SOCIETY FOR SCIENCE AND EDUCATION
UNITED KINGDOM
TMLAI TRANSACTIONS ON MACHINE LEARNING AND ARTIFICIAL INTELLIGENCE
VOLUME 4, ISSUE 1
ISSN 2054-7390

Using Machine Learning Algorithms for Cloud Client Prediction Models in a Web VM Resource Provisioning Environment

Samuel A. Ajila and Akindele A. Bankole
Department of Systems and Computer Engineering, Carleton University, 1125 Colonel By Drive, Ottawa K1S 5B6, ON Canada,
ajila@sce.carleton.ca

ABSTRACT

In order to meet Service Level Agreement (SLA) requirements, efficient scaling of Virtual Machine (VM) resources in cloud computing needs to be provisioned ahead due to the instantiation time required by the VM. One way to do this is by predicting future resource demands. The existing research on VM resource provisioning are either reactive in their approach or use only non-business level metrics. In this research, a Cloud client prediction model for TPC-W benchmark web application is developed and evaluated using three machine learning techniques: Support Vector Regression (SVR), Neural Networks (NN) and Linear Regression (LR). Business level metrics for Response Time and Throughput are included in the prediction model with the aim of providing cloud clients with a more robust scaling decision choice. Results and analysis from the experiments carried out on Amazon Elastic Compute Cloud (EC2) show that Support Vector Regression provides the best prediction model for random-like workload traffic pattern.

Keywords— Cloud Computing, Resource Provisioning, Prediction, Machine Learning, Support Vector Machine, Neural Network, Linear Regression

1 Introduction

The three main markets associated with cloud computing include Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS) and Software-as-a-service (SaaS) [35]. Popular providers of these services are Amazon Elastic Compute Cloud (Amazon EC2), Google App engine and Salesforce respectively. These services (IaaS, PaaS and SaaS) can be made accessible to the public, otherwise called public cloud or restricted for private use (private cloud). Sometimes, these services can be hosted on a hybrid cloud which is a composition of both public and private clouds. One area that researchers have focused on is resource management. Quiroz et al [37] described four stages of data center resource management: Virtual Machine (VM) Provisioning, Resource Provisioning (includes mapping and scheduling requests onto distributed physical resources), Run-time Management and Workload Modeling. In this work, focus will be on VM Provisioning. In trying to meet up with both client Service Level Agreement (SLA) for Quality of Service (QoS) and their own operating cost, cloud providers are faced with the challenges of under-provisioning and over-provisioning Under-provisioning often leads to SLA penalty resulting in revenue loss on the part of the cloud providers [7], [19], [20] and also a poor Quality of Experience (QoE) for the cloud clients. On the other hand, over-provisioning can lead to excessive energy consumption, culminating in high operating cost and waste of resources [7], [19], [20]; though this has no negative impact on the client.

DOI: 10.14738/tmlai.41.1690
Publication Date: 27th February, 2016
URL: http://dx.doi.org/10.14738/tmlai.41.1690

Accurate VM provisioning is a challenging research area that seeks to address the two extremes especially where user workload requirements cannot be determined a priori. Furthermore, VM boot up time has been reported to span various time durations before it is ready to operate [3], [28], [32], [35], [37], [39]; specifically from between 5 and 10 minutes [3], [32], and between 5 and 15 minutes [39]. It is believed that during this time of system and resource unavailability, requests cannot be serviced which can lead to penalty on the part of the cloud providers. Multiplying this lag time over several server instantiations in a data center can result in heavy cumulative penalties. These penalties or compensations to the client cannot redeem the poor QoE the customers must have perceived. To this end, cloud clients can take a proactive step to mitigate reputational loss by controlling their VM provisioning using the Cloud provider's API. One of the numerous strategies available to the client is a predictive approach wherein insight into the future resource usage (CPU, memory, network and disk I/O utilization) may help in scaling decisions ahead of time, thus, compensating for the start-up lag time [13]. Present monitoring metrics made available to clients are limited to CPU utilization, memory and network. These may not give a holistic view of the QoS. For instance, a web server may not necessarily be saturated for an SLA breach to occur. Therefore, CPU based scaling decisions may not achieve the goal of accurate VM provisioning. Several predictive resource usage approaches exist, such as pattern matching and machine learning. In fact, the use of machine learning as a predictive tool to allow dynamic scaling is one way of mitigating the challenge of resource scaling [6]. In this work, we evaluated the ability in forecasting future resource usage in a multi-tier web application of the following machine learning techniques: Neural Network (NN), Linear Regression (LR) and Support Vector Regression (SVR). In addition, the cloud watch metrics is extended by including two business level metrics: Throughput and Response time.

We used Amazon EC2, a public cloud for resource provisioning. The selection of this cloud provider is based on the availability of documentation, open source Application Programming Interface (API) and a vast array of instance types to select from (representing either on-demand, reserved or spot instances). Finally, being an early entrant in providing IaaS, Amazon EC2 boasts of a very good technical support team.

The rest of this paper is organized as follows. Section II discusses the background, state of the art, and the related work, while section III presents the methodology employed in this research work. Section IV discusses the experimental setup by emphasizing feature selection, data collection, feature reduction, CPU utilization and training, parameter selection for response time and throughput. Section V presents the simulation and analyses the results for each model (LR, NN, and SVR). A comparison of the models is carried out and a sensitivity analysis using Little's law is done. Section VI gives a conclusion and suggestion for future work.

2 Background, State Of The Art and Related Works

2.1 Background Works

In our exploratory work [8], we developed and evaluated cloud client prediction models for TPC-W benchmark web application based on Neural Network (NN), Linear Regression (LR) and Support Vector Machine (SVM), and using a linear traffic workload (TPC-W) pattern and 170 minutes of experiment time in terms of data collection. This initial exploratory work sets the stage for a second phase based on a more random and non-linear traffic pattern [1] implemented on a public cloud environment: Amazon Elastic Compute Cloud (Amazon EC2) infrastructure. In this phase, we maintained the same architecture as the first phase and increased the experiment time to 532 minutes (compared to 170 minutes for the first

phase). We noticed that during the second phase of the research work the feature selection, parameter setting for LR, NN, and SVR, and data training play crucial roles in the simulation results. Due to the big variance between the actual and predicted resource provisioning (cf. Appendix C and especially in the case of LR and NN), we decided to take a further study of the feature selection, parameter setting, and training to make sure the final result is as close to reality as possible. So, the content of this paper is a more complete research that includes:

- We maintained the same architecture as phases 1 and 2.
- We retained the inclusion of two SLA metrics: response time and throughput.
- We studied the effects of the various TPC-W workloads used in the experiment on the system performance of the database tier.
- We carried out simulations for the selection of parameters and features for the purposes of data training and learning.
- We maintained the same evaluation of the prediction capability of Support Vector Machine, Neural Network and Linear Regression using three benchmark workloads from TPC-W in:
    - An extended experimentation time frame of 532 minutes as opposed to 170 minutes in the first phase.
    - Traffic patterns wherein workloads spike up and down and then stabilize in a randomized manner; a pattern we reckon to be somewhat realistic.

Amazon EC2 offers three different instance purchasing options: On-Demand Instances, Reserved Instances and Spot Instances.

2.2 State of the Art

2.2.1 Amazon Elastic Compute Cloud (Amazon EC2)

Amazon EC2 is a web service that provides resizable compute capacity in the cloud. Amazon EC2 benefits include: elasticity, multiple instance types, operating systems and software packages, and a commitment to 99.95% availability for each EC2 Region. Finally and importantly, EC2 offer a very low per hour pay rate for the compute capacity consumed. Amazon EC2 has a range of instance types [3]. TABLE 1 summarizes some instance types and their specifications.

Table 1 Amazon Instance Type Specifications

Instance Type	Platform	CPU	Memory (GB)	Disk (GB)	Cost/Hr ($)
M1.Small	32 or 64-bit	1 ECU	1.7	160	0.060
M1.Medium	32 or 64-bit	2 ECU	3.7	410	0.120
M1.Large	64-bit	4 ECU	7.5	850	0.240
M1.Extra Large	64-bit	8 ECU	15	1690	0.480
T1.micro	32 or 64-bit	Up to 2 ECU	0.613	8	0.020
High Memory Extra Large	64-bit	6.5 ECU	17.1	420	0.410
High-CPU Medium	32 or 64-bit	5 ECU	1.7	350	0.145

2.3 Machine Learning

According to Wang and Summers [46], machine learning is the study of algorithms that run on computer systems which can learn complex relationships or patterns from empirical data and make accurate decisions. It is classified machine learning into supervised learning, semi-supervised learning and unsupervised learning. Supervised learning is to deduce a functional relationship from training data that generalizes well to testing data. Unsupervised learning on the other hand seeks to discover relationships between samples or reveal the latent variables behind the observations. Semi-supervised learning falls between supervised and unsupervised by utilizing both labeled and unlabeled data during the training phase [46]. Supervised learning has been employed because its purpose matches the problem area of this work.

2.4 Related Works

Several authors have worked in the area of resource provisioning using different approaches. This section presents some of the related techniques (Threshold-based, Control Theory, Reinforcement Learning, and Time Series).

Han et al [24] proposed and implemented a lightweight approach to enable cost-effective elasticity for cloud applications. Their solution which was centered on the cloud provider's side employed two scaling techniques to support QoS requirements of the application owner: Self-healing and Resource-level scaling. For self-healing, idle resources of one VM can be used to release the overloaded resources in another while resource-level scaling is based on using unallocated resources at a particular physical machine to scale up a VM executing on it. Though their scaling technique (scale up or down) can be completed very fast; in a matter of milliseconds, the reactive scaling mechanism employed would definitely lead to SLA penalty when a new VM provisioning is required. Furthermore, some resource providers may choose not to export the access to hypervisor-level actuators of the cloud computing infrastructure, such as controlling the CPU and memory allocations [34]. This constraint makes their work restrictive and not easily generalized. The work by Hasan, M.Z. et al [25] provided cloud clients (tenants) the ability to set policies which indicated conditions under which resources should be auto-scaled. Their Integrated and Autonomic Cloud Resource Scaler (IACRS) integrates performance metrics from other multiple domains (compute, network and storage) in making scaling decisions. The proposed algorithm is a departure from scaling decisions using the regular singular metric (CPU), the algorithm has not been implemented and thus provides no performance evaluation of any kind. In addition, their approach was also reactive and VM boot up or lag time would result in SLA penalty.

According to Lorido-Botran, T. et al [35], control theory has been applied to automate the management of web server systems and data centers, and it shows interesting results in cloud computing. Control systems are either closed loop or open loop systems. For the open loop system, control action does not depend on the system output (non-feedback) while in the case of closed loop; the controller output $\mu(t)$ tries to force the system output $C(t)$ to be equal to the reference input $R(t)$ at any time t irrespective of the disturbance ΔD [5]. Ghanbari, H. et al [26] used control theory to find a proper reservation action (immediate, in-advance, best effort or auction based reservation) at any given time based on the current system state. This approach tried to minimize the average response time and at the same time minimize resource cost by selecting the most appropriate reservation action. Though the authors agreed that best effort or on-demand reservation may not be provisioned in a timely manner, no discussion on how to

handle this possible reservation action was mentioned. Lim, H.C. et al [34] proposed that cloud customers should be empowered to operate their own dynamic controllers, outside or as extensions to the cloud platform itself. The solution centered on adapting the control policy for cases where fine grained actuators for adjusting CPU entitlements are not made available by cloud providers. They introduced proportional thresholding policy which modifies an integral control by using a dynamic target range (CPU utilization for example), instead of a single target value.

Reinforcement learning (RL) is another type of automatic decision-making approach that can be applied to VM provisioning [35]. It is well suited to cloud computing as it does not require a priori knowledge of the application performance model, but rather learns it as the application runs [18]. Dutreilh, X. et al [18] used the Q-learning algorithm for their work as the Q-function is easy to learn from experience. The approach is; given a controlled system, the learning agent repeatedly observes the current state (workload, number of VMs and performance SLA), takes an action and then a transition to a new state occurs. The new state and corresponding reward is then observed. However, because defining the policy from which decisions can be chosen can take a long time (exploration and exploitation [38]) the authors introduced a convergence speedup phase at regular intervals to hasten the learning process.

Time series analysis could be used to find repeating patterns in the input workload or try to forecast future values. For example, a certain performance metric, such as average CPU load (utilization) will be periodically sampled at fixed intervals. The result will be a time-series X containing a sequence of the last w observations [35]: $X = x_t, x_{t-1}, x_{t-2}, \dots, x_{t-w+1}$ Several authors have used time series analysis for dynamic VM provisioning; for example, the work presented by Sadeka, I. et al [39] also concerns the use of time-series analysis for adaptive resource provisioning in the cloud. Their proposed prediction framework used statistical models that are able to speculate the future surge in resource requirement; thereby enabling proactive scaling to handle temporal bursty workload. The authors evaluated the prediction capabilities of two machine learning algorithms: Neural Network and Linear Regression. Historical data was first collected by using the TPC-W benchmarking e-commerce application hosted on Amazon cloud. The sampled CPU utilization dataset was then used to train both learning algorithms after which a forecast of the future CPU utilization on a 12 minute interval (the average boot up time for a new VM instance) was carried out. The same training procedure was employed with the sliding window technique which works by anchoring the left point of a potential segment at the first data point of a time series, then attempting to approximate the data to the right with increasing longer segments [29]. Performance evaluation of the two learning algorithms showed that Neural Network demonstrated enhanced accurate prediction capability compared to Linear Regression.

Our work aims at analyzing the problem of resource provisioning from the Cloud client's perspective with the ability of the hosted application to make scaling decisions by not only evaluating the future resource utilization but also considering business SLA metrics of response time and throughput; thus providing a tripartite auto-scaling decision matrix. The prediction model that we used to achieve this in a multi-tier web application is a set of machine learning techniques – Neural Network, Linear Regression and Support Vector Machine. These techniques would be evaluated using Mean Absolute Percentage Error (MAPE), Mean Absolute Error (MAE) Root Mean Square Error (RMSE) and PRED (25).

3 Methodology and Tools

In this section, we describe our methodology for predictive resource provisioning for multi-tier web applications using machine learning to develop the performance prediction model (Fig. 1). NN and LR have been widely explored by several authors in building prediction models [13], [11], [19], and [9]. Recently SVM, a powerful classification technique [11] has been gaining significant popularity in time series and regression prediction [8], [10], and [14]. We introduce these learning techniques below.

3.1 Linear Regression

A linear regression model assumes that the regression function $E(Y|X)$ is linear in the input $X_1, \dots, X_p$. It is simple and often provides an adequate and interpretable description of how the inputs affect the output [45]. It is one of the staple methods in statistics and it finds application in numeric prediction especially where both the output or target class and the attributes or features are numeric [47]. The linear regression model has the form:

$$f(X) = \beta_0 + \sum_{j=1}^{p} X_j \beta_j \dots \quad (1).$$

The β_j's are the unknown parameters or coefficients, and the variables X_j are the quantitative inputs or attributes. Typically, the parameters β are estimated from a set of training data $(x_1, y_1) \dots (x_N, y_N)$ [45], [47]. Each $(x_{i1}, x_{i2}, \dots, x_{ip})^{\mathrm{T}}$ is a vector of feature measurements for the ith case. The most popular estimation method is least squares, wherein we pick the coefficients $\beta = (\beta_0, \beta_1, \dots, \beta_p)^{\mathrm{T}}$ to minimize the residual sum of squares (RSS) [58]

$$RSS(\beta) = \sum_{i=1}^{N} (y_i - f(x_i))^2 = \sum_{i=1}^{N} (y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j))^2 \quad (2)$$

Linear regression is an excellent, simple scheme for numeric prediction. However, linear models suffer from the disadvantage of non-linearity: if the data exhibits a non-linear dependency, the best fitting straight line will be found [47].

3.2 Neural Network

A neural network (NN) is a two-stage regression or classification model, typically represented by a network diagram [58]. Several variants of neural network classifier (algorithm) exist, some of which are; feed-forward, back-propagation, time delay and error correction neural network classifier. According to Trevor, H. et al [45], there is typically one output unit Y_1 at the top i.e. $K = 1$ for regression problems though multiple quantitative responses can be handled in a seamless fashion. Derived features Z_m are created from linear combinations of the input, and then the target Y_k is modeled as a function of linear combinations of the Z_m,

$$\begin{aligned} Z_m &= \sigma(\alpha_{0m} + \alpha_m^T X), m = 1, \dots, M, \\ T_k &= \beta_{0k} + \beta_k^T Z, k = 1, \dots, K, \\ f_k(X) &= g_k(T), k = 1, \dots, K, \end{aligned} \quad (3)$$

Where $Z = (Z_1, Z_2, \dots, Z_M)$, and $T = (T_1, T_2, \dots, T_K)$. The activation function $\sigma(v)$ is usually chosen to be the sigmoid $\sigma(v) = \frac{1}{(1+e^v)}$. Sometimes, Gaussian radial basis functions are used for $\sigma(v)$, producing what is known as a radial basis function network [45]. The units in the middle of the network, computing the derived features Z_m, are called hidden units because the values Z_m are not directly observed. The neural

network model has unknown parameters, often called weights, and we seek values for them that make the model fit the training data well. The complete set of weights θ, consists of

$$\{\alpha_{0m}, \alpha_m; m = 1,2, \dots, M\} \quad M(p+1) \text{ weights and } \{\beta_{0k}, \beta_k; k = 1,2, \dots, K\} \quad K(M+1) \text{ weights} \tag{4}$$

For regression, the sum-of-squares errors is used as the error function [45]

$$R(\theta) = \sum_{k=1}^{K} \sum_{i=1}^{N} (y_{ik} - f_k(x_i))^2. \tag{5}$$

The generic approach to minimizing $R(\theta)$ is by gradient decent, called back-propagation.

3.3 Support Vector Machine

According to Sakr, G.E et al [40], Support Vector Machine (SVM) is a machine learning algorithm that uses a linear hyperplane to create a classifier with a maximal margin. For cases where the data is not linearly separable, the SVM maps the data into a higher dimensional space called the feature space. It has the advantage of reducing problems of overfitting or local minima. In addition, it is based on structural risk minimization as opposed to the empirical risk minimization of neural networks [30]. SVM now finds application in regression and is termed Support Vector Regression (SVR). The goal of SVR is to find a function that has at most ε (the precision by which the function is to be approximated) deviation from the actual obtained target for all training data with as much flatness as possible [41], [42].

Given training data (x_i, y_i) $(i = 1, \dots l)$, where x is an n-dimensional input with $x \in R^n$ and the output is $y \in R$, the linear regression model can be written as [22], [42]:

$$f(x) = < w, x > + b\,, \; w, x \in R^n, b \in R \tag{6}$$

where $f(x)$ is the target function and $<.,.>$ denotes the dot product in R^n. To achieve the flatness, w is minimized i.e. $||w||^2 = < w, w >$. This can further be written as a convex optimization problem: minimize $\frac{1}{2}||w||^2$ subject to the constraint

$$\begin{cases} y_i - < w, x_i > - b \leq \varepsilon \\ < w, x_i > + b - y_i \leq \varepsilon \end{cases} \tag{7}$$

Equation (7) assumes that there is always a function f that approximates all pairs of (x_i, y_i) with ε precision. However, according to Dashevskiy and Luo [17]; this function may not be obtainable thus necessitating the introduction of slack variables γ_i, γ_i^* to handle infeasible constraints. Therefore, equation (7) leads to

Minimize $\frac{1}{2}||w||^2 + C \sum_{i=1}^{n} (\gamma_i + \gamma_i^*)$ subject to

$$\begin{cases} yi - < w, x_i > - b \leq \varepsilon + \gamma_i \\ < w, x_i > + b - yi \leq \varepsilon + \gamma_i^* \\ \gamma_i, \gamma_i^* \geq 0 \end{cases} \tag{8}$$

The constant $C > 0$ determines the trade-off between the flatness of f and the amount up to which deviations larger than ε are tolerated. Equation (8) can be reformulated and solved to give the optimal Lagrange multipliers $\alpha \; and \; \alpha^*$ with w and b given as

$$w = \sum_{i=1}^{n} (\alpha - \alpha^*)\, x_i \text{ and} \tag{9}$$

$$b = -\frac{1}{2} < w, (x_r + x_s), x_r and\ x_s \text{ are the support vectors.} \quad (10)$$

Inserting (9) and (10) into (6) yields

$$f(x) = \sum_{i=1}^{n} (\alpha - \alpha^*) < x_i, x > + b \quad (11)$$

This generic approach is usually extended for nonlinear functions. This is done by replacing x_i with $\varphi(x_i)$; a feature space that linearizes the relation between x_i and y_i [20].

Therefore, (11) can be re-written as:

$$f(x) = \sum_{i=1}^{n} (\alpha - \alpha^*)\ K < x_i, x > + b \quad (12)$$

where $K < x_i, x > = < \varphi(x_i), \varphi(x)$ is the so called kernel function.

The four basic kernels used are [20]:

- Linear: $K(x_i, x) = x_i^T x$
- Polynomial: $K(x_i, x) = (\gamma x_i^T x + r)^d, \gamma > 0.$
- Radial basis function (RBF): $K(x_i, x) = \exp(-\gamma ||x_i - x||^2), \gamma > 0.$
- Sigmoid: $K(x_i, x) = \tanh(\gamma x_i^T x + r).$

Where $\gamma, r\ and\ d$ are kernel parameters

4 Setup of the Experiment

4.1 System Architecture

The cloud client prediction model for cloud resource provisioning in a multitier web application environment has the following components in the overall architecture (Figure 1):

- Client infrastructure: This is a High-CPU Instance with 1.7 GB of memory, 5 EC2 Compute Units (2 virtual cores with 2.5 EC2 Compute Units each) and 350 GB of instance storage. The TPC-W emulator is executed on this infrastructure
- Web server infrastructure: This is a 3.75 GB of memory, 2 EC2 Compute Unit (1 virtual core with 2 EC2 Compute Unit) and 410GB instance storage. The Java implementation of the TPC-W benchmark is deployed on a Tomcat web server environment
- Database server infrastructure: This is a 7.5 GB of memory, 4 EC2 Compute Units (2 virtual cores with 2 EC2 Compute Units each) and 850GB instance storage. MYSQL is the relational database management system used.

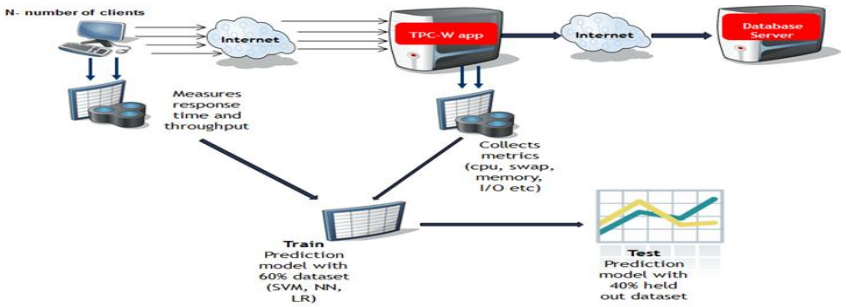

Figure 1. Architecture of the System

The Waikato Environment for Knowledge Analysis (WEKA) is used to train and test the three machine learning techniques. WEKA is a Java data mining software that has a collection of machine learning algorithms including SVR, NN and LR [22]. In this work the Explorer application (a GUI option) which is an environment for exploring data with WEKA is selected. The choice of WEKA is hinged on its open source availability and rich suite of several learning algorithms including SVR, NN and LR.

4.2 Experimental Setup

4.2.1 Feature Selection

We base our prediction models on a continuous observation of a number of specific features [40]. The following initial features are selected for the three target values (CPU utilization, response time and throughput) [2]:

- DiskReadOps: This metric identifies the rate at which an application reads a disk.
- DiskWriteOps: This metric identifies the rate at which an application writes to a hard disk.
- DiskReadBytes: This metric is used to determine the volume of the data the application reads from the hard disk of the instance.
- DiskWriteBytes: This metric is used to determine the volume of the data the application writes onto the hard disk of the instance.
- NetworkIn: This metric identifies the volume of incoming network traffic to an application on a single instance.
- NetworkOut: This metric identifies the volume of outgoing network traffic to an application on a single instance.
- Memory Utilized: This metric collects and sends the memory utilization excluding cache memory and buffers.
- Memory Available: This metric collects and sends available memory used by the operating system and the application.
- Swap Utilized: The amount of swap spaced utilized.

4.2.2 Data collection using TPC-W benchmark

TPC-W has been used by several authors [12], [48] for resource provisioning and capacity planning [39]. Similar to Sadeka et al. [39], a Java implementation of TPC-W that emulates an online bookshop is used. It is deployed on a-two-tier architecture as depicted in Fig. 1. The system resource metrics like CPU utilization and memory used are collected from the web server while the response time and throughput are measured from the "client's" end. TPC-W has a remote browser emulator (RBE) that allows a single node to emulate several clients. The response time in this context is the time lag between when a page request is made to the reception of the last byte of the HTML response page. Similarly, the throughput is the total number of web interactions completed during an experimental run. TPC-W has 14 web interactions characteristics of which 6 belong to the Browsing category and the other 8 to the Ordering category. The three workload mixes used by TPC-W: Browsing, Shopping and Ordering are made up of a combination of Browsing and Ordering categories. For instance, Browsing mix is made up of 95% Browsing category (consists of 6 web interactions that make up the 95%) and 5% Ordering category (consists of 8 web interactions that make up the 5%). For this experiment, the N – number of clients in Fig. 1 refers to

the number of users participating in any of the three workload mixes. Table 2 shows some randomly selected workload mix used in the course of the experiments. During the 1st to 7th minute, there are 84 Shopping mix users, 52 Browsing mix users and 52 Ordering mix users simultaneously making requests to the Web server (a total of 188 user requests). Each workload mix runs for 7 minutes and the choice of this time interval is intuitive as there is no documented time frame for how long workload mix should run. By adjusting the number of emulated clients in a random pattern, a changing workload that sends requests to the web server in a continuous fashion throughout the duration of the experiment is created. Amazon EC2 has a web service that enables monitoring, managing and publishing of various metrics [2]. The traditional ***Top*** command in Linux is not used as this command give metrics for the underlying host and not the actual instance [4]. Feature readings (defined in Section IV) are collected every 60 seconds by some customized Java batch scripts. For instance, to collect CPU utilization the Amazon EC2 API is: "*"mon-get-stats CPUUtilization --start-time 2013-01-08T19:17:00 --end-time 2013-01-08T19:50:00 --period 60 --statistics "+stat+" --namespace "+namesp+" --dimensions "+ dimen"*. The "*--statistics"* parameter returns the average reading over 60 seconds while "*--dimensions*" is the instance-id; the "*--namespace"* is a conceptual container for metrics [2] and for this work the EC2 namespace is used.

Table 2-Experimental Workload Mix for Some Selected Time

Time (minutes)	1-7	56-63	154-161	350-357	490-497	504-511
Shopping mix users	84	168	16	180	248	160
Browsing mix users	52	112	36	320	192	160
Ordering mix users	52	108	28	224	268	160
Total user Requests	**188**	**388**	**80**	**724**	**708**	**480**

Table 3 - Performance Metrics and their Calculations

Metric	Calculation
MAPE (Mean Absolute Percentage Error)	$\frac{1}{n}\sum_{i=1}^{n}\frac{\lvert a_i-p_i\rvert}{a_i}$ where a_i and p_i are the actual and predicted values respectively
RMSE (Root Mean Square Error)	$\sqrt{\frac{\sum_{i=1}^{n}(a_i-p_i)^2}{n}}$
MAE (Mean Absolute Error)	$\frac{1}{n}\sum_{i=1}^{n}\lvert p_i-a_i\rvert$
PRED 25	No. of observations with relative error ≤ 25% / No. of observation

The duration for the entire experiment is 532 minutes. The data is then used to build the prediction model from which forecast can be made for future resource requirement and business level metrics of the web server.

4.2.3 Feature reduction

The importance of selecting the right features for prediction modeling is very critical to reducing the potential source of error as the data mining principle of "junk in, junk out" means erroneous predictions can occur even if the prediction algorithm is optimal [40]. The Weka tool [23] is used to determine the relevance of each feature in an instance to the target class (CPU, response time and throughput). Using attribute selection functionality, the least correlated attributes are eliminated.

4.2.4 Data preprocessing

During this phase, the 6 input features (including CPU utilization, Response Time and Throughput) are scaled to values between 0 and 1. Normalization or scaling is carried out by finding the highest value within each input in the 532 dataset, and dividing all the values within the same feature by the maximum value. The main advantage for normalizing is to avoid attributes in greater numeric ranges dominating those in smaller numeric range [16].

4.2.5 Training of Dataset

As discussed earlier, the goal of this work is to build prediction model that can forecast future resource requirement (using CPU utilization) and two business level SLA – response time and throughput. Towards this end, the normalized sampled dataset is used to train the prediction model. First, training with CPU utilization as the target class is done using the three machine learning techniques discussed above. Next, using the same dataset, models for both response time and throughput are trained. The metrics in Table 3 are used to evaluate both training and testing results of the models. These metrics have been used by other authors [28], [39], and since this work seeks to compare predicted and actual target values, these metrics are good fit.

4.2.6 CPU utilization

- **Neural Network**: Using the Weka tool, the model is trained with the following parameters: learning rate ρ = 0.38, number of hidden layers = 1, number of hidden neurons = 4, momentum = 0.2 and epoch or training time =10000. These parameters gave the best results after several trials based on simulations. Parameter selection is usually based on heuristics as there is no mathematical formula or theory that has been proposed to select the best parameters
- **Linear Regression**: The Weka tool is also used to train the model. The only parameter set was the ridge parameter which was set to the default of 1.0E-8. The ridge parameter minimizes the penalized residual sum of squares. The parameter controls the amount by which the regression coefficients are shrank. The larger the ridge, the greater the shrinkage [45]. Varying the value during simulation had no significant impact on the target value.
- **Support Vector Regression**: SVR has four kernels that can be used to train a model. They are: Linear, Polynomial, Radial Basis Function (RBF) and Sigmoid [30]. The four different kernels were tried with RBF returning the most promising result with the least MAPE value. This is expected as RBF can handle the case when the relationship between features and target value is nonlinear [16]. Before training, the Grid Parameter Search for Regression with cross validation is used (v-fold cross validation) [15] to estimate the C and γ. Cross-validation is a technique used to avoid the over fitting problem [16], [30]. The search range for C was between 2^{-3} to 2^{5} and that of γ between 2^{-10} and 2^{2}. These values are purely heuristics [16], [30]. The search returns the optimal C and γ by using the Mean Square Error to evaluate the accuracy of the various C and γ combinations. The best C and λ was 14 and 0.0092. Using these parameters, the model was trained with the Radial Basis Function (RBF) Kernel.

4.2.7 Response time and Throughput

The business SLA metrics were approached in a similar way as CPU utilization (above). For Throughput, SVR's C and γ were 8 and 0.009 respectively. NN values for ρ, hidden layer, hidden neurons and momentum were 0.4, 1, 3 and 0.2 respectively. Finally the ridge parameter for LR was 1.0E-8. Also, for Response time; SVR's C and γ were 1.05 and 0.009 respectively. NN values for ρ, hidden layer, hidden neurons and momentum were 0.5, 1, 3 and 0.2 respectively. The ridge parameter for LR was 1.0E-8. Tables IV, V and VI list the final parameters used for training the SVM, NN and LR model.

Table 4 - Final Parameters of the SVR CPU Utilization and SLA Prediction Model

Metric	CPU Utilization	Response time	Throughput
C parameter search range	2^{-3} - 2^{5}	2^{-3} - 2^{5}	2^{-3} - 2^{5}
γ parameter search range	2^{-10} - 2^{2}	2^{-10} - 2^{2}	2^{-10} - 2^{2}
C	14	1.05	8
γ	0.0092	0.009	0.009

Table 5 - Final Parameters of The NN CPU Utilization and SLA Prediction Model

Metric	CPU Utilization	Response Time	Throughput
Learning Rate	0.38	0.5	0.4
Number of Hidden Layers	1	1	1
Number of Hidden Neurons	4	3	3
Momentum	0.2	0.2	0.2

This step is very significant as it is possible to obtain impressive results for training data but dismal results when it comes to testing. Furthermore, prediction accuracy is based on the held out test dataset. A training-to-testing ratio of 60%:40% (319:213 minutes) was used as this gave the optimal prediction output for the models after several simulations. A 12 minute prediction interval is adopted. This is based on reports from previous works [3], [32] regarding VM boot up time and motivation from the work of [39]. The prediction trend at the 9th, 10th, 11th and 12th minute is included in Section V to check for consistency and reliability in the prediction models of SVR, NN and LR.

Table 6-Final Parameters Of The LR CPU Utilization And SLA Prediction Model

Metric	CPU Utilization	Response Time	Throughput
Ridge Parameter	1.0E-8	1.0E-8	1.0E-8

5 Simulation Results and Analysis

This section presents the results of the various experimental simulations for determining the prediction capability of the three machine learning techniques: SVR, NN and LR. The objective is to evaluate the accuracy of the selected machine techniques in forecasting future resource usage for random workload traffic patterns over an extended period of time. In addition, the inclusion of business level metrics to the prediction is considered. Results include both training and test datasets.

5.1 Linear Regression Models

5.1.1 CPU utilization training and test results

Table 7 shows the training and testing performance metric results for the LR. Furthermore, Fig. 2 and 3 present the graphical representation of the actual and predicted CPU utilization for the 319 minutes of training and 213 minutes of testing respectively. The training MAPE value was about three times that of testing. The reason for this is that the training dataset's values are steeper than the test dataset. For instance, at the 137th minute, the CPU utilization is approximately 65 percent and this falls to about 5 percent at the 139th minute. Fig. 2 shows this graphically. Aside the test model MAPE result, the other three metric values are worse than the training model result. The reason for this is attributed to the negative predicted values and also the general poor forecasting ability of LR in a non-linear traffic pattern as captured in Figure 3.

Table 7-CPU Utilization Training and Test Performance Metric

Model	MAPE	RMSE	MAE	Pred (25)
Training	113.31	14.70	11.11	0.51
Test	36.19	22.13	15.98	0.36

Table 8-Throughput Training and Test Performance Metric

Model	MAPE	RMSE	MAE	Pred (25)
Training	75.25	4.45	3.22	0.57
Test	24.62	3.72	2.87	0.63

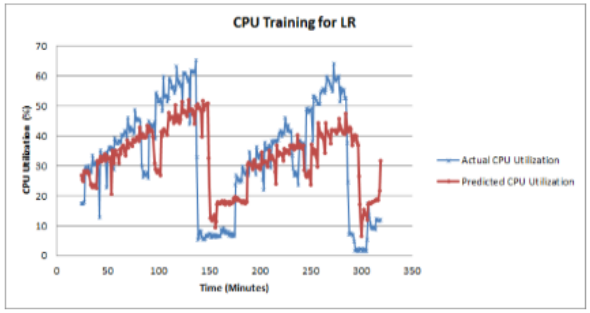

Figure 2. CPU Utilization Actual and Predicted Training Output using LR

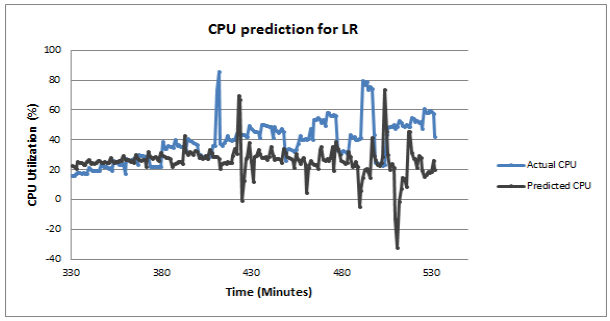

Figure 3. CPU Utilization's Actual and Predicted Test Output using LR

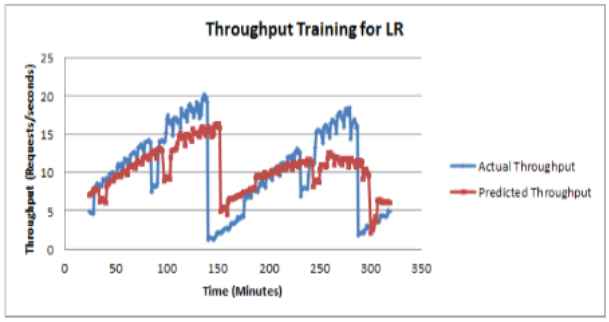

Figure 4. Throughput's Actual and Predicted training using LR

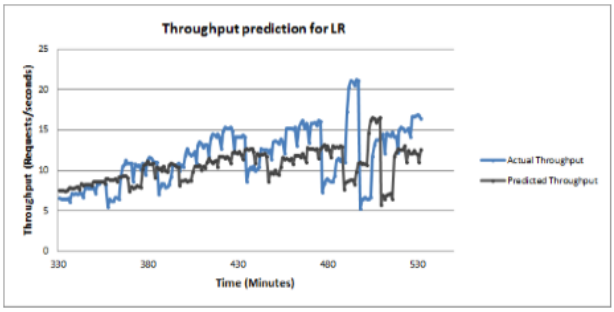

Figure. 5. Throughput Actual and Predicted test output using LR

5.1.2 Throughput training and test results

The Throughput's training and test performance metric results are shown in Table 8. In addition, Fig. 4 and 5 present the graphical representation of the actual and predicted Throughput values for both training and test dataset respectively. The training and test interval are the same as that of CPU Utilization. All test metric results are better than training results. The variances in the actual and predicted values are not as wide as that of CPU utilization. It can also be observed that there are more spikes in the training dataset than in the test dataset, thus further making the test result better than the training result. Fig. 5 shows the Throughput forecast.

5.1.3 Response time training and test results

The Response time's training and test performance metric results are shown in Table 9. Furthermore, Fig. 6 and 7 present the graphical representation of the actual and predicted Response time values for both training and testing dataset respectively. The difference in the training and test results is very close as the traffic patterns are almost similar. The accuracy for this metric is very impressive, though it can be said that the variance between the actual and predicted values are lower than that of the CPU utilization. The minimum and maximum response time values are between 1-12 seconds.

Table 9 - Response Time Training and Test Performance Metric

Model	MAPE	RMSE	MAE	PRED(25)
Training	17.58	1.24	0.81	0.90
Test	12.35	1.39	1.11	0.91

Table 10 - CPU Utilization Training and Test Performance Metric

Model	MAPE	RMSE	MAE	PRED(25)
Training	105.63	14.08	9.48	0.59
Test	50.46	31.08	19.82	0.34

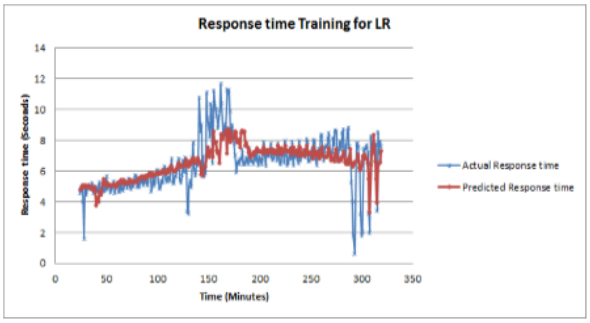

Figure 6. Response time Actual and Predicted training output using LR

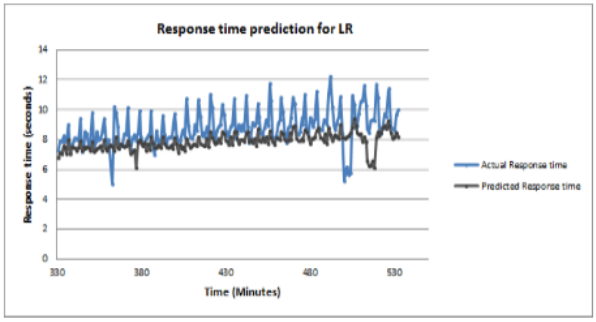

Figure 7. Response time Actual and Predicted test output using LR

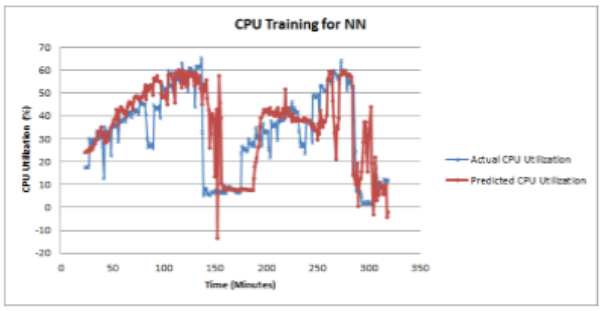

Figure 8. CPU Utilization Actual and Predicted training output using NN

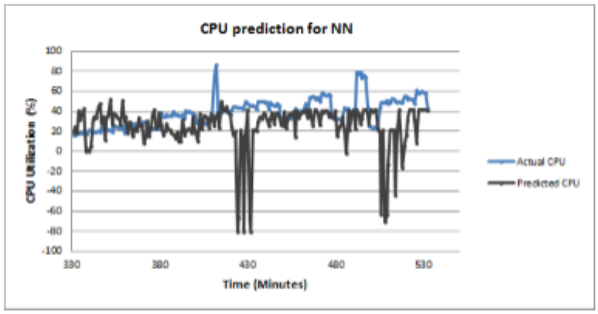

Figure 9. CPU Utilization Actual and Predicted test output using NN

5.2 Neural Network Models

5.2.1 CPU Utilization training and test results

The CPU utilization training and test performance metric results for NN model is shown in Table 10. Fig. 8 and 9 present the graphical representation of the actual and predicted CPU utilization for the 319 minutes of training and 213 minutes of testing respectively. The training MAPE value is also very high and the reason for this is similar to the explanation given in sub-section on LR. It is also observed that the test dataset has some series of negative values in its prediction as shown in Fig. 9. The number of negative predicted values which is more than that of LR contributed to the poorer test metric values.

5.2.2 Throughput training and test results

The Throughput's training and test performance metric results are shown in Table 11. In addition, Fig. 10 and 11 present the graphical representation of the actual and predicted Throughput values for both training and test dataset respectively. Comparing the training and test model results, the test model's metric values are quite better than the training model. Some predicted throughput values in the training model are negative (Fig. 9). However, negative prediction is absent for the test dataset (Fig. 10).

Table 11 - Throughput Training and Test Performance Metric

Model	MAPE	RMSE	MAE	PRED(25)
Training	56.46	6.85	4.96	0.30
Test	38.90	6.12	4.46	0.47

Table 12 - Response Time Training and Test Performance Metric

Model	MAPE	RMSE	MAE	PRED(25)
Training	36.28	3.51	2.38	0.58
Test	17.84	2.02	1.64	0.75

5.2.3 Response time training and test results

The Response time's training and test performance metric results are shown in Table 12. Furthermore, Fig. 12 and 13 present the graphical representation of the actual and predicted Response time values for both training and test dataset respectively. The test model's metric values are better than that of the training model as Fig. 12 shows more erroneous prediction than Fig. 13. That is, the high training metric values (MAPE, RMSE and MAE) are attributed to the large variations in some predicted and actual Response time as shown in Figure 12.

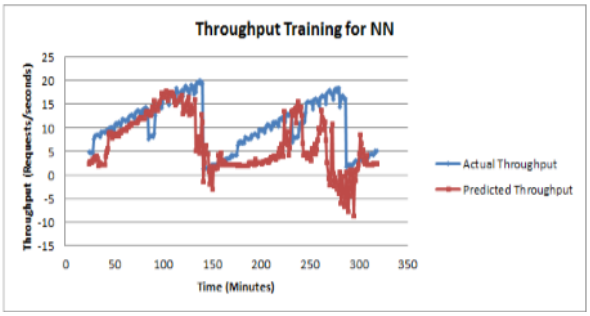

Figure 10 Throughput Actual and Predicted training for NN

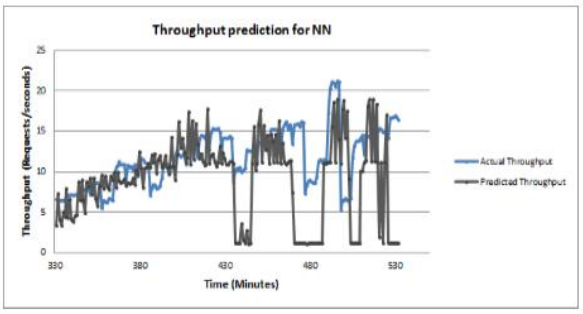

Figure 11 Throughput Actual and Predicted test output for NN

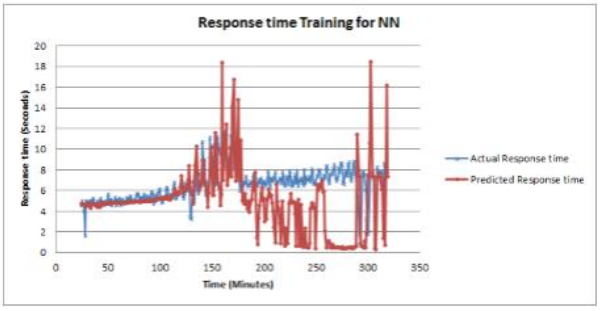

Figure 12 Response time Actual and Predicted training for NN

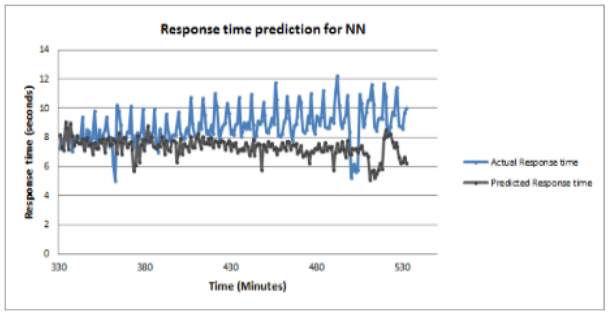

Figure 13 Response time Actual and Predicted test for NN

5.3 Support Vector Machine (Regression) Models

Similar to the two previous sub-sections, CPU utilization, Throughput and Response time training and testing dataset results are presented.

5.3.1 CPU Utilization training and test results

The CPU utilization training and test performance metric results for SVR model is shown in Table 13. Fig. 14 and 15 present the graphical representation of the actual and predicted CPU utilization for the 319 minutes of training and 213 minutes of testing respectively. As discussed in previous sub-sections (LR and

NN), the training model also had a very high MAPE value (107.8). A significant improvement is however observed in the test dataset metric. Fig. 15 shows that all predicted values are positive and very close to the actual values. However, some sudden spikes result into substantial variation in values.

Table 13 - CPU Utilization Training And Test Performance Metric

Model	MAPE	RMSE	MAE	PRED(25)
Training	107.80	15.48	10.09	0.64
Testing	22.84	11.84	8.74	0.64

Table 14 - Throughput Training and Test Performance Metric

Model	MAPE	RMSE	MAE	PRED(25)
Training	78.78	4.74	2.80	0.70
Testing	22.07	3.22	2.41	0.67

5.3.2 Throughput training and test results

The Throughput's training and test performance metric results are shown in Table 14. Fig. 16 and 17 present the graphical representation of the actual and predicted Throughput values for both training and test dataset respectively. The significant difference in the training and test MAPE value is also attributed to the spikes as shown in Fig. 16. For instance, at the 256th minute, the actual Throughput value is approximately 16 requests/second while at the next minute; it drops to approximately 1.8 requests/second. The predicted value at this point is about16 requests/second. The large variation lasted for about 12 minutes before the gap was closed.

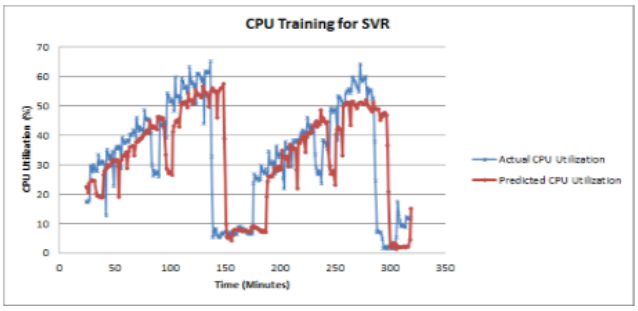

Figure 14. CPU Utilization Actual and Predicted training for SVR

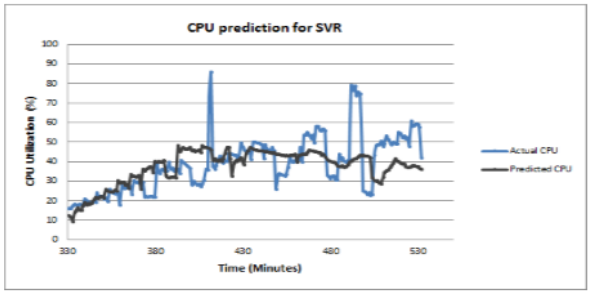

Figure 15. CPU Utilization Actual and Predicted test for SVR

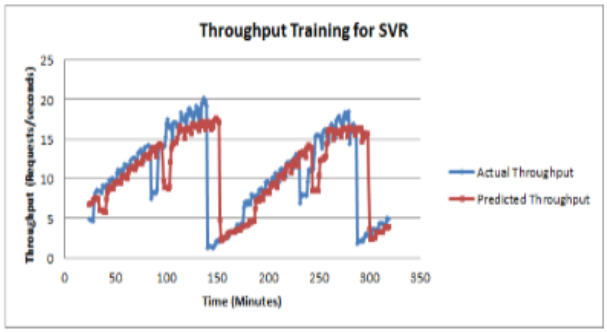

Figure. 16 Throughput Actual and Predicted Training for SVR

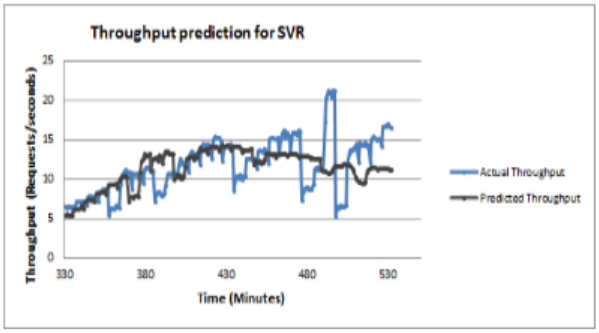

Figure. 17. Throughput Actual and Predicted Test for SVR

5.3.3 Response time training and test results

The Response time's training and test performance metric results are shown in Table 15. Additionally, Fig. 18 and 19 present the graphical representation of the actual and predicted Response time values for both training and test dataset respectively. The training and test models present similar metric values. The graph in Fig. 18 shows some variation in the predicted and actual Response time values especially between the 140th and 160th minute and also towards the end of the training dataset. In the case of Fig. 19, test

dataset, the predicted values follow the trend of the actual values. Less traffic surge is also observed in comparison to Fig. 18. Again, the range of the test dataset is between 5.5 and 12 seconds unlike the wider range for the training dataset (1 to 12 seconds). Therefore, the test metric output result is much better than that of the training.

5.4 Comparison of Prediction Models

The overall CPU utilization values range from 1.73% to 85.96%. The training dataset presented in Tables 7, 10 and 13 show that the MAPE values are above 100 percent with LR having the highest of 113.31. This abnormally high performance metric value is attributed to the fact that the traffic pattern of the workload for the experiment is random. For instance, at the 140th minute, there is a drop from 65% to about 5% CPU utilization. This drop lasted for about 40 minutes after which it surged again. The training behavior of the three models (SVR, NN and LR) for this scenario is shown in Fig. 20. It can be observed that NN shows a zigzag prediction pattern between the 132nd to about the 155th minute after which it gave a near perfect prediction of the CPU utilization. SVR and LR present a better and stable CPU utilization prediction than NN during this same interval. Isolating this randomness would significantly reduce the MAPE values; however, one of the goals of this work is to study how these learning techniques would perform in an almost realistic workload scenario. The PRED (25) metric for SVR reported the highest value of 0.64 or 64%. More importantly, the forecasting (prediction) ability of these techniques gives a more interesting trend.

Table 15 - Response Time Training and Test Performance Metric

Model	MAPE	RMSE	MAE	PRED(25)
Training	19.24	1.43	0.88	0.84
Test	9.92	1.21	0.87	0.93

Table 16 - CPU Utilization Step Prediction For MAPE

Model	9-min	10-min	11-min	12-min
SVR	22.31	22.69	22.78	22.84
NN	53.07	49.90	45.62	50.46
LR	34.43	35.14	35.92	36.19

Figure 18. Response time Actual and Predicted training for SVR

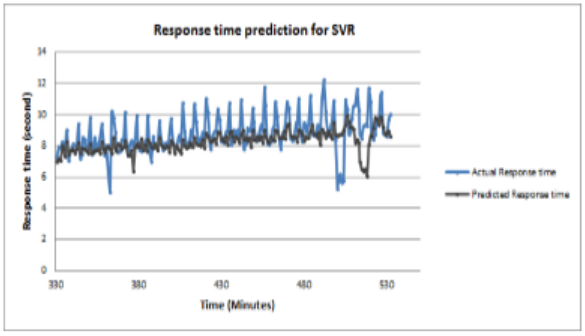

Figure 19. Response time Actual and Predicted test for SVR

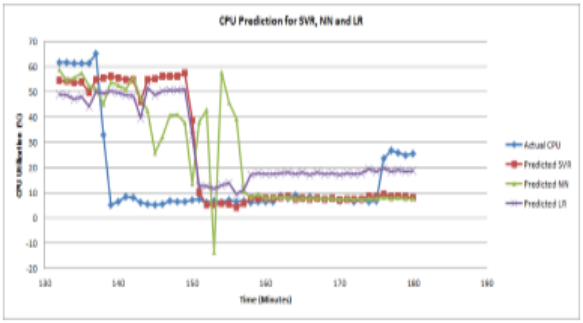

Figure 20. CPU utilization training prediction for SVR, NN and LR at selected time interval

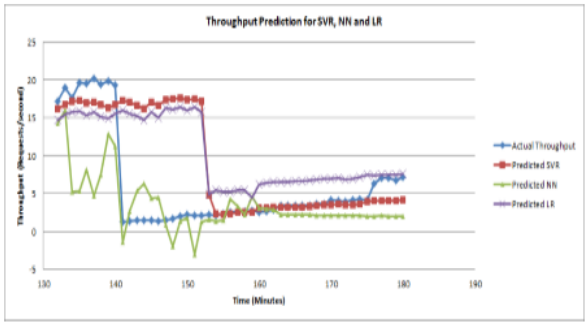

Figure 21. Throughput training prediction for SVR, NN and LR at selected time interval

SVR significantly outperforms the other two models when MAPE, RMSE and MAE performance metrics are considered. Interestingly, the test dataset was made up of short burst of high-low CPU utilization as shown in Fig. 3, 9 and 15. The generalization capability of SVR is brought to fore as it is the least susceptible to the high/low test dataset values that should result in poor forecasting output. NN and LR reports negative CPU utilization, an anomaly that exposes their weakness in random workload forecasting. The MAPE and RMSE step predictions in Tables 16 and 15 respectively show a prediction reliability of SVR and LR as opposed to NN. SVR yields the least MAPE, RMSE and MAE error. Therefore, a conclusion may be drawn in favor of SVR as the strongest and superior prediction model for CPU utilization with LR following closely.

Moving on to the business level metrics of which the Throughput model is analysed first; the throughput values had a range between 1.25 and 21 requests/second. Again, Figure 4-20 shows the selected throughput training result between the 132nd and 180th minute. The SVR and LR models could not adjust immediately to the sharp drop at the 141st minute thus accounting for the high MAPE value. SVR and LR took about 12 minutes to significantly reduce the variance between the predicted and actual throughput values though LR's prediction was not as close to the actual compared to SVR. Fig. 20 and 21 show negative prediction values for NN even though NN had the best training MAPE value of 56.46. Fig. 21 explains the reason for this as though NN had some negative predictions, the variance between predicted and actual throughput is the least. The step prediction outputs for test dataset are summarised in Tables 18 and 14. The dataset is also a mix of high and low throughput values corresponding to the random workload pattern employed in this work. SVR metrics proved to be the best by displaying a strong generalizing attribute, i.e. using the trained model to forecast unseen data (test) in a non-fitting manner. Fig. 5, 11 and 17 show the graph plots of the forecasting ability of LR, NN and SVR respectively. LR comes second behind SVR.

Table 17 - CPU Utilization Step Prediction For RMSE

Model	9-min	10-min	11-min	12-min
SVR	11.86	11.96	11.92	11.84
NN	31.56	29.69	27.64	31.08
LR	20.97	21.43	21.84	22.13

Table 18 - Throughput Step Prediction for MAPE

Model	9-min	10-min	11-min	12-min
SVR	21.38	21.62	21.80	22.07
NN	38.84	36.87	37.39	38.90
LR	25.60	25.25	25.01	24.62

Table 19 - Throughput Step Prediction For RMSE

Model	9-min	10-min	11-min	12-min
SVR	3.17	3.18	3.20	3.22
NN	5.94	5.94	5.97	6.12
LR	3.78	3.75	3.73	3.72

Finally on the business level metric wherein Response time model is analysed; the overall Response time value had a range from about 0.6 to 12 seconds. From the results obtained for the test dataset presented in Tables 9, 11 and 15, SVR performed best in comparison to LR and NN. Furthermore, the step prediction output for the test dataset shown in Tables 20 and 21 reveals the superior prediction capability of SVR. Fig. 19 shows the prediction trend for SVR. The test result shows an impressive forecasting behaviour for the test dataset for SVR. LR had a better prediction result than NN. LR and NN's test performance metric can be seen to be close to that of SVR. The reason for this is the seemingly closeness of the dataset to a linear pattern unlike the throughput and CPU utilization. Furthermore, with less variance, comes the tendency of linearity; an area LR and NN performs well. Response time model has the least range difference in comparison to CPU utilization and Throughput. In spite of this, SVR still shows superiority across board. The prediction consistency of SVR is also brought to fore in Tables 20 and 21.

Table 20 - Response Time Step Prediction for MAPE

Model	9-min	10-min	11-min	12-min
SVR	10.99	11.07	10.49	9.92
NN	15.73	16.40	17.09	17.84
LR	11.81	12.08	12.17	12.35

Table 21 - Response Time Step Prediction For RMSE

Model	9-min	10-min	11-min	12-min
SVR	1.28	1.27	1.25	1.21
NN	1.85	1.94	2.00	2.02
LR	1.37	1.39	1.39	1.39

5.5 Sensitivity analysis

In this sub-section, the validity of the experimental results is analyzed using the Little's law. Customers (user requests) arrive at the system (web server), stay for a while (receiving service) and leave. Little's law states that the average number of users in a system is equal to the departure rate of the user requests from the system multiplied by the average time each user request spends in the system. This can be summarized as [42]: $N = \lambda R$ … (14); where N = number of users in the system, λ=throughput and R = Response time

Little's law is quite general and requires few assumptions. It applies to all stable systems that may contain an arbitrary set of components such as CPU. Using Little's law, the consistency of the measurement data obtained from the experiment can be validated. Due to the large sample space (532 data points), the data from Table 2 is used as subset of the entire dataset in checking the consistency of the measurement data. From equation (14), the Number of users in the system is the Total User Request; the departure rate is the Throughput. The average time spent in the system is calculated and compared with the response time measured experimentally. During the 1st to 7th minute interval, the average throughput measured was 5.29requests/second. The average number of users during this time interval is $\frac{188}{7} \sim 27$. Using equation(14), the average time spent is $\frac{27}{5.29} = 5.1\ seconds$. The measured average response time during this period was 4.66 seconds. The percentage variance would be:

$Variance = \frac{(Average\ time\ spent - measured\ time\ spent)}{measured\ time\ spent} x\ 100$. With this example, Table 22 is completed.

Table 22 - Data Consistency Measurement

Time (minute)	1-7	56-63	154-161	350-357	490-497	498-503	504-511
Average total user requests	27 (188/7)	55 (388/7)	11 (80/7)	103 (724/7)	101 (708/7)	95 (664/7)	69 (480/7)
Average Throughput (Requests/second)	5.29	10.52	2.47	15.92	10.00	12.23	13.27
Average time spent (seconds)	5.10	5.23	4.45	6.47	10.01	7.77	5.2
Measured time spent	4.66	4.92	9.92	7.53	9.06	9.01	9.45
Time variance (%)	9.44	6.31	55.14	14.08	10.49	13.76	44.97

It can be observed that the results at the 154th-161st minute and that of the 504th-511th minute had a high percentage variance. While the period between 154-161 minute duration has an acceptable average throughput (based on numbers of user requests), the latter (504-511) has an unusually high average throughput for the number of users during the 7 minute window. For the 504-511 time interval, the logical explanation for this anomaly could be that the web server is still processing user requests from the 498-503 window when the 504-511 user request batch started sending requests. The measured response time also shows that more requests i.e. greater than the actual average of 69 users must have been requesting for service at the web server. However, the anomaly during the 154-161 window could be attributed to experimental error.

6 Conclusion

In this work, three forecasting models are built using Linear Regression (LR), Neural Network (NN) and Support Vector Regression (SVR) for a two-tier TPC-W web application. Asides from the traditional single metric prediction using CPU utilization, the monitoring metric is extended to include response time and throughput (business SLA metrics). This three-factor combination in the prediction model provides a broader view of the QoS. The user workload traffic employed is random, an approach to simulate a realistic workload pattern. After an extensive simulation lasting about 10 hours, the three machine learning techniques are trained and validated with 60% and 40% of the historical dataset respectively. The performance of SVR, LR and NN are measured using four metrics; MAPE, RMSE, MAE and PRED (25).

Overall, Support Vector Regression (SVR) model displayed superior prediction accuracy over both Neural Network (NN) and Linear Regression (LR) in a 9-12 minute window. Specifically and in terms of the MAPE performance test metric the following key observations from the simulation results are presented.

- In the CPU utilization prediction model, SVR outperformed LR and NN by 58% and 120% respectively
- For the Throughput prediction model, SVR again outperformed LR and NN by 12% and 76% respectively; and finally,
- The Response time prediction model saw SVR outperforming LR and NN by 26% and 80% respectively.
- The clear prediction superiority of SVR shows strong generalization ability in a non-linear model (random-like workload pattern). SVR can optimally map the non-linear input data to a higher dimension feature space via the Kernel function (RBF in this case), then perform linear regression

in the higher dimensional feature space [41]. The absence of the Kernel function in LR and NN makes it difficult for them to perform well in non-linear models.

Therefore, based on these experimental results SVR may be accepted as the best prediction model. Consequently, cloud clients can employ SVR to build their prediction models. Furthermore, the addition of business level SLA metrics (response time and throughput) into the prediction model paves the way for a three-factor combination decision matrix for scaling VM resources. The inclusion of response time and throughput further broadens the view of the QoS of client applications as these business level metrics may have degraded long before an application reaches its set CPU utilization threshold.

In this study, forecasting future resource usage using machine learning techniques has shown promising results. However, some areas have been identified for further research and they are presented in this section.

- This study has focused only on the web server tier. Further work to include the database tier may be worth investigating. With this inclusion, unsaturated/saturated webserver and database combination can be modeled and subsequent forecasting made using the same machine learning techniques.
- Investigating the combination of SVR and other predicting techniques that may further increase the prediction accuracy is another future direction.
- In order to further validate the forecasting strength of machine learning techniques and specifically SVR, the use of other application workloads that are not web based is another interesting investigation that may be pursued.

REFERENCES

[1] Ajila A. S. and Bankole A. A., Cloud Client Prediction Models Using Machine Learning Techniques, COMPSAC 2013 - The 37th Annual International Computer Software & Applications Conference, Kyoto, Japan July 22-26, 2013

[2] "Amazon CloudWatch Developer Guide API Version 2010-08-01", 2013. [Online]. Available: http://awsdocs.s3.amazonaws.com/AmazonCloudWatch/latest/acw-dg.pdf.

[3] "Amazon elastic compute cloud (amazon ec2)", 2013. [Online]. Available: http://aws.amazon.com/ec2.

[4] "Amazon Web services Discussion Forums", 2013. [Online]. Available: https://forums.aws.amazon.com/thread.jspa?threadID=67697

[5] Ali-Eldin, A., Tordsson, J and Elmroth E., "An adaptive hybrid elasticity controller for cloud infrastructures". IEEE Network Operations and Management Symposium (NOMS) pp 204-212. Hawaii, USA. April, 2012.

[6] Armbrust, M. et al., "A view of cloud computing". Commun. ACM. 53, 4 pp. 50–58, April, 2010.

[7] Armbrust, M. et al., Above the Clouds: A Berkeley View of Cloud Computing. 2009

[8] Bankole A., and Ajila S.A., "Cloud Client Prediction Models for Cloud Resource Provisioning in a Multitier Web Application Environment", in 7th IEEE International Symposium on Service-Oriented System Engineering (IEEESOSE 2013), San Francisco Bay, USA March 25 – 28, 2013.

[9] Bertholon, B., Varrette, S., Bouvry, P., "Certicloud: A Novel TPM-based Approach to Ensure Cloud IaaS Security". IEEE International Conference on Cloud Computing (CLOUD). pp. 121–130. 2011.

[10] Boniface, M. et al., "Platform-as-a-Service Architecture for Real-Time Quality of Service Management in Clouds". 5th International Conference on Internet and Web Applications and Services (ICIW). pp. 155–160, Barcelona, Spain. May, 2010.

[11] Borgetto, D., Maurer, M., Da-Costa, G., Pierson, J., and Brandic, I., "Energy-Efficient and SLA-Aware Management of IaaS clouds". 3rd International Conference on Future Energy Systems: Where Energy, Computing and Communication Meet (e-Energy). pp. 1–10. Madrid, Spain. May, 2012.

[12] Cain , H. W., Rajwar, R., Marden, M., and Lipasti, M., "An Architectural Evaluation of Java TPC-W" in Proceedings of the Seventh International Symposium on High- Performance Computer Architecture, Nuevo Leone, Mexico. January, 2001.

[13] Caron, E., Desprez, F., and Muresan, A., "Forecasting for Grid and Cloud Computing On-Demand Resources Based on Pattern Matching" 2nd International Conference on Cloud Computing Technology and Science (CloudCom). pp.456-463, Indianapolis, USA. November, 2010.

[14] Chieu, T.C., Mohindra, A., Karve, A.A., and Segal A., "Dynamic Scaling of Web Applications in a Virtualized Cloud Computing Environment". IEEE International Conference on e-Business Engineering ,ICEBE '09. pp 281-286. Macau, China. October, 2009.

[15] Chih-Chung, C. and Chih-Jen , L., "LIBSVM : a library for support vector machines". ACM Transactions on Intelligent Systems and Technology, 2:27:1--27:27, 2011. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm

[16] Chih-Wei, H., Chang, C.C, and Lin C., "A practical guide to support vector classification". Technical report, Department of Computer Science and Information Engineering, National Taiwan University, Taipei, 2003. [Online]. Available: http://www.csie.ntu.edu.tw/∼cjlin/libsvm/

[17] Dashevskiy, M. and Luo, Z., "Time series prediction with performance guarantee". IET Communications. Vol. 5, Issue 8, pp. 1044–1051. 2010.

[18] Dutreilh, X., Kirgizov, S., Melekhova, O., Malenfant, J., Rivierre, N., and Truck I., "Using Reinforcement Learning for Autonomic Resource Allocation in Clouds: Towards a Fully Automated Workflow" Proceedings of the 7th International Conference on Autonomic and Autonomous Systems. pp 67-74. Mestre, Italy. May, 2011.

[19] Fang, W., Lu, Z., Wu, J and Cao Z., "RPPS: A Novel Resource Prediction and Provisioning Scheme in Cloud Data Center". IEEE Ninth International Conference on Services Computing (SCC). pp.609 –616, Washington DC, USA. June, 2012.

[20] Gandhi, A., Chen, Y, Gmach, D., Arlitt, M., and Marwah, M., "Hybrid resource provisioning for minimizing data center SLA violations and power consumption". Sustainable Computing: Informatics and Systems. pp. 91–104. Orlando, Florida, USA. June, 2012.

[21] Ghanbari, H., Simmons, B., Litoiu, M., Barna, C., and Iszlai, G., "Optimal autoscaling in a IaaS cloud". Proceedings of the 9th international conference on Autonomic computing. pp 173-178. San Jose, California, USA. September, 2012.

[22] Guosheng, H., Hu, L., Li, H., Li, K., and Liu, W., "Grid Resources Prediction with Support Vector Regression and Particle Swarm Optimization," 3rd International Joint Conference on Computational Science and Optimization (CSO), vol.1, pp.417-422, China. May, 2010.

[23] Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., and Witten, I.H., "The WEKA Data Mining Software: An Update", SIGKDD Explorations, Volume 11, Issue 1. 2009.

[24] Han, R., Guo L., Ghanem, M.M., and Guo, Y., "Lightweight Resource Scaling for Cloud Applications". 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid), pp 644 –651, Ottawa Canada, May. 2012.

[25] Hasan, M.Z., Magana, E., Clemm, A., Tucker, L., and Gudreddi, S.L.D., "Integrated and autonomic cloud resource scaling". IEEE Network Operations and Management Symposium (NOMS). pp 1327-1334. Hawaii, USA. April, 2012.

[26] Hilley, D., Cloud Computing: A Taxonomy of Platform and Infrastructure-level Offerings: 2009. https://smartech.gatech.edu/handle/1853/34402. Accessed: 2013-01-24.

[27] Holehouse A., "Stanford Machine Learning". [Online]. Available: http://www.holehouse.org/mlclass/index.html

[28] Imam, M.T., Miskhat, S.F., Rahman, R.M., and Amin, M.A., "Neural network and regression based processor load prediction for efficient scaling of Grid and Cloud resources". 14th International Conference on Computer and Information Technology (ICCIT). pp 333-338, Bangladesh, India. December, 2011.

[29] Keogh, E., Chu, S., Hart, D., and Pazzani, M., "An online algorithm for segmenting time series". Proceedings of IEEE International Conference on Data Mining. pp 289-296. San Jose, California, USA. November, 2001.

[30] Khashman, A. and Nwulu, N.I., "Intelligent prediction of crude oil price using Support Vector Machines", in IEEE 9th International Symposium on Applied Machine Intelligence and Informatics (SAMI), pp.165-169, Smolenice, Slovakia. January, 2011.

[31] Kulkarni, P., "Reinforcement and Systematic Machine Learning For Decision Making", Wiley-IEEE Press, 2012.

[32] Kupferman, J., Silverman, J., Jara, P., and Browne, J., "Scaling Into the Cloud". University of California, Santa Barbara, Tech. Rep. http://cs.ucsb.edu/~jkupferman/docs/ScalingIntoTheClouds.pdf. 2009.

[33] Lim, H.C., Babu, S., and Chase, J.S., "Automated control for elastic storage". Proceedings of the 7th international conference on Autonomic computing. pp 1-10. Washington DC, USA. June, 2010.

[34] Lim, H.C., Babu, S., and Chase, J.S., "Automated control in cloud computing: challenges and opportunities". Proceedings of the 1st workshop on Automated control for datacenters and clouds. pp 13-18. Barcelona, Spain. June, 2009.

[35] Lorido-Botran, T., Alonso-Miguel, J., and Lozano, J.A., "Auto-scaling Techniques for Elastic Applications in Cloud Environments" Department of Computer Architecture and Technology, University of Basque Country, Tech. Rep. EHU-KAT-IK-09-12. September, 2012.

[36] Muppala, S., Zhou X., and Zhang, L., "Regression-based resource provisioning for session slowdown guarantee in multi-tier Internet servers". Journal of Parallel and Distributed Computing. pp. 362-375 March, 2012.

[37] Quiroz, A., Kim, H., Parashar, M., Gnanasambandam, N., and Sharma N., "Towards autonomic workload provisioning for enterprise Grids and clouds" in Grid Computing, 2009 10th IEEE/ACM International Conference. pp 50-57, Banff, Alberta, Canada. October, 2009.

[38] Richard S. S and Andrew B.B., "Reinforcement Learning an Introduction" http://www.scribd.com/doc/92878651/Reinforcement-Learning-an-Introduction-Richard-S-Sutton-Andrew-G-Barto. Accessed: 2013-01-02.

[39] Sadeka, I., Keung, J., Lee, K., and Liu, A., "Empirical prediction models for adaptive resource provisioning in the cloud", Future Generation Computer Systems, vol. 28, no. 1, pp 155 – 165, January, 2012.

[40] Sakr, G.E., Elhajj, I.H., Mitri, G., Wejinya, U.C., "Artificial intelligence for forest fire prediction" IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM), pp.1311-1316, Montreal, Canada. July, 2010.

[41] Sapankevych, N and Sankar, R., "Time Series Prediction Using Support Vector Machines: A Survey," Computational Intelligence Magazine, IEEE, vol.4, no.2, pp.24-38, May 2009.

[42] Smola, A.J and Scholkopf, B., "A Tutorial on Support Vector Regression" in Statistics and Computing vol 14, pp. 199 – 222, August, 2004.

[43] Tian, C., Wamg, Y., Qi, F., and Yin, B., "Decision model for provisioning virtual resources in Amazon EC2". 8th International Conference on Network and Service Management (CNSM), pp. 159–163, Las Vegas, USA. October, 2012.

[44] TPC, TPC-W Benchmark, Transaction Processing Performance Council (TPC), San Francisco, CA, USA, 2003.

[45] Trevor, H., Tibshirani, R., and Friedman, J., "The Elements of Statistical Learning: Data Mining, Inference, and Prediction", New York: Springer, February, 2009.

[46] Wang, S. and Summers, R.M. "Machine learning and radiology". Medical Image Analysis. Vol 16, Issue 5. pp. 933-951. 2012.

[47] Witten, I. H and Frank, E., "Data Mining Practical Machine Learning Tools and Techniques with Java Implementations", San Diego: Academic Press, 2000.

[48] Wood, T., Cherkasova, L., Ozonat, K., and Shenoy, P., "Profiling and Modeling Resource Usage of Virtualized Applications" Proceedings of the 9th ACM/IFIP/USENIX International Conference on Middleware, pp. 366-387, New York, USA. December, 2008.

