Towards Effective Service Discovery using Feature Selection and Supervised Learning Algorithms

Heyam H. Al-Baity[1], Norah I. AlShowiman[2]
IT department, College of Computer and Information Sciences
King Saud University, Riyadh, Saudi Arabia

***Abstract*—With the rapid development of web service technologies, the number and variety of web services available on the internet are rapidly increasing. Currently, service registries support human classification, which has been observed to have certain limitations, such as poor query results with low precision and recall rates. With the huge amount of available web services, efficient web service discovery has become a challenging issue. Therefore, to support the effective application of web services, automatic web service classification is required. In recent years, many researchers have approached web service classification problems by applying machine learning methods to automatically classify web services. The ultimate goal of our work is to construct a classifier model that can accurately classify previously unseen web services into the proper categories. This paper presents an intensive investigation on the impact of incorporating feature selection methods (filter and wrapper) on the performance of four state-of-the-art machine learning classifiers. The purpose of employing feature selection is to find a subset of features that maximizes classification accuracy and improves the speed of traditional machine learning classifiers. The effectiveness of the proposed classification method has been evaluated through comprehensive experiments on real-world web service datasets. The results demonstrated that our approach outperforms other state-of-the-art methods.**

Keywords*—Web service discovery; Web Service Description Language (WSDL); supervised machine learning; classification; feature selection

I. Introduction

The number of the available services that are published on the internet is increasing rapidly. These services are provided by different domains (e.g., education, finance, and health) and are available anywhere and anytime. Therefore, finding suitable web services for users quickly and efficiently has become a challenging issue and key problem to be solved. Web services are client and server applications that communicate over the World Wide Web [1, 2]. Basically, a web service works as a request-response form, where a client requests a service from a service provider through a request message. Upon receiving a request, a service provider will respond with a response message. The Web Service Description Language (WSDL), which uses an XML format, can be used to describe web services [3]. WSDL documents are stored in a centralized web service repository called Universal Description, Discovery, and Integration (UDDI). UDDI allows service providers to register their services and clients to discover these services.

Classifying web services into different categories according to their functionality is an efficient method of web service classification. This classification process is typically performed manually by domain experts. However, with the rapid growth of web services on the internet, it has become impractical to organize, classify, and manage web services manually as this requires intensive human effort. Additionally, it is an error-prone task due to the large number of categories in web service registries [4, 5]. Therefore, combining machine learning (ML) techniques to classify and manage web services automatically is an important task.

Based on the popularity of web services and the potential benefits that can be obtained from automatic web service classification, research in this field has recently gained significant attention. Several ML methods have been proposed to automatically classify web services [6]. However, it is still an open problem and further improvements can be achieved.

In this paper, we propose an enhanced method for the automatic classification for web services. Our approach is essentially based on the combination of feature selection methods and supervised ML classifiers. Specifically, feature selection methods have been used prior to performing the classification tasks. The main goal of incorporating feature selection is to select only the significant attributes from the dataset for the classifier. This reduces the size of the dataset and significantly improves the efficiency and accuracy of the classifier. To the best of our knowledge, no extensive work has been performed in this area.

The remainder of this paper is organized as follows. Section II describes the key concepts that are required to implement the proposed approach. In Section III, an overview of previous works related to solving the problem of classification of web services is presented. Section IV describes our methodology. Section V provides a detailed description of our approach and experimental results for real-world service description data. Finally, Section VI concludes this paper and discusses potential future work.

II. Background

In this section, some important concepts necessary for the rest of the paper will be presented. Firstly, we will focus on the main employed classifiers. Secondly, the feature selection strategy will be explained.

A. Overview of Employed Classifiers

1) Support Vector Machine (SVM): SVM is one of the most popular ML algorithms for classification and regression analysis. It operates based on the concept of finding a hyperplane that maximizes the margin between two classes [7]. SVM learns from a training dataset, where each data sample is associated with a class label. It is effective for problems with large dimensionality, such as image and text categorization, because each data sample in the training set is represented as a point in an n-dimensional space, where n is the number of features. SVM then maps new data samples to the closest classes [8, 9]. In general, data samples will be represented in different categories when a large gap divides them. This gap is called a hyperplane. Many hyperplanes can separate a group of data, but the most optimal hyperplane is the hyperplane that creates the largest separation or maximum gap space between classes.

2) Decision Tree (DT): The DT is a recursive predictive classifier that predicts outcomes based on input data. It uses a tree structure to visualize a dataset, and represents sequences and consequences [10]. It can be represented using "IF, THEN" rules to be easily understood. The core concept of a DT is to repeatedly split a dataset into smaller datasets according to descriptive features until a sufficiently small dataset is obtained that contains data points with a single label. A DT has three types of nodes. First, the root node is the topmost node in the tree and has two or more branches. It is used to store predictors. Second, internal nodes or non-leaf nodes represent attributes of the root node. Finally, leaf nodes have no further branches and represent the outcomes of all prior decisions. A branch represents the outcome of a test, which is labeled in leaf nodes. The depth of a node is the minimum number of steps required to reach the node when starting from the root node. The path from the root node to a leaf node contains series decisions that can be converted into a decision rule. There are many types of DT algorithms, with the most popular being C4.5.

3) Naive Bayes (NB): NB is a probabilistic prediction classifier. It uses Bayes' theorem of probability to predict the probability of a tuple belonging to a specific class [11]. NB is useful for very large datasets and is easy to implement because it does not require prior knowledge of data and assumes that attributes are independent. NB begins by learning the conditional probabilities of any input attributes representing categorical data and outputs a class label with a corresponding probability score.

4) Neural Network (NN): NN algorithms have become very popular over the past few years. NNs are inspired by the architectural depth of the human brain. They can be used for regression or classification problems. The main purpose of an NN is to convert inputs into significant outputs by allowing the computer to behave like a human brain to solve problems [12]. Generally, NNs are organized in layers. An artificial NN refers to any structure of interconnected neurons. NNs have achieved excellent results for speech recognition, visual object recognition, object detection, natural language processing, etc.

B. Feature Selection Strategy

Feature selection is the process of automatically selecting a subset of the most relevant features for a problem from an original feature set for use in a model. In this manner, irrelevant or redundant features can be removed without losing any important information. The goal of this technique is to increase the ability of a model by minimizing redundancy and maximizing relevant data. Furthermore, it decreases the required storage space and time for processing [13]. The basic steps of feature selection methods are subset generation, subset evaluation, stopping criterion, and result validation. Typically, each feature has a binary weight of one if it is selected and zero if it is not selected.

There are three general classes of feature selection algorithms: filter methods, wrapper methods and embedded methods.

1) Filter methods: Filter methods measure the general characteristics of training data, such as distance, consistency, dependency, information, and correlation, without using any specific classifiers. Filter methods apply a statistical measure to assign a score to each feature [14]. Such methods consist of two steps. In the first step, features are arranged based on two types of schemes: univariate or multivariate schemes. A univariate scheme rates each feature independently, whereas a multivariate scheme ranks features together in one step. Multivariate schemes are well suited to redundant features. In the second step, the features with the highest scores are used in the classification process. Examples of filter methods are relief, Fisher score, and information gain filters.

2) Wrapper methods: Wrapper methods use the predictive accuracy of a classifier itself to determine the quality of selected features. The general steps for wrapper models are to (1) select a subset of features, and (2) use the target classifier to evaluate the selected features [15]. Steps one and two are repeated until an acceptable performance level is reached. Examples of this wrapper method are sequential forward selection, sequential backward selection, and genetic algorithm wrappers.

The main difference between filter and wrapper methods is that filter methods act as a preprocessing step and work independently of the learning algorithm when selecting the features, whereas wrapper methods operate in the context of the learning algorithm. Typically, wrapper methods provide better predictive accuracy than filter methods. However, wrapper methods can be very slow because of the repeated calls to the learning algorithm.

3) Embedded methods: Embedded methods attempt to narrow the gap between filter and wrapper methods by combining the advantages of each type of method. Such methods interact with learning algorithms and perform feature selection during model construction without splitting the data into training and testing sets. Examples of embedded methods are DT, random forest, NB, and SVM methods.

III. Related Work

Classifying web services automatically is considered to be one of the most important issues for web services. Many researchers have focused on this problem in their research. We will highlight some recent studies on web service classification that are based on ML approaches. The reviewed research papers will be categorized according to the type of ML algorithms used (supervised, unsupervised, and hybrid (supervised and unsupervised)).

A. Supervised Approaches

Laachemi et al. [16] proposed an approach that uses an SVM and enhances classification accuracy based on a stochastic local search (SLS). As a preprocessing step, they scaled or resized the values of quality web service (QWS) dataset attributes [17] to use them in the SVM classifier to finding optimal solutions. The accuracy of this SLS-SVM approach was 84.86%, which is better than the accuracy of an SVM alone or other similar classifiers (NB, metaAdaBoostMl, lazyL Wang-Landau, lazyKstar and treesDS).

Liu et al. [18] combined an SVM classifier with latent Dirichlet allocation (LDA)-based topic models to classify large-scale services and scale down the cost of manually labeling services for training a classifier. The LDA algorithm was used to extract high-level topics from services. The SVM was used as the base classifier for this approach because it performs well on text classification. This is important for classifying web services based on their descriptions. Additionally, they introduced a pool-based active learning strategy to decrease the cost of manual labeling of services, which is required for building a training set. The dataset used for testing was the distributed reliability assessment mechanism for web services (WS-DREAM). In the preprocessing stage, they manually labeled services with informative descriptions based on their functionality. Next, they looked for categories with high numbers of services and selected ten categories. Therefore, as the number of the service increases, the LDA-SVM will provide more accurate results than an SVM alone. Experimental results clearly demonstrated the effectiveness of this active learning service classification framework.

Mustafa et al. [19] proposed a novel classification model using a multi-layer perceptron neural network (MLPNN) optimized via Tabu search (TS). The MLPNN is a model inspired by neuroscience that is used to predict events. They used MLPs with back propagation (BPP) and the Levenberg-Marquardt (LM) algorithm to train an MLP classifier and TS to optimize the classifier. They used a QWS dataset similar to the dataset used in [16]. Experimental results demonstrated that the MLP-TS model achieved superior accuracy, precision, recall, and root mean squared error (RMSE) compared to the un-optimized MLP-LM and MLP-BPP models.

Raj et al. [4] proposed a method to improve web service selection using the K-nearest neighbors (KNN) algorithm based on quality of service (QoS) parameters. They implemented this method on a large dataset containing 5,825 web services. The proposed classification approach begins by utilizing the KNN algorithm as a feature selection method to select a smaller, but more discriminative set of features based on QoS parameters. The classification process is then applied using the selected features. The results indicate that this approach can speed up the selection process compared to manually selected results.

Liu et al. [20] used a semantic web service classification method based on NB to enhance web service classification accuracy. They trained a classifier starting from the service interface level and then used an NB model to classify web services. They used the OWLS-TC dataset, which consists of 1,000 web services. To identify optimal classification features, they used the information gain metric, which led to improved service classification efficiency. The proposed model involves three stages: data preparation, classifier training, and application. The final accuracy of the classification was over 90% and the information gain values differed based on differences in the service attributes.

B. Unsupervised Approaches

Zhao et al. [21], proposed a novel clustering method called the multiple attribute object NetClus (MAO-NetClus) for web service classification to improve the accuracy of service recommendation. This method is based on a heterogeneous information network that focuses on geographic information (i.e., the relationships between web services and their locations). This approach was tested on the WS-DREAM web service dataset. They evaluated the performance of the MAO-NetClus algorithm in different scenarios with different data sparseness, cluster quantity, and iteration numbers. Their approach overcame the limitations of typical web service clustering methods, which are generally based on web service description information, and achieved better performance than the original NetClus algorithm. Additionally, it improved the accuracy of service recommendation.

Tian et al. [22] proposed a web service clustering approach based on clustering web services with short textual descriptions. They used a tag-aided dual author topical model (TD-ATM) to enhance short text clustering by using tags to find long texts on Wikipedia. They used the programmable web dataset, which contains 11,339 web services. Feature extraction was incorporated as a final step during preprocessing to produce the inputs of for the TD-ATM. To evaluate the performance of the TD-ATM, they crawled 4,402 web services and 1,065 Wikipedia pages with a focus on the tags of the web services. They evaluated their approach using the common metrics of entropy, purity, and F-measure. The TD-ATM achieved superior results when compared to previous web service clustering approaches, such as the K-means and agglomerative models, and ATM Long (ATM-L) and ATM Short (ATMS).

C. Hybrid Approaches

Rupasingha et al. [23] improved web service clustering by using ontology learning and a SVM. They used a SVM to calculate the semantic similarity in a generated ontology of web services instead of an edge-count-based method. This approach calculates similarity based on the summary of hybrid term similarity (HTS) and summary of context-aware similarity

(CAS) methods. To cluster web services, they used WSDL files. The proposed hybrid approach includes five phases. Phase one is a feature extraction process that describes the characteristics of each web service. Creating an ontology for each extracted feature is performed in phase two. Phase three calculates the web service similarity values of the ontologies using the SVM. The integration of five different features to calculate a final similarity value is performed in phase four. In the final phase, an agglomerative clustering algorithm (i.e., HTS) is used to cluster the web services. The results indicate decreased purity and increased entropy compared to other approaches with an increasing number of web services. The efficiency and accuracy of this hybrid approach combining HTS and CAS are better than those of each method individually.

Helmy et al. [24] proposed a novel approach that enhances web service clustering by using supervised machine learning techniques. They used DT, NB, and deep learning (DL) classification methods. The WSDL service retrieval test collection dataset was used in their study. This dataset contains 1,088 WSDL services classified into nine domains. The process for web service clustering uses different steps based on the RapidMiner software package. In the first step, which focuses on the preprocessing of data, tokenization based on non-letters and English stop word removal, are used to filter the WSDL files. Feature extraction is the second step of the classification process. In this step, information is extracted from each service's WSDL file. The third step is trimming the extracted features and building a feature-vector-space model, which is applied in the clustering step to produce outputs. To evaluate the effectiveness of this algorithm, the authors computed its precision, recall, accuracy, and F1-score. For the sake of comparison, they tested three techniques, namely the DT, NB, and DL methods, to determine which techniques provide the best results in terms of accuracy and efficiency. The DL technique resulted in the highest accuracy compared to the other approaches, but required more processing time compared to the NB and DT methods.

D. *Discussion of Related Work*

The aim of the review of literature is to investigate and classify recent research on web service classification using ML approaches. Table I contains a summary of the main features of previous research papers, including information regarding the publication year, where or not feature extraction or selection was used, types of ML approaches, applied algorithms, datasets, evaluation metrics, and results of experiments. During our literature review, the following trends have been noticed:

- The field of web service classification is becoming increasingly important based on the increasing number of web services available on the internet.
- Recently, researchers have largely focused on ML algorithms, which can provide efficient web service discovery and overcome manual classification problems.
- The average number of web services in the test datasets was 3,738.
- Feature extraction mechanisms are used to optimize data based on both supervised and unsupervised approaches.
- The most popular classifiers in the literature are the SVM, DT, and NB model.
- The most commonly used evaluation metrics are accuracy, entropy, purity, precision, recall, and F-measure.

Based on previous work on supervised approaches, the algorithms that achieved the best accuracy for classifying the web services can be summarized as follows:

- An NB model achieved 90% accuracy in [20].
- Approaches that combined SVM classifier with LDA [3] or SLS [16] achieved better accuracy compared to SVM classifiers alone.
- Deploying metaheuristic optimization techniques (e.g., TS) with MLPNN classifier enhanced the accuracy of model results [19].

Therefore, based on the above research efforts, the aim of this work is to improve the classification accuracy for web services using supervised ML algorithms combined with feature selection methods. Such combination has the potential to achieve superior accuracy and has not yet been investigated in the literature. In the proposed model, two feature selection methods (filter and wrapper) are combined with four ML classifiers that achieved the highest accuracy for classifying web services in literature. The four classifiers are: SVM, NB model, DT (C4.5), and NN. Comprehensive experiments were conducted to test the performance of the classifiers with and without feature selection methods in order to show the effect of incorporating feature selection methods to the classification process and to find the best approach for web service classification.

TABLE I. COMPARISON of LITERATURE REVIEW PAPERS

PaperRef. No.	Year	Feature Select./ Extract.	Machine Learning		Evaluation		
			Approach	*Algorithms*	*Dataset*	*Metrics*	*Results*
[23]	2015	Extraction	Hybrid	Ontology learning and SVM	Not specified	Purity, entropy, precision, recall, and F-measure	Purity decreased and entropy increased with an increasing number of web services. Average precision: 24.59%,4.69%, 9.16% Average recall: 29.04%, 2.04%, and 1.59% Average F-measure: 28.31%, 3.59%, and 5.42%
[19]	2015	------	Supervised	MMLP-TS	364 Web services	Accuracy, precision, recall, RMSE	Accuracy: 97%
[4]	2015	KNN for Feature Selection	Supervised	KNN	5,825 Web services	Accuracy	The approach speed-up the selection process compared to the manually selected results.
[16]	2016	------	Supervised	SVM and SLS	364 Web services	Accuracy	Accuracy: 84.86%
[3]	2016	------	Supervised	SVM classifier with LDA	3,738 Web services	Accuracy	LDA-SVM active learning yields more accurate results than SVM
[20]	2016	------	Supervised	NB	1,000 Web services	Accuracy	Accuracy: >90%
[21]	2016	------	Unsupervised	NNovel web service clustering method: MAO-NetClus.	3,738 Web services	Accuracy	MAO-NetClus yields better performance than original NetClus
[22]	2016	Extraction	Unsupervised	Tags	11,339 Web service	Entropy, Purity, F-measure	Entropy: 19.6% , purity: 27.9% , and F-measure: 26%,
[24]	2017	Extraction	Hybrid	DT, NB, and DL	1,088 WSDL services	Precision (p), recall (R), accuracy (A), F1-score (F)	DT: (p): 86.76, (R): 86.78, (A): 85.63, (F): 86.8 NB: (p): 90.5, (R): 90.4, (A): 90.34, (F): 90.4 DL: (p): 91.24, (R): 93.79, (A): 90.80, (F): 91.8

IV. METHODOLOGY

In this study, Anaconda Python is used as our experimental platform. To implement the classification algorithms, the Spyder (3.2.3) compiler is used, which is an advanced interactive development environment for Python language. Spyder has built-in integration with a number of popular scientific packages that are used for different programming purposes, such as Matplotlib, NumPy, and scikit-learn.

The implementation process begins by importing the required packages. First, the NumPy library is imported, which is used to initialize the classification models. Second, the scikit-learn library is imported, which is the most widely used library for implementing ML algorithms.

For SVM classifier, the sklearn.svm.SVC class of scikit-learn is imported. Multiclass support was handled based on a one-versus-one scheme. The kernel parameter was set to a nonlinear kernel radial basis function. The remaining parameters were set empirically as follows: the regularization parameter C was set to 1 and the maximum number of iterations was set to 1000. The function gridSearchCV() is used to optimize the parameters.

For DT classifier, the DecisionTreeClassifier class has been imported to perform multi-class classification on the used dataset with a fully grown tree.

The DT employs a greedy algorithm that divides inputs via recursive binary splitting until reaching the leaf nodes. The maximum depth of the tree was set to 3 and the minimum number of samples required to create a leaf node was set to 5 for the initial values.

For NN classifier, MLP is used by importing the MLPClassifier class from the sklearn.neural_network library (sklearn.neural_network.MLPClassifier). This class trains a model iteratively to prevent overfitting. The number of hidden layers was set to 10 and each layer contains 8 neurons. The maximum number of iterations was set to 200. For the NB classifier, the built-in NB algorithm (MultinomialNB) is used, which is suitable for classifying discrete features.

Following the split percentage validation technique, the used datasets have been divided into training set (60%) for constructing the classification models and the rest of the percentage data is used to test the developed models.

V. Experiments and Results

A. Experiment Design

Each classification process for the proposed models consists of four general steps: processing of dataset, applying feature selection to the dataset, classifying web services based on the subset of features obtained from feature selection methods, and finally selecting the appropriate web service from the classification results.

B. Datasets

In our experiments, a real-world web service dataset called QWS dataset [17] is used as our test collection for QoS prediction. This dataset contains many different web services and each web service is defined by the nine quality attributes presented in Table II. QWS attributes indicate the performance of the web service and determine which services satisfy a given set of user requirements. Web services in the QWS dataset are pre-classified into four categories: 1) platinum (high quality), 2) gold, 3) silver, and 4) bronze (low quality). The prediction of QoS is based on the quality rating provided by the web service relevancy function (WSRF), which ranks web service quality using the nine attributes. Based on its WSRF rank, each web service will have a service classification number in the range of 1–4. This number is useful for classifying web services into service categories. The QWS dataset covers many domains and uses a web service crawler engine to collect services.

The QWS[1] dataset is publicly available and has two free versions. The first version (Version 1.0) consists of 364 web services and was created in 2007. The updated QWS dataset (Version 2.0) has a set of 2,507 web services and QWS measurements taken in 2008 using a web service broker framework. In this study, both versions of the dataset are used. The first version is used for the sake of comparison to previous methods from [16] and [19], and the second version is used to test the performance of the proposed approach on a larger dataset. In our experiments, services were divided into two parts. One part was used as training data, and the other was used as testing data.

C. Evaluation Metrics

To assess the performance of the four proposed classification models, the following commonly used performance metrics were calculated: accuracy, precision, sensitivity, specificity, error rate, and execution time. A brief description of these evaluation techniques is provided below [25].

- Accuracy, sometimes referred to as classification rate, is the total number of correct predictions over the total number of samples in the dataset. The maximum accuracy rate is 100%.

Accuracy=$\frac{\mathrm{TP+TN}}{\mathrm{P+N}}$,

where TP=true positive, TN=true negative, P=positive, and N=negative.

- Error rate is the total number of incorrect predictions over the total number of samples in the dataset. The best error rate is zero and the worst is one.

Error Rate=$\frac{\mathrm{FP+FN}}{\mathrm{P+N}}$

- Sensitivity, sometimes referred to as recall or the true positive rate, is the total number of correct positive predictions over the total number of positive samples. The best sensitivity is one and the worst is zero.

Sensitivity = $\frac{\mathrm{TP}}{\mathrm{P}}$

- Specificity, sometimes referred to as the true negative rate, is the total number of correct negative predictions over the total number of negative samples. The best specificity is one and the worst is zero.

Specificity = $\frac{\mathrm{TN}}{\mathrm{N}}$

- Precision, sometimes referred to as the positive predictive value, is the total number of correct positive predictions over the total number of positive predictions. The best precision is one and the worst is zero.

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

TABLE II. QWS Dataset Quality Attributes

#	Attribute Name	Description	Units
1	Response Time	Time required sending a request and receiving a response.	ms
2	Availability	Number of successful invocations/total invocations.	%
3	Throughput	Total number of invocations for a given time period.	invocations/second
4	Successability	Number of responses/number of request messages.	%
5	Reliability	Ratio of the number of error messages to total messages.	%
6	Compliance	The extent to which a WSDL document follows WSDL specifications.	%
7	Best Practices	The extent to which a web service follows the WS-I basic profile.	%
8	Latency	Time required for the server to process a given request.	ms
9	Documentation	Measure of documentation (i.e., description tags) in WSDL document.	%

[1]http://www.uoguelph.ca/~qmahmoud/qws/#Service_Classification_,

D. Results Analysis and Comparisons

The proposed approach was validated in three phases. First, we compared the enhanced web service classification approach to two previous methods from [16] and [19] that used the QWS dataset (Version 1.0). Second, we tested the enhanced classification approach on the QWS dataset (Version 1.0) with and without using feature selection strategies. Third, we tested the enhanced classification approach on the QWS dataset (Version 2.0) with and without using feature selection strategies. The objective of evaluating the proposed classification models on different sizes of datasets was to study their scalability.

1) QWS dataset (Version 1.0) Experiment 1: For the sake of comparison and to clarify the impact of incorporating feature selection methods to the classification process, the proposed approach is compared to the method from [16], which used an NB classifier and SVM classifier combined with SLS. In addition, the proposed approach is compared to the method from [19], which used an NN classifier combined with TS.

As shown in Fig. 1 and Table III, the SVM classifier in our study (without incorporating feature selection) achieved a higher accuracy value (93%) compared to the accuracy value (84.86%) of the SVM classifier with SLS from [16]. Additionally, the SVM classifier with both feature selection methods (filter and wrapper) achieved better accuracy compared to the SVM classifier with SLS from [16].

The NB classifier in our study (without using feature selection) achieved the same accuracy value as the NB classifier from [16], which was 81%. However, when incorporating the feature selection methods (filter and wrapper) into NB classifier, the accuracy of the classifier slightly decreased compared to the classification accuracy of the NB classifier from [16].

The NN classifier in [19] was implemented using the MLP-TS. The accuracy of the MLP-TS model was 97% and that of our NN classifier (without using feature selection) was 92%. This value decreased to 84% when using feature selection (filter and wrapper), as shown in Table III. This indicates that the NN classifier performs better when using stochastic TS algorithms on small datasets. Fig. 2 presents the accuracy results of these two classifiers.

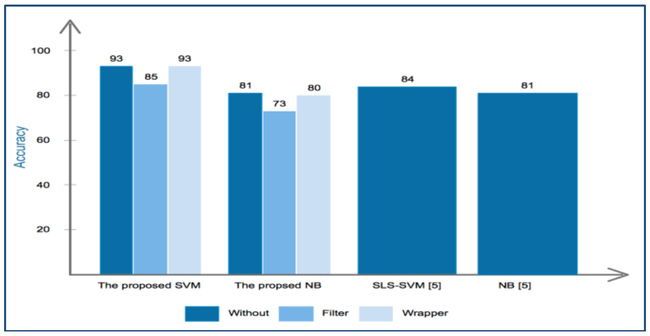

Fig. 1. Accuracy Comparison between our SVM and NB Classifiers (with and without Feature Selection) and the Method from [16].

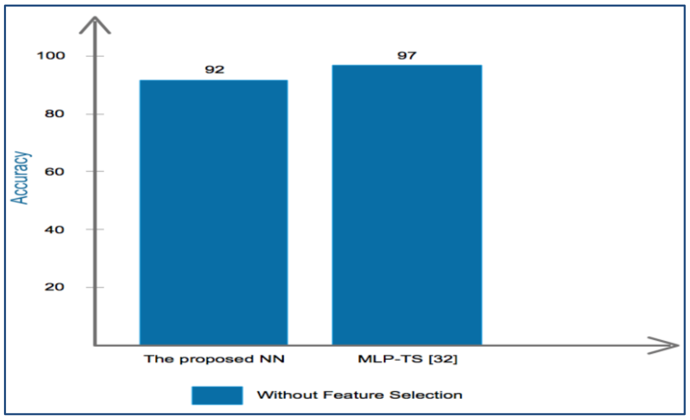

Fig. 2. Accuracy Comparison between our NN Classifier and Method from [19].

2) Experiment 2: The accuracy values of the four employed classifiers using QWS dataset (Version 0.1) are presented in Fig. 3. A detailed comparison between the four classifiers (with and without feature selection strategies) is provided in Table III. It is clear that the SVM classifier achieves the best accuracy results when using the wrapper feature selection method or no feature selection method. The NB classifier with the filter feature selection method achieved the worst accuracy value. NB is considered to be the fastest classifier and required only 0.01 second to complete the classification task with or without using any feature selection methods. The classifier with the lowest error rate (0.04) was the NN classifier with the wrapper feature selection method, meaning that the NN classifier predicted nearly all samples correctly.

Generally, it has been observed that deploying the wrapper method for the feature selection process enhanced the accuracy and error rate more than the filter method. However, the wrapper method has longer execution time.

It is important to note that the impact of using feature selection methods during the classification process did not clearly appear in the results of the above two experiments due to the small size of the QWS dataset used.

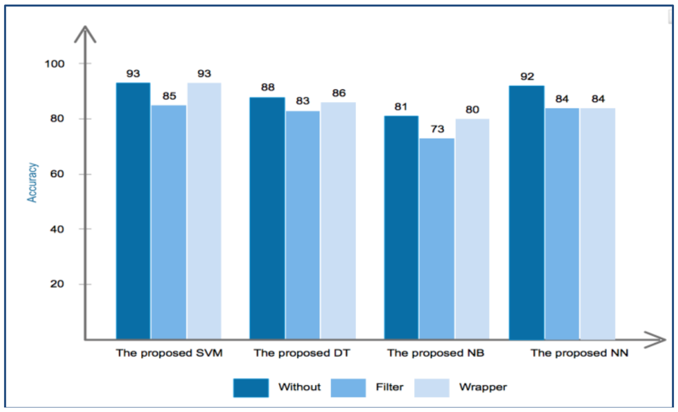

Fig. 3. Accuracy Comparison between the Employed Classifiers (SVM, DT, NB, and NN) with and without Feature Selection Strategies on QWS Dataset (Version 1.0).

TABLE III. EVALUATION RESULTS OF THE EMPLOYED CLASSIFIERS ON QWS DATASET (VERSION 1.0)

Dataset		**Dataset Version 1.0**		
Classifier	***Evaluation Metrics***	***Without Feature Selection***	***Filter Feature Selection***	***Wrapper Feature Selection***
SVM	Execution time in sec	49.41	74.39	115.31
	Accuracy	0.93	0.85	0.93
	Specificity	0.95	0.89	0.95
	Precision	0.88	0.72	0.87
	Error Rate	0.065	0.15	0.06
	Sensitivity	0.87	0.72	0.88
DT	Execution time in sec	0.01	0.01	0.01
	Accuracy	0.88	0.83	0.86
	Specificity	0.91	0.88	0.91
	Precision	0.75	0.65	0.71
	Error Rate	0.13	0.16	0.13
	Sensitivity	0.73	0.72	0.72
NB	Execution time in sec	0.01	0.01	0.01
	Accuracy	0.81	0.73	0.80
	Specificity	0.86	0.82	0.86
	Precision	0.73	0.54	0.66
	Error Rate	0.18	0.26	0.19
	Sensitivity	0.63	0.53	0.65
NN	Execution time in sec	0.12	0.12	0.16
	Accuracy	0.92	0.84	0.84
	Specificity	0.94	0.89	0.89
	Precision	0.83	0.69	0.74
	Error Rate	0.07	0.15	0.15
	Sensitivity	0.86	0.68	0.70

3) QWS dataset (Version 2.0) Experiment 3: In this experiment, performance comparisons were performed between the employed four classifiers (SVM, NB, DT, and NN) on a larger version of QWS dataset (Version 2.0) with and without incorporating feature selection strategies. This new version of QWS was publicly available after conducting Experiments 1 and 2.

It is clear from Table IV that the SVM classifier with the wrapper method achieved the best accuracy value, but had a longest execution time compared to SVM without feature selection and the remaining classifiers. This indicates that the SVM classifier takes full advantage of the wrapper feature selection method to enhance its accuracy, but incurs significant computational costs. However, the error rates of SVM without feature selection and SVM with the wrapper method were the same.

The DT classifier (with and without feature selection methods) has faster execution times compared to SVM, but achieved lower classification accuracy and a higher error rate. The execution time of NB classifier was faster than that of the DT classifier when incorporating the two feature selection methods. However, it achieved worse accuracy and error rate values.

The NN classifier had the fastest execution time among all classifiers. The accuracy values of the NN classifier with the wrapper method and without feature selection were nearly the same. However, the accuracy deteriorated when using the filter method. In contrast, the NN classifier without feature selection showed a higher error rate compared to the wrapper method. This implies that unlike the filter method, the use of the wrapper method with the NN classifier does not adversely affect the accuracy and error rate of the classifier.

Although the SVM classifier achieved superior performance compared to the other classifiers in this experiment, the NN classifier achieved competitive results with a significant decrease in execution time compared to SVM.

As shown in Fig. 4, the classifier with the highest accuracy result was SVM with the wrapper method, followed by NN classifier. The classifiers with the filter method generally achieved worse results compared to their performance with the wrapper method.

In conclusion, after evaluating the classifiers, it has been found that the fastest classifiers when using the QWS dataset (Version 1.0) were the NB and DT classifiers. In contrast, for the QWS dataset (Version 2.0), the NN was the fastest. The classifiers that achieved the best classification accuracy and lowest error rates was the SVM followed by the NN on both versions of the QWS dataset.

The results revealed that the NN classifier achieved slightly better results on the larger QWS dataset (Version 2.0). Therefore, it can conclude that the NN classifier achieves better performance when considering larger datasets. However, incorporating feature selection with the NN classifier did not significantly enhance the accuracy of the model due to the nature of the NN classifier, which provides automatic prediction of hidden features.

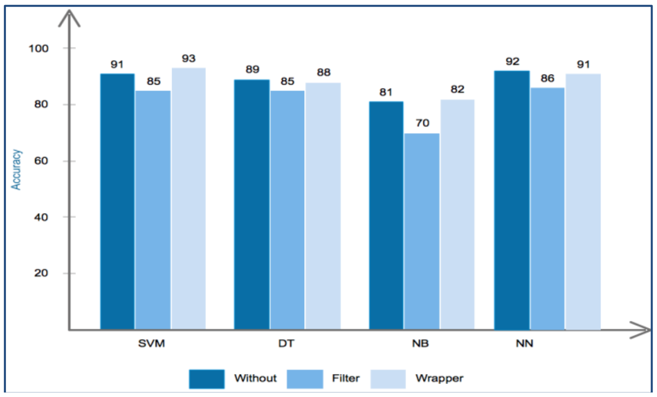

Fig. 4. Accuracy Comparison between the Employed Classifiers (SVM, DT, NB, and NN) with and without Feature Selection Strategies on QWS Dataset (Version 2.0).

TABLE IV. EVALUATION RESULTS FOR THE PROPOSED CLASSIFIERS ON THE QWS DATASET (VERSION 2.0)

Dataset		**Dataset version 2.0**		
Classifier	***Evaluation Metrics***	***Without Feature Selection***	***Filter Feature Selection***	***Wrapper Feature Selection***
SVM	Execution time in sec	200.41	243.39	287.31
	Accuracy	0.91	0.85	0.93
	Specificity	0.94	0.89	0.95
	Precision	0.86	0.72	0.87
	Error Rate	0.064	0.15	0.06
	Sensitivity	0.85	0.72	0.88
DT	Execution time in sec	0.56	0.76	0.88
	Accuracy	0.89	0.85	0.88
	Specificity	0.98	0.90	0.91
	Precision	0.72	0.79	0.71
	Error Rate	0.11	0.19	0.13
	Sensitivity	0.70	0.72	0.72
NB	Execution time in sec	0.56	0.64	0.77
	Accuracy	0.81	0.70	0.82
	Specificity	0.86	0.77	0.88
	Precision	0.73	0.50	0.69
	Error Rate	0.18	0.27	0.22
	Sensitivity	0.63	0.50	0.69
NN	Execution time in sec	0.43	0.43	0.55
	Accuracy	0.92	0.86	0.91
	Specificity	0.94	0.58	0.93
	Precision	0.83	0.73	0.82
	Error Rate	0.07	0.11	0.04
	Sensitivity	0.86	0.68	0.80

Furthermore, when combining the wrapper method with the four classifiers, the overall accuracy of the classifiers enhanced. Generally speaking, the wrapper method yields better accuracy than the filter method because it uses a preselected learning algorithm to evaluate and select an optimal subset of the features, which results in the best classifier performance. However, wrapper methods suffer from being computationally expensive. The SVM classifier achieved the highest accuracy compared to the other classifiers when using the wrapper method on both versions of the QWS dataset. The NN classifier came in second place when using the wrapper method on the QWS dataset (Version 2.0).

Moreover, it has been noticed that deploying the filter method with all four classifiers on both versions of the QWS dataset reduced their performance. The reason of this lower-than-expected performance is the size and dimensionality of the dataset. Filter methods work better on large, high-dimensional datasets

Finally, to enhance classifier performance by using feature selection strategies (filter and wrapper methods), several important factors must be considered, including dataset size, dimensionality of the dataset, the nature of the classifier, and nature of the classification/prediction problem. For example, filter methods are better suited to problems that must be solved online or in batches because they are faster compared to wrapper methods.

VI. CONCLUSIONS AND FUTURE WORK

Effective web service classification is a crucial issue for web services. In this study, we focused on the problem of supervised classification of web services. A novel automated classification method that combines state-of-the-art ML classifiers (SVM, DT, NB, and NN) with feature selection methods (filter and wrapper) is proposed. The purpose of employing feature selection methods in the classification process is to find the effective feature subset prior to training and testing the classifier. We expected the classification accuracy of the ML classifiers to improve when using these methods. Intensive experiments to evaluate the proposed approach were conducted on a publicly available real-world dataset for web services called the QWS dataset and comparisons to related methods were made. Preliminary results revealed a slight improvement in classification accuracy. Additionally, the proposed approach outperformed other algorithms discussed in the literature.

Our future work will proceed in two different directions. First, we will extend our experimental study to a larger dataset with higher dimensionality to investigate the impact of these factors on the performance of the proposed approach. Second, we will study the effects of employing DL algorithms for web service classification problems.

ACKNOWLEDGMENT

This work has been supported by a grant from the Research Center of the Female Scientific and Medical Colleges, Deanship of Scientific Research, King Saud University. The authors would like also to thank the Deanship of Scientific Research and RSSU at King Saud University for their technical support.

REFERENCES

[1] A. S. Mustafa and Y. S. Kumaraswamy, “Data mining algorithms for Web-services classification,” in International Conference on Contemporary Computing and Informatics (IC3I), 2014, pp. 951-956. IEEE.

[2] P. Kaur, “Web Content Classification: A Survey,” International Journal of Computer Trends and Technology (IJCTT), vol. 10, no. 2, 2014.

[3] X. Liu, S. Agarwal, C. Ding, and Q. Yu, “An LDA-SVM Active Learning Framework for Web Service Classification,” IEEE International Conference on Web Services (ICWS), San Francisco, CA, 2016, pp. 49–56.

[4] M. Raj and S. Pragasam, “QoS based classification using K-Nearest Neighbor algorithm for effective Web service selection,” IEEE International Conference on Electrical, Computer and Communication Technologies (ICECCT), Coimbatore, 2015, pp. 1–4.

[5] F. Deng, “Web Service Matching based on Semantic Classification,” M.S. thesis, School of Health and Society, Department of Computer Science, pp. 9–20, 2012.

[6] M. Cracknell, "Machine learning for geological mapping: algorithms and applications," Ph.D. dissertation. University of Tasmania, Australia, 2015.

[7] M. Praveena and V. Jaiganesh, "A literature review on supervised machine learning algorithms and boosting process," International Journal of Computer Applications, vol.169, no. 8, pp. 32-35, 2017.

[8] S. B. Kotsiantis, I. Zaharakis, and P. Pintelas, "Supervised machine learning: A review of classification techniques," Emerging artificial intelligence applications in computer engineering, pp. 3–24, 2007.

[9] J. Yang and X. Zhou, "Semi-automatic Web Service Classification Using Machine Learning," International Journal of u-and e-Service, Science and Technology, vol. 8, no. 4, pp. 339-348, 2015.

[10] J. Fürnkranz, D. Gamberger, and N. Lavrač, "Foundations of rule learning," Springer Science & Business Media, 2012.

[11] T. Raj, T. F. Michael, K. Ravichandran, and K. Rajesh, "Domain specific Web service composition by parameter classification using naïve Bayes algorithm," World Applied Sciences Journal, vol. 29, pp. 99–105, 2014.

[12] J. Schmidhuber, "Deep learning in neural networks: An overview," Neural networks, vol. 61, pp. 85–117, 2015.

[13] J. Gui, Z. Sun, S. Ji, D. Tao, and T. Tan, "Feature selection based on structured sparsity: A comprehensive study," IEEE transactions on neural networks and learning systems, vol. 28, no. 7, pp. 1490-507, 2016.

[14] M. Mwadulo, "A review on feature selection methods for classification tasks," International Journal of Computer Applications Technology and Research, vol. 5, no. 6, pp. 395-402, 2016.

[15] Y. Dhote, S. Agrawal, and A. J. Deen, "A Survey on Feature Selection Techniques for Internet Traffic Classification," in International Conference on Computational Intelligence and Communication Networks (CICN), Jabalpur, 2015, pp. 1375–1380.

[16] A. Laachemi and D. Boughaci, "A stochastic local search combined with support vector machine for Web services classification," in International Conference on Advanced Aspects of Software Engineering (ICAASE), Constantine, 2016, pp. 9–16.

[17] E. Al-Masri and Q. H. Mahmoud, "Discovering the best Web service: A neural network-based solution," in IEEE International Conference on Systems, Man and Cybernetics, 2009, pp. 4250–4255.

[18] X. Liu, S. Agarwal, C. Ding, and Q. Yu, "An LDA-SVM Active Learning Framework for Web Service Classification," in IEEE International Conference on Web Services (ICWS), San Francisco, CA, 2016, pp. 49–56.

[19] A. Syed Mustafa and Y. Kumara Swamy, "Web Service classification using Multi-Layer Perceptron optimized with Tabu search," in IEEE International Advance Computing Conference (IACC), Banglore, 2015, pp. 290–294.

[20] J. Liu, Z. Tian, P. Liu, J. Jiang, and Z. Li, "An Approach of Semantic Web Service Classification Based on Naive Bayes," in IEEE International Conference on Services Computing (SCC), San Francisco, CA, 2016, pp. 356–362.

[21] H. Zhao, J. Wen, J. Zhao, and F. Luo, "A new model-based Web service clustering algorithm," in IEEE Region 10 Conference (TENCON), Singapore, 2016, pp. 3468–3472.

[22] G. Tian, J. Wang, K. He, and C. Sun, "Leveraging Auxiliary Knowledge for Web Service Clustering," Chinese Journal of Electronics, vol. 25, no. 5, pp. 858–865, 2016.

[23] M. Rupasingha, I. Paik, and B. T. G. S. Kumara, "Calculating Web service similarity using ontology learning with machine learning," in IEEE International Conference on Computational Intelligence and Computing Research (ICCIC), Madurai, 2015, pp. 1–8.

[24] A. Helmy and M. Geith, "An Enhanced Approach for Web Services Clustering using Supervised Machine Learning Techniques," International Journal of Scientific & Engineering Research, vol. 8, no. 1, pp. 158–170, 2017.

[25] M. Sokolova and G. Lapalme, "A systematic analysis of performance measures for classification tasks," Information Processing & Management, vol. *45*, no. 4, pp. 427-437, 2009