Journal of Cloud Computing:
Advances, Systems and Applications

RESEARCH

Open Access

Algorithm selection using edge ML and case-based reasoning

Rahman Ali[1], Muhammad Sadiq Hassan Zada[2], Asad Masood Khatak[3] and Jamil Hussain[4*]

Abstract

In practical data mining, a wide range of classification algorithms is employed for prediction tasks. However, selecting the best algorithm poses a challenging task for machine learning practitioners and experts, primarily due to the inherent variability in the characteristics of classification problems, referred to as datasets, and the unpredictable performance of these algorithms. Dataset characteristics are quantified in terms of meta-features, while classifier performance is evaluated using various performance metrics. The assessment of classifiers through empirical methods across multiple classification datasets, while considering multiple performance metrics, presents a computationally expensive and time-consuming obstacle in the pursuit of selecting the optimal algorithm. Furthermore, the scarcity of sufficient training data, denoted by dimensions representing the number of datasets and the feature space described by meta-feature perspectives, adds further complexity to the process of algorithm selection using classical machine learning methods. This research paper presents an integrated framework called eML-CBR that combines edge edge-ML and case-based reasoning methodologies to accurately address the algorithm selection problem. It adapts a multi-level, multi-view case-based reasoning methodology, considering data from diverse feature dimensions and the algorithms from multiple performance aspects, that distributes computations to both cloud edges and centralized nodes. On the edge, the first-level reasoning employs machine learning methods to recommend a family of classification algorithms, while at the second level, it recommends a list of the top-k algorithms within that family. This list is further refined by an algorithm conflict resolver module. The eML-CBR framework offers a suite of contributions, including integrated algorithm selection, multi-view meta-feature extraction, innovative performance criteria, improved algorithm recommendation, data scarcity mitigation through incremental learning, and an open-source CBR module, reshaping research paradigms. The CBR module, trained on 100 datasets and tested with 52 datasets using 9 decision tree algorithms, achieved an accuracy of 94% for correct classifier recommendations within the top k=3 algorithms, making it highly suitable for practical classification applications.

Keywords Algorithm selection, Machine learning, Meta learning, Edge ML, Edge computing

Introduction

In data mining, prediction problems are commonly and frequently encountered, and they are effectively addressed through the utilization of machine learning algorithms. Machine learning experts have devised and created a large number of algorithms tailored for data mining applications [1]. Similarly, to improve classification accuracy, especially on difficult problems, researchers are developing new and innovative methods for combining and designing new classifiers, such as ensemble or exploit the intrinsic structure classes [2].

*Correspondence:
Jamil Hussain
jamil@sejong.ac.kr
[1] Quaid-e-Azam College of Commerce, University of Peshawar, Peshawar, Pakistan
[2] University of Derby, Kedleston Rd, Derby, United Kingdom
[3] College of Technological Innovation, Zayed University, Abu Dhabi, UAE
[4] Department of Data Science, Sejong University, Seoul, South Korea

This makes the algorithm space vast, which is categorized into distinct families, encompassing decision trees, Bayes classifiers, rule-based learners, meta-learners, multi-instance classifiers, and function classifiers and lazy classifiers [3]. Among these categories, decision tree-based algorithms are widely acknowledged and employed as a prominent data classification technique due to their efficiency, user-friendly nature, and comparatively modest complexity [4]. In various applications, including medicine, finance, public policy, and more, they serve as valuable tools for facilitating decision-making [5].

Picking an appropriate classification algorithm (a decision tree in this case) for a given dataset is a difficult and challenging task not only for end users but also for machine learning practitioners [6]. This is because different algorithms behave differently on the same problem due to the inherent characteristics of the data. For example, some algorithms are better suited for linear data, while others are better suited for non-linear data. Additionally, some algorithms are more computationally efficient than others [7].

One approach involves empirically evaluating all available classifiers for a given classification problem and selecting the one with the best results as the winner [7, 8]. However, this method faces the challenge of exhaustive search, leading to computational complexity [9]. Several studies have demonstrated that there isn't a universal classification algorithm suitable for all classification problems. For instance, if the same classifier is applied to a different problem, it may yield suboptimal results, thus affirming the established "No Free Lunch" theorem [10]. The reason is that the results of classifiers depend on the specific characteristics of each problem, making the task of classifier selection as a meta-learning approach [11]. In this approach, the meta-characteristics of classification problems are computed, and classifiers are evaluated based on their performance on these problems. Subsequently, a mapping is learned between problem features and the classifier(s) that exhibit the best performance, enabling the recommendation of an appropriate classifier [12]. Hence, the task of automatic algorithm selection through meta-learning can be represented as a four-step process model [13]. This model encompasses the following stages: classifier characterization, where classifier performance is assessed; problem characterization, involving the extraction of inherent meta-characteristics of the problem; the mapping and learning of problem meta-characteristics against classifier performance, and ultimately the recommendation of suitable classifier(s) for a new problem.

Classifier characterization represents the user's objective in developing an application, such as accuracy or computational efficiency. It can be assessed using performance evaluation metrics [14]. The research community has approached classifier characterization from both uni-metric and multi-metric perspectives, often referred to as meta-target. Problem characterization involves extracting the underlying data behaviors that reflect its unseen nature, measured through meta-features or meta-characteristics. Various types of meta-characteristics have been identified, including statistical, information theoretic, model-based, land-marking, and complexity [15, 16].

Recently, Q. Song et al. [17] has used a new dataset characterization method for computing datasets features and computed performance of seventeen classification algorithms over 84 UCI publically available datasets [18]. Mapping meta-characteristics and classifiers performance is the process of aligning each problem against the appropriate classifier. The objective of the process is to make algorithm selection problem as a machine learning problem where meta-characteristics form a feature-vector and label(s) of the classifier(s), with best performance, as the class label. Identification of the class-label is a challenging task and researchers have approached the issue using various approaches, such as multiple comparison method. As a result of these methods, some of the problems have more than one applicable classifiers as the class label. This makes the problem of algorithm selection is a single-class and multi-class problem and research community has approached them using single-label learning and multi-label learning. For learning association or mapping function between problems meta-characteristics and class label(s), researcher have used different approaches that can broadly be categorized – define categories - as decision tree-based learner (e.g., C4.5 [19]), rule-based learner [20], linear regression [21] and instance-based learner (e.g., k-NN [17, 22]). Finally, for the selection of appropriate classifier(s) on the fly, researchers have used different approaches.

In the realm of classification algorithm selection, meta-learning has been employed extensively. EFFECT [23] offers an interpretable meta-learning framework to explain recommendation results and algorithm performance in specific business scenarios. AMLBID [24] automates algorithm selection for analyzing industrial data, while [15, 16] utilizes meta-learning to assess data characteristics and recommend algorithms across various datasets.

Summary of the existing research work on automatic algorithm selection reveals several limitations. One significant challenge is the computational complexity involved in empirically assessing classifiers across

multiple datasets and employing various performance metrics. This complexity can make the algorithm selection process time-consuming and resource-intensive, particularly for large datasets. Another limitation stems from the scarcity of sufficient training data required for effective machine learning, including the representation of diverse datasets and their associated meta-features. Additionally, the unpredictable performance of classification algorithms on different datasets adds an element of uncertainty to the selection process. Furthermore, the variability in dataset characteristics, such as data distribution and feature space complexity, makes finding a universally optimal algorithm difficult. Lastly, determining appropriate performance metrics for evaluation can be a challenging task, as different datasets may necessitate different metrics for meaningful assessment. These limitations underscore the need for innovative approaches like meta-learning to address the complexities inherent in algorithm selection for data mining.

To address the issues highlighted earlier on the subject problem of algorithm selection, this research paper proposes an integrated framework called eML-CBR that combines edge machine learning (ML) and case-based reasoning (CBR) methodologies. It adapts a multi-level, multi-view CBR methodology that considers data from diverse feature dimensions and algorithms from multiple performance aspects. The computation is distributed to both cloud edges and centralized nodes. On the edge, the first-level reasoning employs machine learning methods to recommend a family of classification algorithms, while at the second level, it recommends a list of the top-k algorithms within that family. This list is further refined by an algorithm conflict resolver module.

Key contributions of the research work are enlisted as follows.

The eML-CBR boasts several key contributions that are set to transform the field:

- Pioneering the design and development of an integrated algorithm selection framework that seamlessly integrates edge-ML with CBR methodology.
- Thoroughly exploring and extracting multi-view meta-features from datasets to provide a deeper insight into their intricacies.
- Devising a new multi-objective criteria with weighted summation to accurately assess algorithm performances.
- Enhancing algorithm recommendation significantly through the integration of the algorithm conflict resolver (ACR).
- Mitigating the persistent challenge of data scarcity by introducing the concept of incremental evolutionary learning using the CBR methodology.
- Releasing the CBR module as a stand-alone open-source software to the research community in the field.

The remainder of this paper is structured as follows: Section "Edge ML and Case-based Reasoning for Algorithm Selection" provides an in-depth overview of Edge ML and Case-based Reasoning for Algorithm Selection. In Section "Multi-views Case-based Reasoning for Algorithm Selection", we delve into the Multi-views Case-based Reasoning methodology applied to Algorithm Selection. Section "Implementation, experiments and evaluation" is dedicated to the implementation, experiments, and evaluation of our proposed methodology. Finally, in Section "Conclusion and Future Work", we conclude our work and outline potential avenues for future research.

Edge ML and case-based reasoning for algorithm selection

Definition of algorithm selection problem

Based on the Rice model [25], the algorithm selection problem can be defined as follows: given a problem p as input and a set of candidate machine learning algorithms A that can learn the same p with varying performance levels Y, the objective of an algorithm selection method is to find and select an algorithm $a \in A$ that can learn p with the best possible performance. Now, we introduce the notation that will be used throughout this paper. Let P denote a set of historical problems (i.e., classification datasets in this case) with F as the feature vector representing the meta-features of each problem $p \in P$, and let A be a set of classification algorithms capable of solving P with some performance level Y.

Algorithm selection as an edge ML problem

Edge machine learning, abbreviated as edge ML, involves executing machine learning algorithms on edge devices located close to the data source rather than on remote cloud servers, enabling faster decision-making. In the proposed study, the algorithm selection problem is divided into two levels, taking place at both local and remote nodes within the distributed system. At the first level, we select the appropriate family of ML algorithms, including probabilistic, decision tree, function-based, lazy learners, meta-learners, and rules-based families. At the second level, we choose the appropriate ML algorithm on the remote cloud server. In both levels, we employ a case-based reasoning methodology, rendering the algorithm selection process a multi-level reasoning process.

Overview of the proposed Edge ML computing environment is shown in Fig. 1.

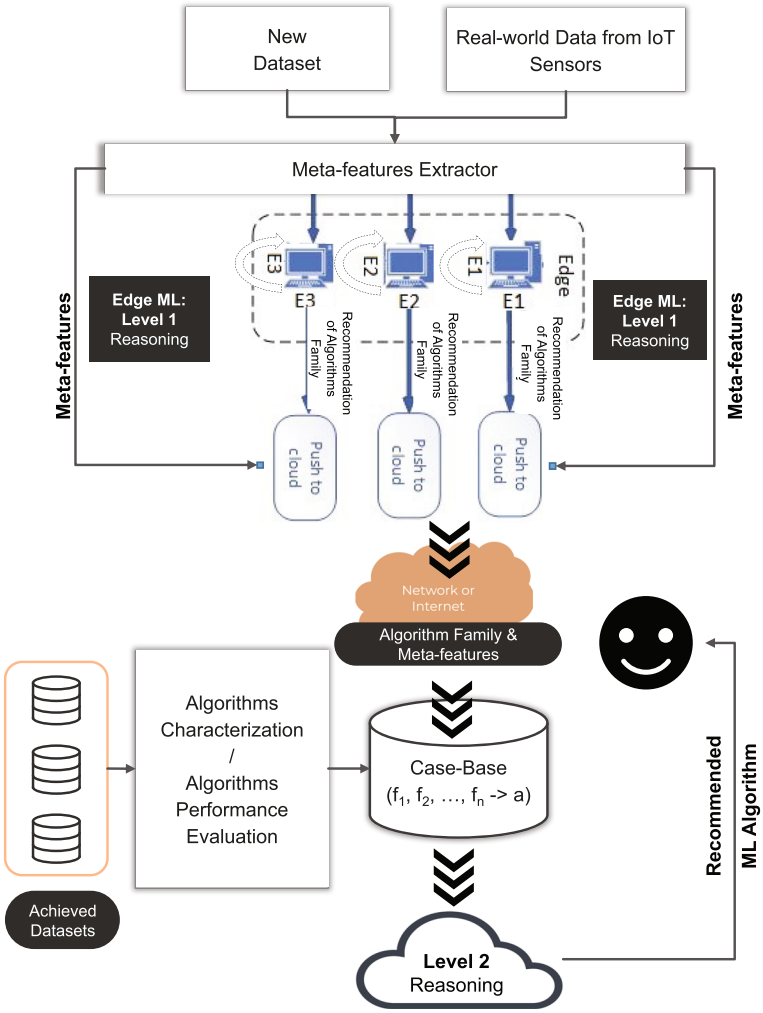

Fig. 1 Overview of the proposed edge ML computing environment

The proposed methodology for edge ML computing in ML algorithm selection is supported by a hierarchical machine learning approach. In this approach, the first layer of the hierarchy recommends a family of algorithms, and the second layer of the hierarchy selects the appropriate algorithm within the chosen family. In this framework, both the edge device and the cloud server act as computing nodes, both locally and remotely. The paper will now focus on the case-based reasoning methodology used on both the edge and cloud devices.

Multi-views case-based reasoning for algorithm selection

As previously discussed, algorithm selection is an edge ML computing problem wherein case-based reasoning (CBR) is employed at both the local device and the centralized cloud server. This process encompasses three key modules: (i) datasets and algorithms characterization, (ii) model creation, and (iii) algorithm recommendation. CBR utilizes diverse families of data characteristics, rendering it a multi-view CBR approach. An architectural depiction of the proposed framework, inspired by the Rice framework [12], is presented in Fig. 2.

The subsequent section explains the workings of each module, with its technical details and the methods used.

Datasets and Algorithms Characterization (DAC)

The datasets and algorithms characterization (DAC) module plays a crucial role by extracting meta-features (denoted as $f \in F$) for each dataset (d) and aligning them with the most suitable algorithm ($a \in A$). This alignment process prepares instances for the training dataset,

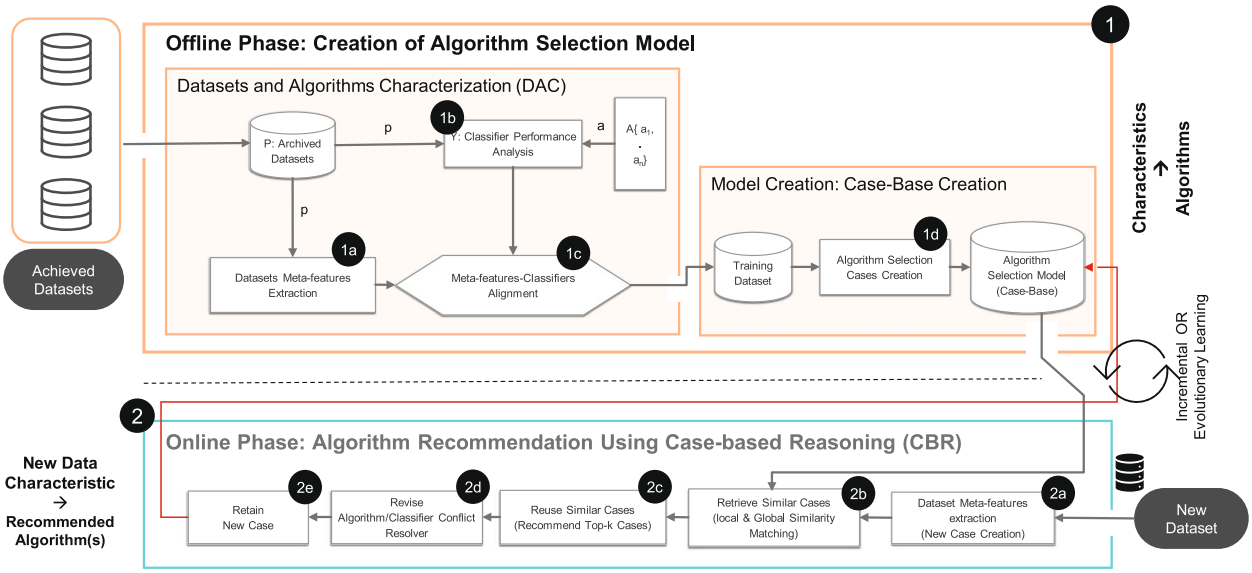

Fig. 2 Case-based reasoning for algorithm selection

Case-Base in this case, facilitating the subsequent stages of the algorithm selection.

Multi-view meta-features extraction

In this study, we propose a new approach that involves the extraction of multi-view meta-features from the archived datasets, with each meta-feature belonging to a unique feature family. These families include general, simple statistical, advanced statistical, and information theoretic. The rationales behind the selection of these four families of meta-feature come from the fact that they represent global view for different classification data types and can be computed in real-time, supporting practical data mining applications, such as algorithm selection. The general view includes simple measurements, computed for the entire dataset, offering a global perspective using aggregated values. The basic statistical view encompasses measurements related to dataset dimensionality and attribute ratios. Advanced statistical features provide valuable insights about a dataset, such as the distribution of numeric attributes, the balance between positive and negative instances, the accuracy of default classification, the presence of incomplete data, and the distinct values in nominal attributes. By examining these advanced statistical characteristics, analysts can gain deeper knowledge and make informed decisions when working with datasets, enabling more effective data mining and analysis for best algorithm selection. Similarly, as each dataset contains both continuous and symbolic data features. To enhance algorithm selection, we extract symbolic meta-features known as information-theoretic features. These features, based on entropy, measure data purity relative to class labels. This approach is unique as compared to single-view approaches [26, 17] because it extracts diverse or multi-view meta-features from the datasets. This diverse set of meta-features enables the utilization of a multi-view learning approach in the classifier selection process. This approach is aligned with the fundamental concept of approaching the algorithm selection problem from multiple aspects, considering various factors and perspectives. This concept is illustrated in Fig. 3 and the characteristics considered are listed in Tables 1, 2, 3 and 4.

These meta-features are computed using [14, 27] available on GitHub.

Machine learning algorithms characterization

In this section, we characterize the performance of the decision tree algorithms listed in Table 5. Our objective is to identify the best-performing algorithms for the dataset at hand. We assess performance using a multifaceted evaluation criteria, considering the weighted average F-score and standard deviation. Results for these criteria are obtained using the Weka experimenter environment [3], employing the default parameters for the algorithms and standardized 10x10-fold cross-validation. To determine the best-performing algorithm (where $a \in A$) for a specific dataset (where ($p \in P$)), we apply PerformanceEval algorithm 1, as illustrated in Fig. 4.

Begin Input:
A set of algorithms A = {a1, ..., a_n}
A set of problems P = {p1, ..., p_n}
Output: A set of best classifiers A'' for each problem p
Procedure:

1. Initialize PerformanceMatrix as an empty matrix
2. For each problem 'p' in 'P':
   - 2.1. For each algorithm 'a' in 'A':
     - 2.1.1. Perform 10x10-fold cross validation to compute performance metrics (Wgt.Avg.F-score and Stdev) for 'a' on 'p'
     - 2.1.2. Store the performance metrics in PerformanceMatrix[a][p]
   - 2.2. Initialize A' as an empty set
   - 2.3. For each algorithm a' in A':
     - 2.3.1. For each algorithm b' in A':
       - 2.3.1.1. If a' ≠ b':
         - 2.3.1.1.1. Perform a statistical significance test between a' and b' based on their performance in PerformanceMatrix
         - 2.3.1.1.2. If a' is statistically significant:
           - 2.3.1.1.2.1. Add a' to A'
   - 2.4. If A' is empty:
     - 2.4.1. Print ("No significant classifiers found for 'p'.")
     - 2.4.2. Go to step 2
   - 2.5. Initialize DS_1 as an empty list
   - 2.6. Sort A' based on the Wgt.Avg.F-score in descending order
   - 2.7. Choose first algorithm a1 from A'
   - 2.8. For each algorithm a' in A':
     - 2.8.1. If Wgt.Avg.F-score of a' is equal to Wgt.Avg.F-score of a1
     - 2.8.2. Add a' to DS_1
   - 2.9. Initialize BestClassifiers as an empty list:
   - 2.10. If DS_1 contains more than one classifier:
     - 2.10.1. Sort DS_1 based on the minimum Stdev in ascending order
     - 2.10.2. Choose first algorithm b1 from DS_1
     - 2.10.3. For each algorithm b' in DS_1:
       - 2.10.3.1. If Stdev of b' is equal to Stdev of b1
         - 2.10.3.1.1. Add b' to BestClassifiers
   - 2.11. Else:
     - 2.11.1. Set BestClassifiers = DS_1
   - 2.12. If BestClassifiers contains more than one classifier:
     - 2.12.1. Print("Multiple best classifiers found for p: ", BestClassifiers)
   - 2.13. Else:
     - 2.13.1. Print("Best classifier for p: ", BestClassifiers[0])
   - 2.14. Go to Step 2

End

Algorithm 1. PerformanceEval: A comprehensive performance evaluation approach

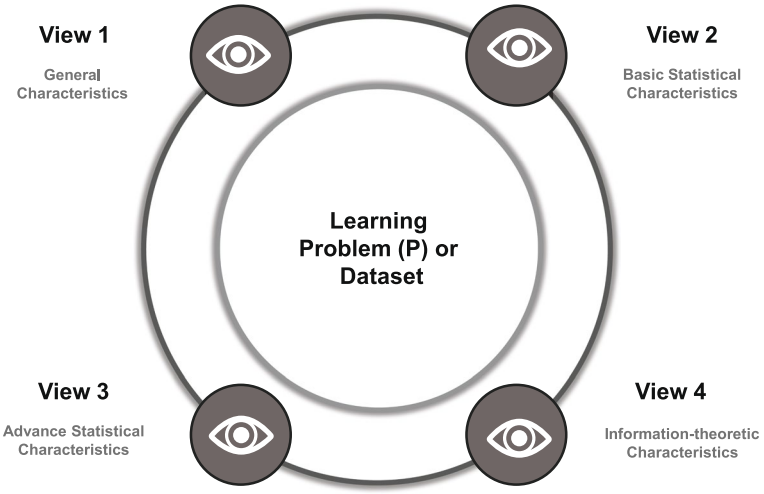

Fig. 3 Enriching Algorithm Selection with Multi-View Dataset Representation

Table 1 General characteristics of dataset

Feature ID	Description
GC 1	InstanceCount
GC 2	NumAttributes
GC 3	ClassCount
GC 4	NumBinaryAtts
GC 5	NumNominalAtts
GC 6	NumNumericAtts
GC 7	NumMissingValues

Table 2 Basic statistical characteristics

Feature ID	Description
BS 1	PercentageOfBinaryAtts
BS 2	PercentageOfNominalAtts
BS 3	PercentageOfNumericAtts
BS 4	MeanSkewnessOfNumericAtts
BS 5	MeanKurtosisOfNumericAtts
BS 6	Dimensionality

Table 3 Advanced statistical characteristics

Feature ID	Description
AS 1	MeanStdDevOfNumericAtts
AS 2	MeanMeansOfNumericAtts
AS 3	NegativePercentage
AS 4	PositivePercentage
AS 5	DefaultAccuracy
AS 6	IncompleteInstanceCount
AS 7	PercentageOfMissingValues
AS 8	MinNominalAttDistinctValues
AS 9	MaxNominalAttDistinctValues
AS 10	StdvNominalAttDistinctValues
AS 11	MeanNominalAttDistinctValues

Table 4 Information theoretic characteristics

Feature ID	Description
IT 1	ClassEntropy
IT 2	MeanAttributeEntropy
IT 3	MeanMutualInformation
IT 4	EquivalentNumberOfAtts
IT 5	NoiseToSignalRatio

Model creation (Case-Base Creation)

Once the datasets and classifiers are characterized, as described in previous sections, the best classifier(s) are assigned to the set of meta-features (e.g., *F→bestClassifier(s)*) using a simple alignment function to produce a single training dataset or Case-Base. The mapping of features against classifiers forms resolved cases for a CBR process.

The rationale behind using case-base creation is that different machine learning algorithms can also be used as predictive models, but the small number of training instances makes it difficult for conventional classifiers. To overcome this issue, we adapt the CBR model with some enhancements in the case base creation and retrieval phases.

For case representation, we adapt propositional case representation schemes [28], where a case is represented

Table 5 Decision tree algorithms in the Weka environment

Algorithm ID	Decision Tree Algorithm
A 1	trees.BFTree
A 2	trees.FT
A 3	trees.J48
A 4	trees.J48graft
A 5	trees.LADTree
A 6	trees.RandomForest
A 7	trees.RandomTree
A 8	trees.REPTree
A 9	trees.SimpleCart

as a proposition containing key-value pair format. In the proposed algorithm selection scenario, a case contains data characteristics (i.e., extracted meta-features) as problem description and the best algorithm name as solution description. A generic structure of the proposed Case-Base, using feature-vector representation, is shown in Table 6.

The meta-features 1-29, shown in Table 6, are multiple views of data characteristics given in Tables 1, 2, 3 and 4. Similarly, the best-classifier (last column) is the label of one or more, best decision tree classifiers, from Table 5. The size of the case-base is 100 resolved cases, authored from 100 freely available classification datasets collected from UCI [29] and OpenML [30] machine learning repositories. A subset of datasets used for case-base creation is provided in Table 7 along with brief descriptions of the general characteristics of the datasets

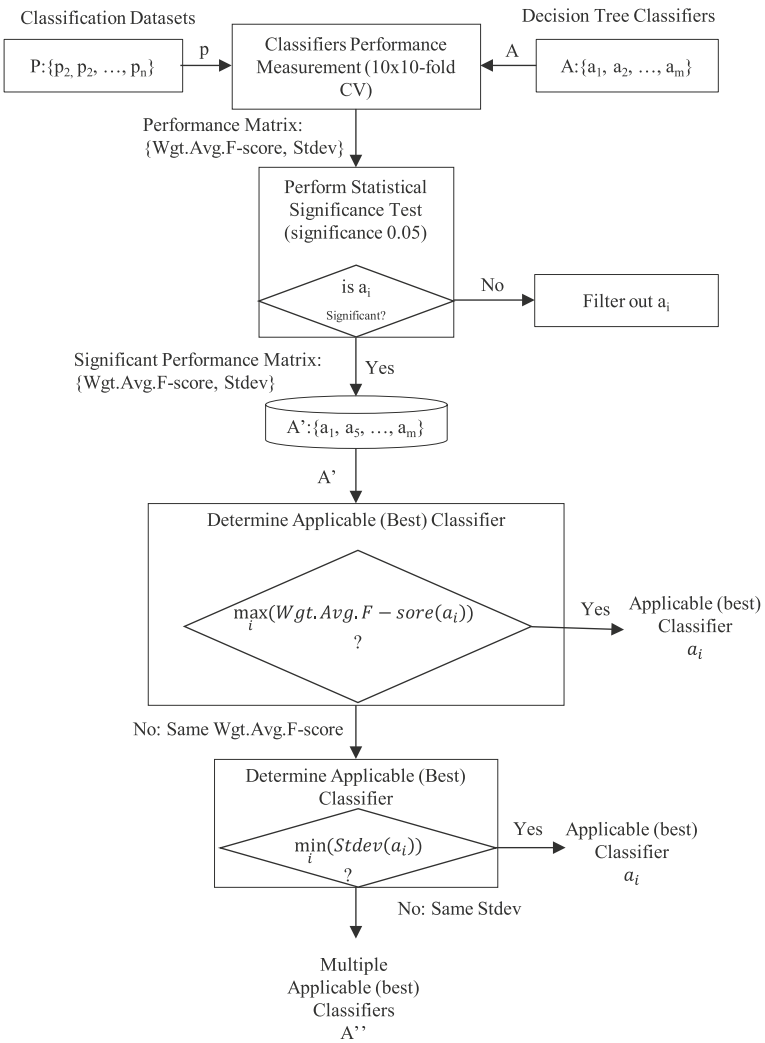

Fig. 4 Multi-criteria analysis for algorithm performance evaluation

Table 6 Representation of algorithm selection case base

Problem or Dataset Description/Characterization					Algorithm Characterization
Resolved Case-ID	**Meta- F_1**	**Meta-F_2**	**…**	**Meta-F_m**	**Best-Algorithm**
R_1	F_1	F_2	…	F_m	A_l
R_2	F_1	F_2	…	F_m	A_1
…	…	…	…	…	…
R_n	F_1	F_2	…	F_m	A_3

* In this case: n=100 and m=29

In the proposed Case-Base, all the features are real numbers, so their data types are set to numeric.

Algorithm recommendation using CBR

Algorithm recommendation using CBR represents the online phase of algorithm selection, suggesting the top-k suitable algorithms to end users through the CBR cycle. The rationale for using CBR methodology for algorithm recommendation, as opposed to state-of-the-art machine learning methods, is that algorithm recommendation is an estimation problem, in which CBR excels, rather than a discrete solution produced by conventional machine learning methods. In the study, we enhance the classical CBR with the use of accurate similarity functions, multi-view features extraction, and incremental learning to improve the algorithms selection process. Methodology of CBR cycle follows the steps given below.

New case preparation - multi-view meta-features extraction

A Query Case *(Q)* is prepared from a given new dataset by extracting multi-view features with the help of a meta-feature extractor to form a feature vector. For this purpose, the same dataset characterization mechanism described in the offline phase is used. As different families of the features are extracted, a multi-view reasoning process is initiated.

CBR cycle – 4R

The CBR cycle comprises retrieve, reuse, revise and retain steps that are performed in sequential order as explained below.

In the retrieve step, similarity functions are defined to match the meta-features of the query dataset Q against the resolved cases R in the Case-Base, retrieving the top-k cases as suggested solutions. For individual meta-feature similarity matching, we use the local similarity function, as shown in Eq. 1. For matching a new case with the existing resolved cases in the Case-Base, we employ a global similarity function, as shown in Eq. 2

$$\mathrm{Sim_l}\left(\mathrm{nC_{mf_i}}, \mathrm{eC_{mf_i}}\right) = \mathrm{idealSim_{mf_i}} - \frac{\mathrm{d_l}\left(\mathrm{nC_{mf_i}}, \mathrm{eC_{mf_i}}\right)}{\mathrm{d_g}\left(\mathrm{Max_{mf_i}}, \mathrm{Min_{mf_i}}\right)} \tag{1}$$

where, $\mathbf{idealSim_{mf_i}}$=1 & $\mathbf{d_g}\left(\mathbf{Max_{mf_i}}, \mathbf{Min_{mf_i}}\right)$ is the global interval or range of the values of each continuous value meta-feature. Similarly, $\mathbf{nC_{mf_i}}$ represents meta-feature of new case and $\mathbf{eC_{mf_i}}$ represents meta-feature of existing cases.

$$\mathrm{Sim_g(nC, eC)} = \frac{\propto_1 * \mathrm{Sim_l}\left(\mathrm{nC_{mf_i}}, \mathrm{eC_{mf_i}}\right) + \cdots + \propto_n * \mathrm{Sim_l}\left(\mathrm{nC_{mf_n}}, \mathrm{eC_{mf_n}}\right)}{\propto_1 + \propto_2 + \cdots + \propto_n} \tag{2}$$

Where, $\propto_i$ is the weight value of each $\mathbf{mf_i}$ in the Case-Base and we assigned equal weight value to each meta-features, based on the assumption that all the 29 meta-features are equally important for selecting the best algorithm.

Reuse In the reuse step, the solution part, i.e., the label of best algorithm, of the top-k similar cases are assigned to the problem description part of the new case as a suggested solution (recommended algorithm in this case).

This process of retrieve and reuse is algorithmically presented in Algorithm 2.

Table 7 Subset of datasets for case-base creation with brief descriptions

ID	Dataset Name	General Characteristics							
		Attributes	**NominalAtts**	**NumericAtts**	**BinaryAtts**	**Classes**	**IncompInstances**	**Instances**	**MissingValues**
1	abalone.arff	9	1	7	0	3	0	4177	0
2	abe_148.arff	6	0	5	0	2	0	66	0
3	acute-inflammations.arff	7	5	1	5	2	0	120	0
…	…	…	…	…	…	…	…	…	…
…	…	…	…	…	…	…	…	…	…
100	car.arff	7	6	0	0	4	0	1728	0

Begin
inputs: Q – // Query Case, prepared from extracted meta features of the new dataset (d)
$R - \{r_1, r_2, \ldots, r_n\}$; // Case-Base e.g. all previously resolved cases
K – // int, number of CBR cases to return
output: $C - \{c_1, c_2, \ldots, c_k\}$; // Collection of top-k most similar CBR Cases
1 I = calculateSimilarityIntervals () // Calculates similarity interval, weight for each feature
2 S = buildNNConfig(I) // procedure : builds and returns NN similarity configuration
3 RR = evaluateSimilarity(R, Q, S) // procedure defined below
4 C = selectTopK(RR, K) //Select top K CBR cases from RR
End
PROCEDURE: buildNNConfig(I)
Begin
inputs: $I-\{i_1, i_2, \ldots, i_n\}$; //Similarity Intervals, i denotes an interval for specific feature, where i ∈ I
output: similarityConfig // returns NN similarity configuration,
1. simConfig = new NNConfig(); // initialize similarity configuration
2. simConfig.setDescriptionSimFunction(global.average) //Global similarity function, see eq. 2
3. foreach SimilarityInterval i in I
4. simConfig.addMapping(i.getFeature, interval(i.getInterval)) //interval local similarity function, see eq. 1
5. simConfig.setWeight(i.getFeature, i.getWeight)
6. return simConfig
End
PROCEDURE: evaluateSimilarity(R, Q, S)
Begin
inputs: Q – //the queryCase, built from extracted meta features
$R - \{r_1, r_2, \ldots, r_n\}$; // case base e.g. all previously resolved cases
S – // NN similarity configuration
output: RetrievalResults : RR
Description: evaluates similarity of each c_i where $c_i \in C$ against this queryCase Q using similarity function mapped in NN similarityConfig S, and returns a collection of retrievalResults RR
return RR
End

Algorithm 2. Top-k algorithm selection using case-based reasoning

Descriptions of each of the procedures, used in this algorithm 2, are given below.

CalculateSimilarityIntervals: This procedure loops through all meta-features, calculates interval value and defines weight for each of feature. The interval value is computed using $d_g(Max_{mf_i}, Min_{mf_i})$, while the weight assigned to each meta-feature is same, i.e., 1.

BuildNNConfig(*I*): This procedure performs the main task of finding nearest neighbor computation. The set of tasks performed using this procedure are: initialize NNConfig, set *global similarity function* as per Eq. 2, map a *local similarity function* with each feature as per Eq. 1, s*et weight* for each feature, i.e., assign 1 to each feature in this case and return NNConfig.

evaluateSimilarity(R, Q, S): evaluates *similarity* of *each* c_i where $c_i \in C$ against the *queryCase Q* using *similarity function* mapped in *NN similarityConfig S,* and returns a collection of *retrievalResults RR* (most similar cases).

selectTopK(RR, K): this procedure *Selects top K most similar CBR cases from the collection of retrievalResult RR.*

The final output of the retrieve and reuse steps, denoted as RR, consists of a list of the top-k (with k=3) cases that exhibit the highest similarity scores in comparison to the query case (Q). If RR yields the top-k algorithms with distinct Wgt.Avg.F-score values, the top-ranked one is recommended as the most suitable algorithm and assigned as the label to the feature vector as the class label. Otherwise, the revise step of the CBR is initiated to uniquely identify the most suitable algorithm.

Revise – algorithm conflict resolver (ACR) In the revision step of the proposed CBR approach, the unique algorithm recommended in the reuse step is added as a new instance into the existing Case-Base. However, if more than one algorithm with similar or statistically insignificant differences in the similarity scores are recommended, then either any of the recommended algorithms is randomly selected as the final recommendation, or conflict resolution becomes necessary among the competing algorithms. To resolve this conflict, we propose a method known as the Algorithm Conflict Resolver (ACR). This method performs meta-reasoning at the meta-characteristics level of the classifiers (e.g., decision tree length, number of rules, depth, among others) rather than focusing on the data characteristics. The proposed ACR employs a multi-objective criteria with weighted summation, as presented in Eq. 3, which takes into account the comprehensibility characteristics of the classifiers listed in Table 8. This allows us to recommend the most comprehensible classifier.

Table 8 Characterization of decision tree classifiers

ID	DT Classifier Comprehensibility Characteristics
1	measureNumRules
2	measurePercentAttsUsedByDT
3	measureTreeSize
4	measureNumLeaves
5	measureNumPredictionLeaves

$$\boldsymbol{a_i} = \sum (\boldsymbol{w_{nj}} * \mathbf{C(i, j)} \tag{3}$$

Where: $\boldsymbol{i}$ represents algorithm in A(algorithm space) and $\boldsymbol{j}$ represents criteria in $\boldsymbol{C}$(criteria space)

- $\boldsymbol{a_i}$ is the weighted sum score for algorithm $\mathbf{a} \in \mathbf{A}$
- $\boldsymbol{w_{nj}}$ is the normalized weight assigned to criterion $\boldsymbol{c_j}$, calculated using Eq. 4.
- $\boldsymbol{C(i,j)}$ is the performance or evaluation score of algorithm $\boldsymbol{a_i}$ on criterion $\boldsymbol{c_j}$

$$\boldsymbol{w_{nj}} = \frac{(\boldsymbol{w_i} - \boldsymbol{min}(\boldsymbol{w})}{\boldsymbol{max}(\boldsymbol{w}) - \boldsymbol{min}(\boldsymbol{w})} \tag{4}$$

where: $\boldsymbol{w_i}$ is the original weight assigned to criterion c_i by the user, using the AHP process

- min($\boldsymbol{w}$) is the minimum weight for all criteria
- max($\boldsymbol{w}$) is the maximum weight for all criteria

Working of the ACR algorithm is shown in Algorithm 3.

Begin

Input:

- Weights for criteria w1, w2, w3, w4, and w5

- Algorithms space A with various algorithms

- AlgPerformancValue C(n,m) *// n denotes no. algorithm, and j, denotes no. criteria*

- Comprehensibility criteria: measureNumRules, measurePercentAttsUsedByDT, measureTreeSize, measureNumLeaves, measureNumPredictionLeaves

Output:- Selected algorithm 'a' from the algorithmic space A

Procedure:

// NormalizeWeights:

1. w_i_normalized <- use equation 4, where i=1, …, 5

2. C(n, m) <- InputCriteriaMatrix(): *// Input C(n, m), with dimensions (n x m), where 'n' is the number of algorithms and 'm' is the number of criteria measures.*

// CalculateWeightedCriteriaMatrix:

3. For i in 1:n // loop over algorithms in Matrix C
   3.1. For j in 1:m // loop over criteria in Matrix C
      3.1.1. C_weighted(i, j) <- C(i, j) x wj_normalized

// CalculateWeightedSums:

4. For i in 1:n // loop over algorithms in Matrix C_weighted
   4.1. Initialize WeightedSum(i) to 0
   4.2. For j in 1:m // loop over criteria in Matrix C_weighted
      4.2.1. WeightedSum(i) <- WeightedSum(i) + C_weighted(i, j)

// CalculateRelativeImportance:

5. TotalWeightedSum <- Sum (WeightedSum) // sum all rows
6. For each algorithm index i in WeightedSum:
   6.1. RelativeImportance(i) <- WeightedSum(i) / TotalWeightedSum

// SelectAlgorithm():

7. Initialize MaxRelativeImportance to -1
8. For each algorithm index i in RelativeImportance:
   8.1. If RelativeImportance(i) > MaxRelativeImportance:
      8.1.1. Set MaxRelativeImportance <- RelativeImportance(i)
      8.1.2. Set SelectedAlgorithm <- Algorithm i

// OutputResult:

9. Output SelectedAlgorithm as the final choice

End

Algorithm 3. Algorithms Conflict Resolver (ACR)

Given that conflict resolution is application-dependent, we employ a semi-automatic expert-based criteria weighting approach, adopting the analytical hierarchy process (AHP) pairwise comparison processes. The algorithm used for AHP-based criteria weighting is presented in Algorithm 4.

Begin

Input:

- Matrix A with value a(i, j), representing pairwise comparisons values entered by the experts for the comprehensibility criteria
- Number of criteria n
- Number of algorithms m

Output:

- Weights w1, w2, w3, ..., wn for algorithms comprehensibility criteria

Procedure:

1. Normalize matrix A to obtain AN by using equation 4.
2. Calculate weighted averages for each criteria of AN to get W(i) using equation 3
3. Calculate the consistency vector CV by normalizing W(i) values.
4. Calculate CI = (λ_max - n) / (n - 1), where λ_max is the largest eigenvalue of AN.
5. Calculate RI using a predefined table or formula based on n.
6. Calculate CR = CI / RI to check matrix consistency.
7. If CR exceeds a predefined threshold (e.g., 0.1), revise pairwise comparisons.
8. Compute weights as w1, w2, w3, ..., wn based on CV.
9. Normalize the weights to ensure they sum up to 1: w1, w2, w3, ..., wn = CV / ∑(CV).
10. Output the computed weights for evaluation criteria.

End

Algorithm 4. Weight calculation using Analytic Hierarchy Process (AHP)

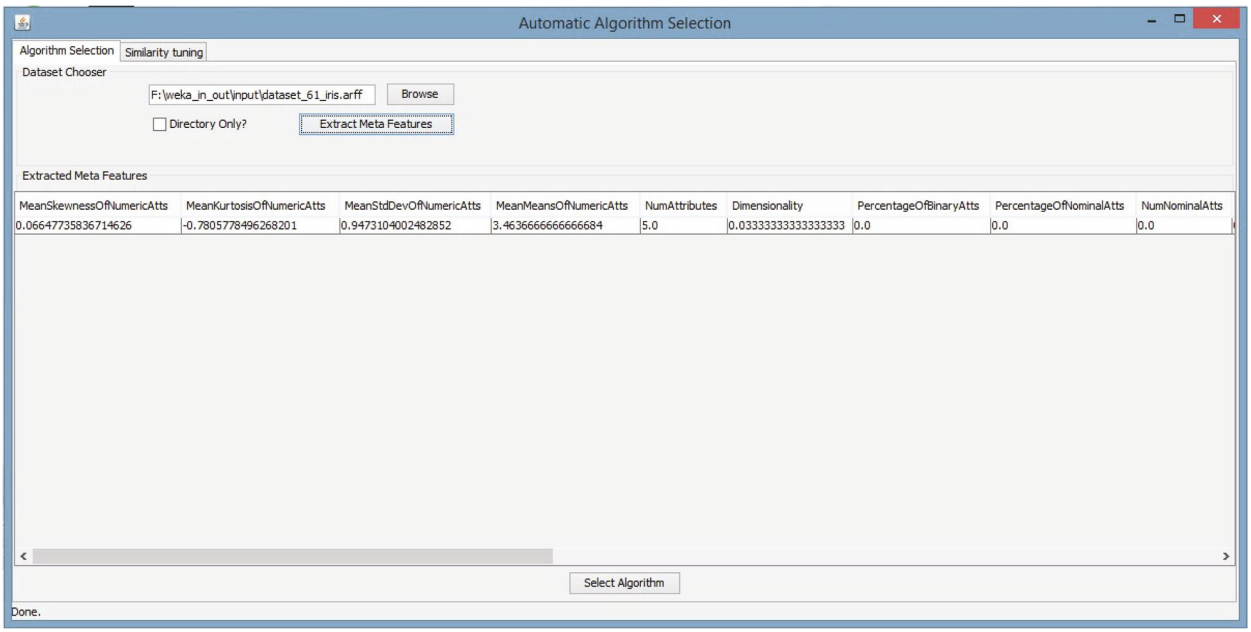

Fig. 5 Interface of the CBR Application for Meta-Features Extraction

In the retention step, Q, represented by a meta-features containing features from all the families, along with the final recommended algorithm either from the reuse or revise steps of CBR, is incorporated into the Case-Base. This process incrementally enlarges the size of the Case-Base and improves the quality of the CBR for recommending the most suitable algorithm. This demonstrates the proposed algorithm recommendation model as an instance of an incremental and evolutionary learning process.

Implementation, experiments and evaluation

This section describes the implementation of the proposed system and presents experiments conducted to validate the methodology.

Implementation

The proposed multi-view case-based reasoning methodology for accurate classifier selection has been implemented in the Java environment as an open-source application. The key components of the methodology include the extraction of multiple categories of meta-features from the dataset and meta-reasoning by utilizing the Case-Base. These meta-features are computed using the OpenML [30] data characteristics (DC) open-source library, which is freely available on GitHub [27] . For the CBR-based reasoning process, we utilized jColibri 2.0, a case-based reasoning framework [31], where we implemented our custom case similarity functions to ensure accurate matching of existing cases. The resulting CBR-based incremental learning and reasoning system has been released as an open-source application on GitHub, featuring an extensible and adaptable implementation strategy [32], enabling end-users to utilize it for selecting a suitable decision tree classifier for their application's data. The interface of the CBR application for meta-feature extraction is displayed in Fig. 5.

The process of multi-view reasoning using CBR is shown in Fig. 6.

Experimental setup

Classifiers under consideration

We conducted experiments on the nine most commonly used multi-class classification algorithms, as listed in Table 5. These algorithms are implemented in the Weka machine learning library [3]. We utilized them with their default parameters

Training and testing datasets To train and test the proposed methodology, two disjoint sets of datasets are utilized. For training, the CBR model, i.e., Case-Base, is constructed using 100 multi-class classification datasets, as shown in Table 9. These datasets are sourced from the UCI machine learning repository [29] and OpenML repositories [30]. Similarly, a separate set of 52 datasets is employed for testing the methodology. All the classifiers listed in Table 5 are evaluated for each of the test datasets

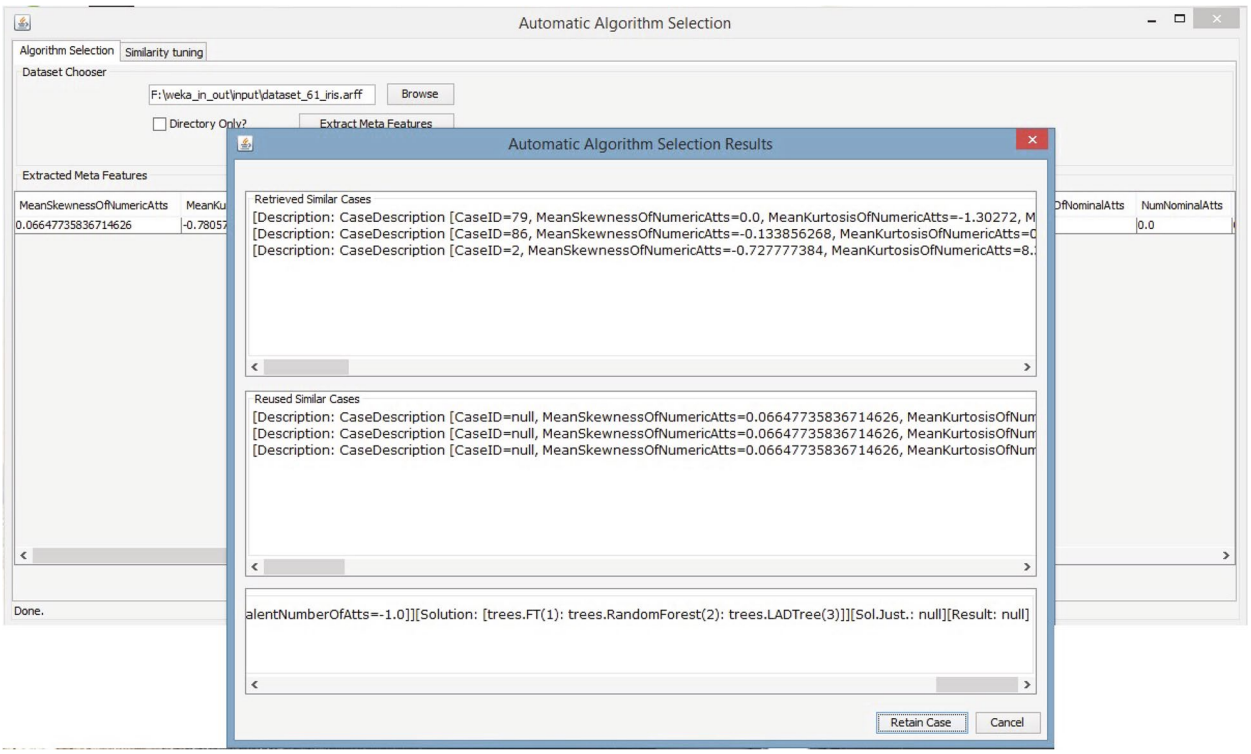

Fig. 6 Case-Based Reasoning (CBR) for optimal machine learning algorithm selection

using the method implemented and described in Fig. 6. Subsequently, the best classifiers are determined to assess the performance of the proposed methodology.

Evaluation methodology and criteria To evaluate the accuracy of the proposed method, the following steps are used:

- For each given dataset (test datasets in this case), meta-features are extracted using the developed meta-feature extractor to prepare a Query Case (Q).
- The CBR methodology is used to recommend the top-k (k=3) best classifiers for each Q.
- Measure the similarity between the recommended top-k (k=3) classifiers and the actual classifiers of those datasets. If the recommended classifier for a given dataset belongs to any of the top-k (k=3) classifiers, the recommendation is declared as correct; otherwise, it is considered incorrect.

Experiments and results analysis When experiments were conducted on the test Case-Base consisting of 52 datasets, shown in Table 10, and the results were evaluated, 48 out of the 52 datasets received accurate classifier recommendations. As a result, the overall accuracy of the proposed methodology was determined to be 94% for appropriate algorithm recommendations with a top-k value of 3 algorithms.

$$\text{Overall Accuracy} = \frac{\text{Number of Accurate Recommendations} \times 100}{\text{Total Number of Datasets}} = \frac{48 \times 100}{52} = 94\%$$

The results in Table 10 show that only 3 out of 52 classifiers, namely Dataset 30, Dataset 38, and Dataset 48, were not recommended with correct classifiers. Similarly, for a top-k value of 1, the proposed methodology correctly recommended accurate classifiers for 30 datasets, resulting in an accuracy of 57.6% (calculated as 30 * 100 / 52). To analyze the results for a top-k value of 2, the methodology correctly recommended classifiers for 38 datasets and achieved an accuracy of 73% (calculated as 38 * 100 / 52).

Table 9 Training datasets for case-base creation with brief descriptions

ID	Dataset Name	General Characteristics							
		Attributes	NominalAtts	NumericAtts	BinaryAtts	Classes	Incomplnstances	Instances	MissingValues
1	abalone.arff	9	1	7	0	3	0	4177	0
2	abe_148.arff	6	0	5	0	2	0	66	0
3	acute-inflammations.arff	7	5	1	5	2	0	120	0
4	ada_agnostic.arff	49	0	48	0	2	0	4562	0
5	ada_prior.arff	15	8	6	1	2	88	4562	88
6	adult- 4000.arff	15	8	6	1	2	0	3983	0
7	adult- 80000.arff	15	8	6	1	2	0	8000	0
8	ailerons - 5840.arff	41	0	40	0	2	0	5795	0
9	analcatdata_aids.arff	5	2	2	0	2	0	50	0
10	analcatdata_apnea1.arff	4	2	1	0	2	0	475	0
11	analcatdata_apnea2.arff	4	2	1	0	2	0	475	0
12	analcatdata_asbestos_ciup-dated	4	2	1	1	2	0	83	0
13	analcatdata_authorship.arff	71	0	70	0	4	0	841	0
14	analcatdata_bankruptcy.arff	7	1	5	0	2	0	50	0
15	analcatdata_birthday.arff	4	2	1	0	2	30	365	30
16	analcatdata_bondrate.arff	12	7	4	1	5	1	57	1
17	analcatdata_boxing1.arff	4	3	0	1	2	0	120	0
18	analcatdata_boxing2.arff	4	3	0	1	2	0	132	0
19	analcatdata_braziltourism.arff	9	4	4	1	7	49	412	96
20	analcatdata_broadway.arff	10	6	3	1	5	6	95	9
21	analcatdata_broadwaymult.arff	8	4	3	1	7	18	285	27
22	analcatdata_chall101.arff	3	1	1	0	2	0	138	0
23	analcatdata_challenger.arff	6	4	1	0	2	0	23	0
24	analcatdata_chlamydia.arff	4	3	0	1	2	0	100	0
25	analcatdata_creditscore.arff	7	3	3	2	2	0	100	0
26	analcatdata_currency.arff	4	2	1	0	7	0	31	0
27	analcatdata_cyyoung8092.arff	11	3	7	2	2	0	97	0
28	analcatdata_cyyoung9302.arff	11	4	6	2	2	0	92	0
29	analcatdata_dmft.arff	5	4	0	1	6	0	797	0
30	analcatdata_donner.arff	4	3	0	1	2	0	28	0
31	analcatdata_draft.arff	6	2	3	0	2	1	366	1
32	analcatdata_election2000.arff	16	1	14	0	2	0	67	0
33	analcatdata_esr.arff	3	0	2	0	2	0	32	0
34	analcatdata_famufsu.arff	4	2	1	0	2	0	14	0
35	analcatdata_fraud.arff	12	11	0	10	2	0	42	0
36	analcatdata_germangss.arff	6	4	1	2	4	0	400	0
37	analcatdata_gsssexsurvey.arff	10	5	4	5	5	6	159	6
38	analcatdata_gviolence.arff	10	1	8	0	2	0	74	0
39	analcatdata_halloffame.arff	18	2	15	0	3	20	1340	20
40	analcatdata_happiness.arff	4	2	1	0	3	0	60	0
41	analcatdata_homerun.arff	28	14	13	7	2	1	163	9
42	analcatdata_impeach.arff	11	8	2	4	2	0	100	0
43	analcatdata_japansolvent.arff	10	1	8	0	2	0	52	0
44	analcatdata_lawsuit.arff	5	1	3	1	2	0	264	0
45	analcatdata_marketing.arff	33	32	0	0	5	35	347	79
46	analcatdata_michiganacc.arff	5	2	2	0	2	0	108	0
47	analcatdata_ncaa.arff	20	15	4	1	2	0	120	0
48	analcatdata_neavote.arff	4	2	1	0	2	0	100	0
49	analcatdata_negotiation.arff	6	1	4	1	2	17	92	26
50	analcatdata_olympic2000.arff	13	1	11	0	2	0	66	0

Table 9 (continued)

ID	Dataset Name	General Characteristics							
		Attributes	NominalAtts	NumericAtts	BinaryAtts	Classes	IncompInstances	Instances	MissingValues
51	analcatdata_reviewer.arff	9	8	0	0	2	367	379	1368
52	analcatdata_runshoes.arff	11	6	4	5	2	14	60	14
53	analcatdata_supreme.arff	8	0	7	0	2	0	4052	0
54	analcatdata_uktrainacc.arff	17	0	16	0	2	25	31	150
55	analcatdata_votesurvey.arff	5	1	3	1	4	0	48	0
56	analcatdata_whale.arff	8	1	6	1	2	5	228	15
57	analcatdata_wildcat.arff	6	2	3	2	2	0	163	0
58	anneal.arff	39	32	6	14	6	0	898	0
59	anneal.ORIG.arff	39	32	6	7	6	898	898	22175
60	appendicitis.arff	8	0	7	0	2	0	106	0
61	ar1.arff	30	0	29	0	2	0	121	0
62	ar3.arff	30	0	29	0	2	0	63	0
63	ar4.arff	30	0	29	0	2	0	107	0
64	ar5.arff	30	0	29	0	2	0	36	0
65	arsenic-female-bladder.arff	5	1	3	0	2	0	559	0
66	arsenic-female-lung.arff	5	1	3	0	2	0	559	0
67	arsenic-male-bladder.arff	5	1	3	0	2	0	559	0
68	arsenic-male-lung.arff	5	1	3	0	2	0	559	0
69	audiology (binary version of audiology).arff	70	69	0	61	2	222	226	317
70	australian.arff.arff	15	0	14	0	2	0	690	0
71	automobile.arff	26	10	15	3	6	0	159	0
72	autoMpg.arff	8	3	4	0	2	6	398	6
73	autos.arff	26	10	15	4	7	46	205	59
74	autoUniv-au6-1000.arff	41	3	37	2	8	0	1000	0
75	autoUniv-au7-1100.arff	13	4	8	2	5	0	1100	0
76	autoUniv-au7-700.arff	13	4	8	2	3	0	700	0
77	backache.arff	33	26	6	22	2	0	180	0
78	badges2.arff	12	3	8	3	2	0	294	0
79	balance-scale.arff	5	0	4	0	3	0	625	0
80	balloon.arff	3	0	2	0	2	0	2001	0
81	banana.arff	3	0	2	0	2	0	5300	0
82	bands.arff	20	0	19	0	2	0	365	0
83	bank32nh - 1956.arff	33	0	32	0	2	0	1918	0
84	bank8FM.arff	9	0	8	0	2	0	8192	0
85	banknote-authentication.arff	5	0	4	0	2	0	1372	0
86	baskball.arff	5	0	4	0	2	0	96	0
87	BC-breast-cancer-data.arff	10	9	0	3	2	9	286	9
88	biomed.arff	9	1	7	0	2	15	209	15
89	blogger.arff	6	5	0	2	2	0	100	0
90	blood-transfusion-service-center.arff.arff	5	0	4	0	2	0	748	0
91	bodyfat.arff	15	0	14	0	2	0	252	0
92	bolts.arff	8	0	7	0	2	0	40	0
93	boston.arff	14	2	11	1	2	0	506	0
94	boston_corrected.arff	21	3	17	1	2	0	506	0
95	breast-cancer.arff	10	9	0	3	2	9	286	9
96	breastTumor.arff	10	8	1	4	2	9	286	9
97	bridges_version1.arff	13	9	3	2	6	37	107	73
98	bridges_version2.arff	13	12	0	2	6	37	107	73
99	bupa.arff	7	0	6	0	2	0	345	0
100	car.arff	7	6	0	0	4	0	1728	0

Table 10 Performance results of case-based reasoning for algorithm selection

ID	Dataset	Position of Recommended Algorithm in Top-3 Algorithms
Dataset 1	cardiotocographt-10clas	1
Dataset 2	cars	1
Dataset 3	cars_with_names	2
Dataset 4	CastMetal1	1
Dataset 5	chess-small	1
Dataset 6	cholesterol	1
Dataset 7	chscase-adopt	3
Dataset 8	chscase-census2	3
Dataset 9	chscase-census3	2
Dataset 10	chscase-census5	1
Dataset 11	chscase-census6	2
Dataset 12	chscase-funds	1
Dataset 13	chscase-geyser1	1
Dataset 14	hscase-health	1
Dataset 15	chscase-vine1	1
Dataset 16	chscase-vine2	3
Dataset 17	chscase-whale	1
Dataset 18	cjs	1
Dataset 19	cleveland	1
Dataset 21	climate-simulation-crac	3
Dataset 22	cloud	3
Dataset 23	cm1_req	1
Dataset 24	cmc	1
Dataset 25	horse-colic	1
Dataset 26	horse-colic.ORIG	2
Dataset 27	colleges-aaup	1
Dataset 28	colleges-usnews	2
Dataset 29	collins	1
Dataset 30	onfidence	2
Dataset 31	**ontact-lenses**	**9**
Dataset 32	contraceptive	3
Dataset 33	costamadre1	1
Dataset 34	cps_85_wages	3
Dataset 35	cpu	3
Dataset 36	cpu_act	1
Dataset 37	cpu_small	1
Dataset 38	credit-rating	3
Dataset 39	**crx**	**4**
Dataset 40	DATATRIEVE	1
Dataset 41	dbworld-subjects-stemme	1
Dataset 42	dbworld-subjects	1
Dataset 43	delta_ailerons	1
Dataset 44	dermatology	1
Dataset 45	desharnais.csv-weka.fil	2
Dataset 46	pima_diabetes	1
Dataset 47	diggle-Table_A1_Luteniz	2
Dataset 48	disclosure-X_BIAS	1
Dataset 49	**disclosure-X_NOISE**	**4**
Dataset 50	disclosure-X_TAMPERED	3
Dataset 51	disclosure-Z	3
Dataset 52	dresses-sales	1
Dataset 53	eastWest	1

Conclusion and future work

This research centers on the automatic selection of machine learning (ML) algorithms through the integration of edge machine learning (edge ML) and a case-based reasoning (CBR) methodology. Edge ML enhances the capabilities of CBR by facilitating the recommendation of ML algorithm families at edge nodes and the selection of the actual algorithm at remote cloud servers. This integration serves to enhance system performance by significantly expediting algorithm recommendations while minimizing associated costs.

In the future, our research endeavors will expand upon the current study. This expansion will encompass the practical implementation of edge ML computing, a facet not covered in this research. Additionally, we aim to augment the case-based approach by introducing more meta-characteristics and incorporating a broader array of algorithm families. These enhancements are designed to transform our platform into a universal and versatile tool for machine learning practitioners. Through these developments, we seek to provide a comprehensive and adaptable solution to the challenges of ML algorithm selection.

Authors' contributions

R.A. proposed the idea, formulated the problem, designed the algorithms and wrote the main manuscript. M.S.H.Z implemented the algorithm selection platform as an open source software. A.M.K. rigorously reviewed the paper before and after the revision and J.H. prepared the figures, reviewed the manuscript, interpreted the results and provided financial support.

Funding

This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) IITP-2017-0-00655, Lean UX core technology and platform for any digital artifacts UX evaluation.

Availability of data and materials

Data can be made available to the researchers on their personal requests to the corresponding author.

Declarations

Ethics approval and consent to participate

"Not applicable".

Competing interests
The authors declare no competing interests.

Received: 2 January 2023 Accepted: 5 November 2023
Published online: 21 November 2023

References

1. Koerich, A.L. *Improving classification performance using metaclasses*. in *SMC'03 Conference Proceedings. 2003 IEEE International Conference on Systems, Man and Cybernetics. Conference Theme-System Security and Assurance (Cat. No. 03CH37483)*. 2003. IEEE.
2. Tavakoli, S., *Signal classification using weighted orthogonal regression method.* arXiv preprint arXiv:2010.05979, 2020.
3. Bouckaert RR et al (2010) WEKA--experiences with a java open-source project. J Mach Learn Res 11:2533–2541
4. Jalernrat, S., *Data Mining Using Decision Tree Algorithms.* University of the Thai Chamber of Commerce Journal, 2013: p. 11-43.
5. Engel, J., T. Erickson, and L. Martignon. *Teaching about decision trees for classification problems*. in *IASE Satellite Meeting*, https://iase-web.org/documents/papers/sat2019/IASE2019% 20Satellite% 20132_E NGEL. *pdf*. 2019.
6. Géron, A., *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow*. 2022: " O'Reilly Media, Inc.".
7. Ali R, Lee S, Chung TC (2017) Accurate multi-criteria decision making methodology for recommending machine learning algorithm. Expert Syst Appl 71:257–278
8. Reif M et al (2014) Automatic classifier selection for non-experts. Pattern Anal Appl 17:83–96
9. Brodley, C.E. *Addressing the selective superiority problem: Automatic algorithm/model class selection*. in *Proceedings of the Tenth International Conference on Machine Learning*. 1993. Citeseer.
10. Wolpert DH, Macready WG (1997) No free lunch theorems for optimization. IEEE Trans Evol Comput 1(1):67–82
11. Aha, D.W. *Generalizing from Case studies: A Case Study*. in *Ninth International Conference on Machine Learning*. 1992. Citeseer.
12. Smith-Miles KA (2008) Cross-disciplinary perspectives on meta-learning for algorithm selection. ACM Comput Surv 41(1):1–25
13. Monteiro JP et al (2021) Meta-learning and the new challenges of machine learning. Int J Intell Syst 36(11):6240–6272
14. Ali, R., et al. *A case-based meta-learning and reasoning framework for classifiers selection*. in *Proceedings of the 12th international conference on ubiquitous information management and communication*. 2018.
15. Bernado-Mansilla E, Ho TK (2005) Domain of competence of XCS classifier system in complexity measurement space. Evol Comput IEEE Trans 9(1):82–104
16. Pise N, Kulkarni P. Algorithm selection for classification problems. in 2016 SAI Computing Conference (SAI). 2016. IEEE.
17. Song Q, Wang G, Wang C (2012) Automatic recommendation of classification algorithms based on data set characteristics. Pattern Recognit 45(7):2672–2689
18. Bache, K. and M. Lichman, *UCI Machine Learning Repository [*http://archive.ics.uci.edu/ml]. 2013, Irvine, CA: University of California, School of Information and Computer Science.
19. Brazdil P, Gama J, Henery B (1994) Characterizing the applicability of classification algorithms using meta-level learning. in European Conference on Machine Learning: ECML-94. Springer
20. Ali S, Smith KA (2006) On learning algorithm selection for classification. Applied Soft Computing 6(2):119–138
21. Gama J, Brazdil P (1995) Characterization of classification algorithms. Progress in Artificial Intelligence. Springer, pp 189–200
22. Brazdil PB, Soares C, Da Costa JP (2003) Ranking learning algorithms: using IBL and meta-learning on accuracy and time results. Mach Learn 50(3):251–277
23. Shao X et al (2023) EFFECT: Explainable framework for meta-learning in automatic classification algorithm selection. Inform Sci 622:211–234
24. Garouani M et al (2022) Using meta-learning for automated algorithms selection and configuration: an experimental framework for industrial big data. J Big Data 9(1):57
25. Rice JR (1976) The algorithm selection problem. Adv Comput 15:65–118
26. Wang G et al (2014) A generic multilabel learning-based classification algorithm recommendation method. ACM Trans Knowl Discov Data 9(1):7
27. Sun, Q., *Integrated Fantail library*. 2014, GitHub.
28. Sarkheyli A, Sa'ffker D (2015) Case indexing in case-based reasoning by applying situation operator model as knowledge representation model. IFAC-PapersOnLine 48(1):81–86
29. Lichman, M., *UCI machine learning repository [*http://archive.ics.uci.edu/ml*]. Irvine, CA: University of California, School of Information and Computer Science.* 2013.
30. Van Rijn JN et al (2013) OpenML: A collaborative science platform. Machine learning and knowledge discovery in databases. Springer, pp 645–649
31. Bello-Tomás JJ, González-Calero PA, Díaz-Agudo BJ (2004) An object-oriented framework for building cbr systems. Advances in case-based reasoning. Springer, pp 32–46
32. Rahman, A. and S. Muhammad, *Automatic-algorithm-selector*. 2016, GitHub.

Publisher's Note