SLA Management For Virtual Machine Live Migration Using Machine Learning with Modified Kernel and Statistical Approach

M. K. Hassan
Future University
Khartoum, Sudan
memo1023@gmail.com

A. Babiker
Neelain University
Khartoum, Sudan
amin.31766@hotmail.com

M. B. M. Amien
University of Gezira
Wad Madani, Sudan
magdy_baker@yahoo.co.uk

M. Hamad
Universiti Teknologi Malaysia
Johor Bahru, Malaysia
Taza1040@gmail.com

Abstract—**Application of cloud computing is rising substantially due to its capability to deliver scalable computational power. System attempts to allocate a maximum number of resources in a manner that ensures that all the service level agreements (SLAs) are maintained. Virtualization is considered as a core technology of cloud computing. Virtual machine (VM) instances allow cloud providers to utilize datacenter resources more efficiently. Moreover, by using dynamic VM consolidation using live migration, VMs can be placed according to their current resource requirements on the minimal number of physical nodes and consequently maintaining SLAs. Accordingly, non optimized and inefficient VMs consolidation may lead to performance degradation. Therefore, to ensure acceptable quality of service (QoS) and SLA, a machine learning technique with modified kernel for VMs live migrations based on adaptive prediction of utilization thresholds is presented. The efficiency of the proposed technique is validated with different workload patterns from Planet Lab servers.**

Keywords-virtual machine; migration; machine learning; SLA

I. INTRODUCTION

Resource optimization has been improved significantly by virtualization. It introduced isolation between application and the physical resource [1], it allows live virtual machines (VMs) to seamlessly move between physical hosts. This allows service providers to host high availability applications and to better commit to their level of service governed by a service level agreement (SLA). Most applications in telecommunication industry permit small fraction of downtime or no downtime at all [2]. SLA violation may occur due to server's resources being over utilized. Therefore, high availability and fault-tolerant systems are crucial in order to maintain such a demanding policy which is costly in terms of capital and operational expenses. Tightly controlled live migration can provide solution to this problem by moving VMs with little or no interruption but that is often against the agreed SLA. However, these interruptions can cause performance degradation which varies between applications [3-4]. Thus, predicting live migration at the earliest time possible will significantly contribute to reducing any performance degradation due to SLA violation or from the duration of any interruption. Our objective is to provide a machine learning and statistical based predictive model to predict VM migration and consequently maintaining the SLA. It is a heuristic based predictive model where future SLA violation is to be predicted, then migration decision will be made by a machine learning algorithm classifier. CPU utilization, inter VM bandwidth utilization and memory utilization will be used as potential classifiers.

II. RELATED WORK

Resource optimization in cloud based data center has been extensively investigated in recent years. Authors in [5] suggested that live migration is to be handled by global policies applied to redistribute the VMs, suggestion based on resources classification into local and global policies. Authors in [6] have adopted priority based approach to allocate the resources in the virtualized clusters. In [9] dynamic consolidation problem was addressed by using a heuristic based approach for the bin packing problem. In [8] a threshold-based reactive approach to dynamic workload consolidation has been proposed. However it was applicable for certain types of applications. Popular approaches such as VMware distributed power management [7] have the drawback that they operate on fixed threshold values which is not suitable for dynamic and unpredictable workloads [9]. In our proposed model, we introduce an approach to set the threshold values dynamically, depending on VMs historical predicted data of the resource usage by each VM and machine learning as a decision making approach. Static and dynamic resource assignment policies in virtualized data centers is discussed in [10, 11]. Authors in [12-15] classified VM consolidation as centralized and decentralized. It was suggested VM migration trigger point to be based on predefined threshold, where on other hand other approaches [13, 15] trigger migration after workload analyzed based on learned-intelligent QoS-based threshold and predictive heuristic methods [16]. Nevertheless, a few set of approaches [14, 15] studied workload-independent QoS-based threshold approaches for the purpose of SLA violation avoidance and efficient migration management. In [16] VM placement problem with traffic-aware balancing (VMPPTB) has been discussed and a longest processing time based placement algorithm (LPTBP algorithm) is designed to solve it. In addition to that, locality-aware VM placement problem with traffic-aware balancing

(LVMPPTB) is proposed. Authors in [17] proposed a VM placement algorithm named ATEA (adaptive three-threshold energy-aware algorithm) to reduce the energy consumption and SLA violation. It is based on historical data collected from resource usage by VMs to migrate VMs on heavily loaded to lightly loaded hosts. All previous works did not consider inter VM bandwidth and memory utilization effects on VM consolidation problem and on SLA definition especially in applications that do not tolerate any downtime or performance degradation like the telecommunication applications [18].

III. SLA VIOLATIONS DETECTION

The VMs experience dynamic variable workloads, in a way that hardware resources consumed by a VM arbitrarily change over time. During this variation SLA can be breached or the host can be over utilized, i.e. if all the VMs request their maximum allowed physical resources. In such case, the algorithm must have perfect knowledge of the time when the SLA violation will occur before it actually occurs. Live migrations can have negative impact on the performance of applications in a VM during a migration. The length of a live migration depends on the total amount of memory used by the VM and the available network bandwidth. The migration time and performance degradation experienced by a VM_j is expressed in (1) [15].

$$Tm_j = M_j / B_j \quad (1)$$

$$U_{dj} = 0.1 \int_{t_o}^{t_0+T_m} U_j(t)d(t) \quad (2)$$

where U_{dj} is the performance degradation by VM_j during migration, $t_0$is the time when the migration starts, Tm_j is the time taken to complete the migration,U_j is the CPU utilization by VM_j , M_j is the amount of memory used by VM_j and B_j is the available network bandwidth.[15]

$$S_j = x_{dj} + U_{dj} \quad (3)$$

In (3) x_{dj} is the performance degradation when the allocated resource utilization for VM_j is not aligned with the agreed SLA and S_j is the total performance degradation by VM_j. Thus, from (3) in order to minimize the total performance degradation, either U_{dj} or x_{dj} should be minimized. In this work we concentrate on minimizing x_{dj} as $U_{dj}$is not only influenced by the CPU utilization U_j, but is also depended on the amount of memory used and the available network bandwidth as well. Moreover, it is more likely that VM_j will experience performance degradation while the host resources utilization is above the agreed SLA more than during the actual live migration. In order to avoid SLA violation and performance degradation, host should perform regular check on the system utilization where an SLA violation detection algorithm should be executed. One of the earliest methods relied on setting static CPU utilization threshold to differentiate between overload and non-overload states of the host. It is simple but inefficient for dynamic workloads, particularly when different types of applications share a physical node. In such case the system should be able to automatically adjust its behavior based on the workload patterns adopted by the applications [15].

A. Local Regression

Our approach, as depicted in Figure 1, relies firstly on work load prediction based on statistical analysis of historical data collected during the VMs' lifetime. Local regression (LR) has proved its efficiency as predictor method [17]. It is a model used to build up a curve from localized subsets of data that approximate the original input with the original data. LR algorithm, derived from local regression algorithm. For each new observation a new trend line is found [15]

$$\hat{g}(x) = \hat{a} + \hat{b}x \quad (4)$$

This trend line is used to predict the next observation $\hat{g}(x_{k+1})$. The new observation can be a host resource utilization such as CPU and memory

$$\hat{g}(x_{k+1}) \geq 1, \qquad x_{k+1} - x_k \leq t_m \quad (5)$$

where t_m is the maximum time required for a VM migration

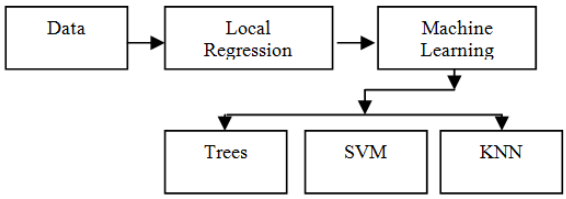

Fig. 1. Proposed Method

B. Classification Trees

When a class is already known on prior in the training samples, classification trees are effective. Let t_p be a parent node and t_l, t_r left and right child nodes of the parent node respectively. Assume the learning sample with variable matrix X with M number of variables x_j and N observations. Let class vector Y consist of N observations with total amount of K classes. Classification tree is based on splitting rule that performs the splitting of learning sample into smaller parts. We already know that each time data have to be divided into two parts with maximum homogeneity of left and right child nodes will be equivalent to the maximization of change of impurity function Δi(t): [19]

$$\Delta i(t) = i(t_p) - E[i(t_c)] \quad (6)$$

where t_c represents the left and right child nodes of the parent node. Assuming that the P_l, P_r probabilities of right and left nodes, we get:[19]

$$\Delta i(t) = i(t_p) - P_l i(t_l) - P_r i(t_r) \quad (7)$$

Therefore, at each node classification trees solves the following maximization problem:[20]

$$\arg\max x_j \leq x_j^R \text{ , } \quad j=1,..., M\ [i(t_p - P_l i(t_l) - P_r i(t_r)] \quad (8)$$

From (8) all possible values of all variables in matrix X for the best split question will be searched through $x_j<x_k^K$ which will maximize the change of impurity measure Δi(t) [19].

C. Support Vector Machine

Support vector machines (SVMs) support nonlinear classification and can find the hyper plane of maximal margin. Given a training set of N data points $\{y_k x_k\}_{k=1}^N$ where $x_k \in \mathbb{R}^n$ is the kth output pattern, the support vector method approach aims at constructing a classifier of the form:

$$y(x) = sign[\sum_{k=1}^{N} a_k y_k \Psi(x, x_k) + b] \quad (9)$$

Input (10) shows an example of hard-margin SVM with noise free training data to be correctly classified by a linear function. Data points D (or training set) are represented mathematically [20, 21]

$$D = \{(x_1, y_1), (x_2, y_2), ..., (x_m, y_m)\} \quad (10)$$

where x_i is a n-dimensional real vector, y_i is either 1 or -1 indicating the class to which the point x_i belongs to. The SVM classification function F(x) takes the form [20]

$$F(x) = w \cdot x - b \quad (11)$$

where b is the bias and w is the weight vector , which will be calculated during the training process. First, to correctly classify the training set F(x) (or w and b) we must return negative numbers for negative data points and positive numbers otherwise, for every point x_i in D in (12) [20]

$$wx-b>0 \text{ if } y_i = 1 \text{ , } wx-b<0 \text{ if } y_i = -1 \quad (12)$$

D. K-Nearest Neighbors

The k-nearest neighbors (KNN) is one of the simplest methods for pattern classification. When combined with prior knowledge it can be used to produce significant results [22]. In the KNN each unlabeled example is classified by the majority label among its k - nearest neighbors in the training set. Therefore its classification performance depends on the distance metric used to identify nearest neighbors. In the case of missing prior knowledge, Euclidean distances between examples represented as vector inputs are used to measure the similarities in most KNN classifiers. Let $\{\vec{x_i}, y_i\}_{i=1}^n$ denote a training set of n labeled examples with inputs $\vec{x_i} \in \mathbb{R}$ and discrete (in our case binary) class labels y_i. We use the binary matrix $y_{ij} \in \{0,1\}$ to indicate whether or not the labels match. Our goal is to learn a linear transformation which we will use to compute squared distances as [21]:

$$D(\vec{x_i}, \vec{x_j}) - || L(\vec{x_i} - \vec{x_j}) ||^2 \quad (13)$$

IV. Results and Analysis

In this work, workload was collected from CoMon project [16]. Extracted data was part of more than a thousand VMs's resource uilization distributed across the world. Samples were selected from 6 servers for a period of one week with 5 minutes measurement interval. CPUs and other resources were adjusted manually with different resource utilizations for the purpose of this experiment. The collected data did not contain memory and inter VM bandwidth utilization. In this work average TPR, Friedman rank summations and average ranking were used as performance metrics. TPR is defined as VM migration being correctly classified due to high utilizations. Since the provided training set is not very large, cross validation has been used to train, test and validate the classification techniques, the provided data is divided into 5 folds and each fold is held out in turn for training and testing.

At the initial stage, data is predicted using local regression provided that prediction window is less than or equal to the migration time as in the bond $x_{k+1} - x_k \leq t_m$ and this is crucial to maintain the SLA then different classification techniques are investigated. Sample data were collected for the six servers named 146CS4, cs-planetlab3, cs-planetlab4, Fobos, jupiter_cs and node1. Figure 2 shows sample resource utilization for 146CS4 for one week. For the purpose of the experiment, SLA has been identified as 90%, 80% and 70% for the 146CS4, cs-planetlab3 and cs-planetlab4 servers and 80%,70% and 60% for Fobos, jupiter_cs and node1 severs for the CPU, memory and Inter VM bandwidth utilization respectively.

Using the workload data described above, all algorithms mentioned in Table I have been applied and analyzed. TPR results are shown in Tables II and III respectively and final ranking is shown in Table IV. Friedman test was conducted to assess the statistical significance for the obtained results. Friedman test was chosen for multi classifier performance assessment due to it is non parametric nature and wide use in multi domain analysis [22]. The null-hypothesis tested is defined as that all classifiers perform the same and the obtained differences are not significantly random. Algorithms will be ranked for each data set separately. Under the null-hypothesis, all the algorithms are equivalent and so their ranks R_j should be calculated as [22]:

$$X_F^2 = [\frac{12}{nk(k+1)}] \sum_{j=1}^{k} (R_j)^2 - 3k(k+1) \quad (14)$$

TABLE I. CLASSIFICATION ALGORITHMS

Algorithm	Table Column Head
Complex Tree	Fine distinction with Maximum number of Leave Splits 100
Medium Tree	Maximum number of Leave Splits 20
Simple Tree	Coars distinction with Maximum number of Leave Splits 4
KNN Coars	Coars distinction between classes wih neightbours set to 100
KNN Cosine	Uses Cosine distance metrics
KNN Cubic	Uses Cubic distance metrics
KNN Fine	Uses fine detailed distance metrics with neighbour set to 1
KNN Medium	Neighbour set to 10
SVM Coars Gausian	Coars distinction with Gausian kernel
SVM Cubic	Uses Cubic kernel
SVM Fine Gausian	Fine detailed distinction Gausian kernel
SVM Liner	Uses Linear Kernel
SVM Medium	Fewer Distinctions are used
SVM Quad	Uses quadretic Kernel

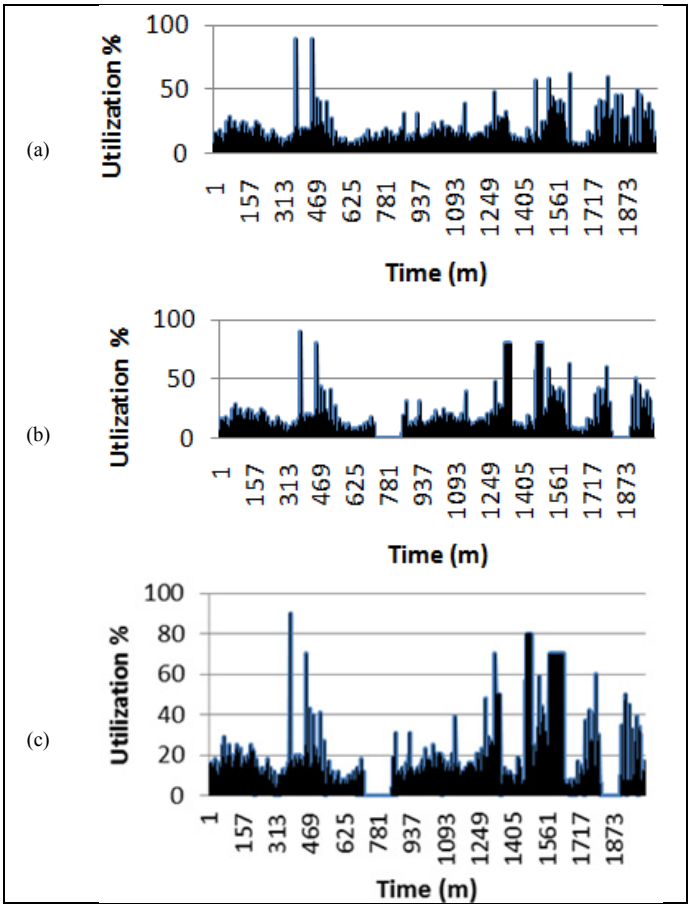

Fig. 2. 146CS4 server resource utilization: (a) CPU (b) Memory (c) InterVM

TABLE II. TPR RESULTS

Algorithm	TPR		
	146CS4 (90-80-70)_105	**cs-planetlab3 (90-80-70)_54**	**cs-planetlab4 (90-80-70)_117**
Complex Tree	89.7	96.4	95.9
Medium Tree	89.7	96.4	97.5
Simple Tree	89.7	96.4	96.7
KNN Coars	88.9	0	57.4
KNN Cosine	89.7	0	96
KNN Cubic	89.7	94.6	95.1
KNN Fine	82.9	96.4	98.4
KNN Medium	89.7	94.6	95.1
SVM Coars Gausian	97.5	100	98.4
SVM Cubic	98.4	98.2	95.1
SVM Fine Gausian	97.5	94.6	98.4
SVM Liner	89.7	98.2	96.7
SVM Medium	98.4	98.4	98.4
SVM Quad	89.7	96.4	98.4

The Friedman statistic is distributed according to X_F^2 with k−1 degrees of freedom. Level of significance at $p < 0.05$ is chosen. If the null-hypothesis is rejected, then we can proceed with a post-hoc test. The Nemenyi test is used to compare classifiers to each other. If the corresponding average ranks differ by at least the critical difference, the performance of two classifiers will be considered significantly different. In this work average TPR, Friedman Rank summations and average ranking are used as performance metrics.

TABLE III. TPR RESULTS

Algorithm	TPR		
	fobos(80-70-60)_25	**jupiter_cs (80-70-60)_17**	**node1 (80-70-60)_57**
Complex Tree	96	88.2	100
Medium Tree	96	88.2	100
Simple Tree	96	88.2	100
KNN Coars	0	0	0
KNN Cosine	52	10	59
KNN Cubic	92	20	86
KNN Fine	92	47.1	93
KNN Medium	92	10	86
SVM Coars Gausian	100	10	80.7
SVM Cubic	96.5	64	93
SVM Fine Gausian	44	5	75.4
SVM Liner	96	11.8	68.4
SVM Medium	88	41.2	41.2
SVM Quad	88	52.9	96.5

TABLE IV. FINAL RANKING

Algorithm	Final Ranking		
	Average TPR	**Friedman Rank Summations**	**Average Ranking**
Medium Tree	94.63333	13	2.166667
Simple Tree	94	14	2. 333333
Complex Tree	94.3	16	2.666667
SVM Cubic	90	16	2.666667
SVM Coars	79.4	17	2.833333
SVM Quad	86.89	17	2.833333
KNN Fine	84.96	20	3.333333
SVM Medium	77.26	23	3.833333
SVM Liner	76.8	24	4
SVM Fine	68.3	27	4.5
KNN Cubic	76.2	28	4.666667
KNN Medium	76.2	28	4.666667
KNN Cosine	49.6	33	5.5
KNN Coars	24.2	41	6.833333

From Table IV we see that the tree based algorithms show better performance compared to other algorithms, whereas KNN shows the worst performance. Medium tree has average TPR of 94.63% with Friedman Rank summations of 13 and overall Friedman ranking of 2.166667. Now to measure statistical significance using Friedman test X_F^2 was calculated as 154.66 and found to be larger than the critical value of p=23.68 which indicates the statistical significance of the obtained TPR values. Accordingly null hypothesis that tested classifiers have the same performance is rejected. Thus, Nemenyi test is used to pinpoint where the significance lies. q_a is found to be 3.353 and based on two pair testing ,we failed to reject the null hypothesis between medium, simple and complex tree in addition to SVM Cubic, SVM Coars, SVM quad and KNN Fine. However, medium tree shows better performance in terms of average TPR, Friedman Rank

summations and average Friedman ranking. In addition to that, medium tree algorithm results were consistent among all tested data.

V. CONCLUSION

In this work, machine learning based approach with modified kernel along with Friedman rank summation and average ranking have been used for dynamic live migration based on adaptive prediction of utilization thresholds. From the analysis of the proposed approach different classification techniques have been used to predict VM migration. It is shown that the regression trees have more accuracy compared to SVM and KNN. This approach can be used to manage SLA in virtualized cloud based data centers for critical applications like telecommunication ones, especially applications with strict SLA. Further analysis can be made on other dynamic consolidation problems such as VM placement following the approach presented in this paper.

REFERENCES

[1] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, A. Warfield, "Xen and the art of virtualization", 19th ACM Symposium on Operating Systems Principles, pp 164-177, 2003

[2] S. Akoush, R. Sohan, A. Rice, A. W. Moore, A. Hopper, "Predicting the Performance of Virtual Machine Migration", IEEE International Symposium on Modeling, Analysis & Simulation of Computer and Telecommunication Systems, 2010

[3] C. Clark, K. Fraser, S. Hand, J. G. Hansen, E. Jul, C. Limpach, I. Pratt, A. Warfield, "Live migration of virtual machines", 2nd Symposium on Networked Systems Design and Implementation, pp 273-286, 2005

[4] A. B. Nagarajan , F. Mueller, C. Engelmann, L. Scott, "Proactive fault tolerance for HPC with Xen virtualization", 21st Annual International Conference on Supercomputing, pp 23–32, 2007

[5] R. Nathuji, K. Schwan, "Virtual power: Coordinated power management in virtualized enterprise systems", ACM SIGOPS Operating Systems Review, Vol. 41, No. 6, pp. 265-278, 2007

[6] Y. Song, H. Wang, Y. Li, B. Feng, Y. Sun, "Multi-tiered on-demand resource scheduling for VM-based data center", 9th IEEE/ACM International Symposium on Cluster Computing and the Grid, pp. 148-155, 2009

[7] VMware Inc, VMware distributed power management concepts and use, 2010

[8] A. Beloglazov, R. Buyya.,Adaptive Threshold-Based Approach for Energy-Efficient Consolidation of Virtual Machines in Cloud Data Centers, Dept. of Computer Science and Software Engineering, University of Melbourne, 2010

[9] T. C. Ferreto, M. A. S. Netto, R. N. Calheiros, C. A. F. De Rose, "Server consolidation with migration control for virtualized data centers", Future Generation Compute Systems, Vol. 27, No. 8, pp 1027–1034, 2011

[10] T. Wood, G.Tarasuk-Levin, P. Shenoy, P. Desnoyers, E. Cecchet, M. D. Corner, "Memory buddies: exploiting page sharing for smart co-location in virtualized data centers", ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments, pp. 31-40,2009

[11] T. Hirofuchi, H. Nakada, S. Itoh, S. Sekiguchi, "Reactive consolidation of virtual machines enabled by post copy live migration", 5th international workshop on Virtualization technologies in distributed computing, pp 11-18, 2011

[12] D. Kakadia, N. Kopri, V. Varma, "Network-aware virtual machine consolidation for large data centers", 3rd International Workshop on Network-Aware Data Management, 2013

[13] H. Mi, H. Wang, G. Yin, Y. Zhou, D. Shi, L. Yuan, "Online self-reconfiguration with performance guarantee for energy-efficient large-scale cloud computing data centers", IEEE International Conference on Service Computing, pp. 514-521, 2010

[14] M. Sindelar, R. K. Sitaraman, P. Shenoy, "Sharing-aware algorithms for virtual machine co location", AMC 23rd symposium on Parallelism in algorithms architectures, pp. 367-378, 2011

[15] A. Beloglazov, "Energy-efficient management of virtual machines in data centers for cloud computing", PhD Thesis, Department of Computer Science, Melbourne University, 2013

[16] T. Chen, X. Gao, G. Chen, "Optimized Virtual Machine Placement with Traffic-Aware Balancing in Data Center Networks", Scientific Programming, Vol. 2016, Article ID 3101658, 2016

[17] Z. Zhou, Z. Hu, K. Li, "Virtual Machine Placement Algorithm for Both Energy-Awareness and SLA Violation Reduction in Cloud Data Centers", Scientific Programming, Vol. 2016, Article ID 5612039, 2016

[18] M. Khalaf Alla H. M., A. Babiker, M. B. M. Amien, M. Hamad, "Review in cloud based next generation telecommunication network", Jurnal Teknology, Vol. 78, No. 6, pp. 51–57, 2016

[19] R. Timofeev, Classification and Regression Trees (CART). Theory and Applications, MSc Thesis, Humboldt University, Berlin, 2004

[20] H. Yu, S. Kim, "SVM Tutorial: Classification, Regression, and Ranking", In: Handbook of Natural Computing, pp. 479-506, Springer, 2012

[21] K. Q. Weinberger, L. K. Saul, "Distance Metric Learning for Large Margin Nearest Neighbor Classification", Journal of Machine Learning Research, Vol. 10, pp. 207-244, 2009

[22] N. Japkowicz, M. Shah, Evaluating Learning Algorithms: A Classification Perspective, Cambridge Press, 2011