# An Empirical Analysis on Web Service Anti-Pattern Detection using a Machine Learning Framework

Lov Kumar
NIT Rourkela, India
lovkumar505@gmail.com

Ashish Sureka
Ashoka University, India
ashish.sureka@ashoka.edu.in

***Abstract*—Web Services are application components characterised by interoperability, extensibility, distributed application development and service oriented architecture. A complex distributed application can be developed by combing several third-party web-services. Anti-patterns are counter-productive and poor design and practices. Web-services suffers from a multitude of anti-patterns such as God object Web service and Fine grained Web service. Our work is motivated by the need to build techniques for automatically detecting common web-services anti-patterns by doing a static analysis of the source code implementing a web-service. Our approach is based on the premise that summary values of object oriented source code metrics computed at a web-service level can be used as a predictor for anti-patterns. We present an empirical analysis of $4$ data sampling technique to encounter the class imbalance problem, $5$ feature ranking technique to identify the most informative and relevant features and $8$ machine learning algorithms for predicting $5$ different types of anti-patterns on $226$ real-world web-services across several domains. We conclude that it is possible to predict anti-patterns using source code metrics and a machine learning framework. Our analysis reveals that the best performing classification algorithm is Random Forest, best performing data sampling technique is SMOTE and the best performing feature ranking method is OneR.**

***Index Terms*—Anti-Patterns, Empirical Software Engineering, Feature Selection, Imbalanced Learning, Machine Learning, Predictive Modeling, Service Oriented Architecture, Software Analytics, Source Code Analysis, Source Code Metrics, Web Services.**

## I. Research Motivation and Aim

Web Services are used for building distributed software systems based on the Service Oriented Architecture (SOA) paradigm [1][2][3]. A complex distributed system can be developed by combining web-services from third-party providers. A web-service providing a certain functionality defined by the web service interface and description is published by a web-service provider which can be discovered and invoked by a distributed application [1][2][3]. Web services are developed based on pre-defined standards to support interoperability. Similar to object oriented systems, web services also suffers from anti-patterns due to bad programming practises, design and implementation [4][5][6][7][8][9][10]. Anti-patterns in web-services leads to maintenance and evolution related problems. There are several web-services anti-patterns [4][5][6][7][8][9][10]. The five web-services anti-patterns which we consider in the study presented this paper are: GOWS : God object Web service, FGWS : Fine grained Web service, DWS : Data Web service, CWS : Chatty Web service, AWS: Ambiguous Web service. For example, the GOWS anti-pattern results because of implementing a lot of methods related to multitude of business and technical abstractions into a single service. AWS anti-pattern has implications in reusability and discoverability of web-services due to ambiguous names. Previous research shows that anti-patterns in web-services is a serious and important problem and there are lack of approaches for automatically detecting web-service anti-patterns from their source code and Web Service Definition Language (WSDL) documents [6][7][8][9][10]. For example, Ouni et al. mention that automatic detection of anti-patterns in web services is relatively unexplored [6]. Their analysis reveals that there are lack of approaches for anti-pattern detection in web-services and research in this direction is still in its infancy [6]. The first approach for web-service anti-pattern detection was proposed by Ouni et al. in the year $2015$ [6]. Similarly, Palma et al. also mention that extensive usage and importance of web-services, there are no or very less work on automated approaches for web-services anti-pattern detection [8]. They emphasize the need for building automated approaches for web-services anti-pattern identification to improve web-services maintenance and evolution [8].

The work presented in this paper is motivated by the need to build tools and techniques for automatic detection of anti-patterns in web-services. In particular, our motivation is to investigate the application of machine learning based techniques for building predictive models from source code metrics as features for the task of detecting web-services anti-patterns. The specific research aims of the work presented in this paper are the following:

1) To investigate the application of different machine learning classification algorithms, feature selection and ranking techniques and data sampling techniques for the task of web-services anti-pattern detection using source code metrics as features. To examine the correlation between several object oriented metrics derived from the source code implementing a web-service and presence of common web-services anti-patterns.
2) To conduct an in-depth empirical analysis on real-world web-services dataset and investigate the performance of the machine learning algorithms and object oriented source code metrics based features for predicting web-

services anti-patterns. To examine the relative performance of various classification algorithms, feature ranking techniques, data sampling methods to encounter the class imbalance problem and object oriented source code metrics using Area Under an ROC Curve (AUC) and statistical hypothesis based testing approaches.

## II. Related Work and Research Contributions

Ouni et al. propose a search based approach for web-service anti-pattern detection [6][7]. Their approach is based on Genetic Programming (GP) which generates anti-pattern identification rules from examples of web-service anti-patterns [6][7]. Mateos et al. conduct a research on web-service anti-patterns and conclude that there is a correlation between object-oriented metrics in the source code implementing a web-service and presence of anti-patterns in the Web Service Description Language (WSDL) of a web-service [11]. Their research reveals several bad practises in public WSDL documents which gets manifested in the automatically generated WSDL documents because of poor implementation practises in the source code [11]. Their work consists of investigating the feasibility of web-service anti-patterns using object oriented metrics [11]. Coscia et al. present an empirical analysis on the influence of object oriented source code metrics driven code refactoring on the WSDL file and document quality [12]. Their work is based on the premise that there is a correlation between the occurrences of web-service anti-patterns manifested in WSDL and several source code metrics implementing the web service [12].

Palma et al. propose an approach for detecting anti-patterns in web-services [8][9]. They specify $10$ web-services anti-pattern and conduct experiments belonging to the weather and financial domain. Their detection algorithm is based on identifying the relevant properties of web-services specific anti-patterns [8][9]. Moha et al. present a method for web-services anti-pattern detection which is able to identify anti-patterns such as Multi Service and Tiny Service. They conduct experiments on a service based system in the domain of home automation are able to achieve a precision of around $90\%$ [10]. Nayrolles et al. mention that there is no tool support which exists for the detection of anti-patterns in web-services and service oriented systems [13]. They describe a prototype tool called as SODA which is based on specifying anti-patterns in the form of rule cards and then generating detection algorithms which conforms to the web-services anti-pattern specifications [13].

**Research Contributions :** In context to existing work, the study presented in this paper makes two main research contributions. The first novel and unique research contribution of our work is the application of $18$ object oriented source code metrics as features or independent variables, $8$ classification algorithms, $5$ feature selection and ranking technique and $4$ data sampling techniques to counter the class imbalance problem for the task of detecting $5$ different types of web-service anti-patterns. The second contribution of our work to the body of knowledge on web-service anti-pattern detection is an in-depth empirical analysis of the proposed techniques and frameworks on a publicly available dataset consisting of $226$ web-services across several domains. We conduct a series of experiments investigating the performance of several machine learning parameter combinations and examine their relative effectiveness using AUC based performance metrics and statistical hypothesis based testing.

## III. Solution Approach and Research Framework

Figure 1 illustrates our research framework and solution approach. As shown in Figure 1, our approach consists of multiple steps. The first step consists of computing the features of attributes from the $226$ web-services. The features

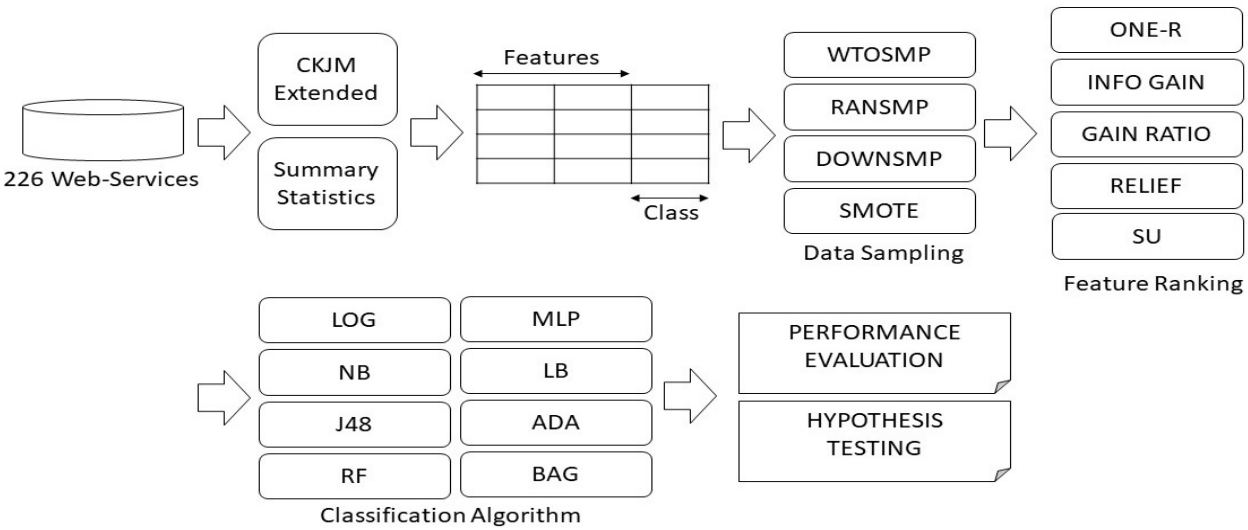


Fig. 1: Research Framework and Solution Approach

are computed by obtaining the object oriented metrics using the CKJM Extended Tool and then computing summary statistics from it. The dataset has imbalanced class distribution and hence we apply several data sampling techniques to encounter the data imbalance problem. We apply several feature ranking techniques to identify informative features and remove irrelevant or redundant features (refer to Figure 1). As shown in Figure 1, we apply different classification algorithms to build predictive models and perform a comparison between the classifiers. Finally, we evaluate the performance of the classifiers using performance evaluation metrics and draw conclusions based on the results of the statistical hypothesis testing.

**Source Code Metrics:** We compute the object oriented source code metrics using the CKJM extended tool[1]. The object oriented metrics are computed at the web-service level for the Java source code implementing the web-services. The CKJM extended tool computes several Chidamber and Kemerer (defined in [14][15]) Java Metrics. The metrics are computed by processing the bytecode of the compiled Java files. We compute a total of 18 size and structure metrics such as WMC: Weighted methods per class, DIT: Depth of Inheritance Tree, NOC: Number of Children, CBO: Coupling between object classes, LCOM3: Lack of cohesion in methods Henderson-Sellers version, LCO: Lines of Code, DAM: Data Access Metric. A complete list of the metrics with their abbreviations and expanded-form is provided in the CKJM tool documentation. We compute the aggregate and summary values at the web-service level once the 18 object oriented metrics values are computed for every class in the web service. The values which we compute at the web service level are: minimum, maximum, mean, median, first quartile, third quartile, standard deviation, skewness and kurtosis. We compute these values for all the 18 metrics except in cases where such values cannot be computed. These summary values obtained from the data are the attributes for the classification problem.

**Data Sampling Techniques to Learn from Imbalanced Data:** Our experimental dataset has an imbalanced class distribution. The number of web services containing a particular anti-pattern is significantly lower (5% to 10% of the dataset) than the number of web services without the anti-pattern. Dealing with an imbalanced data containing unequal instanced of different classes is one the technical challenges of our problem. We use three different types of sampling techniques to overcome the unbalanced 2-class machine learning problem. We apply a technique called as Downsampling (DOWNSMP) which creates a balanced dataset such that the number of instanced of the minority class is the same as the number of instances of the majority class [16][17]. Another sapling technique which we use is RANSMP which is a nave oversampling approach. RANSMP consists of generating new and more samples for the class which is under-represented and is in minority. We apply a technique called as SMOTE (Synthetic Minority Over-sampling Technique) which is based on the combination of over-sampling the minority class and over-sampling the majority class [16].

**Feature Ranking and Selection:** Liu et al. present a survey of several feature selection algorithms [18]. We apply five different feature selection approaches (refer to Figure 1): OneR - FR1, Information Gain (IG) - FR2, Gain Ratio (GR) - FR3, RELEIF (RF) - FR4 and Symmetric Uncertainty (SU) - FR5. We also consider a case with no feature ranking (AM : All Metrics).

**Learning Algorithms:** As illustrated in Figure 1, we apply eight different types of classifiers or learning algorithms: J48 : Decision Tree, RF : Random Forest, LB : Logit Boost, ADA: AdaBoost, BAG : Bagging, MLP : Multilayer Perceptron, NB : Naive Bayes, LOG: Logistic regression. Lim et al. present a comparison of the predictive accuracy, complexity and training time of 33 learning algorithms [19]. We apply both classical algorithms (such as Decision Tree and Nave Bayes) as well as relatively modern and new statistical algorithms to examine the generalizability of our empirical results and findings and identify the best performing classification algorithm.

**Performance Evaluation :** Effectiveness of a learning algorithm or classifier can be measured using several different performance metrics. Sokolova present a detailed comprehensive analysis of the performance measures for predictive modelling and classification task [20]. We use metrics such as Area under the ROC (Receiver Operating Characteristics) Curve (AUC), f-measure and accuracy to measure the performance of the learning algorithm. We use f-measure because the performance measure incorporates both precision and recall in its calculation. We use AUC because it has an a property or characterises of being insensitive to changes in the class distribution [20]. We apply 10 fold cross validation method for removing bias while evaluating the predictive models.

## IV. Experimental Dataset

We conduct experiments on 226 publicly available web-services[2] shared on GitHub by the authors of the paper by Ouni et al. [6]. We conduct analysis on publicly available dataset so that our experimental results can be replicated and used for benchmarking and comparison. Juristo et al. and Kitchenham et al. work on experimental software engineering emphasizes the need of easy replication for increasing the reliability and validity of outcome in experiments [21][22]. Since our approach presented in this paper is the first work on applying several machine learning algorithms, feature ranking techniques and data sampling techniques, our objective is to

[1] gromit.iiar.pwr.wroc.pl/p_inf/ckjm/

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

facilitate researchers to reproduce our work and use our results as baseline. The web-services used in our experiments belong to several domains such as education, media, weather, science, shipping, financial, search and travel. The web-service data is of high quality as the anti-patters are manually validated by the authors of the paper [6] who shared the dataset online on GitHub. The web-services belonging to the financial domain had the maximum number of anti-patterns.

Table I shows the number and percentages of 5 anti-patterns in our experimental dataset consisting of 226 web-services. We observe that the GOWS anti-pattern is present in 21 out of 226 web-services. There are 5.75% of web-services containing FGWS anti-pattern. The DWS, CWS and AWS anti-pattern are present in 14, 21 and 24 web-services respectively. As can be inferred from the Table I, the dataset has an imbalanced class distribution with respect to each anti-pattern. The dataset has an intrinsic imbalance characteristics due the nature of the dataspace. The number of web-services having a particular anti-pattern will naturally be in minority but not rare. Table I shows that an anti-pattern is minority (5% to 10%) but not rare (for example, lower than 1%). Also our dataset is small in-terms of the sample size and as shown in Table I, our problem is a machine learning problem consisting of a small sample size and imbalanced data [6][12] [11].

TABLE I: Number and Percentages of 5 Anti-Patterns in 226 Web-Services

| | **GOWS** | **FGWS** | **DWS** | **CWS** | **AWS** |
|---|---|---|---|---|---|
| # **AP** | 21 | 13 | 14 | 21 | 24 |
| % **AP** | 9.29 | 5.75 | 6.19 | 9.29 | 10.62 |

## V. Experimental Results and Findings

We follow a research methodology consisting of formulation of *four grounded research questions* guiding our scientific study. We apply the Sandberg et al. approach of formulating research questions based on existing work, literature and theories in our area [23]. Our main aim is to formulate research questions (RQ) and provide answers to them for facilitating development of tools and creation of methods for automatically detecting web-services anti-patterns through static analysis of source code.

**RQ 1:** Is there a statistically significant difference between the aggregate values of source code metrics computed at the web-service level between the two groups consisting of web-services containing a given anti-pattern and the web-services not containing the anti-pattern?

Our dataset consists of 226 web-services and our focus is on detecting five types of anti-patterns. For each of the five anti-patterns, we create two dataset. One dataset consists of web-services containing the anti-pattern and the other dataset consists of web-services which does not contain the anti-pattern. For example, for the GOWS anti-pattern, one dataset consists of 21 web-services and the other dataset consists of 105 web-services. For the FGWS anti-pattern, one dataset consists of 13 web-services and the other dataset consists of 213 web-services. In this experiment, the two datasets or the two samples are independent as there are no common web-services between the two datasets. We compute 148 values for each of the web-service consisting of aggregate and summary values like mean, median, standard deviation, Q1 and Q3 for each of the 18 object-oriented source code metrics. The number of values is 148 as there are some aggregate and

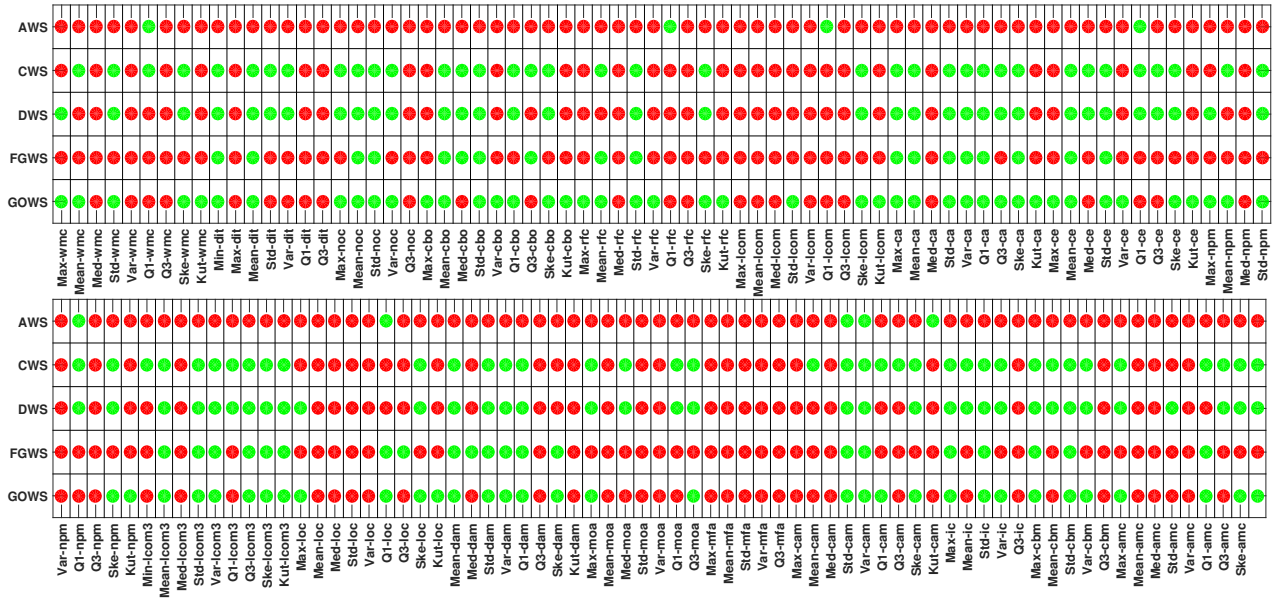


Fig. 2: Hypothsis Testing Results - based on p-value of source code metrics

summary values which cannot be computed (or have 0 or null values) for some of the metrics. We state the null hypothesis $H_0$ as a claim that there is no statistically significant difference in the mean value of the feature (independent variable) for the two datasets. The null hypothesis if accepted means that there is no influence of the metric or the independent variable on the dependent variable (dependent variable representing whether the web service contains the anti-pattern). The alternate hypothesis in this experiment $H_a$ is the mathematical opposite of the null hypothesis $H_0$ which states that there is a statistically significant difference between the means of the metric value for the two dataset and hence the independent variable is potentially a predictor of the dependent variable.

We apply the Wilcoxon rank-sum test for hypothesis testing. Our objective is to determine if the null hypothesis should be accepted or rejected. We use the commonly used Wilcoxon rank-sum test methodology because the approach does not assume that the population follows a normal distribution. We use Wilcoxon rank-sum test as it is much more efficient than other types of parametric tests which assumes that the data (population and samples being analysed or compared) is normally distributed. Hence we do not make any parametric assumptions while conducting the null test. We set the significance level as $0.05$ and compare the p-value with $0.05$. We reject the null hypothesis ($H_0$) if the computed p-value is less than the cut-off value of $0.05$. We reject the null hypothesis at a p-value of less than $0.05$ as our inference is that the difference between the mean of the two groups is not because of chance and instead the difference in the mean is statistically significant. Figure 2 displays an illustration which reveals the cases for which the null hypothesis is accepted and rejected. A red dot denotes that the null hypothesis is rejected and a green dot denotes that the null hypothesis is accepted. Figure 2 reveals several metrics for which the number of red dots are $\geq 3$. Our analysis reveals that there are $91$ metrics having red dots $\geq 3$. We set the cutoff (a heuristic) as number of red dots $\geq 3$ and use $91$ metrics as one set of features (Reduce Metric - RM) for the learning algorithm. We discard the remaining $57$ metrics as non-informative or irrelevant based on the statistical significance test results. We conduct experiments with both the sets: AM (All $148$ metrics) and RM (A reduced set of $91$ metrics). Following is the answer to RQ1.

**Answer RQ1:** *Statistical significance test reveals that out of the* $148$ *source code metrics as features,* $91$ *metrics influences anti-pattern and there is a relation between* $91$ *metrics and occurrence of the five types of anti-patterns.*

**Feature Ranking Detailed Results :** Table II displays the output of all the feature ranking technique. Table II displays the rank of every feature across all the anti-patterns for every feature ranking technique. We select (and highlight using shaded cell in Table II) top $k$ attributes which is a logarithmic function of the total number of features. Table II reveals the ranking of each feature where the value in the cell represents the rank of the given feature. Table II shows that the rank of the feature SKE_RFC is $1$ for the GOWS and Information Gain Combination. We observe that there are some feature which are ranked highly across anti-patterns type and across feature selection techniques. We also notice that there are some attributes which are ranked higher for a particular anti-pattern or ranking technique. Table II shows that different feature selection techniques produces different output. An interesting finding is that few source code metrics have a higher rank for some anti-patterns while the same metrics is ranked lower for other anti-pattern within the context of a particular feature selection technique.

**Detailed Results of** $960$ **Predictive Models:** Tables III, IV and V presents a detailed analysis and result of all the experimental executions for every combination of the experimental parameters. We are using $4$ data sampling techniques, $8$ learning algorithms $5$ feature ranking techniques (total $6$ after including the all metrics case) and $5$ different types of anti-patterns. Thus the total number of unique classification models built by us are $4X8X6X5 = 960$. We use $10$ fold cross validation while evaluating the accuracy of each classifier. Table III shows that the accuracy of LB classifier for the case of Reduced Metrics, WTOSMP sampling technique and AWS anti-pattern is $88.05$. Table III reveals a high f-measure value ($0.94$) for the NB and MLP classifier for the case of AWS anti-pattern and WTOSMP sampling technique. Table IV presents detailed results for the all-metrics case. Table IV shows that the f-measure value for NB and LOG classifier is $0.93$ for the case of WTOSMP sampling technique and GOWS anti-pattern. From Table IV, we infer that the accuracy of RF classifier is above $97\%$ for SMOTE and FGWS and DWS anti-pattern. Table V provided detailed result across all feature ranking methods. From Table V, we infer that the f-measure value of the RF classifier is above $0.9$ for several combinations of data sampling technique and feature ranking techniques.

**Distribution Comparison and Visualization using Boxplot:** Tables VI, VII and VIII shows the descriptive statistics of the performance of the various combinations of data sampling, feature ranking and learning algorithms. Figures 3, 4 and 5 shows the boxplots for comparing the degree of dispersion, interquartile range, outliers and skewness in the accuracy, f-measure and AUC values for all the combinations of techniques. The red line in the boxplot of Figures 3, 4 and 5 marks the median point of the data (mid-point) and the red line thus divides the box into two segments. Figure 3 shows that the median AUC value of NB and RF is higher than the corresponding values for other learning algorithms such as LOG, J48, MLP, LB, ADA and BAG. Figure 3 shows that the median value for LOG is the lowest in comparison to other classification algorithms. From the distributional characteristics of data shown the bog-plots of Figure 4, we infer that the inter-quartile range representing the middle box signifying the middle $50\%$ of the values of the given variable range for LOG and BAG is more than the corresponding

TABLE II: Feature Rank Across All Anti-Patterns and Across All Feature Ranking Techniques

| | Info Gain | | | | | Gain Ratio | | | | | OneR | | | | | Chi Square | | | | | SU |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | GOWS | FGWS | DWS | CWS | AWS | GOWS | FGWS | DWS | CWS | AWS | GOWS | FGWS | DWS | CWS | AWS | GOWS | FGWS | DWS | CWS | AWS | All |
| Med_wmc | 23 | 58 | 38 | 44 | 58 | 21 | 58 | 40 | 44 | 58 | 44 | 54 | 22 | 51 | 58 | 21 | 58 | 43 | 49 | 58 | 58 |
| Q3_wmc | 7 | 27 | 12 | 2 | 21 | 16 | 27 | 15 | 10 | 21 | 8 | 21 | 2 | 29 | 29 | 16 | 27 | 15 | 1 | 21 | 21 |
| Kut_wmc | 46 | 18 | 31 | 55 | 19 | 46 | 18 | 33 | 56 | 19 | 39 | 19 | 15 | 19 | 28 | 46 | 18 | 34 | 50 | 19 | 19 |
| Max_dit | 34 | 15 | 33 | 54 | 17 | 26 | 15 | 23 | 52 | 17 | 53 | 17 | 16 | 26 | 27 | 26 | 16 | 20 | 55 | 17 | 17 |
| Q1_dit | 26 | 17 | 1 | 9 | 18 | 23 | 17 | 2 | 6 | 18 | 42 | 18 | 42 | 46 | 23 | 23 | 19 | 2 | 9 | 18 | 18 |
| Q3_dit | 36 | 28 | 11 | 43 | 22 | 33 | 28 | 14 | 34 | 22 | 51 | 22 | 53 | 38 | 24 | 33 | 28 | 14 | 32 | 22 | 22 |
| Ske_dit | 41 | 26 | 3 | 11 | 23 | 34 | 26 | 1 | 5 | 23 | 25 | 23 | 48 | 58 | 25 | 36 | 26 | 3 | 10 | 23 | 23 |
| Kut_dit | 40 | 29 | 2 | 10 | 24 | 35 | 29 | 3 | 4 | 24 | 22 | 24 | 49 | 57 | 40 | 37 | 29 | 1 | 8 | 24 | 24 |
| Mean_noc | 20 | 11 | 15 | 15 | 25 | 25 | 4 | 17 | 2 | 25 | 48 | 25 | 17 | 10 | 20 | 25 | 10 | 18 | 5 | 25 | 25 |
| Med_noc | 57 | 19 | 54 | 57 | 28 | 50 | 19 | 54 | 54 | 28 | 18 | 28 | 18 | 23 | 8 | 50 | 17 | 54 | 57 | 28 | 28 |
| Std_noc | 3 | 31 | 21 | 39 | 27 | 9 | 31 | 25 | 38 | 27 | 41 | 27 | 3 | 6 | 42 | 8 | 31 | 24 | 34 | 27 | 27 |
| Q1_noc | 55 | 24 | 45 | 35 | 26 | 55 | 24 | 38 | 28 | 26 | 19 | 26 | 19 | 35 | 2 | 55 | 24 | 38 | 26 | 26 | 26 |
| Ske_noc | 12 | 32 | 30 | 49 | 20 | 12 | 32 | 32 | 49 | 20 | 7 | 20 | 52 | 49 | 11 | 11 | 32 | 33 | 43 | 20 | 20 |
| Max_cbo | 42 | 7 | 49 | 41 | 15 | 43 | 5 | 50 | 42 | 15 | 40 | 15 | 12 | 22 | 16 | 41 | 7 | 50 | 48 | 15 | 15 |
| Med_cbo | 16 | 30 | 29 | 34 | 5 | 14 | 30 | 22 | 36 | 5 | 58 | 5 | 54 | 24 | 13 | 14 | 30 | 19 | 44 | 5 | 5 |
| Q3_cbo | 2 | 4 | 14 | 48 | 14 | 11 | 9 | 18 | 45 | 14 | 45 | 14 | 13 | 56 | 4 | 12 | 2 | 17 | 37 | 14 | 14 |
| Q1_rfc | 4 | 25 | 9 | 6 | 6 | 13 | 25 | 11 | 12 | 6 | 46 | 6 | 34 | 1 | 41 | 13 | 25 | 10 | 3 | 6 | 6 |
| Ske_rfc | 1 | 16 | 7 | 8 | 4 | 1 | 16 | 9 | 35 | 4 | 1 | 4 | 58 | 55 | 14 | 1 | 13 | 9 | 14 | 4 | 4 |
| Kut_rfc | 19 | 10 | 13 | 20 | 2 | 24 | 3 | 16 | 22 | 2 | 56 | 2 | 56 | 47 | 15 | 24 | 9 | 16 | 22 | 2 | 2 |
| Max_lcom | 9 | 8 | 6 | 5 | 3 | 2 | 14 | 6 | 8 | 3 | 4 | 3 | 46 | 42 | 44 | 2 | 14 | 7 | 6 | 3 | 3 |
| Mean_lcom | 10 | 9 | 5 | 4 | 7 | 3 | 13 | 7 | 7 | 7 | 3 | 7 | 44 | 40 | 45 | 3 | 15 | 6 | 7 | 7 | 7 |
| Med_lcom | 45 | 23 | 53 | 53 | 8 | 44 | 23 | 52 | 51 | 8 | 47 | 8 | 9 | 25 | 18 | 45 | 22 | 53 | 54 | 8 | 8 |
| Std_lcom | 49 | 22 | 57 | 58 | 9 | 56 | 22 | 56 | 58 | 9 | 23 | 9 | 11 | 20 | 19 | 56 | 23 | 57 | 58 | 9 | 9 |
| Var_lcom | 11 | 33 | 10 | 27 | 10 | 10 | 33 | 12 | 25 | 10 | 11 | 10 | 6 | 53 | 17 | 10 | 33 | 12 | 27 | 10 | 10 |
| Ske_lcom | 21 | 12 | 18 | 7 | 13 | 27 | 2 | 5 | 3 | 13 | 49 | 13 | 35 | 16 | 12 | 27 | 8 | 5 | 4 | 13 | 13 |
| Min_ca | 8 | 35 | 22 | 40 | 12 | 8 | 35 | 24 | 37 | 12 | 12 | 12 | 45 | 9 | 56 | 7 | 35 | 25 | 35 | 12 | 12 |
| Mean_ca | 54 | 36 | 46 | 32 | 11 | 51 | 36 | 4 | 1 | 11 | 16 | 11 | 55 | 2 | 10 | 51 | 36 | 13 | 11 | 11 | 11 |
| Std_ca | 5 | 50 | 20 | 33 | 29 | 5 | 50 | 21 | 33 | 29 | 6 | 29 | 50 | 48 | 3 | 5 | 50 | 23 | 31 | 29 | 29 |
| Kut_ca | 22 | 6 | 37 | 42 | 30 | 20 | 1 | 39 | 43 | 30 | 43 | 30 | 21 | 54 | 1 | 20 | 6 | 42 | 47 | 30 | 30 |
| Max_ce | 47 | 52 | 51 | 52 | 31 | 52 | 52 | 53 | 55 | 31 | 13 | 31 | 8 | 28 | 5 | 52 | 52 | 51 | 51 | 31 | 31 |
| Med_ce | 6 | 51 | 16 | 1 | 35 | 15 | 51 | 13 | 9 | 35 | 9 | 35 | 1 | 7 | 6 | 15 | 51 | 11 | 2 | 35 | 35 |
| Q1_ce | 33 | 1 | 34 | 29 | 50 | 38 | 6 | 36 | 31 | 50 | 24 | 56 | 10 | 52 | 7 | 42 | 1 | 36 | 41 | 50 | 50 |
| Ske_ce | 24 | 13 | 43 | 25 | 49 | 18 | 12 | 46 | 26 | 49 | 54 | 57 | 47 | 3 | 50 | 18 | 11 | 44 | 29 | 49 | 49 |
| Kut_ce | 25 | 14 | 42 | 26 | 48 | 17 | 11 | 44 | 27 | 48 | 55 | 58 | 51 | 4 | 49 | 19 | 12 | 46 | 30 | 48 | 48 |
| Max_npm | 53 | 5 | 50 | 31 | 51 | 53 | 10 | 43 | 29 | 51 | 27 | 48 | 20 | 37 | 9 | 53 | 5 | 39 | 28 | 51 | 51 |
| Mean_npm | 58 | 53 | 32 | 13 | 47 | 48 | 53 | 34 | 19 | 47 | 37 | 47 | 14 | 44 | 57 | 48 | 53 | 35 | 23 | 47 | 47 |
| Med_npm | 35 | 57 | 41 | 30 | 52 | 28 | 57 | 45 | 32 | 52 | 50 | 51 | 43 | 39 | 21 | 28 | 57 | 45 | 33 | 52 | 52 |
| Kut_npm | 48 | 56 | 58 | 56 | 54 | 58 | 56 | 55 | 53 | 54 | 38 | 49 | 32 | 36 | 22 | 58 | 56 | 58 | 56 | 54 | 54 |
| Max_lcom3 | 13 | 55 | 4 | 21 | 57 | 7 | 55 | 10 | 21 | 57 | 52 | 52 | 37 | 5 | 33 | 6 | 55 | 4 | 13 | 57 | 57 |
| Med_lcom3 | 51 | 54 | 55 | 50 | 56 | 49 | 54 | 58 | 48 | 56 | 36 | 53 | 29 | 30 | 35 | 49 | 54 | 56 | 52 | 56 | 56 |
| Var_lcom3 | 38 | 48 | 47 | 37 | 55 | 40 | 48 | 49 | 40 | 55 | 35 | 55 | 28 | 45 | 36 | 39 | 48 | 49 | 46 | 55 | 55 |
| Q1_lcom3 | 39 | 47 | 48 | 36 | 53 | 41 | 47 | 48 | 41 | 53 | 34 | 50 | 27 | 43 | 37 | 40 | 47 | 48 | 45 | 53 | 53 |
| Q3_lcom3 | 43 | 46 | 52 | 51 | 46 | 39 | 46 | 51 | 50 | 46 | 33 | 46 | 23 | 34 | 38 | 44 | 46 | 52 | 53 | 46 | 46 |
| Max_loc | 17 | 39 | 35 | 38 | 45 | 22 | 39 | 37 | 39 | 45 | 10 | 45 | 5 | 8 | 54 | 22 | 39 | 37 | 36 | 45 | 45 |
| Q3_loc | 44 | 38 | 44 | 28 | 44 | 45 | 38 | 47 | 30 | 44 | 28 | 44 | 25 | 32 | 39 | 43 | 38 | 47 | 38 | 44 | 44 |
| Q1_dam | 56 | 3 | 40 | 46 | 36 | 54 | 7 | 41 | 47 | 36 | 29 | 36 | 24 | 33 | 34 | 54 | 4 | 41 | 40 | 36 | 36 |
| Q3_dam | 52 | 2 | 39 | 47 | 34 | 57 | 8 | 42 | 46 | 34 | 30 | 34 | 26 | 18 | 32 | 57 | 3 | 40 | 39 | 34 | 34 |
| Min_moa | 37 | 37 | 36 | 22 | 32 | 42 | 37 | 35 | 20 | 32 | 31 | 32 | 57 | 17 | 55 | 38 | 37 | 26 | 21 | 32 | 32 |
| Mean_moa | 32 | 21 | 27 | 24 | 33 | 37 | 20 | 27 | 24 | 33 | 32 | 33 | 30 | 31 | 31 | 34 | 20 | 29 | 25 | 33 | 33 |
| Std_moa | 27 | 40 | 28 | 18 | 37 | 29 | 40 | 31 | 14 | 37 | 26 | 37 | 39 | 13 | 52 | 29 | 40 | 30 | 18 | 37 | 37 |
| Var_moa | 29 | 41 | 26 | 16 | 38 | 31 | 41 | 26 | 17 | 38 | 17 | 38 | 41 | 14 | 47 | 31 | 41 | 32 | 19 | 38 | 38 |
| Q3_moa | 31 | 20 | 25 | 23 | 39 | 36 | 21 | 30 | 23 | 39 | 15 | 39 | 33 | 21 | 26 | 35 | 21 | 31 | 24 | 39 | 39 |
| Kut_moa | 28 | 42 | 24 | 17 | 40 | 30 | 42 | 29 | 15 | 40 | 14 | 40 | 40 | 11 | 46 | 30 | 42 | 28 | 16 | 40 | 40 |
| Min_mfa | 30 | 45 | 23 | 19 | 43 | 32 | 45 | 28 | 16 | 43 | 21 | 43 | 38 | 12 | 48 | 32 | 45 | 27 | 17 | 43 | 43 |
| Mean_mfa | 18 | 44 | 19 | 14 | 42 | 19 | 44 | 20 | 18 | 42 | 57 | 42 | 4 | 15 | 53 | 17 | 44 | 22 | 20 | 42 | 42 |
| Q3_mfa | 50 | 43 | 56 | 45 | 41 | 47 | 43 | 57 | 57 | 41 | 20 | 41 | 31 | 27 | 30 | 47 | 43 | 55 | 42 | 41 | 41 |
| Kut_mfa | 15 | 49 | 8 | 3 | 16 | 6 | 49 | 8 | 11 | 16 | 5 | 16 | 36 | 41 | 51 | 9 | 49 | 8 | 12 | 16 | 16 |
| Min_cam | 14 | 34 | 17 | 12 | 1 | 4 | 34 | 19 | 13 | 1 | 2 | 1 | 7 | 50 | 43 | 4 | 34 | 21 | 15 | 1 | 1 |

TABLE III: Performance Parameters (Reduced Metrics)

| | | Accuracy | | | | | | | | F-Measure | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | J48 | RF | LB | ADA | BAG | MLP | NB | LOG | J48 | RF | LB | ADA | BAG | MLP | NB | LOG |
| WTOSMP | GOWS | 89.38 | 91.15 | 89.38 | 90.27 | 90.27 | 91.59 | 86.28 | 90.71 | 0.94 | 0.95 | 0.94 | 0.95 | 0.95 | 0.95 | 0.92 | 0.95 |
| WTOSMP | FGWS | 94.69 | 95.13 | 94.69 | 93.81 | 94.25 | 93.36 | 74.78 | 93.81 | 0.97 | 0.97 | 0.97 | 0.97 | 0.97 | 0.97 | 0.85 | 0.97 |
| WTOSMP | DWS | 94.69 | 94.25 | 94.25 | 93.81 | 93.81 | 95.13 | 86.73 | 92.92 | 0.97 | 0.97 | 0.97 | 0.97 | 0.97 | 0.97 | 0.93 | 0.96 |
| WTOSMP | CWS | 91.59 | 92.92 | 90.71 | 88.94 | 90.71 | 91.15 | 83.19 | 89.82 | 0.95 | 0.96 | 0.95 | 0.94 | 0.95 | 0.95 | 0.9 | 0.95 |
| WTOSMP | AWS | 89.38 | 89.38 | 88.05 | 88.94 | 89.38 | 89.38 | 88.5 | 89.38 | 0.94 | 0.94 | 0.94 | 0.94 | 0.94 | 0.94 | 0.94 | 0.94 |
| RANSMP | GOWS | 77 | 81.33 | 77 | 80 | 81.33 | 81 | 79 | 79.67 | 0.82 | 0.86 | 0.83 | 0.85 | 0.85 | 0.85 | 0.83 | 0.85 |
| RANSMP | FGWS | 68.65 | 74.59 | 71.35 | 72.43 | 68.65 | 65.41 | 64.86 | 65.41 | 0.76 | 0.81 | 0.78 | 0.8 | 0.78 | 0.74 | 0.69 | 0.73 |
| RANSMP | DWS | 77.5 | 79.5 | 82 | 80.5 | 79.5 | 82 | 83 | 83.5 | 0.83 | 0.84 | 0.86 | 0.85 | 0.84 | 0.86 | 0.86 | 0.87 |
| RANSMP | CWS | 75.33 | 77 | 74.67 | 77 | 77 | 75.67 | 78.33 | 76.67 | 0.81 | 0.83 | 0.81 | 0.83 | 0.83 | 0.81 | 0.82 | 0.82 |
| RANSMP | AWS | 62.61 | 62.61 | 61.16 | 60.58 | 60 | 55.94 | 47.83 | 55.36 | 0.77 | 0.73 | 0.72 | 0.73 | 0.72 | 0.67 | 0.58 | 0.7 |
| DOWNSMP | GOWS | 79.04 | 80.55 | 78.79 | 78.3 | 80.52 | 79.6 | 77.7 | 78.77 | 0.81 | 0.83 | 0.81 | 0.81 | 0.83 | 0.82 | 0.8 | 0.82 |
| DOWNSMP | FGWS | 75.83 | 80.99 | 77.41 | 76.67 | 74.52 | 73.27 | 64.04 | 72.37 | 0.82 | 0.86 | 0.83 | 0.83 | 0.82 | 0.8 | 0.7 | 0.8 |
| DOWNSMP | DWS | 77.16 | 77.9 | 77.83 | 78.61 | 77.6 | 81.77 | 73.24 | 78.24 | 0.82 | 0.83 | 0.83 | 0.84 | 0.83 | 0.86 | 0.77 | 0.84 |
| DOWNSMP | CWS | 68.05 | 75.66 | 72.77 | 70.83 | 74.53 | 70.02 | 78.63 | 73.21 | 0.7 | 0.77 | 0.75 | 0.73 | 0.76 | 0.73 | 0.8 | 0.76 |
| DOWNSMP | AWS | 52 | 52.77 | 54.09 | 52.78 | 55.06 | 50.29 | 49.23 | 49.01 | 0.55 | 0.56 | 0.54 | 0.54 | 0.55 | 0.51 | 0.46 | 0.53 |
| SMOTE | GOWS | 88.57 | 92.5 | 90 | 88.21 | 90.36 | 90 | 81.07 | 87.86 | 0.92 | 0.95 | 0.93 | 0.91 | 0.93 | 0.93 | 0.86 | 0.92 |
| SMOTE | FGWS | 94.98 | 94.59 | 90.35 | 81.08 | 93.44 | 85.71 | 61 | 79.54 | 0.97 | 0.97 | 0.94 | 0.88 | 0.96 | 0.91 | 0.71 | 0.88 |
| SMOTE | DWS | 95.04 | 95.04 | 91.98 | 91.98 | 93.51 | 94.27 | 87.02 | 87.4 | 0.97 | 0.97 | 0.95 | 0.95 | 0.96 | 0.96 | 0.91 | 0.92 |
| SMOTE | CWS | 91.46 | 92.88 | 86.48 | 83.63 | 92.53 | 89.68 | 82.21 | 82.92 | 0.94 | 0.95 | 0.9 | 0.89 | 0.95 | 0.93 | 0.86 | 0.88 |
| SMOTE | AWS | 80.65 | 86.38 | 74.19 | 70.97 | 82.08 | 73.84 | 58.42 | 63.44 | 0.86 | 0.9 | 0.82 | 0.81 | 0.87 | 0.82 | 0.66 | 0.77 |

TABLE IV: Performance Parameters (All Metrics)

| | | Accuracy | | | | | | | | F-Measure | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | J48 | RF | LB | ADA | BAG | MLP | NB | LOG | J48 | RF | LB | ADA | BAG | MLP | NB | LOG |
| WTOSMP | GOWS | 92.04 | 92.04 | 92.92 | 92.48 | 93.36 | 92.48 | 88.05 | 87.61 | 0.96 | 0.96 | 0.96 | 0.96 | 0.96 | 0.96 | 0.93 | 0.93 |
| WTOSMP | FGWS | 93.81 | 95.58 | 95.13 | 94.69 | 94.25 | 94.69 | 61.06 | 88.5 | 0.97 | 0.98 | 0.97 | 0.97 | 0.97 | 0.97 | 0.74 | 0.94 |
| WTOSMP | DWS | 95.13 | 97.35 | 96.46 | 96.02 | 94.69 | 95.58 | 95.58 | 93.36 | 0.97 | 0.99 | 0.98 | 0.98 | 0.97 | 0.98 | 0.98 | 0.96 |
| WTOSMP | CWS | 91.59 | 92.48 | 92.48 | 92.92 | 92.04 | 92.04 | 81.42 | 85.84 | 0.95 | 0.96 | 0.96 | 0.96 | 0.96 | 0.96 | 0.89 | 0.92 |
| WTOSMP | AWS | 87.17 | 88.05 | 83.63 | 89.38 | 89.38 | 86.28 | 57.08 | 79.65 | 0.93 | 0.94 | 0.91 | 0.94 | 0.94 | 0.92 | 0.71 | 0.88 |
| RANSMP | GOWS | 80 | 83.67 | 81 | 80.33 | 84.33 | 82.67 | 82.67 | 74.67 | 0.85 | 0.88 | 0.86 | 0.85 | 0.88 | 0.87 | 0.86 | 0.81 |
| RANSMP | FGWS | 77.3 | 81.08 | 77.3 | 72.97 | 72.97 | 71.35 | 71.35 | 67.03 | 0.81 | 0.86 | 0.82 | 0.78 | 0.8 | 0.76 | 0.75 | 0.71 |
| RANSMP | DWS | 92.5 | 93 | 93.5 | 93.5 | 92 | 91 | 92 | 87.5 | 0.94 | 0.94 | 0.95 | 0.95 | 0.94 | 0.93 | 0.94 | 0.9 |
| RANSMP | CWS | 79.33 | 85 | 78.67 | 78 | 79.67 | 79.33 | 80 | 75.33 | 0.84 | 0.88 | 0.83 | 0.83 | 0.84 | 0.84 | 0.83 | 0.81 |
| RANSMP | AWS | 56.52 | 66.09 | 60 | 59.71 | 60.58 | 60.58 | 53.91 | 60.29 | 0.65 | 0.76 | 0.7 | 0.71 | 0.72 | 0.69 | 0.63 | 0.69 |
| DOWNSMP | GOWS | 77.04 | 82.47 | 80.71 | 78.83 | 81.05 | 80.82 | 83.82 | 69.46 | 0.8 | 0.85 | 0.83 | 0.82 | 0.83 | 0.83 | 0.85 | 0.74 |
| DOWNSMP | FGWS | 74.46 | 81.84 | 77.68 | 75.72 | 75.28 | 73.98 | 71.27 | 70.69 | 0.8 | 0.87 | 0.83 | 0.82 | 0.82 | 0.8 | 0.75 | 0.76 |
| DOWNSMP | DWS | 91.2 | 90.56 | 91.14 | 90.67 | 91.48 | 88.59 | 92.09 | 83.62 | 0.93 | 0.93 | 0.93 | 0.93 | 0.94 | 0.91 | 0.94 | 0.87 |
| DOWNSMP | CWS | 77.48 | 84.29 | 82.24 | 82.94 | 78.49 | 80.25 | 84.67 | 73.82 | 0.8 | 0.86 | 0.85 | 0.85 | 0.81 | 0.82 | 0.86 | 0.76 |
| DOWNSMP | AWS | 61.69 | 57.67 | 61.83 | 61.44 | 54.67 | 60.78 | 47.44 | 54.26 | 0.64 | 0.6 | 0.64 | 0.64 | 0.53 | 0.63 | 0.48 | 0.55 |
| SMOTE | GOWS | 89.63 | 93.7 | 89.63 | 90.74 | 91.48 | 91.85 | 91.11 | 91.48 | 0.93 | 0.95 | 0.93 | 0.93 | 0.94 | 0.94 | 0.93 | 0.94 |
| SMOTE | FGWS | 94.35 | 97.58 | 96.37 | 89.52 | 96.37 | 95.16 | 69.35 | 87.5 | 0.96 | 0.98 | 0.98 | 0.94 | 0.98 | 0.97 | 0.77 | 0.92 |
| SMOTE | DWS | 95.33 | 97.67 | 96.89 | 96.89 | 96.5 | 97.28 | 97.67 | 97.28 | 0.97 | 0.99 | 0.98 | 0.98 | 0.98 | 0.98 | 0.98 | 0.98 |
| SMOTE | CWS | 93.49 | 96.93 | 94.64 | 93.1 | 92.34 | 95.4 | 88.51 | 94.64 | 0.95 | 0.98 | 0.96 | 0.95 | 0.94 | 0.97 | 0.91 | 0.96 |
| SMOTE | AWS | 81.96 | 88.63 | 84.71 | 78.04 | 84.31 | 90.98 | 60.78 | 85.49 | 0.86 | 0.91 | 0.88 | 0.83 | 0.88 | 0.93 | 0.63 | 0.88 |

TABLE V: Performance Parameters (Feature Ranking Methods)

| | | Accuracy | | | | | | | | F-Measure | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | J48 | RF | LB | ADA | BAG | MLP | NB | LOG | J48 | RF | LB | ADA | BAG | MLP | NB | LOG |
| WTOSMP | FR1 | 92.48 | 93.81 | 92.92 | 92.48 | 93.36 | 92.48 | 91.15 | 91.59 | 0.96 | 0.97 | 0.96 | 0.96 | 0.96 | 0.96 | 0.95 | 0.95 |
| WTOSMP | FR2 | 88.05 | 88.05 | 88.05 | 90.27 | 89.82 | 89.38 | 75.22 | 89.38 | 0.94 | 0.94 | 0.94 | 0.95 | 0.95 | 0.94 | 0.85 | 0.94 |
| WTOSMP | FR3 | 91.59 | 92.48 | 94.69 | 93.36 | 90.27 | 93.81 | 89.38 | 92.48 | 0.96 | 0.96 | 0.97 | 0.96 | 0.95 | 0.97 | 0.94 | 0.96 |
| WTOSMP | FR4 | 90.71 | 87.17 | 91.15 | 90.27 | 90.71 | 91.59 | 87.17 | 91.15 | 0.95 | 0.93 | 0.95 | 0.95 | 0.95 | 0.96 | 0.93 | 0.95 |
| WTOSMP | FR5 | 91.59 | 92.48 | 91.59 | 92.04 | 92.92 | 92.48 | 89.82 | 92.48 | 0.95 | 0.96 | 0.95 | 0.96 | 0.96 | 0.96 | 0.94 | 0.96 |
| RANSMP | FR1 | 78.7 | 81.99 | 81.84 | 81.74 | 83.99 | 83.08 | 84.92 | 81.33 | 0.82 | 0.85 | 0.84 | 0.84 | 0.86 | 0.86 | 0.87 | 0.84 |
| RANSMP | FR2 | 72.8 | 70.55 | 71.18 | 73.6 | 74.06 | 78.09 | 77.38 | 77.65 | 0.75 | 0.73 | 0.74 | 0.76 | 0.77 | 0.8 | 0.78 | 0.8 |
| RANSMP | FR3 | 80.16 | 80.74 | 79.33 | 78.95 | 79.45 | 82.33 | 83.06 | 81.8 | 0.82 | 0.83 | 0.82 | 0.82 | 0.82 | 0.85 | 0.86 | 0.84 |
| RANSMP | FR4 | 74.21 | 66.82 | 69.3 | 71.53 | 72.09 | 75.63 | 70.29 | 74.15 | 0.75 | 0.71 | 0.73 | 0.76 | 0.76 | 0.77 | 0.77 | 0.78 |
| RANSMP | FR5 | 77.8 | 80.77 | 79.12 | 80.58 | 82.04 | 81.27 | 86.93 | 80.03 | 0.81 | 0.83 | 0.82 | 0.83 | 0.84 | 0.84 | 0.89 | 0.82 |
| DOWNSMP | FR1 | 88.45 | 93.14 | 91.7 | 90.98 | 89.17 | 91.7 | 87.73 | 91.34 | 0.92 | 0.95 | 0.94 | 0.94 | 0.92 | 0.94 | 0.91 | 0.94 |
| DOWNSMP | FR2 | 85.15 | 89.49 | 85.15 | 84.78 | 85.87 | 88.41 | 77.9 | 82.97 | 0.89 | 0.92 | 0.9 | 0.9 | 0.9 | 0.92 | 0.82 | 0.88 |
| DOWNSMP | FR3 | 87.46 | 89.25 | 89.25 | 87.81 | 89.61 | 92.83 | 89.25 | 91.76 | 0.91 | 0.92 | 0.93 | 0.91 | 0.93 | 0.95 | 0.93 | 0.94 |
| DOWNSMP | FR4 | 87.19 | 88.61 | 85.41 | 80.78 | 83.99 | 85.05 | 78.29 | 86.12 | 0.91 | 0.92 | 0.9 | 0.87 | 0.89 | 0.89 | 0.85 | 0.9 |
| DOWNSMP | FR5 | 89.82 | 90.55 | 90.55 | 88.73 | 91.27 | 90.91 | 88.73 | 92 | 0.93 | 0.93 | 0.93 | 0.92 | 0.94 | 0.94 | 0.92 | 0.94 |
| SMOTE | FR1 | 82.53 | 83.1 | 81.13 | 83.45 | 83.74 | 81.31 | 86.32 | 80.14 | 0.85 | 0.86 | 0.84 | 0.86 | 0.86 | 0.84 | 0.88 | 0.83 |
| SMOTE | FR2 | 73.61 | 73.38 | 71.89 | 75.39 | 74.9 | 80.25 | 78.11 | 77.5 | 0.77 | 0.77 | 0.75 | 0.78 | 0.78 | 0.82 | 0.78 | 0.81 |
| SMOTE | FR3 | 75.34 | 78.17 | 75.58 | 79 | 78.69 | 81.41 | 82.71 | 80.57 | 0.79 | 0.81 | 0.79 | 0.82 | 0.82 | 0.84 | 0.86 | 0.84 |
| SMOTE | FR4 | 73.66 | 70.47 | 72.65 | 74.21 | 75.05 | 74.24 | 72.42 | 76.29 | 0.76 | 0.75 | 0.76 | 0.78 | 0.78 | 0.77 | 0.79 | 0.8 |
| SMOTE | FR5 | 78.68 | 82.92 | 81.42 | 80.86 | 83.99 | 82.82 | 86.07 | 83.17 | 0.81 | 0.85 | 0.83 | 0.83 | 0.86 | 0.85 | 0.88 | 0.85 |

values for NB, J48, RF, MLP, LB and ADA. Similarly, we observe that the inter-quartile range for RANSMP is more than the inter-quartile ranges of WTOSMP, SMOTE and DOWNSMP. Box-plots in Figure 3 and 4 are for all metrics and reduced metrics whereas the box-plot in Figure 5 is overall and for all the combinations. From Figure 5, we observe that the boxplot for the AUC values for LOG, LB, SMOTE and FR1 are comparatively taller. This means that there is more variation in the AUC values obtained from multiple executions. While the box-plots in Figures 3, 4 and 5 shows the results in the form of a visualization, Tables VI, VII and VIII presents exact values obtained from the experiments for all the combinations of the experimental parameters.

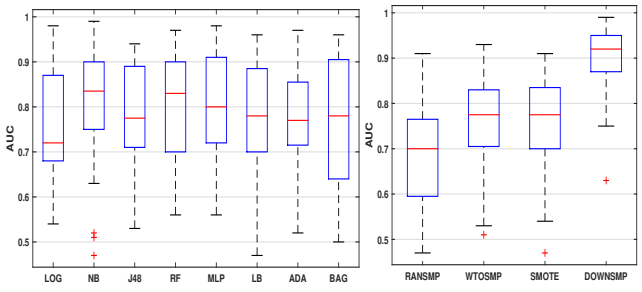


Fig. 3: Boxplot: AUC (All Metrics)

TABLE VI: Descriptive Statistics of the Performance in-terms of AUC (All Metrics)

| AUC | | | | | | | |
|---|---|---|---|---|---|---|---|
| | **Min** | **Max** | **Mean** | **Median** | **Std Dev** | **Q1** | **Q3** |
| LOG | 0.54 | 0.98 | 0.76 | 0.72 | 0.13 | 0.68 | 0.87 |
| NB | 0.47 | 0.99 | 0.79 | 0.84 | 0.15 | 0.75 | 0.90 |
| J48 | 0.53 | 0.94 | 0.77 | 0.78 | 0.12 | 0.71 | 0.89 |
| RF | 0.56 | 0.97 | 0.80 | 0.83 | 0.13 | 0.70 | 0.90 |
| MLP | 0.56 | 0.98 | 0.79 | 0.80 | 0.13 | 0.72 | 0.91 |
| LB | 0.47 | 0.96 | 0.77 | 0.78 | 0.13 | 0.70 | 0.89 |
| ADA | 0.52 | 0.97 | 0.76 | 0.77 | 0.13 | 0.72 | 0.86 |
| BAG | 0.50 | 0.96 | 0.75 | 0.78 | 0.15 | 0.64 | 0.91 |
| RANSMP | 0.47 | 0.91 | 0.69 | 0.70 | 0.11 | 0.60 | 0.77 |
| WTOSMP | 0.51 | 0.93 | 0.75 | 0.78 | 0.12 | 0.71 | 0.83 |
| SMOTE | 0.47 | 0.91 | 0.75 | 0.78 | 0.11 | 0.70 | 0.84 |
| DOWNSMP | 0.63 | 0.99 | 0.90 | 0.92 | 0.07 | 0.87 | 0.95 |

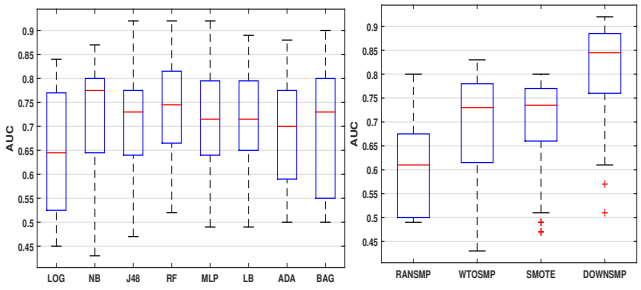


Fig. 4: Boxplot: AUC (Reduced Metrics)

TABLE VII: Descriptive Statistics of the Performance in-terms of AUC (Reduced Metrics)

| AUC | | | | | | | |
|---|---|---|---|---|---|---|---|
| | **Min** | **Max** | **Mean** | **Median** | **Std Dev** | **Q1** | **Q3** |
| LOG | 0.45 | 0.84 | 0.66 | 0.65 | 0.13 | 0.53 | 0.77 |
| NB | 0.43 | 0.87 | 0.71 | 0.78 | 0.13 | 0.65 | 0.80 |
| J48 | 0.47 | 0.92 | 0.71 | 0.73 | 0.13 | 0.64 | 0.78 |
| RF | 0.52 | 0.92 | 0.74 | 0.75 | 0.12 | 0.67 | 0.82 |
| MLP | 0.49 | 0.92 | 0.71 | 0.72 | 0.13 | 0.64 | 0.80 |
| LB | 0.49 | 0.89 | 0.71 | 0.72 | 0.11 | 0.65 | 0.80 |
| ADA | 0.50 | 0.88 | 0.68 | 0.70 | 0.12 | 0.59 | 0.78 |
| BAG | 0.50 | 0.90 | 0.69 | 0.73 | 0.15 | 0.55 | 0.80 |
| RANSMP | 0.49 | 0.80 | 0.61 | 0.61 | 0.09 | 0.50 | 0.68 |
| WTOSMP | 0.43 | 0.83 | 0.69 | 0.73 | 0.11 | 0.62 | 0.78 |
| SMOTE | 0.47 | 0.80 | 0.69 | 0.74 | 0.10 | 0.66 | 0.77 |
| DOWNSMP | 0.51 | 0.92 | 0.81 | 0.85 | 0.11 | 0.76 | 0.89 |

**Null Hypothesis Statistical Significance Testing**: We apply the nonparametric Wilcoxon signed-rank test for statistical hypothesis testing (with a Bonferroni correction) to investigate if the two samples consisting of AUC values generated from two different combinations of experimental parameters were selected from population having same or different distributions. The nonparametric Wilcoxon signed-rank test is a type of paired t-test used to check if the null hypothesis can be accepted or rejected. Figures 6, 7 and 8 shows the output of the hypothesis testing consisting of a comparison between four data sampling methods, eight classification algorithms and six feature ranking methods on AUC performance metrics. Figures 6, 7 and 8 consists of a red dot or a green dot. A red dot means that H0 (null hypothesis) is rejected and a green dot means

TABLE VIII: Descriptive Statistics of the Performance in-terms of AUC (Overall)

| AUC | | | | | | | |
|---|---|---|---|---|---|---|---|
| | **Min** | **Max** | **Mean** | **Median** | **Std Dev** | **Q1** | **Q3** |
| LOG | 0.44 | 0.94 | 0.67 | 0.68 | 0.14 | 0.52 | 0.79 |
| NB | 0.46 | 0.95 | 0.73 | 0.76 | 0.13 | 0.69 | 0.82 |
| J48 | 0.49 | 0.96 | 0.72 | 0.75 | 0.13 | 0.60 | 0.82 |
| RF | 0.54 | 0.98 | 0.76 | 0.76 | 0.11 | 0.68 | 0.86 |
| MLP | 0.49 | 0.98 | 0.71 | 0.73 | 0.13 | 0.59 | 0.81 |
| LB | 0.50 | 0.99 | 0.73 | 0.74 | 0.12 | 0.65 | 0.81 |
| ADA | 0.50 | 0.98 | 0.71 | 0.73 | 0.12 | 0.60 | 0.80 |
| BAG | 0.50 | 0.98 | 0.72 | 0.75 | 0.14 | 0.60 | 0.83 |
| RANSMP | 0.44 | 0.90 | 0.70 | 0.71 | 0.12 | 0.59 | 0.80 |
| WTOSMP | 0.44 | 0.90 | 0.72 | 0.75 | 0.12 | 0.61 | 0.81 |
| SMOTE | 0.44 | 0.92 | 0.65 | 0.64 | 0.12 | 0.54 | 0.76 |
| DOWNSMP | 0.44 | 0.90 | 0.68 | 0.71 | 0.14 | 0.54 | 0.80 |
| FR1 | 0.44 | 0.99 | 0.74 | 0.76 | 0.13 | 0.65 | 0.83 |
| FR2 | 0.44 | 0.98 | 0.70 | 0.71 | 0.13 | 0.59 | 0.81 |
| FR3 | 0.47 | 0.96 | 0.72 | 0.76 | 0.13 | 0.62 | 0.82 |
| FR4 | 0.46 | 0.97 | 0.71 | 0.73 | 0.13 | 0.60 | 0.81 |
| FR5 | 0.47 | 0.96 | 0.72 | 0.74 | 0.12 | 0.63 | 0.81 |

TABLE IX: Mean Difference AUC (Data Sampling Methods)

| | WTOSMP | RANSMP | DOWNSMP | SMOTE |
|---|---|---|---|---|
| WTOSMP | 0.000 | -0.086 | -0.084 | -0.193 |
| RANSMP | 0.086 | 0.000 | 0.002 | -0.107 |
| DOWNSMP | 0.084 | -0.002 | 0.000 | -0.109 |
| SMOTE | 0.193 | 0.107 | 0.109 | 0.000 |

that H0 is accepted. H0 being accepted means that there is no statistically significant difference between the two techniques or combination of parameters under comparison. While performing statistical significance testing, we use a standard cut-off value of $0.05$ (signifying that the null hypothesis is rejected when $p \leq 0.05$ and not rejected when $p > 0.05$).

According to Figure 6, there are a total of $6$ possible comparisons as there are $4$ data sampling methods. We observe from Figure 6 that $5$ out of $6$ cells contains red dots and only $1$ cell contains a green dot. Figure 6 reveals that there is no statistically significant difference between DOWNSMP and RANSMP. However, there is a statistically significant difference between the performance of SMOTE and WTOSMP, SMOTE and RANSMP, SMOTE and DOWNSMP, DOWNSMP and WTOSMP, RANSMP and WTOSMP. Figure 7 shows the pairwise comparison between the eight classification techniques. Figure 7 reveals that there is a statistical significant difference between LOG and the remaining 7 classifiers. We observe that NB shows difference with LOG and not any other classification algorithm. We notice that RF and J48 performance are different based on the statistical hypothesis testing results as shown by the red dot at their intersecting cell in the Figure 7. Figure 7 shows that classification algorithm influences the accuracy results. Figure 8 displays the pairwise comparison results for the $6$ feature ranking methods. Figure 7 reveals there is a statistical significant difference between the performances between several feature ranking methods as shown by the red dots in the intersecting cell. From Figure 7, we infer that the feature ranking method FR1 results in statistically significant

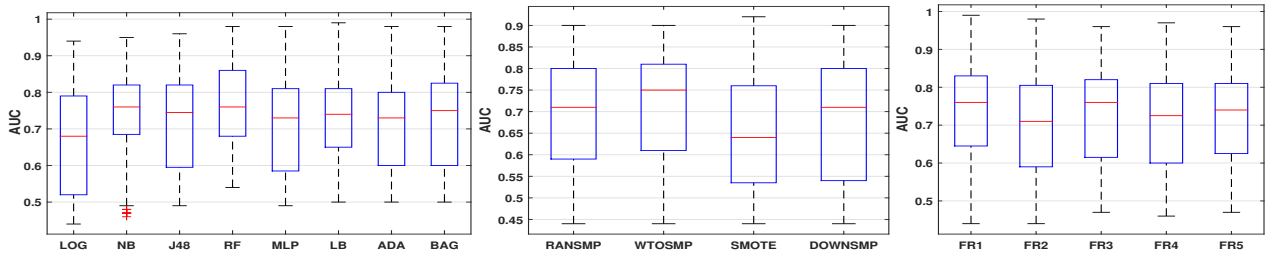


Fig. 5: Boxplot: AUC (Overall)

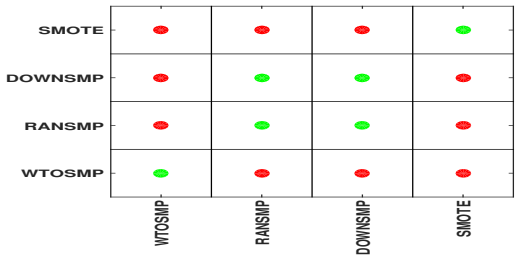


Fig. 6: Wilcoxon Signed Rank Test Analysis: P-Value of AUC (Data Sampling Methods)


Fig. 7: Wilcoxon Signed Rank Test Analysis: P-Value of AUC (Classification Techniques)

different performance than the other feature ranking methods. We do not observe any difference in performance between FR4 and FR2 as well as FR4 and FR3.

**Mean AUC Value Difference**: Tables IX, X and XI shows the difference between the mean AUC values for various data sampling techniques, feature ranking techniques and classification algorithms. A higher AUC value means a better performance. The cells in Tables IX, X and XI displays the difference between the mean AUC value of the technique mentioned in the row heading and the technique mentioned in the column heading. A positive value of the cell indicates that the technique mentioned in the row heading outperforms the technique mentioned in the column heading. From Table IX, we infer that the NB outperforms LOG classification algorithm in-terms of the AUC value by 0.056. Table IX shows that NB outperforms all other classification techniques except RF. Table IX reveals that RF performs the best as its mean AUC value is higher than the mean AUC values

TABLE X: Mean Difference AUC (Classification Algorithm)

| | LOG | NB | J48 | RF | MLP | LB | ADA | BAG |
|---|---|---|---|---|---|---|---|---|
| LOG | 0.000 | -0.056 | -0.043 | -0.086 | -0.044 | -0.054 | -0.031 | -0.039 |
| NB | 0.056 | 0.000 | 0.013 | -0.030 | 0.013 | 0.003 | 0.025 | 0.017 |
| J48 | 0.043 | -0.013 | 0.000 | -0.043 | 0.000 | -0.010 | 0.012 | 0.004 |
| RF | 0.086 | 0.030 | 0.043 | 0.000 | 0.042 | 0.032 | 0.055 | 0.047 |
| MLP | 0.044 | -0.013 | 0.000 | -0.042 | 0.000 | -0.010 | 0.012 | 0.005 |
| LB | 0.054 | -0.003 | 0.010 | -0.032 | 0.010 | 0.000 | 0.022 | 0.015 |
| ADA | 0.031 | -0.025 | -0.012 | -0.055 | -0.012 | -0.022 | 0.000 | -0.008 |
| BAG | 0.039 | -0.017 | -0.004 | -0.047 | -0.005 | -0.015 | 0.008 | 0.000 |

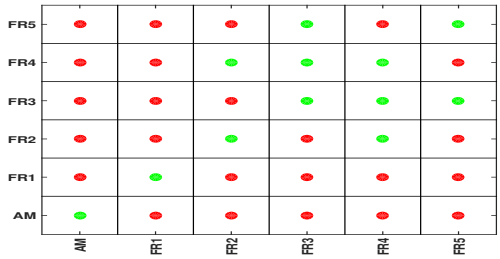


Fig. 8: Wilcoxon Signed Rank Test Analysis: P-Value of AUC (Feature Ranking Methods)

of all other classification algorithms. We observe that the mean AUC value of LOG is lower than the mean AUC values of all other classification algorithms. The highest difference between the mean AUC values of classification algorithms is between RF and LOG which is 0.086. Table X shows that SMOTE is the best performing data sampling method in comparison to other sampling methods used in our experiments. The difference in the mean AUC values of SMOTE and WTOSMP is the highest which is 0.193. There is a minor difference between the overall performance of DOWNSMP and RANSMP. The second best performing technique out of the four techniques is RANSMP. Table XI reveals that AM results in the best performance in comparison to the five feature ranking methods. All the values in the cells for which AM is in the row heading are positive. The maximum difference is between AM and FR4. We observe that FR2 is the weakest approach in comparison to all other approaches. There is a minor difference in the performance of FR2 and FR4.

**RQ 2:** Is there a statistically significant difference in the prediction performance of five different feature selection techniques and four different data sampling techniques?

TABLE XI: Mean Difference AUC (Feature Ranking Methods)

| | AM | FR1 | FR2 | FR3 | FR4 | FR5 |
|---|---|---|---|---|---|---|
| AM | 0.000 | 0.038 | 0.072 | 0.053 | 0.067 | 0.049 |
| FR1 | -0.038 | 0.000 | 0.034 | 0.016 | 0.029 | 0.012 |
| FR2 | -0.072 | -0.034 | 0.000 | -0.019 | -0.005 | -0.023 |
| FR3 | -0.053 | -0.016 | 0.019 | 0.000 | 0.013 | -0.004 |
| FR4 | -0.067 | -0.029 | 0.005 | -0.013 | 0.000 | -0.017 |
| FR5 | -0.049 | -0.012 | 0.023 | 0.004 | 0.017 | 0.000 |

Which of the feature selection and data sampling technique(s) yields the best result?

**Answer RQ 2:** *There is a statistically significant difference in the performance of various feature selection techniques and data sampling techniques. SMOTE is the best performing data sampling method in comparison to other sampling methods used in our experiments. OneR feature ranking outperforms all other feature ranking techniques.*

**RQ 3:** What is the relative performance of the eight different types of learning algorithms and classifiers in-terms of the AUC, accuracy and f-measure metrics? Is there a statistically significant difference in the prediction performance of eight different learning algorithms or classifiers?

**Answer RQ 3:** *There is a statistically significant difference in the performance of the eight different types of learning algorithms. Random Forest learning algorithms performs best followed by Nave Bayes for the task of web-service anti-pattern detection using source code metrics.*

## VI. Conclusion

Our main conclusion is that it is possible to predict the occurrence of web-service anti-patterns using object oriented source code metrics derived from the source code of the web services and machine learning classifiers. We observe a correlation between the value of certain source code metrics and web-services anti-patterns. We conduct a series of experiments by building 960 classification models. Our conclusion is that data sampling method is an important pre-processing step since the class distribution is imbalanced and there is a statistically significant difference between the performances of various data sampling methods. Our results indicate that SMOTE is the best performing data sampling method. Since our dataset consists of several dimensions, we observe that feature ranking method us useful to reduce the dimension of the dataset and identify informative features. Our results reveal that OneR feature ranking method outperforms all other feature ranking method used in our experiments. In terms of the learning algorithms, we conclude that there is a statistically significant difference between the performance of the 8 classifiers and Random Forest approach outperforms all other classification algorithms.

## References


[1] Y. Kun, W. Xiao-Ling, and Z. Ao-Ying, "Underlying techniques for web services: A survey," in *Journal of software*. Citeseer, 2004.

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

[3] 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, pp. 86–93, 2002.

[4] J. Kral and M. Zemlicka, "The most important service-oriented antipatterns," in *Software Engineering Advances, 2007. ICSEA 2007. International Conference on*. IEEE, 2007, pp. 29–29.

[5] J. Král and M. Žemlicka, "Popular soa antipatterns," in *Future Computing, Service Computation, Cognitive, Adaptive, Content, Patterns, 2009. COMPUTATIONWORLD'09. Computation World:*. IEEE, 2009, pp. 271–276.

[6] A. Ouni, M. Kessentini, K. Inoue, and M. O. Cinnéide, "Search-based web service antipatterns detection," *IEEE Transactions on Services Computing*, 2015.

[7] A. Ouni, R. Gaikovina Kula, M. Kessentini, and K. Inoue, "Web service antipatterns detection using genetic programming," in *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*. ACM, 2015, pp. 1351–1358.

[8] F. Palma, N. Moha, G. Tremblay, and Y.-G. Guéhéneuc, "Specification and detection of soa antipatterns in web services," in *European Conference on Software Architecture*. Springer, 2014, pp. 58–73.

[9] F. Palma, M. Nayrolles, N. Moha, Y.-G. Guéhéneuc, B. Baudry, and J.-M. Jézéquel, "Soa antipatterns: An approach for their specification and detection," *International Journal of Cooperative Information Systems*, vol. 22, no. 04, p. 1341004, 2013.

[10] N. Moha, F. Palma, M. Nayrolles, B. J. Conseil, Y.-G. Guéhéneuc, B. Baudry, and J.-M. Jézéquel, "Specification and detection of soa antipatterns." in *ICSOC*. Springer, 2012, pp. 1–16.

[11] C. Mateos, M. Crasso, A. Zunino, and J. L. O. Coscia, "Detecting wsdl bad practices in code–first web services," *International Journal of Web and Grid Services*, vol. 7, no. 4, pp. 357–387, 2011.

[12] J. L. Ordiales Coscia, C. Mateos, M. Crasso, and A. Zunino, "Anti-pattern free code-first web services for state-of-the-art java wsdl generation tools," *International Journal of Web and Grid Services*, vol. 9, no. 2, pp. 107–126, 2013.

[13] M. Nayrolles, F. Palma, N. Moha, and Y.-G. Guéhéneuc, "Soda: A tool support for the detection of soa antipatterns," in *International Conference on Service-Oriented Computing*. Springer, 2012, pp. 451–455.

[14] S. R. Chidamber and C. F. Kemerer, "A metrics suite for object oriented design," *IEEE Transactions on software engineering*, vol. 20, no. 6, pp. 476–493, 1994.

[15] S. Chidamber and C. F. Kemerer, *Towards a metrics suite for object oriented design*. ACM, 1991, vol. 26, no. 11.

[16] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: synthetic minority over-sampling technique," *Journal of artificial intelligence research*, vol. 16, pp. 321–357, 2002.

[17] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Transactions on knowledge and data engineering*, vol. 21, no. 9, pp. 1263–1284, 2009.

[18] H. Liu and L. Yu, "Toward integrating feature selection algorithms for classification and clustering," *IEEE Transactions on knowledge and data engineering*, vol. 17, no. 4, pp. 491–502, 2005.

[19] T.-S. Lim, W.-Y. Loh, and Y.-S. Shih, "A comparison of prediction accuracy, complexity, and training time of thirty-three old and new classification algorithms," *Machine learning*, vol. 40, no. 3, pp. 203–228, 2000.

[20] M. Sokolova and G. Lapalme, "Performance measures in classification of human communications," *Advances in Artificial Intelligence*, pp. 159–170, 2007.

[21] N. Juristo and O. S. Gómez, "Replication of software engineering experiments," in *Empirical software engineering and verification*. Springer, 2012, pp. 60–88.

[22] B. Kitchenham, "The role of replications in empirical software engineeringa word of warning," *Empirical Software Engineering*, vol. 13, no. 2, pp. 219–221, 2008.

[23] J. Sandberg and M. Alvesson, "Ways of constructing research questions: gap-spotting or problematization?" *Organization*, vol. 18, no. 1, pp. 23–44, 2011.