EPiC Series in Computing

Volume 79, 2021, Pages 171–180

Proceedings of ISCA 34th International Conference on Computer Applications in Industry and Engineering

An Empirical Analysis on the Detection of Web ServiceAnti-Patterns Using Various Combinations of WSDL Metrics

Sahithi Tummalapalli[1], Lov kumar[1], Lalita Bhanu Murthy Neti[1], and SantanuKumar Rath[2]

[1] BITS Pilani Hyderabad Campus, India.

2 NIT, Rourkela, India.

p20170433@hyderabad.bits-pilani.ac.in, lovkumar@hyderabad.bits-pilani.ac.in, bhanu@hyderabad.bits-pilani.ac.in, skrath@nitrkl.ac.in

Abstract

Many IT enterprises today use Service Oriented Architecture(SOA) as the effective architectural approach for building their systems. Service-Based Systems(SBS) like other complex frameworks are liable to change to fit in the new user requirements. These may lead to the deterioration of the quality and design of the software systems and may cause the materialization of poor solutions called Anti-patterns. Similar to object-oriented systems, web services also suffer from anti-patterns due to bad programming practices, design, and implementation. An anti-pattern is defined as a commonly used process, structure, or pattern of action that, despite initially appearing to be an effective and appropriate response to a problem, has more bad consequences than good ones. Anti- pattern detection using Web Service Description Language(WSDL) metrics can be used as a part of the software development life cycle to reduce the maintenance of the software system and also to improve the quality of the software. The work is motivated by the need to develop an automatic predictive model for the prediction of web services anti- patterns using static analysis of the WSDL metrics. The core ideology of this work is to empirically investigate the effectiveness of classifier techniques i.e, ensemble and deep learning techniques in the prediction of web service anti-patterns. In this paper, we present an empirical analysis on the application of seven feature selection techniques, six data sampling techniques, and ten classifier techniques for the prediction of four different types of anti-patterns. The results confirm the predictive ability of WSDL metrics in the prediction of SOA anti-patterns.

1. Introduction

A web service is an assortment of protocols and requirements utilized for trading data among applications. Web services are advanced dependent on standards that aid interoperability. They are utilized for the growing distributed system based on service-oriented architecture. Software developers can identify all the web services required to build a specific application and invoke all the desired web services. There are several benefits of using web services; for example, they use the SOAP mechanism, which is more efficient than regular HTTP. They help develop applications that are independent of programming languages.

SOA (Service Oriented Architecture) permits building various kinds of Service-Based Systems (SBSs) similar to Amazon, eBay, Dropbox, etc. The advancement of such systems raises sev- eral demanding situations. SBSs should advance to fit new user prerequisites and adapt new execution contexts, including the addition of the latest devices and technology. The design and Quality of Service (QoS) of SBSs may additionally debase the design because of a majority of these modifications and frequently result in a standard negative solution to habitual prob- lems, referred to as Anti-patterns[3]. These systems inside the design demonstrate a violation of fundamental design principles and negatively sway design quality. Anti-patterns makes it difficult for the evolution and advancement of the software system, but they also tend to help to detect problems within the code, the architecture, and the management of software projects. The web service anti-patterns which we considered in this paper are GOWS: God Object Web Service(AP1), FGWS: Fine-Grained Web Service(AP2), CWS: Chatty Web Service(AP3), and DWS: Data Web Service(AP4). Regardless of the general use of Web services, no particular and automatic methodology for detecting such anti-patterns from their Web Service Definition Language(WSDL) files exists to date. The motivation behind the paper is thus to explore the techniques to detect the anti-patterns using WSDL metrics automatically. In this paper, the WSDL metric set is used to detect the anti-patterns instead of the Object-oriented metrics used widely. We empirically investigate the effectiveness of 6 data sampling techniques, seven feature selection techniques, four different subsets of WSDL quantitative metrics, and ten classification techniques in the detection of web service anti-patterns.

2. Objectives and Research Questions

The primary objective of the work presented in this paper is to investigate the application of ensemble and deep learning techniques in the prediction of SOA anti-patterns using WSDL metrics as features. The other objective is to build tools and techniques for the automatic prediction of anti-patterns and investigate the relationship between the Web Service Description Language(WSDL) metrics and the incidence of anti-patterns in web services. The following research questions(RQ) have been answered in this work:

- **RQ1:** Is there an essential differentiation between the performance displayed by the five data sampling techniques over the original data?
- **RQ2:** Is there a quantifiably enormous distinction between the performance of the mod- els developed by utilizing the features selected by applying the seven feature selection techniques and the subset of WSDL quantitative metrics over the exhibition of the model produced by utilizing all the WSDL code quantitative metrics?
- **RQ3:** What is the general execution of the ensemble techniques and deep learning tech- niques considered concerning AUC and f-measure metrics? Is there a genuinely critical differentiation in the expected execution of the ten classifier techniques?

3. Related Work

Upadhyaya et al. [7] proposed an approach to detect 9 SOA patterns. It is observed from the literature reviewed here that that the research on SOA anti-pattern detection still needs to be explored thoroughly. Peter Chen et al. [1] developed a framework that automatically flagsthe performance anti-patterns in the source code using the automated developed ORM(object-relational model) performance anti-patterns. Ouni et al. [4] introduced innovative genetic programming to detect web services anti-pattern by generating detection rules based on threshold values and a combination of different metrics. The validation of the above approach is done on 310 Web services to detect the five anti-patterns. Ouni et al. [5] used cooperative parallel evolutionary algorithms (P-EA), an automated approach to detect the anti-patterns. The idea behind their innovation is that the combination of several detection algorithms executing in parallel optimization processes would give better results. The results compared with the random search and the population-based searches gave a precision score of 0.89. Jaffar et al. [2] argued in his paper that classes taking part in anti-pattern and patterns of software designs have dependencies with other classes, i.e., unvarying and mutating dependencies, that may spread issues to different classes. A significant portion of the proposed approaches is based on source code metrics, code-based analysis, or generation of rule cards to detect anti-patterns. In this paper, we propose investigating the effectiveness of WSDL metrics in detecting SOA anti-patterns.

4. Experimental Dataset

Recent studies investigated the use of source code metrics in the detection of anti-patterns. While the indicative power of these source code metrics is validated on Object-oriented and service-oriented architectures, WSDL metric set effectiveness is not validated yet for SOA anti-pattern detection.

In this study, we have used the dataset from the GitHub repository [1]. The dataset has the files with .wsdl extension. These WSDL files are collected from the web services of various domains such as education, finance, travel, etc. We compute the WSDL metrics for each WSDL file.

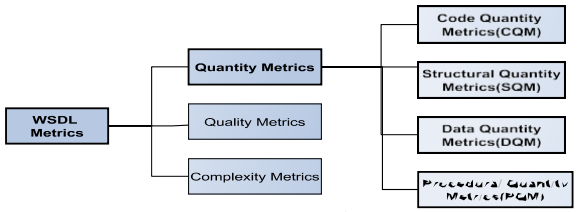

Figure 1: WSDL Metric Set Taxonomy

A close investigation revealed that the GOWS anti-pattern exists in 21 out of 226 web services considered. The percentage of the existing FGWS, CWS, DWS, and AWS anti-patternsin the dataset is 5.75, 6.19, 9.29, and 10.62, respectively. These low percentages indicate the presence of a class imbalance problem in the dataset considered. The number of instances in the minority class(anti-patterns existing) is far less than the number of cases in the majority class(anti-patterns not existing). In this case, the number of web services in which anti-pattern exists is in the minority (5% to 11%) but not rare(<2%). The other cause for concern here is the sample size of the dataset, which is small. Therefore, our dataset has two issues: one is class imbalance distribution, and the other is the sample size.

[1]https://github.com/ouniali/WSantipatterns

5. Research Framework

Figure 2 illustrates the framework for the prediction of SOA anti-patterns using WSDL metrics as input. The framework, as shown in Fig 2, is a multi-step procedure that is discussed in detail in this Section. Firstly, ROSE2 tool[6] is used for computing the WSDL metrics from the web services in the dataset. Next, we investigate the role of quantity metrics among the WSDL metrics in detecting SOA anti-patterns. Then we use feature selection techniques dis- cussed in section 5.3 for significant features selection. Besides the significant features selected using the mentioned feature selection techniques, we use Structural Quantity Metrics(SQM), Procedural Quantity Metrics(PQM), Data Quantity Metrics(DQM), and All WSDL quantity Metrics(AM) as input for the models generated for the detection of web service anti-patterns. Next, to deal with the class imbalance problem as discussed in Section , we use various variants of Synthetic Minority Oversampling Technique(SMOTE), i.e., Borderline SMOTE(BSMOTE), SVM-SMOTE(SSMOTE), SMOTE- Edited Nearest Neighbour(SMOTEENN), and SMOTE- TOMEK(SMOTEOM) besides the original dataset(OD). Further, We use various ensembling techniques besides the deep learning technique with a specific number of hidden layers to train the predictive models for anti-pattern detection. Finally, the performance parameters such as Area Under Curve (AUC) and Accuracy are utilized in this work to gauge the effect and reliability of the models generated for SOA anti-patterns detection.

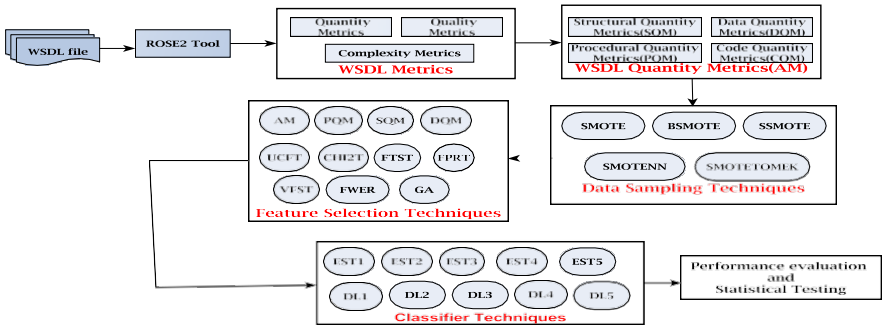

Figure 2: Proposed Framework

5.1 Computation of WSDL Metrics

The dataset considered in this study has web services in WSDL format for which the WSDL metrics are computed using the ROSE2 tool[6]. ROSE2 is a tool that uses the principles of rough set theory and rules discovery techniques. WSDL metrics include Complexity metrics, Qualitative metrics, and Quantity metrics. In this paper, we aim to investigate the role of WSDL quantity metrics in anti-patterns detection. The taxonomy of the WSDL metric set obtained is shown in Figure 1.

5.2 Data Sampling Techniques

The selection of an appropriate sampling technique plays a critical role in the research study, as it significantly impacts the quality of our results and findings. As discussed in section 5, the dataset considered is having a class imbalance problem, and we are choosing the data sampling

technique SMOTE and its variants to solve this problem. In this paper, we are considering five different data sampling techniques namely SMOTE, Borderline Smote(BSMOTE), SVM-SMOTE(SSMOTE), SMOTE- Edited Nearest Neighbour(SMOTENN), and SMOTETOMEK along with the original dataset(OD) to generate the predictive models.

5.3 Feature Selection Techniques

WSDL metrics computed have some irrelevant and redundant features. Research revealed that the high dimensional feature space consisting of irrelevant and redundant features decreases the performance of the classifiers. The presence of many features, i.e., WSDL code metrics in our case, pose an intrinsic challenge to classifier algorithms. Hence, it is important to remove these irrelevant features, for which we employed several feature selection techniques in this work. We use various techniques such as feature selection using low variance (VFST), Uncorrelated features data(UCFT), CHI2 value Test(CHI2T), ANOVA F-value between label and feature(FTST), False Positive Rate Test(FPRT), Family Wise Error Rate(FWER) and Genetic Algorithm(GA) for selecting significant features.

5.4 Classifier Techniques

In this paper, we have applied five different ensemble techniques i.e., Bagging classifier (EST1), Random Forest Classifier (EST2), Extra Trees Classifier (EST3), AdaBoost Classifier (EST4), Gradient Boosting Classifier (EST5) and a deep learning technique with a different number of hidden layers i.e., DL with one hidden layer(DL1), DL with two hidden layers (DL2), DL with three hidden layers (DL3), DL with four hidden layers (DL4) and DL with five hidden layers (DL5) for training the predictive models for detecting anti-patterns.

6. Experimental Results

In this work, we empirically investigated the application of eleven feature selection techniques, six data sampling techniques, and ten classifier techniques to predict four different types of anti-patterns.

- Table 1 shows the experimental results in terms of accuracy for the models developed for the detection of GOWS anti-pattern. The results for the other three anti-patterns, along with AUC results, were not included due to space constraints.
- From Table 1, it has been observed that the value of AUC parameters of the model trained using Random Forest Classifier(EST2) and Extra Trees Classifier(EST3) are higher than those of the other models.
- The performance of the models developed using all the WSDL quantitative metrics(AM) and Feature selection using Low Variance (VFST) showed similar and high performance compared to the models developed using other feature sets as input.

7. Comparative Analysis

RQ1: Is there an essential differentiation between the performance displayed by the five data sampling techniques over the original data?

In this section, we analyzed the differences in the performance of the dataset generated after applying the class imbalance techniques and the original dataset. We use box plots to represent the Accuracy and the Area Under Curve(AUC) for the generated models. We have also

used Wilcoxon rank-sum test to compare the models' performance using different data sets generated.

Table 1: Accuracy of all models: GOWS anti-pattern

Data Sampling Technique	Feature Selection Technique	EST 1	EST 2	EST 3	EST 4	EST 5	DL1	DL2	DL3	DL4	DL5	Data Sampling Technique	Feature Selection Technique	EST1	EST 2	EST 3	EST 4	EST 5	DL1	DL2	DL3	DL4	DL5
ORG	AM	90.91	91.41	92.93	90.91	88.38	89.39	90.91	92.42	90.91	90.4	SVMSMOTE	AM	84.21	90.6	92.48	86.09	82.71	80.83	81.58	94.74	90.98	91.73
ORG	SQM	90.91	91.92	94.44	89.9	88.89	91.41	90.4	90.4	90.4	90.91	SVMSMOTE	SQM	91.16	90.88	93.09	87.85	88.95	83.7	88.12	89.5	90.33	90.06
ORG	DQM	89.39	91.41	91.41	91.41	87.37	88.89	89.9	90.91	90.4	89.39	SVMSMOTE	DQM	87.68	92.96	94.37	90.49	88.73	69.37	74.65	79.93	80.63	81.34
ORG	PQM	91.41	89.9	90.4	89.39	88.89	90.4	90.4	89.39	88.89	88.89	SVMSMOTE	PQM	89.47	90.46	93.42	89.47	86.18	68.75	67.76	77.96	76.64	72.04
ORG	VFST	90.91	90.91	93.94	90.91	88.38	89.9	90.4	91.92	90.91	91.41	SVMSMOTE	VFST	84.7	92.91	95.15	86.94	89.18	81.34	81.72	90.3	88.06	87.31
ORG	UCFT	89.9	89.9	88.89	88.89	87.88	91.41	91.41	91.41	91.41	91.41	SVMSMOTE	UCFT	78.67	88.46	90.21	84.62	82.17	63.64	63.29	61.54	63.64	61.54
ORG	CHI2T	87.88	85.86	88.38	85.35	84.85	91.41	91.41	91.41	91.41	91.41	SVMSMOTE	CHI2T	78.52	78.89	81.11	79.63	77.41	66.67	67.04	67.78	64.81	64.07
ORG	FTST	89.39	89.39	87.88	90.4	89.39	91.41	91.41	90.91	91.41	91.41	SVMSMOTE	FTST	90.06	92.82	88.4	88.95	88.95	86.74	69.34	87.85	88.95	87.85
ORG	FPRT	91.92	92.42	90.91	91.41	88.38	89.9	90.91	91.41	90.91	90.91	SVMSMOTE	FPRT	87.5	91.79	93.93	91.43	90	82.5	83.57	87.5	85.71	85
ORG	FWER	91.41	90.4	90.91	91.41	88.38	89.9	90.4	91.92	90.4	90.91	SVMSMOTE	FWER	87.19	92.53	92.88	88.26	90.04	81.85	81.14	87.54	85.05	85.05
ORG	GA	91.41	89.39	92.42	85.86	87.88	91.41	91.41	90.4	89.9	90.4	SVMSMOTE	GA	86.23	89.13	93.84	89.13	89.49	65.22	67.03	75.36	67.75	72.1
SMOTE	AM	78.45	89.78	91.71	86.46	88.12	69.61	70.99	93.37	87.83	90.88	SMOTEENN	AM	89.61	94.16	95.13	94.81	91.56	83.12	82.14	92.86	91.56	91.23
SMOTE	SQM	82.6	90.06	87.57	88.89	80.94	70.99	69.89	77.07	75.69	76.8	SMOTEENN	SQM	95.07	95.77	98.94	96.48	95.77	64.08	85.92	95.07	95.42	95.07
SMOTE	DQM	79.28	88.4	89.78	79.83	80.66	72.1	71.55	79.83	76.24	79.01	SMOTEENN	DQM	92.28	91.95	96.64	91.95	92.62	82.55	85.37	87.92	89.26	88.26
SMOTE	PQM	76.24	86.19	87.02	87.02	83.98	67.68	68.78	79.56	76.8	78.45	SMOTEENN	PQM	89.62	94.23	96.54	96.15	95.77	89.23	90.38	91.92	91.54	91.54
SMOTE	VFST	83.43	89.5	90.33	85.08	85.36	77.9	76.8	88.67	86.19	86.46	SMOTEENN	VFST	89.97	93.65	96.66	94.31	90.97	84.95	84.95	92.31	87.96	88.63
SMOTE	UCFT	78.73	87.29	88.95	83.98	83.15	47.51	60.77	69.61	70.99	46.96	SMOTEENN	UCFT	92.34	97.58	95.56	99.19	97.18	77.42	72.98	78.63	81.85	68.15
SMOTE	CHI2T	82.6	81.49	83.43	80.11	81.22	56.91	68.51	77.62	67.96	64.64	SMOTEENN	CHI2T	97.5	98.21	99.29	99.29	97.5	89.29	74.29	96.43	76.79	90.36
SMOTE	FTST	80.94	80.39	81.49	80.66	81.49	72.1	63.26	76.24	74.31	74.03	SMOTEENN	FTST	96.97	98.11	98.13	97.73	97.35	85.98	91.29	96.59	94.7	93.94
SMOTE	FPRT	76.8	87.29	91.71	84.81	82.87	75.14	71.76	84.53	86.19	79.28	SMOTEENN	FPRT	90.66	94.12	97.58	93.08	91.7	87.22	85.47	86.59	86.16	86.51
SMOTE	FWER	79.01	88.12	91.99	84.53	82.04	69.34	67.96	85.91	82.87	82.87	SMOTEENN	FWER	90.24	95.29	96.97	93.6	90.57	87.21	89.9	91.25	89.9	88.55
SMOTE	GA	80.94	84.25	87.57	84.25	80.66	73.2	75.97	75.69	74.59	74.31	SMOTEENN	GA	89.68	94.66	95.37	93.95	92.88	87.19	88.61	89.68	88.97	89.68
BSMOTE	AM	89.5	92.82	91.16	88.4	89.5	82.04	85.36	90.61	92.27	91.16	SMOTETOMEK	AM	79.55	90.34	92.9	84.09	86.93	78.41	80.97	94.32	93.75	92.05
BSMOTE	SQM	87.85	90.33	91.44	87.02	88.12	69.34	67.13	75.97	78.18	77.62	SMOTETOMEK	SQM	85.8	90.63	92.61	83.24	83.52	65.63	62.22	77.84	75.28	77.56
BSMOTE	DQM	91.71	94.48	94.48	91.71	91.16	77.62	79.83	83.98	87.85	84.81	SMOTETOMEK	DQM	83.05	88.79	91.95	82.47	82.18	74.14	74.71	80.75	77.59	77.59
BSMOTE	PQM	88.95	90.88	91.99	89.5	90.33	78.45	80.11	88.67	88.95	88.4	SMOTETOMEK	PQM	75.99	86.44	87.85	85.31	82.2	70.34	70.62	76.55	74.01	74.86
BSMOTE	VFST	90.06	92.54	92.82	90.33	90.88	85.91	88.12	92.54	90.88	90.33	SMOTETOMEK	VFST	79.49	89.04	92.13	82.87	87.08	75.28	75.56	92.42	84.27	83.71
BSMOTE	UCFT	82.6	90.61	90.61	85.08	82.87	54.7	61.88	62.43	60.77	61.88	SMOTETOMEK	UCFT	83.64	88.48	91.21	88.79	87.27	52.73	58.48	63.94	62.73	70.3
BSMOTE	CHI2T	88.95	85.36	85.08	85.36	84.81	58.29	64.64	82.04	72.1	76.8	SMOTETOMEK	CHI2T	82.94	84.41	85.88	82.06	82.35	66.47	56.18	77.35	76.76	72.06
BSMOTE	FTST	88.95	90.61	90.61	86.74	88.12	77.35	78.18	78.18	83.98	79.56	SMOTETOMEK	FTST	79.07	82.85	83.43	81.1	81.4	70.35	70.35	71.22	66.57	71.8
BSMOTE	FPRT	87.57	92.27	93.65	88.95	90.06	79.01	80.94	91.99	86.74	86.19	SMOTETOMEK	FPRT	80	89.43	90.86	84.57	86.29	77.43	81.14	88.57	84	86.57
BSMOTE	FWER	88.95	91.99	91.16	85.64	89.5	80.39	85.64	89.5	90.06	88.95	SMOTETOMEK	FWER	76.4	86.8	92.13	82.02	79.78	69.94	74.72	87.36	86.24	85.39
BSMOTE	GA	90.88	92.82	92.54	90.06	90.61	75.97	76.8	82.87	82.04	81.22	SMOTETOMEK	GA	82.95	87.28	91.33	83.24	81.79	70.81	71.97	72.83	74.28	72.54

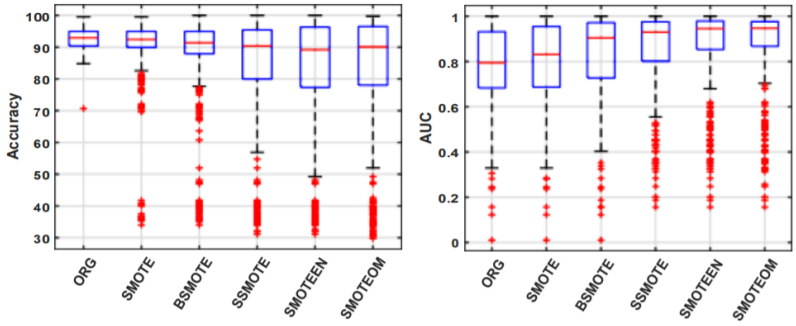

Figure 3: Box-plot for Accuracy and AUC: Data Sampling Techniques

Comparision of various sampling techniques using Descriptive Statistics and Box-plot diagram: From Figure 3 and Table 2, we compare the differences in performances between the models generated using the original dataset and the unbiased samples obtained from the dataset after applying the considered data sampling techniques. Box-plot represents the first and the third quartile, and the red line represents the median value. From Figure 3 and Table 2, we observed that the mean AUC value of the model developed using SMOTEOM is high when compared to the other models. The model developed using SMOTEOM achieved 0.87 average AUC value, 1.00 max AUC, and 0.98 Q3 AUC value, i.e., 25% models developed using SMOTEOM have 0.98 AUC value. The performance level of the models developed using SMOTEEN is very similar to the performance shown by the models developed using SMOTEOM. However, the model developed using all the WSDL quantitative metrics(AM) has a low predictive ability compared to other techniques.

Comparison of Data Sampling Techniques: Significant Test: In this study, Wilcoxon signed-rank test is applied on the AUC, F-measure, and accuracy for statistically comparing the predictive ability of web service anti-pattern detection techniques using different sampling techniques. The primary motivation of this statistical testing is to find whether the models developed using different sampling techniques have a significant enhancement or not. pvalue is

used in this test to determine whether to accept or reject the null hypothesis. The considered null hypothesis for this work is:" The web service anti-pattern detection models developed us- ing different sampling techniques are significantly the same." The considered null hypothesis is accepted if the pvalue obtained using the rank-sum test is more significant than 0.05. Table 3 depicts the results of the Wilcoxon signed-rank test on different pairs of data sampling techniques. From Table 3, we observed that most of the comparison points are having values less than 0.05. We conclude that the models developed by considering different sampling techniques as input are significantly different for most cases.

Table 2: Statistical Measures: Data Sampling Techniques

	Min	Median	Q3	Max	Q1	Mean
ORG	0.01	0.79	0.93	1.00	0.68	0.79
SMOTE	0.01	0.83	0.95	1.00	0.69	0.80
BSMOTE	0.01	0.90	0.97	1.00	0.73	0.83
SSMOTE	0.16	0.93	0.98	1.00	0.80	0.86
SMOTEEN	0.16	0.95	0.98	1.00	0.85	0.87
SMOTEOM	0.16	0.95	0.98	1.00	0.87	0.87

Table 3: Significance Test: Data Sampling Techniques

	ORG	SMOTE	BSMOTE	SSMOTE	SMOTEEN	SMOTEOM
ORG	1	0.037	4.26E-07	3.20E-16	1.43E-22	6.77E-24
SMOTE	0.037	1	0.005	1.46E-08	3.54E-13	3.07E-14
BSMOTE	4.26E-07	0.005	1	0.006	9.87E-06	1.65E-06
SSMOTE	3.20E-16	1.46E-08	0.006253687	1	0.066	0.024
SMOTEEN	1.43E-22	3.54E-13	9.87E-06	0.066	1	0.664
SMOTEOM	6.77E-24	3.07E-14	1.65E-06	0.024	0.664	1

RQ2: Is there a quantifiably enormous distinction between the performance of models developed by utilizing the features selected by applying the seven feature selection techniques and the subset of WSDL quantitative metrics over the exhibi- tion of the model produced by using all the WSDL code quantitative metrics?

This study used eleven different feature selection techniques to remove redundant features and select the right sets of relevant features. We have validated the performance of the models developed using a different subset of WSDL quantitative features using various performance values such as AUC, F-measure, and accuracy and compared their performance using Descriptive Statistics, box-plot, and significant tests.

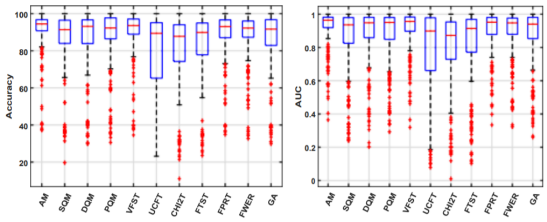

Figure 4: Box-plot for Accuracy and AUC: Feature Selection Techniques

Comparison of Different sets of Features using Descriptive Statistics and Box-plot diagram: Figure 4 represents the performance value, i.e., AUC and accuracy of the

models trained using the subset of WSDL quantitative features(SQM; DQM; PQM), selected set of features using different feature selection techniques (VFST; UCFT; CHI2T; FTST; FPRT;FWER; GA) and all the WSDL Quantitative features(AM). From Figure 4 and Table 4, we can see that the models developed using VFST and AM have slightly better performance compared to other techniques. The models developed using VFST achieved 0.91 mean AUC, 1.00 max AUC, and 0.98 Q3 AUC, i.e., 25% models developed using VFST have 0.98 AUC value. It is also observed that the models developed using AM have very similar performance to the modelsdeveloped using VFST, but the number of features in AM is very high compared to the number of features selected using the VFST technique.

Table 4: Statistical Measures: Feature Selection Techniques

	Min	Median	Q3	Max	Q1	Mean
AM	0.36	0.96	0.98	1.00	0.92	0.92
SQM	0.24	0.94	0.98	1.00	0.83	0.86
DQM	0.20	0.95	0.98	1.00	0.86	0.88
PQM	0.29	0.95	0.98	1.00	0.85	0.88
VFST	0.32	0.96	0.98	1.00	0.90	0.91
UCFT	0.08	0.90	0.98	1.00	0.66	0.78
CHI2T	0.01	0.87	0.95	1.00	0.73	0.79
FTST	0.10	0.91	0.97	1.00	0.77	0.82
FPRT	0.33	0.95	0.98	1.00	0.88	0.90
FWER	0.32	0.95	0.98	1.00	0.88	0.90
GA	0.26	0.94	0.98	1.00	0.85	0.87

Table 5: Significance Test: Feature Selection Techniques

	AM	SQM	DQM	PQM	VFST	UCFT	CHI2T	FTST	FPRT	FWER	GA
AM	1.00	0.00	0.01	0.03	0.34	0.00	0.00	0.00	0.04	0.00	0.00
SQM	0.00	1.00	0.22	0.15	0.00	0.01	0.00	0.01	0.07	0.21	0.85
DQM	0.01	0.22	1.00	0.92	0.12	0.00	0.00	0.00	0.64	0.89	0.34
PQM	0.03	0.15	0.92	1.00	0.17	0.00	0.00	0.00	0.75	0.78	0.32
VFST	0.34	0.00	0.12	0.17	1.00	0.00	0.00	0.00	0.26	0.08	0.01
UCFT	0.00	0.01	0.00	0.00	0.00	1.00	0.22	0.59	0.00	0.00	0.01
CHI2T	0.00	0.00	0.00	0.00	0.00	0.22	1.00	0.02	0.00	0.00	0.00
FTST	0.00	0.01	0.00	0.00	0.00	0.59	0.02	1.00	0.00	0.00	0.00
FPRT	0.04	0.07	0.64	0.75	0.26	0.00	0.00	0.00	1.00	0.50	0.15
FWER	0.00	0.21	0.89	0.78	0.08	0.00	0.00	0.00	0.50	1.00	0.41
GA	0.00	0.85	0.34	0.32	0.01	0.01	0.00	0.00	0.15	0.41	1.00

Comparison of Different Sets of Features: Significance Test: In this section, the Wilcoxon rank-sum test is applied to the accuracy and AUC for statistically comparing the predictive ability of web service anti-pattern detection techniques developed using different setsof features as input. The motive of this testing is to determine whether the performance of the developed models depends on input sets of features. The null hypothesis in this section: ”The web service prediction models developed by considering different sets of feature as input are significantly same.”The considered null hypothesis is accepted if the obtained p-values using Wilcoxon signed-rank test are greater than 0.05. The results of the Wilcoxon signed-rank test are shown in Table 5. From the information in Table 5, we observe that most of the comparison points are having a p-value less than 0.05. Hence, we conclude that the models developed using different feature sets as input are significantly different.

RQ3: What is the general execution of the ensemble techniques and deep learn- ing techniques considered concerning AUC and f-measure metrics? Is there a genuinely critical differentiation in the expected performance of the ten classifier techniques?

The predictive ability of web service anti-pattern detection models developed using different classification techniques are computed using performance measures such as Accuracy and AUC. They are compared with the help of Descriptive Statistics, Box-plot, and significance tests. In this work, we used five ensemble learning techniques and five deep learning techniques with

5-fold cross-validation to train anti-pattern prediction models.

Table 6: Statistical Measures: Classification Techniques

	Min	Median	Q3	Max	Q1	Mean
EST1	0.64	0.94	0.98	1.00	0.88	0.92
EST2	0.45	0.98	0.99	1.00	0.95	0.95
EST3	0.46	0.98	0.99	1.00	0.96	0.96
EST4	0.60	0.95	0.98	1.00	0.90	0.93
EST5	0.16	0.95	0.98	1.00	0.88	0.89
DL1	0.01	0.85	0.95	1.00	0.54	0.73
DL2	0.10	0.88	0.96	1.00	0.65	0.78
DL3	0.14	0.93	0.97	1.00	0.76	0.84
DL4	0.12	0.93	0.97	1.00	0.75	0.83
DL5	0.10	0.92	0.97	1.00	0.74	0.82

Table 7: Significance Test: Classification Techniques

	EST 1	EST 2	EST 3	EST 4	EST 5	DL1	DL2	DL 3	DL 4	DL5
EST 1	1.00	0.00	0.00	0.06	0.96	0.00	0.00	0.00	0.00	0.00
EST 2	0.00	1.00	0.06	0.00	0.00	0.00	0.00	0.00	0.00	0.00
EST 3	0.00	0.06	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
EST 4	0.06	0.00	0.00	1.00	0.13	0.00	0.00	0.00	0.00	0.00
EST 5	0.96	0.00	0.00	0.13	1.00	0.00	0.00	0.00	0.00	0.00
DL1	0.00	0.00	0.00	0.00	0.00	1.00	0.12	0.00	0.00	0.00
DL2	0.00	0.00	0.00	0.00	0.00	0.12	1.00	0.00	0.01	0.04
DL3	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.48	0.25
DL4	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.48	1.00	0.64
DL5	0.00	0.00	0.00	0.00	0.00	0.00	0.04	0.25	0.64	1.00

Comparison of Classification Techniques using Descriptive Statistics and Box- plot diagram: Figure 5 and Table 6 shows the performance measures, i.e., AUC and Accuracy of different classifier techniques using Box-plot diagrams and descriptive statistics. From Figure 5 and Table 6, we observe that the models trained using EST2 and EST3 have better predictive ability to detect the web service anti-patterns as compared to other models. The models developed using EST2 achieved 0.98 median AUC, 0.95 mean AUC, 0.99 Q3 AUC, and 1.00 max AUC. Similarly, the models developed using EST3 achieved 0.98 median AUC, 0.96 mean AUC, 0.99 Q3 AUC, and 1.00 max AUC. However, the models developed using the Deep learning technique with one hidden layer(DL1) have a low predictive ability compared to other methods.

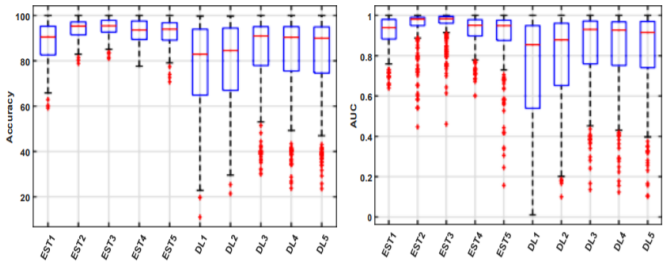

Figure 5: Box-plot for Accuracy and AUC: Classifier Techniques

Comparison of Classification Techniques: Significant Test: In this study, the Wilcoxon signed-rank test is applied to the AUC and Accuracy for statistically comparing the predictive ability of web service anti-pattern detection models using different classifiers. The ideology of this test is to find whether the models developed using different classifier techniques are significantly different or not. The null hypothesis in this study is "The web service anti-pattern prediction models trained using different classification algorithms are significantly same." The considered null hypothesis is accepted if the p-value obtained using Wilcoxon signed- rank test is greater than 0.05. Table 7 depicts the results of the Wilcoxon signed-rank test on different pairs

of classifiers. From Table 7, we observe that most of the comparison points have a value less than 0.05. Hence, we conclude that the models trained using different classifiers are significantly different for most cases.

8. Conclusion and Future Work

The primary motivation of this work is to develop an automatic predictive model for the prediction of web services anti-patterns using static analysis of the WSDL metrics. In this work, we empirically investigated the effectiveness of classifier techniques, i.e., ensemble and deep learning techniques, in predicting web service anti-patterns. Experimental analysis revealed that the model trained using Random Forest Classifier(EST2) and Extra Trees Classifier(EST3) have better performance than other models. We observed that the models developed using all the WSDL quantitative metrics(AM) and Feature selection using Low Variance (VFST) showed similar and high performance compared to the models developed using other feature sets as input. It is also observed that the models developed after applying SMOTEOM have better performance when compared to the other models. There is much more scope for predicting the web service anti-patterns using the WSDL metrics.

References

[1] Tse-Hsun Chen, Weiyi Shang, Zhen Ming Jiang, Ahmed E Hassan, Mohamed Nasser, and Parmin-der Flora. Detecting performance anti-patterns for applications developed using object-relational mapping. In *Proceedings of the 36th International Conference on Software Engineering*, pages 1001–1012, 2014.

[2] Fehmi Jaafar, Yann-Gaël Guéhéneuc, Sylvie Hamel, Foutse Khomh, and Mohammad Zulkernine. Evaluating the impact of design pattern and anti-pattern dependencies on changes and faults. *Empirical Software Engineering*, 21(3):896–931, 2016.

[3] Jaroslav Král and Michal Zemlicka. Crucial service-oriented antipatterns. vol. 2. *International Academy, Research and Industry Association (IARIA)*, pages 160–171, 2008.

[4] Ali Ouni, Raula Gaikovina Kula, Marouane Kessentini, and Katsuro Inoue. Web service antipatterns detection using genetic programming. In *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, pages 1351–1358. ACM, 2015.

[5] Ali Ouni, Marouane Kessentini, Katsuro Inoue, and Mel O Cinnéide. Search-based web service antipatterns detection. *IEEE Transactions on Services Computing*, 10(4):603–617, 2015.

[6] Bartłomiej Predki, Roman Słowiński, Jerzy Stefanowski, Robert Susmaga, and Szymon Wilk. Rose- software implementation of the rough set theory. In *International Conference on Rough Sets and Current Trends in Computing*, pages 605–608. Springer, 1998.

[7] Bipin Upadhyaya, Ran Tang, and Ying Zou. An approach for mining service composition patterns from execution logs. *Journal of Software: Evolution and Process*, 25(8):841–870, 2013.