# An Empirical Analysis on the Prediction of Web Service Anti-patterns Using Source Code Metrics and Ensemble Techniques

Sahithi Tummalapalli[1](✉), Juhi Mittal[1], Lov Kumar[1], Lalitha Bhanu Murthy Neti[1], and Santanu Kumar Rath[2]

[1] BITS Pilani Hyderabad, Hyderabad, India
{P20170433,f20160298,lovkumar,bhanu}@hyderabad.bitspilani.ac.in
[2] NIT, Rourkela, India
skrath@nitrkl.ac.in

**Abstract.** Today's software program enterprise uses web services to construct distributed software systems based on the Service Oriented Architecture (SOA) paradigm. The web service description is posted by a web service provider, which may be observed and invoked by a distributed application. Service-Based Systems (SBS) need to conform themselves through years to fit within the new user necessities. These may result in the deterioration of the quality and design of the software systems and might reason the materialization of insufficient solutions called Anti-patterns. Anti-pattern detection using object-oriented source code metrics may be used as part of the software program improvement life cycle to lessen the maintenance of the software system and enhance the quality of the software. The work is motivated by developing an automatic predictive model for predicting web services anti-patterns using static evaluations of the source code metrics. The center ideology of this work is to empirically investigate the effectiveness of different variants of data sampling technique, Synthetic Minority Over Sampling TEchnique (SMOTE), and the ensemble learning techniques in the prediction of web service anti-patterns.

**Keywords:** Anti-pattern · WSDL · Ensemble techniques · Code quality

## 1 Introduction

Service-Oriented Architecture (SOA) is a tiered structure that assists corporations in sharing information and logic between different applications and usage modes. An excellent SOA solution leads to loosely coupled devices that empower the readiness expected to align IT and the enterprise team. A wide variety of technologies, particularly OSGi, SCA, and web services, are used for imposing the SOA structure. Various service-based systems (SBS), starting from business frameworks to cloud-based frameworks, are built using SOA architectures. The developing requirement of the customers forces the SBS's to conform to fit the new needs of the users. This evolving may additionally cause the deterioration of the design and quality of the software-based systems, ensuing in a systematic strategy to a repeating hassle, referred to as Anti-patterns [4]. Anti-patterns are the structures in the design that suggests infringement of critical design concepts and contrarily sway design quality [4]. These are not accidental but rather normal slip-ups and are nearly consistently accompanied with sincere intentions. Anti-patterns makes it challenging for the advancement and maintenance of the software program systems; however, they likewise will assist in figuring out troubles within the design, the code, and the management of software program initiatives. In this paper, we have developed models for the detection of four unique anti-patterns, namely: AP1: Chatty Web Service(CWS); AP2: Fine-Grained Web Service (FGWS); AP3: Data Web Service (DWS); AP4: God Web Service (GWS).

The vital motivation of the work added in this paper is to investigate the utilization of ensemble learning techniques in the detection of web-service anti-patterns. This work is roused by the need to fabricate procedures and tools to detect anti-patterns in web services automatically.

## 2 Related Work

Moha et al. [6] introduced a novel framework for specification and detection of anti-patterns in Service-based systems to detect new patterns like Tiny Service and Multiservice and achieved a precision of more than 0.9. Ouni et al. [8] 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. Dimitrios et al. [12] used the Protege platform, a web-based environment, to facilitate collaborative ontology editing allowing multiple users to edit and enrich the anti-pattern ontology simultaneously. Palma et al. [9] used a rule-based search to detect and identify BP anti-patterns in the Business Process Execution Language (BPEL) processes generated via orchestrating web services. Coscia et al. [7] proposed a statistical correlation analysis on the WSDL-level service metrics and the number of traditional OO metrics and found a correlation between them. SODA-W, an extension of the SOFA framework, detects the SOAP and REST anti-patterns using the pre-established DSL. Upadhyaya et al. [15] proposed an approach to detect 9 SOA patterns. It is observed from the literature reviewed here that the research on SOA anti-pattern detection still needs to be explored thoroughly. Dimitrios et al. [13] have proposed a novel OWL ontology-based knowledge system, SPARSE, that helps in detecting anti-patterns. The ontology provides documentation for the anti-patterns, describing their relationship with other anti-patterns through their causes, symptoms, and consequences. Jaffar et al. [3] 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 other classes. In this paper, authors have empirically investigated the consequences of dependencies in object-oriented system by focusing and analysing the relationship between the presence of co-change and static dependencies and change proness, fault proness and fault types that the classes are exhibiting. Kumar et al. [5] proposed an approach for the automatic detection of anti-patterns by static analysis of the source code. In this paper, author proposed that the aggregate values of the source code metrics computed at the web-service level can be used as predictors for anti-pattern detection. Saluja et al. [11] proposed a new optimized algorithm that uses dynamic metrics for execution in addition to the static metrics. The results obtained are further optimized using genetic algorithms. The proposed results achieved better results than the existing methods and had a recall rate of approximately 0.9

## 3 Dataset

The data set with 226 publicly available web services that are shared by Ouni et al. on GitHub[1] are used for experiments in this paper. Figure 1 shows the distribution of the web services in which the anti-patterns exists (#AP) and does not exist (#NAP).

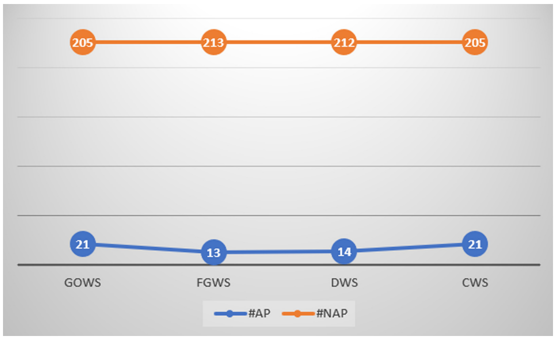


**Fig. 1.** Distribution of anti-patterns in web services

## 4 Proposed Solution Framework

Figure 2 shows the detailed overview of the proposed framework. Figure 2 depicts that the proposed framework is a multi−step procedure consisting of computing CK metrics from the WSDL file, applying aggregation metrics for computing

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

metrics at file level, handling class imbalance problem using different variants of SMOTE discussed in Sect. 4.2, removal of irrelevant features using techniques such as PCA and RSA discussed in Sect. 4.3, and lastly the development of anti-pattern prediction models using five different ensemble learning techniques. First, the java files are extracted from each of the WSDL file, for which the CK metrics discussed in Table 3 are computed using CKJM tool. To convert the metrics computed at file level to system level, aggregation measures which are discussed in Table 3 are applied. This forms the dataset using which the anti-pattern prediction models are developed. Next, we used different variants of SMOTE technique for handling the class imbalance problem. Further, we compare the models trained using balanced data with the models developed using original data. After this, we use features selected using three different feature selection techniques namely, significant features using rank−sum test, Rough Set Analysis (RSA), and Principal Component Analysis (PCA). Finally, five ensemble learning techniques namely: Bagging Classifier (EST1), Random Forest Classifier (EST2), Extra Trees Classifier (EST3), AdaBoost Classifier (EST4), Gradient Boosting Classifier (EST5) are used to generate models for the prediction of web service anti-patterns. We use performance measures such as: AUC, F-measure, and accuracy for computing and comparing the performance of the models generated for the prediction of web service anti-patterns.

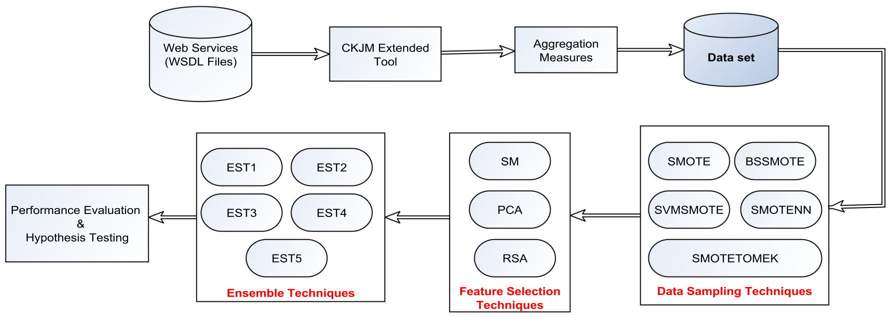


**Fig. 2.** Proposed framework

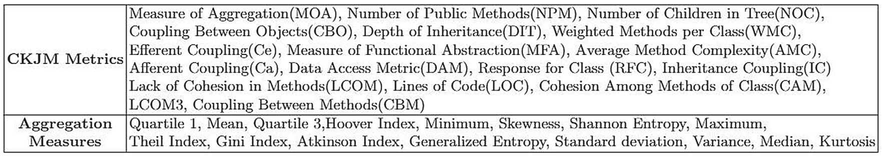
### 4.1 Preprocessing of the Dataset

Preprocessing of the dataset involves the extraction of java files from the WSDL files (raw data) from which the source code metrics are computed. The dataset considered has a collection of 226 WSDL files. In this paper, A step-wise procedure for preprocessing of data is detailed here.

**Step-1:** Source code metrics computation:
We used WSDL2Java tool in this work to extract java files from each of the WSDL file, and the Chidamber and Kemerer metrics (CK metrics) along

with other java metrics are computed for each of the java file using CKJM extended tool[2]. The list of various CK metrics used in this paper are listed in Fig. 3. The definition of each of the CK metric along with their computation formula are documented in [2].

**Step-2:** Aggregation measures on the source code metrics:
In this study, our objective is to develop one model for predicting an anti-pattern present in the WSDL file. Here, we have used CK metrics to measure each java file and the metrics computed here are at the file level. Further a total of sixteen aggregation measures are applied to the metrics computed at the file level to obtain the metrics at the system level. The list of aggregation measures used in this paper are given in Fig. 3.



| | |
|---|---|
| CKJM Metrics | Measure of Aggregation(MOA), Number of Public Methods(NPM), Number of Children in Tree(NOC), Coupling Between Objects(CBO), Depth of Inheritance(DIT), Weighted Methods per Class(WMC), Efferent Coupling(Ce), Measure of Functional Abstraction(MFA), Average Method Complexity(AMC), Afferent Coupling(Ca), Data Access Metric(DAM), Response for Class (RFC), Inheritance Coupling(IC) Lack of Cohesion in Methods(LCOM), Lines of Code(LOC), Cohesion Among Methods of Class(CAM), LCOM3, Coupling Between Methods(CBM) |
| Aggregation Measures | Quartile 1, Mean, Quartile 3,Hoover Index, Minimum, Skewness, Shannon Entropy, Maximum, Theil Index, Gini Index, Atkinson Index, Generalized Entropy, Standard deviation, Variance, Median, Kurtosis |

**Fig. 3.** List of CKJM metrics and aggregation measures

## 4.2 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 Sect. 3, 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 [1]. In this paper, we are considering the five different data sampling techniques namely SMOTE, Borderline Smote (BSMOTE), SVM-SMOTE (SVMSMOTE), SMOTE- Edited Nearest Neighbour (SMOTEENN), and SMOTETOMEK along with the original dataset (OD) to generate the predictive models.

## 4.3 Effectiveness of Metrics

A total of three models are considered in this study where the occurrence of an anti-pattern is considered as the dependent variable, and the source code metrics computed are taken as an independent variable for developing the relation.

**Subset of Features Selected as Significant Features (SIGF):** In our previous work, We applied a set of feature selection techniques on original source code metrics to obtain the significant source code metrics(SM). This

[2] http://gromit.iiar.pwr.wroc.pl/pinf/ckjm/.

**Table 1.** Source code metrics selected using RSA: all anti-patterns

| Anti-pattern | Reduced set of metrics |
|---|---|
| GOWS | Hoover index (WMC), Min (DIT), Mean (DIT), Max (RFC), skewness (LCOM), skewness (Ca), Q1 (DAM), Gini index (DAM), Gini index (IC) |
| FGWS | Median (CBO), Gini index (LCOM), skewness (Ca), Mean (LCOM3), Std (LCOM3), Gini index (CAM), Hoover index (CAM) |
| DWS | Q1 (WMC), Var (NOC), Gini index (CBO), skewness (LCOM), Median (MOA) |
| CWS | Mean (DIT), Q1 (CBO), Generalized entropy (CBO), Mean (RFC), skewness (Ca), Generalized entropy (MOA), skewness (AMC) |

features are used as input to develop the predictive models for various anti-patterns prediction [14].

$$\textit{Anti-pattern}\ \text{predictability} = f(\text{Significant features}) \tag{1}$$

**Subset of Features Selected Using RSA:** In order to reduce the complexity of the model developed, it is important to remove irrelevant features. For this purpose, we use a feature reduction technique known as **"Rough Set Analysis (RSA)"** to obtain a reduced set of features. Here the anti-pattern predictability is defined as a function of a reduced set of metrics.

$$\textit{Anti-pattern}\ \text{predictability} = f(\text{Reduced set of features}) \tag{2}$$

RSA enables the developer to find the subset of the original source code metrics that are most illuminating removing all the irrelevant attributes with minimal information loss [10]. Table 1 shows the reduced significant feature set for all the anti-patterns considered in this study. For example, Hoover index (WMC), Min (DIT), Mean (DIT), Max (RFC), skewness (LCOM), skewness (Ca), Q1 (DAM), Gini index (DAM), Gini index (IC) are selected features using RSA analysis for GOWS anti-pattern.

**Subset of Features Selected Using PCA:** A feature extraction technique known as **"Principle Component Analysis(PCA)"** is used to develop a model with less complexity using reduced set of features. PCA reduces the dimensionality of the data. It helps in reducing the computational complexity of the model. The primary idea of PCA is to diminish the dimensionality of a dataset comprising of numerous features correlated with one another, either vigorously or softly, while holding the variation present in the dataset, up to the greatest degree. Table 2a illustrates the eigenvalue, % variance, % cumulative for principal component (PC) domain metrics selected for GOWS anti-pattern. Similarly, 22, 21, and 21 PCs are selected for the FGWS, DWS, and CWS anti-patterns, respectively.

$$\textit{Anti-pattern}\ \text{predictability} = f(\text{Extracted set of features}) \tag{3}$$

The same is done by transforming the features into a new set of features, which are known as the principal components or simply, the pc′s.

## 4.4 Classifier Techniques

In this paper, we applied five ensemble techniques for training the predictive models for the detection of web service anti-patterns. The ensemble techniques we have used in this paper are: Bagging classifier (EST1), Random Forest classifier (EST2), Extra Trees classifier (EST3), AdaBoost classifier (EST4) and Gradient Boosting classifier (EST5).

**Table 2.** Subset of features selected using PCA for anti-patterns

(a) GOWS

| | Eigen value | Variance (%) | Cumulative (%) |
|---|---|---|---|
| pc-1 | 32.87 | 20.81 | 20.81 |
| pc-2 | 24.39 | 15.43 | 36.24 |
| pc-3 | 20.40 | 12.91 | 49.15 |
| pc-4 | 13.19 | 8.35 | 57.50 |
| pc-5 | 10.22 | 6.47 | 63.97 |
| pc-6 | 9.12 | 5.77 | 69.74 |
| pc-7 | 8.23 | 5.21 | 74.95 |
| pc-8 | 5.50 | 3.48 | 78.43 |
| pc-9 | 3.27 | 2.07 | 80.50 |
| pc-10 | 3.02 | 1.91 | 82.41 |
| pc-11 | 2.89 | 1.83 | 84.24 |
| pc-12 | 2.88 | 1.82 | 86.06 |
| pc-13 | 2.60 | 1.64 | 87.70 |
| pc-14 | 2.43 | 1.54 | 89.24 |
| pc-15 | 2.32 | 1.47 | 90.71 |
| pc-16 | 2.02 | 1.28 | 91.99 |
| pc-17 | 1.84 | 1.17 | 93.15 |
| pc-18 | 1.62 | 1.03 | 94.18 |

(b) FGWS

| | Eigen value | Variance(%) | Cumulative(%) |
|---|---|---|---|
| pc-1 | 30.37 | 20.38 | 20.38 |
| pc-2 | 24.28 | 16.30 | 36.68 |
| pc-3 | 19.59 | 13.15 | 49.83 |
| pc-4 | 11.98 | 8.04 | 57.87 |
| pc-5 | 10.98 | 7.37 | 65.24 |
| pc-6 | 7.96 | 5.35 | 70.59 |
| pc-7 | 5.07 | 3.40 | 73.99 |
| pc-8 | 4.31 | 2.89 | 76.88 |
| pc-9 | 3.47 | 2.33 | 79.21 |
| pc-10 | 2.99 | 2.00 | 81.21 |
| pc-11 | 2.44 | 1.64 | 82.85 |
| pc-12 | 2.41 | 1.62 | 84.47 |
| pc-13 | 2.38 | 1.60 | 86.07 |
| pc-14 | 2.30 | 1.55 | 87.61 |
| pc-15 | 2.03 | 1.36 | 88.97 |
| pc-16 | 1.92 | 1.29 | 90.26 |
| pc-17 | 1.87 | 1.25 | 91.51 |
| pc-18 | 1.62 | 1.09 | 92.60 |
| pc-19 | 1.23 | 0.83 | 93.43 |
| pc-20 | 1.19 | 0.80 | 94.23 |
| pc-21 | 1.07 | 0.72 | 94.94 |
| pc-22 | 1.04 | 0.70 | 95.64 |

## 5 Experimental Results

In this work, five sampling techniques besides the original data set (OD), features selected by three different feature selection techniques and five ensemble

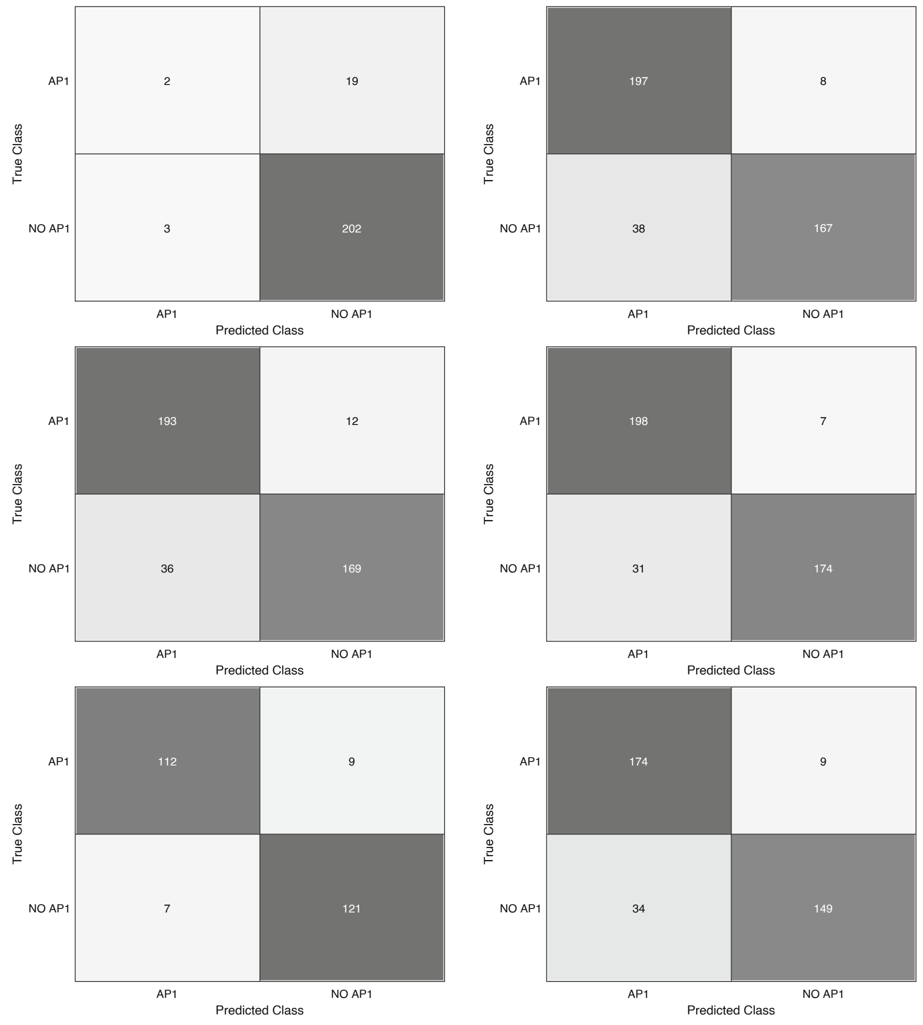


**Fig. 4.** Confusion matrix of EST1

techniques are applied for generating the models for the detection of four web service anti-patterns. A total of $6 \times 3 \times 5 \times 4 = 240$ predictive models are built for anti-pattern detection in this study. The predictive ability of these models are evaluated using Accuracy, and AUC performance values. Table 3 shows the Accuracy values for all the models generated. Figure 4 shows the confusion matrix obtained for the ensemble technique EST1 i.e. Bagging classifier. From Table 3 and Fig. 4, we observed that the:

- The performance values of the models trained on data after applying sampling techniques is better than models trained on the original data.

**Table 3.** Accuracy of all models

| Data Sampling Techniques | Feature Selection Techniques | Antipatterns | EST–1 | EST–2 | EST–3 | EST–4 | EST–5 | DataSampling Techniques | Feature Selection Techniques | Antipatterns | EST–1 | EST–2 | EST–3 | EST–4 | EST–5 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **ORG** | **SM** | **AP1** | 90.27 | 92.04 | 91.59 | 92.48 | 91.15 | **SVMSMOTE** | **SM** | **AP1** | 90.73 | 94.39 | 95.37 | 90.98 | 91.95 |
| **ORG** | **SM** | **AP2** | 94.25 | 95.13 | 95.13 | 93.81 | 91.59 | **SVMSMOTE** | **SM** | **AP2** | 86.61 | 95.83 | 96.13 | 92.26 | 93.75 |
| **ORG** | **SM** | **AP3** | 96.46 | 96.02 | 97.79 | 97.35 | 97.35 | **SVMSMOTE** | **SM** | **AP3** | 93.13 | 99.40 | 99.70 | 98.81 | 98.81 |
| **ORG** | **SM** | **AP4** | 95.13 | 94.69 | 96.02 | 93.81 | 93.36 | **SVMSMOTE** | **SM** | **AP4** | 90.83 | 96.33 | 96.64 | 92.97 | 94.80 |
| **ORG** | **PCA** | **AP1** | 90.71 | 89.82 | 92.48 | 91.59 | 90.71 | **SVMSMOTE** | **PCA** | **AP1** | 91.46 | 94.39 | 95.85 | 90.49 | 90.98 |
| **ORG** | **PCA** | **AP2** | 94.25 | 95.13 | 95.58 | 93.36 | 92.48 | **SVMSMOTE** | **PCA** | **AP2** | 84.82 | 94.64 | 96.73 | 88.69 | 86.31 |
| **ORG** | **PCA** | **AP3** | 94.69 | 93.81 | 94.69 | 94.69 | 88.94 | **SVMSMOTE** | **PCA** | **AP3** | 97.41 | 99.06 | 98.82 | 97.64 | 96.46 |
| **ORG** | **PCA** | **AP4** | 91.15 | 93.36 | 92.92 | 93.36 | 87.17 | **SVMSMOTE** | **PCA** | **AP4** | 89.51 | 95.61 | 96.10 | 89.02 | 91.71 |
| **ORG** | **RSA** | **AP1** | 91.15 | 90.71 | 92.92 | 90.71 | 90.71 | **SVMSMOTE** | **RSA** | **AP1** | 93.17 | 94.39 | 96.10 | 93.90 | 91.71 |
| **ORG** | **RSA** | **AP2** | 94.25 | 94.25 | 95.58 | 94.69 | 94.25 | **SVMSMOTE** | **RSA** | **AP2** | 89.88 | 94.94 | 97.32 | 88.69 | 88.99 |
| **ORG** | **RSA** | **AP3** | 94.69 | 97.79 | 98.23 | 96.90 | 90.71 | **SVMSMOTE** | **RSA** | **AP3** | 98.11 | 99.29 | 100.00 | 97.88 | 97.88 |
| **ORG** | **RSA** | **AP4** | 91.59 | 96.02 | 95.58 | 95.13 | 93.81 | **SVMSMOTE** | **RSA** | **AP4** | 91.74 | 96.02 | 96.64 | 95.41 | 94.80 |
| **SMOTE** | **SM** | **AP1** | 88.78 | 93.41 | 96.10 | 91.46 | 90.24 | **SMOTEENN** | **SM** | **AP1** | 93.57 | 99.60 | 99.20 | 98.39 | 97.59 |
| **SMOTE** | **SM** | **AP2** | 86.15 | 95.77 | 98.36 | 91.08 | 94.13 | **SMOTEENN** | **SM** | **AP2** | 88.52 | 94.43 | 97.05 | 94.43 | 88.52 |
| **SMOTE** | **SM** | **AP3** | 94.10 | 99.53 | 99.76 | 99.53 | 98.82 | **SMOTEENN** | **SM** | **AP3** | 93.69 | 99.40 | 99.40 | 99.70 | 98.50 |
| **SMOTE** | **SM** | **AP4** | 87.32 | 97.32 | 98.05 | 95.61 | 94.63 | **SMOTEENN** | **SM** | **AP4** | 87.40 | 98.09 | 98.47 | 95.04 | 96.18 |
| **SMOTE** | **PCA** | **AP1** | 89.02 | 94.15 | 95.85 | 91.22 | 89.76 | **SMOTEENN** | **PCA** | **AP1** | 93.31 | 96.94 | 98.33 | 90.53 | 93.59 |
| **SMOTE** | **PCA** | **AP2** | 84.98 | 95.77 | 97.18 | 90.14 | 89.20 | **SMOTEENN** | **PCA** | **AP2** | 90.60 | 95.04 | 97.39 | 90.60 | 92.43 |
| **SMOTE** | **PCA** | **AP3** | 95.99 | 99.29 | 99.53 | 98.82 | 97.88 | **SMOTEENN** | **PCA** | **AP3** | 97.54 | 99.26 | 99.26 | 99.51 | 97.04 |
| **SMOTE** | **PCA** | **AP4** | 88.29 | 95.61 | 95.37 | 91.22 | 91.95 | **SMOTEENN** | **PCA** | **AP4** | 94.54 | 97.27 | 98.36 | 91.26 | 92.90 |
| **SMOTE** | **RSA** | **AP1** | 90.24 | 93.66 | 95.61 | 90.00 | 89.02 | **SMOTEENN** | **RSA** | **AP1** | 98.58 | 98.58 | 98.94 | 98.23 | 97.87 |
| **SMOTE** | **RSA** | **AP2** | 90.85 | 95.07 | 97.18 | 88.03 | 89.44 | **SMOTEENN** | **RSA** | **AP2** | 91.62 | 96.22 | 97.57 | 90.27 | 91.08 |
| **SMOTE** | **RSA** | **AP3** | 97.88 | 99.29 | 99.76 | 98.58 | 98.11 | **SMOTEENN** | **RSA** | **AP3** | 98.28 | 98.53 | 98.53 | 99.26 | 99.75 |
| **SMOTE** | **RSA** | **AP4** | 91.46 | 95.37 | 96.83 | 95.12 | 93.41 | **SMOTEENN** | **RSA** | **AP4** | 94.13 | 97.21 | 99.72 | 97.77 | 94.97 |
| **BSMOTE** | **SM** | **AP1** | 88.29 | 92.93 | 95.37 | 91.95 | 89.76 | **SMOTETOMEK** | **SM** | **AP1** | 88.25 | 93.44 | 95.36 | 94.81 | 92.62 |
| **BSMOTE** | **SM** | **AP2** | 84.51 | 96.01 | 97.89 | 92.96 | 91.55 | **SMOTETOMEK** | **SM** | **AP2** | 87.43 | 96.34 | 97.64 | 91.88 | 90.84 |
| **BSMOTE** | **SM** | **AP3** | 94.58 | 99.29 | 100.00 | 99.29 | 98.35 | **SMOTETOMEK** | **SM** | **AP3** | 94.06 | 99.50 | 100.00 | 99.26 | 98.27 |
| **BSMOTE** | **SM** | **AP4** | 90.24 | 96.83 | 97.80 | 95.12 | 96.34 | **SMOTETOMEK** | **SM** | **AP4** | 87.77 | 97.87 | 97.61 | 95.48 | 96.54 |
| **BSMOTE** | **PCA** | **AP1** | 90.98 | 95.85 | 96.34 | 93.41 | 91.71 | **SMOTETOMEK** | **PCA** | **AP1** | 90.49 | 94.15 | 95.85 | 91.22 | 89.76 |
| **BSMOTE** | **PCA** | **AP2** | 84.51 | 95.54 | 97.18 | 90.14 | 89.20 | **SMOTETOMEK** | **PCA** | **AP2** | 88.03 | 97.42 | 97.18 | 90.14 | 89.20 |
| **BSMOTE** | **PCA** | **AP3** | 96.93 | 100.00 | 99.53 | 98.82 | 97.17 | **SMOTETOMEK** | **PCA** | **AP3** | 95.75 | 99.29 | 99.53 | 98.82 | 97.88 |
| **BSMOTE** | **PCA** | **AP4** | 88.05 | 94.39 | 96.10 | 90.73 | 91.46 | **SMOTETOMEK** | **PCA** | **AP4** | 85.61 | 96.83 | 95.37 | 91.22 | 91.95 |
| **BSMOTE** | **RSA** | **AP1** | 88.29 | 90.73 | 94.88 | 89.76 | 90.00 | **SMOTETOMEK** | **RSA** | **AP1** | 92.27 | 94.33 | 95.36 | 89.18 | 90.98 |
| **BSMOTE** | **RSA** | **AP2** | 88.26 | 95.77 | 97.18 | 88.97 | 88.97 | **SMOTETOMEK** | **RSA** | **AP2** | 89.20 | 94.84 | 97.18 | 88.03 | 89.44 |
| **BSMOTE** | **RSA** | **AP3** | 99.06 | 98.82 | 100.00 | 98.35 | 98.11 | **SMOTETOMEK** | **RSA** | **AP3** | 99.05 | 98.58 | 99.05 | 98.82 | 98.58 |
| **BSMOTE** | **RSA** | **AP4** | 92.44 | 96.59 | 97.80 | 94.88 | 96.10 | **SMOTETOMEK** | **RSA** | **AP4** | 93.25 | 96.75 | 97.75 | 95.75 | 93.25 |

– SMOTEENN is showing the best performance, while the model developed using the original data (ORG) is showing the worst performance.
– The model trained using features selected by PCA as input have better performance.
– The performance of the model trained using EST3 i.e. Extra Trees classifier with a mean accuracy of 97.13 is higher when compared to the models trained using other ensemble techniques.

## 6 Competitive Analysis

In this section, we compare the performance of the various models generated using Box-plots, Descriptive statistics and Wilcoxon ranksum test.

### 6.1 Data Sampling Techniques

Figure 5 depicts the performance values, i.e., Accuracy, AUC, and F-measure of the models developed using different variants of SMOTE using Box-plot diagrams. Table 4 shows the descriptive statistics for the different SMOTE techniques used in this study. From Fig. 5 and Table 4, we infer that the technique SMOTEENN is showing the best performance, with 0.989 mean, 1.000 max, 0.986 Q1 and 0.999 Q3 AUC values. The model developed using the original data(ORG) is showing the worst performance with the AUC value of 0.823. It is also observed that the model developed with the dataset after applying the data sampling technique(any) is showing better performance when compared to the model developed using the original data, as the sampling technique deals with the class imbalance problem.

**Table 4.** Descriptive statistics of data sampling techniques

| | Accuracy | | | | | |
|---|---|---|---|---|---|---|
| | Max | Min | Median | Mean | Q3 | Q1 |
| ORG | 98.23 | 87.17 | 93.81 | 93.60 | 95.13 | 91.59 |
| SMOTE | 99.76 | 84.98 | 95.10 | 94.11 | 97.60 | 90.54 |
| BSMOTE | 100.00 | 84.51 | 95.24 | 94.20 | 97.49 | 90.73 |
| SVMSMOTE | 100.00 | 84.82 | 94.80 | 94.20 | 96.68 | 91.59 |
| SMOTEENN | 99.75 | 87.40 | 97.33 | 96.00 | 98.53 | 93.64 |
| SMOTETOMEK | 100.00 | 85.61 | 95.36 | 94.37 | 97.70 | 91.10 |

**Table 5.** P-value: data sampling techniques

| | ORG | SMOTE | BSMOTE | SVMSMOTE | SMOTEENN | SMOTETOMEK |
|---|---|---|---|---|---|---|
| ORG | 1 | 0 | 0 | 0 | 0 | 0 |
| SMOTE | 0 | 1 | 1 | 0 | 0 | 1 |
| BSMOTE | 0 | 1 | 1 | 0 | 0 | 1 |
| SVMSMOTE | 0 | 0 | 0 | 1 | 0 | 0 |
| SMOTEENN | 0 | 0 | 0 | 0 | 1 | 0 |
| SMOTETOMEK | 0 | 1 | 1 | 0 | 0 | 1 |

Wilcoxon signed-rank test is used in this study for statistically comparing the performance of the various web service anti-pattern prediction models developed using different variants of smote sampled data and the original data. The main

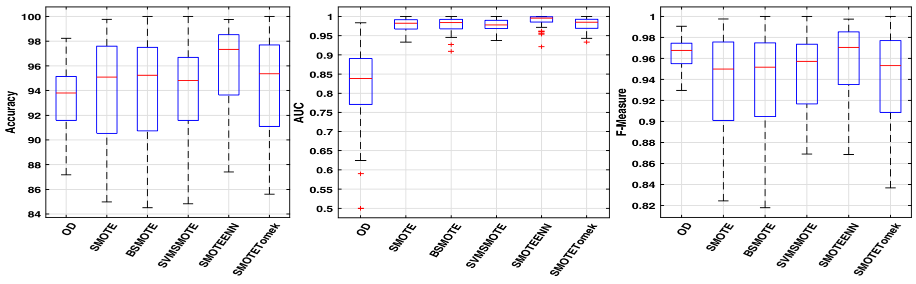


**Fig. 5.** Box-plot: data sampling techniques

motivation of the wilcoxon signed-rank test is to find whether there is a significant difference between the performance of the various models developed using different Smote sampled data or not. The null hypothesis considered for this paper is: "The web service anti-pattern prediction model trained using different variants of smote sampled data are not significantly different". The considered null hypothesis is accepted, if the p-value obtained using the wilcoxon signed-rank test is '1'. Table 5 shows the p-values obtained for the models developed using all the data sampling techniques along with the original dataset. A close inspection of Table shows that most of the comparison points are having p-values as '0', i.e., the considered hypothesis is rejected. Hence we conclude that there is a significant difference between the performance of the models generated using different variants of smote sampled data and the original data.

## 6.2 Feature Selection Techniques

Figure 6 depicts the performance values, i.e., Accuracy, AUC, and F-measure of the models developed using the features selected by different feature selection techniques as input using Box-plot diagrams. Table 6 shows the descriptive statistics for the various feature selection techniques used in this study. Table 6 show that the mean value of the model developed using the features selected by PCA as input is higher than the models developed using the features selected by rank-sum test, and RSA as input. From Fig. 6, we observe that the inter-quartile range for AUC value for the model generated using PCA is compa ratively small when compared to the other models. This indicates that the performance parameters obtained using multiple executions in PCA are showing less variation when compared to other models.

The null-hypothesis considered in this section is: "The performance of the anti-pattern prediction models developed using features selected by different feature selection techniques are significantly different." The defined null-hypothesis is accepted, if the p-value obtained using the wilcoxon signed rank test is '1'. Table 7 shows the p-values of the models developed using various combination of features as input. From Table 7, we observed that most of the comparison points have p-value as '1' i.e. the defined null-hypothesis is accepted. Therefore, there is

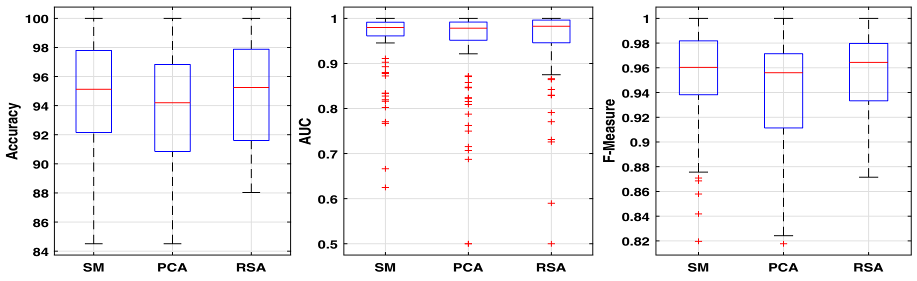


**Fig. 6.** Box-plot: feature selection techniques

**Table 6.** Descriptive statistics of feature selection techniques

| | Accuracy | | | | | |
|---|---|---|---|---|---|---|
| | Max | Min | Median | Mean | Q3 | Q1 |
| SM | 100.00 | 84.51 | 95.13 | 94.71 | 97.80 | 92.15 |
| PCA | 100.00 | 84.51 | 94.20 | 93.72 | 96.83 | 90.85 |
| RSA | 100.00 | 88.03 | 95.25 | 94.84 | 97.88 | 91.61 |

no significant difference between the performance of the models builds utilizing the features selected by using three different feature selection techniques.

**Table 7.** P-value: feature selection

| | SM | PCA | RSA |
|---|---|---|---|
| SM | 1 | 0 | |
| PCA | 0 | 1 | 0 |
| RSA | 1 | 0 | 1 |

## 6.3 Classifier Techniques

Figure 7 shows the box-plot diagram of the AUC, Accuracy and F-measure of the classifier techniques. Table 8 shows the descriptive statistics for the models trained using distinct ensemble techniques. From Table 8 and Fig. 7, we observed that the performance of the model trained using EST3 i.e. Extra Trees classifier is higher when compared to the models trained using other ensemble techniques. The model trained using EST3 is showing good performance with a mean accuracy of 97.13, median accuracy 97.18 and min accuracy of 91.59.

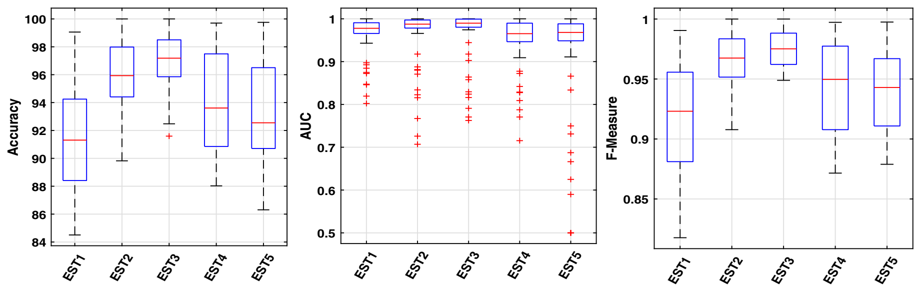


**Fig. 7.** Box-plot: classifier techniques

**Table 8.** Statistical description for ensemble techniques

| | Accuracy | | | | | |
|---|---|---|---|---|---|---|
| | Max | Min | Median | Mean | Q3 | Q1 |
| EST1 | 99.06 | 84.51 | 91.31 | 91.67 | 94.25 | 88.41 |
| EST2 | 100.00 | 89.82 | 95.93 | 96.11 | 97.98 | 94.41 |
| EST3 | 100.00 | 91.59 | 97.18 | 97.13 | 98.50 | 95.85 |
| EST4 | 99.70 | 88.03 | 93.61 | 93.88 | 97.49 | 90.85 |
| EST5 | 99.75 | 86.31 | 92.55 | 93.27 | 96.50 | 90.71 |

The null-hypothesis considered in this work is: "The performance of the anti-pattern prediction models trained using various ensemble techniques are significantly different. " The defined null-hypothesis is accepted, if the p-value obtained using the wilcoxon signed rank test is '$1'$ and is rejected if the p-value is '0'. Table 9 shows the p-values of the models trained using different ensemble techniques. From Table 9, we observed that most of the comparison points have p-value as '0' i.e. the defined null-hypothesis is rejected. Hence we conclude that there is a significant difference between the performance of the models trained using various ensemble techniques.

**Table 9.** P-value: ensemble techniques

| | EST1 | EST2 | EST3 | EST4 | EST5 |
|---|---|---|---|---|---|
| EST1 | 1 | 0 | 0 | 0 | 0 |
| EST2 | 0 | 1 | 0 | 0 | 0 |
| EST3 | 0 | 0 | 1 | 0 | 0 |
| EST4 | 0 | 0 | 0 | 1 | 1 |
| EST5 | 0 | 0 | 0 | 1 | 1 |

## 7 Conclusion

We present the empirical analysis on anti-pattern prediction models developed using data sampling, feature selection and ensemble techniques. Five-fold cross validation is used for validating the performance of the models built. We used three performance parameters i.e. Accuracy, F-measure and AUC to compare the performance of the models built. We observed that the performance values of the models trained on data after applying sampling techniques is better than models trained on the original data. Wilcoxon sign rank test suggested that model trained using balanced data have significant improvement in predicting anti-patterns. It is observed that the performance of the model trained using features selected by PCA as input have better performance. Wilcoxon sign rank test suggested that there is no significant difference between the performance of the models builds utilizing the features selected by using three different feature selection techniques. We also observe that the model trained using EST3 i.e. Extra Trees classifier is showing good performance with a mean accuracy of 97.13.

## References

1. Batista, G.E., Prati, R.C., Monard, M.C.: A study of the behavior of several methods for balancing machine learning training data. ACM SIGKDD Explor. Newsl. **6**(1), 20–29 (2004)
2. Chidamber, S.R., Kemerer, C.F.: A metrics suite for object oriented design. IEEE Trans. Softw. Eng. **20**(6), 476–493 (1994)
3. Jaafar, F., Guéhéneuc, Y.-G., Hamel, S., Khomh, F., Zulkernine, M.: Evaluating the impact of design pattern and anti-pattern dependencies on changes and faults. Empir. Softw. Eng. **21**(3), 896–931 (2015). https://doi.org/10.1007/s10664-015-9361-0
4. Král, J., Zemlicka, M.: Crucial service-oriented antipatterns, vol. 2, pp. 160–171. International Academy, Research and Industry Association (IARIA) (2008)
5. Kumar, L., Sureka, A.: An empirical analysis on web service anti-pattern detection using a machine learning framework. In: 2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC), vol. 1, pp. 2–11. IEEE (2018)
6. Moha, N., et al.: Specification and detection of SOA antipatterns. In: Service-Oriented Computing, pp. 1–16 (2012)
7. Ordiales Coscia, J.L., Mateos, C., Crasso, M., Zunino, A.: Anti-pattern free code-first web services for state-of-the-art Java WSDL generation tools. Int. J. Web Grid Serv. **9**(2), 107–126 (2013)
8. Ouni, A., Kula, R.G., Kessentini, M., Inoue, K.: Web service antipatterns detection using genetic programming. In: Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation, pp. 1351–1358. ACM (2015)
9. Palma, F., Nayrolles, M., Moha, N., Guéhéneuc, Y.G., Baudry, B., Jézéquel, J.M.: SOA antipatterns: an approach for their specification and detection. Int. J. Coop. Inf. Syst. **22**(04), 1341004 (2013)
10. Pawlak, Z.: Rough Sets: Theoretical Aspects of Reasoning About Data, vol. 9. Springer Science and Business Media (2012). https://doi.org/10.1007/978-94-011-3534-4
11. Saluja, S., Batra, U.: Optimized approach for antipattern detection in service computing architecture. J. Inf. Optim. Sci. **40**(5), 1069–1080 (2019)
12. Settas, D., Cerone, A., Fenz, S.: Enhancing ontology-based antipattern detection using Bayesian networks. Expert Syst. Appl. **39**(10), 9041–9053 (2012)
13. Settas, D.L., Meditskos, G., Stamelos, I.G., Bassiliades, N.: SPARSE: a symptom-based antipattern retrieval knowledge-based system using semantic web technologies. Expert Syst. Appl. **38**(6), 7633–7646 (2011)
14. Tummalapalli, S., Kumar, L., Neti, L.B.M.: An empirical framework for web service anti-pattern prediction using machine learning techniques. In: 2019 9th Annual Information Technology, Electromechanical Engineering and Microelectronics Conference (IEMECON), pp. 137–143. IEEE (2019)
15. Upadhyaya, B., Tang, R., Zou, Y.: An approach for mining service composition patterns from execution logs. J. Softw. Evol. Process **25**(8), 841–870 (2013)