Predicting Quality of Service (QoS) Parameters using Extreme Learning Machines with Various Kernel Methods

Lov Kumar
NIT Rourkela, India
lovkumar505@gmail.com

Santanu Kumar Rath
NIT Rourkela, India
skrath@nitrkl.ac.in

Ashish Sureka
ABB Corporate Research, India
ashish.sureka@in.abb.com

***Abstract*—Web services which are language and platform independent self-contained web-based distributed application components represented by their interfaces can have different Quality of Service (QoS) characteristics such as performance, reliability and scalability. One of the major objectives of a web service provider and implementer is to be able to estimate and improve the QoS parameters of their web service as its clients application are dependent on the overall quality of the service. We hypothesize that the QoS parameters have a correlation with several source code metrics and hence can be estimated by analyzing the source code. We investigate the predictive power of 37 different software metrics (Chidamber and Kemerer, Harry M. Sneed, Baski & Misra) to estimate 15 QoS attributes. We develop QoS prediction models using Extreme Learning Machines (ELM) with various kernel methods. Since the performance of the classifiers depends on the software metrics that are used to build the prediction model, we also examine two different feature selection techniques i.e., Principal Component Analysis (PCA), and Rough Set Analysis (RSA) for dimensionality reduction and removing irrelevant features. The performance of QoS prediction models are compared using three different types of performance parameters i.e., MAE, MMRE, RMSE. Our experimental results demonstrate that the model developed by extreme learning machine with RBF kernel achieves better results as compared to the other models in terms of the predictive accuracy.**

***Index Terms*—Extreme Learning Machines, Predictive Modeling, Quality of service (QoS) Parameters, Software Metrics, Source Code Analysis, Web Service Definition Language (WSDL)**

I. Research Motivation and Aim

Web services are distributed web application components which can be implemented in different languages, deployed on different client and server platforms, are represented by interfaces and communicate using open protocols [1][2]. Web service implementers and providers need to comply with common web service standards so that they can be language and platform independent and can be discovered and used by other applications [1][2]. Applications and business solutions using web services (which integrate and combine several services) expect high Quality of Service (QoS) such as performance, scalability and reliability as their application is dependent on the service. Measuring quality of service attributes and characteristics of web services and understanding their relationship with source code metrics can help developers control and estimate maintainability by analyzing the source code [3][4][5][6]. The work presented in this paper is motivated by the need to investigate the correlation between QoS attributes such as response time, availability, throughput, reliability, modularity, testability and interoperability and source code metrics such as classic object oriented metrics (Chidamber and Kemerer) as well as other well-known metrics such as Baski & Misra and Harry M. Sneed metrics. Specifically, our research aim is to study the correlation between 15 web service quality attributes and 37 source code metrics and then build machine learning based predictive models for estimating the quality of a given service based on the computed source code metrics. Our aim is to conduct experiments on a real-world dataset and also examine the extent to which feature selection techniques such as Principal Component Analysis (PCA) and Rough Set Analysis (RSA) can be used for dimensionality reduction and filter irrelevant features.

II. Related Work, Research Contributions and Research Framework

Related Work: Coscia et al. investigate the potential of obtaining more maintainable services by exploiting Object-Oriented metrics (OO) values from the source code implementing services [3]. Their approach proposed the use of OO metrics as early indicators to guide software developers towards obtaining more maintainable services [3]. Coscia and Crasso et al. present a statistical correlation analysis demonstrating that classic software engineering metrics (such as WMC, CBO, RFC, CAM, TPC, APC and LCOM) can be used to predict the most relevant quality attributes of WSDL documents [4]. Mateos et al. found that there is a high correlation between well-known object-oriented metrics taken in the code implementing services and the occurrences of anti-patterns in their WSDLs [5]. Kumar et al. use different object-oriented software metrics and Support Vector Machines with different type of kernels for predicting maintainability of services [6]. Their experimental results demonstrate that maintainability of SOC paradigm can be predicted by application of 11 object-oriented metrics [6]. Olatunji et al. develop an extreme learning machine (ELM) maintainability prediction model for objectoriented software systems [7].

Research Contributions: The main research contribution of the study presented in this paper is the application of 37 source code metrics (Chidamber and Kemerer, Harry M. Sneed, Baski & Misra) for predicting 15 Quality of Service (QoS) or maintainability parameters for web services by employing Extreme Learning Machines (ELM) using various kernel methods and two feature selection techniques (Principal Component Analysis and Rough Set Analysis). To the best of our knowledge, the research presented in this paper is the first such in-depth empirical study on publicly available well-known dataset.

Research Framework: Figure 1 displays our research framework and methodology. The framework consists of multiple steps. As shown in Figure 1, we first compute the QoS parameters for the web services in our dataset. We compute 37 source code metrics belonging to 3 different metrics suite. We apply two different feature selection methodology (Rough Set Analysis and Principal Component Analysis) for the purpose of dimensionality reduction and removing irrelevant features. We apply Extreme Learning Machines (ELM) with three different kernel functions (linear, polynomial and RBF). We create 6 sets of metrics suite, 2 feature selection techniques and 3 kernel functions and evaluate the performance of all the combinations resulting in a comprehensive and in-depth experimental evaluation. Finally, we evaluate the performance of various models using wide used estimator evaluation metrics and conduct statistical tests to identify best learning algorithms.

III. Experimental Dataset

We use a subset of QWS Dataset[1] for our experimental analysis. The QWS Dataset provided by Al-Masri et al. includes a sets of 2507 Web services and their 9 QWS parameters (such as response time, availability, throughput, compliance and latency) which are measured using Web service benchmark tools [8][9]. Al-Masri et al. collect the Web services using their Web Service Crawler Engine (WSCE) and majority of the Web services are obtained from public sources. We observe that 524 out of 2507 Web Service have their corresponding WSDL file. Baski et al. present a suite of metrics to evaluate the quality of the XML web service in terms of its maintainability [10]. We apply the Baski and Misra metrics suite tool on the 524 WSDL files and obtained successful parsing for 200 files. We use the metrics proposed by Baski et al. as predictor variables. We could not include 324 WSDL files as part of our experimental dataset as we were unable to parse them for computing Baski and Misra metrics. Hence, we finally use 200 Web services for the experiments presented in this paper. Redistribution of the data on the web is not permitted according to the dataset usage guidelines and hence we provide a list[2] of the 200 Web services used in our study so that our research can be reproduced and replicated for benchmark or comparison. Figure 2 shows a scatter plot for the number of Java files for the 200 WSDL files in our dataset. The X-axis represents the WSDL File ID and the Y-axis represent the number of Java files. Figure 2 shows that there are several web services implemented using more than 100 Java files.

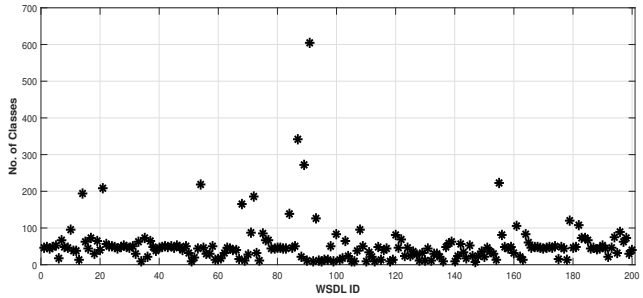

Fig. 2. Scatter Plot for the Number of Java Files for the 200 WSDL Files in Experimental Dataset

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

[2]http://bit.ly/1S8020w

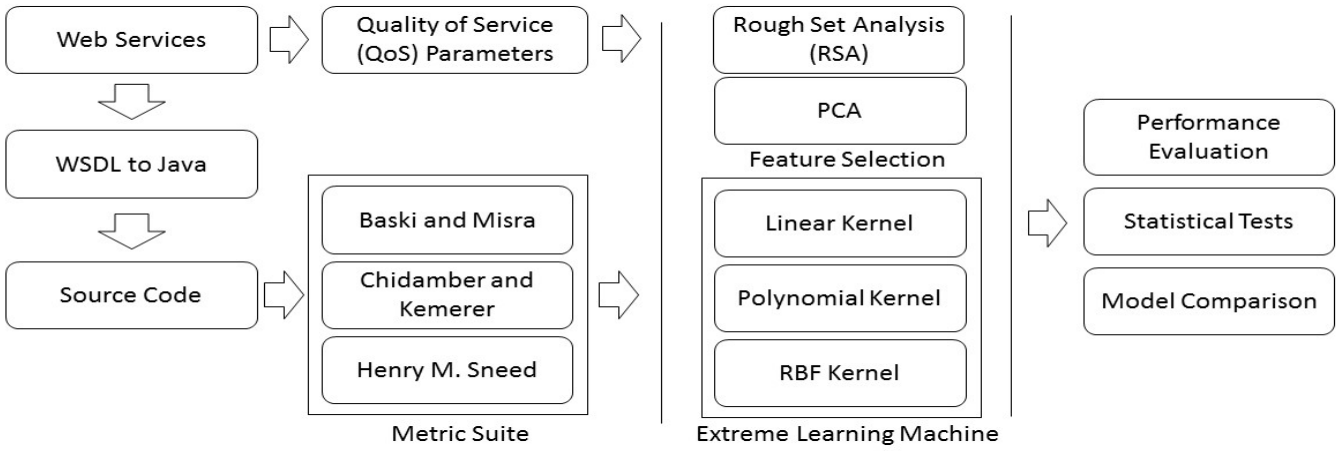

Fig. 1. Research Methodology and Framework

IV. Dependent Variables QoS Parameters

Table I shows the descriptive statistics of 9 QoS parameters provided by the creators of QWS dataset. The owners of QWS dataset provide QoS parameter values for all the 2507 web services. However, Table I displays the descriptive statistics computed by us for the 200 web services used in our experimental dataset. Table I reveals substantial variation or dispersion in the parameter values across 200 web services which shows variability in the quality across services. Sneed et al. describes a tool supported method for measuring web service interfaces [11]. The extended version of their tool can be used to compute maintainability, modularity, reusability, testability, interoperability and conformity of web services. We calculate these values for the 200 web services in our dataset and assign them as dependent variables. Table II displays the descriptive statistics for the QoS parameters calculated using Sneed's Tool. Hence, we have a total of 15 dependent variables.

TABLE I
Descriptive statistics of QoS parameters provided by QWS dataset

Parameter	Min	Max	Mean	Median	Std Dev	Skewness	Kurtosis
Response Time	57.00	1664.62	325.11	252.20	289.33	3.15	13.12
Availability	13.00	100.00	86.65	89.00	12.57	-2.63	12.06
Throughput	0.20	36.90	7.04	4.00	6.94	1.57	5.79
Successability	14.00	100.00	90.19	96.00	13.61	-2.57	10.81
Reliability	33.00	83.00	66.64	73.00	9.61	-0.60	2.96
Compliance	67.00	100.00	92.19	100.00	9.78	-0.90	2.61
Best Practices	57.00	93.00	78.81	82.00	7.70	-0.68	2.67
Latency	0.74	1337.00	42.81	12.20	106.23	9.56	112.68
Documentation	1.00	96.00	29.37	32.00	26.97	1.06	3.31

TABLE II
Descriptive statistics of QoS parameters calculated using Sneeds Tool

Parameter	Min	Max	Mean	Median	Std Dev	Skewness	Kurtosis
Maintainability	0.00	77.67	31.07	28.17	24.29	0.37	2.02
Modularity	0.10	0.81	0.22	0.17	0.13	2.02	7.10
Reusability	0.10	0.90	0.38	0.35	0.17	0.32	2.94
Testability	0.10	0.66	0.19	0.16	0.09	2.58	10.71
Interoperability	0.14	0.90	0.51	0.41	0.23	0.65	2.01
Conformity	0.43	0.98	0.79	0.87	0.15	-0.47	1.57

V. Predictor Variables - Source Code Metrics

Chidamber and Kemerer Metrics: We compute several size and structure software metrics from the bytecode of the compiled Java files in our experimental dataset using CKJM extended[3] [12][13]. CKJM extended is an extended version of tool for calculating Chidamber and Kemerer Java Metrics and many other metrics such as weighted methods per class, coupling between object classes, lines of code, measure of functional abstraction, average method complexity and McCabe's Cyclomatic Complexity. We use the WSDL2Java Axis2 code generator[4] which comes built-in with an Eclipse plug-in to generate Java class files from the 200 WSDL files in our experimental dataset. We then compile the Java files to generate the bytecode for computing the size and structure software metrics using the CKJM extended tool. The minimum number of Java files are 7 and the maximum is 605. The mean, median, standard deviation, skewness and kurtosis is 52.39, 45.50, 59.06, 5.43 and 43.94 respectively. Table III displays the descriptive statistics for 19 size and structure software metrics computed using CKJM Extended Tool for the 200 Web services in our dataset. The mean value of AMC as 61.94 means that the mean of the average method size calculated in terms of the number of Java binary codes in the method for each class is 62. We compute the standard deviation for all the 19 metrics to quantify the amount of dispersion and spread in the values. We observe (refer to Table III) that few metrics such as DIT, NOC, MFA, CAM, IC and CBM have low standard deviation which means that the data points are close to the mean. However, we observe that LCOM, LCO, AMC and CC have relatively high values of standard deviation which means that the data points are dispersed over a wider range of values.

TABLE III
Descriptive statistics of Object-Oriented Metrics

Metrics	Min	Max	Mean	Median	Std Dev	Skewness	Kurtosis
WMC	9.48	13.57	11.01	10.96	0.48	0.81	6.54
DIT	0.87	1.02	0.98	0.98	0.02	-2.07	9.72
NOC	0.00	0.13	0.01	0.01	0.02	2.64	12.55
CBO	4.10	12.55	10.70	11.01	1.33	-1.15	5.18
RFC	12.78	44.55	40.35	41.48	4.13	-3.13	15.15
LCOM	74.03	405.49	120.94	108.67	45.70	2.99	13.96
Ca	0.64	3.92	2.91	2.99	0.62	-0.50	2.72
Ce	3.49	9.50	8.24	8.37	0.90	-1.30	6.01
NPM	4.88	9.27	6.55	6.47	0.48	1.07	7.90
LCOM3	1.18	1.50	1.32	1.31	0.06	0.27	2.75
LCO	76.14	493.64	399.18	411.50	54.03	-2.61	11.71
DAM	0.21	0.45	0.37	0.37	0.04	-0.45	4.53
MOA	0.02	2.28	0.60	0.53	0.28	2.00	11.03
MFA	0.00	0.02	0.00	0.00	0.00	1.92	8.43
CAM	0.39	0.43	0.40	0.40	0.01	0.22	4.54
IC	0.00	0.05	0.01	0.01	0.01	0.79	3.68
CBM	0.00	0.05	0.01	0.01	0.01	0.79	3.68
AMC	7.68	82.86	61.94	64.37	10.75	-1.69	6.97
CC	18.17	71.39	42.77	43.74	9.58	-0.29	3.67

TABLE IV
Descriptive statistics of Harry M. Sneed's Metrics Suite

Metrics	Min	Max	Mean	Median	Std Dev	Skewness	Kurtosis
Data Complexity	0.10	0.81	0.28	0.27	0.17	0.60	2.59
Relation Complexity	0.10	0.90	0.87	0.90	0.07	-7.72	83.58
Format Complexity	0.14	0.72	0.60	0.64	0.09	-1.05	5.24
Structure Complexity	0.15	0.90	0.61	0.63	0.17	-0.13	2.63
Data Flow Complexity	0.10	0.90	0.87	0.90	0.10	-5.64	39.14
Language Complexity	0.16	0.88	0.61	0.56	0.21	0.03	1.78
Object Point	42.00	4581.00	299.32	200.00	483.31	5.67	41.67
Data Point	29.00	3124.00	222.75	152.00	347.48	5.16	34.85
Function Point	6.00	776.00	53.73	32.00	94.21	5.36	35.33
Major Rule Violation	2.00	109.00	26.39	10.00	26.13	0.62	1.97
Medium Rule Violation	2.00	16.00	5.02	5.00	1.94	0.56	6.71
Minor Rule Violation	2.00	586.00	49.51	35.50	63.14	4.26	30.60

Harry M. Sneed Metrics: Sneed's tool implements metrics for quantity, quality and complexity of web service interfaces. The values of all the metrics are statically computed from a service interface in WSDL as the suite of metrics is based on the WSDL schema element occurrences [11]. We compute six interface complexity metrics for all the 200 web services in our dataset. The six interface complexity metrics are computed between a scale of 0.0 to 1.0. A value between 0.0

[3]http://gromit.iiar.pwr.wroc.pl/p_inf/ckjm/

[4]https://axis.apache.org/axis2/java/core/tools/eclipse/wsdl2java-plugin.html

TABLE V
DESCRIPTIVE STATISTICS OF BASKI AND MISRA METRICS SUITE

Metrics	Min	Max	Mean	Median	Std Dev	Skewness	Kurtosis
OPS	0.00	108.00	7.76	5.00	13.74	5.41	35.37
DW	0.00	2052.00	114.63	62.00	216.46	6.56	53.80
MDC	0.00	17.00	4.40	5.00	2.62	1.97	8.87
DMR	0.00	1.00	0.53	0.50	0.22	0.50	3.32
ME	0.00	3.80	1.73	2.12	0.73	-0.43	3.58
MRS	0.00	72.00	3.65	2.60	6.25	7.99	79.12

and 0.4 represents low complexity and a value between 0.4 and 0.6 indicates average complexity. A value of more than 0.6 falls in the range of high complexity wherein any value above 0.8 reveals that there are major issues with the code design [4][11]. Table IV shows the minimum, maximum, mean, median and standard deviation of the size complexity values for all the web services on our dataset. In addition to 6 interface complexity metrics, we measure 6 more metrics using the extended version of the tool provided to us by the author himself: object point, data point, function point, major, medium and minor rule violations. Table IV displays the descriptive statistics for the 12 metrics for all the web services in our dataset.

Baski and Misra Metrics: We compute 6 metrics proposed by Baski and Misra [10]. Their metrics are based on the analysis of the structure of the exchanged messages described in WSDL which becomes the basis for measuring the data complexity. Their metric suitbe is based on WSDL and XSD schema elements occurrences. Table V reveals the descriptive statistics of the 6 metrics: Data Weight of a WSDL (DW), Distinct Message Ratio (DMR), Distinct Message Count (DMC), Message Entropy (ME), Message Repetition Scale (MRS) and Operations Per Service (OPS).

VI. CODE METRICS - CORRELATION ANALYSIS

We compute the association between 37 metrics consisting of dependent and independent variables using the Pearson's correlations coefficient (r). The coefficient of correlation r measures the strength and direction of the linear relationship between two variables. Figure 3 displays our experimental results on correlation analysis between the 37 metrics. In Figure 3, a Black circle represents an r value between 0.7 and 1.0 indicating a strong positive linear relationship. A white circle r rvalue between 0.3 and 0.7 indicate a weak positive linear relationship. A black square r represents a value between -1 and -0.7 indicating a strong negative linear relationship. A white square r represents a value between -0.7 and -0.3 indicating a weak negative linear relationship. A blank circle represents no linear relationships between the two variables. For example, based on Figure 3, we infer that there is a strong positive linear relationship between OPS and four other variables MRS, OP, DP and FP. On the other hand, we observe a weak linear relationship between ILC and IDC as well as ISC and IDC. Figure 3 reveals association between different suite of metrics and not just associations between metrics within the same suite. For example, DMR is part of Baski and Misra metrics suite. DMR has a strong negative correlation with ISC (Structure Complexity), OP (Object Point), DP (Data Point), FP (Function Point) and MERV (Medium Rule Violation) which is part of Harry M. Sneed metrics suite.

VII. FEATURE EXTRACTION AND SELECTION USING PCA AND RSA

We investigate the application of Principal Component Analysis (PCA) and Rough Set Analysis (RSA) as a data preprocessing step for feature extraction and selection [14]. Our

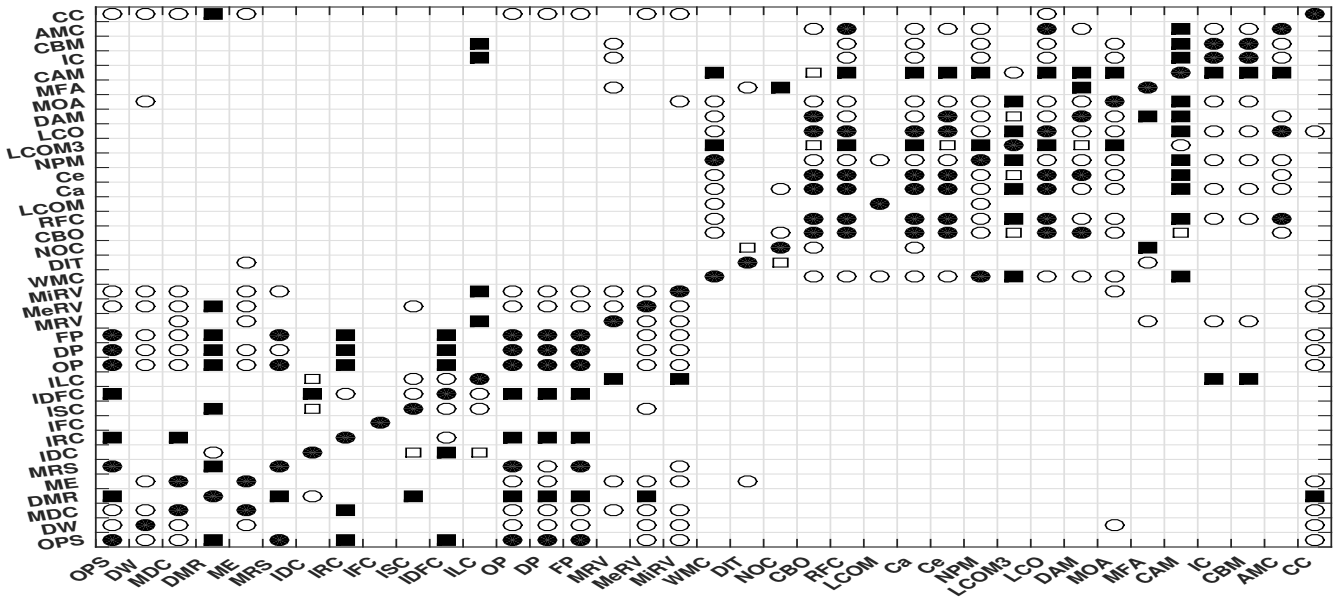

Fig. 3. Pearson's Correlation Coefficient between 37 Metrics

objective behind using PCA and RSA is to identify features which are relevant in-terms of high predictive power and impact on the dependent variable and filter irrelevant features which have little or no impact on the classifier accuracy [14]. We apply PCA and varimax rotation method on all source code metrics. The experimental results of PCA analysis is shown in Table VI. Table VI reveals the relationship between the original source code metrics and the domain metrics. For each PC (Principal Components), we provide the eigenvalue, variance percent, cumulative percent and source code metrics interpretation (refer to Table VI). In PCA the order of the eigenvalues from highest to lowest indicates the principal components in the order of significance. Among all Principal Components, we select only those which have Eigen value greater than 1. Our analysis reveals that 9 PCs have Eigen value greater than 1 (refer to Table VI). Table VI shows the mapping of each component to the most important metric for that component. Table VII shows the optimal subset of features for every dependent variable derived from the original set of 37 source code metrics based features after applying RSA. We apply the RSA procedure 15 times (one for each dependent variable). Table VII reveals that it is possible to reduce the number of features substantially and several features from the original set are found to be uncorrelated.

TABLE VI
FEATURE EXTRACTION USING PRINCIPAL COMPONENT ANALYSIS - DESCRIPTIVE STATISTICS

PC	Eigenvalue	% variance	Cumulative %	Metrics Interpretation
PC1	6.4	17.3	17.3	CBO, RFC, Ca, Ce, LCOM3, LCO, DAM, CAM
PC2	5.8	15.76	33.06	OPS, MRS, IRC, IDFC, OP, DP, FP
PC3	3.67	9.94	43.00	DW, MDC, MeRV, MiRV, CC, ME
PC4	3.39	9.16	52.17	DMR, IDC,ISC, ILC
PC5	3.34	9.03	61.2	IC, CBM, MOA
PC6	2.5	6.77	67.98	IFC, DIT, NOC, MFA
PC7	2.23	6.02	74.00	WMC, NPM
PC8	2.14	5.79	79.79	MRV, AMC
PC9	1.36	3.7	83.5	LCOM

TABLE VII
SOURCE CODE METRICS (FEATURE) SELECTION OUTPUT USING ROUGH SET ANALYSIS (RSA)

QoS	Selected Metrics
Response Time	DMR, SC, LC, WMC, Ca, LCOM3, MFA, CAM, IC, CC
Availability	FC, SC, LC, MeRV, MiRV, WMC, Ca, LCOM3, MFA, CAM, IC, CC
Throughput	ME, FC, SC, LC, MRV, MeRV, MiRV, Ce, MOA, MFA, CAM, CBM, CC
Successability	ME, FC, SC, DFC, LC, MRV, WMC, LCOM3, LCO, DAM, MOA, CAM
Reliability	FC, SC, DFC, LC, WMC, LCOM3, LCO, MOA, MFA, CAM, CBM
Compliance	ME, FC, SC, DFC, LC, MRV, WMC, MiRV, Ca, CC, DAM, MOA, CAM, NPM
Best Practices	ME, FC, SC, DFC, LC, MRV, MiRV, WMC, Ca, NPM, MOA, MFA, CAM, CC
Latency	DMR, ME, DC, FC, DFC, LC, MRV, NOC, NPM, LCO, MOA, CAM, IC
Documentation	ME, FC, SC, DFC, LC, MRV, MeRV, WMC, Ca, NPM, CAM, IC, CC
Maintainability	DP, Ce, LCOM3, MOA, MFA, CAM, CBM
Modularity	DMR, ME, SC, DFC, LC, DP, MRV, MiRV, WMC, Ca, MOA, IC, AMC
Reusability	MDC, DMR, FC, SC, DFC, LC, LCOM, LCOM3
Testability	ME, FC, SC, LC, MiRV, DIT, NOC, CC, RFC
Interoperability	SC, LC, MeRV, MiRV, WMC, DIT, CBO, MFA, CC
Conformity	ME, FC, DFC, LC, MRV, WMC, Ca, CAM

VIII. APPLICATION OF EXTREME LEARNING MACHINES (ELMS)

Huan et al. mention that Extreme Learning Machines (ELMs) have shown to outperform computational intelligence techniques such as Artificial Neural Networks (ANNs) and Support Vector Machines (SVMs) in-terms of learning speed and computational scalability [15]. ELM has demonstrated good potential to resolving regression and classification problems [15] and our objective is to investigate if ELMs can be successfully applied in the domain of web service QoS prediction using source code metrics. Selection of an appropriate kernel function depending on the application domain and dataset is an important and core issue [16].

Ding et al. mention that there is a correlation between the generalization performance and learning performance with the kernel function [16] as in the case of traditional neural networks [16]. Hence, we investigate the performance of the ELM based classifier using three different kernel functions: linear, polynomial and RBF. ELMs can be used with different kernel functions and one can create hybrid kernel functions also. The most basic, simplest and fastest is the linear kernel function which is used as a baseline for comparison with more complicated kernel functions such as polynomial and RBF. Table VIII shows the performance of the ELM based classifier with linear kernel function. Table IX shows the performance of the ELM based predictive model with second degree polynomial kernel. The polynomial kernel is more sophisticated than the linear kernel and uses non-linear equations instead of the linear equations for the purpose of regression and classification and is expected to result in better accuracy in comparison to the classifier with linear kernel. The Radial Basis Function kernel (RBF or Gaussian) is a popular kernel function and widely used in Support Vector Machine (SVM) learning algorithm. We use linear kernel to investigate if the data is linearly separable but also use polynomial and RBF kernel to examine if our data is not linearly separable (computing a non-linear decision boundary). We employ four different performance metrics (MAE, MMRE, RMSE and r-value) to study the accuracy of the classifiers. The Mean Absolute Error (MAE) measures the difference between the predicted or forecasted value and the actual values (average of the absolute errors). Table VIII and IX reveals that the forecast for several predictive model is very accurate as the MAE value is less than 0.05. For example, the MAE value for HMS, AM and PCA metrics for predicting Conformity is 0.03. Table VIII and IX reveals that in general the predictive accuracy for response time, latency, modularity and conformity is better than the predictive accuracy of other QoS parameters.

Kitchenham et al. mention that Mean Magnitude of Relative Error (MMRE) is a widely used assessment criterion for evaluating the predictive accuracy and overall performance of competing software prediction models and particularly the software estimation models [17]. MMRE computes the difference between actual and predicted value relative to the actual value. Table X shows that the MMRE values for ELM with RBF kernel is between 0.30 to 0.35 for response time, availability and successability and indicates good estimation ability of the classifier. Table VIII and Table IX reveals that the MMRE values for conformity QoS parameter are as low as 0.05, 0.06, 0.10 and 0.11. Root Mean Square Error (RMSE) or Root Mean square Deviation root-mean-

TABLE VIII
PERFORMANCE MATRIX FOR ELM WITH LINEAR KERNEL

	Response Time	Availability	Throughput	Successability	Reliability	Compliance	Best Practices	Latency	Documentation	Maintainability	Modularity	Reusability	Testability	Interoperability	Conformity
	MAE														
BMS	0.11	0.19	0.18	0.21	0.25	0.28	0.19	0.11	0.23	0.20	0.11	0.16	0.12	0.16	0.18
HMS	0.10	0.16	0.18	0.19	0.25	0.27	0.19	0.11	0.22	0.15	0.08	0.15	0.07	0.14	0.06
OOM	0.09	0.14	0.18	0.17	0.25	0.25	0.18	0.11	0.22	0.12	0.15	0.15	0.14	0.22	0.18
AM	0.10	0.15	0.18	0.18	0.26	0.27	0.20	0.11	0.21	0.11	0.07	0.13	0.06	0.11	0.06
PCA	0.10	0.16	0.19	0.18	0.26	0.26	0.21	0.12	0.21	0.16	0.09	0.14	0.10	0.14	0.09
RSA	0.11	0.17	0.18	0.20	0.25	0.26	0.19	0.11	0.23	0.27	0.16	0.17	0.15	0.25	0.26
	MMRE														
BMS	0.37	0.35	0.73	0.38	0.66	0.61	0.37	0.55	0.90	0.76	0.41	0.61	0.46	0.35	0.28
HMS	0.36	0.33	0.73	0.36	0.69	0.61	0.39	0.56	0.93	0.60	0.29	0.60	0.26	0.31	0.11
OOM	0.34	0.32	0.74	0.35	0.71	0.61	0.37	0.55	0.89	0.47	0.54	0.57	0.55	0.50	0.33
AM	0.36	0.34	0.71	0.37	0.71	0.64	0.47	0.54	0.90	0.42	0.22	0.46	0.20	0.25	0.10
PCA	0.34	0.34	0.74	0.36	0.69	0.59	0.49	0.60	0.88	0.60	0.30	0.59	0.37	0.33	0.16
RSA	0.36	0.34	0.71	0.37	0.69	0.57	0.36	0.54	0.88	1.09	0.60	0.67	0.54	0.52	0.44
	RMSE														
BMS	0.16	0.26	0.22	0.28	0.29	0.33	0.24	0.15	0.29	0.26	0.17	0.22	0.17	0.22	0.22
HMS	0.15	0.22	0.22	0.24	0.29	0.31	0.23	0.15	0.28	0.21	0.12	0.19	0.11	0.18	0.08
OOM	0.15	0.21	0.22	0.23	0.29	0.30	0.22	0.14	0.28	0.16	0.22	0.19	0.21	0.28	0.23
AM	0.15	0.22	0.22	0.24	0.29	0.32	0.24	0.14	0.27	0.14	0.11	0.16	0.10	0.15	0.08
PCA	0.15	0.22	0.22	0.25	0.29	0.31	0.25	0.16	0.27	0.21	0.13	0.20	0.14	0.19	0.13
RSA	0.17	0.26	0.22	0.29	0.30	0.31	0.23	0.14	0.29	0.34	0.22	0.23	0.21	0.31	0.32
	r-value														
BMS	0.34	0.30	0.30	0.45	0.50	0.49	0.47	0.45	0.41	0.84	0.81	0.50	0.90	0.88	0.79
HMS	0.35	0.33	0.39	0.38	0.21	0.28	0.62	0.21	0.23	0.88	0.90	0.83	0.94	0.88	0.98
OOM	0.66	0.62	0.57	0.27	0.65	0.48	0.64	0.48	0.33	0.94	0.74	0.78	0.51	0.67	0.77
AM	0.32	0.35	0.37	0.26	0.34	0.30	0.34	0.58	0.27	0.94	0.98	0.90	0.93	0.92	0.99
PCA	0.40	0.32	0.37	0.39	0.41	0.40	0.29	0.36	0.37	0.87	0.94	0.79	0.86	0.90	0.99
RSA	0.29	0.77	0.49	0.45	0.37	0.48	0.55	0.31	0.43	0.11	0.36	0.48	0.70	0.77	0.43

TABLE IX
PERFORMANCE MATRIX FOR ELM WITH POLYNOMIAL KERNEL

	Response Time	Availability	Throughput	Successability	Reliability	Compliance	Best Practices	Latency	Documentation	Maintainability	Modularity	Reusability	Testability	Interoperability	Conformity
BMS	0.10	0.14	0.18	0.17	0.24	0.26	0.19	0.11	0.23	0.17	0.11	0.16	0.11	0.15	0.12
HMS	0.11	0.15	0.19	0.17	0.26	0.28	0.19	0.11	0.23	0.12	0.05	0.11	0.04	0.10	0.03
OOM	0.12	0.15	0.21	0.18	0.27	0.28	0.19	0.11	0.23	0.12	0.13	0.14	0.15	0.18	0.12
AM	0.13	0.20	0.20	0.23	0.28	0.33	0.23	0.13	0.28	0.10	0.06	0.12	0.05	0.11	0.03
PCA	0.11	0.15	0.20	0.18	0.24	0.26	0.19	0.12	0.23	0.10	0.06	0.13	0.07	0.11	0.03
RSA	0.10	0.14	0.18	0.16	0.24	0.26	0.18	0.11	0.22	0.25	0.14	0.16	0.14	0.23	0.23
	MMRE														
BMS	0.36	0.32	0.75	0.35	0.68	0.62	0.39	0.58	0.94	0.71	0.41	0.67	0.44	0.36	0.20
HMS	0.38	0.33	0.76	0.35	0.71	0.64	0.40	0.57	0.93	0.52	0.19	0.40	0.18	0.19	0.05
OOM	0.42	0.34	0.81	0.37	0.77	0.64	0.37	0.56	0.94	0.44	0.48	0.49	0.57	0.41	0.22
AM	0.48	0.41	0.82	0.46	0.73	0.70	0.54	0.67	1.18	0.42	0.25	0.41	0.18	0.21	0.05
PCA	0.36	0.35	0.80	0.37	0.69	0.59	0.47	0.60	0.93	0.38	0.24	0.50	0.28	0.23	0.06
RSA	0.36	0.31	0.74	0.33	0.68	0.59	0.36	0.57	0.91	1.04	0.52	0.67	0.53	0.51	0.41
	RMSE														
BMS	0.15	0.20	0.22	0.23	0.28	0.31	0.22	0.15	0.29	0.23	0.16	0.22	0.17	0.20	0.16
HMS	0.16	0.21	0.23	0.23	0.30	0.33	0.23	0.16	0.29	0.18	0.08	0.17	0.07	0.13	0.04
OOM	0.18	0.22	0.26	0.25	0.32	0.34	0.23	0.15	0.30	0.15	0.20	0.19	0.21	0.25	0.18
AM	0.19	0.27	0.25	0.32	0.33	0.41	0.28	0.17	0.36	0.16	0.10	0.16	0.07	0.17	0.05
PCA	0.15	0.22	0.24	0.24	0.28	0.31	0.24	0.16	0.29	0.13	0.11	0.18	0.11	0.16	0.06
RSA	0.16	0.20	0.22	0.22	0.28	0.31	0.22	0.15	0.28	0.31	0.20	0.21	0.20	0.28	0.26
	r-value														
BMS	0.46	0.30	0.39	0.16	0.36	0.16	0.38	0.07	0.09	0.86	0.88	0.45	0.93	0.89	0.89
HMS	0.18	0.31	0.19	0.38	-0.02	0.19	0.24	0.46	0.37	0.94	0.97	0.90	1.00	0.95	1.00
OOM	0.08	0.29	0.38	0.22	0.36	0.16	0.34	0.42	0.46	0.96	0.77	0.72	0.59	0.75	0.92
AM	0.12	0.31	0.40	-0.01	0.40	0.19	0.42	0.50	0.37	0.95	0.99	0.91	0.98	0.94	1.00
PCA	0.18	-0.04	0.46	0.19	0.29	0.39	0.58	0.51	0.51	0.94	0.98	0.90	0.99	0.95	1.00
RSA	0.25	0.37	0.54	0.26	0.39	0.24	0.42	0.13	0.26	0.60	0.75	0.45	0.51	0.70	0.62

square deviation computes the sample standard deviation of the differences between predicted values by the estimator and actual values. RMSE is also an indicator of the predicted and observed values. From Table VIII, we infer that the best RMSE value in-case of ELM with linear kernel is for response time and latency QoS parameters. The minimum RMSE value obtained is 0.08 for HMS metrics and conformity parameter in-case of linear kernel. From Table VIII, we observe that in-case of polynomial kernel, the performance of PCA based feature extraction technique is better for some parameters in comparison to RSA based feature selection technique and similarly the performance of RSA is better than PCA for some parameters. We do not observe a dominate approach between PCA and RSA.

IX. COMPARING ALGORITHMS USING STATISTICAL SIGNIFICANCE TESTING

Our objective is to compare several learning algorithms and assess which algorithm is better. Dietterich et al. review 5 approximate statistical tests for determining whether one learning algorithm outperforms another on a particular learning task and dataset [18]. We apply the 10-fold cross-validated paired t-test as described in the paper by Dietterich et al. [18]. We have several combination of subsets of metrics and ELM kernel functions as learning algorithms. We consider 6 different sets

TABLE X
PERFORMANCE MATRIX FOR ELM WITH RBF KERNEL

	Response Time	Availability	Throughput	Successability	Reliability	Compliance	Best Practices	Latency	Documentation	Maintainability	Modularity	Reusability	Testability	Interoperability	Conformity
BMS	0.10	0.14	0.18	0.17	0.24	0.26	0.18	0.11	0.22	0.25	0.16	0.16	0.14	0.24	0.25
HMS	0.10	0.14	0.18	0.17	0.24	0.26	0.18	0.11	0.22	0.25	0.16	0.18	0.14	0.25	0.29
OOM	0.09	0.13	0.18	0.16	0.24	0.25	0.18	0.11	0.22	0.24	0.17	0.16	0.15	0.25	0.27
AM	0.09	0.13	0.19	0.17	0.24	0.26	0.20	0.11	0.21	0.24	0.17	0.17	0.14	0.24	0.31
PCA	0.09	0.13	0.19	0.16	0.24	0.25	0.20	0.11	0.21	0.24	0.17	0.17	0.14	0.24	0.30
RSA	0.10	0.13	0.18	0.16	0.24	0.25	0.18	0.11	0.22	0.26	0.16	0.16	0.14	0.26	0.26
	MMRE														
BMS	0.34	0.31	0.74	0.35	0.67	0.60	0.39	0.56	0.91	1.03	0.60	0.71	0.55	0.52	0.47
HMS	0.34	0.32	0.74	0.35	0.66	0.61	0.38	0.56	0.92	1.05	0.58	0.76	0.53	0.53	0.65
OOM	0.33	0.31	0.73	0.34	0.70	0.58	0.38	0.56	0.89	1.03	0.61	0.65	0.53	0.56	0.49
AM	0.32	0.32	0.74	0.35	0.68	0.60	0.48	0.56	0.90	0.97	0.59	0.72	0.50	0.54	0.69
PCA	0.31	0.30	0.75	0.34	0.68	0.59	0.49	0.58	0.88	0.96	0.60	0.73	0.49	0.54	0.68
RSA	0.34	0.30	0.73	0.33	0.68	0.57	0.37	0.55	0.90	1.12	0.61	0.70	0.55	0.55	0.47
	RMSE														
BMS	0.15	0.20	0.21	0.22	0.27	0.29	0.22	0.15	0.28	0.31	0.21	0.22	0.20	0.29	0.27
HMS	0.15	0.20	0.22	0.22	0.27	0.30	0.22	0.15	0.28	0.31	0.21	0.24	0.20	0.29	0.31
OOM	0.14	0.19	0.22	0.21	0.28	0.29	0.22	0.14	0.27	0.30	0.23	0.21	0.21	0.30	0.28
AM	0.14	0.20	0.22	0.22	0.28	0.29	0.23	0.14	0.27	0.30	0.23	0.23	0.20	0.29	0.32
PCA	0.14	0.19	0.22	0.21	0.28	0.29	0.24	0.16	0.27	0.30	0.23	0.23	0.20	0.29	0.32
RSA	0.15	0.19	0.21	0.21	0.27	0.29	0.21	0.14	0.27	0.31	0.22	0.22	0.20	0.30	0.28
	r-value														
BMS	0.31	0.50	0.19	0.02	0.08	0.24	0.50	0.46	0.15	0.87	0.81	0.47	0.81	0.78	0.93
HMS	0.34	0.18	0.43	0.38	0.29	0.33	0.33	0.38	0.34	0.93	0.96	0.48	0.91	0.78	0.97
OOM	0.57	0.34	0.35	0.44	0.55	0.37	0.64	0.29	0.26	0.82	0.58	0.71	0.53	0.70	0.83
AM	0.63	0.39	0.29	0.20	0.48	0.28	0.32	0.30	0.34	0.96	0.94	0.62	0.83	0.91	0.95
PCA	0.50	0.20	0.32	0.00	0.44	0.41	0.64	0.26	0.24	0.92	0.73	0.68	0.75	0.85	0.98
RSA	0.21	0.39	0.36	0.42	0.38	0.46	0.61	0.35	0.43	0.42	0.59	0.20	0.44	0.56	0.57

of metrics: All Metrics (AM), Only Object Oriented Metrics (OOM), Harry M. Sneed's Metrics (HMS), Baski and Misra Metrics (BMS), Metrics derived after executing PCA, and metrics derived after executing RSA. We consider 6 sets of metrics as input to develop a model to predict 15 different QoS parameters. We investigate the application of extreme learning machine with three different types of kernel functions: linear kernel, polynomial kernel, and radial basis function with three different performance parameters. Hence, for each subset of metrics, a total number of three set (one for each performance measure) are used, each with 45 data points (3 kernels multiplied by 15 QoS parameters). Table XI displays the result of the 10-fold cross-validated paired t-test analysis. For each of the 3 kernels (Linear, Polynomial and RBF), 6 different subset of metrics are considered as input with three different performance parameters. The three different performance parameters are: Mean Absolute Error (MAE), Mean Magnitude of Relative Error (MMRE) and Root Mean

TABLE XI
EXPERIMENTAL RESULTS ON T-TEST BETWEEN DIFFERENT SET OF METRICS

MAE												
	Mean Difference						**p-value**					
	BMS	HMS	OOM	AM	PCA	RSA	BMS	HMS	OOM	AM	PCA	RSA
BMS	0.000	0.014	0.000	0.009	0.011	-0.013	NaN	0.004	0.986	0.140	0.008	0.008
HMS	-0.014	0.000	-0.014	-0.004	-0.002	-0.026	0.004	NaN	0.017	0.164	0.175	0.003
OOM	0.000	0.014	0.000	0.009	0.011	-0.013	0.986	0.017	NaN	0.155	0.025	0.026
AM	-0.009	0.004	-0.009	0.000	0.002	-0.022	0.140	0.164	0.155	NaN	0.570	0.035
PCA	-0.011	0.002	-0.011	-0.002	0.000	-0.024	0.008	0.175	0.025	0.570	NaN	0.005
RSA	0.013	0.026	0.013	0.022	0.024	0.000	0.008	0.003	0.026	0.035	0.005	NaN
MMRE												
	Mean Difference						**p-value**					
	BMS	HMS	OOM	AM	PCA	RSA	BMS	HMS	OOM	AM	PCA	RSA
BMS	0.000	0.037	0.000	0.026	0.027	-0.037	NaN	0.011	0.990	0.182	0.051	0.009
HMS	-0.037	0.000	-0.037	-0.010	-0.010	-0.074	0.011	NaN	0.039	0.346	0.199	0.005
OOM	0.000	0.037	0.000	0.026	0.027	-0.037	0.990	0.039	NaN	0.199	0.081	0.078
AM	-0.026	0.010	-0.026	0.000	0.000	-0.063	0.182	0.346	0.199	NaN	0.967	0.048
PCA	-0.027	0.010	-0.027	0.000	0.000	-0.064	0.051	0.199	0.081	0.967	NaN	0.014
RSA	0.037	0.074	0.037	0.063	0.064	0.000	0.009	0.005	0.078	0.048	0.014	NaN
RMSE												
	Mean Difference						**p-value**					
	BMS	HMS	OOM	AM	PCA	RSA	BMS	HMS	OOM	AM	PCA	RSA
BMS	0.000	0.018	-0.001	0.010	0.013	-0.014	NaN	0.002	0.781	0.205	0.008	0.005
HMS	-0.018	0.000	-0.020	-0.008	-0.005	-0.033	0.002	NaN	0.010	0.066	0.038	0.002
OOM	0.001	0.020	0.000	0.012	0.014	-0.013	0.781	0.010	NaN	0.167	0.021	0.050
AM	-0.010	0.008	-0.012	0.000	0.003	-0.025	0.205	0.066	0.167	NaN	0.575	0.045
PCA	-0.013	0.005	-0.014	-0.003	0.000	-0.027	0.008	0.038	0.021	0.575	NaN	0.004
RSA	0.014	0.033	0.013	0.025	0.027	0.000	0.005	0.002	0.050	0.045	0.004	NaN

TABLE XII
EXPERIMENTAL RESULTS ON T-TEST BETWEEN THREE DIFFERENT KERNELS

Mean Difference											
MAE				**MRE**				**RMSE**			
	Linear	**Polynomial**	**RBF**		**Linear**	**Polynomial**	**RBF**		**Linear**	**Polynomial**	**RBF**
Linear	0.000	0.007	-0.021	**Linear**	0.000	0.005	-0.084	**Linear**	0.000	0.005	-0.018
Polynomial	-0.007	0.000	-0.028	**Polynomial**	-0.005	0.000	-0.090	**Polynomial**	-0.005	0.000	-0.023
RBF	0.021	0.028	0.000	**RBF**	0.084	0.090	0.000	**RBF**	0.018	0.023	0.000
p-value											
MAE				**MRE**				**RMSE**			
	Linear	**Polynomial**	**RBF**		**Linear**	**Polynomial**	**RBF**		**Linear**	**Polynomial**	**RBF**
Linear	NaN	0.13	0.001	**Linear**	NaN	0.467	0.000	**Linear**	NaN	0.121	0.007
Polynomial	0.13	NaN	0.000	**Polynomial**	0.467	NaN	0.000	**Polynomial**	0.121	NaN	0.005
RBF	0.001	0.000	NaN	**RBF**	0.000	0.000	NaN	**RBF**	0.007	0.005	NaN

Squared Error (RMSE). Hence for each kernel a total three set (one for each performance measure) are used, each with 90 data points (six subsets of metrics multiplied by 15 QoS). The experimental results of t-test analysis for different performance parameter (MAE, MMRE and RMSE) and three different ELM kernels are summarized in Table XII. Table XII contains two parts. The first part of the table XII shows the mean difference value and second part shows the p-value between different pairs. Table XII reveals that there is no significant difference between the kernel function, due to the fact that p-value is greater than 0.05. However, by closely examining the value of mean difference, polynomial kernel yields better result compared to other kernels function i.e., linear and RBF kernel functions.

X. CONCLUSION

We develop a predictive model to estimate QoS parameters of web services using source code (implementing the services) metrics. We experiment with six different sets of metrics as input to develop a prediction model. The performance of these sets of metrics are evaluated using Extreme Learning Machines (ELM) with various kernel functions such as linear, polynomial and RBF kernel function. From the correlation analysis between metrics, we observe that there exists a high correlation between Object-Oriented metrics and WSDL metrics. From t-test analysis, we infer that in most of the cases, there is the difference between the various sets of metrics in terms of the performance of the estimator is not substantial but moderate. We observe that the predictive model developed using Harry M. Sneed (HMS) metrics yields better result compared to other sets of metrics such as all metrics and Baski and Misra metrics. From t-test analysis, we can also interpret that difference between the three kernel functions in-terms of their influence on the predictive accuracy is moderate. We conclude that none of the feature selection technique dominate the other and one feature selection method is better than the other for some QoS parameters and vice-versa. By assessing the value of mean difference, we infer that the polynomial kernel for ELM yields better result compared to other kernels function i.e., linear and RBF kernel functions. From performance results, it is observed that the performance of the predictive model or estimator varies with the different sets of software metrics, feature selection technique and the kernel functions. Finally, we conclude that it is possible to estimate the QoS parameters using ELM and source code metrics.

REFERENCES

[1] F. Curbera, M. Duftler, R. Khalaf, W. Nagy, N. Mukhi, and S. Weerawarana, "Unraveling the web services web: an introduction to soap, wsdl, and uddi," *IEEE Internet computing*, vol. 6, no. 2, p. 86, 2002.

[2] E. Newcomer and G. Lomow, *Understanding SOA with Web services*. Addison-Wesley, 2005.

[3] J. L. O. Coscia, M. Crasso, C. Mateos, A. Zunino, and S. Misra, *Predicting Web Service Maintainability via Object-Oriented Metrics: A Statistics-Based Approach*. Springer Berlin Heidelberg, 2012, pp. 29–39.

[4] J. L. O. Coscia, M. Crasso, C. Mateos, and A. Zunino, "Estimating web service interface quality through conventional object-oriented metrics." *CLEI Electron. J.*, vol. 16, 2013.

[5] C. Mateos, M. Crasso, A. Zunino, and J. L. O. Coscia, "Detecting wsdl bad practices in code-first web services," *Int. J. Web Grid Serv.*, vol. 7, no. 4, pp. 357–387, Jan. 2011.

[6] L. Kumar, M. Kumar, and S. K. Rath, "Maintainability prediction of web service using support vector machine with various kernel methods," *International Journal of System Assurance Engineering and Management*, pp. 1–18, 2016.

[7] S. O. Olatunji, Z. Rasheed, K. Sattar, A. Al-Mana, M. Alshayeb, and E. El-Sebakhy, "Extreme learning machine as maintainability prediction model for object-oriented software systems," *Journal of Computing*, vol. 2, no. 8, pp. 49–56, 2010.

[8] E. Al-Masri and Q. H. Mahmoud, "Qos-based discovery and ranking of web services," in *Computer Communications and Networks, 2007. ICCCN 2007. Proceedings of 16th International Conference on*, 2007, pp. 529–534.

[9] E. Al Masri and Q. H. Mahmoud, "Investigating web services on the world wide web," *Proceedings of the 17th International Conference on World Wide Web*, pp. 795–804, 2008.

[10] D. Baski and S. Misra, "Metrics suite for maintainability of extensible markup language web services," *IET Software*, vol. 5, no. 3, pp. 320–341, 2011.

[11] H. M. Sneed, "Measuring web service interfaces," in *Web Systems Evolution (WSE), 2010 12th IEEE International Symposium on*, 2010, pp. 111–115.

[12] M. Jureczko and D. Spinellis, *Using Object-Oriented Design Metrics to Predict Software Defects*, ser. Monographs of System Dependability, 2010, vol. Models and Methodology of System Dependability, pp. 69–81.

[13] D. Spinellis, "Tool writing: a forgotten art? (software tools)," *IEEE Software*, vol. 22, no. 4, pp. 9–11, 2005.

[14] R. W. Swiniarski and A. Skowron, "Rough set methods in feature selection and recognition," *Pattern recognition letters*, vol. 24, no. 6, pp. 833–849, 2003.

[15] G.-B. Huang, D. H. Wang, and Y. Lan, "Extreme learning machines: a survey," *International Journal of Machine Learning and Cybernetics*, vol. 2, no. 2, pp. 107–122, 2011.

[16] S. Ding, Y. Zhang, X. Xu, and L. Bao, "A novel extreme learning machine based on hybrid kernel function," *Journal of Computers*, vol. 8, no. 8, pp. 2110–2117, 2013.

[17] B. A. Kitchenham, L. M. Pickard, S. G. MacDonell, and M. J. Shepperd, "What accuracy statistics really measure [software estimation]," *IEE Proceedings-Software*, vol. 148, no. 3, pp. 81–85, 2001.

[18] T. G. Dietterich, "Approximate statistical tests for comparing supervised classification learning algorithms," *Neural Comput.*, vol. 10, no. 7, pp. 1895–1923, Oct. 1998.