithm selection [9] and hyper parameter optimization [10], so as to maximize the prediction performance of the final machine learning model. Since many of these steps are complicated so that many people are difficult to understand it, AutoML is proposed as an artificial intelligence-based solution to overcome the growing challenge of applications of machine learning. The end-to-end process of automated applied machine learning provides the advantages for creating simpler solutions and creating these solutions or models faster and producing these models that are generally superior to our hand-designed models.

For traditional machine learning topics, there are many stages including the process of completing feature selection, model selection and etc. There are many different options. For example, in the data preprocessing stage, how to perform data preprocessing [3, 4], what type of feature selector or constructor function should be chosen. For a classification problem, which model should be used during the model selection phase, such as logistic regression [11], SVM [12] or naive bayes [13], and how to set the parameter of each model? Etc. So how to choose the best model? Model selection usually involves two steps: the first step is selecting a model, and the other step is setting the model's parameters. Though experienced machine learning developer often have good intuitions on what choices are appropriate for the problem domain, it also needs several weeks tinkering with model parameters and data transformations until the whole model achieves an acceptable level of performance.

In order to solve the above problems, it needs to be solved by using automated machine learning. Automated model selection is integrated with data preprocessing, feature selection, and model selection, equivalent to a "black box" that automatically perform data processing, feature selection, and model selection. The goal is to automatically select the most appropriate model and set its optimal parameters. At present, the methods of existing automatic model selection include Auto-WEKA [14, 15] and Auto-sklearn [16] who based on Bayesian optimization, and TPOT [17] based on evolutionary algorithm. Auto-WEKA and Auto-sklearn are the first frameworks to try to apply AutoML to machine learning, and to find the best combination of data preprocessing, hyper parameter adjustment and model selection. The purpose of basing on Bayesian optimization is to avoid exhaustive network search parameters. TPOT is the most classic framework to use the evolutionary algorithms to achieve automatic selection of models. The main idea is to optimize the machine learning pipeline. TPOT implements the pipeline in the form of a tree, using genetic programming to optimize the pipeline operation and operation parameters, such as the number of trees in the random forest or the number of feature pairs to be selected during feature selection, thereby maximizing the classification accuracy of the pipeline.

In this paper, reinforcement learning [18, 19] will be used to automate the selection and tuning of models. As shown in Figure 1, it is the framework flow of this learning. First of all, preprocessing the input data, then using reinforcement learning to achieve feature selection and model selection, and evaluating the model, outputting the best model as the prediction model. The part of data preprocessing will adopt the meta-learning [20] method. Meta-learning can learn and predict according to the characteristics of the dataset, and then obtain information such as data distribution, and it can also initialize the later reinforcement learning framework. In particular, for a large data set, collecting a set of meta-features can effectively calculate the characteristics of the dataset and it is helpful for determining which algorithm to use on the new dataset. This method can be used to optimize the AutoML framework and complements reinforcement learning.

The contribution of this paper is as follows:

(1) We use meta-learning to select the state and action space by learning the distribution of datasets and the feature of datasets. Meta-learning as the data-preprocess module can initialize the Reinforcement Learning module and provide initialization parameters.

(2) We use reinforcement algorithm to select the features and models, then tune models by it, and find the best combination between them through feature selection and model selection.

(3) We use the prior knowledge to balance the exploit space and exploration space and improve the search efficiency while not completely limiting the search space by adding prior knowledge.

Our experimental results show that the automatic model selection method based on reinforcement learning has better accuracy than traditional manual experiments. Compared with the automatic model selection method based on evolutionary algorithm TPOT, the search speed is much faster and can achieve better accuracy than TPOT.

The outline of this paper is organized as follows. The section 2 illustrates the related work. The main idea is shown in section 3. An evaluation setup and results are shown in section 4. Section 5 concludes this paper.

2. Related Work

2.1. Meta-Learning

In data preprocessing, we use the meta-learning method, which can improve efficiency. This method of processing data draws on the approach of M. Feurer et al. [16] in the Auto-sklearn system. Meta learning is a sub-area of machine learning that applies the ability of automatic learning to metadata. The idea is to learn the process of “learning training” in order to design a form to quickly learn new skills or adapt to the new environment’s model. Specifically, it is expected that a good meta-learning model can be well adapted or extended to new tasks and new environments that have never been encountered during training. The adaptation process to the new task is essentially a mini learning session. This process can be completed quickly with a small amount of training. Finally, the adapted model can complete the new task. Therefore, meta-learning is also called learning to learn. Meta learning does not depend on the amount of data in generalization and can be completed with only a small number of samples on new tasks.

In the data preprocessing stage, the traditional methods are data cleaning, data integration, data transformation and data protocol, etc. After that, the performance of the machine learning algorithm needs to be obtained manually from the previous task. This process is time-consuming and labour-intensive. The meta-learning domain [21] simulates this strategy by inferring the performance of learning algorithms across datasets. Therefore, we apply meta-learning to select instances of our given machine learning frameworks that may perform well on new datasets. In other words, for a large number of datasets, we collect both performance data and a set of meta-features.

This method of using meta-learning as a data pre-processing has been successfully applied on [22, 23, and 24]. So we use this approach as the data-preprocess can not only get the data feature efficiently, but also the data distribution.

2.2. Bayes-based Automated Model Selection

Auto-WEKA and Auto-sklearn were the first to try to apply AutoML to machine learning. Both of these frameworks are automated model selection frameworks based on Bayesian optimization.

Auto-WEKA system is an automatic model selection framework based on Bayesian optimization proposed by Thornton et al. [14]. It uses the machine learning framework WEKA and tree-based Bayesian optimization methods [25, 26].For the problem of simultaneously selecting the learning algorithm and setting its hyper parameters, it is solved by the fully automatic method and utilizes Bayesian optimization. Auto-WEKA reduces the machine learning process to the Algorithm Selection and Hyper-parameter optimization (CASH). The search space of the Auto-WEKA framework includes 39 classification and regression algorithms, including 27 Base classifiers such as KNN, SVM, LR, etc. 10 Meta classifiers, such as AdaBoostM1, Logit Boost, etc., 2 ensemble methods, Vote and stacking. The Meta classifier can select any of the base classifiers as input, and the ensemble classifier can use up to five base classifiers as inputs. On the data side, using k-fold cross-validation, it divides the training data into k equal-sized partitions, and simply performs k independent runs of parallel optimization methods and selects the result of the lowest cross-validation error. There are two search strategies for CASH problems: Sequential Model-based Algorithm Configuration (SMAC) and Tree-structured Parzen Estimator (TPE), both of which belong to the SMBO algorithm, which is a category of Bayesian optimization algorithms.

Auto-sklearn is proposed by M. Feurer et al. in 2015[16] an AutoML machine learning framework based on the scikit-learn machine learning package in Python environment. Based on Auto-WEKA, two components are added to the Bayesian hyper parameter optimization of the AutoML framework: meta-learning for initializing the Bayesian optimizer and automatic integration of the model during optimization. Auto-sklearn's search space includes: 15 classification algorithms, such as KNN, AdaBoost, SVM, etc., 14 preprocessing methods, such as PCA, ICA, etc., and 4 data preprocessing methods, one-hot encoding, imputation, balancing and rescaling.

2.3. EA-based Automated Model Selection

In addition to the Bayesian method, an evolutionary algorithm can be used to implement automatic selection of the model. The most classic model selection method based on evolutionary algorithm is the TPOT (Tree-based Pipeline Optimization Tool) framework proposed by Randal S. Olson et al. [17]. This is a framework for automatically designing and optimizing machine learning pipelines based on the scikit-learn package. TPOT includes four types of pipeline operations. The first is data preprocessing op (StandardScaler, RobustScaler, and Polynomial Features), and the second is feature construction op (Randomized PCA, a variant of principal component analysis using random singular value decomposition) The third is feature selection op (including four strategies: recursive feature elimination strategy, Select Best, Select Percentile, Variance Threshold), and the fourth is model selection op, which implements models based on individual and collection trees (decision tree, random forest) and Gradient Boosting Classifier), non-probability and probability linear models (SVM and Logistic Regression), and KNeighborsClassifier. TPOT uses a tree structure to combine these four types of ops into a flexible pipeline structure. TPOT automatically generates and optimizes these tree-based pipelines using genetic programming methods.

3. Main idea

3.1. Motivation

In the traditional machine learning method, the following problems exist:

(1) For different datasets, in the traditional machine learning method, a specific model is generally selected according to the characteristics of the dataset, and then trained, and no single machine learning method can process all the data sets;

(2) In the traditional machine learning method, different methods have multiple parameters to choose from, and some machine learning methods rely heavily on hyper parameter optimization.

Our goals are as follows:

(1) For the problem of using different models for different datasets in the traditional machine learning method, a variety of machine learning methods can be combined, and the best effect model in the framework is automatically selected for the input dataset according to the input dataset as an output;

(2) For some machine learning methods that rely on hyper parameter optimization, model selection and hyper parameter optimization can be combined to select and optimize parameters while selecting models.

At present, the hyper parameter optimization problem has been well-appeared in the Auto-WEKA and Auto-sklearn frameworks. Their framework uses Bayesian optimization. If you use reinforcement learning to select models and hyper parameter optimization, will it get a different effect?

3.2. Framework

In the above content, the framework structure that this article will take has already been introduced, as shown in Figure 1. We divide the framework into two parts to achieve automated model selection. The first part is the preprocessing stage of data, which adopts the method of meta-learning; the second part is the stage of feature selection and model selection, which adopts the method of reinforcement learning.

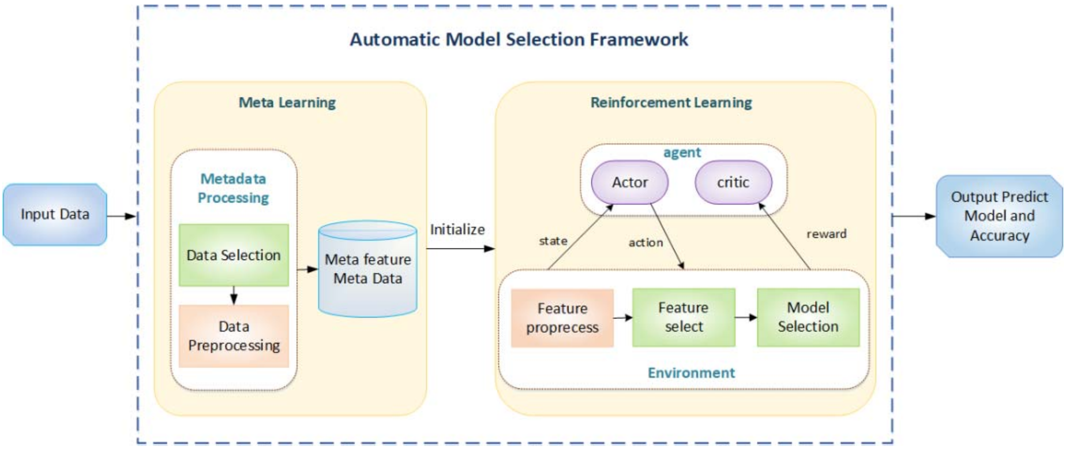


Figure 1. Automated Model Selection Framework

3.2.1. Meta-Learning. The data preprocessing stage, which usually consists of several steps, such as data selection, data pre-processing, data mining, and evaluation or interpretation [27]. The most important step in this process is data pre-processing step. Data pre-processing is so important that usually take 50-80% of analysis time [28]. The reason is that proper pre-processing of the data set can lead to better results. One can apply the best learning algorithm, but if the data is not well-prepared, the algorithm may perform poorly (e.g., bad predictive accuracy).

In this paper, we use meta-learning to preprocess data. The initialization parameters of "search space" and "action space" can be determined by meta-learning. Meta-learning is to solve the problem of designing a machine learning model with the ability to acquire knowledge diversity, which can quickly learn new concepts and skills through a small number of training samples based on past experience and knowledge. In other words, the meta-learning model can be well adapted or extended to new tasks and new environments that have never been encountered during training. This adaptation process to a new task is essentially a mini learning session, which can be done quickly with a small amount of training. Finally, the adapted model can complete new tasks. As shown in Figure 2a below, it is the basic schematic of meta-learning. As shown on the left side of Figure 2a, there is a training data set with only a small number of data samples, and their labels have no correlation and are randomly sampled. Using these small amount of data samples to get a Meta model after training. As shown on the right side of Figure 2a, it is the prediction dataset. The data and categories are not in the training set. The prediction dataset can predict the type of data through the Meta model. That is to say, using the training data sample to fit a Meta model with generalization ability, this model can quickly get a high-quality classifier by generalizing only a small number of images on the test data sample.

In this paper, we will process the data in this way. The basic flow chart is shown in Figure 2b. The whole process involves two stages of training with one model. For the training phase, input different data sets dataset 1, dataset 2,···, dataset n. Get their meta data or meta feature, then get the meta model through the learner. For the prediction phase, input prediction data, we call that is Raw data, then the Raw data passes through the meta model trained in the previous stage, next through processor operator, finally we will get the processed data. This data will be used in the reinforcement learning stage as input data.

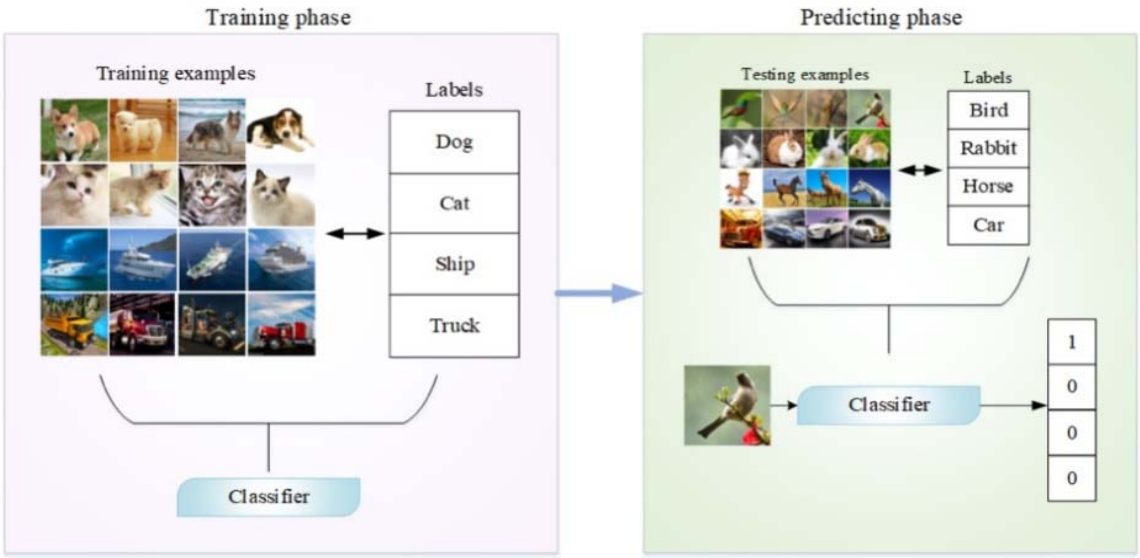


Figure 2a. The Basic Process of Meta Learning.

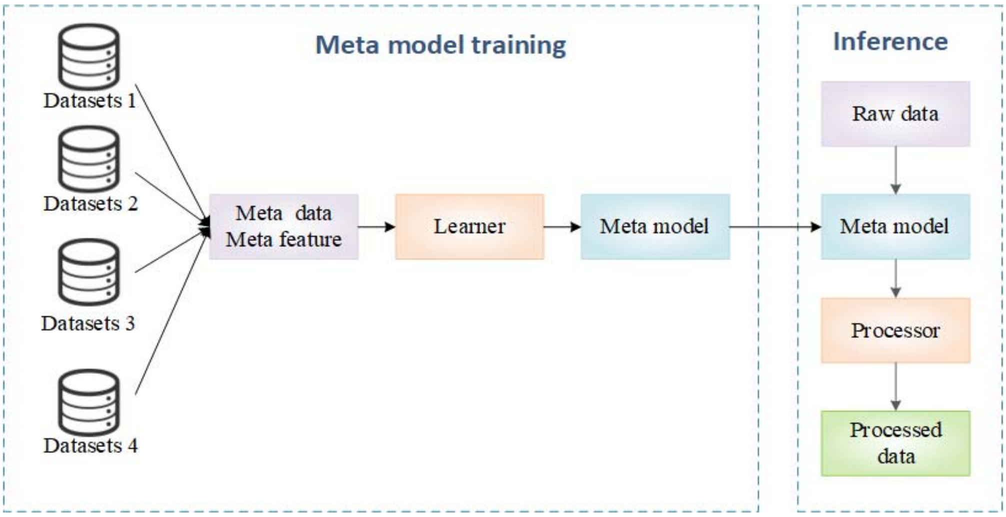


Figure 2b. Meta Learning Framework.

3.2.2. Optimization

In the feature selection and model selection phase, we use reinforcement learning to optimize the framework. Our reinforcement learning method uses the DDPG (Deep Deterministic Policy Gradient) algorithm [19]. The optimization process is divided into three parts: feature preprocessing, feature selection, and model selection. The feature preprocessing includes 12 methods, the feature selection includes 5 methods, and the model selection includes 11 models. Each method also includes many parameters in the middle. The specific content will be introduced in the experimental part. Before we use reinforcement learning to deal with problems, we must first define the state space, action space and reward.

State Space In this paper, we represent the state space, *S*, as a collection of three processes:

$$S = [N, S_{fp}, S_{fs}, \mathrm{m}\ S_{ms}] \tag{1}$$

N Is the executed step; S_{fp} is the state space of the feature process method; S_{fs} is the state space of the feature selection method, and S_{ms} is the method included in the model selection.

Action Space We represent the action space, A, as a method and its parameters that are selected:

$$A = [m, p] \tag{2}$$

m Is the selected method, and p is the set of parameters in the method.

Reward Here, we define reward as accuracy:

$$R = acc \tag{3}$$

Interaction Process After defining the state space and the action space, how does the framework interact as shown on the right side of Figure 1, that is the interaction process of the framework, the initial states_0, which is defined as the default value. The interaction process is divided into three stages. The first stage: In the state s_0, the agent selects a preprocessing method from the feature preprocess as the action a_0, and executes a_0 in the environment to obtain the state s_1 feedback to the agent; The second stage: in the state s_1, the agent selects an operator from the feature selection as the action a_1, and executes a_1 in the environment to obtain the state s_2 feedback to the agent; The third stage: in the state s_2, the agent selects a model from the model selection as the action a_2, and a_2 is executed in the environment, then the state s_3 is fed back to the agent, at the same time the reward is calculated and fed back to the agent. The above three stages are an episode. By looping through these three stages, the best model of the predicted result is final output as the prediction model.

4. Experiment

We now describe an experimental study of the performance that can be achieved by reinforcement-algorithm framework for automatic model selection on various datasets. We will introduce it in two parts. The selection of the dataset and experimental setup will be introduced in the first part (section 4.1). The experimental results and analysis will be introduced in the second part (section 4.2).

4.1. Experiment Setup

We evaluated our framework on 13 datasets (see Table 1), they are from UCI or Kaggle. We randomly split the dataset into 80% training and 20% testing. We specify more than 4000 datasets as “large dataset”, and all others as “small”. And these datasets cover a diverse range of application areas, which guarantee the breadth and universal applicability of our experiments. We used 11 classifiers, 5 feature preprocessing, 12 data preprocessing methods. These preprocessing methods and classifiers are similar to TPOT, and our experiments will be compared to TPOT.

Table 1. Datasets Used; Classes and size refer to the number of attributes of elements and the size in the dataset.

Datasets	Wheat seeds	Heart	Absent	Trial	Titanic	Red wine	Car	Whitewine	App	Gestures	Credit cards	Bank	Adult data
Classes	7	14	21	18	12	12	6	12	17	65	24	17	14
Size	210	303	740	777	1309	1599	1728	4898	7197	11647	30000	45211	48842

4.2. Results and Analysis

We conducted handcraft experiments, TPOT and reinforcement-based, and recorded their accuracy and runtime. We use i7-6700HQ 8cores CPU to get the result through these three methods. Among them, TPOT has evolved 20 generations on each dataset, and our framework iterates 1000 times on each dataset. Figure 3a shows the result of accuracy and is arranged in ascending order of dataset, and detailed values are shown in Table 2. We can see that, in general, the accuracy of manual experiments is lower than that of TPOT and our framework. The accuracy of TPOT and our framework is similar, and the accuracy of results of about 50% of datasets is higher than that of TPOT, which is 50% lower than TPOT. Maybe our framework is more suitable for "small dataset". In the result of "small dataset", 90% of the datasets are better than TPOT, and in the "large dataset" results, only 20% of the results are better than TPOT.

In addition to the comparison of accuracy, time comparison is also carried out. The time of manual experiment is not used as a reference here. Because manual experiment requires manual preprocess, this time is basically several hours or more, which is not good for recording. Therefore only the time of the TPOT and our framework is recorded. As shown in Figure 3b, a comparison of the time results of the two methods is shown, and detailed values are shown in Table 3a. It can be seen from the experimental results that the iteration time of our framework is much smaller than that of TPOT due to the large difference in time. Therefore, some processing is performed on the experimental results, and all values are taken as logs (See Table 3b). As can be seen from the result, as the dataset grows larger, the time of evolution and iteration also increases. But the time results of all the datasets show that our framework time is much lower than the time required by TPOT, whether it is "small dataset" or "large dataset".

Table 2. The accuracy between three methods (Handcraft vs. TPOT vs. RA-based).

Datasets Accuracy	Wheat Seeds	Heart	Absent	Trial	Titanic	Red wine	Car	White wine	App	gestures	Credit cards	Bank	Adult data
Handcraft	0.9623	0.836	1	1	0.8899	0.6531	0.9393	0.5714	0.9958	0.9302	0.8212	0.9033	0.8662
TPOT	0.9623	0.8689	1	1	0.8732	0.6781	1	0.6775	0.9951	0.9627	0.8214	0.9079	0.8742
RA-Based	0.9811	0.9016	1	0.9935	0.8947	0.6813	0.9971	0.6724	0.9965	0.9559	0.8218	0.9104	0.8661

Table 3a. Running time comparison (TPOT vs. RA-based).

Datasets Time(s)	Wheat Seeds	Heart	Absent	Trial	Titanic	Red wine	Car	White wine	App	Gestures	Credit cards	Bank	Adult data
TPOT	657.5	877.41	367.65	635.47	968.36	4899.32	2368.22	3798.12	3064.31	102890	23306.31	83901.7	21475
RA-Based	47.1	35.24	93.65	74.99	170.33	83.86	53.2462	325.66	522.47	2317	4655.64	2275.36	1810

Table 3b. Running time in logarithmic comparison (TPOT vs. RA-based).

Datasets Time/log(t)	Wheat Seeds	Heart	Absent	Trial	Titanic	Red wine	Car	White wine	App	Gestures	Credit cards	Bank	Adult data
TPOT	2.8179	2.9432	2.5654	2.8031	2.986	3.6901	3.3744	3.5796	3.4863	5.0124	4.3675	4.9238	4.3319
RA-Based	1.673	1.547	1.9715	1.8751	2.2313	1.9236	1.7263	2.5128	2.7181	3.3649	3.6679	3.3571	3.2578

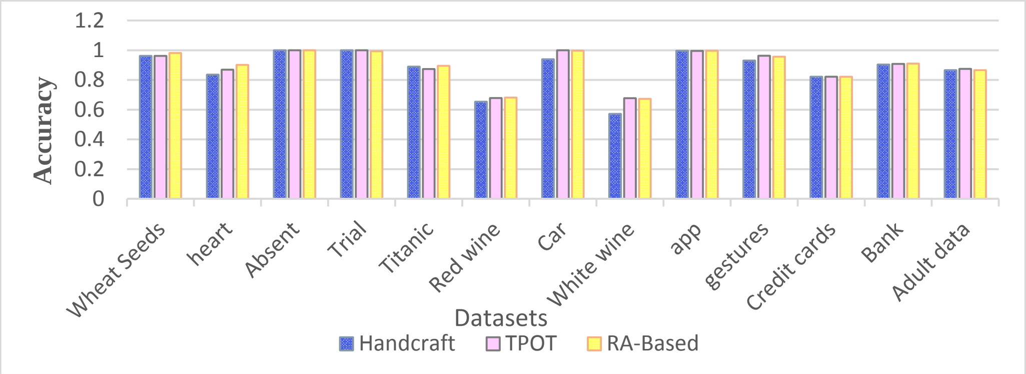


Figure 3a. The accuracy between three methods (Handcraft vs. TPOT vs. RA-based).

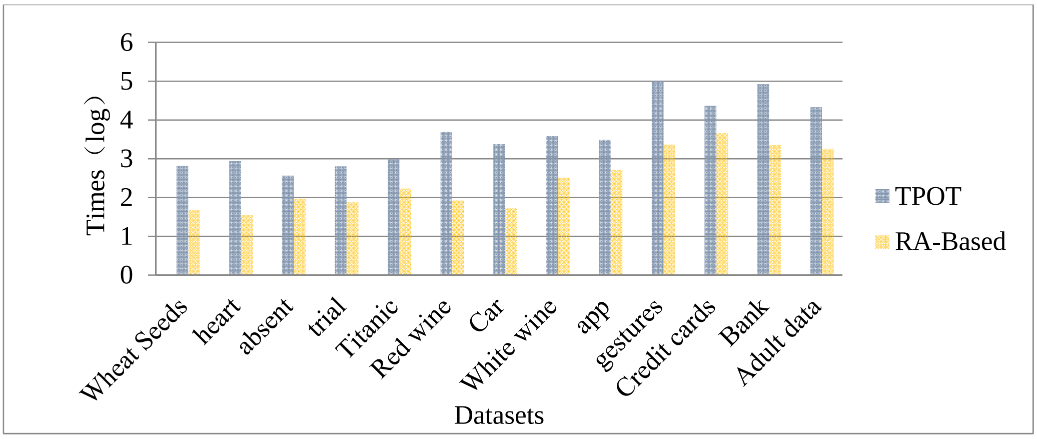


Figure 3b. Running time comparison (TPOT vs. RA-based).

Figure 4 shows our framework's selection of preprocessing methods and classifiers, with only the selected methods listed. As shown in Figure 4a, about 60% of the datasets selected data preprocessing, 40% of which chose the Binarizer method, 25% chose FastICA, 25% chose Polynomial Features, and the remaining 10% chose StandardScaler. As shown in Figure 4b, the selection of the classifier is shown, in which Gaussian naïve bayes are selected more frequently, followed by XGBoost, Logistic regression and extra trees. In addition to the data preprocessing and classification methods, the feature preprocessing method was selected in 70% of the datasets, and the Select we method was selected in all the datasets.

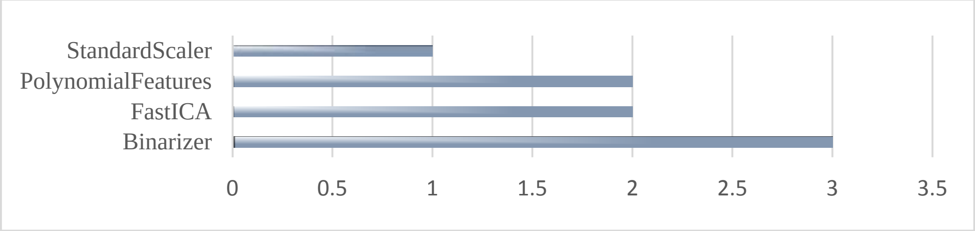


Figure 4a. Data pre-processing selection result.

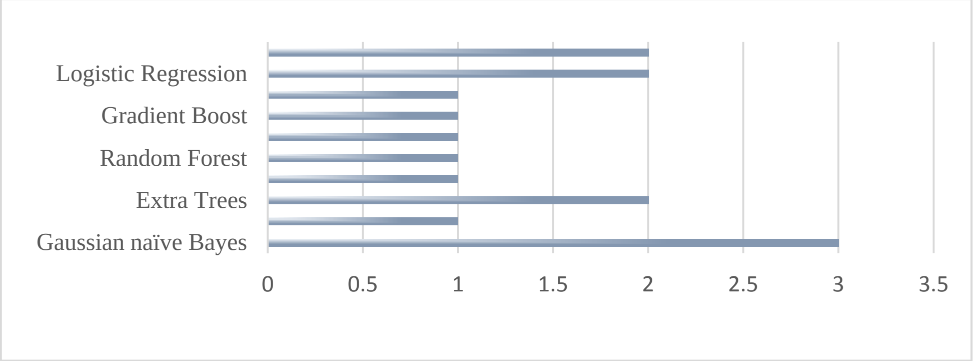


Figure 4b. Classifier selection result.

5. Conclusion

In this paper, we propose an automatic model selection framework based on reinforcement learning. This framework can solve the problem of time consumption when dealing with problems manually. We demonstrate the expressiveness of our framework on 13 datasets from different domains. By implementing an auto-tuning process, we optimize the whole pathway without human involvement, and this method can be extended to more datasets. Overall, the system is competitive with Human experiment for the datasets we tested on. The experiment results demonstrate that our system can achieve higher accuracy with less time cost.

In future work, we can improve our framework by optimizing the process of meta-learning data processing and improving the searching strategy of reinforcement learning.

References

[1] Hutter F, Caruana R, Bardenet R, Bilenko M, Guyon I, Kegl B, and Larochelle H. "AutoML 2014 @ ICML". AutoML 2014 Workshop @ ICML. Retrieved 2018-03 - 28.

[2] Yao Q, Wang M, Chen Y, et al. Taking Human out of Learning Applications: A Survey on Automated Machine Learning [J]. 2018.

[3] García S, Luengo J, Herrera F. Data Preprocessing in Data Mining [J]. Computer Science, 2000, 72.

[4] P. Brazdil, C. Giraud-Carrier, C. Soares, and R. Vilalta. Metalearning: Applications to Data Mining. Springer, 2009.

[5] Turner C R, Wolf A L, Fuggetta A, et al. Feature Engineering [C]. Iwssd. IEEE Computer Society, 1998.

[6] Guyon, Isabelle; Elisseeff, André (2003). "An Introduction to Variable and Feature Selection". JMLR. 3.

[7] Mladenic D, Grobelnik M. Feature Selection for Unbalanced Class Distribution and Naive Bayes [C]. Sixteenth International Conference on Machine Learning. 1999.

[8] Burnham K P, Anderson D R. Model Selection and Multimodel Inference[M]. Model selection and multimodel inference. 2002.

[9] Kadioglu S, Malitsky Y, Sabharwal A, et al. Algorithm selection and scheduling[C]. International Conference on Principles & Practice of Constraint Programming. 2011.

[10] Bergstra J, Bengio Y. Random search for hyper-parameter optimization [J]. Journal of Machine Learning Research, 2012, 13 (1): 281 - 305.

[11] AndrewCucchiara. Applied Logistic Regression [J]. Technometrics, 2012, 34 (3): 2.

[12] Sch C, Laptev I, Caputo B. Recognizing Human Actions: A Local SVM Approach[C]. International Conference on Pattern Recognition. 2004.

[13] Kononenko I. Semi-naive bayesian classifier [J]. Lecture Notes in Computer Science, 1991, 482: 206 - 219.

[14] C. Thornton, F. Hutter, H. Hoos, and K. Leyton-Brown. Auto-WEKA: combined selection and hyperparameter optimization of classification algorithms. In Proc. of KDD’13, pages 847–855, 2013.

[15] Kotthoff L, Thornton C, Hoos H H, et al. Auto-WEKA 2.0: Automatic model selection and hyperparameter optimization in WEKA [J]. The Journal of Machine Learning Research, 2017, 18 (1): 826 - 830.

[16] Feurer M, Klein A, Eggensperger K, et al. Efficient and robust automated machine learning [C]. Advances in neural information processing systems. 2015: 2962 - 2970.

[17] Olson R S, Urbanowicz R J, Andrews P C, et al. Automating biomedical data science through tree-based pipeline optimization[C]. European Conference on the Applications of Evolutionary Computation. Springer, Cham, 2016: 123 - 137.

[18] Thrun S, Littman M L. Reinforcement Learning: An Introduction [J]. IEEE Transactions on Neural Networks, 2005, 16 (1): 285 - 286.

[19] Lillicrap T P, Hunt J J, Pritzel A, et al. Continuous control with deep reinforcement learning [J]. arXiv preprint arXiv:1509.02971, 2015.

[20] Vilalta R, Drissi Y. A Perspective View and Survey of Meta-Learning[M]. A perspective view and survey of meta-learning. 2002: 77 - 95.

[21] P. Brazdil, C. Giraud-Carrier, C. Soares, and R. Vilalta. Metalearning: Applications to Data Mining. Springer, 2009.

[22] E. Brochu, V. Cora, and N. de Freitas. A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. CoRR, abs/1012.2599, 2010.

[23] M. Feurer, J. Springenberg, and F. Hutter. Initializing Bayesian hyperparameter optimization via metalearning. In Proc. of AAAI’15, pages 1128–1135, 2015.

[24] Reif M, F. Shafait, and A. Dengel. Meta-learning for evolutionary parameter optimization of classifiers. Machine Learning, 87:357 – 380, 2012.

[25] F. Hutter, H. Hoos, and K. Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In Proc. of LION’11, pages 507 – 523, 2011.

[26] J. Bergstra, R. Bardenet, Y. Bengio, and B. K´egl. Algorithms for hyper-parameter optimization. In Proc. of NIPS’11, pages 2546 – 2554, 2011.

[27] U. M. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. From data mining to knowledge discovery in databases. AI Magazine, 1996.

[28] M. A. Munson. A study on the importance of and time spent on different modeling steps. SIGKDD Explor. Newsl., 13 (2), May 2012.