Detection of SLA Violation for Big Data Analytics Applications in Cloud

Xuezhi Zeng, Saurabh Garg, Mutaz Barika, Sanat Bista, Deepak Puthal, Albert Y. Zomaya, Rajiv Ranjan

Abstract—SLA violations do happen in real world. An SLA violation represents the failure of guaranteeing a service, which leads to unwanted consequences such as penalty payments, profit margin reduction, reputation degradation, customer churn and service interruptions. Hence, in the context of cloud-hosted big data analytics applications (BDAAs), it is paramount for providers to predict and prevent SLA violations. While machine learning-based techniques have been applied to detect SLA violations for web service or general cloud service, the study on detecting SLA violations dedicated for cloud-hosted BDAAs is still lacking. In this paper, we propose four machine learning techniques and integrate 12 resampling methods to detect SLA violations for batch-based BDAAs in the cloud. We evaluate the efficiency of the proposed techniques in comparison with ideal and baseline classifiers based on a real-world trace dataset (Alibaba). Our work not only helps providers to choose the best performing prediction technique, but also provides them capabilities to uncover the hidden pattern of multiple configurations of BDAAs across layers.

Index Terms—Big Data; Big Data Analytics Application; Service Level Agreement; Machine Learning; Resampling; Service Layer; SLA Violation; Neural Network

1 Introduction

Service Level Agreement (SLA) that represents the contract between providers and customers is one of the effective methods to manage and guarantee the quality of service (QoS) promised. Nowadays, in the context of big data analytics applications (BDAAs) in the cloud, SLAs play an integral role in governing the relationships between providers and customers. Besides setting the expectations by dictating the quality and the type of service, SLAs are also increasingly considered as a strong differentiator allowing a provider to offer different levels of service guarantees and to differentiate itself from its competitors.

SLAs are very important for both parties. On the one hand, the provider needs to avoid having penalties due to failure of providing the agreed service. On the other hand, the customer favors on-demand service and without any interruptions. The failure of guaranteeing a service leads to unwanted consequences such as penalty payments, profit margin reduction, reputation degradation, customer churn and service interruptions. We mean by profit margin, the provider revenue minus all expenses, so the the profit margin reduction is the decline in its profit. Customer churn means the present of customers discontinue doing business with the organization. This failure is called SLA violations. SLA violations do happen in real world and have caused both providers and customers heavy costs. Hence, it is paramount for providers to predict and prevent SLA violations for BDAAs as much as possible before they happen. However, accurate prediction of SLA violation for BDAAs is extremely challenging with manifold reasons:

- The violation status of BDAA jobs is actually determined by multiple configurations across different layers (see Figure 1). For example, if the user submits a job at BDSaaS layer that required high CPU or memory resources along with the requirements of low response time and deployment costs, the consideration of SLAs at platform and infrastructure levels are required. If BDPaaS layer allocates instances for batch processing this job, but it fails to achieve the execution cost requirement and/or select and configure big data processing platform to meet user-defined response time, this could impact considerably the freshness of this batch processing job and result in the violation. In this case, the violation status is specified by detecting the root causes of such violation across different layers based on the logs of both data flow and QoS metrices. Accordingly, the configurations at each layer are extremely important and determine the violation status of BDAA jobs. However, the hidden pattern between these multiple configurations across BDAA layers regarding the status of SLA violation is far from clear and needs to be well uncovered and utilized.
- In the real world of SLA management in BDAAs, the SLA violation is very rare event, which results in data skewness meaning that the number of violated jobs are much less than the number of unviolated one. This could cause deficient classification models and bring lots of challenges for most supervised learning algorithms because they tend to minimize loss by labeling every sample with the majority class(es),

- *Xuezhi Zeng is with the Research School of Computer Science, The Australian National University; e-mail: xuezhi.zeng@anu.edu.au*
- *Saurabh Garg and Mutaz Barika are with the Discipline of ICT - School of Technology, Environments and Design (TED), University of Tasmania: e-mail: saurabh.garg@utas.edu.au; mutaz.barika@utas.edu.au*
- *Sanat Bista is with Australian Government agency. e-mail: sanat.bista@gmail.com*
- *Albert Y. Zomaya is with the School of Computer Science, The University of Sydney; e-mail: albert.zomaya@sydney.edu.au*
- *Deepak Puthal and Rajiv Ranjan are with the School of Computing, Newcastle University. e-mail: dputhal88@gmail.com; rranjans@gmail.com*

leading to poor recall on the minority class(es). However, true misclassification costs may be much greater when minority class instances are missed. e.g., incorrectly predicting the actual "violated" jobs will cause the degradation of reputation and loss of profit for providers.

To address the aforementioned challenges, we explore 4 representative machine learning techniques (i.e., Logistic Regression, Artificial Neural Network, Random Forest, Extreme Gradient Boosting) with the integration of 12 resampling techniques (grouped by oversampling, undersampling and combination) to detect SLA violations for batch-based BDAAs workload in cloud. We also conduct extensive experiments in a real dataset to evaluate the efficiency of the proposed techniques.

The remainder of this paper is structured as follows. In Section 2, we present our research question and methodology. Section 3 presents the layer architecture of BDAAs. Section 4 discusses related works. Section 5 mathematically formulates the SLA violation detection problem. In Section 6, we discuss how we extract multiple features across layers and define SLA violation status based on the Alibaba trace dataset. Section 7 introduces four machine learning-based prediction models and articulates their fundamental working mechanism in terms of prediction. In Section 8, twelve diverse resampling techniques are presented. We then discuss 5 evaluation metrics in Section 9. In Section 10, extensive experiments results are presented and major findings are analysed. We further discuss a novel mathematical model for the providers' profit in Section 11. Section 12 analyses and discusses the experiment results based on the above mathematical model. In Section 13, We conclude our paper by pointing out two potential future directions.

2 Research Question and Methodology

2.1 Research Question

In this paper, we addressed the challenging question that is how to detect SLA violations for cloud-hosted big data analytics application across layers before they happen to maximize the providers' profit.

2.2 Research Methodology and Contributions

The question of SLA violation detection has been addressed previously in the context of web service and general cloud service. Our methodology differentiates itself in the following aspects: (i) we choose a newly released workload trace dataset published by Alibaba, which is a good representation of batch-based BDAA workload; (ii) we detect SLA violations for this batch-based BDAA through the exploration of the four representative machine learning techniques (i.e., Logistics Regression, Artificial Neural Network, Random Forest, and Extreme Gradient Boosting); (iii) we apply 12 diverse resampling techniques (5 oversampling, 5 undersampling and 2 combined resampling) into the above four machine learning predictors to handle data skewness problem; (iv) we conduct extensive experiments to evaluate the efficiency of SLA violation detection based on five metrics (i.e., accuracy, precision, recall, F_2 score, and ROC); (v) we design a mathematical model to formulate the provider's profit and investigate how the outcome of a prediction technique impact providers' profit. This work is one of the first attempts towards detection of SLA violations dedicated for big data analytics applications through an integrated application of machine learning and resampling techniques. This work will help providers to choose the best performing prediction technique, and most importantly, it can uncover the hidden patterns of the multiple configurations of BDAAs across layers and hence provide insightful information for providers' decision-making process.

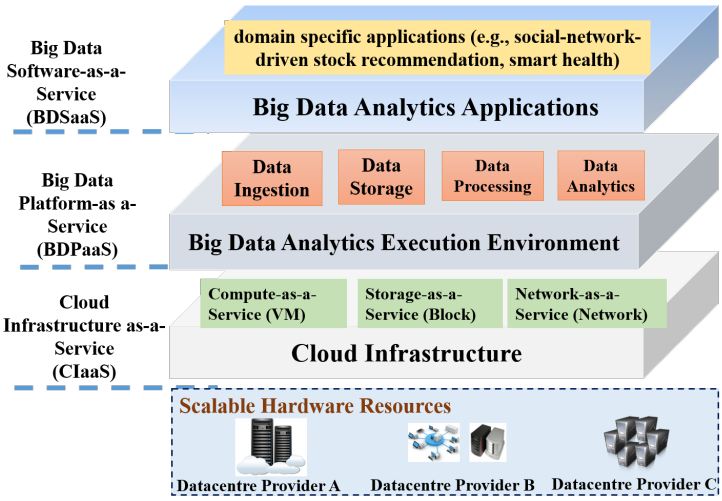


Fig. 1. The layer-based architecture of big data analytics applications in cloud

3 Layer-based Architecture of Big Analytics Applications in Cloud

According to the works in [1], [2], [3], [4], [5], a typical cloud-hosted BDAA spans multiple layers. Each layer serves a different function and consists of different components/frameworks. We give a pictorial representation of a layer-based architecture for cloud-hosted BDAAs, which is shown in Figure 1.

It is observed that there are three layers from top to bottom: Big Data Software as a Service (BDSaaS), Big Data Platform as a Service (BDPaaS) and Cloud Infrastructure as a Service (CIaaS). Beyond the top level are usually end users who request analytics service through the interface. It is not difficult to understand that an end user is a client of the BDSaaS, where BDPaaS and CIaaS are service providers to BDSaaS. The BDPaaS provider provides big data analytics platforms, while CIaaS provider provides scalable hardware resources in a virtualized environment. The details of each layer are described in the Appendix A.

4 Related Works

Many machine learning-based approaches have been applied in recent years to tackle SLA violation prediction problem.

Leither et al. [6] proposed an approach to predict SLA violations in runtime mode for compound web services. They built a regression-based prediction model that takes typical service quality data and process instance data as input, such as availability, system workload, response time, ordered products and customer identifiers. They implemented their prediction model by using a fully Java-based machine learning toolkit called WEKA. However, this toolkit is not

scalable to suit the situation in real world, where the scale of the dataset is much bigger in comparison to the one which is used in this paper. Moreover, the layered big data analytics application framework is fundamentally different from their composite web services.

Jules et al. [7] proposed a Bayesian Network-based model that calculates and constantly updates the reputation of a trusted provider. Moreover, they introduced a probabilistic ontology-based technique that can forecast SLA violations in terms of contract parameters. Although their approach achieved a decent performance, the dataset is artificially produced by simulation, which has defect in representing a real application workload. For example, SLA violations occupy 40% in their generated dataset, which disregards the reality that SLA violations are very rare (< 10%) in real application deployment scenarios.

The authors in the paper [8] focused on predicting SLA violations for cloud services. They built a prediction model that uses Naive Bayesian algorithm and takes service quality datasets regarding historically measured web service as the input. In this paper, they investigated and validated the most valid feature combinations for prediction. Still, their web application workloads are fundamentally different from big data analytics application workloads and hence is not applicable to predict SLA violations for cloud-hosted BDAAs.

Hemmat et al. [9] systematically compared the performance of two machine learning classification models for predicting SLA violation by analyzing Google cluster trace dataset [10]. They explore several methods of handling unbalanced data. Despite its good performance, the authors resample the training data before the cross validation, which leads to the problem of information leakage and overfitting. Also, the features they extracted are task-oriented rather than application as a whole. Hence, the prediction outcome of SLA violation is actually for tasks, which is not what a provider really cares about.

Uriarte et al. [11] performed their SLA violation approach using Google Cluster trace dataset as well. They cluster the resource usage and duration of services using an unsupervised learning-based technique. If a service in a cluster is detected a violation, the prospective resources will be allocated to the other services hosted in the same cluster in order to avoid the further violation. Although it is helpful for avoiding violation in this cluster, explicit violation forecast towards each service is lacked.

In summary, our work compared with the aforementioned works focuses on specific big data application type (i.e. batch-based BDAA) and integrates different diverse skewness handling techniques in machine learning algorithms along with the consideration of provider profit.

5 PROBLEM FORMULATION OF SLA VIOLATION DETECTION

We formulate the detection of SLA violation for cloud-hosted BDAAs as a binary classification problem and define the notations to describe this problem in Table 1.

Given a set of features extracted from the Alibaba cloud trace (see Section 5), what is the probability of failure of a submitted batch job, which results in SLA violation, or the probability that this batch job is successfully processed without SLA violation?

TABLE 1
Notation used in formulating SLA violation detection problem

Symbols	Description
X	Represents a space containing descriptions of batch jobs. $X = (X_1; X_2; :::; X_m)$, where X_m statistically describes the property of a batch job from an aspect, such as the requested CPU configuration (e.g., speed, number of cores), or requested memory and so on
Y	Represents a space labeling SLA violation status. For the binary problem, there are only two classes. Y = [0, 1]. Y=0 means the batch job is unviolated and Y=1 means the batch job is violated
n	The number of observed samples
D_n	Denotes the training dataset. $D_n = [(x_1; y_1); :::; (x_i; y_i); i = 1; 2; :::; n] \in (X; Y)$

A binary classification problem aims to find a function $: X \; ! \; Y$ based on D_n that gives a new sample $X_{new} \; 2 \; X$, predicts $\hat{Y} = \; (X_{new}) \; 2 \; Y$.

6 DATASET

In this paper, we pursue our exploration on a BDAA workload dataset released by Alibaba in the September of 2017 [12]. As a leading public cloud platforms over the world, Alibaba Cloud is running millions of batch jobs or online services across hundreds of datacenters every day using the latest big data technologies. This dataset contains a workload trace of a BDAA cluster. Each trace file includes the job statistics over 12 hours and has information about 1300 virtual machines that execute both offline batch jobs and online service in this cluster. As regards to the batch processing BDAA workloads, the trace details the information including job ids, task ids, instances types, and machines' hardware configuration. To the best of our knowledge, this dataset has not been extensively utilized by the research community. Lu et al. [13] performed a characterization of the Alibaba cloud trace and disclosed four types of imbalance (i.e., spatial imbalance, temporal imbalance, proportion imbalance of resources utilization per workload, and resource demands and runtime statistics imbalance). The work in [14] investigates the elasticity and plasticity of resource allocation of the Alibaba trace. The authors in [15] focused on providing a unique and microscopic view about how the co-located workloads interact and impact each other.

In the Alibaba dataset, three types of information are available as regards to job deployment and execution including machine utilization, runtime batch processing job workload information, and runtime online service workload. For the sake of confidentiality, Alibaba has excluded or obfuscated part of the dataset. For instance, they normalized the values regarding the utilization of disk and memory. Service workload is given a numeric id, which is unique in the trace period. No service and task names are given. In this paper, we focus on the batch workload information whose entity relationship diagram is shown in the Figure 1 of Appendix B.

It is seen that batch workloads are depicted in two tables: "Batch Task" table and "Batch Instance" table. A batch workload is submitted by the user in the form of a job. Each job comprises various tasks that were submitted to the cluster and forms a directed acyclic graph (very

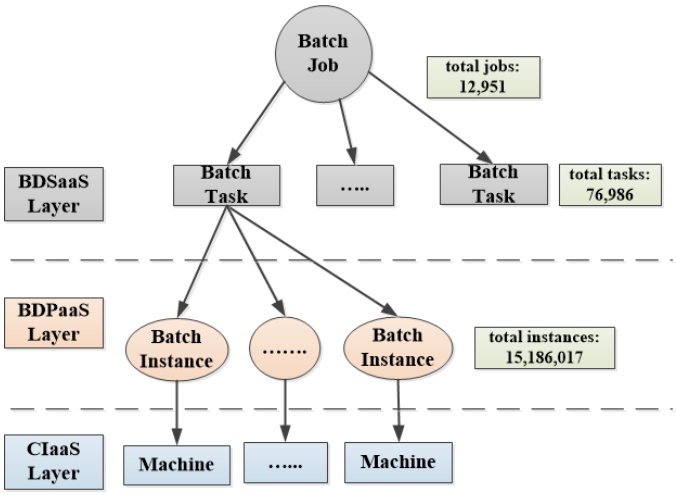


Fig. 2. The structure of batch workload in Alibaba trace dataset

similar to MapReduce execution graph) according to the data dependency. The event cycle of the tasks inside each job are traced in the "Batch Task" table. Each task consists of numerous instances and executes diverse computing logic. The instance is the smallest scheduling unit of the batch workload in Alibaba's cloud platform. All instances inside a task process the identical binary code with same multiple resource demands, but executing different fragments of data. Such execution flow is in line with the MapReduce programming model. Each instance within a task pertains to one job and is then assigned to a specific cluster computing machine that utilizes a Linux container to execute that task instance. While a majority of batch processing job can include hundreds of tasks instances, few selected ones can also include an extremely large number of tasks instances. For example, the research work in [13] reported that a job had 60000+ task instances. In our research, the metadata related to tasks instances is tracked in "Batch Instance" table. The meta information of machine in an Alibaba cluster is captured in the following two tables (i.e., "Machine Events" and "Machine Utilization" table). The "Machine Events" table shows three types of events (e.g., addition, soft error or hard error), and reflects the normalized physical capacity of each machine along the aspects in terms of RAM size and CPU cores. "Machine Utilization" table records the attributes of each machine such as machine ID, utilization of CPU, utilization of memory and so on.

The configurations of job, task, instance and machine are critical elements for batch workloads in BDAAs across different layers. The structure of batch workloads running on Alibaba cluster machines is demonstrated in Figure 2.

6.1 Feature Extraction

From Alibaba dataset, we notice that there are multiple jobs, where each job depends on its multiple tasks. The computing logic requirements of each task need to be achieved by scheduling its instances on cluster computing machines. In other words, each task has its own requirements including number of instances, and machine CPU and memory required for each instance. This means that failing to achieve the machine CPU and memory requested by any instance in a task contributes to SLA violation. Similarly, multiple tasks within a job contribute to SLA violation in case of the requirements of any task does not meet.

To apply machine learning techniques to detect SLA violation, we need to extract features based on the batch workload information in this dataset. Since a job consists of multiple tasks, for generality, let a job $J = (T_1; :::; T_i; :::; T_m)$. For each task T_i in J, it consists of multiple instances. Let $T_i = (Inst_1^i; Inst_2^i; :::; Inst_k^i; :::; Inst_N^i)$, where N_i denotes the number of instances of T_i always takes an integer value 1.

Let $Inst_k^i(real\_cpu\_max)$ denotes the maximum CPU numbers of actual instance running for the instance, $Inst_k^i(real\_cpu\_avg)$ denotes the average CPU numbers of actual instance running for the instance, $Inst_k^i(real\_mem\_max)$ denotes the maximum normalized memory for the instance, $Inst_k^i(real\_mem\_avg)$ denotes the average normalized memory for the instance. $Inst_k^i(cpu\_capacity)$ denotes the normalized physical CPU capacity of the machine that has been used by the instance, $Inst_k^i(mem\_capacity)$ denotes the normalized physical memory capacity of the machine that has been used by the instance , $T_i(cpu\_requested)$ denotes the CPU requested for each instance of the task, and $T_i(mem\_requested)$ denotes the normalized memory requested for each instance of the task. Then, for each job J, we do a set of mathematical aggregation operations on the above variables in order to extract features across layers. Finally, ten features are generated as follows (all the features have been normalised based on the number of tasks):

BDSaaS layer:

$$X_1 : cpu\_requested\_per\_job = \frac{\sum_{i=1}^{m} T_i(cpu\_requested)}{m}$$

$$X_2 : memory\_requested\_per\_job = \frac{\sum_{i=1}^{m} T_i(memory\_requested)}{m}$$

$$X_3 : num\_tasks\_per\_job = m$$

BDPaaS layer:

$$X_4 : num\_instances\_per\_job = \frac{\sum_{i=1}^{m} N_i}{m}$$

$$X_5 : real\_cpu\_max\_per\_job = \frac{\sum_{i=1}^{m} \sum_{k=1}^{N_i} Inst_k^i(real\_cpu\_max)=N_i}{m}$$

$$X_6 : real\_cpu\_avg\_per\_job = \frac{\sum_{i=1}^{m} \sum_{k=1}^{N_i} Inst_k^i(real\_cpu\_avg)=N_i}{m}$$

$$X_7 : real\_mem\_max\_per\_job = \frac{\sum_{i=1}^{m} \sum_{k=1}^{N_i} Inst_k^i(real\_mem\_max)=N_i}{m}$$

$$X_8 : real\_mem\_avg\_per\_job = \frac{\sum_{i=1}^{m} \sum_{k=1}^{N_i} Inst_k^i(real\_mem\_avg)=N_i}{m}$$

CIaaS layer:

$$X_9 : cpu\_capacity\_per\_job = \frac{\sum_{i=1}^{m} \sum_{k=1}^{N_i} Inst_k^i(cpu\_capacity)=N_i}{m}$$

$$X_{10}: mem\_capacity\_per\_job = \frac{\sum_{i=1}^{m}\sum_{k=1}^{N_i} Inst_k^i(mem\_capacity)}{m}$$

As high level features, the above ten features (X_1 X_{10}) provide an effective representation of each batch job J. We could also consider other criterion such as Linux CPU load, disk space requested or running trials number as features. However, we refrain from using redundant features for the purpose of preventing the model suffering from overfitting.

6.2 SLA Violation Definition

Besides the features, we also need to formulate the target SLA violation status [0: "non-violated", 1: "violated"] for each batch J. According to the dataset, there are four main types of states for a batch task:

- Terminated: a task goes to 'Terminated' when all its instances are done, meaning this batch task successfully processed
- Waiting: a task is not initialized yet
- Failed: a task fails
- Running: a task is being processed

In this dataset, most of tasks are terminated in a normal mode, while 2000 plus have "waiting" state. Since a job consists of multiple tasks, we can set the criteria that for one job, only the state of all of its tasks is "Terminated", the job is regarded as "Terminated". If one of its task's state is "Failed", the job is regarded as "Failed". Similarly, if one of its task' state is "Running", the job is regarded as "Running".

There are 12951 jobs in total. We have performed data quality check and pre-processing work as the foremost step. We remove some "abnormal" jobs that have corresponding tasks with missing values in the dataset, and then drop some "abnormal" jobs that have latest finished task is earlier than the earliest created tasks. After this, 11897 valid jobs remain, where 10774 jobs are labeled "Terminated" (successfully processed), and 1123 jobs are labeled "Failed" based on the above criteria.

In order to determine SLA violations, we need to have particular details in terms of the service quality and service level objectives. Although such information is not available in this dataset, we could discover SLA violation of a job according to the availability of its tasks. Specifically, we detect a job as "violated" if at least one of its corresponding tasks is failed and unusable, representing this job failure causing loss for both providers and customers. Accordingly, we detect a job as "non-violated" if all tasks of this job are terminated normally,representing that this job is successfully finished.

Based on this SLA violation definition, the percentage of "non-violated" batch jobs (10774 in total) among all the valid jobs (11897 in total) is 90.56%, and hence the ratio of SLA violation is only 9.44%. We can conclude that the quantity of violated and non-violated jobs are highly skewed, which is visualized in the Figure 2 of Appendix C.

6.3 Examples of Features and SLA Violation Status

For the ten features (X_1 X_{10}), their feature ID, feature name, description, type and corresponding layer are detailed in Table 2. Further, Table 3 gives specific samples of two classes (non-violated and violated) in terms of these feature values.

TABLE 2
Summary of the ten features and their classification by layer

Layer	Feature ID	Feature Name	Description	Type
BDSaaS	X_1	*cpu_requested_per_job*	A floating point number indicating the amount of requested CPU	Float
	X_2	*mem_requested_per_job*	A floating point number indicating the normalized amount of requested memory	Float
	X_3	*num_tasks_per_job*	An integer representing how many tasks a batch job has, with one as the minimum number of tasks	Integer
BDPaaS	X_4	*num_instances_per_job*	A floating point number indicating the number of instances per job, with 1.0 as the minimum number of instances	Float
	X_5	*real_cpu_max_per_job*	A floating point number indicating maximum CPU numbers of actual instance running	Float
	X_6	*real_cpu_avg_per_job*	A floating point number indicating average CPU numbers of actual instance running	Float
	X_7	*real_mem_max_per_job*	A floating point number indicating maximum normalized memory of actual instance running	Float
	X_8	*real_mem_avg_per_job*	A floating point number indicating average normalized memory of actual instance running	Float
CIaaS	X_9	*cpu_capacity_per_job*	A floating point number indicating the capacity of CPU of a machine	Float
	X_{10}	*mem_capacity_per_job*	A floating point number indicating the capacity of normalized memory of a machine	Float

It is observed that the range of values in the above table varies widely such as the value of feature X_2 in the violated example is 0.0055 while the value of feature X_1 in the violated example is 100. This is because, in this dataset, the values for disk and memory utilization have been normalized for confidentiality reasons while the values for requested CPU have not been normalized.

It is worth noting that the absent of normalization will cause objective functions of most machine learning algorithms work improperly. This is because the majority of classifiers in these algorithms use Euclidean distance method to calculate the distance between two points. The value of distance will be dominated by a particular feature that has a broad range of values. Therefore, we normalize all the above features (from X_1 to X_{10}) by applying standardization methods such as Min-Max scaling [16], such that each feature has approximately proportionate contribution to the final Euclidean distance.

7 PREDICTION MODELS

After extracting ten features and then defining SLA and its status [0: "non-violated", 1: "violated"], all these features will be utilized in prediction models to detect SLA violation. For detecting SLA violation in the Alibaba dataset, there are many machine learning prediction models that can be used. However, we particularly selected Logistics Regression (LR) [17], Artificial Neural Network (ANN) [18], Random Forest (RF) [19] and Extreme Gradient Boosting (XGB) [20] for the

TABLE 3
Samples of two classes (violated and non-violated Job)

Feature	Feature Description	Violated Job	Non-Violated Job
X_1	*cpu_requested_per_job*	100	50
X_2	*mem_requested_per_job*	0.0055	0.0094
X_3	*num_tasks_per_job*	7	14
X_4	*num_instances_per_job*	85.57	87.86
X_5	*real_cpu_max_per_job*	0.7819	4.1895
X_6	*real_cpu_avg_per_job*	0.4324	0.2622
X_7	*real_mem_max_per_job*	0.0083	0.0143
X_8	*real_mem_avg_per_job*	0.0057	0.0098
X_9	*cpu_capacity_per_job*	63.5719	63.8789
X_{10}	*mem_capacity_per_job*	0.6853	0.6886

following reasons (Details of these prediction models can be seen in Appendix D).

- LR is one of the elementary and likely most commonly used machine learning algorithms for solving all classification problem. It is easy to implement, fast to train and returns probability scores
- ANN is one of advanced machine learning models leveraging deep learning technology. It works by splitting the problem of classification into a layered network of simpler elements. Hence, it is very meaningful to apply ANN to detect SLA violation for layer-based big data analytics applications
- RF is one of popular ensemble machine learning models. It is an improvement over bagged decision trees and returns the importance of all features, which provides insightful information regarding features contribution to SLA violation detection
- XGB is also one of ensemble machine learning models and returns the importance of all features. Unlike RF, it is based on boosting concept

8 TACKLING UNBALANCED DATA

We will feed the dataset to the aforementioned four machine learning prediction models to detect future SLA violations. Since the two classes [0: ”non-violated”, 1: ”violated”] in the Alibaba dataset are heavily unbalanced, this is known as data skewness which makes the classification task extremely hard. This is because the classifier will always tend to predict the dominant class. However, the true misclassification costs may be much greater when minority class instances are missed. For instance,, incorrectly predicting the actual ”violated” jobs will cause the degradation of reputation and loss of profit for providers.

As a broadly adopted approach in handling unbalanced datasets, resampling efficiently changes the distribution training data for the purpose of biasing the classifier towards the minority class. Basically, resampling can be divided into two groups (i.e., undersampling and oversampling). Undersampling denotes removing samples from the majority class, while oversampling represents adding more examples from the minority class. Different forms of resampling techniques are detailed in Appendix E. The resampling techniques discussed in this paper are by no means an exhaustive list.

9 EVALUATION METRICS

In order to measure the performance of the above prediction models in the dataset, evaluation metrics are required,

TABLE 4
Confusion matrix in binary classification

	Positive (Real)	Negative (Real)
Positive(Prediction)	True Positive (TP)	False Positive (FP)
Negative(Prediction)	False Negative (FN)	True Negative (TN)

which helps us indicate how skilfully a model will perform. Thus, after a prediction model is trained on the training set, it must be validated on an unseen testing dataset. This approach is beneficial in choosing a model that will have reasonably decent performance on unknown dataset. In this paper, five diverse evaluation metrics have been used. They include Accuracy, Precision, Recall, Receiver Operating Characteristic (ROC) area, and F score.

Let us first present the confusion matrix in Table 4 which will help us to define the above metrics. We can see that in a confusion matrix, the correctly classified instances are in the diagonal of the matrix, the True Positive (TP) and True Negative (TN) cases. The misclassified instances are the False Positive (FP) and False Negative (FN) ones. In our problem, FP denotes the quantity of examples that are erroneously classified as ”violated” where the real label are ”non-violated”. Similarly, TP represents the number of examples that are correctly classified as ”violated” where the real label are ”violated”. Regarding SLA violation detection, it is worth noticing that the most important value to increase is the number of TPs cases, which correspond with the correctly detected SLA violations. Metrics involving the TNs are usually not useful because this number is usually much higher than its TP counterpart, as SLA violations are rare events. Therefore, our objective is to find the right balance of FNs and FPs, while maximizing the TP observations. Usually, minimizing the FN instances is prioritized over minimizing the FPs due to the higher value in detecting new SLA violations and higher loss that new SLA violations cause. Based on the above confusion matrix, different metrics can be computed depending on what we are interested in measuring (details of these evaluation metrics are described in Appendix F).

10 EVALUATION AND ANALYSIS OF THE PREDICTION MODELS

In order to train and validate the skill of the four prediction models on unknown data, a 10-fold stratified cross validation has been used on 11897 batch jobs obtained from the Alibaba trace.

The 10-fold stratified cross-validation method validates predication technique by splitting the dataset into 10 equal size subsets, ensuring that each subset is a representation of the entire data. In each fold, the original dataset has been randomly split into two parts, where one part is used for training phrase while the remainder is retained as test set for validation phrase. The ratio of ”non-violated” and ”violated” jobs in the validation part remains equally with the ratio in the original dataset. To avoid leaking the information of validation data to training data, which often results in overfitting, it is critical to perform the cross validation before resampling. Resampling can only be done on the samples which are applied for training the particular type of machine learning predictor.

The value of accuracy, precision, recall, F_2 score and

ROC area for each prediction technique (representing a particular predictor performing either in the original imbalanced dataset or in the resampled dataset using a particular skewness handling technique) are calculated in each fold. Then, their statistics information can be acquired after repeating the experiments ten times. Moreover, we introduced two more classifiers to compare how skilful the four prediction models perform.

Ideal Classifier: represents a classifier that can perfectly predict those actual "non-violated" batch jobs as "non-violated" and those actual "violated" batch jobs as "violated". In this case, all of evaluation metrics (i.e., accuracy, precision, recall, F_2 and ROC area) are 1.0. This is an ideal classifier and could be regarded as the best case.

Baseline Classifier: this classifier applies the simplest rule on our SLA violation detection problem that it simply predicts every sample in the dataset as the majority class (i.e., "non-violated"). It can be derived that the accuracy of Baseline classifier is 0.9056 and the ROC area is 0.5. The value of precision, recall and F_2 score are 0.0. This classifier involves the less effort of prediction and could be regarded as the baseline.

10.1 Results of Cross Validation

For each prediction model, the table of the cross validation result is detailed in Appendix G.

10.1.1 Logistics Regression

It is observed from Table 1 in Appendix G that LR using all of the resampling techniques except one-sided selection (OSS) outperform LR in the original dataset regarding F_2 score and ROC area. However, none of the above techniques achieve an acceptable F_2 score. The top F_2 score achieved by Borderline-1 is only 59.96%. It is also found that Borderline-1 has the highest ROC value (81.18%) outperforming other techniques, Regarding the recall value, NearMiss-1, NearMiss-2 and Borderline-2 rank the top three, 92.07%, 90.02%, and 89.04% respectively.

In summary, LR's performance is under par as regards to predict SLA violation on this Alibaba dataset. It means LR is a simple model without the capability of finding the hidden pattern among the multiple configurations across layers. It gives us an indicator that a more complex model should be explored.

10.1.2 Artificial Neural Network

It can be seen from Table 2 in Appendix G that the prediction outcome is much better than that of logistics regression (LR). The highest recall value is 93.14%, achieved by Borderline-2. Regarding F_2 score, 62% of the above 13 approaches achieve higher than 60%. The top three F_2 score achiever are SMOTE-ENN, Borderline-1 and ROS, 71.7%, 69.66%, and 69.55% respectively. Notably, SMOTE-ENN also ranks the top in the ROC area (87.93%), with an acceptable accuracy value of 85.58%.

It can be concluded that although ANN that we designed and implemented is as simple as ten features in the input layer and three neurons in the hidden layer, it achieves a very decent detection of SLA violation in this dataset. In comparison to LR, ANN is much better at understanding the hidden relationship of those multiple configurations across layers and hence leads to better prediction efficiency. It is worth noting that the performance of ANN in the original dataset is unacceptable (recall and F_2 score are only 51.83% and 54.77% respectively), which shows that ANN suffers from data skewness.

10.1.3 Random Forest

Based on Table 3 in Appendix G, it is seen that RF outperforms LR and ANN. This time, NearMiss-2 wins the highest recall values, which is 95.9%. In comparison with the top recall winner by ANN, RF also improves the recall value by 3% using NearMiss-2. Among the above 13 approaches, 54% of them achieve F_2 score that is above 80%, which is very encouraging. The top F_2 score is 83.57%, achieved by borderline-2. Compared with the top F_2 score winner by ANN, RF significantly improved the F_2 score by 17% through the application of borderline-2. Regarding ROC area, SMOTE-ENN achieved 91.95%, ranking the first. Also, SMOTE-ENN has a very high accuracy rate of 95.24%.

It demonstrates that RF is more effective than LR and ANN in predicting SLA violation in this Alibaba dataset. Even without resampling, it still achieves an admissible performance (accuracy = 97.04%, recall = 74.08% and F_2 = 77.22%). RF has better performance because it is a bagging-based tree classifier, which are less sensitive to class distributions, while LR and ANN are very sensitive with the highly biased class distribution and cannot generate any acceptable results without leveraging resampling techniques.

10.1.4 Extreme Gradient Boosting

It is shown in Table 4 in Appendix G that XGB plus NearMiss2 achieved a stunning recall value of 97.15%. In comparison with the top recall score achiever by random forest (RF) plus NearMiss-2, XGB plus NearMiss-2 improved the recall score by 1.3%. Regarding F_2 score, the performance of XGB plus random oversampling (ROC) (F_2 score = 82.99%) is marginally lower than that of RF plus borderline-2 (F_2 score = 83.57%).

It is also found that without using resampling technique, the performance of XGB (accuracy = 96.91%, recall = 74.17%, F_2 = 77.07% and ROC area = 96.91%) is very similar with the performance of RF (accuracy = 97.04%, recall = 74.08%, F_2 = 77.22% and ROC area = 86.76%)). Like RF, XGB has better performance because it is also a decision tree-based classifier, which is more tolerable to class distributions. Thus, it achieves decent performance even without resampling.

10.2 Analysis of Prediction Outcome

According to the above extensive experiments results, for each evaluation metrics, we can study how skilful these prediction techniques are when compared with the "Ideal" and "Baseline" classifiers.

According to Figure 3, it is observed that among the top 10 accuracy value, only one undersampling technique (one-sided selection) performs well, the remaining are all oversampling techniques. Moreover, two predictors (random forest and extreme gradient boosting) occupy the top ten accuracy list. Specifically, RF plus one-side selection (OSS), and XGB plus OSS achieve very high accuracy, 97.02%, and 96.92% respectively. However, their accuracy value are gently lower than the accuracy achieved by RF in the original dataset. It is also noted that RF or XGB plus some resampling techniques (one-sided selection, random oversampling, SMOTE, Borderline family, ADASYN and SMOTE - Tomek) significantly improve the accuracy com-

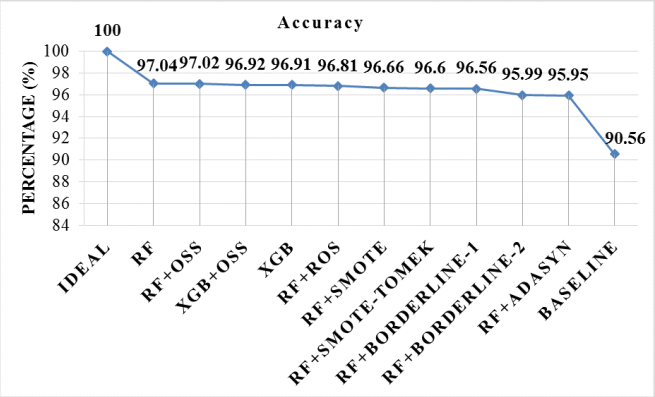


Fig. 3. Top 10 techniques measured by accuracy

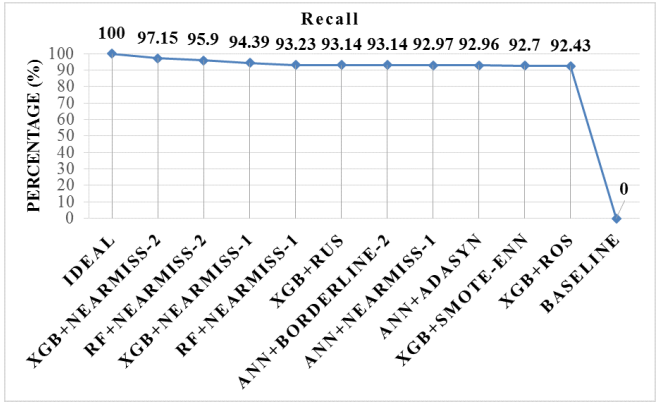


Fig. 5. Top 10 techniques measured by recall

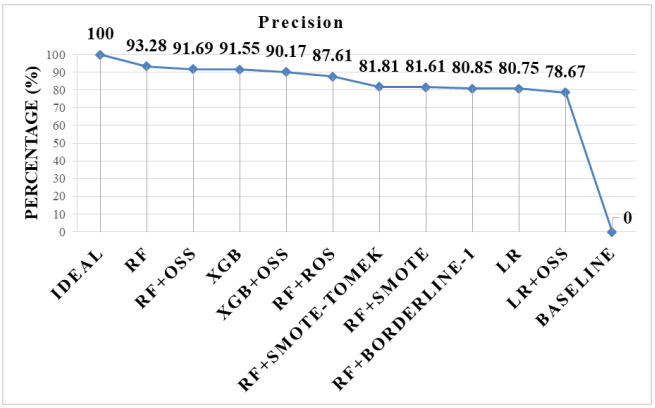


Fig. 4. Top 10 techniques measured by precision

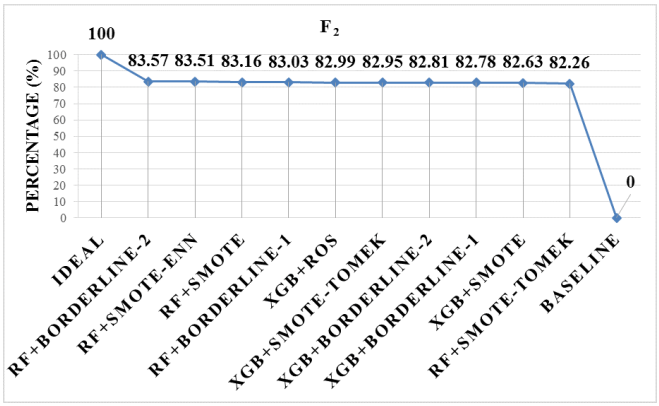


Fig. 6. Top 10 techniques measured by F_2

pared with the Baseline classifier (accuracy = 90.56%). It is concluded that random forest is the technique winning the top accuracy value.

From Figure 4, it is found that among the top 10 prediction techniques, only one uses the undersampling technique (one-sided selection), the remaining are all oversampling techniques. One-sided selection (OSS) performs better in achieving high precision compared with other resampling techniques. Take random forest (RF) as an example, the precision value of RF plus OSS is higher than RF plus other resampling techniques. However, the precision of OSS plus OSS (91.69%) is slightly lower than the precision of RF without using resampling techniques (93.28%). This is the same case for RF plus OSS, and extreme gradient boosting (XGB) plus OSS. Notably, these top ten prediction techniques significantly outperform the baseline classifier with the precision value only 0. It is concluded that random forest is the technique winning the top precision value.

According to Figure 5, it is clearly seen that the highest recall value (97.15%) is achieved by XGB plus NearMiss-2. For the predictors of XGB and RF, it is also observed that NearMiss-2 outperforms its counterpart NearMiss-1 regarding recall value. For instance, the recall value of NearMiss-2 plus XGB is 2.92% higher than that of NearMiss-1 plus XGB. Similarly, the recall value of NearMiss-2 plus RF is 2.86% higher than that of NearMiss-1 plus RF. Moreover, it is interesting to find that ANN occupies three positions (i.e., ANN + Boderline-2, ANN +NearMiss-1, and ANN + ADASYN) in the top 10 prediction techniques regarding recall value. Specifically, ANN plus Borderline-2 surpasses ANN plus NearMiss-1, and ANN plus NearMiss-1 performs slightly better than ANN plus ADASYN. Since the recall value of Baseline classifier is only 0, these top 10 prediction techniques considerably improve their recall value. It is concluded that XGB plus NearMiss-2 is the technique winning the top recall value.

Figure 6 shows that RF and XGB play a dominant role in those techniques achieving top ten F_2 value. From the perspective of resampling techniques, the top ten F_2 value is only achieved by oversampling techniques. Especially, SMOTE, its variants the family of Borderline, two combined resampling method (SMOTE-Tomek and SMOTE-ENN), and random oversampling (ROS) are the most outstanding methods to get high F_2 value. RF beats XGB regarding F_2 score because the top 4 F_2 value are all achieved by RF. Specifically, RF plus Borderline-2 won the top F_2 value, followed by RF plus SMOTE-ENN, RF plus SMOTE, and RF plus Borderline-1. It is concluded that random forest (RF) plus Borderline-2 is the technique winning the top F_2 value.

Figure 7 presents that still RF and XGB stand out in the top ten ROC. Regarding resampling techniques, the top ten ROC value are dominantly achieved by oversampling techniques. Only one undersampling technique (random undersampling) achieved a decent ROC value. Especially, random oversampling (ROS), SMOTE, its variants the family of Borderline, ADASYN, two combined resampling method (SMOTE-Tomek and SMOTE-ENN), and random undersampling are the most superior methods to get high

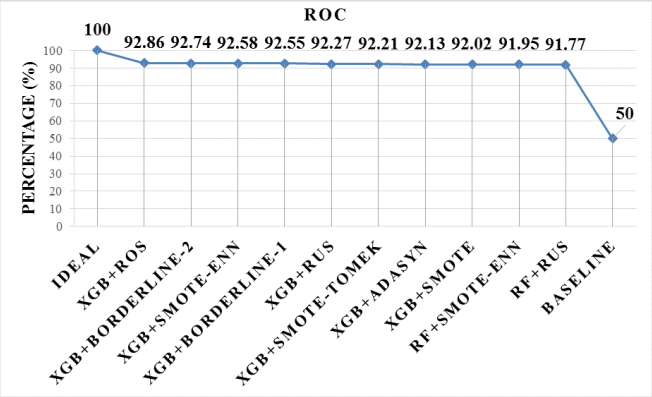


Fig. 7. Top 10 techniques measured by ROC

ROC value. This time XGB excels RF because the top 8 ROC value are all achieved by XGB. Specifically, XGB plus ROS ranks the top, followed by XGB plus Borderline-2, and XGB by SMOTE-ENN. It is concluded that extreme gradient boosting (XGB) + random oversampling (ROS) is the technique winning the top ROC value.

We summarise our finding as follows:

- The technique of winning the top accuracy and precision value: RF.
- The technique of winning the top recall value: XGB + NearMiss-2.
- The technique of winning the top F_2 value: RF + Borderline-2.
- The technique of winning the top ROC value: XGB + ROS.

11 MATHEMATICAL MODEL AND ASSUMPTIONS ON PROVIDERS' PROFIT

So far, we identified the four prediction techniques in achieving the highest value of accuracy, precision, recall, F_2 and ROC respectively. However, it is not clear for providers how each prediction technique impacts the profit they can earn. Hence, providers would not be able to choose which prediction technique is best given their workload characteristics. In order to answer this question, we further investigate the capability of these 4 prediction techniques in achieving the profit and compare them with the Ideal and Baseline classifiers.

For clarity and quick reference, we develop a set of mathematical symbols in Table 5 to characterize the elements in our mathematical model frequently used hereafter.

For simplicity, we assume that each job J_i in J shares the equal expense E and revenue R, hence, their profit margin is also equal. Therefore, we have the following formulas:

$$Margin = \frac{R-E}{R}$$

$$Profit_{max} = Margin \quad R \quad (N_{TN} + N_{FP})$$

Profit of the provider will be dependent on how many jobs that are not violated. Therefore, number of false positive and true negatives are considered to calculate the profit.

11.1 Admission Control Policy and Profit Loss Matrix

Given the outcome of each prediction technique, providers consider the following admission control policies shown in Table 6 in order to optimize their profit:

TABLE 5
Notation used in the mathematical model of providers' profit

Symbols	Description
J	A set of batch jobs or workloads
J_i	a batch job or workload instance in J
N_{TN}	The quantity of True Negative (TN) jobs in J
N_{FP}	The quantity of False Positive (FP) jobs in J
N_{FN}	The quantity of False Negative (FN) jobs in J
N_{TP}	The quantity of True Positive (TP) jobs in J
E_i	The expense of processing a job J_i
R_i	The revenue that a provider receives if he processes the job J_i successfully
E_i	the expense of processing a job J_i
$Margin_i$	The profit margin that a provider makes if successfully processing a job J_i
PenaltyRate	The penalty ratio when a provider violated SLA
$Profit_J$	The actual profit that a provider makes after processing J
$Profit_{max}$	The maximum amount of profit the provider can make
ProfitRatio	The percentage of the actual profit after processing J over the maximized profit that a provider makes. It is defined by $\frac{Profit_J}{Profit_{max}}$

TABLE 6
Admission control policy matrix by providers

	Violated (Real)	Non-Violated (Real)
Violated (Prediction)	Reject	Reject
Non-Violated (Prediction)	Accept	Accept

- Accept: provider will accept a batch job J_i if it is predicted as "non-violated". In this case, N_{TN} plus N_{FP} jobs will be accepted by providers.
- Reject: providers will reject a batch job J_i if it is predicted as "violated". In this case, N_{FN} plus N_{TP} batch jobs will be rejected by providers.

According to Table 6, there are four situations considered:

- TN (True Negative): means a job J_i is actually "non-violated" and the prediction technique successfully predicts it as "non-violated". This is a good situation, and hence no profit loss will be produced. Clearly, providers will accept this job J_i
- FN (False Negative): means a job J_i is actually "violated" and the prediction technique misclassifies it as "non-violated". In this case, providers will accept this job J_i. However, processing J_i in the production environment will cause loss not only the cost E, but also the penalty (equals to $PenaltyRate \quad E$) generated due to SLA violation contract. The total profit loss can be calculated as $(1 + PenaltyRate) \quad E$
- TP (True Positive): means the job J_i is actually "violated" and the prediction technique successfully predicts it as "violated". This is a good situation and providers would reject J_i. In this case, no profit loss will be produced
- FP (False Positive): means the job J_i is actually "non-violated" and the classifier mistakenly predicts it as "violated". In this case, providers will reject to process this job J_i. However, giving up processing J_i in the production environment will cause the profit loss with the value of $(R \quad E)$.

As profit loss is generated either by accepting or rejecting admission control policy, matrix based on the profit loss of every decision can be derived from the above admission

TABLE 7
Profit loss matrix for SLA violation detection problem (Unit: $/job)

	Violated (Real)	Non-Violated (Real)
Violated (Prediction)	0	*-(R-E)*
Non-Violated (Prediction)	*(1+PenaltyRatio) * E*	0

TABLE 8
Different combination of PenaltyRate and ProfitMargin

PenaltyRate	PenaltyMargin	Comment
0.1	0.1	Low Penalty Rate, Low Profit Margin
0.25	0.1	Medium Penalty Rate, Low Profit Margin
1	0.1	High Penalty Rate, Low Profit Margin
0.1	0.2	Low Penalty Rate, Medium Profit Margin
0.25	0.2	Medium Penalty Rate, Medium Profit Margin
1	0.2	High Penalty Rate, Medium Profit Margin
0.1	0.3	Low Penalty Rate, High Profit Margin
0.25	0.3	Medium Penalty Rate, High Profit Margin
1	0.3	High Penalty Rate, High Profit Margin

control matrix and can be useful to evaluate the business value (profit-oriented) of a prediction technique. The details of profit loss information are shown in Table 7.

11.2 The Formulation of a Provider's Profit

Now, we can derive the formula of a provider's profit $Profit_J$ after processing J based on the SLA violation detection outcome by a particular prediction technique.

$$Profit_J = (R-E)*(N_{TN}+N_{FP}) - (1+PenaltyRate)*E*N_{FN} - (R-E)*N_{FP}$$
$$= Margin*R*N_{TN} - (1+PenaltyRate)*R*(1-Margin)*N_{FN}$$

The final profit will depend on the number of non-violations predicated corrected and the number of violations that are not predicated correctly.

Then, the ProfitRatio can be calculated according to the following formula:

$$ProfitRatio = \frac{Profit_J}{Profit_{max}} = \frac{Margin*R*N_{TN} - (1+PenaltyRate)*R*(1-Margin)*N_{FN}}{Margin*R*(N_{TN}+N_{FP})} = \frac{Margin*N_{TN} - (1+PenaltyRate)*(1-Margin)*N_{FN}}{Margin*(N_{TN}+N_{FP})}$$

It is seen that ProfitRatio is irrelevant with the revenue R, but dependents on the PenaltyRate and ProfitMargin. To explore how a provider's profit varies as the PenaltyRate and ProfitMargin changes given by the outcome of a prediction technique, we consider three types of PenaltyRate = [Low, Medium, High], which matches Alibaba's SLA definition [21], and three different types of ProfitMargin = [Low, Medium, High]. Their possible combinations are detailed in Table 8.

12 EXPERIMENTS AND DISCUSSION

To measure how much profit a prediction technique can generate based on its prediction outcome, we calculate the value of ProfitRatio using the above formula for each of the four prediction techniques over the nine different combinations of PenaltyRate and ProfitMargin described in Table 8. The result in comparison with the ProfitRatio value of the Ideal and Baseline classifier is shown in Figure 8.

It is found that for each prediction technique using the same PenaltyRate, the value of ProfitRatio increases as the profit margin increases. This is because providers can get more profit from a prediction technique that correctly predicts more true negative jobs. For example, XGB plus NearMiss-2 (the top recall winner) increases its ProfitRatio 4.98% from low profit margin to medium profit margin, and 1.59% from medium profit margin to high profit margin given a low PenaltyRate.

Overall, XGB plus ROS and RF plus Borderline-2 are the top two prediction techniques to acquire profit. Next is RF (the top accuracy and precision value winner), while XGB+NearMiss-2 (the top recall value winner) ranks the last. Specifically, when a low profit margin is applied, it is seen that XGB plus ROS (the top ROC winner) clearly outperforms RF plus borderline-2 (the top F_2 value). The most improvement ratio by XGB+ROS over RF+borderline-2 is 11.7% when a high PenaltyRate is set. On the contrary, if a high profit margin is set, it is seen that RF plus borderline-2 (the top F_2 value) is slightly stronger than XGB plus ROS (the top ROC winner) regarding the capability of making a profit. Similarly, if a medium profit margin is applied, RF plus borderline-2 (the top F_2 value) is gently better than XGB plus ROS (the top ROC winner) except a high PenaltyRate is set.

Even though XGB plus NearMiss-2 achieves the top recall value, it performs worse in maximizing the profit. The reason is that XGB+NearMiss-2 featured by a very high recall value (97.15%) and an extremely low precision value (only 13.64%), suffers lots of false positive cases. According to the admission control policies, lots of false positive misclassification makes providers reject these batch jobs, and hence causes a heavy loss of profit. Comparatively, RF plus borderline-2 (precision = 75.36%, recall = 86.02%), XGB plus ROS (precision = 59.12%, recall = 92.43%), and RF (precision = 93.28%, recall = 74.08%) reasonably balanced the precision and recall, and hence they achieve a more decent profit.

Moreover, it is observed that the Baseline classifier fluctuates significantly over the nine combinations of PenaltyRate and ProfitMargin. Specifically, when a low profit margin is applied, the profit it gets is minus. This is because the precision and recall value of Baseline classifier is zero, which causes a very high penalty, and the benefit from correctly predicting true negative jobs is very limited when a low profit margin is set. This situation has been changed since higher profit margins (e.g., medium ProfitMargin and high ProfitMargin) are applied. It can be concluded that RF plus borderline-2 and XGB plus ROS are the two best prediction techniques that provider can make optimal profit based on their prediction outcome of SLA violation.

It is worth noting that both RF and XGB provide human interpretable results. According to the aforementioned working mechanism of RF and XGB, multiple single decision trees are generated and work together towards the final prediction outcome. Each single decision tree provides a very intuitive way to understand how it works on the prediction because it follows a method of decision-making that is very similar to how humans make decisions with a chain of simple rules. Both RF and XGB can visualize any single decision trees inside them. An example of the selected single decision trees (the depth of the tree is set to 3 for better visualization) for RF and XGB respectively is shown in Figure 7 in Appendix H. It is observed from Figure 7(a) that among totally five appeared features, there are three features (i.e.,X6,X7, and X8) belong to BDPaaS layer. Similarly, according to Figure 7(b), among totally five appeared features, there are four features (i.e.,X4,X5,X7, and

X8) belong to BDPaaS layer. This demonstrates that the features at BDPaaS layer (i.e.,X4X8) play a dominant part in determining the final violation status compared to other features from BDSaaSand CIaaS layer.

In addition, both RF and XGB allow us to disclose each feature's importance playing in the classification of SLA violation. Based on our well-trained model, the average importance of these ten features based on 10-fold stratified cross validation is shown in Figure 9(a) for RF plus Borderline-2 and in Figure 9(b) for XGB plus ROS respectively. It is observed that the *real_cpu_max* has the highest contribution, followed by *memory_requested* and the number of instances for XGB plus ROS. Also, it is seen that the *real_cpu_max* has the top contribution, next is *real_cpu_avg*, followed by *real_mem_avg* for RF + Borderline-2. Moreover, *memory_capacity* and *cpu_requested* have much lower contributions than other features in both RF and XGB predictors.

Further, we aggregate the feature importance by layers and present the graphical representation in Figure 10. It is found that for XGB plus ROS, the aggregated features importance at BDPaaS layer is dominant, occupying 63%, roughly two times than that at BDSaaS layer (29%), and eight times than that at CIaaS layer (8%). Similarly, for RF plus Borderline-2, the aggregated features importance at BDPaaS layer is also dominant, occupying 72%, roughly three times than that at BDSaaS layer (20%), and nine times than that at CIaaS layer (8%).

It can be concluded that both XGB and RF attach more importance to the features from BDPaaS layer (i.e., *real_cpu_max*, *real_cpu_avg*, *number_of_instances*, and *real_mem_avg*) compared to the feature (*cpu_requested*) at BDSaaS layer and feature (*memory_capcacity*) at CIaaS layer. This is a further evidence that the features at BDPaaS layer (i.e., X_4 X_8) are determinant factors to detect final violation status.

Such findings uncover the hidden patterns of the multiple configurations across layers and provide insightful information to providers for decision making. Concretely, BDPaaS layer needs more attention that can better serve the batch job workloads. For example, at BDPaaS layer, scheduling a batch job that better allocates the CPU or memory or reasonably split the job into the number of instances can improve the efficiency in reducing SLA violations. On the contrary, paying attention to the capacity of memory or CPU at CIaaS layer or the requested CPU at BDSaaS layer, will generate insignificant efficiency in reducing the SLA violations.

13 CONCLUSIONS AND FUTURE WORKS

In this paper, we addressed the problem of detecting SLA violations for a real cloud-hosted BDAA. We used the dataset that is newly released by Alibaba and contains detailed trace information regarding batch workloads among 1300 machines in 12 hours. We explored four diverse machine learning-based predictors (i.e., logistics regression, artificial neural network, random forest, and extreme gradient boosting) to detect the SLA violations. Since the dataset is heavily skewed, we also examined 12 different resampling techniques to handle the challenge of data skewness in

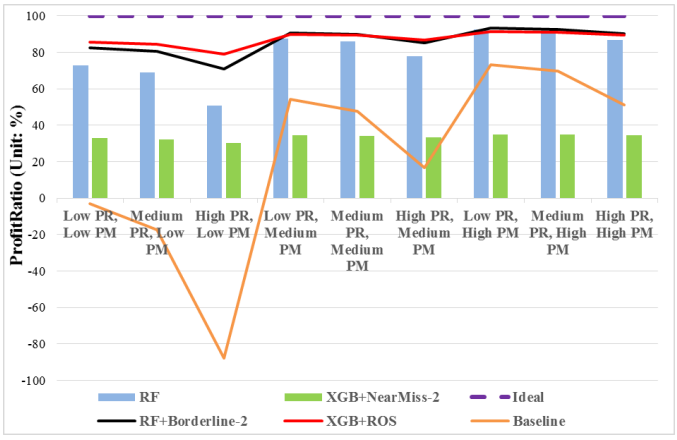
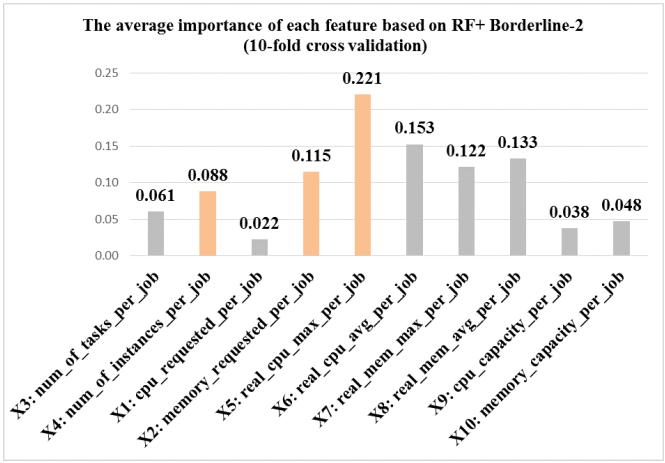
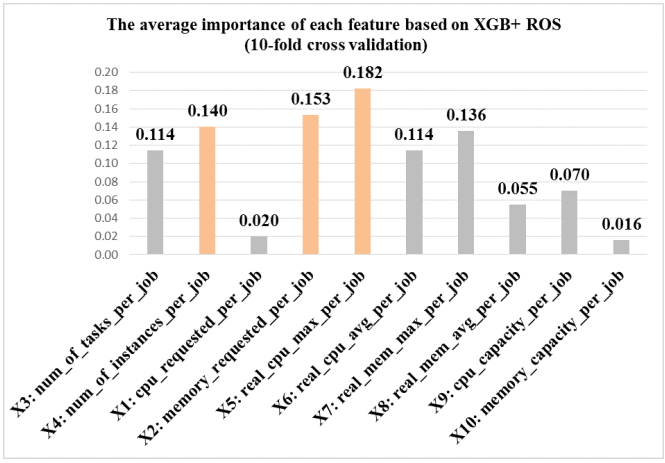


Fig. 8. The capability of a prediction technique in achieving profit on different PenaltyRatio and ProfitMargin combination

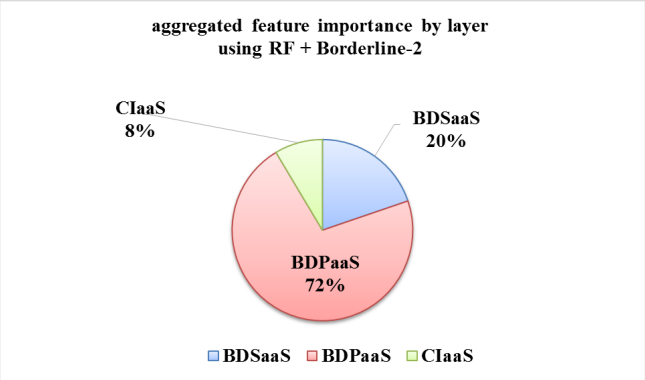


(a) Random forest plus Borderline-2

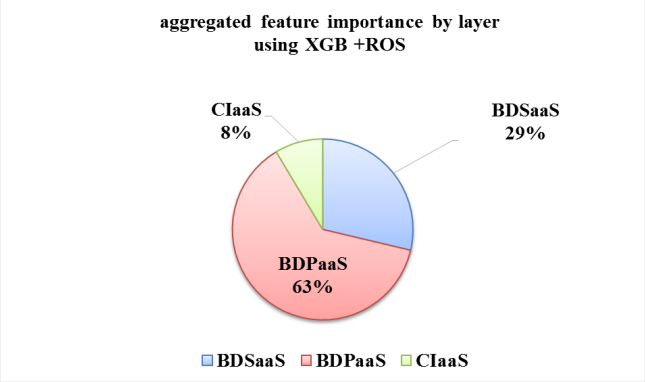


(b) Extreme gradient boosting plus random oversampling

Fig. 9. The average importance of each feature in RF and XGB



(a) Random forest plus Borderline-2



(b) Extreme gradient boosting plus random oversampling

Fig. 10. The aggregated feature importance by layer in RF and XGB

order to acquire better performance. Standard statistical significance metrics, such as accuracy, precision, recall, F_2, and ROC has been applied to test the practicality and efficiency of the predictors. Most importantly, we designed a novel mathematical model regarding provider's profit and evaluated the capability of these prediction techniques in helping providers' optimizing their profits.

Two future works might arise from our study on the detection of SLA violation for BDAAs in cloud. The first research direction is to formulate our problem as an anomaly detection problem, such that diverse techniques could be investigated to identify unusual patterns (i.e., the batch job is "violated"). Our work is very constructive for future researchers in the application of the anomaly detection technique to detect SLA violations for BDAAs in cloud because the extracted features and proposed mathematical profit model can be utilized by future researchers. The second research direction is to extend our prediction techniques and data skewness handling techniques to stream-based BDAA workload. Still, our work is very helpful to conduct further investigation.

REFERENCES

[1] D. Lehmann, D. Fekete, and G. Vossen, "Technology selection for big data and analytical applications," Working Papers, ERCIS-European Research Center for Information Systems, Tech. Rep., 2016.

[2] I. A. T. Hashem, I. Yaqoob, N. B. Anuar, S. Mokhtar, A. Gani, and S. U. Khan, "The rise of big data on cloud computing: Review and open research issues," *Information systems*, vol. 47, pp. 98–115, 2015.

[3] H. Hu, Y. Wen, T.-S. Chua, and X. Li, "Toward scalable systems for big data analytics: A technology tutorial," *IEEE access*, vol. 2, pp. 652–687, 2014.

[4] L. M. Pham, "A big data analytics framework for iot applications in the cloud," *VNU Journal of Science: Computer Science and Communication Engineering*, vol. 31, no. 2, 2015.

[5] L. Wang, Y. Ma, J. Yan, V. Chang, and A. Y. Zomaya, "pipscloud: High performance cloud computing for remote sensing big data management and processing," vol. 78. Elsevier, 2018, pp. 353–368.

[6] P. Leitner, B. Wetzstein, F. Rosenberg, A. Michlmayr, S. Dustdar, and F. Leymann, "Runtime prediction of service level agreement violations for composite services," in *Service-oriented computing. ICSOC/ServiceWave 2009 workshops*. Springer, 2010, pp. 176–186.

[7] O. Jules, A. Hafid, and M. A. Serhani, "Bayesian network, and probabilistic ontology driven trust model for sla management of cloud services," in *Cloud Networking (CloudNet), 2014 IEEE 3rd International Conference on*. IEEE, 2014, pp. 77–83.

[8] B. Tang and M. Tang, "Bayesian model-based prediction of service level agreement violations for cloud services," in *Theoretical Aspects of Software Engineering Conference (TASE), 2014*. IEEE, 2014, pp. 170–176.

[9] R. A. Hemmat and A. Hafid, "Sla violation prediction in cloud computing: A machine learning perspective," *arXiv preprint arXiv:1611.10338*, 2016.

[10] (2013) Google cluster workload traces. [Online]. Available: https://github.com/google/cluster-data

[11] R. B. Uriarte, S. Tsaftaris, and F. Tiezzi, "Service clustering for autonomic clouds using random forest," in *Cluster, Cloud and Grid Computing (CCGrid), 2015 15th IEEE/ACM International Symposium on*. IEEE, 2015, pp. 515–524.

[12] (September 2017) Alibaba cluster trace v2017. [Online]. Available: https://github.com/alibaba/clusterdata/tree/master/cluster-trace-v2017

[13] C. Lu, K. Ye, G. Xu, C.-Z. Xu, and T. Bai, "Imbalance in the cloud: an analysis on alibaba cluster trace," in *Big Data (Big Data), 2017 IEEE International Conference on*. IEEE, 2017, pp. 2884–2892.

[14] Q. Liu and Z. Yu, "The elasticity and plasticity in semi-containerized co-locating cloud workload: a view from alibaba trace," in *Proceedings of ACM Symposium on Cloud Computing (SOCC)*, 2018.

[15] Y. Cheng, Z. Chai, and A. Anwar, "Characterizing co-located datacenter workloads: An alibaba case study," *arXiv preprint arXiv:1808.02919*, 2018.

[16] O. Kramer, "Scikit-learn," in *Machine learning for evolution strategies*. Springer, 2016, pp. 45–53.

[17] S. Sperandei, "Understanding logistic regression analysis," *Biochemia medica: Biochemia medica*, vol. 24, no. 1, pp. 12–18, 2014.

[18] A. K. Jain, J. Mao, and K. Mohiuddin, "Artificial neural networks: A tutorial," *Computer*, no. 3, pp. 31–44, 1996.

[19] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[20] L. Torlay, M. Perrone-Bertolotti, E. Thomas, and M. Baciu, "Machine learning–xgboost analysis of language networks to classify patients with epilepsy," *Brain informatics*, vol. 4, no. 3, p. 159, 2017.

[21] (September 2018) Alibaba elastic compute service service level agreement. [Online]. Available: https://www.alibabacloud.com/help/doc-detail/42436.htm/

Xuezhi Zeng has a PhD from the Australian National University. He has research and development interests in Machine Learning, Deep Learning, Nature Language Processing, Knowledge Graph and Big Data.