International Journal of Electrical and Computer Engineering (IJECE)
Vol. 8, No. 5, October 2018, pp. 2994~3002
ISSN: 2088-8708, DOI: 10.11591/ijece.v8i5.pp2994-3002

An Architecture for Simplified and Automated Machine Learning

Jittapoo Poolwan, Sucha Smanchat
Faculty of Information Technology, King Mongkut's University of Technology North Bangkok, Thailand

Article Info

Article history:

Received Oct 30, 2017
Revised Jan 12, 2018
Accepted Aug 5, 2018

Keyword:

Auto ML
Data classification
Data mining
Machine learning

ABSTRACT

Recently, machine learning has been adopted by businesses to analyze their vast data in order to make strategic decision. However, knowledge in machine learning and technical skill are usually required to prepare data and perform machine learning tasks. This obstacle prevents smaller businesses with no technical knowledge to utilize machine learning. In this paper, we propose an architecture for simplified and automated machine learning process currently supporting the data classification task. The architecture includes a method for characterizing datasets, which allows for simplifying and automating machine learning model and hyperparameter selection based on historical execution configurations. Users can simply upload their datasets via a web browser, and the system will determine the possible models and their hyperparameter configurations for the users to choose from. The prototype shows the feasibility of the proposed architecture. Although the accuracy is still limited by the small execution history and the cleanliness of the input datasets, the architecture can minimize user involvement in the machine learning process so that non-technical users can perform data classification through a web browser without installing any software.

Copyright © 2018 Institute of Advanced Engineering and Science.
All rights reserved.

Corresponding Author:

Jittapoo Poolwan,
Faculty of Information Technology,
King Mongkut's University of Technology North Bangkok,
1518 Pracharat 1 Rd, Wongsawang, Bangsue, Bangkok, 11000, Thailand.
Email: j.poolwan@gmail.com

1. INTRODUCTION

In this modern world, businesses usually utilize data mining to help making sense of their data leading to strategic decision. Among different methods in data mining and analytics, machine learning can be employed to generate values from immense data such as by the data classification. However, it often requires technical skill and knowledge to implement and utilize machine learning. This obstacle prevents smaller businesses with limited technical knowledge to employ machine learning in order to compete in the markets.

Many machine learning services and software libraries have become available during the past few years such as Amazon Machine Learning and Tensor Flow [1]. These services attempt to ease the machine learning process for users by reducing human effort. However, tasks such as data preprocessing, selecting an appropriate machine learning model, and hyperparameter turning are non-trivial and still require technical skill to perform [2-4].

This paper presents an architecture design for a simplified and automated machine learning process for data classification, which is one of the most common data mining tasks. Our aim is to minimize user involvement so that end users, with limited knowledge and technical skill in machine learning, can simply upload their training data to the system via a web browser and use the auto-generated classification model to classify their future data without having to install any software. The system currently supports five classification models including K-nearest neighbors, decision trees, artificial neural network, support vector machine, and Naïve Bayes.

To achieve our objective, several components in machine learning process need to be automated [5]. Firstly, the system needs to know (or at least approximate) the type of each data field in a dataset uploaded by the user, and then prepares the dataset for machine learning process. At this stage, the dataset is assumed to be in tabular format and the last column is the class. Secondly, the system needs to select an appropriate machine learning model and its hyperparameter or offer choices to the user based on certain criteria such as the classification accuracy. The selection approach should also consider response time because end users may not be familiar with long processing time consumed by existing hyperparameter selection approaches [5-7]. Finally, the system should automatically build and train the selected machine learning model, and then provides an interface for users to perform classification on their data.

2. RELATED WORK

With the advancement in cloud computing, several machine learning services have been launched as cloud services over the past few years. For example, Amazon Machine Learning offers cloud-based machine learning service that supports model building and data prediction. It facilitates machine learning by providing tools for data transformation, model evaluation and visualization.

Apart from commercial services, machine learning libraries have also been published. Tensor Flow [1] is a graph-based dataflow for numerical computation. It facilitates machine learning, especially in neural network and deep learning, by providing programming library and machine learning dataflow process execution. Scikit-learn [8], available in Python programming language, is a machine learning library facilitating data mining and analytics tasks such as data preprocessing, data classification, data clustering, and model selection.

Although these services and software librarie are available, they merely facilitate the machine learning process and programming. Users still needs technical knowledge and skill to effectively utilize these services and libraries. To address this issue, few attempts have been made to automate machine learning process. Auto-WEKA [6, 7] proposes a method for machine learning model and hyperparameter selection. It evaluates different machine learning models for classification paired with varying associated hyperparameters based on sequential mode-based algorithm configuration (SMAC) to minimize the misclassification rate. The drawback of this method is the time taken to make the selection of an optimal model along with its hyperparameter. While acceptable for advanced users, this duration may not be satisfactory for end users.

Auto-sklearn [5], based on Scikit-learn library, is a tool for an automation of machine learning process emphasizing on model selection and hyperparameter tuning. This tool employs Bayesian optimization to find an optimal machine learning model and its hyperparameters by iterating across classification algorithms and preprocessors available in the system. Although efficient in terms of accuracy, this method is computationally expensive as it has to train and test available models and preprocessors across domains of hyperparameters (although not entirely) for each new dataset. This may lead to latency in the automation, which may not be friendly to end users. In addition, users need to be able to clean their data before feeding them to the process as the input data are used to generate meta-features that are employed in model selection.

Despite these efforts, there is still a shortcoming in the automation of machine learning process for end users. Technical skills are still required of users when determining and preprocessing different types of data. Eventhough Auto-sklearn offers a high performing hyperparameter tuning technique, it is still not simplified for non-technical users. Thus, we aim to go further by automating these technical components so that end users with non-technical background can easily apply machine learning, specifically for data classification in this work, to their businesses.

3. RESEARCH METHOD

As mentioned earlier, to automate the machine learning process, it is necessary to automate several steps including the approximation of data type, dataset preprocessing, machine learning model selection, and model building and execution. In this work, five classification models including K-nearest neighbors, decision trees (C4.5), artificial neural network, support vector machine, and Naïve Bayes are considered. These five models represent different categories of machine learning techniques [9, 10]. We propose an architecture with five components to address these issues as depicted in Figure 1.

3.1. Data Scanner

The first task in machine learning is the preparation of dataset. We assume that end user has limited technical knowledge so the system has to approximate the types of data in the input dataset. Data Scanner in

the proposed architecture handles this task by scanning the input dataset [11, 12]. For small datasets, it is possible to scan the whole datasets. However, it is not practical for datasets of very large size leading to long latency. Thus, samples are randomly selected from the dataset for scanning. The size of the sample is defined by $S=1000ln(N)$ where N is the size of a given dataset. For example, if a dataset contains one million records, the sample size would be 13816 records.

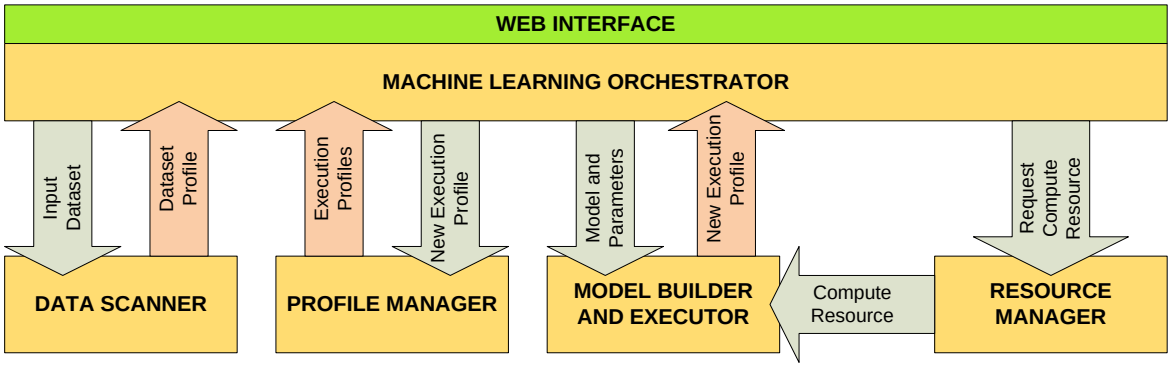

Figure 1. Architecture overview

We characterize each data field into three aspects, namely, data type, data scale, and data format. Data types considered in the proposed architecture are basic data types including integer, floating point, string and boolean. Data scales are the data measurement scales: nominal, interval and ratio scales. Finally, data formats include *binary*, *day of month*, *day of week*, *day_string*, *month*, *month_string*, *gender*, *hour24*, *hour12*, *minute*, *year*, and *quarter*. We choose these data formats as they are commonly found in many datasets in the UCI repository [13] in order to create dataset profile.

When scanning, each data field is assigned a type where applicable in the respectively order of integer, floating point, and string. For example, “machine101” would be assigned as string because it cannot be converted into integer or floating point. Boolean type is assigned when every instance of a given data field takes the value implying boolean such as 0, 1, true, FALSE, t, and F.

Regarding data scale, if a data field is previously identified as boolean or string, then it is assigned to the nominal scale. For an integer or a floating point, if there exist a negative value, then it is assigned to the interval scale; otherwise it is assigned to the ratio scale. The reason for excluding the ordinal scale is that determining whether a data field represents an order requires very specific methods for different data types.

A data field is assigned to one or more data formats if all of its values meet the criteria as specified in Table 1. This is intended as to cover the different possibilities of the dataset for determining similarity between datasets later on. From the three aspects, there are 19 data characteristics in total that are used to construct dataset profile.

Table 1. Data Format Criteria

Data Format	Criteria
binary	Any data type with only two different values
day_of_month	Integer value in the range of 1-31
day_of_week	Integer value in the range of 1-7
day_str	String representing days of week such as “Sunday” and “sat”
Month	Integer value in the range 1-12
month_str	String representing months of year such as “April” and “feb”
gender	String representing gender such as “male”, “m”, “FEMALE”, and “F”
hour24	Integer value in the range 0-23
hour12	Integer value in the range 0-11
minute	Integer value in the range 0-59
year	Integer value in the range 1800-2100
quarter	Integer value in the range 1-4

From our early test by repeating the scan of the same datasets on five different datasets in the UCI repository, shown in Table 2, it is found that this approach with the defined sample size mentioned earlier can consistently approximate the types of data.

3.2. Profile Manager

In the proposed architecture, the selection of machine learning model and its hyperparameters utilizes the history of model training execution on previous datasets. The Profile Manager component manages these execution profiles, which are grouped according to each machine learning model (i.e. there are five groups of profiles for the five supported classification models). Each execution profile is composed of a *dataset profile*, *hyperparameters* used, *resource utilization*, and *execution result* as shown in Figure 2.

Figure 2. Execution profile

Because we want to allow end users to simply upload their datasets, the datasets may not be clean and appropriate for machine learning process. Thus, we cannot apply sophisticated meta-features such as those used in Auto-sklearn [5] to characterize datasets. In our approach, the *dataset profile* contains the name of the data file and a set of 19 integers that are the numbers of occurences of the 19 data characteristics described in the previous section. For example, the famous "Iris" dataset contains five data fields: four floating points and one string. Subsequently, four ratio-scale data and one nomial-scale data are possible. As for the data formats, these five data fields in the Iris dataset do not match any defined data format. Thus, the profile of the "Iris" dataset is as shown in Table 2, along with the examples of four other datasets.

Table 2. Examples of dataset profiles

File Name	integer	float	string	boolean	nominal	interval	ratio	binary	day_of_month	day_of_week	day_str	month	month_str	gender	hour24	hour12	minute	year	quarter
Abalone	1	7	1	0	1	0	8	0	1	0	0	0	0	1	1	0	1	0	0
Adult	6	0	9	0	9	0	6	2	1	0	0	0	0	1	1	0	1	0	0
Iris	0	4	1	0	1	0	4	0	0	0	0	0	0	0	0	0	0	0	0
Wine-quality-white	1	11	0	0	0	0	12	0	1	0	0	1	0	0	1	1	1	0	0
Yeast	0	8	2	0	2	0	8	1	0	0	0	0	0	0	0	0	0	0	0

Hyperparameters in the execution profile are specific to each machine learning model of each execution. Only important hyperparameters, as listed in Table 3, are currently available for adjustment during hyperparameter selection in this work to avoid complexity. Additional hyperparameters can simply be included in future work by adding to the execution profile. *Resource utilization* is composed of CPU utilization, memory utilization, and processing time during model building. *Execution result* contains the values of accuracy, recall, precision, and F-measure from the model validation. An example of execution profile of KNN model is shown in Figure 3.

Table 3. Hyperparameters for the Support Models

Model	Parameters
K-Nearest Neighbors	K-value, weight type (uniform or distance)
Decision Trees	metric (entropy or gini), splitter
Artificial Neural Network	number of hidden nodes, number of hidden layers
Support Vector Machine	number of class value, kernel (rbf, linear, poly, or sigmoid)
Naïve Bayes	none

Data Profile	Hyper Parameters	Resource Utilization	Execution Result
0,9,1,0,1,0,9,0,0,0,0,0,0,0,0,0,0,0,0	10,uniform	1,17.381,1.181,1.181,20.538	0.493,0.493,0.463,0.418
0,9,1,0,1,0,9,0,0,0,0,0,0,0,0,0,0,0,0	11,uniform	1,17.381,1.324,1.324,23.014	0.409,0.409,0.402,0.384
0,9,1,0,1,0,9,0,0,0,0,0,0,0,0,0,0,0,0	12,uniform	1,17.381,1.465,1.465,25.474	0.449,0.449,0.489,0.442
0,9,1,0,1,0,9,0,0,0,0,0,0,0,0,0,0,0,0	13,uniform	1,17.381,1.607,1.607,27.934	0.409,0.409,0.429,0.394
0,9,1,0,1,0,9,0,0,0,0,0,0,0,0,0,0,0,0	14,uniform	1,17.381,1.749,1.749,30.412	0.440,0.440,0.521,0.445
0,9,1,0,1,0,9,0,0,0,0,0,0,0,0,0,0,0,0	15,uniform	1,17.381,1.904,1.904,33.101	0.430,0.430,0.470,0.420
0,9,1,0,1,0,9,0,0,0,0,0,0,0,0,0,0,0,0	16,uniform	1,17.381,2.047,2.047,35.581	0.409,0.409,0.426,0.389
0,9,1,0,1,0,9,0,0,0,0,0,0,0,0,0,0,0,0	17,uniform	1,17.381,2.189,2.189,38.052	0.409,0.409,0.443,0.390
0,9,1,0,1,0,9,0,0,0,0,0,0,0,0,0,0,0,0	18,uniform	1,17.381,2.331,2.331,40.526	0.410,0.410,0.455,0.389
0,9,1,0,1,0,9,0,0,0,0,0,0,0,0,0,0,0,0	19,uniform	1,17.381,2.474,2.474,43.001	0.430,0.430,0.461,0.409

Figure 3. An example of execution profile

3.3. Machine Learning Orchestrator

The Machine Learning Orchestrator is the main component that manages the machine learning process by interacting with other components. When a user uploads a new dataset into the system, Machine Learning Orchestrator invokes the Data Scanner to scan the dataset and create its data profile as described earlier. The next step is to determine appropriate models and the associated hyperparameters. To avoid the latency caused by fine-tuning approaches as in existing work [4, 5, 7], we employ a simpler approach of selecting the model and its parameter from historical execution profiles to achieve faster response, although possibly sacrificing optimal accuracy.

The execution profiles of each of the five classification models are first clustered using K-Mean clustering [14]. K-Mean clustering is employed because of its fast processing time in order to archive fast response time for general users [15]. K is set to the number of unique dataset profiles that have been trained previously with each machine learning model (regardless of the names of the datasets) so that the resulting clusters are unique based on the 19 data characteristics. Subsequently, the profile cluster with the minimum Euclidean distance from the new dataset profile is selected. The Euclidean distance *dist* is calculated as shown in Equation 1 where do_i are the dataset characteristics of a cluster and dn_i are those of the new dataset.

$$dist = \sqrt{\sum_{i=1}^{19} (do_i - dn_i)^2} \quad (1)$$

Hyperparameters are selected from the execution profiles in the profile cluster with the minimum *dist*. For each profile cluster of the five machine learning models, the profile with the highest accuracy is selected. Finally, these five hyperparameter configurations are presented to the user along with the accuracy and processing time. The user can then simply select one of the models or the system can automatically select the one with the highest accuracy. The selected configuration will then be executed by the Model Builder and Executor component. The model selection process is depicted in Figure 4.

In this approach, hyperparmeter selection relies heavily on execution history. Thus, it is necessary to have a large number of historical execution profiles with different hyperparameters in order for the system to offer better choices for the user. Initially, the execution profiles have been created by training the five classification models with different hyperparameters on a number of datasets from the UCI repository. The pool of execution profiles will continue to grow as users use the system. Also, the Machine Learning Orchestrator will be modified to automatically build more profiles of different models and hyperparameters on the available datasets when the system becomes idle. Once the model training is completed, this component generates a graphical user interface on a web browser with input fields corresponding to the input dataset for data classification based on the finished model returned by the Model Builder and Executor component. Users can then perform data classification on their own through this interface.

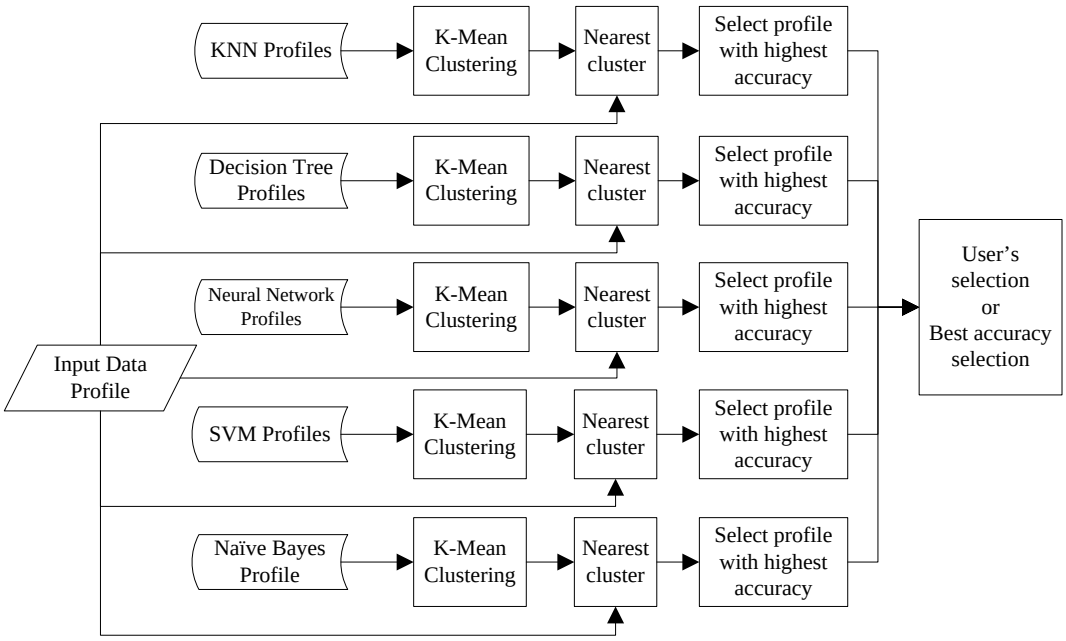

Figure 4. Model selection process

3.4. Model Builder and Executor

This component takes the input dataset and the selected hyperparameter configuration to train the selected machine learning model. The input dataset is first preprocessed: any record with missing value is excluded from the model building; boolean values, binary values and genders are converted to 0 or 1; strings representing days of week are converted to 1-7; strings representing months are converted to 1-12; other string values are converted into integers.

The preprocessed dataset is then used to train the selected model. The Scikit-learn library [8] is used for model building using 10-fold cross validation. Once the training is completed, the performance information including the classification accuracy, recall, precision, F-measure, processing duration, CPU utilization, and memory utilization is recorded (along with the model and its hyperparameters) as *execution profile* in the Profile Manager. The resulting model is then returned to the Machine Learning Orchestrator to generate a GUI for the users.

3.5. Resource Manager

Resource Manager component manages the computation resources that are used in the system. This component aims to support machine learning execution in the cloud. Currently, the prototype is being test on Amazon EC2. When a user uploads a new dataset, it is directed to Amazon S3 for storage. Once the user starts working on the dataset, it is staged onto the EC2 instance and then the model building is performed. At this stage, the user can request to run machine learning process on a separate EC2 instance. However, advanced cloud features such as autoscaling and resource allocation have not been included in our scope.

4. RESULTS AND DISCUSSION

The prototype of the proposed architecture is implemented in Python language and employs the Scikit-learn library [8] for machine learning programming. For the evaluation of our approach to automate machine learning process, we test the prototype using the datasets from the UCI repository. To initialize the prototype for the test, execution profiles in the Profile Manager must first be populated. For this purpose, 24 datasets for classification from the UCI repository are used to train the five supported machine learning models with varying combinations of hyperparameters, resulting in approximately 4000 profiles.

Another 31 datasets are then used to test the prototype. These 31 datasets simulate unseen datasets from users, each of which will go through K-Mean clustering to find the profile cluster with the minimum Euclidean distance. Then, the profile with the highest accuracy in the cluster of each classification model as mentioned earlier will be selected. The final selection is the classification model and its hyperparameters with

the highest accuracy among the five models. During this test, the time taken for the system to select a model and hyperparameters through K-Mean clustering of execution profiles is negligible.

For each of the 31 datasets, we report in Table 4 the selected model, its accuracy from the execution profile, and the accuracy of the model when taking the dataset in question as the new input. In addition, we also run each of the 31 datasets through the other four classification models for comparison. The last column in Table 4 indicates whether the selected model actually performs best among the five models; the best model and accuracy are otherwise supplied.

From Table 4, the prototype can select the best classficiation model for 20 out of 31 datasets. Among these 20 datasets, the actual accuracy is at least 5 percents worse than previous executions in 8 datasets (highlighted in bold type). There is one dataset, *sensor_readings_24*, where the actual accuracy is higher. This sacrifice of accuracy is expected because our approach does not perform fine-tuning of the hyperparameters and feature selection to adjust the model to new datasets in order to avoid latency in automating model selection.

As for the other 11 datasets for which the prototype does not select the best model, the actual accuracy values are worse to higher degrees. We attribute this mainly to the small size of execution profiles, which limits the possible combinations of hyperparameters. Another possible reason is that the accuracy is affected by the datasets themselves because the system allows non-technical users to upload datasets without cleaning them. As we probe the datasets with low accuracy, it is found that the *Wholesale customers data - Region* dataset contains only 439 records, which are insufficient to train an accurate model. The *tae* dataset contains only 150 records and the *SPECTF Heart - train* dataset contains only 79 records of binary values. The *MONK's Problems* dataset contains only 414 records with a column of descriptive strings. The *turkiye-student-evaluation_generic* dataset contains 5-point Likert Scale ordinal data, which are not yet handled properly by the Data Scanner.

Table 4. Evaluation Result

Dataset Name	Selected Model	Profile Accuracy	Accuracy With New Dataset	Difference	Best of Five
Breast Tissue	Naïve Bayes	0.763	0.656	**-0.107**	Yes
Ecoli	Naïve Bayes	0.842	0.830	-0.012	Yes
Lung Cancer	Naïve Bayes	0.644	0.580	**-0.064**	Yes
MONK's Problems	Naïve Bayes	0.632	0.626	-0.006	Yes
Parkinsons	Decision Tree	0.877	0.820	**-0.057**	Yes
SPECTF Heart2 - train	Naïve Bayes	0.840	0.758	**-0.082**	Yes
Statlog (German Credit Data)-numeric	Naïve Bayes	0.744	0.723	-0.021	Yes
Statlog (German Credit Data)	Naïve Bayes	0.716	0.731	+0.015	Yes
Statlog (Heart)	Naïve Bayes	0.846	0.836	-0.010	Yes
Statlog (Vehicle Silhouettes)	Decision Tree	0.714	0.680	-0.034	Yes
Urban Land Cover - training	Naïve Bayes	0.821	0.765	**-0.056**	Yes
Wholesale customers data -Chanal	Naïve Bayes	0.905	0.902	-0.003	Yes
Wilt - training	KNN	0.985	0.990	+0.005	Yes
Yeast	Naïve Bayes	0.577	0.511	**-0.066**	Yes
Zoo	Naïve Bayes	0.968	0.950	-0.018	Yes
leaf	Naïve Bayes	0.775	0.716	**-0.059**	Yes
pendigits - train	KNN	0.986	0.990	+0.004	Yes
sensor_readings_24	Decision Tree	0.877	0.960	***+0.083***	Yes
tae	Decision Tree	0.640	0.660	+0.020	Yes
vertebral_column_data_3C	KNN	0.985	0.840	**-0.145**	Yes
Iris	Decision Tree	0.993	0.950	-0.043	Neural Network (0.970, -0.023)
Lenses	Naïve Bayes	0.950	0.850	**-0.100**	Decision Tree (0.880, -0.07)
MAGIC Gamma Telescope	Naïve Bayes	0.763	0.727	-0.036	Neural Network (0.822, +0.059)
SPECTF Heart - train	Naïve Bayes	0.805	0.685	**-0.120**	SVM (0.740, -0.065)
Waveform	Decision Tree	0.877	0.75	**-0.127**	KNN & NN (0.850, -0.027)
Wholesale customers data -Region	Naïve Bayes	0.905	0.478	**-0.427**	KNN & SVM (0.720, -0.185)
sensor_readings_4	KNN	0.985	0.96	-0.025	Decision Tree (0.990, +0.005)
student-mat	SVM	0.939	0.94	+0.001	KNN (0.950, +0.011)
student-por	SVM	0.939	0.88	**-0.059**	KNN (0.890, -0.049)
turkiye-student-evaluation_generic	KNN	1.00	0.32	**-0.680**	SVM (0.370, -0.630)
winequality-white	KNN	0.632	0.45	**-0.182**	Decision Tree (0.630, -0.002)

5. CONCLUSION

This paper proposes an architecture for the automation of machine learning process so that non-technical users can perform data classification task. Unlike existing systems, which focus mostly on model selection and hyperparameter tuning, we also allow users to upload data without cleaning and preprocessing

them first. For this requirement, a dataset profiling method is proposed so that it is possible to find similar datasets through K-Mean clustering from previous executions and select the classification model and its hyperparameters accordingly. With our approach, a user can simply upload training data, chooses among the classification models that have been determined by the system, and then performs the data classification task on their future data through web browser.

The evaluation based on our prototype running on Amazon EC2 shows the feasibility of the approach and that it can select classification models based on similiar datasets previously processed with negligible response time. However, at this early stage, the accuracy of the classification is limited by previous execution configurations. Also, for uncleaned data, the accuracy may becomes unpredictable. As future work, the system will perform automatic model training using different machine learning models and varying hyperparameter combinations on available datasets when the system is idle in order to improve the model and hyperparameter selection. Also, it is possible to include a feature selection step after getting the model and hyperparameters from the previous executions to improve accuracy. Finally, for the scalability of the system, the execution of machine learning can be improved with appropriate resource allocation so that they can run on multiple virtual machines in the cloud.

REFERENCES

[1] M. Abadi, *et al.* (2015). *TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems*.

[2] F. Hutter, H. H. Hoos and T. Stützle, "Automatic algorithm configuration based on local search", in *AAAI*, 2007, pp. 1152-1157.

[3] J.S. Bergstra, *et al.*, "Algorithms for hyper-parameter optimization", *Advances in Neural Information Processing Systems*, vol. 24, pp. 2546-2554, 2011.

[4] B. Komer, J. Bergstra and C. Eliasmith, "Hyperopt-sklearn: automatic hyperparameter configuration for scikit-learn", in *Proceedings of the 13th Python in Science Conference (SciPy 2014)*, 2014.

[5] M. Feurer, *et al.*, "Efficient and robust automated machine learning", *Advances in Neural Information Processing Systems (NIPS 2015)*, vol. 28, pp. 2962-2970, 2015.

[6] L. Kotthoff, *et al.*, "Auto-WEKA 2.0: Automatic model selection and hyperparameter optimization in WEKA", *Journal of Machine Learning Research*, vol. 18, pp. 1-5, 2017.

[7] C. Thornton, *et al.*, "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*, Chicago, Illinois, USA, 2013, pp. 847-855, doi: 10.1145/2487575.2487629.

[8] F. Pedregosa, *et al.*, "Scikit-learn: Machine Learning in Python", *Journal of Machine Learning Research*, vol. 12, pp. 2825-2830, 2011.

[9] M. Abdar, *et al.*, "Comparing Performance of Data Mining Algorithms in Prediction Heart Diseases", *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 5, pp. 1569-1576, 2015.

[10] S.B. Kotsiantis, I.D. Zaharakis and P.E. Pintelas, "Machine learning: a review of classification and combining techniques", *Artificial Intelligence Review*, vol. 26, pp. 159-190, November 01 2006, doi: 10.1007/s10462-007-9052-3.

[11] S.F. Qiang Guan, "auto-AID: A data mining framework for autonomic anomaly identification in networked computer systems", *Performance Computing and Communications Conference (IPCCC)*, 2010, doi: 10.1109/PCCC.2010.5682334.

[12] S. Wang, *et al.*, "Early detection of numerical typing errors using data mining techniques", *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, vol. 41, pp. 1199-1212, 2011.

[13] *The UCI Machine Learning Repository*. Available: https://archive.ics.uci.edu/ml/datasets.html Accessed 2016.

[14] R.N.V.J. Mohan, R.S. Rao and K.R.S. Rao, "Efficient K-Means Cluster Reliability on Ternary Face Recognition Using Angle Oriented Approach", *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 2, 2013, doi: http://dx.doi.org/10.11591/ij-ict.v2i1.1779.

[15] B.S. Chandana, K. Srinivas and R.K. Kumar, "Clustering Algorithm Combined with Hill Climbing for Classification of Remote Sensing Image", *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 4, pp. 923-930, 2014, doi: http://dx.doi.org/10.11591/ijece.v4i6.6608.

BIOGRAPHIES OF AUTHORS

Jittapoo Poolwan received his master's degree in Information Technology from King Mongkut's University of Technology North Bangkok, in 2008. He is currently pursuing Ph.D. degree in Information Technology at the same institute. His research interests are data mining, machine learning and online intelligent forecasting system.

Sucha Smanchat is currently a lecturer at the Faculty of Information Technology, King Mongkut's University of Technology North Bangkok, Thailand. He obtained his PhD degree at Monash University in Melbourne, Australia. During his study, he was involved with the development of a prototype scheduler for Nimrod/K system. His current research interests are in cloud workflow scheduling techniques, and machine learning and algorithms for bioinformatics.