Computer & Robotics

Enhancing the accuracy of Quality-Based Web Service Categorization via Advanced Feature Development

Mehdi Nozad Bonab[a], Jafar Tanha [*,b], Mohammad Masdari[a]

[a]*Department of Computer Engineering, Urmia Branch, Islamic Azad University, Urmia, Iran*
[b]*Electrical and Computer Engineering Department, University of Tabriz, Tabriz, Iran*
Received 22 August 2024, Accepted 06 Octobger 2024

Abstract

Web services facilitate machine-to-machine communication over the Internet and require precise classification to ensure reliable and efficient service delivery. Accurate classification plays a crucial role in service discovery, recommendation systems, and service composition. Web service brokers help users select the most suitable service based on quality parameters. Currently, there is a limited availability of datasets focused on web service quality; among them, the QWS dataset — containing nine quality features — is one of the most prominent. However, this dataset omits important non-functional attributes such as security, interoperability, scalability, and robustness, which are vital for effective web service discovery. In this study, we propose enhancing the QWS dataset through feature engineering to derive additional informative features from existing ones. Experimental results using the SSL-WSC algorithm demonstrate that this approach significantly improves web service classification performance, evidenced by a 5.05% increase in F1-Score, a 5.69% boost in accuracy, and a 6.92% rise in precision.

Keywords: Web services, classification, quality, feature engineering, machine learning

1.Introduction

Web services are increasingly used for interoperability and communication among diverse systems and applications [1]. Brokers play a crucial role in provisioning and managing web services. They act as intermediaries between users and service providers, helping to establish connections. By aggregating and integrating user requirements, brokers enable cost reduction in service utilization. Moreover, brokers simplify the access to web services for users, allowing them to search, compare, and select appropriate services (Fig. 1) [2]. Additionally, brokers can manage and allocate resources necessary for service consumption, monitor the delivery of quality services by applying quality criteria, and contribute significantly to ensuring the security and privacy of interactions between users and web services.

The classification of web services is a crucial task in service-oriented computing as it aids in discovering and efficiently utilizing web services based on their features [3]. A ranking system calculates the relative value of different services based on the user's required service quality and the characteristics of available services. Once comparisons with other services are made, the system can recommend the appropriate service to the user. The recent proliferation of web

* Corresponding author:
E-mail address:jtanha.2022@gmail.com (J. Tanha).

service providers has led to an increase in the number of web services offering similar functionalities. The main difference among these similar web services is their performance quality. The service quality component in the web services field encompasses non-functional features such as cost and execution time, availability, success rate, and security, among others. Only a few datasets based on web service quality are available in web services. This is because generating a QoS dataset is a challenging, expensive, and time-consuming task as services need to be discovered, and their QoS behavior observed over time to calculate non-functional feature values [4]. Therefore, gathering service information from various sources and generating dataset structures requires more human effort.

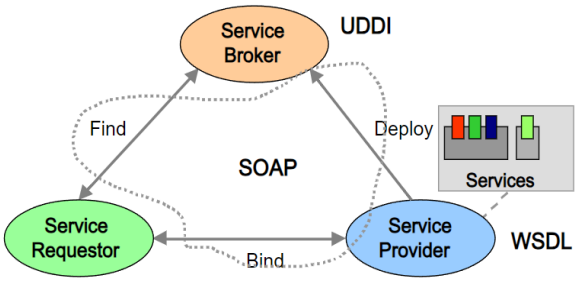


Fig. 1. Web service architecture.

Web services embody three fundamental components: Simple Object Access Protocol (SOAP), Universal Description, Discovery, and Integration (UDDI), and Web Services Description Language (WSDL)[5]. SOAP is a lightweight XML-driven protocol designed for data exchange in decentralized and distributed environments, working autonomously of platforms and programming languages [6]. SOAP over HTTP streamlines communication processes through proxies and firewalls, offering unparalleled ease compared to conventional remote execution technologies. UDDI is a robust registry that facilitates the publication of service specifics and operates as a structured directory system [7]. As a platform-neutral framework, UDDI enables the comprehensive description of services, the exploration of enterprises, and the integration of commercial services utilizing the Internet. Furthermore, as an XML-centered language, WSDL establishes a robust mechanism for the universal exposure of web service definitions, setting unwavering standards that web service providers must adhere to when transmitting SOAP messages.

The QWS dataset contains qualitative attributes of 2871 web services, with 364 services categorized into four quality levels[8]. However, the dataset only includes nine quality features and overlooks crucial non-functional features such as security, interoperability, scalability, and robustness. These non-functional features are essential for applications related to national security and financial transactions. One of the critical technologies for deriving insights from data and uncovering underlying patterns is the field of machine learning and data mining [9, 10]. Given the need to select the best service through system analysis and recommendations, data mining and machine learning technologies are vital for creating an automated system in this domain [11].

Acquiring precise information about services entails collecting data from multiple service providers. There are diverse methods and approaches to data mining, each tailored for specific applications to extract various types of knowledge. In the available dataset, only a small part of the available data has been labeled, while a significant part needs to be labeled. Our previous work introduced the SSL-WSC algorithm [4] using a semi-supervised learning approach [12]. This algorithm used a two-step process to label the unlabeled data within the QWS dataset, resulting in enhanced performance in service classification based on three evaluation metrics: F1-score, Accuracy, and Precision.

In today's dynamic digital landscape, the accurate classification of web services is essential for optimizing user experience and ensuring the delivery of high-quality services. Feature engineering lies at the heart of this effort, significantly enhancing the performance of machine learning models by introducing new features or modifying existing ones. This process is paramount in fully leveraging machine learning for web service classification, organization, and optimization, ultimately resulting in heightened service quality and user satisfaction. In the field of web services classification, various algorithms have been presented. Our goal in this paper is to improve the efficiency of existing algorithms by enhancing the QWS dataset. This is done by adding additional features through feature engineering, which increases classification accuracy by more accurately labeling unlabeled services. We have named this new dataset extended QWS (EQWS) .The key contributions of this paper can be outlined as follows:

- Enhancing the performance of the semi-supervised SSL-WSC algorithm to better classify and access the required web services based on quality.

- Providing empirical formulas to create new features using feature engineering for QWS datasets and generate new EQWS datasets.
- Striking a balance between user demands and the optimal web services for them
- Decreasing the costs associated with gathering additional information on web services.
- Improving the accuracy of web service classification.
- Presenting suggestions for future work aimed at enhancing the classification of web services.

The remaining sections of the paper are organized as follows: the second section reviews related works, The third section introduces the EQWS, the fourth section explains the experiments and results, and finally, the fifth section includes conclusions and suggestions for future work.

2.Related Works

To improve the web services features in datasets, one can consider exploring methods such as feature engineering, feature extraction, and feature selection. Feature engineering is the process of creating new features from the existing set of features. This can be achieved by transforming the existing features or combining them in a new way. For example, if the dataset contains a feature for the service date, feature engineering can be used to generate new features like the day of the week or time of day. Feature engineering requires domain expertise and creativity to identify new features useful for the analysis [13].

Feature extraction is the process transforming existing features into a new set of features. This can be achieved using techniques such as Principal Component Analysis (PCA) or Singular Value Decomposition (SVD). These methods help in reducing the dimensionality of the dataset while retaining the most important information. Feature extraction is particularly useful when dealing with datasets that have a large number of features and the aim is to simplify the analysis [14].

Feature selection involves selecting the most relevant features from the existing features. This can be done using techniques such as correlation analysis or mutual information. Feature selection helps to reduce the dimensionality of the dataset, which can improve the performance of machine learning models and reduce overfitting [15].

The importance and challenges of feature engineering for classification issues in machine learning were addressed by Nargesian et al. [13]. They provided practical solutions for designing, selecting, and combining effective features, including investigating feature correlations, using advanced feature selection methods, creating new features from combining existing features, analyzing the physical meaning of features, and evaluating models with the designed features. Additionally, Rodriguez et al. [16] presented a new method for feature extraction from web services that focuses on analyzing the text and structure of their WSDL. This method aims to identify essential features for categorizing and discovering web services. The results suggested that using features extracted by this method leads to improved performance in categorizing web services compared to other feature extraction approaches. Furthermore, this method has low computational complexity, making it suitable for practical applications. However, the limitations of this method include focusing only on the textual and structural features of WSDL, a lack of attention to dynamic web services, no evaluation in real environments, and limited comparison with other advanced feature extraction methods.

Kanter et al. [17] have developed a new method based on Collaborative Filtering to identify essential features in web services. The main purpose of this method is to enhance the performance of categorizing and discovering web services. This approach can effectively pinpoint valuable features by analyzing the interactions between features and web services. In another study, Srivastava et al. introduced a technique termed "Dropout" to address the issue of "Over-fitting" in neural networks[18]. Overfitting occurs when a neural network becomes overly reliant on its training data, leading to poor performance when presented with new and unfamiliar data. The "Dropout" technique randomly excludes some neurons from the model during the training process. This prevents the network from becoming overly reliant on specific training data and thus improves its performance when dealing with new data. This simple yet effective technique can directly enhance the performance of machine learning models and influence the selection process of appropriate features.

Chia et al. [19] have introduced a groundbreaking approach to adaptive feature engineering, significantly enhancing performance in ranking systems and user-specific recommendations. This method automatically derives effective features from

the data using deep learning. The article's empirical findings unequivocally demonstrate that this approach outperforms existing methods. However, its application has been predominantly focused on addressing ranking and recommendation issues, with the potential for further exploration in other areas of machine learning.

The methods proposed for creating new features each have their strengths and weaknesses. The choice of method depends on the data type and the analysis's ultimate goal. Feature engineering techniques can be used to generate new interoperability, security, scalability, and robustness features from the existing features in the QWS dataset.

3. Extended QWS(EQWS)

This section introduces the QWS dataset in detail, and the new features this article considers to be added to this dataset are examined from different aspects. Additionally, it presents the formulas for obtaining the values of each of these new features.

3.1. QWS Dataset

It is challenging to discover services and observe their QoS behavior over time to compute the values of their non-functional features for creating a quality dataset. In reality, only a few datasets for web services based on their quality are available.

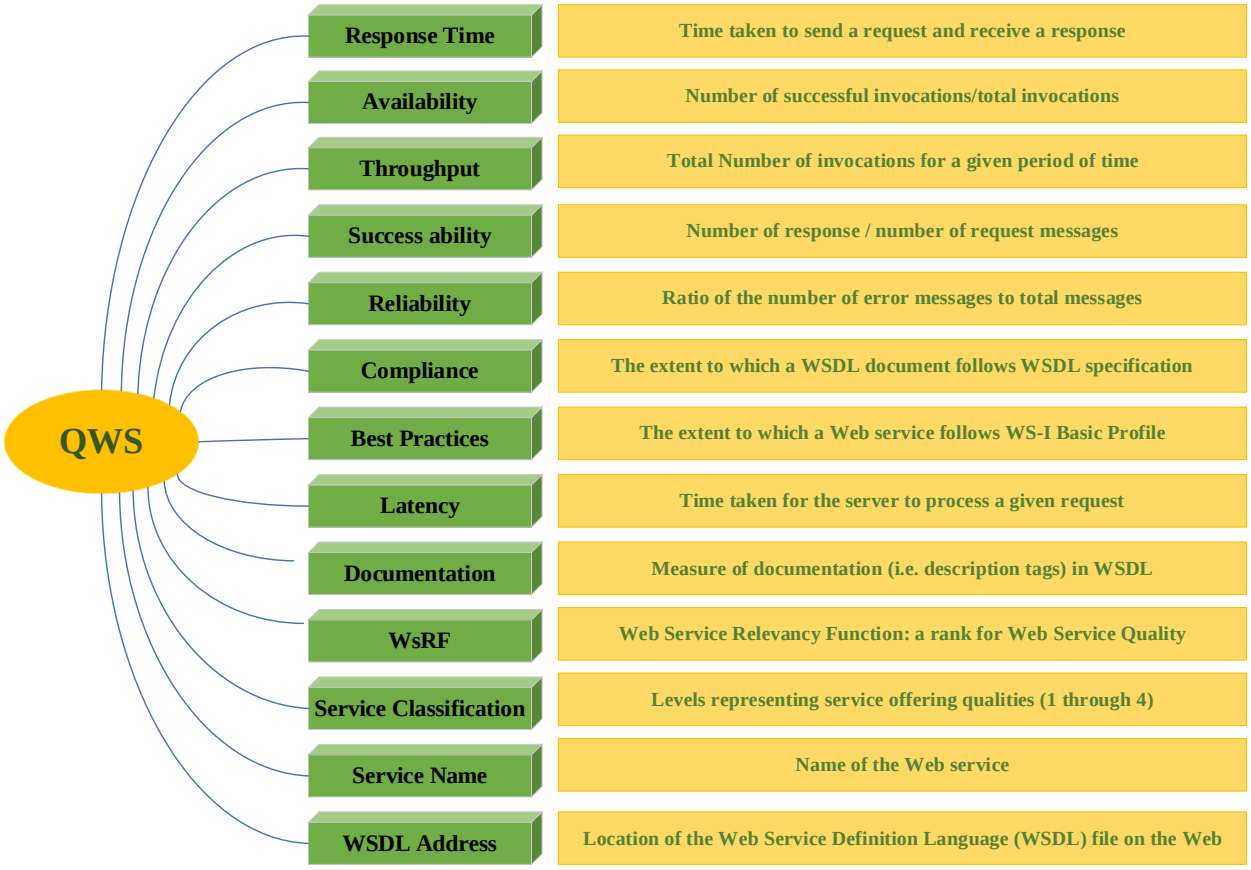


Fig. 2. QWS Dataset Fields

This paper utilizes the QWS dataset, which contains details of 2871 real web services. Within this dataset, 364 web services are categorized into four classes based on their quality, and 2507 unlabeled data points exist. The QWS dataset consists of 9 quality features used to evaluate web services: Response Time, Availability, Throughput, Success ability, Reliability, Compliance, Best Practices, latency, and documentation. These Features are essential for ensuring high-quality web services that deliver expected results to users. Furthermore, the labeled data includes a service classification feature that assigns a value of one to four, representing the quality level of each service (1: Platinum, 2: Gold, 3: Silver, and 4: Bronze). Fig. 2 briefly overviews the QWS dataset fields, where the first nine boxes display

the dataset features and the last four boxes present supplementary information.

3.2. QWS with new features

This paper introduces new features to enrich the dataset by establishing connections between these features and target classes. New features can enhance classification accuracy by capturing more nuanced characteristics of web services. For this purpose, we extract new features such as Interoperability, Security, Scalability, and Robustness, which are inherently sensitive parameters, using specific formulas from existing features. Therefore, generating new features will create a new EQWS dataset that will enable access to a class of services for defense programs related to national security, heavy financial transactions, etc.

A. Interoperability

Interoperability refers to the ability of web services to work together with other web services. This feature can measure the compatibility of web services with other system services. Additionally, this feature enhances classification accuracy by recording service compatibility and ease of integration with external entities. For instance, one can calculate the percentage of requests successfully processed by other web services in the system. A combination of existing features with specific relationships can be utilized to extract the new interoperability feature from the QWS dataset. Table 1 gives some suggestions on how to extract the Interoperability feature from the features present in the QWS dataset:

Table 1
The impact of existing features' value on creating Interoperability feature

Features	Effect of feature(s) on creating Interoperability feature
Compliance and Documentation	A service compliant with standards and well-documented is more likely to be interoperable with other systems.
Reliability and Availability	High reliability and immediate availability can signify improved interoperability since the systems are always accessible and dependable for sharing data.
Response Time and Latency	Reduced response time and latency can enhance interoperability, as delays in communication can hinder interoperability among systems. Increased latency and response times could signal underlying communication issues between systems.
Throughput	Throughput can also impact interoperability, especially when high data transfer rates are required for seamless service integration.
Success Ability and Best Practices	Services with high success rates and follow best practices are more likely to be interoperable with other systems, as they are designed for efficient communication and interaction.
Compliance and Best Practices	Adherence to compliance and best practices could enhance interoperability by ensuring that systems follow industry standards for data exchange.
Documentation	Good documentation could also improve interoperability by providing clear guidelines on how different systems can interact and exchange data effectively.

By analyzing these relationships and exploring potential connections in the dataset, we have utilized compliance, documentation, and availability Features to generate the Interoperability feature values (equation 1)[20].

$$Interoperability = \sqrt{\frac{(Availability + Compliance + Documentation)}{3}} \quad (1)$$

B. Security

Security refers to the protection of web services from unauthorized access or attacks. This feature can measure the security level of web services by, for example, calculating the percentage of authenticated and encrypted requests. To extract a new feature for security from the QWS dataset, we can consider the relationships between the existing features. Table 2 summarizes the potential relationships of the existing features to extract the new feature.

Table 2
The impact of existing features' value on creating Security feature

Features	Effect of feature(s) on creating Security feature
Reliability and Availability	High reliability and availability could be positively correlated with security. Secure systems are often reliable and consistently available to prevent unauthorized access or breaches.
Compliance and Best Practices	Adherence to compliance standards and best practices is crucial for security. Systems that follow security protocols and industry standards are more likely to be secure.
Documentation	Good documentation can positively impact security by providing guidelines for implementing security measures and ensuring that security protocols are correctly followed.
Response Time and Latency	Lower response time and latency could be associated with better security. Quick response times and low latency indicate efficient security measures in place to handle data securely.
Throughput	Higher throughput might indicate better security, as it suggests that systems can handle a larger volume of data securely and efficiently.
Success ability	A high success rate in data exchange could be related to security, as successful data transactions often imply secure communication between systems.

By analyzing these relationships and exploring potential connections in the dataset, in this paper, we have used the two attribute values of compliance and reliability to generate security Feature values across web services in the QWS dataset. This new feature helps evaluate and understand systems' security aspects(equation 2) [21].

$$Security = 1 - (1 - compliance) \times (1 - Reliability) \quad (2)$$

C. Scalability

Scalability in the context of web services refers to the ability of web services to handle increasing numbers of requests. The relationships between the existing features can be considered to extract a new feature for "scalability" from the QWS dataset. Table 3 summarizes how existing features can potentially be related to deriving the new scalability feature.

Table 3
The impact of existing features' value on creating Scalability feature

Features	Effect of feature(s) on creating Scalability feature
Throughput	Higher throughput is often associated with better scalability. Systems with higher throughput can handle more data or requests, indicating scalability.
Response Time	Lower response time can be indicative of good scalability. Systems that can maintain low response times even under increasing workloads are likely to be more scalable.
Availability	High availability is crucial for scalability. Highly available systems can continue to function smoothly as the workload increases, showing scalability.
Reliability	Systems with high reliability are often more scalable, as they can handle increased demands without compromising performance or stability.
Latency	Lower latency can be linked to better scalability. Systems with low latency can efficiently process requests even as the workload grows, showcasing scalability.
Documentation	Good documentation can also play a role in scalability by providing guidelines for scaling the system effectively as demands increase.

By analyzing the above relationships in the dataset, we have used the values of the two throughput and latency characteristics to create scalability characteristics. This new feature helps evaluate and understand services' scalability aspects (equation 3) [22].

$$Scalability = \frac{Throughput}{1 + \sqrt{Latency}} \quad (3)$$

D. Robustness

Robustness refers to the ability of web services to effectively handle errors or failures and unexpected situations. This feature can be used to measure the level of robustness of web services. For instance, one can calculate the percentage of requests successfully processed despite errors or crashes. To derive a new feature for "robustness" from the QWS dataset, one can explore the relationships between the existing features. Table 4 suggests how to extract the Robustness feature from the features present in the QWS dataset.

Table 4
The impact of existing features' value on creating Robustness feature

Features	Effect of feature(s) on creating Robustness feature
Reliability and Availability	High reliability and availability are often indicative of robust systems. Robust systems can maintain reliability and availability even when faced with unexpected challenges.
Compliance and Best Practices	Adherence to compliance standards and best practices can contribute to system robustness. Systems that follow industry standards and best practices are more likely to be robust in handling various scenarios.
Response Time and Latency	Systems with low response time and latency may be considered more robust. Quick response times and low latency can help a system recover quickly from errors or unexpected events.
Documentation	Good documentation can also enhance system robustness by effectively providing guidelines for handling errors, exceptions, and unexpected situations.
Success ability	A high success rate in data exchange may indicate robustness. Systems that can maintain a high success rate even in challenging conditions will likely be more robust.

By examining the above relationships and exploring potential connections in the dataset, we created new Robustness feature values from the combination of reliability and response time feature values, which will help to evaluate and understand Robustness (equation 4) [20].

$$Robustness = \frac{Reliability}{1 + \sqrt{Response\ Time}} \quad (4)$$

Using formulas 1 to 4, four new features have been added to the QWS dataset, and the labeling of services and, of course, the classification of web services has been done using the implementation of the SSL-WSC algorithm on the new EQWS dataset.

4. Experimentations Results

In this section, we examined the impact of newly added features to the QWS dataset on enhancing the efficiency of the SSL-WSS algorithm, which we recently introduced[4]. We have demonstrated a clear improvement in quality-based web service classification. All algorithms have been implemented using Python (version 3.11.7) and on a personal computer with an Intel Core i7-3720QM processor and 32 GB of RAM. The following subsections detail the basic classification algorithms, evaluation criteria, introduction of the SSL-WSC algorithm and its parameter settings, and the results of the implementations.

4.1. Basic classification algorithms

In the implementation of the proposed method, the experiments have been 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 basic classification algorithms. These classifiers have been used to run the SSL-WSC semi-supervised algorithm on the new EQWS dataset and compare it with that algorithm running on the original QWS dataset.

Table 5 shows the modified parameter values used for the different classifiers in the experiments. This table provides an overview of the parameter values used for each classifier, including learning rate, maximum depth, number of estimators, etc. Default values are provided for the parameters not mentioned in the table [4].

Table 5
Modified parameters in base classifier settings

Underlying Classifier	Parameter Settings
Decision Tree	max_depth=3
SVM	probability = True
Logistic egression	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'

4.2. Evaluation Measures

After generating the new EQWS dataset and employing the Base classifier algorithms discussed earlier for model training and testing, we assessed the performance of the proposed method in precision, accuracy, and F1-Score. This assessment enables us to analyze the influence of extra features in improving the classification of web services.

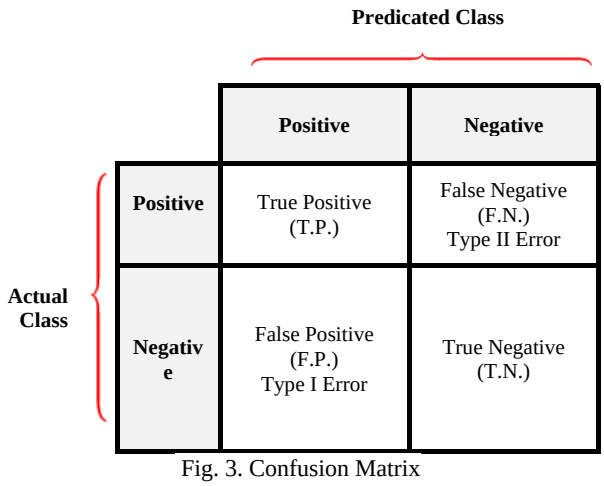


Fig. 3. Confusion Matrix

In the field of classification, the main objective is to achieve the highest possible accuracy and correctly identify categories. In artificial intelligence, the confusion matrix is a matrix that shows the performance of algorithms, allowing for a more comprehensive evaluation of the model's performance (Fig. 3) [23]. Each column of the matrix represents the predicted class for each data (web service), while each row contains the actual class of each data [24]. The proposed method has been evaluated based on the criteria of accuracy, precision, and F1 score and compared with the results obtained from implementing the SSL-WSC algorithm with the original dataset.

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

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

According to the formula, the accuracy of a model can be calculated by summing the true positive and true negative samples and dividing it by the sum of all entries of the Confusion Matrix. True positives and true negatives refer to samples that are correctly classified by the model and are in the main diameter of the Confusion Matrix.

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

F1-Score is a statistical measure used to evaluate performance, calculated as the harmonic mean between recall and precision with equal weight. Usually, in machine learning, the F1-Score index is widely used to evaluate the accuracy of classification models [25].

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

A Type I error is a false positive where the model detects the presence of a condition when it does not exist, and a Type II error is a false negative where the model fails to determine the presence of a condition when it does exist. Both errors can have serious consequences depending on the situation[26].

4.3. Baseline SSL-WSC algorithm

In this article, we have explored enhancing the performance of our previous algorithm, SSL-WSC, by expanding the features of the QWS dataset. As a result, we have adopted the default parameters for implementing the proposed approach, maintaining consistency with the settings for implementing the

SSL-WSC semi-supervised algorithm on the original dataset [4].

Different scenarios are considered in the implementation of the proposed method. The test set size within the labeled data is fixed at 20% and 30% in different implementations, representing common values in many machine-learning approaches. The training steps are repeated 10, 20, 30, and 40 times, with dynamically updated threshold values of 60, 70, 80, and 90 in each iteration. It is worth noting that the results presented are derived from an average of 10 runs of the SSL-WSC algorithm using the proposed methodology outlined in this paper, each run utilizing distinct partitions of training and testing data.

When introducing the SSL-WSC algorithm, a two-step approach was employed to select a subset of the unlabeled data to incorporate into the labeled set, with one step involving the utilization of data distance. In the aforementioned paper, we optionally used Mahalanobis, Manhattan, and Minkowski distances to compute the distance among the known distance functions. The results of implementing the semi-supervised SSL-WSC algorithm show that the Mahalanobis method emerged as the most effective approach for distance calculation. These findings were consistent when the test section size was 20% of the labeled data. Therefore, the outcomes of the proposed method are solely reported for these specific scenarios in this paper.

Typically, the Mahalanobis distance computation is utilized when features are interdependent within a dataset. Within the QWS dataset, interrelations are observed among different features, such as response time with delay and throughput, as well as accessibility with other attributes. Therefore, using the Mahalanobis technique for distance computation is suitable for this dataset.

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

$$dist_{\boldsymbol{Mahalanobis}}(A, B) = \sqrt{\sum_{i=1}^{d} \frac{(a_i - b_i)^2}{{v_i}^2}} \qquad (8)$$

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

4.4. The Effect of new features on the Performance of the SSL-WSC algorithm

In this paper, the three parameters, average (Avg), maximum (Max), and standard deviation (Std), are used to evaluate F1-Score, Accuracy, and Precision criteria. Table 6 compares the implementation results of the SSL-WSC semi-supervised algorithm with the original QWS dataset and the new EQWS dataset for different classifiers to classify quality-based web services.

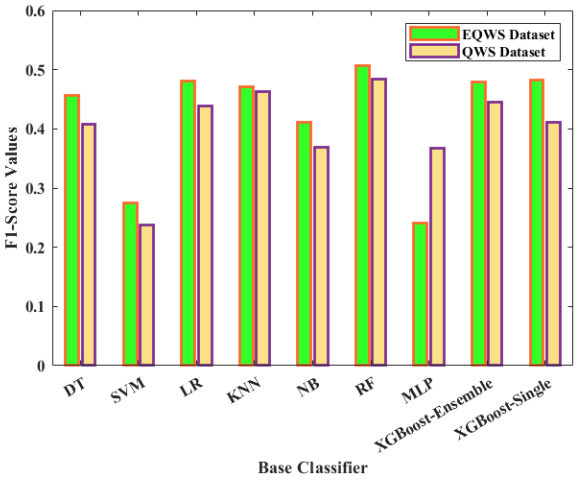
The results in Table 6 demonstrate that, aside from the basic MLP algorithm, the implementation of the proposed method yields notable improvements across all three criteria for most classifiers. For the F1-Score and accuracy metrics, the XGboost (single) classifier shows a maximum improvement of 17.40% and 17.61%, respectively, and for the precision metric, the SVM classifier shows a maximum improvement of 25.46%. In general, the proposed method with the new dataset, compared to the original dataset, has improved the performance of the SSL-WSC algorithm in most of the classifiers and evaluation criteria, and the most significant improvement has been observed in the XGboost classifier. On average, the proposed method increases the performance by 5.05%, 5.69%, and 6.92% in F1-Score, accuracy, and precision, respectively. Therefore, using the EQWS dataset with more features in the proposed method to create classification models can perform better than the previous method with the original dataset.

Also, the small standard deviation values in the proposed dataset for some base classifier algorithms compared to the original method show that implementing the SSL-WSC algorithm using the proposed method works optimally like the original method. As a result, the algorithm is stable.

Table 6
Average (Avg), maximum (Max), and standard deviation (Std) of F1-score, Accuracy, and Precision criteria obtained from the implementation of the SSL-WSC algorithm with the proposed EQWS dataset and the original QWS dataset

			Decision Tree	SVM	Logistic Regression	k-Nearest Neighbors	Naive Bayes	Random Forest	Multilayer Perceptron	XGboost (Ensemble)	XGboost (Single)
F1-Score	**EQWS**	**Avg**	45.67%	27.47%	48.09%	47.11%	41.12%	50.67%	24.06%	47.85%	48.22%
		Max	48.67%	38.36%	55.07%	56.88%	48.85%	62.25%	28.86%	56.03%	55.99%
		Std	0.0172	0.0616	0.0425	0.0545	0.0477	0.0717	0.0331	0.0594	0.0648
	QWS	**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
	The amount of improvement of SSL-WSC with the EQWS dataset		11.83%	15.58%	9.53%	1.92%	11.43%	4.67%	-34.41%	7.51%	17.40%
Accuracy	**EQWS**	**Avg**	47.95%	37.81%	48.63%	47.67%	42.19%	50.96%	34.11%	47.95%	48.49%
		Max	52.05%	46.58%	56.16%	57.53%	49.32%	63.01%	36.99%	56.16%	56.16%
		Std	0.0212	0.0438	0.0421	0.0561	0.0382	0.0730	0.0290	0.0558	0.0628
	QWS	**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
	The amount of improvement of SSL-WSC with the EQWS dataset		10.41%	3.76%	9.23%	2.35%	9.22%	4.79%	-13.54%	7.36%	17.61%
Precision	**EQWS**	**Avg**	49.72%	31.08%	49.67%	49.66%	46.37%	51.40%	29.40%	48.63%	49.24%
		Max	56.75%	59.81%	56.23%	59.57%	55.15%	65.49%	47.41%	56.86%	56.16%
		Std	0.0425	0.1148	0.0441	0.0544	0.0636	0.0784	0.1053	0.0625	0.0600
	QWS	**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
	The amount of improvement of SSL-WSC with the EQWS dataset		12.25%	25.46%	11.12%	2.14%	5.00%	3.50%	-20.64%	6.06%	17.41%

Fig. 4 represents the average results obtained for f1-score, accuracy, and precision criteria for better comparison. As shown in the figures, adding new non-functional qualitative features to the QWS dataset to label the unlabeled data and thus more accurately classify the data in the proposed method outperforms the original dataset using the SSL-WSC algorithm.



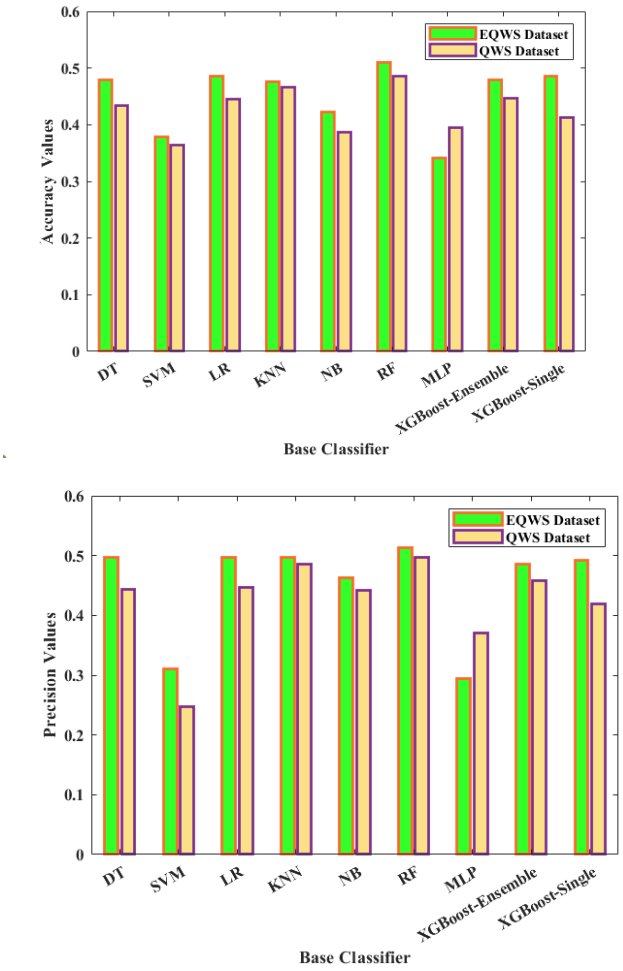


Fig. 4. Comparison of the implementation of the SSL-WSC algorithm using the original QWS dataset and the EQWS dataset in terms of F1-Score, accuracy, and precision criteria

4.5. Discussion

The presence of appropriate features in the datasets about quality-based web services can enhance service classification. However, gathering data about these features can be challenging. Calculating values for new features by analyzing their relationships with existing features using feature engineering can help classify web services with similar functionality and lead to favorable results.

5. Conclusions and Future Works

The Internet provides a platform for sharing services, and web service brokers help users choose the right service from a wide range of similar services based on ratings. Service quality is important in evaluating the service needs of the user. However, collecting information about the quality characteristics of services is challenging and time-consuming. Consequently, service providers resort to data mining and machine learning techniques to ensure that users receive the best possible service and use service classification to identify the most appropriate service. However, the small number of features in the datasets has made us use the feature engineering method in this paper to create new features from the features in the datasets. New non-functional features such as interoperability, security, scalability, and robustness are crucial for applications related to national security and financial transactions. The results of the experiments show that the process of upgrading the famous QWS dataset significantly improves the accuracy of web service classification compared to the original dataset under the implementation of the SSL-WSC semi-supervised algorithm. This is evidenced by the 5.05% increase in F1-Score, 5.69% increase in accuracy, and 6.92% increase in precision evaluation criteria. The enriched EQWS dataset provides a more comprehensive representation of the web service features and thus increases the efficiency of the classification models. This approach has great potential for advanced web service classification and implications for various service-oriented computing applications.

In future research, it would be beneficial to investigate more advanced feature engineering techniques and consider integrating domain-specific knowledge to enhance the dataset. Additionally, exploring ensemble methods and deep learning architectures to classify web services using augmented datasets could be an intriguing approach to consider.

Conflict of Interest

The authors whose names are listed immediately below certify that they have no affiliations with or involvement in any organization or entity with any financial interest or non-financial interest in the subject matter or materials discussed in this manuscript.

Mehdi Nozad Bonab, Jafar Tanha & Mohammad Masdari

Supplementary description

We are continuing our research from a previous paper where we introduced the SSL-WSC semi-supervised algorithm. In this new paper, we focused on boosting the performance of existing datasets through feature engineering. You can find additional

details from the previous article in Section 4.3. For more information, please see the original article at the following address: https://doi.org/10.1109/ACCESS.2024.3385341

References

1. Amin, M.M., et al., *Interoperability framework for integrated e-health services.* Bulletin of Electrical Engineering and Informatics, 2020. **9**(1): p. 354-361.
2. Masdari, M., M. Nozad Bonab, and S. Ozdemir, *QoS-driven metaheuristic service composition schemes: a comprehensive overview.* Artificial Intelligence Review, 2021. **54**: p. 3749-3816.
3. Ye, H., et al., *Web services classification based on wide & Bi-LSTM model.* IEEE Access, 2019. **7**: p. 43697-43706.
4. Bonab, M.N., J. Tanha, and M. Masdari, *A Semi-supervised Learning Approach to Quality-based Web Service Classification.* IEEE Access, 2024.
5. Al-Shargabi, B., S. Al-Jawarneh, and S. Hayajneh, *A cloudlet based security and trust model for e-government web services.* Journal of Theoretical and Applied Information Technology, 2020. **98**(1): p. 27-37.
6. Moritz, G., F. Golatowski, and D. Timmermann. *A lightweight SOAP over CoAP transport binding for resource constraint networks*. in *2011 IEEE eighth international conference on mobile ad-hoc and sensor systems*. 2011. IEEE.
7. Das, M.S., A. Govardhan, and D.V. Lakshmi, *Classification of web services using data mining algorithms and improved learning model.* TELKOMNIKA (Telecommunication Computing Electronics and Control), 2019. **17**(6): p. 3191-3202.
8. Al-Masri, E. and Q.H. Mahmoud. *Qos-based discovery and ranking of web services*. in *2007 16th international conference on computer communications and networks*. 2007. IEEE.
9. Brunton, S.L., B.R. Noack, and P. Koumoutsakos, *Machine learning for fluid mechanics.* Annual review of fluid mechanics, 2020. **52**: p. 477-508.
10. Kaur, M.J., V.P. Mishra, and P. Maheshwari, *The convergence of digital twin, IoT, and machine learning: transforming data into action.* Digital twin technologies and smart cities, 2020: p. 3-17.
11. Hasnain, M., et al., *Machine learning methods for trust-based selection of web services.* KSII Transactions on Internet and Information Systems (TIIS), 2022. **16**(1): p. 38-59.
12. Tanha, J., M. Van Someren, and H. Afsarmanesh, *Semi-supervised self-training for decision tree classifiers.* International Journal of Machine Learning and Cybernetics, 2017. **8**: p. 355-370.
13. Nargesian, F., et al. *Learning Feature Engineering for Classification*. in *Ijcai*. 2017.
14. Guyon, I. and A. Elisseeff, *An introduction to feature extraction*, in *Feature extraction: foundations and applications*. 2006, Springer. p. 1-25.
15. Kumar, V. and S. Minz, *Feature selection.* SmartCR, 2014. **4**(3): p. 211-229.
16. Rodriguez-Mier, P., et al., *An integrated semantic web service discovery and composition framework.* IEEE transactions on services computing, 2015. **9**(4): p. 537-550.
17. Kanter, J.M. and K. Veeramachaneni. *Deep feature synthesis: Towards automating data science endeavors*. in *2015 IEEE international conference on data science and advanced analytics (DSAA)*. 2015. IEEE.
18. Srivastava, N., et al., *Dropout: a simple way to prevent neural networks from overfitting.* The journal of machine learning research, 2014. **15**(1): p. 1929-1958.
19. Chia, Z.L., et al., *Machine Learning and feature engineering-based study into sarcasm and irony classification with application to cyberbullying detection.* Information Processing & Management, 2021. **58**(4): p. 102600.
20. Sommerville, I., *Engineering software products*. Vol. 355. 2020: Pearson London.
21. Thomas, D. and A. Hunt, *The Pragmatic Programmer: your journey to mastery*. 2019: Addison-Wesley Professional.
22. Burns, B., *Designing distributed systems: patterns and paradigms for scalable, reliable services*. 2018: " O'Reilly Media, Inc.".
23. Wu, M.-T., *Confusion matrix and minimum cross-entropy metrics based motion recognition system in the classroom.* Scientific Reports, 2022. **12**(1): p. 3095.
24. Hasnain, M., et al., *Evaluating trust prediction and confusion matrix measures for web services ranking.* Ieee Access, 2020. **8**: p. 90847-90861.
25. Grandini, M., E. Bagli, and G. Visani, *Metrics for multi-class classification: an overview*. arXiv preprint arXiv:2008.05756, 2020.
26. Ruuska, S., et al., *Evaluation of the confusion matrix method in the validation of an automated system for measuring feeding behaviour of cattle.* Behavioural processes, 2018. **148**: p. 56-62.