IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 14 February 2024, accepted 26 March 2024, date of publication 4 April 2024, date of current version 15 April 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3385341

RESEARCH ARTICLE

A Semi-Supervised Learning Approach to Quality-Based Web Service Classification

MEHDI NOZAD BONAB[1], JAFAR TANHA[2], AND MOHAMMAD MASDARI[1]
[1]Department of Computer Engineering, Islamic Azad University, Urmia Branch, Urmia 5716963896, Iran
[2]Electrical and Computer Engineering Department, University of Tabriz, Tabriz 5166616471, Iran

Corresponding author: Jafar Tanha (tanha@tabrizu.ac.ir)

ABSTRACT The Internet provides a platform for sharing services, and web service brokers help users to choose the suitable service among similar services based on ranking. The quality of service is important in evaluating the services the user needs. However, finding a quality-based data label in many fields can be time-consuming and difficult. Thus, machine learning is required to classify and choose the best service in this field. The selection process is done through analysis and recommendations by the system. This article introduces the SSL-WSC algorithm, which classifies unlabeled data through semi-supervised self-training learning using a small amount of labeled data. This algorithm labels the data using a two-step method of calculating a score for each service and dynamic thresholding. The quality features of web services obtained from the QWS dataset were used to evaluate the performance of the proposed algorithm. The experimental results in different scenarios showed that using proposed semi-supervised learning algorithms to create classification models led to better results, so it improved the F1-score, accuracy, and precision, on average, by 11.26%, 9.43% and 9.53%, respectively, as compared to the supervised method.

INDEX TERMS Classification, machine learning, quality, semi-supervised learning, web services.

I. INTRODUCTION

Web services are software systems designed for electronic supply and demand through machine-to-machine interaction on a network [1], [2]. These systems have a machine-readable interface definition called Web Services Description Language (WSDL) and use protocols such as Simple Object Access Protocol (SOAP) to transmit messages [3]. Web services follow Service-Oriented Architecture (SOA) and adhere to its standards [4]. SOA is an approach to producing distributed systems that provide software functions through services [5]. These services can be used to build new services or called by other software.

Service brokers provide an interface between service providers and clients, allowing users to compare and select different types of services (Figure 1) [6]. The brokers are responsible for three tasks: ranking, selecting, and composing services. A ranking system calculates the relative value of different services based on the quality of service required by the client and the characteristics of the existing services. After comparing with other services, it offers the most appropriate service to the user. Due to the increasing number of service providers, web services with similar functions have also increased. The only difference between these similar web services is their performance quality. The service quality of web services includes a series of non-operational features such as execution cost and time, availability, execution success rate, security, etc [7], [8]. Therefore, selecting a service that can meet clients' quality standards from a range of services has become a crucial challenge [9], [10]. Each web service is designed to perform a specific task, and a user's workflow may consist of multiple tasks. For each task, providers have a set of candidate services with varying levels of service quality. Hence, choosing the suitable web service for each workflow task is essential to ensure maximum service quality [11]. Nowadays, the most popular web services are related to the following topics: Generative artificial intelligence, Social media, E-commerce, Video Streaming, News, Messaging, Metaverse and Gaming, Financial Services, and Cryptocurrency Services.

The associate editor coordinating the review of this manuscript and approving it for publication was Zhangbing Zhou.

Getting accurate information about services involves gathering data from multiple service providers. Nowadays, data mining technology is used to analyze this data. There are different approaches and methods for data mining, each appropriate for a specific application that can extract certain types of knowledge. One such method is classification, which involves finding the general pattern of existing data and predicting the new data class using that pattern. Data classification is used in various fields, including data analysis, data mining, statistical inference, and machine learning [12].

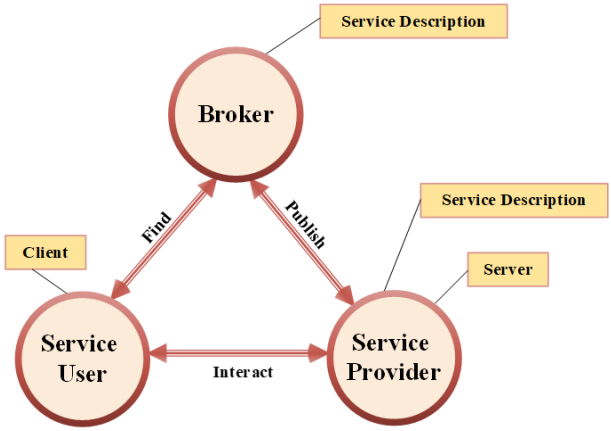

FIGURE 1. Web service architecture.

Machine learning and data mining tools make learning from data and discovering its hidden structure possible [13], [14]. To choose the best service, an automated system that applies machine learning and data mining is essential for system analysis and suggestion [15]. Three types of machine learning methods include supervised, unsupervised, and semi-supervised [16]. Labeled data is used to train the algorithm in supervised learning, where the machine learns a function from input to output [17]. However, obtaining data labels can be a time-consuming and challenging task in many fields. In unsupervised learning, data is unlabeled and referred to as raw data [18]. This type of learning aims to group related data into classes. While real learning often doesn't occur, this method provides the user with awareness of the data's status.

A new approach to learning called semi-supervised learning has been proposed due to the limitations of previous methods. It involves using a small amount of labeled data and a large amount of unlabeled data to create a classification model [19]. Semi-supervised learning combines the conceptual information contained in unlabeled data with the explicit information of labeled data to improve the efficiency and performance of the classifier. The benefits of semi-supervised learning in the context of quality-based web service classifications are manifold. It allows for more efficient use of resources by reducing the need for extensive manual labeling. Additionally, it can enhance the accuracy and robustness of models by incorporating a broader range of information from the unlabeled data. There are various approaches to semi-supervised learning, including self-training, co-training, Ensemble-Co-Training, Tri-training, generative models, Co-Forest, Graph-based, Transductive Support Vector Machine (TSVM), Semi-Supervised SVM (S3VM), and Margin-based Approaches like MSAB and MSSBoost [20], [21], [22]. Semi-supervised learning holds immense potential in the realm of quality-based web service classification. It can help improve the accuracy and efficiency of classification and assessment tasks while reducing the need for extensive manual labeling. Semi-supervised learning algorithms can extract valuable patterns and insights by leveraging labeled and unlabeled data, improving classification and assessment models. This approach enhances the scalability and adaptability of the process. It has two main advantages over other learning techniques: 1) It is not necessary to know the label of all datasets beforehand, unlike supervised learning., and 2) It has higher accuracy than unsupervised learning.

This paper introduces a novel method for quality-based web service classification using self-learning. The primary problem with self-training is that even though a subset of data is selected with high precision for each iteration, some of these selections may be incorrectly labeled in reality. These mistakes are dispersed throughout the learning process, thereby decreasing the accuracy of the underlying classifier. Thus, the main objective of this paper is to establish an alternative selection criterion that can aid in the proper selection of these subsets and enhance the classification accuracy. In this paper, we use the QWS dataset to evaluate the performance of the proposed algorithm [23], [24]. The results of experiments under different scenarios indicate that the proposed algorithm outperforms supervised algorithms in terms of various evaluation criteria such as accuracy, precision, and F1-Score. Additionally, the proposed algorithm includes better results than the supervised method regarding standard deviation values. The main contributions of this paper are summarized as follows:

- Developing SSL-WSC, a semi-supervised learning algorithm that builds a self-training model for the classification of quality-based web services
- Creating a balance between user requests and desirable services for them
- Providing a way to calculate the score for each data point in multi-class datasets
- Proposing a two-step method (based on distance and as well as dynamic threshold-based probability estimation) to select a subset of unlabeled data to add to the labeled dataset
- Reducing human resources costs for labeling web services
- Increasing the accuracy of web services classification
- Providing ideas for future works to improve the classification of web services

The remainder sections of this paper are organized as follows: Section II reviews the related literature, Section III presents the proposed method, Sections IV and V provide

details on the experiments and results, and finally, Section VI includes the conclusion and suggestions for future works.

II. RELATED WORKS

Service providers store their web services using UDDI (Universal Description, Discovery, and Integration) [25]. UDDI is a platform-independent, XML-based registry for businesses to list their web services. It allows service providers to publish information about their services and for consumers to discover and utilize them. Service applicants can search for the services they need on UDDI using only the keywords provided by the service providers. Currently, UDDI requires service providers to register their services in relevant categories, and requesters must search for the appropriate category and related services () [26]. Web services can be classified manually or automatically, but manual classification methods are ineffective due to their large size and functional diversity. Considerable research has been conducted on automated classification techniques using machine learning algorithms of supervised and unsupervised types to address this issue [27], [28], [29], [30], [31]. However, in previous research, the semi-supervised learning method has not been effectively used to classify web services.

Various unsupervised machine learning approaches have been used in web service classifying in different ways. These include fuzzy c-means, expectation-maximization clustering, quality threshold clustering, k-means, kernel k-mean clustering, and density-based clustering [32].

Liu and Wong [33] utilized text-mining techniques to create a representative feature vector for a set of services, which were then classified based on similarity computed using a Grand Similarity criterion. Similarly, Kamath et al. [28] proposed a crawler-based system for gathering descriptions of services and automatically generating labels for each service using similarity analysis techniques. They used these labels and pair-wise service similarity values for hierarchical clustering of services, time optimization, and precision of service discovery. By combining machine learning and text mining methods, Crasso et al. [34] could automate the classification of services based on their semantic descriptions. Katakis et al. [35] extended the feature vector by combining the interface details and semantic labeling of OWL-S service advertisements and then classified it using the Näive Bayes classifier. Wang et al. [36] implemented a hierarchical classification approach that closely resembled the UNSPSC (United Nations Standard Products and Services Code) classification. This approach automatically categorizes services based on domain-specific conditions. The team utilized support vector machines to classify the documents, using the UNSPSC classification as a multilevel tree. Each non-leaf node, or parent class, corresponded to a sub-classification system. A K-Means algorithm was proposed by Xing et al. [37] to cluster document-based services with corresponding labels of the Mashup service, based on the similarity of the Mashup service. Li et al. [29] developed a graph convolutional neural network using residual learning and an attention mechanism that assigned varying weights to the graph nodes independent of the entire graph structure. The residual learning technique enhanced the depth of the network and prevented gradient explosion while acquiring more features.

Supervised algorithms label every web service based on existing classes using training data. These algorithms analyze the training data and create an inferential function to classify new data. Researchers have widely used automated web service classification within predetermined categories. Various machine learning methods, such as Bayesian classifier, decision tree, K-Nearest Neighbor (KNN), support vector machines (SVM), neural networks, latent semantic analysis, genetic algorithms, and more, have been examined to classify web services.

Crasso et al. [38] introduced an Automated Web Service Classification(AWSC) framework that utilized text mining and machine learning techniques to improve the precision of web service classification. To achieve this goal, they examined the relationship between web service categories and standard descriptions. On the other hand, Shafiq et al. [39] proposed a combination of semantics and machine learning to improve the web service discovery process. They used non-functional properties of web services as lightweight semantics and the Bayesian classification technique for dynamic classification. Sharma et al. [30] suggested a hybrid approach incorporating machine learning, data mining, statistical techniques, logical reasoning, and semantic relationship measurement to classify web services automatically. This approach improved classification accuracy by merging semantic information into service profiles and transforming web service profiles into semantically enriched vectors. The proposed approach benefited registry administrators and was used to register and receive better services.

Support vector machines (SVM) and the k-nearest Neighbor (KNN) classifiers were used to evaluate their approach. Chipa et al. [27] explored the applications of supervised machine learning methods such as K-Nearest Neighbors, Support Vector Machine, Decision Trees, and Gaussian Naive Bayes for web service classification. In [31], Vijay et al. presented a new approach with K-Means and improved fuzzy with KNN for automatic query characterization to classify web services. El-Sayyad et al. [1] introduced a new classification algorithm that utilized domain ontology and a semantic similarity-based classifier. This classifier used a dimensionality reduction method to improve performance by removing undifferentiated information. Ye et al. [4] used word embedding to represent words as numerical vectors in their research. They then employed RNN-based or CNN-based neural networks to extract implicit features to classify services. However, these methods often relied on functional description documents, which might lead to data scarcity issues if the documents were insufficient. If the descriptive documents are insufficient, these techniques may not be able to achieve the desired results. In contrast, Sheng et al. [40]

TABLE 1. A comparison of the works done on the classification of web services based on machine learning.

Scheme	Main Idea	Evaluation measures	Using method
[33]	Using text mining techniques	GENIA domain corpus and BNC general corpus	Unsupervised Learning
[28]	Using a crawler-based system	Precision and Recall	Unsupervised Learning
[34]	Using query-by-example method	WSQBE	Unsupervised Learning
[35]	Combining machine learning and text-mining methods	10-fold cross-validation	Unsupervised Learning
[36]	Using a hierarchical classification similar to the UNSPSC	UNSPSC	Unsupervised Learning
[37]	Using the K-Means algorithm based on the Mashup service	Precision and Recall	Unsupervised Learning
[29]	Using the graph convolutional neural network based on the attention mechanism and residual learning.	Top-1 Accuracy Top-5 Accuracy	Unsupervised Learning
[38]	Using the AWSC (Automated Web Service Classification) framework	Accuracy	|Supervised Learning
[39]	Proposing a combination approach of semantics and machine learning	Precision	Supervised Learning
[30]	Merging the semantic information into service profiles	Accuracy, Precision, and Recall	Supervised Learning
[27]	Introducing applications of supervised machine learning methods for classifying web services	Accuracy	Supervised Learning
[31]	Improving fuzzy with KNN for automatic query characterization to classify web services	Precision, Recall, F1-Score, and Support	Supervised Learning
[1]	Using domain ontology and a semantic similarity-based classifier	Precision, Recall, Accuracy, Error, and F-measure	Supervised Learning
[4]	Using RNN-based or CNN-based neural networks	Precision, Recall, F-measure, Purity, and Entropy.	Neural Networks
[40]	Proposing a reinforcement learning approach		Reinforcement Learning
[41]	Combining heterogeneous information networks and generative adversarial networks	F1-Score	Supervised Learning

proposed a reinforcement learning approach for classifying information based on the type of web services that are available on the Internet. This technique can filter out extraneous data and extract the necessary information to accurately classify the desired web services. In [41], Moreno-Vallejo et al. used artificial neural networks to detect and classify fake news, techniques used due to their learning capabilities. The problem of early summarization was introduced by Xie et al. [42], and they proposed a method for categorizing web services that fuse heterogeneous information networks and generative adversarial networks(SCGAN) to address this problem. In Table 1, An evaluation of web services classification by using machine learning is presented.

III. SEMI-SUPERVISED SELF-TRAINING ALGORITHM

As the proposed algorithm is a new semi-supervised self-training algorithm, semi-supervised learning is explained briefly in this section. In semi-supervised machine learning, there are two data points: a small set of labeled data and a more extensive set of unlabeled data. The labeled dataset, $X_l = \{x_1, x_2, \ldots, x_l\}$, contains a known number of data points with corresponding labels, $Y_l = \{y_1, y_2, \ldots, y_k\}$ $(k < l)$. The unlabeled dataset $X_u = \{x_{l+1}, x_{l+2}, \ldots, x_{l+u}\}$ contains a more significant number of data points without corresponding labels. The labeled and unlabeled data are randomly selected from the same distribution [22]. This study uses the QWS dataset with $|X_l| = n_l \ll |X_u| = n_u$.

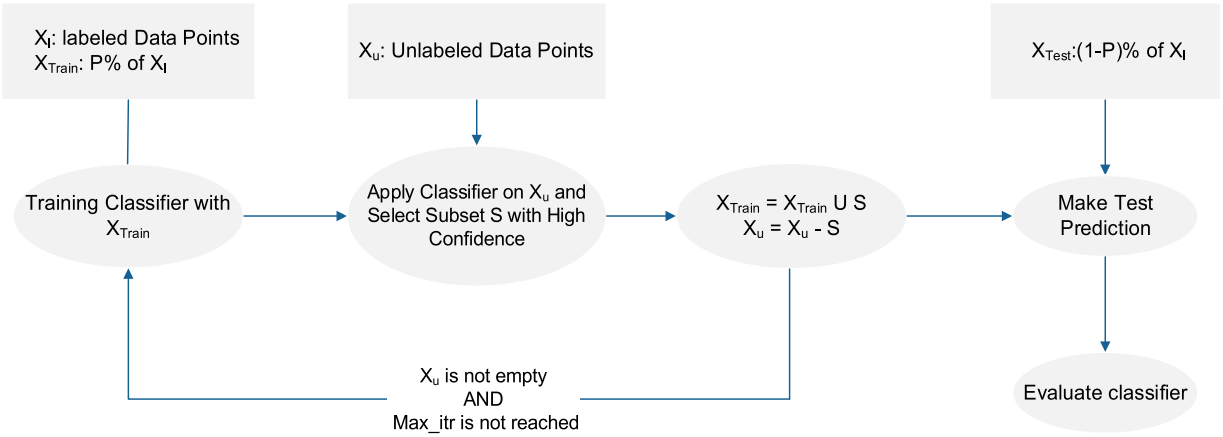

FIGURE 2. An overview of semi-supervised self-training algorithm.

A self-training algorithm is a process that involves using a base classifier, such as a decision tree, to predict and label unlabeled samples [22], [43]. It begins with training on a small set of labeled samples, called a basic training set. The classifier then predicts labeling the unlabeled samples with a degree of confidence (prediction step). From this set, a subset of samples with high-confidence predictions, called S, is selected and added to the set $X_l (X_l = X_l \cup S)$ to form a new dataset (selection step). The classifier is then retrained with this new dataset (retraining step). Typically, S contains several instances of the set X_u with high-confidence predictions for their labels. This process is repeated until there is either no unlabeled data remaining or the maximum number of iterations has been reached. Figure 2 illustrates the flowchart of this self-learning algorithm.

IV. THE PROPOSED SSL-WSC ALGORITHM

Semi-supervised self-training is a commonly used method in machine learning due to its simple structure. However, the challenge lies in selecting accurate data in each iteration. In practice, there may be mislabeled selections, which can spread throughout the learning process and lower the classifier's accuracy. This paper proposes a new selection criterion called SSL-WSC to address this issue. This semi-supervised algorithm aims to classify web services based on their quality, ultimately improving the accuracy.

This algorithm proposes a semi-supervised self-training algorithm with improvements for classifying web services. The SSL-WSC algorithm differs from the standard algorithm in choosing the S subset. In the standard self-training algorithm (Figure 2), the classifier becomes overfitting due to the small number of members in the initial X_{Train} dataset. This issue can cause the probabilities to be very close, with only slight differences. Therefore, selection based on the highest degree of prediction reliability is insufficient and can increase the error rate in subsequent iterations. To avoid this problem, the SSL-WSC algorithm introduces a distance-based method for selecting the set S and the selection based on the highest degree of confidence. The choice of set S is made in two phases:

Phase 1 – Selection Based on Distance: The SSL-WSC algorithm calculates the distance between each instance of the X_u set and the X_{Train} instances belonging to each class (samples with the same label). Four distance values are calculated for each data point in the QWS dataset with four different classes: Platinum, Gold, Silver, and Bronze. The distance calculation algorithms are explained at the end of phase 1.

Afterward, a score is assigned to each sample using several methods to get better results, and the most appropriate method is selected. The considered methods are explained below:

Method 1: In this method, the sample score is determined by calculating the sum of the absolute values of the difference between the two distances for each sample. The top M samples with the highest scores are selected from the samples. For the proposed algorithm, M is optionally set to 20% of the unlabeled data, assuming the selected set is S'. A score of zero indicates that the values of all distances for each sample are equal and the probability of each class chosen for that sample is the same. A lower score implies that the probability of assigning different classes to that sample is about the same (Figure 3.a), making it impossible to consider the label of one of the classes confidently. In contrast, a higher score means that the similarity of the sample to different classes is vastly different, making the label assigned to that sample more certain (Figure 3.c). In Figure 3.a, even though the label C_1 is assigned to sample d_1, labels *C2, C3*, and *C4* may also be assigned. For example, suppose d is an unlabeled data sample that can take its label from four different classes ($k = 4$). Assuming that the distance between data point d and the four classes (*C1, C2, C3,* and *C4*) are 2, 2.5, 6, and 6.5, respectively, the score of data d is calculated using the following formula:

$$\begin{aligned} Score\_1(d) = (|2-2.5|+|2-6|+|2-6.5|)+(|2.5-2| \\ + |2.5-6|+|2.5-6.5|)+(|6-2|+|6 \\ - 2.5|+|6-6.5|)+(|6.5-2|+|6.5-2.5| \\ + |6.5-6|) = 34 \end{aligned} \quad (1)$$

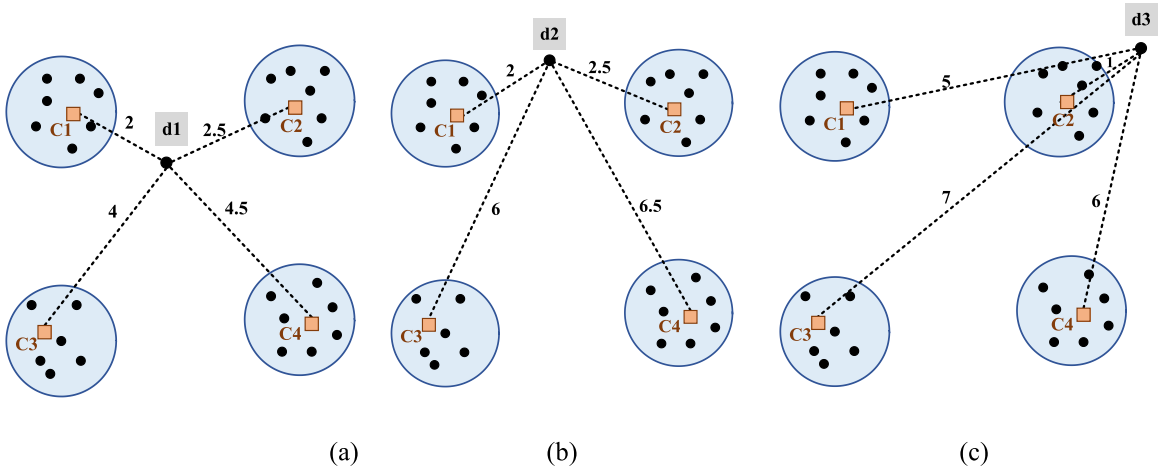

FIGURE 3. Distances of three unlabeled data samples, d1, d2, and d3 from data centers C1, C2, C3, and C4.

Method 2: Based on this method, the score of each sample in the set X_u is calculated by summing up its distances from the X_{Train} samples belonging to each class. The number of M samples is then chosen from the samples with the highest score. A lower score indicates that the sample's similarity to different classes is almost the same, making it difficult to assign a label to that instance. Conversely, a higher score means the sample's similarity to different classes is very different, resulting in a more reliable label assignment. The score for data d is calculated based on the conditions outlined in the first method:

$$Score_2\,(d) = 2 + 2.5 + 6 + 6.5 = 17 \tag{2}$$

Method 3: This method involves calculating the score of each instance of the set X_u by calculating the smallest distance of that instance from the X_{Train} instances belonging to each class. The number of M samples is then chosen from the samples with the lower score. A score of zero indicates that the sample is assigned to a class with perfect confidence, while a lower score means that the data sample is closer to a specific class, and its label is more likely to be the label of the class it is close to (as shown in Figure 3.c). With the conditions stated in the first method for data d, the score of data d is calculated as follows:

$$Score_3\,(d) = 2 \tag{3}$$

Method 4: In this method, a combination of the previous three methods is used to select the best samples for the set S'. The samples with the highest score from the first and second methods and those with the lowest score from the third method are considered. To score the samples in the set X_u, the following equation is used.

$$Score_4\,(d) = \frac{Score_1\,(d) + Score_2\,(d)}{Score_3\,(d)} = \frac{34 + 17}{2} = 25.5 \tag{4}$$

The high score indicates that the values of the first and second methods are much larger than the third, so the selected label is correct for the data sample with high confidence.

In Table 2, those mentioned above four scoring methods are computed and presented for the three data samples *d1*, *d2*, and *d3* of Figure 3. It is worth noting that for $Score_3$ lower scores contribute more to selecting the data sample for set S', while higher scores play a more important role for other methods. As illustrated in Figure 3.a, for the data sample *d1*, choosing a label is a complex task, given that the distances of each data sample from the data centers are almost close values (i.e., 2, 2.5, 4, and 4.5). Therefore, four labels may be chosen for this data sample. In contrast, for data sample *d2* (Figure 3.b), it is evident that *C1* and *C2* labels are more likely to be chosen than *C3* and *C4*, based on the distances (i.e., 2, 2.5, 6, and 6.5) from the data centers. However, one cannot choose between *C1* and *C2* labels with high certainty. Finally, sample *d3* (Figure 3.c), at distances of 1, 5, 6, and 7 from the data centers, can be assigned the label *C2* with high confidence. Therefore, from the three unlabeled data samples, *d1*, *d2*, and *d3*, *d3* should be selected to be added to the set S'. The above description is confirmed by comparing the scores obtained for these three data sets in Table 2. As seen in this table, data *d3* has a higher priority than data *d1* and *d2* in terms of all four scoring methods for selection (It has the highest value for $Score_1$, $Score_2$, and $Score_4$ and the lowest score for $Score_3$).

Among the known distance functions, Mahalanobis, Manhattan, and Minkowski distances were optionally used to calculate the distance at the proposed algorithm. These distance measures are commonly used in various fields, such as machine learning, statistics, and data analysis, to identify the similarity or difference between two or more data points. By utilizing these distance functions, we can accurately determine the distance between data points and make informed decisions based on the calculated results.

Assuming two data $A = (a_1, a_2, \ldots, a_d)$ and $B = (b_1, b_2, \ldots b_d)$. The distances between A and B can be calculated using the following formulas:

TABLE 2. Comparison of four scoring methods using three unlabeled data samples.

Data point	Distances from the center of classes				Four scoring methods			
	C1	C2	C3	C4	$Score_1$	$Score_2$	$Score_3$	$Score_4$
d_1	2	2.5	6	6.5	34	17	2	25.5
d_2	2	2.5	4.5	4	18	13	2	15.5
d_3	5	1	7	6	**38**	**19**	**1**	**57**

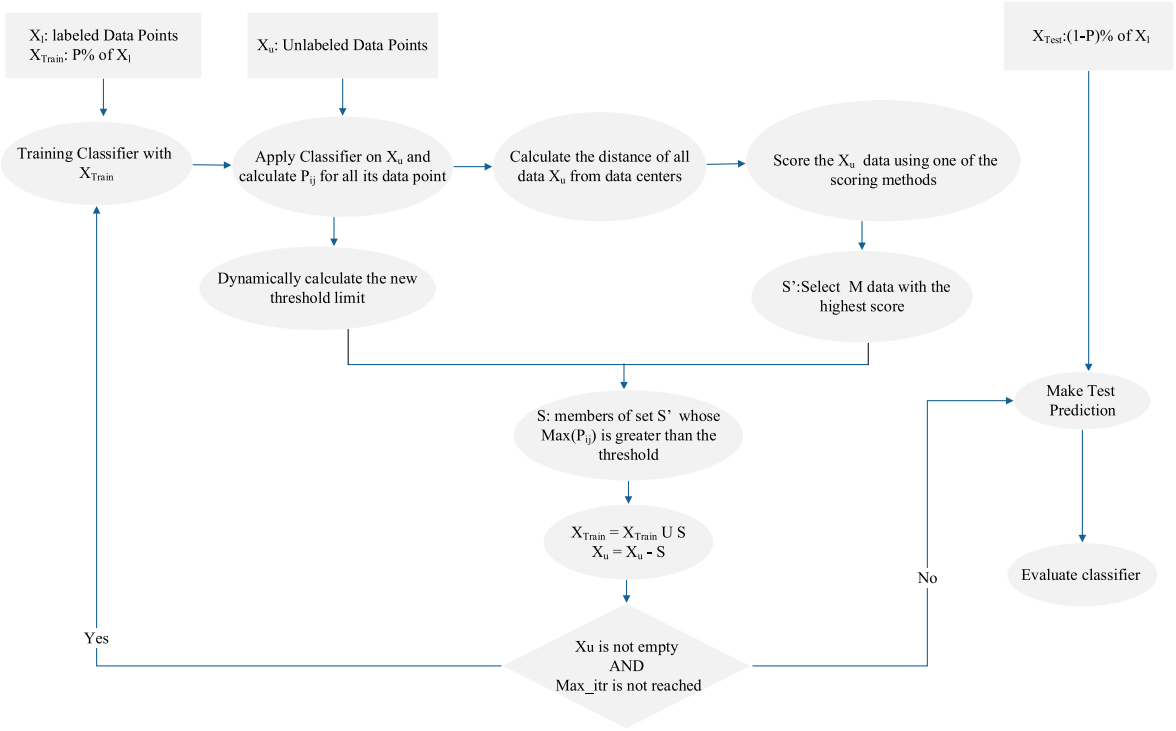

FIGURE 4. Outline of the proposed SSL-WSC algorithm.

The formula gives the Mahalanobis distance between A and B:

$$dist_{\boldsymbol{Mahalanobis}}(A,B) = \sqrt{\sum_{i=1}^{d} \frac{(a_i - b_i)^2}{v_i^2}} \tag{5}$$

where $V = (v_1, v_2, \ldots v_d)$ is the standard deviation of A and B, and d is their dimension.

The formula gives the Manhattan distance between A and B:

$$dist_{\boldsymbol{Manhattan}}(A,B) = \sum_{i=1}^{d} |a_i - b_i| \tag{6}$$

where |. | denotes the absolute value.

The following formula defines the Minkowski distance between A and B in a given n-dimensional space:

$$dist_{\boldsymbol{Minkowski}}(A,B) = \sqrt[q]{\sum_{i=1}^{d} |a_i - b_i|^q} \tag{7}$$

So that $|a_i - b_i|$ represents the absolute difference between the ith coordinate of A and B. The Minkowski distance formula is a generalized version of the distance metric formula, where q is a positive integer representing the distance degree. If q=1, Minkowski distance becomes the Manhattan distance. If q=2, the distance is equivalent to the Euclidean distance. Finally, if q approaches infinity, the Minkowski distance becomes the Chebyshev distance. We have considered the case where q equals 2 for our purposes.

Phase 2 - Selection Based on Reliability: Based on the data selected in phase 1 (set S'), the data whose probability of belonging to a particular class is greater than or equal to the threshold is chosen. These data samples are then added to the X_{Train} set along with the corresponding class label with the highest confidence. However, there may be cases where the maximum probability of belonging a data sample to a particular class may be lower than the threshold value. If this occurs for all data, none of the data will be chosen to be added to the X_{Train} set. To address this, the threshold value is dynamically chosen and updated in each iteration of the algorithm implementation. Thus, the lowest value is the new threshold between the maximum probability of belonging

SSL-WSC Algorithm

Input Parameters:
X_l: Initial labeled data points
X_u: Unlabeled data points
F: Underlying classifier
Max_{iter}: Maximum number of iterations
T: Threshold for selection

1. $X_{Train} \leftarrow p\%$ of X_l
2. $X_{Test} \leftarrow (1-p)\%$ of X_l
3. $t \leftarrow 1$
4. **while** (X_u *is not* empty) AND ($t \leq Max_{iter}$)
5. $Model \leftarrow BaseClassifier\ (X_l, F)$
6. ***Phase1****:*
7. **foreach** d_i in X_u
8. $P_{ij} \leftarrow$ belonging probability of d_i to class C_j $(j = 1 \dots k)$ using $Model$
9. $dist_{ij} \leftarrow$ distance of d_i from the center of the class C_j $(j = 1 \dots k)$ using any distance function such as Mahalanobis
10. $Score_1(d_i) \leftarrow \sum_{\alpha=1}^{k} \sum_{\beta=1,\beta\neq\alpha}^{k} |dist_{i\alpha} - dist_{i\beta}|$
11. $Score_2(d_i) \leftarrow \sum_{\alpha=1}^{k} dist_{i\alpha}$
12. $Score_3(d_i) \leftarrow \min_{\alpha=1\dots k} \{dist_{i\alpha}\}$
13. $Score_4(d_i) \leftarrow (Score_1(d_i) + Score_2(d_i))/Score_3(d_i)$
14. **end foreach**
15. $S' \leftarrow$ Select M data with the highest score (Using one of the methods of lines 10 to 13) from Unlabeled data along with their label //In the case of $Score_2$, fewer scores are preferred
16. ***Phase2****:*
17. $T \leftarrow \min\left\{T, \max\left\{\max\{p_{ij} | j = 1 \dots k\} | \, i = 1 \dots M\right\}\right\}$ //P_{ij} is the probability of belonging ith data from S' to jth class
18. $S \leftarrow \{\}$
19. **foreach** d_i in S'
20. $\mathbb{P}_i \leftarrow \max_{j=1\dots k} \{P_{ij}\}$
21. **if** ($\mathbb{P}_i \geq T$)
22. $S \leftarrow S \cup d_i$
23. **end if**
24. **end for**
25. $X_{Train} \leftarrow X_{Train} \cup S$
26. $X_u \leftarrow X_u - S$
27. $t \leftarrow t + 1$
28. Retrain $Model$ by the new training set X_{Train}
29. **end while**
30. **Output:** Labeling of all data point

FIGURE 5. Proposed SSL-WSC algorithm.

data samples in set S' to a particular class and the input threshold value.

$$T = \min\left\{T, \max\left\{\max\left\{p_{ij} | j = 1 \dots k\right\} | i = 1 \dots M\right\}\right\} \quad (8)$$

In the above equation, p_{ij} is the probability of assigning the i^{th} data of the set S' to the class j, and M is the number of members of the set S'. Figure 4 displays the flowchart of the proposed SSL-WSC algorithm.

In Figure 5, the pseudo-code of the proposed algorithm is presented.

Based on the information provided, the proposed algorithm has a complexity of $O(t \times (M + n_u))$. The values of n_u and

M are equivalent to the number of members of sets X_u and S' respectively, while t refers to the number of iterations of the algorithm during each model training. The t value of the algorithm is set to 10, 20, 30, and 40 for each training step.

V. EXPERIMENTAL RESULTS

This section presents the experimental results of the proposed algorithm for quality-based web services classification. All algorithms have been implemented using Python (version 3.11.7) and executed in a personal computer with an Intel Core i7-3720QM processor and 32GB of RAM. The following subsections explain base classification algorithms, evaluation criteria, and dataset in detail.

A. THE UNDERLYING CLASSIFIER ALGORITHMS

As per the information provided, the experiments used Decision Tree (D.T.), Support-Vector Machines (SVM), Logistic Regression (L.R.), K-Nearest Neighbors (KNN), Gaussian Naive Bayes (N.B.), Random Forest Classifier (R.F.), multilayer perceptron (MLP), and XGBoost (single and ensemble) as the base classifier algorithms. These classifiers were used to evaluate the performance of the proposed SSL-WSC algorithm and compare it with the supervised mode of these classifiers.

TABLE 3. Modified parameters in base classifier settings.

Underlying Classifier	Parameter Settings
Decision Tree	max_depth=3
SVM	probability = True
Logistic Regression	max_iter=1500
Random Forest Classifier	max_depth=25 n_estimators=10 max_features=1
MLP	solver='adam' alpha=1e-3 hidden_layer_sizes= (64,4) random_state=1
XGBoost	objective="multi: SoftMax" random_state=42 learning_rate=0.001 max_depth = 10 n_estimators = 15 eval_metric = 'mlogloss'

Table 3 shows the modified parameter values used for the different classifiers in the experiments. This table provides a comprehensive overview of the parameter values used for each classifier, including the learning rate, maximum depth, number of estimators, and so on. Default values are provided for the parameters that are not listed in the table.

B. EVALUATION MEASURES

In the case of classification, the ultimate objective is to attain the highest possible accuracy and correctly identify the categories. In the field of artificial intelligence, the confusion matrix (Figure 6) is a matrix that represents the performance of algorithms [44]. Each column of the matrix represents the predicted class for each data point, while each row contains the actual class of each data [45]. A more comprehensive evaluation of the model's performance is possible through the confusion matrix. The proposed algorithm has been evaluated and compared to other algorithms based on precision, accuracy, and F1-Score criteria.

Precision is the ratio of true positive samples to the total number of positively predicted samples. Samples that the model labels as positive are known as true positives. On the other hand, False Positives are negative samples that the model incorrectly labels as positive.

$$Precision = \frac{TP}{TP + FP} \tag{9}$$

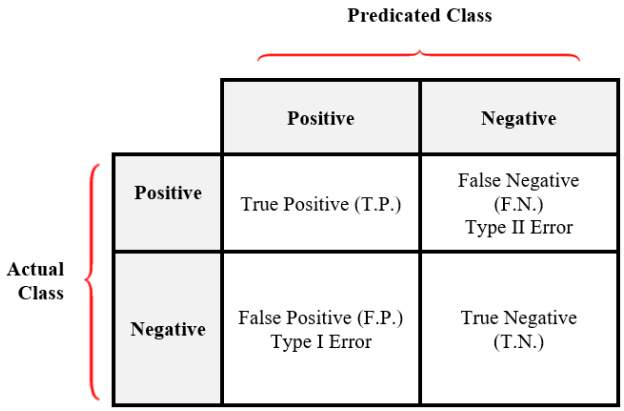

FIGURE 6. Confusion matrix.

As per the formula, the accuracy of a model can be calculated by summing up the True Positive and True Negative samples and dividing it by the sum of all the confusion matrix entries. True Positives and True Negatives refer to the samples correctly classified by the model and located on the main diagonal of the confusion matrix.

$$Accurarcy = \frac{TP + TN}{TP + TN + FP + FN} \tag{10}$$

F1-Score is a statistical criterion used to rate performance, which is calculated as the harmonic mean between recall and precision with equal weight. It is commonly used in Machine Learning to evaluate the accuracy of classification models, with a higher F1-Score indicating better performance [46].

$$F1 - Score = \frac{2.Precision \times Recall}{Precision + Recall} = \frac{2.TP}{2.TP + FP + FN} \tag{11}$$

A Type I Error is a false positive where the model identifies the presence of a condition when it's not there, and a Type II Error is a false negative where the model fails to determine the presence of a condition when it's there. Both errors can have serious consequences depending on the situation [47].

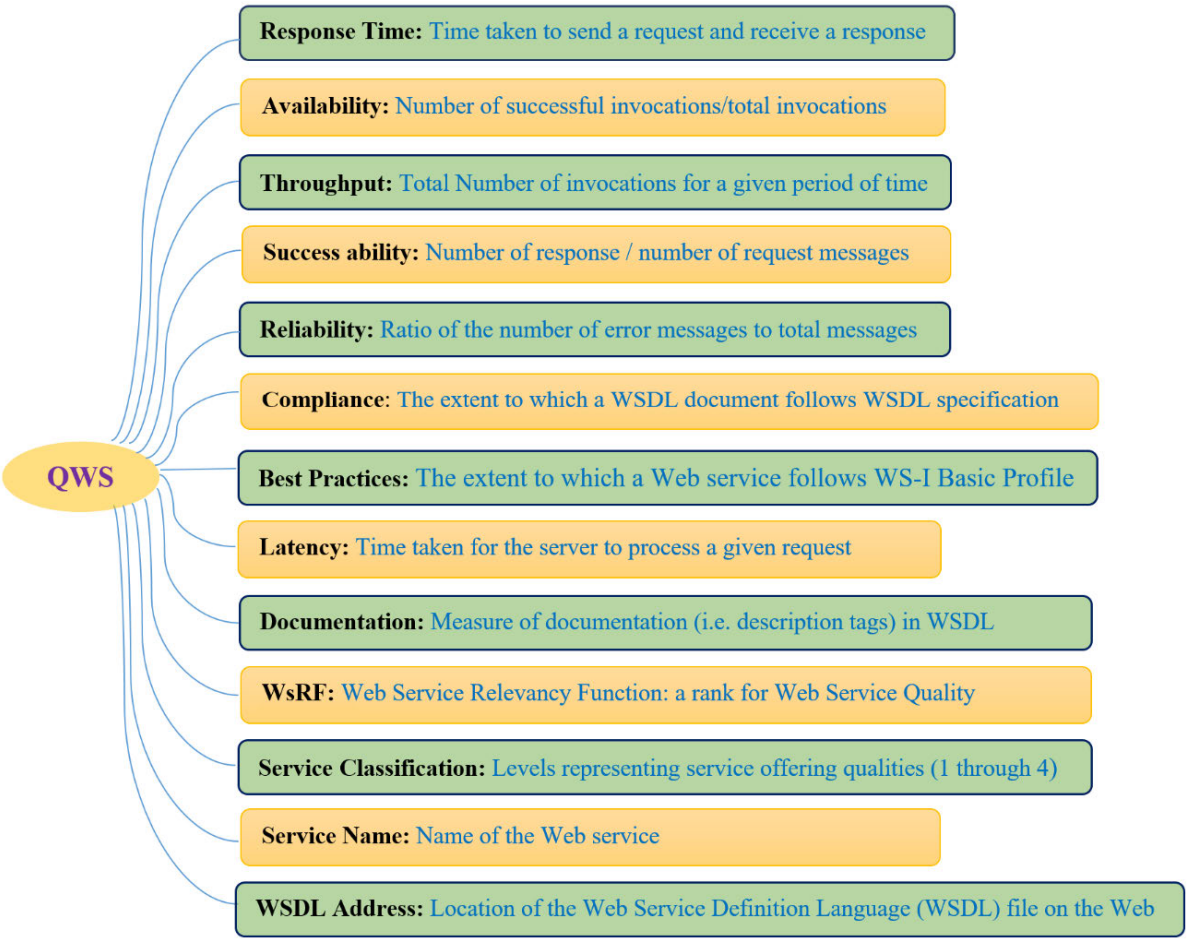

FIGURE 7. The QWS dataset's fields.

C. DATASET

Generating quality datasets for web services is a challenging task that involves discovering services and observing their QoS behavior over time to calculate non-functional feature values [48]. According to the information, only a few datasets are available for web services based on their quality. This paper uses the QWS dataset, which contains information on 2871 real web services. This dataset systematically categorizes 364 labeled web services into four classes. Labels indicate the quality level of each service. Additionally, the dataset includes 2507 unlabeled data [23], [24]. The QWS dataset is a collection of nine quality features that are used to assess web services. These features are pretty comprehensive and cover a range of aspects. The features included in the dataset are Reliability, Availability, Latency, Response Time, Throughput, Success Ability, Compliance, Best Practice, and documentation. These features are essential in ensuring that web services are high quality and can deliver the expected results to users. Also, the labeled data is equipped with a Service Classification feature, which assigns a value ranging from one to four to each data as its label (1: Platinum, 2: Gold, 3: Silver, and 4: Bronze). Figure 7 gives a concise overview of the fields in the QWS dataset, where the first nine boxes represent the dataset features, and the last four boxes provide supplementary information [6].

D. EVALUATION OF PROPOSED SSL-WSC ALGORITHM IN DIFFERENT SCENARIOS

In this subsection, the results obtained from the implementation of the proposed SSL-WSC algorithm using nine base classifier algorithms (D.T., SVM, L.R., KNN, NB, R.F., MLP, and XGBoost as a single and ensemble) for labeling the QWS dataset are compared with the results obtained from the implementation of the supervised algorithm on the same classifiers.

Various scenarios have been considered in the implementation of the proposed SSL-WSC algorithm. The test section size in labeled data (X_{Test}) was set to 20% and 30% in different implementations, which are conventional values in most machine learning implementations. The number of repetitions of the training step was set to 10, 20, 30, and 40. In each iteration, the threshold values of 60, 70, 80, and 90, which were updated dynamically, were considered. The reason for determining these values for the threshold limit is that these

TABLE 4. Results from the implementation of the proposed SSL-WSC algorithm compared to the supervised methods using nine different classifiers. (I: Improved, F: Fixed, W: Worsened.)

F1_Score Measure

Distance	Mahalanobis						Manhattan						Minkowski					
X_{Test} size	20%			30%			20%			30%			20%			30%		
Changes	I	F	W	I	F	W	I	F	W	I	F	W	I	F	W	I	F	W
$Score_1$	9	0	0	7	0	2	9	0	0	8	0	1	9	0	2	9	0	0
$Score_2$	9	0	0	7	0	2	9	0	0	8	0	1	9	0	0	9	0	0
$Score_3$	7	0	2	7	0	2	4	0	5	7	0	2	4	0	5	6	0	3
$Score_4$	9	0	0	7	0	2	9	0	0	8	0	1	7	0	2	7	0	2

(a)

Accuracy Measure

Distance	Mahalanobis						Manhattan						Minkowski					
X_{Test} size	20%			30%			20%			30%			20%			30%		
Changes	I	F	W	I	F	W	I	F	W	I	F	W	I	F	W	I	F	W
$Score_1$	9	0	0	7	0	2	9	0	0	8	0	1	8	1	0	9	0	0
$Score_2$	9	0	0	8	0	1	9	0	0	8	0	1	9	0	0	9	0	0
$Score_3$	8	0	1	7	0	2	7	0	2	7	0	2	7	0	2	7	0	2
$Score_4$	9	0	0	5	1	3	9	0	0	7	0	2	6	1	2	7	0	2

(b)

Precision Measure

Distance	Mahalanobis						Manhattan						Minkowski					
X_{Test} size	20%			30%			20%			30%			20%			30%		
Changes	I	F	W	I	F	W	I	F	W	I	F	W	I	F	W	I	F	W
$Score_1$	9	0	0	7	0	2	9	0	0	6	0	3	8	1	0	7	0	2
$Score_2$	8	0	1	7	0	2	8	0	1	7	0	2	8	0	1	9	0	0
$Score_3$	8	0	1	7	0	2	8	0	1	6	0	3	9	0	0	6	0	3
$Score_4$	9	0	0	4	0	5	6	0	3	6	0	3	6	0	3	7	0	2

(c)

values are close to the results of the accuracy evaluated in labeling for web services. Additionally, the value of M is equivalent to 20% of the unlabeled dataset in each algorithm iteration. The F1-Score criterion values are given in Table 4 -a, accuracy criterion values are in Table 4 -b, and precision criterion evaluation results are in Table 4 -c.

It is important to note that each value in Table 4 is obtained from the average of ten executions of the SSL-WSC algorithm, each time with different parts of the training and test data. The values in the I, F, and W columns indicate whether the SSL-WSC algorithm Improved, Fixed, or Worsened compared to the supervised method in consecutive runs using the nine base classifiers. For example, I = 6, F = 1, and W = 2 means that the results of the proposed algorithm are better than the supervised method for six classifiers, remain fixed in one case, and worsen in two cases.

Based on the results presented in Table 4, it can be concluded that the Mahalanobis method is the most effective method for calculating distances. Additionally, the first scoring method (Score_1) outperforms the others. These findings were consistent when the test section size was 20% of the labeled data. Therefore, the results of the SSL-WSC algorithm are only reported for these specific cases. It is important to note that the proposed algorithm has made significant improvements in all scenarios and conditions.

Typically, the Mahalanobis distance calculation is employed when there is a correlation between features in the dataset. In the case of the QWS dataset, there is a correlation between various features, such as response time with delay and throughput, and accessibility with other features. Therefore, using the Mahalanobis method for distance calculation was suitable for this dataset.

TABLE 5. The average (Avg), maximum (Max), and standard deviation (Std) of F1-score, Accuracy, and Precision measures obtained from the SSL-WSC and supervised algorithm.

			Decision Tree	SVM	Logistic Regression	k-Nearest Neighbors	Naive Bayes	Random Forest	Multilayer Perceptron	XGboost (Ensemble)	XGboost (Single)
F1-Score	SSL-WSC	Avg	40.84%	23.77%	43.91%	46.22%	36.91%	48.41%	36.69%	44.51%	41.07%
		Max	47.07%	30.6%	52.21%	52.27%	47.63%	56.22%	43.49%	52.37%	46.83%
		Std	0.0479	0.0497	0.0481	0.0447	0.0539	0.0359	0.0435	0.0402	0.0498
	Supervised	Avg	37.6%	22.94%	42.24%	45.07%	31.09%	42.04%	27.16%	39.94%	40.18%
		Max	42.56%	32.6%	52.22%	49.62	38.64%	53.47%	37.18%	50.54%	50.54%
		Std	0.0438	0.0480	0.0542	0.0498	0.0597	0.0654	0.0569	0.0626	0.0606
	The amount of improvement of SSL-WSC compared to the Supervised algorithm		8.62%	3.62%	3.95%	2.55%	18.72%	15.15%	35.08 %	11.44%	2.22%
Accuracy	SSL-WSC	Avg	43.42%	36.44%	44.52%	46.58%	38.63%	48.63%	39.45%	44.66%	41.23%
		Max	49.32%	39.73%	52.05%	52.05%	49.32%	56.16%	46.58%	52.05%	46.57%
		Std	0.0438	0.0254	0.043	0.0463	0.0485	0.0359	0.0474	0.0374	0.0496
	Supervised	Avg	38.90%	35.75%	42.6%	45.62%	34.38%	42.47%	31.5%	40.27%	40.55%
		Max	43.84	42.74	52.05	50.68	43.84	53.42	39.73	50.68	50.68
		Std	0.0398	0.0263	0.0532	0.0494	0.0556	0.066	0.0607	0.0604	0.0572
	The amount of improvement of SSL-WSC compared to the Supervised algorithm		11.62%	1.93%	4.5%	2.1%	12.36%	14.50%	25.23%	10.90%	1.68%
Precision	SSL-WSC	Avg	44.29%	24.78%	44.70%	48.62%	44.17%	49.67%	37.05%	45.85%	41.94%
		Max	57.91%	59.32%	52.81%	55.06%	54.67%	59.07%	42.67%	53.40%	48.72%
		Std	0.066	0.1323	0.0463	0.041	0.0567	0.042	0.0334	0.0479	0.0519
	Supervised	Avg	39.59%	21.46%	43.62%	46.56%	43.20%	43.73%	30.65%	40.78%	40.97%
		Max	45.85%	36.63%	52.74%	51.51%	57.19%	57.78%	52.41%	51.5%	51.5%
		Std	0.0598	0.0755	0.0562	0.05	0.0893	0.0748	0.0927	0.0665	0.0643
	The amount of improvement of SSL-WSC compared to the Supervised algorithm		11.87%	15.47%	2.47%	4.42%	2.25%	13.58%	20.88%	12.43%	2.37%

E. THE COMPARISON OF THE SSL-WSC WITH THE SUPERVISED METHOD

In this paper, the evaluation of three parameters - average (Avg), maximum (Max), and standard deviation (Std) of the F1-Score, Accuracy, and Precision measures for the proposed SSL-WSC algorithm and the supervised algorithm are presented. Table 5 compares the SSL-WSC algorithm's implementation results and the supervised mode for various classifiers.

As the results presented in Table 5 demonstrate, the SSL-WSC method yields significant improvements in all three measures for most classifiers. For the F1-Score criterion, the MLP classifier shows a maximum improvement of 35.08%, whereas the XGboost (Single) classifier shows a minimum improvement of 2.22%. For the Accuracy criterion, the MLP classifier shows a maximum improvement of 25.23%, whereas the XGboost (single) classifier shows a minimum improvement of 1.68%. Finally, for the precision criterion, the MLP classifier shows a maximum improvement of 20.88%, whereas the Naive Bayes classifier shows a minimum improvement of 2.25%. Overall, it can be concluded that the SSL-WSC algorithm outperforms the supervised method in all classifiers and evaluation criteria, with the most significant improvement observed in the MLP classifier. On average, the SSL-WSC algorithm increases performance by 11.26%, 9.43%, and 9.53% in F1-Score, accuracy, and precision, respectively. Therefore, using semi-supervised learning algorithms to create classification models can perform better than the supervised method.

Table 5 also shows that the proposed SSL-WSC algorithm works better than the supervised method regarding standard deviation. Smaller values of this parameter in the SSL-WSC algorithm than in the supervised mode indicate less variation in the results of the SSL-WSC algorithm in different implementations. In other words, the algorithm is stable.

Figure 8 displays a graphical representation of the average results obtained for the criteria f1-score, accuracy, and precision for a better comparison. As shown in the Figures, the SSL-WSC algorithm outperformed the supervised method in all classifiers and for all three measures. Therefore, a two-step process is used to score the samples of the set X_u and select the members of the set S by the proposed SSL-WSC

algorithm, which performs better than the supervised method.

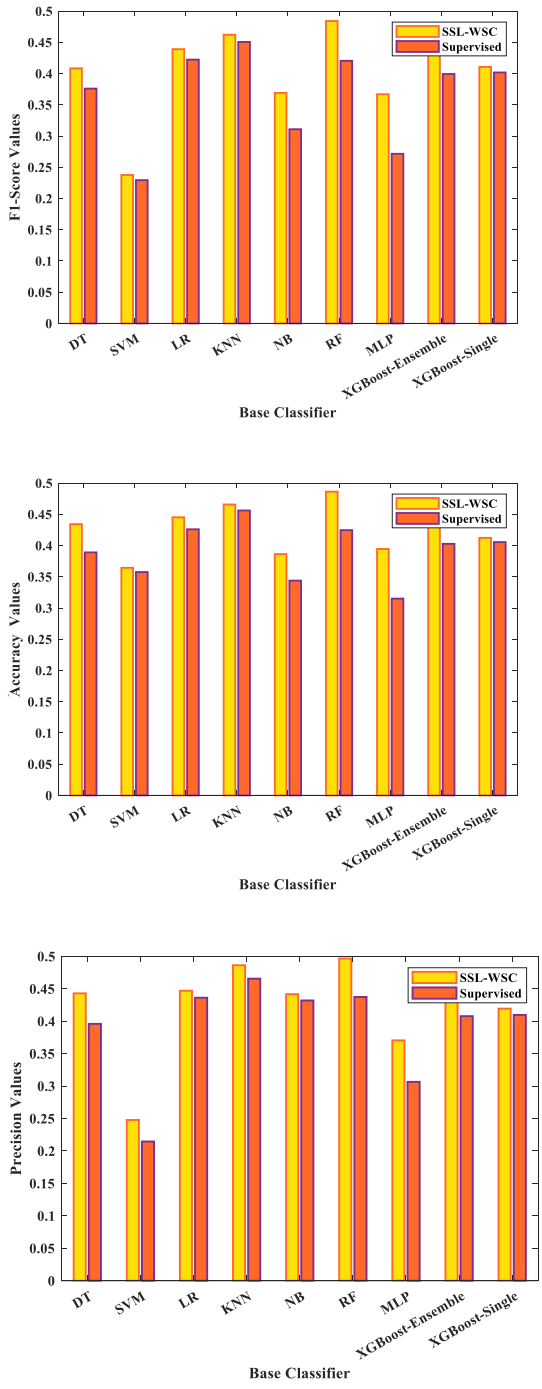

FIGURE 8. Comparison of the proposed algorithm with the supervised method in terms of F1-Score, accuracy, and precision measures.

Considering that the improvement of the proposed SSL-WSC method compared to the supervised method using the MLP classifier is more than other classifiers, in the following, the significance of the improvement obtained by the proposed method based on the MLP classifier is investigated using the T-test. In the tests, the equality of the mean of two independent samples (two F1-score/accuracy/precision sets) is considered as the null hypothesis, H0, and the significance level α is set to 0.05. $p \leq 0.05$ indicates that the H0 can be rejected with the 95% confidence interval.

Looking at the small p-values ($p \leq 0.05$) reported in Table 6, we can conclude that the proposed algorithm's average F1-score/accuracy/precision values are significantly greater than those of the supervised method using the MLP classifier. In other words, the proposed algorithm outperforms the supervised method using the MLP classifier.

TABLE 6. The results obtained from the independent samples T-test on the results obtained from the SSL-WSC method and supervised method using the MLP classifier.

		SSL-WSC algorithm with MLP classifier	Supervised algorithm with MLP classifier
F1-score	Avg.	36.69%	27.16%
	p-value	-	<0.001
Accuracy	Avg.	39.45%	31.5%
	p-value	-	0.008
Precision	Avg.	37.05%	30.65%
	p-value	-	0.017

VI. CONCLUSION AND FUTURE WORK

The sharing of web services is a relatively new application of the Internet. Service brokers act as a go-between for service providers and clients, enabling users to compare and select from a range of ranked services. The critical factor in this ranking is the quality of the services that the user requires. Service providers resort to data mining and machine learning techniques to ensure that users receive the best possible services, using service classification to identify the most appropriate service. However, finding quality-based labels in some fields, such as web services, is challenging and time-consuming. As a result, supervised learning methods may not be the most effective solution. This paper presented a semi-supervised self-training approach addressing classifying web services with minimal labeled data. The proposed method minimizes human effort compared to supervised methods while improving accuracy and scientific significance. During each iteration of the algorithm execution, a two-step semi-supervised self-training learning method is used to add a subset of unlabeled data with high-confidence labels to the labeled set. The first step involves scoring web services using various scoring and distance calculation methods to select unlabeled data with reliable labels. The selected data is compared to a dynamic input threshold in the second step, and some are added to the labeled data. This process is repeated until no unlabeled data remains or the maximum number of iterations has been reached. The proposed algorithm was implemented with various scenarios, considering different parameters such as the size of the test section, the number of repetitions, and classification algorithms. It outperformed the

supervised method in evaluating quality-based web services across different measures, including F1-score, accuracy, and precision. In web services, there are limited datasets due to production issues. Additionally, only a few quality features are considered in these datasets. Future work must be done to increase web service quality features to improve classification confidence.

REFERENCES

[1] S. E. El-Sayyad, A. I. Saleh, and H. A. Ali, "A new semantic web service classification (SWSC) strategy," *Cluster Comput.*, vol. 21, no. 3, pp. 1639–1665, Sep. 2018.

[2] K. C. Li, Y. Xia, F. Xie, W. Liang, and M. Tang, "Predicting new composition relations between web services via link analysis," *Int. J. Comput. Sci. Eng.*, vol. 20, no. 1, p. 88, 2019.

[3] B. Al-Shargabi, S. Al-Jawarneh, and S. Hayajneh, "A cloudlet based security and trust model for e-government web services," *J. Theor. Appl. Inf. Technol.*, vol. 98, no. 1, pp. 27–37, 2020.

[4] H. Ye, "Web services classification based on wide & Bi-LSTM model," *IEEE Access*, vol. 7, pp. 43697–43706, 2019.

[5] K. Zhao, J. Liu, Z. Xu, X. Liu, L. Xue, Z. Xie, Y. Zhou, and X. Wang, "Graph4Web: A relation-aware graph attention network for web service classification," *J. Syst. Softw.*, vol. 190, Aug. 2022, Art. no. 111324.

[6] M. Masdari, M. N. Bonab, and S. Ozdemir, "Correction to: QoS-driven metaheuristic service composition schemes: A comprehensive overview," *Artif. Intell. Rev.*, vol. 55, no. 2, p. 1605, Feb. 2022.

[7] Q. She, X. Wei, G. Nie, and D. Chen, "QoS-aware cloud service composition: A systematic mapping study from the perspective of computational intelligence," *Expert Syst. Appl.*, vol. 138, Dec. 2019, Art. no. 112804.

[8] P. B. Pandharbale, S. N. Mohanty, and A. K. Jagadev, "QoS-aware web services recommendations using dynamic clustering algorithms," *Int. J. Inf. Syst. Model. Design*, vol. 13, no. 6, pp. 1–16, Sep. 2022.

[9] P. Bagga, A. Joshi, and R. Hans, "QoS based web service selection and multi-criteria decision making methods," *Int. J. Interact. Multimedia Artif. Intell.*, vol. 5, no. 4, p. 113, 2019.

[10] S. Rangarajan and R. K. Chandar, "QoS-based architecture for discovery and selection of suitable web services using non-functional properties," *ICST Trans. Scalable Inf. Syst.*, vol. 4, no. 12, Jan. 2017, Art. no. 152102.

[11] M. Ghobaei-Arani, A. A. Rahmanian, M. S. Aslanpour, and S. E. Dashti, "CSA-WSC: Cuckoo search algorithm for web service composition in cloud environments," *Soft Comput.*, vol. 22, no. 24, pp. 8353–8378, Dec. 2018.

[12] Z. Ivezić, *Statistics, Data Mining, and Machine Learning in Astronomy: A Practical Python Guide for the Analysis of Survey Data*. Princeton, NJ, USA: Princeton Univ. Press, 2019.

[13] S. L. Brunton, B. R. Noack, and P. Koumoutsakos, "Machine learning for fluid mechanics," *Annu. Rev. Fluid Mech.*, vol. 52, no. 1, pp. 477–508, Jan. 2020.

[14] M. J. Kaur, V. P. Mishra, and P. Maheshwari, "The convergence of digital twin, IoT, and machine learning: Transforming data into action," in *Digital Twin Technologies and Smart Cities*. Cham, Switzerland: Springer, 2020, pp. 3–17.

[15] M. Hasnain, "Machine learning methods for trust-based selection of web services," *KSII Trans. Internet Inf. Syst.*, vol. 16, no. 1, pp. 38–59, 2022.

[16] Y. Qin, S. Ding, L. Wang, and Y. Wang, "Research progress on semi-supervised clustering," *Cognit. Comput.*, vol. 11, no. 5, pp. 599–612, Oct. 2019.

[17] S. Pattanayak and S. John, *Pro Deep Learning With Tensorflow*. Cham, Switzerland: Springer, 2017.

[18] S. Shalev-Shwartz and S. Ben-David, *Understanding Machine Learning: From Theory to Algorithms*. Cambridge, U.K.: Cambridge Univ. Press, 2014.

[19] J. M. Duarte and L. Berton, "A review of semi-supervised learning for text classification," *Artif. Intell. Rev.*, vol. 56, no. 9, pp. 9401–9469, Sep. 2023.

[20] S. Khezri, J. Tanha, A. Ahmadi, and A. Sharifi, "STDS: Self-training data streams for mining limited labeled data in non-stationary environment," *Appl. Intell.*, vol. 50, no. 5, pp. 1448–1467, May 2020.

[21] J. Tanha, M. van Someren, and H. Afsarmanesh, "Ensemble based co-training," in *Proc. 23rd Benelux Conf. Artif. Intell.*, 2011, pp. 1–8.

[22] J. Tanha, M. van Someren, and H. Afsarmanesh, "Semi-supervised self-training for decision tree classifiers," *Int. J. Mach. Learn. Cybern.*, vol. 8, no. 1, pp. 355–370, Feb. 2017.

[23] E. Al-Masri and Q. H. Mahmoud, "QoS-based discovery and ranking of web services," in *Proc. 16th Int. Conf. Comput. Commun. Netw.*, Aug. 2007, pp. 529–534.

[24] E. Al-Masri and Q. H. Mahmoud, "Toward quality-driven web service discovery," *IT Prof.*, vol. 10, no. 3, pp. 24–28, May 2008.

[25] B. S. Balaji, S. Balakrishnan, K. Venkatachalam, and V. Jeyakrishnan, "RETRACTED ARTICLE: Automated query classification based web service similarity technique using machine learning," *J. Ambient Intell. Humanized Comput.*, vol. 12, no. 6, pp. 6169–6180, Jun. 2021.

[26] M. S. Das, A. Govardhan, and D. V. Lakshmi, "Classification of web services using data mining algorithms and improved learning model," *TELKOMNIKA (Telecommun. Comput. Electron. Control)*, vol. 17, no. 6, p. 3191, Dec. 2019.

[27] M. Chippa, A. Priyadarshini, and R. Mohanty, "Application of machine learning techniques to classify web services," in *Proc. IEEE Int. Conf. Intell. Techn. Control, Optim. Signal Process. (INCOS)*, Apr. 2019, pp. 1–7.

[28] S. Kamath and V. Ananthanarayana, "Similarity analysis of service descriptions for efficient web service discovery," in *Proc. Int. Conf. Data Sci. Adv. Anal. (DSAA)*, Oct. 2014, pp. 142–148.

[29] B. Li, Z. Li, and Y. Yang, "Residual attention graph convolutional network for web services classification," *Neurocomputing*, vol. 440, pp. 45–57, Jun. 2021.

[30] S. Sharma, J. S. Lather, and M. Dave, "Semantic approach for web service classification using machine learning and measures of semantic relatedness," *Service Oriented Comput. Appl.*, vol. 10, no. 3, pp. 221–231, Sep. 2016.

[31] C. Viji, J. B. Raja, R. S. Ponmagal, S. T. Suganthi, P. Parthasarathi, and S. Pandiyan, "RETRACTED: Efficient fuzzy based K-nearest neighbour technique for web services classification," *Microprocessors Microsyst.*, vol. 76, Jul. 2020, Art. no. 103097.

[32] A. E. Ezugwu, A. K. Shukla, M. B. Agbaje, O. N. Oyelade, A. José-García, and J. O. Agushaka, "Automatic clustering algorithms: A systematic review and bibliometric analysis of relevant literature," *Neural Comput. Appl.*, vol. 33, no. 11, pp. 6247–6306, Jun. 2021.

[33] W. Liu and W. Wong, "Web service clustering using text mining techniques," *Int. J. Agent-Oriented Softw. Eng.*, vol. 3, no. 1, p. 6, 2009.

[34] M. Crasso, A. Zunino, and M. Campo, "Easy web service discovery: A query-by-example approach," *Sci. Comput. Program.*, vol. 71, no. 2, pp. 144–164, Apr. 2008.

[35] I. Katakis, "On the combination of textual and semantic descriptions for automated semantic web service classification," in *Proc. IFIP Int. Conf. Artif. Intell. Appl. Innov.* Cham, Switzerland: Springer, 2009, pp. 1–10.

[36] H. Wang, Y. Shi, X. Zhou, Q. Zhou, S. Shao, and A. Bouguettaya, "Web service classification using support vector machine," in *Proc. 22nd IEEE Int. Conf. Tools With Artif. Intell.*, vol. 1, Oct. 2010, pp. 3–6.

[37] H. Xing, "MSCA: Mashup service clustering approach integrating K-means ans Agnes algorithms," *J. Chin. Comput. Syst.*, vol. 36, no. 11, pp. 2492–2497, 2015.

[38] M. Crasso, A. Zunino, and M. Campo, "AWSC: An approach to web service classification based on machine learning techniques," *Inteligencia Artif.*, vol. 12, no. 37, pp. 25–36, Mar. 2008.

[39] O. Shafiq, "Light-weight semantics and Bayesian classification: A hybrid technique for dynamic web service discovery," in *Proc. IEEE Int. Conf. Inf. Reuse Integr.*, Aug. 2010, pp. 121–125.

[40] H. Sheng, Z. Li, J. Liu, and X. Zhang, "Web service classification based on reinforcement learning and structured representation learning," in *Proc. Int. Conf. Artif. Intell. Blockchain Technol. (AIBT)*, Dec. 2021, pp. 21–27.

[41] P. X. Moreno-Vallejo, G. K. Bastidas-Guacho, P. R. Moreno-Costales, and J. J. Chariguaman-Cuji, "Fake news classification web service for Spanish news by using artificial neural networks," *Int. J. Adv. Comput. Sci. Appl.*, vol. 14, no. 3, pp. 1–7, 2023.

[42] X. Xie, J. Liu, B. Cao, M. Peng, G. Kang, Y. Wen, and K. K. Fletcher, "A services classification method based on heterogeneous information networks and generative adversarial networks," *Int. J. Web Services Res.*, vol. 20, no. 1, pp. 1–17, Mar. 2023.

[43] Z. Liu, T. Wen, W. Sun, and Q. Zhang, "Semi-supervised self-training feature weighted clustering decision tree and random forest," *IEEE Access*, vol. 8, pp. 128337–128348, 2020.

[44] M.-T. Wu, "Confusion matrix and minimum cross-entropy metrics based motion recognition system in the classroom," *Sci. Rep.*, vol. 12, no. 1, pp. 1–10, Feb. 2022.
[45] M. Hasnain, M. F. Pasha, I. Ghani, M. Imran, M. Y. Alzahrani, and R. Budiarto, "Evaluating trust prediction and confusion matrix measures for web services ranking," *IEEE Access*, vol. 8, pp. 90847–90861, 2020.
[46] M. Grandini, E. Bagli, and G. Visani, "Metrics for multi-class classification: An overview," 2020, *arXiv:2008.05756*.
[47] S. Ruuska, W. Hämäläinen, S. Kajava, M. Mughal, P. Matilainen, and J. Mononen, "Evaluation of the confusion matrix method in the validation of an automated system for measuring feeding behaviour of cattle," *Behavioural Processes*, vol. 148, pp. 56–62, Mar. 2018.
[48] N. Agarwal, G. Sikka, and L. K. Awasthi, "Enhancing web service clustering using length feature weight method for service description document vector space representation," *Expert Syst. Appl.*, vol. 161, Dec. 2020, Art. no. 113682.

MEHDI NOZAD BONAB was born in Tabriz, Iran. He received the B.Sc. degree in computer software engineering from Islamic Azad University, Shabestar Branch, Iran, in 2002, and the M.Sc. degree in computer software engineering from Islamic Azad University, Qazvin Branch, Iran, in 2007. He is currently pursuing the Ph.D. degree with the Department of Computer Science, Islamic Azad University, Urmia Branch, Iran. Since 2007, he has been a Faculty Member of Islamic Azad University, Marand Branch, Iran. His main research fields are data mining and machine learning.

JAFAR TANHA was born in Bonab, Iran. He received the B.Sc. and M.Sc. degrees in computer science from the University of Amirkabir (Polytechnic), Tehran, Iran, in 1999 and 2001, respectively, and the Ph.D. degree in computer science-artificial intelligence from the University of Amsterdam (UvA), Amsterdam, The Netherlands, in 2013. He was with the INL Institute, Leiden, The Netherlands, as a Researcher, from 2013 to 2015. Since 2015, he has been with the Department of Computer Engineering, Payame-Noor University, Tehran, where he was an Assistant Professor. He has held lecturing positions with Iran University of Science and Technology, Tehran, in 2016. His current position is an Assistant Professor with the University of Tabriz, Tabriz, Iran. His main areas of research interests include machine learning, pattern recognition, data mining, and document analysis.

MOHAMMAD MASDARI received the B.Tech. degree in computer software engineering from Islamic Azad University, Qazvin Branch, Iran, in 2001, the M.Tech. degree in computer software engineering from Islamic Azad University, South Tehran Branch, Tehran, Iran, in 2003, and the Ph.D. degree in computer software engineering from Islamic Azad University, Science and Research Branch, Tehran, in 2014. Since 2003, he has been a Faculty Member of Islamic Azad University, Urmia Branch, Iran, where he is currently an Associate Professor with the Department of Computer Engineering. His research interests include distributed systems and network security.