Cloud Client Prediction Models for Cloud Resource Provisioning in a Multitier Web Application Environment

By

Akindele Abisola Bankole

A thesis submitted to the Faculty of Graduate and Postdoctoral

Affairs in partial fulfillment of the requirements

for the degree of

Master of Applied Science in Electrical and

Computer Engineering

Ottawa-Carleton Institute for Electrical and Computer Engineering

Department of Systems and Computer Engineering

Faculty of Engineering

Carleton University

Ottawa, Ontario, Canada

April 2013

ABSTRACT

In order to meet Service Level Agreement (SLA) requirements, efficient scaling of Virtual Machine (VM) resources in cloud computing needs to be provisioned for before actually required due to the instantiation time required by the VM. One way to proactively provision resources is by predicting future resource demands. Most existing research on VM resource provisioning are either reactive in their approach or use only non-business level metrics such as CPU, Memory and Network utilization in their prediction model. 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 Machines (SVM), 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 subsequent thorough analysis from the experimentation carried out on Amazon Elastic Compute Cloud (EC2) show that Support Vector Machine provides the best prediction model for random-like workload traffic pattern.

ACKNOWLEDGEMENTS

Firstly, I would like to thank Professor Samuel Ajila for his unflinching support, encouragement and guidance during the thesis.

My appreciation also goes to my family for their support and encouragement throughout the period I worked on this thesis.

Finally, I thank my wife Omotayo for her help in proofreading my work on several occasions. In addition, her support and encouragement are also well appreciated.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF APPENDICES

LIST OF SYMBOLS AND ACRONYMS

API	Application Programming Interface
AR1	Auto Regression of Order 1
ARMIA	Auto Regressive Integrated Moving Average
CPU	Central Processing Unit
CRM	Customer Relationship Management
EBS	Elastic Block Store
EC2	Elastic Compute Cloud
ECU	Elastic Compute Unit
GPU	Graphics Processing Unit
GUI	Graphical User Interface
HPC	High-Performance Computing
HTML	HyperText Markup Language
I/O	Input/Output
IaaS	Infrastructure-as-a-Service
IDS	Intrusion Detection System
LR	Linear Regression
MAE	Mean Absolute Error
MAPE	Mean Absolute Percentage Error
NN	Neural Network
PaaS	Platform-as-a-Service
QoE	Quality of Experience
QoS	Quality of Service
RBE	Remote Browser Emulator
RBF	Radial Basis Function

RMSE	Root Mean Square Error
S3	Simple Storage Service
SaaS	Software-as-a-Service
SLA	Service Level Agreement
SVM	Support Vector Machine
SVR	Support Vector Regression
TPC	Transaction Processing Council
VLAN	Virtual Local Area Network
VM	Virtual Machine

CHAPTER 1: INTRODUCTION

1.1. Background

The advent of cloud computing has allowed contemporary business owners (with limited capital for example) to rent and use infrastructure resources or services needed to run their businesses in a pay-as-you-use manner. This usage has been made possible by ubiquitous network connectivity and virtualization [31]. Specifically, Armbrust, M. et al. [9] described cloud computing as both the applications delivered as services over the Internet and the hardware and systems software in the datacentre that provide those services. Merits of cloud computing over the conventional data center include [8]: appearance of infinite computing resources on demand and elimination of an up-front commitment by cloud users. Furthermore, the ability to pay for use of computing resources on a short-term basis as needed and the economies of scale due to very large data centers are additional merits.

From the foregoing, cloud computing is therefore a sharp departure from the traditional method of owning a data center that warehouses infrastructure (networks, servers and cooling systems). While the cloud offers numerous opportunities to providers and users, data security and confidentiality, availability of service and effective resource management techniques are some obstacles and challenges to the growth of cloud computing. The opportunities amidst these obstacles discussed by Ambrust, M et al [8] include: the deployment of encryption, VLANs and firewalls for data confidentiality and auditability challenge. Also, by improving VM support, making use of flash memory (which decreases I/O interference) and providing meta-scheduling abilities for high-performance computing (HPC) systems; the obstacle of performance unpredictability can be mitigated. Furthermore, the challenge of scalable storage presents an opportunity to invent a storage system that is durable, available and can scale up and down on demand. Finally, the challenge of scaling resources quickly in response to load without violating service level agreements can be managed by inventing an auto-scaler that relies on machine learning for prediction and subsequent dynamic resource scaling.

1.1.1. Cloud computing offerings

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) [41]. 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.

This thesis 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.

1.2. Motivation for the Thesis

One area that researchers have focused on is resource management. Quiroz et al [47] described four stages of data center resource management: Virtual Machine (VM) Provisioning[1], Resource Provisioning (includes mapping and scheduling requests onto distributed physical resources), Run-time Management and Workload Modeling. In this thesis, 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 (a starvation or saturation of VM resources that leads to service degradation) and over-provisioning (underutilization and subsequent waste of VM resources). Under-provisioning often leads to SLA penalty resulting into business revenue loss on the part of the cloud providers [9, 24, 25] and also a poor Quality of Experience (QoE) for the cloud client's customers (unacceptable response time for time critical applications for example). On the other hand, over-provisioning can lead to excessive energy consumption, culminating in high

[1] efficient allocation of virtual resources to application jobs as they arrive at service queues, through the creation and allocation of appropriately configured VM instances

operating cost and waste of resources [9, 24, 25]; though this has no negative impact on the client. 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 [2, 34, 38, 41, 47, 49]; specifically from between 5 and 10 minutes [2, 38], and between 5 and 15 minutes [49]. 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 [17]. 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[2] 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 [8]. In this thesis, some selected machine learning techniques' ability in forecasting future resource usage in a multi-tier web application was evaluated. The following machine learning techniques are evaluated in this thesis: Neural Network (NN), Linear Regression (LR) and Support Vector Machine (SVM). In addition, the cloud watch metrics is extended by including business level metrics such Throughput and Response time.

[2] based on CPU utilization metric only

1.3. Goal and Scope of the Thesis

The goal of this thesis is to design and develop a cloud client prediction model for cloud resource provisioning in a multitier web application environment that is capable of forecasting future resource usage and making timely VM provisioning.

The prediction capabilities of three machine learning techniques are analyzed using three benchmark workloads from a Java implementation of TPC-W [57] hosted on Amazon EC2. Three performance metrics are used to evaluate the prediction accuracy of the three machine learning techniques. An important objective of this thesis is the addition of two SLA metrics; response time and throughout to the prediction model. It is pertinent to mention that no author has considered the combination of business level metrics and coarse scale metrics such as CPU, Memory and Network utilizations.

The scope of this thesis is hinged on the IaaS model which offers developers more flexibility in their choice of programming language as opposed to PaaS providers that restrict users to their platform's programming model (like Java and Ruby on Rails) [38, 23]. Finally, the prediction model is built around the web server tier only. It is possible to extend the model to other tiers such as database or application layer.

1.4. Contributions of the Thesis

The contributions of this thesis include:

- The evaluation of the resource usage prediction capability of SVM, NN and LR using three benchmark workloads from TPC-W
- The extension of the prediction model to include business level SLA metrics thus providing wider and better scaling decision options for clients
- The comparison of the prediction capability of SVM, NN and LR under bursty and steady traffic patterns

1.5. Outline of the Thesis

The rest of the thesis is organized as follows. Chapter 2 gives a state of the art review. Chapter 3 describes the methodology adopted and the experimental model for performance analysis. Chapter 4 presents the simulation results obtained through experiments of the various machine learning techniques. Chapter 5 discusses the results while Chapter 6 concludes the thesis and also provides direction for future works.

CHAPTER 2: STATE OF THE ART REVIEW

This chapter discusses the existing work on cloud resource provisioning. Section 2.1 discusses the cloud computing service offerings. The Amazon EC2 infrastructure is also discussed in this section. Section 2.2 provides a background on machine learning and the three selected techniques (SVM, NN and LR). Section 2.3 provides a literature survey on resource provisioning. Section 2.4 provides a literature survey on resource provisioning techniques. Section 2.5 provides a literature survey on cloud resource provisioning and techniques.

2.1. Cloud Computing

This sub-section builds on the brief introductory concept of cloud computing discussed in Section 1.1.1.

2.1.1. Cloud Computing Services

Cloud computing can be classified under the services they provide which are: Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS) and Software-as-a-service (SaaS).

- **Infrastructure-as-a-Service (IaaS)**: This provides general on-demand computing resources such as virtualized servers or various forms of storage as metered resources [31]. According to Hofer and Karagiannis [32], customers buy the resources, instead of having to set up servers, software and data center space themselves, and get billed on the resources consumed. Customers then have to install and run their application on the "rented" infrastructure. Maintenance and support of the underlying infrastructure (VM) is handled by the provider. Customers on the other hand control the entire software stack. Amazon EC2 instance is a classic example of IaaS platform that provides virtualized servers of various capacities. Amazon's data centers are spread across major world Regions[3] and Zones (North America, Europe, Asia and South America). According to Amazon [2], these Zones are engineered to be insulated from failures from other Zones and provide inexpensive, low latency network

[3] Regions are made up of one or more Zones

connectivity to other Zones in the same Region. Amazon also offers storage facilities with or without the EC2 instances. Amazon Elastic Block Store (EBS) is available with EC2 instances while Amazon Simple Storage Service (Amazon S3) can stand alone (without EC2 instance) [32]. With these advances, the low level of IaaS makes it intrinsically difficult for providers like Amazon to offer automatic scalability because the semantics associated with replication are highly application dependent [57]. Extensive research using IaaS model has been carried out by several authors. These include resource provisioning [38, l, 17, 26], energy efficiency [15, 44] and security issues [13, 22].

- **Platform-as-a-Service (PaaS)**: This provides an existent managed higher-level software infrastructure for building particular classes of applications and services. The platform includes the use of underlying computing resources, typically billed similar to IaaS products, although the infrastructure is abstracted away below the platform [31]. Platform services are usually aimed at specific domains, such as the development of web applications, and are dependent on programming language [32]. Google App engine [4] falls in this category and its target programming language is Java or Python. Several authors have used this service model in their works. Boniface, M. et al [14] presented a novel PaaS architecture targeting real-time Quality of Service (QoS) guarantees for online interactive multimedia applications. Sandikkaya and Harmanci [51] focused on security of PaaS clouds. They explored and classified security problems and also proposed countermeasures for PaaS clouds.
- **Software-as-a-Service (SaaS)**: This provides specific, already-created applications as fully or partially remote services. Sometimes it is in the form of web-based applications and other times it consists of standard non-remote applications with Internet-based storage or other network interactions [31]. Customers of SaaS do not manage or control the underlying infrastructure and application platform [32]. An example of this service is Salesforce that offers customer relationship management (CRM) tools [5]. SaaS model has also found application in many research areas. Nascimento and Correia [46] presented a study on the use of anomaly-based intrusion detection system (IDS) with data

from a production environment hosting a SaaS web application of large dimensions. Their goal was to use the IDS to detect previously unknown attacks. Sun, W. et al [55] explored the configuration and customization issues and challenges to SaaS vendors. They went ahead to develop a methodology framework to help SaaS vendors plan and evaluate their capabilities and strategies for service configuration and customization.

2.1.2. Types of Cloud

The services introduced in Section 2.1.1. can be deployed on either a Public or Private cloud.

- **Public cloud:** These are cloud[4] platforms that are made available to the general public in a pay-as-you go manner. Since Public clouds are reachable via the Internet they are susceptible to potential security attacks. Users also have few or no control over the cloud environment [10]. Deployment of encryption algorithms is one way of mitigating the control and security of data in public cloud infrastructures. Small and midsized businesses with a small budget usually patronize public cloud.
- **Private cloud:** These are internal data centers of a business or other organizations, not made available to the general public and are large enough to benefit from the advantages of cloud computing [8]. Organizations using private clouds have great control on the cloud as it potentially runs inside a self-controlled perimeter network boundary [10].

2.1.3. Amazon Elastic Compute Cloud (Amazon EC2)

Amazon EC2 is a web service that provides resizable compute capacity in the cloud. It is designed to make web-scale computing easier for developers. The web service provides the ability to obtain and configure capacity with minimal friction [2]. The web service interface can be used to launch instances with a variety of operating systems, load them

[4] The data center hardware and software

with custom application environments, manage network access permissions, and run the image using as many or few systems as desired. Amazon EC2 highlights some of its service benefits which include: elasticity – the ability to increase or decrease capacity within minutes. Customers also have the choice of selecting from multiple instance types, operating systems and software packages. Furthermore, a commitment to 99.95% availability for each EC2 Region is offered in their SLA. Finally and importantly, they offer a very low per hour pay rate for the compute capacity consumed.

2.1.4. Amazon Instance Type Specifications

Amazon EC2 has a range of instance types including: Standard Instances, Micro Instances, High-Memory Instances, High-CPU Instances, Cluster Compute Instances, High Memory Cluster Instances, Cluster graphic processing unit (GPU) Instances, High I/O Instances and High Storage Instances [2]. 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 ($)[5]
M1.Small	32 or 64-bit	1 ECU[6]	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

2.1.5. Amazon EC2 Instance Options

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

[5] Price for Linux/Unix instances
[6] One EC2 Compute Unit provides the equivalent CPU capacity of a 1.0-1.2 GHz 2007 Opteron or 2007 Xeon processor [2]

- **On-Demand Instances:** On-Demand Instances allow customers to pay for compute capacity by the hour with no long term commitments or upfront payments. It allows for scalability by increasing or decreasing compute capacity depending on the demands of the application; and only paying the specified hourly rate for the instances used [2].
- **Reserved Instances:** This allows customers to make a low one-time, upfront payment for an instance, reserve it for one or three year term, and pay a significantly lower hourly rate for the instance. With reserved instances, customers are guaranteed that their instances will always be available for the operating system and Availability zone where it was purchased.
- **Spot Instances:** Spot Instances provide the ability for customers to purchase compute capacity with no upfront commitment and at hourly rates usually lower than the On-Demand rate. Spot Instances allow customers to specify the maximum hourly price that they are willing to pay to run a particular instance type. The Spot Price fluctuates based on supply and demand for instances, but customers will never pay more than the maximum price they have specified. Tian, C. et al [56] provided a detailed analysis on how the spot instances work.

2.2. Machine Learning

According to Wang and Summers [60], 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 an interdisciplinary field that has close relationships with artificial intelligence, pattern recognition, data mining and theoretical computer science. The authors classified machine learning into supervised learning, semi-supervised learning and unsupervised learning. The purpose of supervised learning is to deduce a functional relationship from training data that generalises 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 [60]. Supervised learning has been employed because its purpose matches the problem area of this thesis.

2.2.1. Supervised learning

This is the learning that takes place based on a class of examples or learning based on expert inputs [37]. It is the most common problem type in machine learning and it finds application in regression and classification problems. Regression problems which deal with predicting continuous valued output [33] would be used to forecast future values in this thesis. Typical algorithms used in solving regression problems include Liner Regression, Neural Networks and Support Vector Machines.

2.2.2. Definition of machine learning terms

Instances: These are the inputs to a machine learning scheme. They are the things to be classified, associated or clustered. Each instance is an individual, independent of the concept to be learned [61]

Attributes: Also called features are the observed variables [20]. For example, in predicting the future CPU utilization, attributes such as memory utilized, disk read/write and network in/out would be possible attributes.

Target class: This is the attribute to be predicted.

2.2.3. Linear Regression

A linear regression model assumes that the regression function $E(Y|X)$ is linear in the input $X_1, \dots, X_p$. They are simple and often provide an adequate and interpretable description of how the inputs affect the output [58]. 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 [61]. The linear regression model has the form:

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

The β_j's are the unknown parameters or coefficients, and the variables X_j are specifically,

the quantitative inputs or attributes. Typically, the parameters β are estimated from a set of training data $(x_1, y_1) \ldots (x_N, y_N)$ [61, 58]. Each $(x_{i1}, x_{i2}, \ldots, x_{ip})^{\mathrm{T}}$ is a vector of feature measurements for the ith case. The most popular estimation method is least squares, wherein we pick the coefficients $\beta = (\beta_0, \beta_1, \ldots, \beta_p)^{\mathrm{T}}$ to minimize the residual sum of squares (RSS) [58]

$$RSS(\beta) = \sum_{i=1}^{N}(y_i - f(x_i))^2$$

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

Provided there are more instances than attributes, choosing weights to minimize the sum of squared differences is really not difficult [61]. The geometry of least-squares fitting in the R^{p+1}-dimensional space occupied by the pairs (X, Y) is shown in Figure 2-1.

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 [61].
In this thesis, the Linear Regression model in the WEKA tool (discussed in Section 3.2) would be used to train the historical dataset. The forecasting accuracy would thereafter be evaluated on the held out dataset (test dataset). Accuracy in this context would be based on the performance metric summarized in Table 2.

2.2.4. Neural Networks

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. Figure 2-2 shows a feed-forward neural network.

According to Trevor, H. et al [58], 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,

$$Z_m = \sigma(\alpha_{0m} + \alpha_m^T X), m = 1, \dots, M,$$

$$T_k = \beta_{0k} + \beta_k^T Z, k = 1, \dots, K, \quad (3)$$

$$f_k(X) = g_k(T), k = 1, \dots, K,$$

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 [58]. 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

$\{\alpha_{0m}, \alpha_m; m = 1,2, \dots, M\}$ $M(p + 1)$ weights,

$\{\beta_{0k}, \beta_k; k = 1,2, \dots, K\}$ $K(M + 1)$ weights. (4)

For regression, the sum-of-squares errors is used as the error function [58]

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

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

2.2.5. Training Neural Networks

Starting Values

Starting values for weights are chosen to be random values near zero so that the model starts out nearly linear, and becomes nonlinear as the weights increase. Use of exact zero weights lead to zero derivatives and perfect symmetry, and the algorithm never moves. Conversely, starting with large weights often leads to poor solutions [58].

Overfitting

Often neural networks have too many weights and will overfit the data at the global minimum$R$. Therefore, an early stopping rule is used to avoid overfitting by training the model only for a while and stopping well before we approach the global minimum [58].

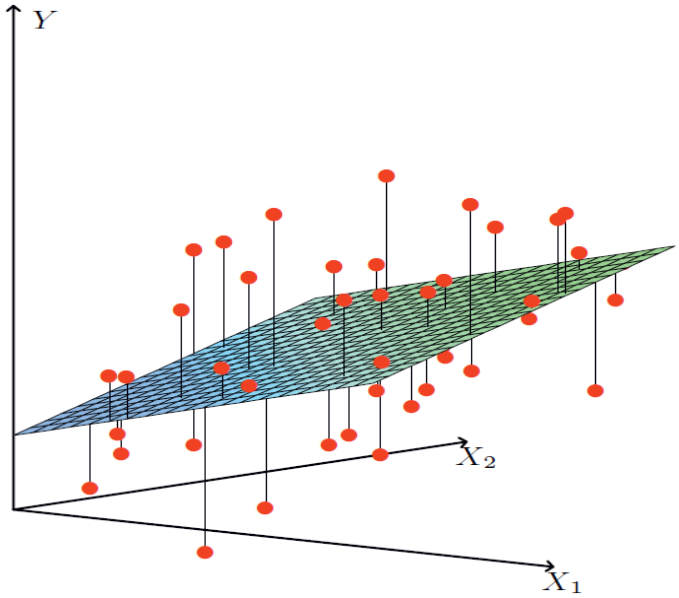

Figure 2-1: Linear least squares fitting with X $\in \boldsymbol{R^2}$ [58].

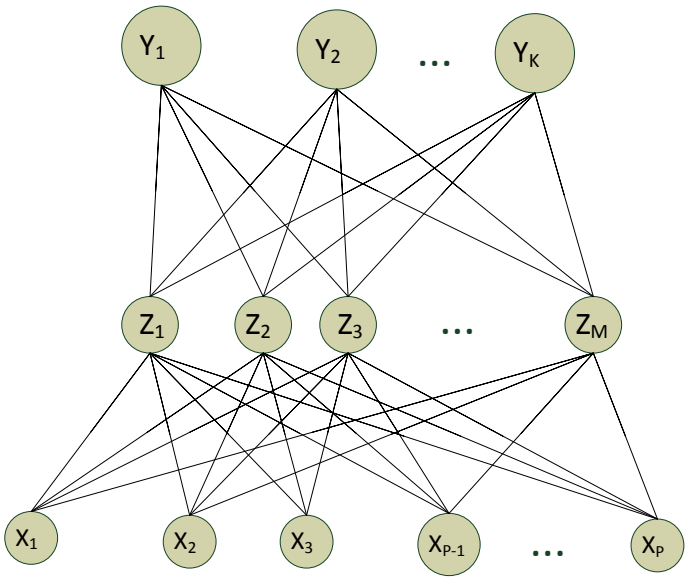

Figure 2-2: Single hidden layer, feed forward neural network [58]

2.2.6. Training Neural Networks

Starting Values

Starting values for weights are chosen to be random values near zero so that the model starts out nearly linear, and becomes nonlinear as the weights increase. Use of exact zero weights lead to zero derivatives and perfect symmetry, and the algorithm never moves. Conversely, starting with large weights often leads to poor solutions [58].

Overfitting

Often neural networks have too many weights and will overfit the data at the global minimum$R$. Therefore, an early stopping rule is used to avoid overfitting by training the model only for a while and stopping well before we approach the global minimum [58].

Scaling of the Inputs

This can have a large effect on the quality of the final solution. Standardizing all inputs to have a mean zero and standard deviation ensures all inputs are treated equally in the regularization process, and allows one to choose a meaningful range for the random starting weights [58].

Number of Hidden Units and Layers

Having too many hidden units is better than too few as with few hidden units, the model might not have enough flexibility to capture the nonlinearities in the data [58]. Typical number of hidden units is in the range of 5 to 100, with the number increasing with the number of inputs and training instances. Choice of the number of hidden layers is guided by the background knowledge and experimentation.

In this thesis, the Neural Network in the WEKA tool (discussed in Section 3.2) would be used to train the historical dataset. Specifically, the back-propagation learning algorithm (available in WEKA) would be employed. The forecasting accuracy of the trained NN model would thereafter be evaluated on the held out dataset (test dataset).

2.2.7. Support Vector Machines

According to Sakr, G.E et al [50], 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 [36]. 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 [52]) deviation from the actual obtained target for all training data with as much flatness as possible [54]. Figure 2-3 shows a regression machine constructed by the support vector algorithm

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 [27] :

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

where $f(x)$ is the target function and $<.,.>$ denotes the dot product in R^n. To achieve the flatness mentioned by [21], we minimize w 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 this may not be obtainable and thus [21] introduces slack variables γ_i, γ_i^* to handle infeasible constraints, with equation (7) leading to

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

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

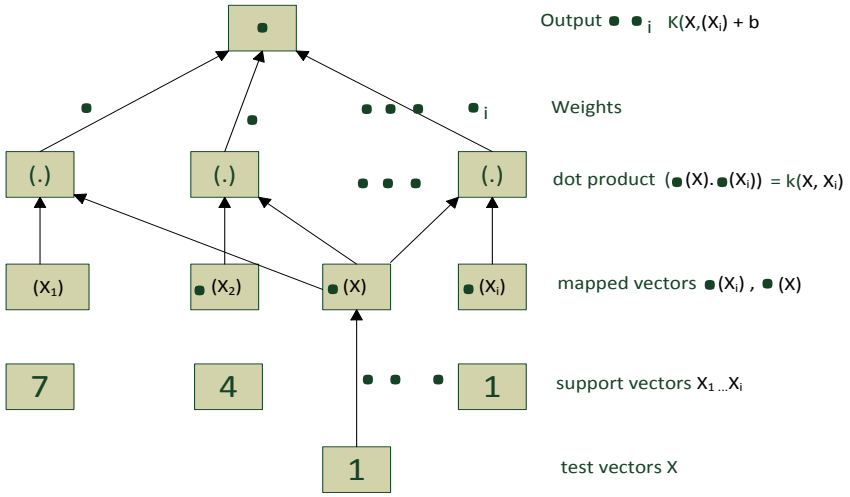

Figure 2-3: Architecture of a regression machine constructed by the support vector algorithm [54]

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

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

$$\text{b} = -\frac{1}{2} < w, (x_r + \; x_s) \tag{10}$$

$x_r and \; x_s$ are the support vectors, thus inserting (9) and (10) into (6) yields

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

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:

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

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

Four basic kernels are found in most SVM books and they are [20]:

- Linear: $K(x_i, x) = x_i^T x$
- Polynomial: $K(x_i, x) = (\gamma x_i^T x + r)^{\mathrm{d}}, \gamma > 0.$
- 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.

In this thesis, the Support Vector Machine in the WEKA tool (discussed in Section 3.2) would also be used to train the historical dataset. Specifically, the Radial Basis Function kernel (RBF); ideal for nonlinear dataset would be employed. Similarly like the other two models, the forecasting accuracy of the trained SVM model would thereafter be evaluated with the held out dataset (test dataset).

2.3. Resource Provisioning

Several research efforts to address resource provisioning have been investigated by various authors. Gandhi, A. et al [25] presented a practical and systematic approach to correctly provision server resources in data centers, such that SLA violations and energy consumption are minimized. Their hybrid method for server provisioning first included a discretization technique on historical workload dataset to identify workload demand patterns. Next, a predictive provisioning technique was used to handle predicted load at "coarse" time scales (hours). At the same time, a reactive provisioning handled any excess workload at "finer" time scales (minutes). Comparing their hybrid provisioning approach to both reactive only and predictive only provisioning, showed a decrease in SLA violations with the hybrid approach. The improvement (decrease in SLA violation) was attributed to the integration of both reactive and dynamic provisioning. Efficient resource provisioning via VM multiplexing was the focus of Meng, X. et al [43]. They proposed a joint-VM provisioning approach in which multiple VMs are consolidated and provisioned together, based on an estimate of their aggregate needs. The benefit of VM multiplexing is that when the peaks and troughs in multiple VMs are temporarily unaligned, these VMs can be consolidated and provisioned together to save capacity. Results from their work showed that joint provisioning outperformed the traditional

approach: individual VM provisioning. For instance, they reported 45% less physical machines for hosting the same number of VMs compared to the traditional approach. Quiroz et al. [47] identified VM provisioning as a problem of the end-to-end data center provisioning and hence explored a decentralized online clustering approach to detect patterns and trends and use same for virtual resources provisioning. The decentralization allowed analysis of incoming jobs from multiple distributed queues. In addition, they employed a Quadratic Response Surface Model (QRSM) to capture workload behavior and thus estimate the application service time (response time).

2.4. Resource Prediction Techniques

Various resource prediction techniques for cloud, grid and network based applications have been studied by authors. Viswanath and Valliyammai [59] proposed a prediction approach that combined Adaptive Neuro based Fuzzy Inference Systems (ANFIS) and clustering process to predict the future CPU load based on the historical data in a grid environment. The historic CPU load data was first divided into sub-clusters using the fuzzy C-means clustering. Each sub-cluster was then fed to local ANFIS prediction models. The appropriate ANFIS cluster is then used to predict the future CPU load value. Prediction with expert advice and conformal predictors were combined to provide performance guarantees on predictions for network traffic demands in the work of Dashevskiy and Luo [21]. Their goals were to construct a predictor that performs not much worse than the best algorithm from a fixed set of algorithms and also to provide valid and efficient prediction interval. Prediction with expert advice was used to achieve the first goal while conformal predictors were used for the second goal. The works of Sarkar, M. et al [53] focused on resource requirement prediction in distributed systems like grids and clouds. They presented a feedback-based job modeling scheme based on clone detection technique. In this technique, the execution data for each job running in the environment were stored in an execution history. Newly submitted jobs were then analyzed to find its clones (match) from the execution history and based on the data stored in the execution history; the resource requirement of the new job was predicted. Their clone detection technique was metric based comparison technique. Resource prediction once a clone level was detected was carried out using linear and multi-linear

regression. Kupferman, J. et al [38] investigated and evaluated both static and dynamic resource provisioning in three different traffic patterns – Weekly or Standard Oscillations, Large Spike and Random by using a scoring algorithm based on availability and cost. The authors established that dynamic resource provisioning outperforms the expensive static provisioning[7] by about 93% in cost reduction. Their resource prediction technique was based on historical data and the employment of linear regression and auto regression of order 1.

2.5. Cloud Resource Provisioning and Techniques

Several authors have worked in the area of resource provisioning using different approaches. This section presents some of the related techniques in two broad categories: predictive and reactive cloud provisioning techniques.

2.5.1. Threshold based provisioning

Han et al [29] 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 [40]. This constraint makes their work restrictive and not generalistic. The work by Hasan, M.Z. et al [30] 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

[7] Static provisioning determines the minimum number of machines required for 100% availability at peak periods of a given trial and then run the machines for the duration of the trial

storage) in making scaling decisions. While their 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. A dynamic scaling algorithm for automated provisioning of VM resources based on threshold number of active sessions in a web application was introduced by Chieu, T.C. et al [18]. However, since a session is a sequence of individual requests of different types, it has been reported that session length is dynamic and unknown at the time of session origination [45]. Muppala, S. et al [45] further went on to state that because of the relative unpredictability of the session length, a metric which is independent of session length is favorable and mandatory for performance guarantee of session based Internet services. It should be noted that the work of Chieu, T.C. et al [18] did not consider the VM boot up time in their active session scaling indicator. Finally, there were no actual experimentation results for their algorithm.

2.5.2. Control Theory based provisioning

According to Lorido-Botran, T. et al [41], 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 [7]. 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. Their 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 in their work. Lim, H.C. et al [40] proposed that cloud customers should be empowered to operate their own dynamic controllers, outside or as extensions to the cloud platform itself. Their 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. Though their proportional thresholding policy performed better than the static thresholding policy, their control approach was reactive thus penalties would be incurred when provisioning a new VM to handle more client requests. Finally, the control target was a single metric – CPU. They did not consider SLA related metrics such as response time and throughput in their control target. Consolidating on their proportional thresholding policy, Lim, H.C. et al [39] applied this policy to an elastic storage. This time around, they addressed the actuator lag stemming from delay in provisioning new instances and also redistributing stored data.

2.5.3. Reinforcement Learning

Reinforcement learning (RL) is another type of automatic decision-making approach that can be applied to VM provisioning [41]. 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 [23]. Dutreilh, X. et al [23] 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 [48]) the authors introduced a convergence speedup phase at regular intervals to hasten the learning process.

2.5.4. Time series analysis provisioning

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 [41]:

$$X = x_t, x_{t-1}, x_{t-2}, \dots, x_{t-w+1} \quad (13)$$

Several authors have used time series analysis for dynamic VM provisioning; for example, [17] presented a resource usage prediction algorithm that used a set of historical data to identify similar usage patterns to predict future usage. The string matching concept which is: given a text string $T = t_0 t_1 \ldots t_n$ and a pattern $P = p_0 p_1 \ldots p_m$ find a substring of consecutive characters from T called $T_{i,j}$ that has the smallest edit distance as possible was used by the authors. Specifically, the past cloud client usage traces is analogous to the text string T; the present usage pattern consisting of the last usage measure of the cloud client is similar to the pattern P while the future usage pattern will be interpolated by using a weighted interpolation and will have an approximate value that will follow after the present pattern. Experiments with their modified string matching algorithm (Knuth-Morris-Pratt) on three different workload traces showed impressive prediction capabilities of 0.9 to 4.8% prediction error. The prediction look ahead was for 100 seconds. Kupferman, J. et al [38] compared three platform-agnostic algorithms for scaling resources dynamically: one developed by RightScale, and two others that predict system loads based on Linear Regression (LR) and AutoRegression of order 1 (AR1). The algorithms were evaluated based on a scoring metric which gauges how well a given algorithm performed based on availability and cost. While the RightScale used a voting mechanism to decide when to scale VM instances up or down, their AR1 model estimates future workload based on a finite history window. The AR1 was able to predict the average load over the next 5 minutes. As for LR, the linear least-squares technique was used to find the polynomial function that is closest to a set of points; subsequently, the algorithm was used to predict the amount of requests per second that will occur in the future (100 seconds). The algorithm then compares the predicted CPU utilization with the current computing resources and scales up or down as necessary. Evaluating each algorithm with three different traffic patterns: Standard Oscillation, Large Spike and Random traces; AR1 gave best performance[8] for standard and large spike traffic pattern while RightScale performed best for random traffic. A cloud resource prediction and provisioning scheme (RPPS) that automatically predicts future demand and performs a proactive resource provisioning for cloud applications was presented by Fang, W. et al [24]. Their scheme employed a load prediction algorithm based on the Autoregressive

[8] The authors developed a scoring algorithm that consists of the number of total and dropped requests and the cost of provisioning the machines.

Integrated Moving Average (ARIMA) model. The resource usage (CPU utilization) time series were fed into the load predictor model to predict short-term resource demands. Based on this model, predictive provisioning of VMs was achieved for two situations: Normal and Sudden load spike workloads. For the normal workload that could be accurately predicted by their model, fine-grained VM-level resource scaling technique[9] was employed. For the sudden load spikes the coarse-grained capacity scaling approach was used i.e. dynamically adding new VM instances. It suffices to state that they considered VM boot up lag in their coarse-grained capacity scaling. Experimental results of their model showed that the prediction model made less than 10% underestimation or overestimation with occasional higher overestimation errors of 20% resulting from peak loads. The authors proposed handling peak or spike loads with pre-reserved resources.

The work presented by Sadeka, I. et al [49] 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 [35]. Performance evaluation[10] of the two learning algorithms showed that Neural Network demonstrated enhanced accurate prediction capability compared to Linear Regression.

[9] This involves adjusting the VM CPU and memory allocation from the hypervisor-level actuators
[10] MAPE, MAE, RMSE and PRED (25)

CHAPTER 3: DESIGN OF EXPERIMENT

This chapter focuses on the experimental design of the client prediction models from three machine learning techniques. Section 3.1 describes the system architecture setup for the experiment. Section 3.2 describes the open source WEKA tool used to train historical dataset. Section 3.3 describes the step by step experimental setup and it includes the parameter value used to train the three machine learning models.

3.1. System Architecture

As mentioned in Section 1.3, the cloud client prediction model for cloud resource provisioning in a multitier web application environment has the following components in the overall architecture (Figure 3-1):

- 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

3.2. WEKA

WEKA is used to train and test the three machine learning techniques. Waikato Environment for Knowledge Analysis (WEKA) is a data mining software in Java that has a collection of several machine learning algorithms for data mining tasks including SVM, NN and LR [28]. Experiments can be carried out using either the command-line or graphical user interface (GUI) option. In this thesis the GUI option is used. Specifically, the Explorer application 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 SVM, NN and LR.

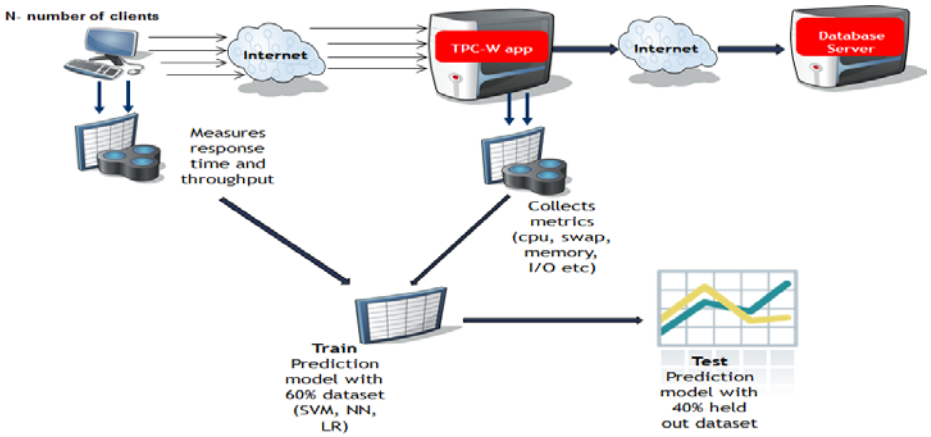

Figure 3-1: Architecture of the System

3.3. Experimental Setup

In this section, the step by step details of experimentation are discussed.

3.3.1. Feature Selection

Usually, prediction models are based on a continuous observation of a number of specific features [50]. The following initial features are selected for the three target values (CPU utilization, response time and throughput) [1]:

i. DiskReadOps: This metric identifies the rate at which an application reads a disk.
ii. DiskWriteOps: This metric identifies the rate at which an application writes to a hard disk.
iii. DiskReadBytes: This metric is used to determine the volume of the data the application reads from the hard disk of the instance.
iv. DiskWriteBytes: This metric is used to determine the volume of the data the application writes onto the hard disk of the instance.

v. NetworkIn: This metric identifies the volume of incoming network traffic to an application on a single instance.

vi. NetworkOut: This metric identifies the volume of outgoing network traffic to an application on a single instance.

vii. Memory Utilized: This metric collects and sends the memory utilization excluding cache memory and buffers.

viii. Memory Available: This metric collects and sends available memory used by the operating system and the application.

ix. Swap Utilized: The amount of swap spaced utilized.

3.3.2. Data collection using TPC-W benchmark

TPC-W has been used by several authors [62, 16] for resource provisioning and capacity planning [49]. Similar to Sadeka et al. [49], a Java implementation of TPC-W that emulates an online bookshop is used. It is deployed on a-two-tier architecture as depicted in Figure. 3-1. 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 is modeled after an online bookstore and includes complex application logic, a significant web serving component including both static and dynamic web pages, and transaction processing connectivity to an online relational database containing product inventory [16]. It has 14 web interactions characteristics of which 6 belong to the Browsing category and the other 8 to the Ordering category as shown in Table A.2 of Appendix A. The characteristics of these web interactions can be seen in Table A.1 (Appendix A). 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 Figure 3-1 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. Appendix B shows the Browsing mix Java batch script that is called every 7 minutes. Amazon EC2 has a web service that enables monitoring, managing and publishing of various metrics [1]. The traditional ***Top*** command in Linux is not used as this command give metrics for the underlying host and not the actual instance [3] . Feature readings (defined in Section 3.3.1) 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[11] for metrics [1] and for this thesis the EC2 namespace is used. Appendix B shows the full Java batch script for CPU utilization. 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.

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**

[11] Examples of other containers include; Amazon Elastic Block Store (EBS), Amazon Relational Database (RDS) [z15]

3.3.3. 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 [50]. The Weka tool [28] 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: DiskReadOps, DiskWriteOps, DiskReadBytes, DiskWriteBytes, Memory Utilized, Memory Available and Swap Utilized. Figure 3-

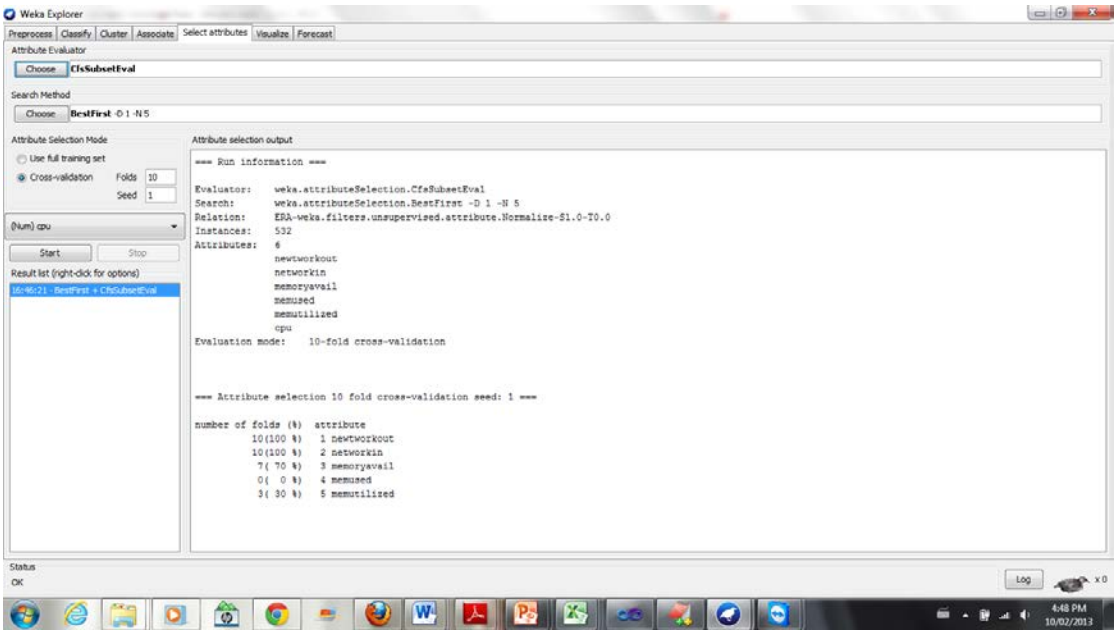

Figure 3-2: Attribute selection option to rank attribute in order of relevance [28]

3.3.4. 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 [20].

3.3.5. Training of Dataset

As discussed earlier, the goal of this thesis 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.

Model 1 - CPU utilization

- **Neural Network**: Using the Weka tool, the model is trained with the following parameters: learning rate $\rho = 0.38$, 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. 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 [58]. Varying the value 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 [36]. 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 [20]. Before training, the Grid Parameter Search for Regression with cross validation is used (v-fold cross validation) [19] (Figure 3 - 3) to estimate the C and γ. Cross-validation is a technique used to avoid the over fitting problem [q, z6]. 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 with guidance from various author's work [q, z6]. 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.

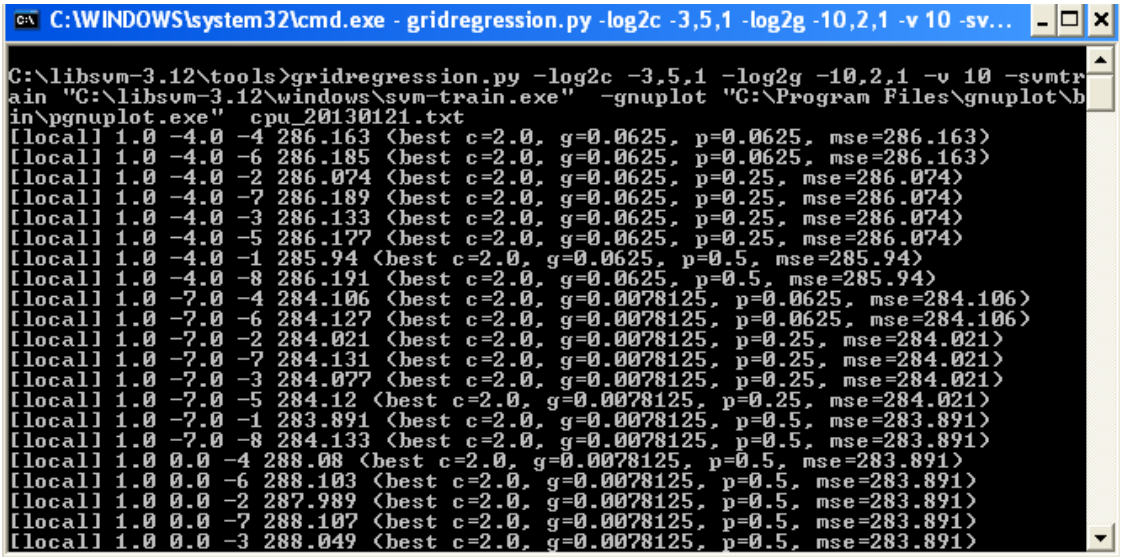

Figure 3-4: Parameter search for SVR [19]

Model 2 – Response time and Throughput

The business SLA metrics were approached in a similar way as Model 1. 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. Similarly, 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 4, 5 and 6 list the final parameters used for training the SVM, NN and LR model.

Table 3: Performance metrics and their calculations

Metric	**Calculation**
MAPE[12]	$\frac{1}{n}\sum_{i=1}^{n}\frac{\lvert a_i-p_i\rvert}{a_i}$ where a_i and p_i are the actual and predicted values respectively
RMSE[13]	$\sqrt{\frac{\sum_{i=1}^{n}(a_i-p_i)^2}{n}}$

[12] Mean Absolute Percentage Error
[13] Root Mean Square Error

MAE[14]	$\frac{1}{n}\sum_{i=1}^{n}\lvert p_i - a_i \rvert$
PRED 25	No. of observations with relative error ≤ 25% / No. of observation

Table 4: Final parameters of the SVM 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

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

3.3.6. Validation (Test) of Dataset

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) was used as this gave the optimal prediction output for the models. A 12 minute prediction interval to test our prediction model is adopted. This is based on reports from previous works [2, 38] regarding VM boot up time and motivation from the work of [49]. The prediction trend at the 9th, 10th, 11th and 12th minute is included to check for consistency and reliability in the prediction models of SVR, NN and LR.

[14] Mean Absolute Error

CHAPTER 4: SIMULATION RESULTS AND ANALYSIS

This chapter presents the results of the various experimental simulations for determining the prediction capability of the three machine learning techniques: SVM, NN and LR. The objective of this section is to present and 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. The parameters used to obtain these results have already been presented in Section 3. Similarly, the performance metrics have also been defined in the previous section (Table 3).

4.1. Linear Regression Models

Results from CPU Utilization, Throughput and Response time for both training and test dataset are presented.

4.1.1. CPU utilization training and test results

Table 7 shows the training and testing performance metric results for the LR. Furthermore, Figures 4-1 and 4-2 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. Figures 4-1 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 4-2.

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

4.1.2. Throughput training and test results

The Throughput's training and test performance metric results are shown in Table 8. In addition, Figures 4-3 and 4-4 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 training metric results are better than test results. In fact, from the 397th minute of the test dataset, a continuous and erroneous negative prediction is forecasted. This significantly contributes to the dismal test results as the difference between the actual and predicted value results into a higher value as opposed to the expected lower difference that occurs if both values are positive. Figure 4-4 shows the downward spiral in Throughput forecast. This again confirms a weak training and forecasting ability of LR in a non-linear traffic pattern.

Table 8: Throughput training and test performance metric

Model	MAPE	RMSE	MAE	PRED(25)
Training	75.25	4.45	3.22	0.57
Test	156.08	23.39	19.44	0.04

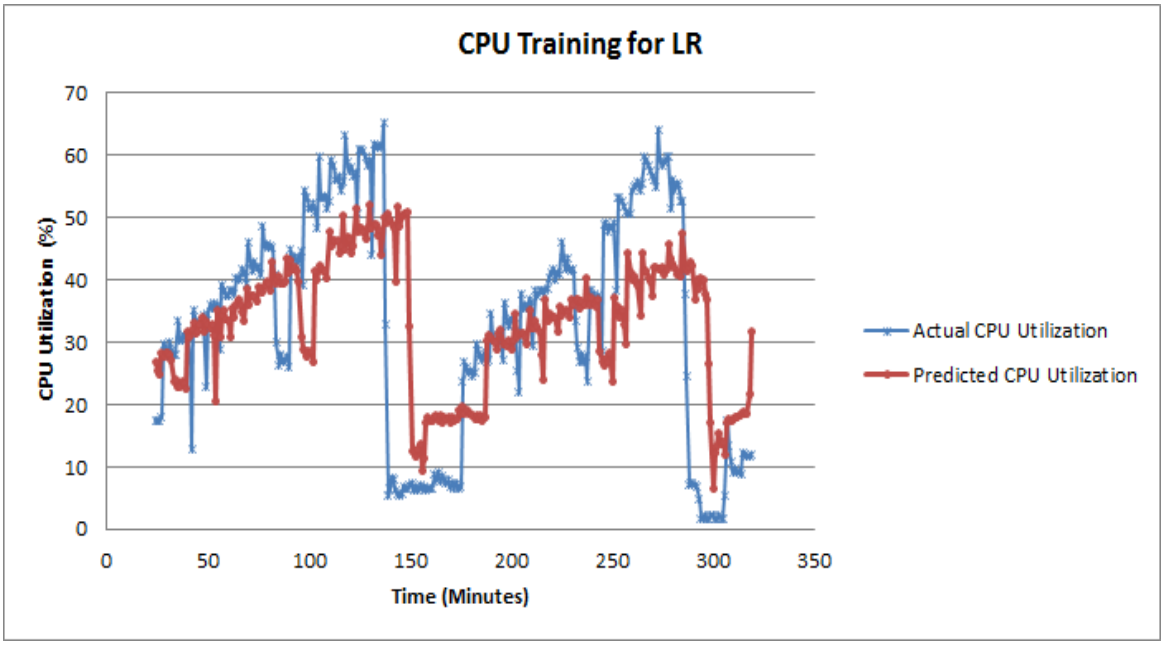

Figure 4-1: CPU Utilization Actual and Predicted training output using LR

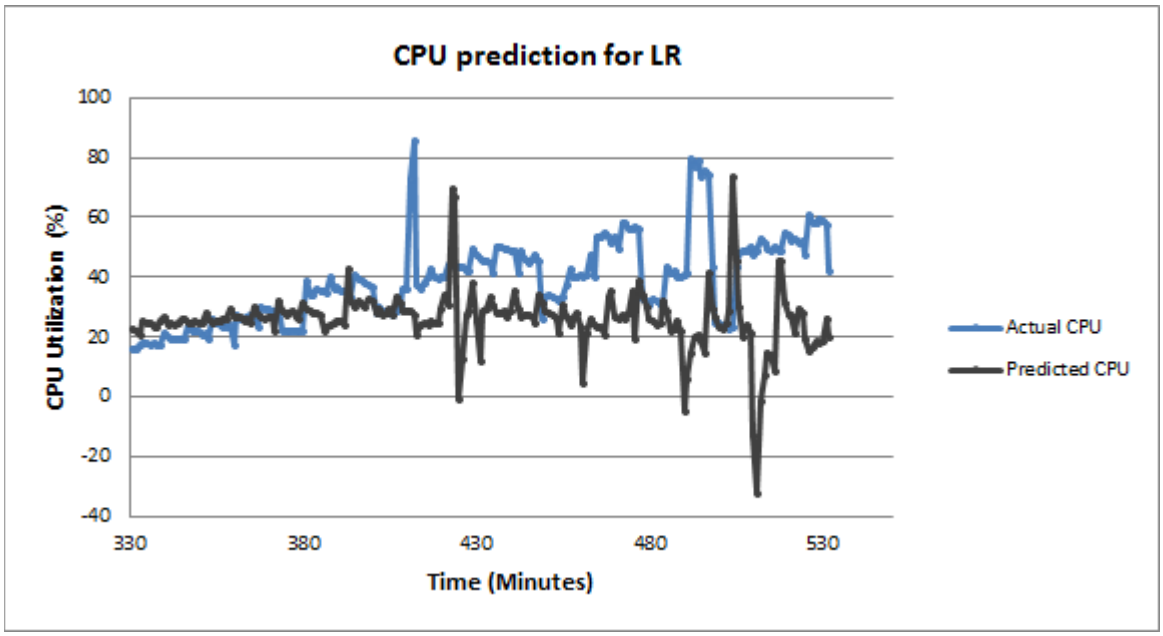

Figure 4-2: CPU Utilization's Actual and Predicted test output using LR

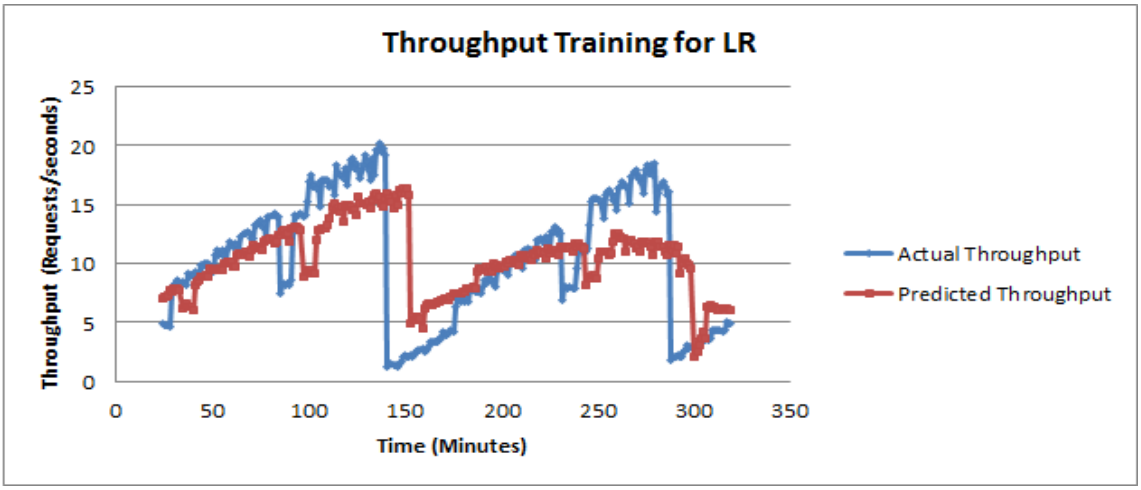

Figure 4-3: Throughput's Actual and Predicted training using LR

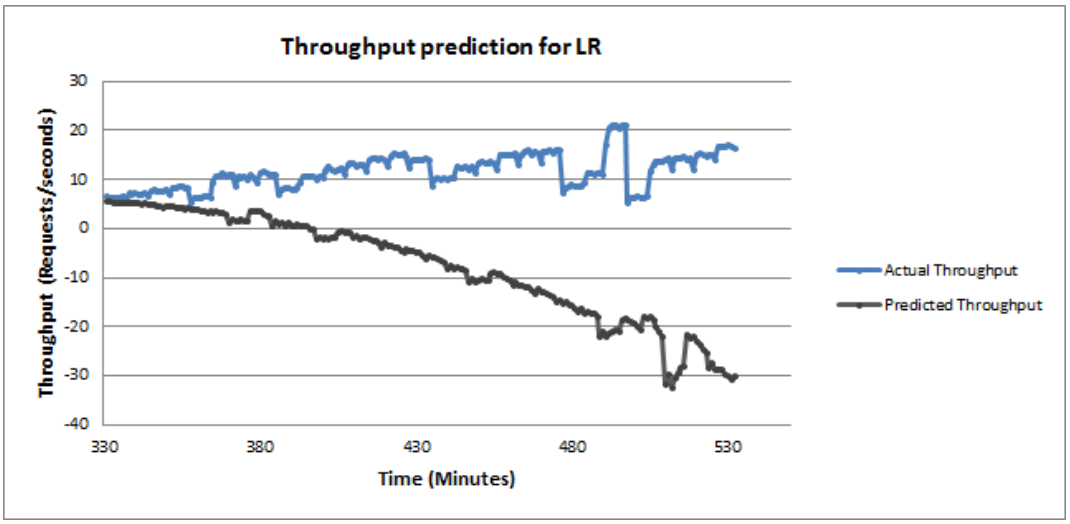

Figure 4-4: Throughput Actual and Predicted test output using LR

4.1.3. Response time training and test results

The Response time's training and test performance metric results are shown in Table 9. Furthermore, Figures 4-5 and 4-6 present the graphical representation of the actual and predicted Response time values for both training and testing dataset respectively. While the training results are of high accuracy, the test performance metric fails to provide an accurate and reliable forecast. The test dataset prediction follows a similar pattern as that of Throughput's test; negative prediction values. Figure 4-6 shows the divergence in predicted and actual values. Though the training metrics are impressive, the validity of the learning algorithm (LR in this case) is hinged on the test dataset performance.

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	87.97	8.78	7.88	0.02

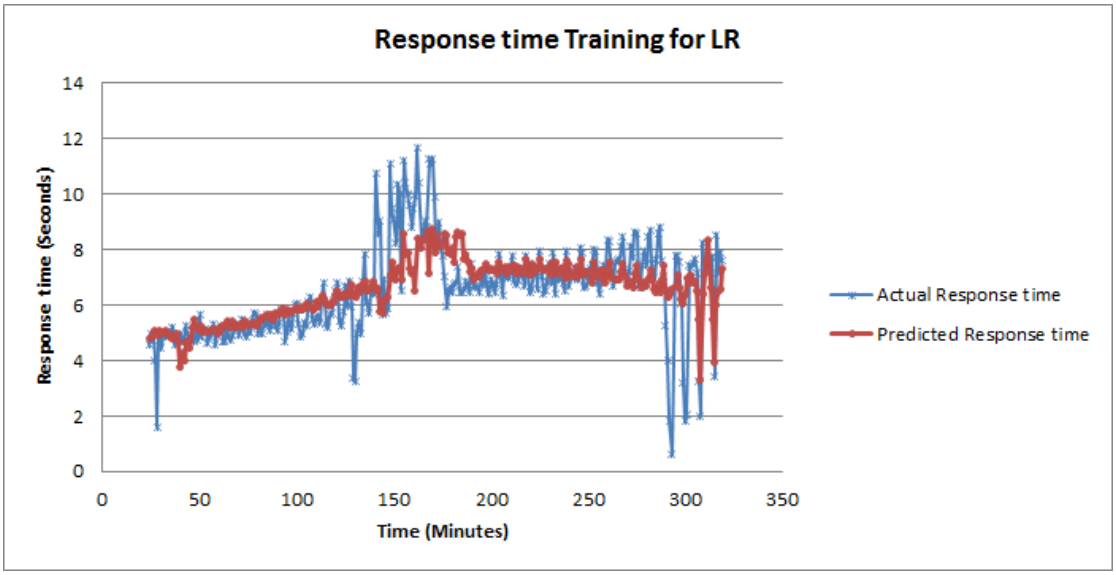

Figure 4-5: Response time Actual and Predicted training output using LR

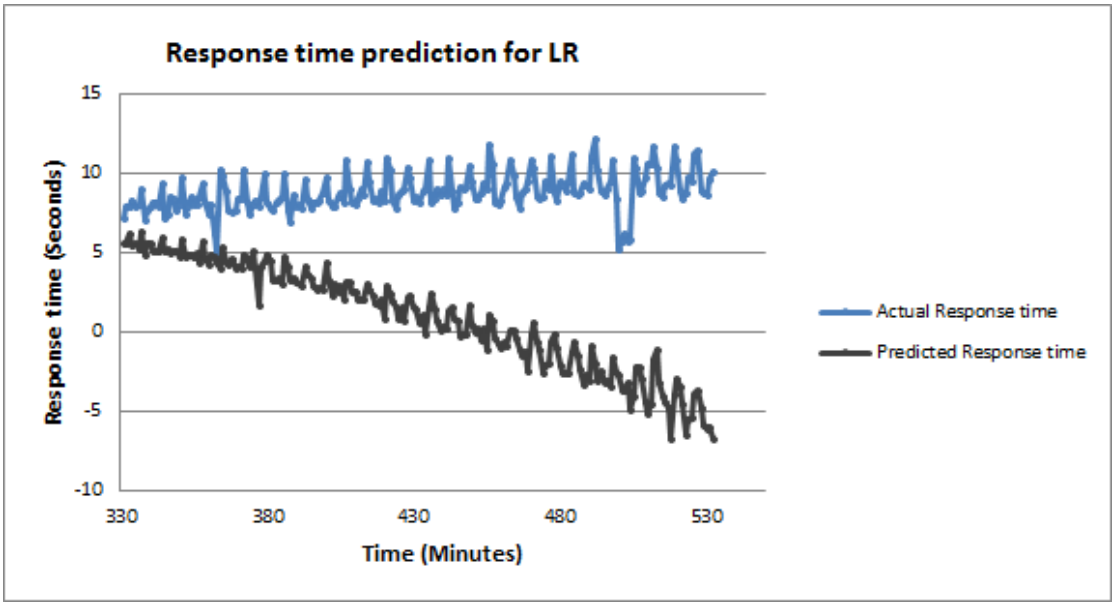

Figure 4-6: Response time Actual and Predicted test output using LR

4.2. Neural Network Models

In the sub-sections that follow, the CPU utilization, Throughput and Response time training and testing dataset results are presented.

4.2.1. CPU Utilization training and test results

The CPU utilization training and test performance metric results for NN model is shown in Table 10. Figures 4-7 and 4-8 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 section 4.1.1. It is also observed that the test dataset has some series of negative values in its prediction as shown in Figure 4-8. The number of negative predicted values which is more than that of LR contributed to the poorer test metric values.

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

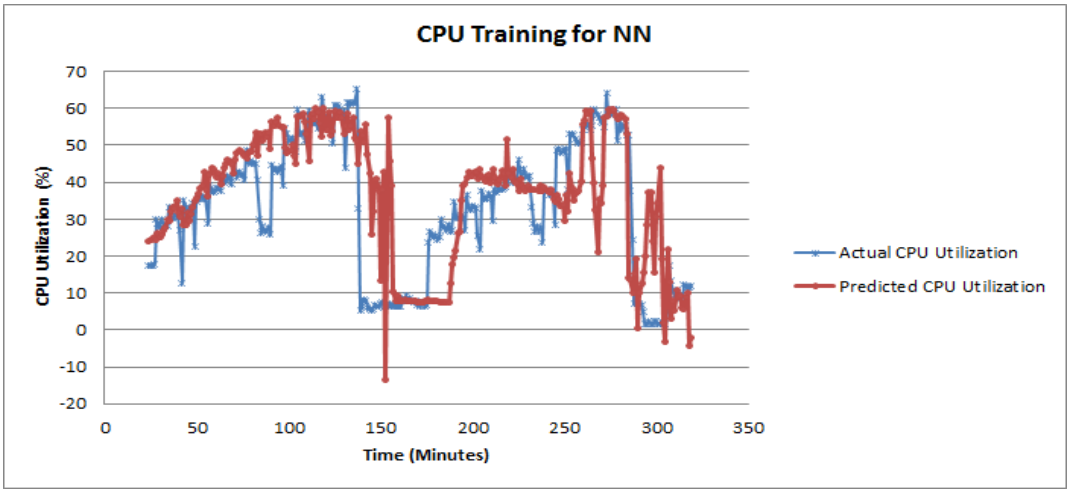

Figure 4-7: CPU Utilization Actual and Predicted training output using NN

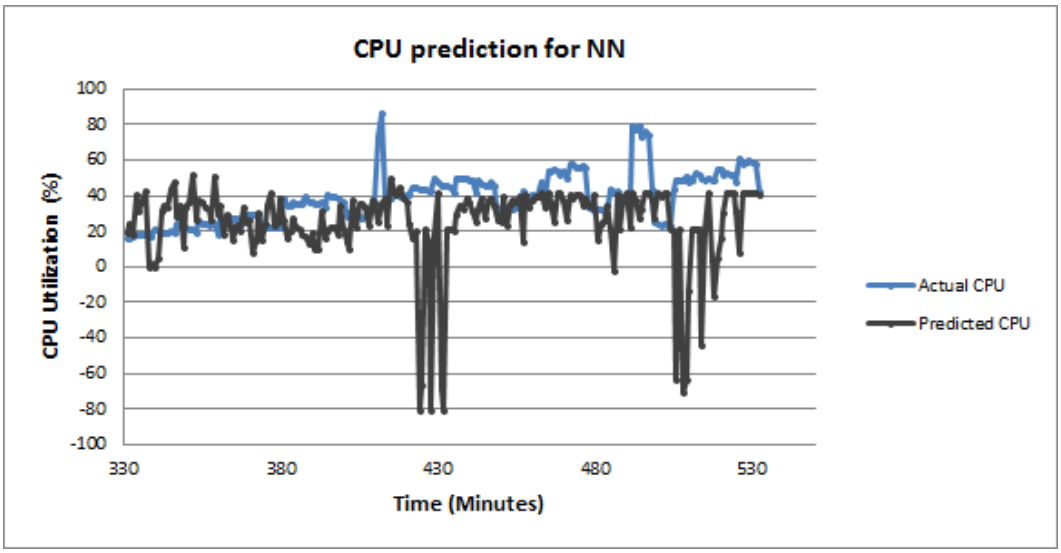

Figure 4-8: CPU Utilization Actual and Predicted test output using NN

4.2.2. Throughput training and test results

The Throughput's training and test performance metric results are shown in Table 11. In addition, Figures 4-9 and 4-10 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 (Figure 4-8). However, negative prediction is absent for the test dataset (Figure 4-9).

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

4.2.3. Response time training and test results

The Response time's training and test performance metric results are shown in Table 12. Furthermore, Figures 4-11 and 4-12 present the graphical representation of the actual and predicted Response time values for both training and test dataset respectively. Both training and test model's metric values are quite similar. Comparing this result (Response time) with the previous two, it is observed that both training and test dataset values are positive. However, 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 4-11.

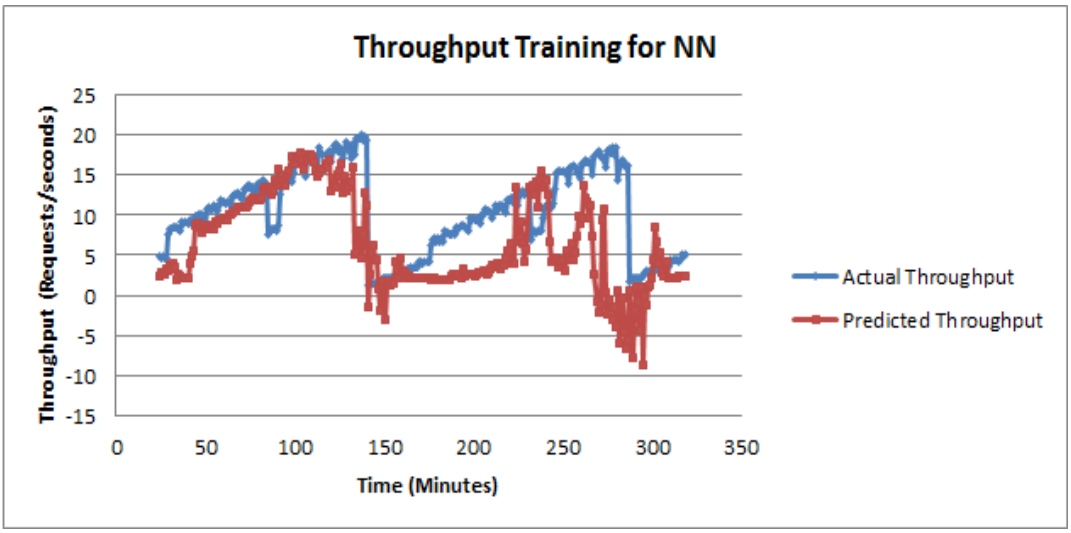

Figure 4-9: Throughput Actual and Predicted training using NN

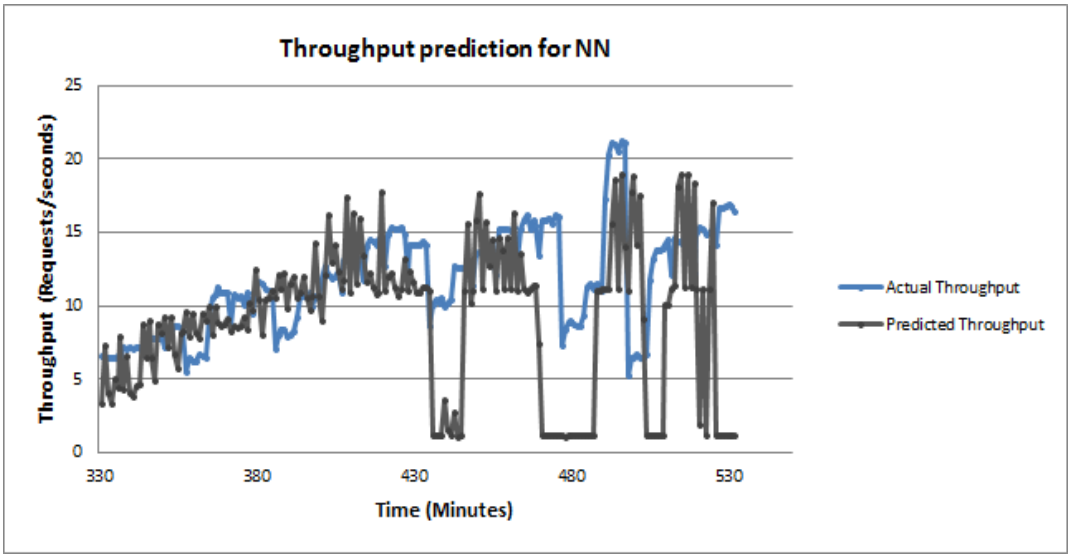

Figure 4-10: Throughput Actual and Predicted test output using NN

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	35.15	3.80	2.94	0.48

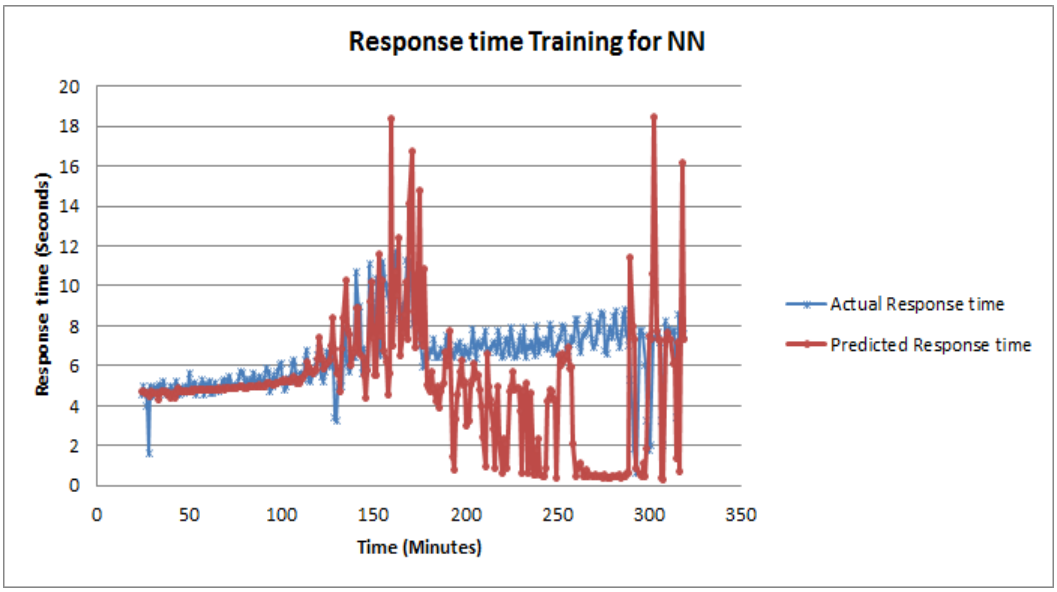

Figure 4-11: Response time Actual and Predicted training output using NN

4.3. 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.

4.3.1. CPU Utilization training and test results

The CPU utilization training and test performance metric results for SVR model is shown in Table 13. Figures 4-13 and 4-14 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 sections, the training model also had a very high MAPE value (107.8). A significant improvement is however observed in the test dataset metric. Figure 4-14 shows that all predicted values are positive and quite 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

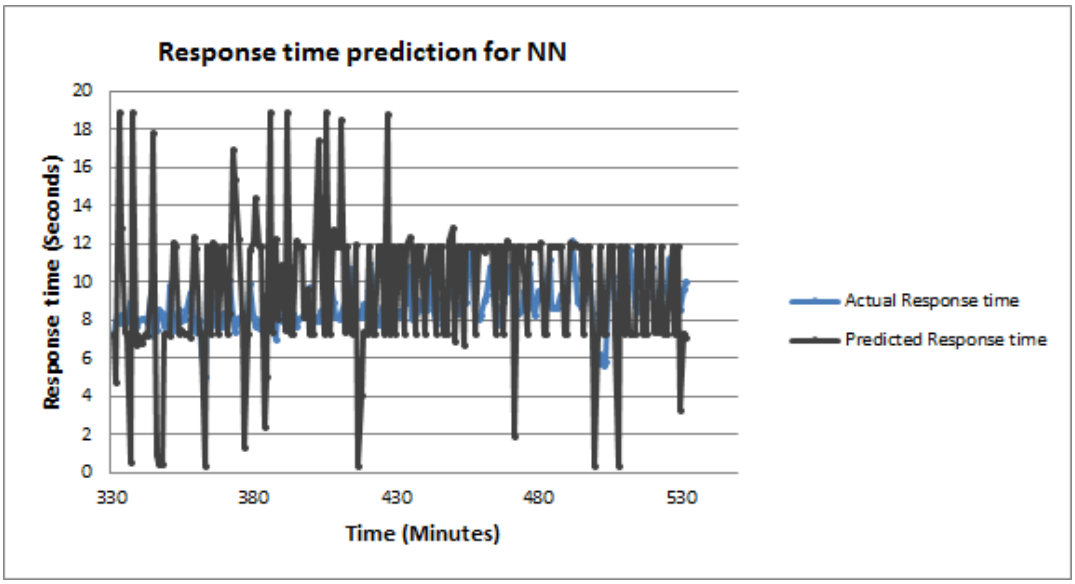

Figure 4-12: Response time Actual and Predicted test output using NN

4.3.2. Throughput training and test results

The Throughput's training and test performance metric results are shown in Table 14. Figures 4-15 and 4-16 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 Figure 4-15. 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 about 16 requests/second. The large variation lasted for about 12 minutes before the gap was closed.

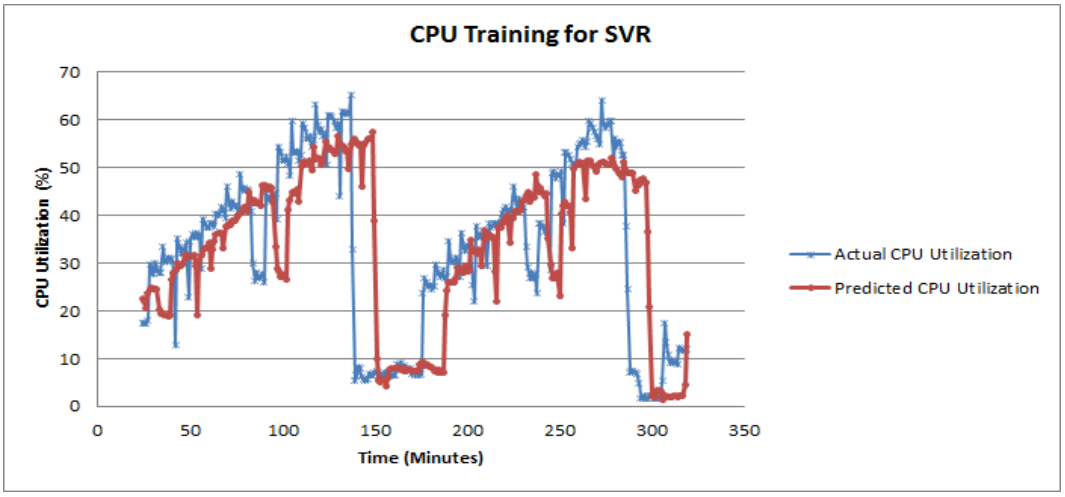

Figure 4-13: CPU Utilization Actual and Predicted training output using SVR

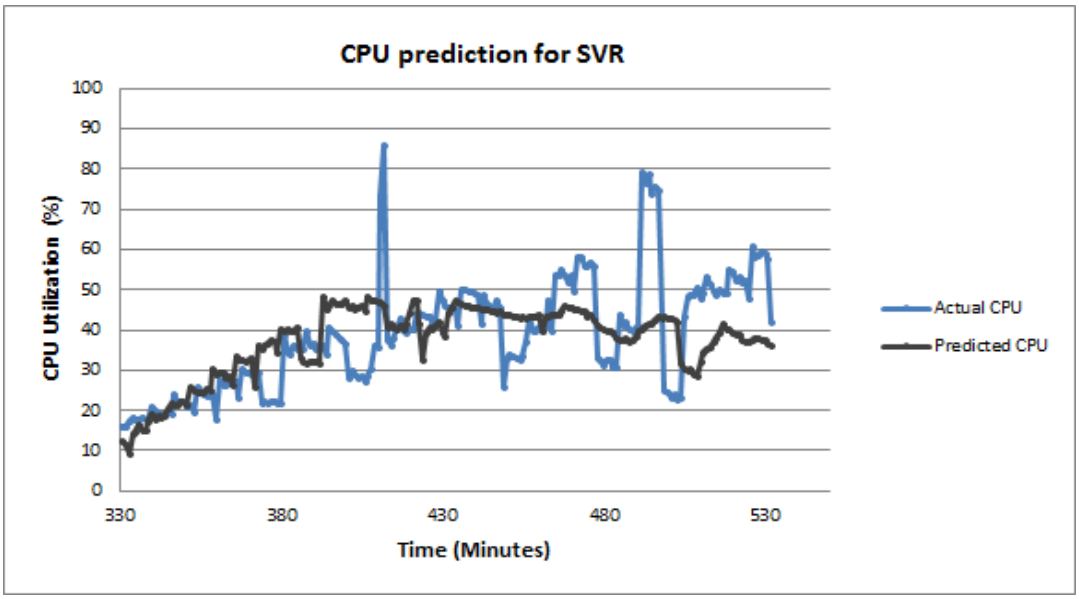

Figure 4-14: CPU Utilization Actual and Predicted test output using SVR

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

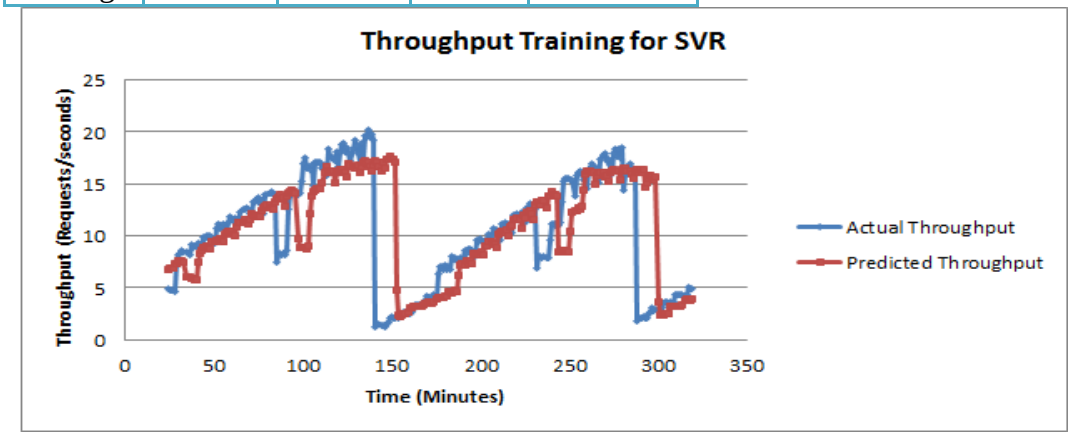

Figure 4-15: Throughput Actual and Predicted Training using SVR

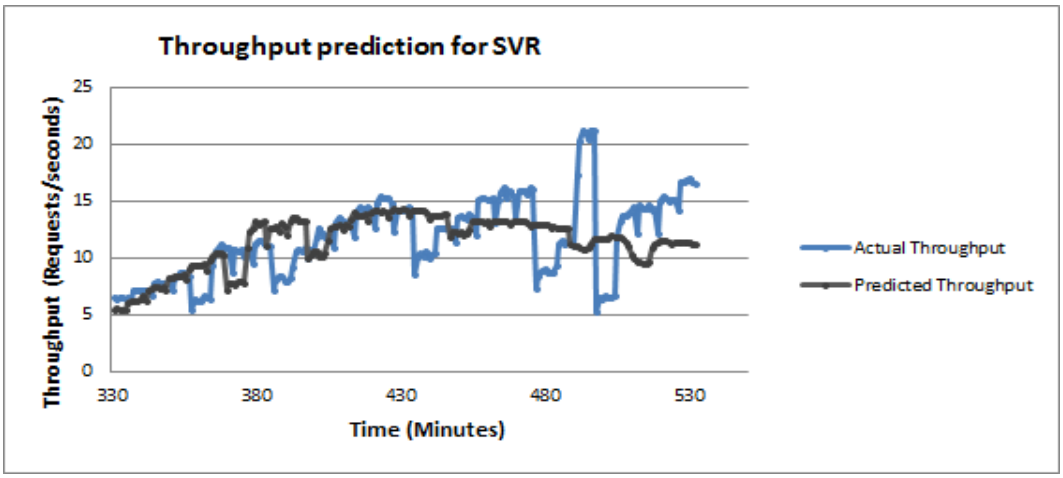

Figure 4-16: Throughput Actual and Predicted Test output using SVR

4.3.3. Response time training and test results

The Response time's training and test performance metric results are shown in Table 15. Additionally, Figures 4-17 and 4-18 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 Figure 4-17 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. Comparing the graph with that of Figure 4-18, test dataset, a diverging trend between the actual and predicted occurs. Therefore, the training metric output result is better (overall) than that of test. The reason for the higher test metric values (apart from MAPE) is; RMSE and MAE are more sensitive to the occasional large error (especially for RMSE in the squaring process) than MAPE [6].

Table 15: Response time training and test performance metric

Model	MAPE	RMSE	MAE	PRED(25)
Training	18.96	1.55	0.85	0.88
Test	14.30	1.74	1.33	0.83

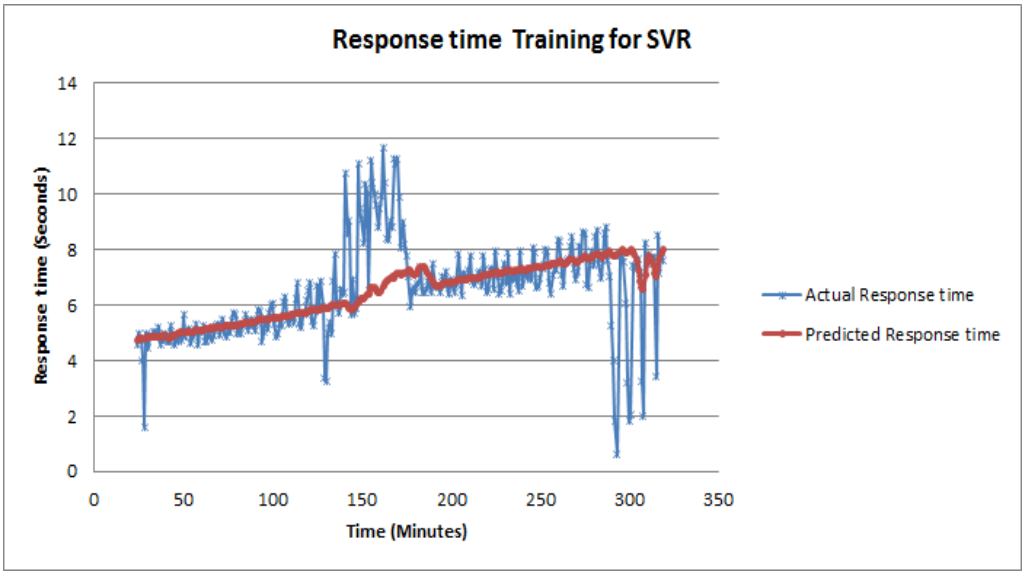

Figure 4-17: Response time Actual and Predicted training output using SVR

4.4. Comparison of Prediction Models

Following the presentation of results in the previous sub-sections, a comparative analysis of the three models is the focus of this section. The overall CPU utilization values range from 1.73% to 85.96%. The training dataset presented in Tables 7, 10 and 13 the MAPE 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 Figure 4-19. 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 thesis 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.

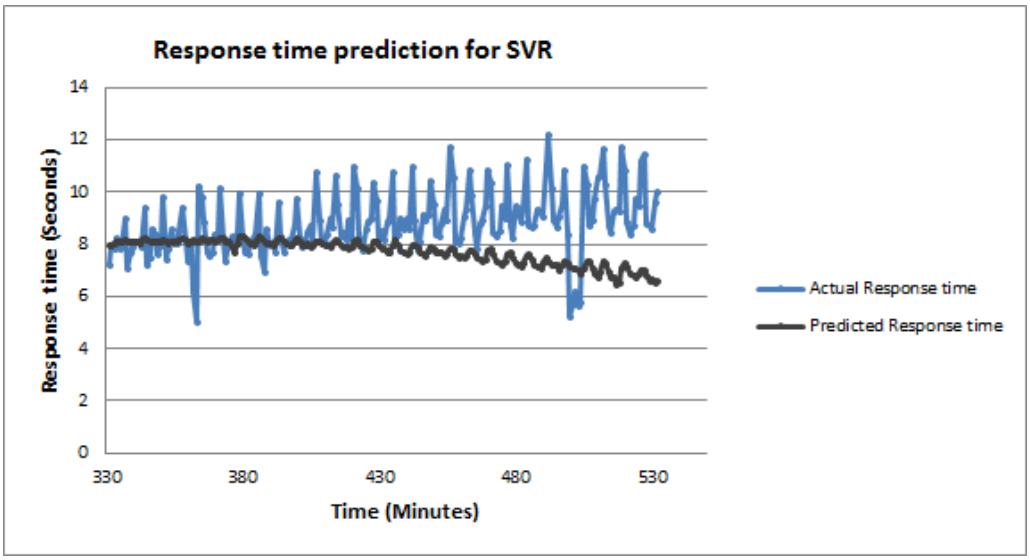

Figure 4-18: Response time Actual and Predicted test output using SVR

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 Figures 4-2, 4-8 and 4-14. The generalization capability of SVR is brought to fore as it is 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[15] predictions in Tables 16 and 17 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.

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

[15] WEKA allows step wise forecasting at pre-defined time intervals

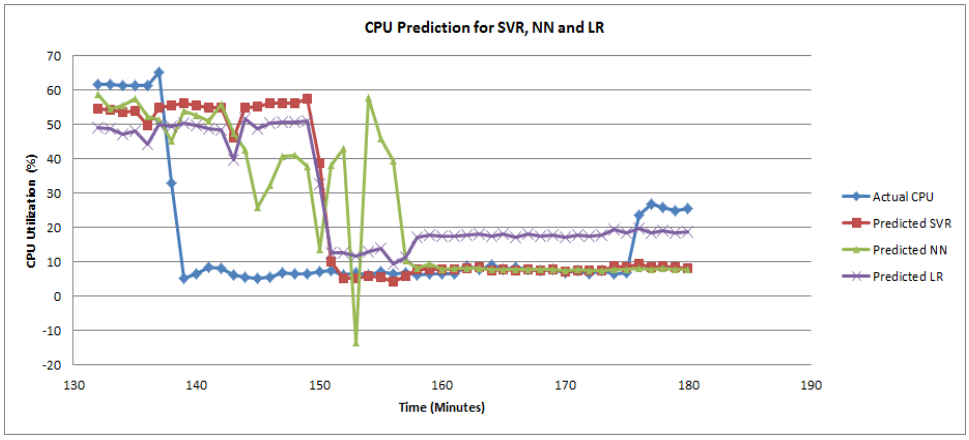

Figure 4-19: CPU utilization training prediction for SVR, NN and LR at selected time interval

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

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. Figures 4-19 and 4-20 show negative prediction values for NN even though NN had the best training MAPE value of 56.46. Figure 4-20 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 19. The dataset is also a mix of high and low throughput values corresponding to the random workload pattern employed for this thesis. 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. Figures 4-10 and 4-16 show the graph plot of the

forecasting ability of NN and SVR respectively. LR however, displayed very poor metric values. Figure 4-4 shows the graphical representation of its forecasting ability.

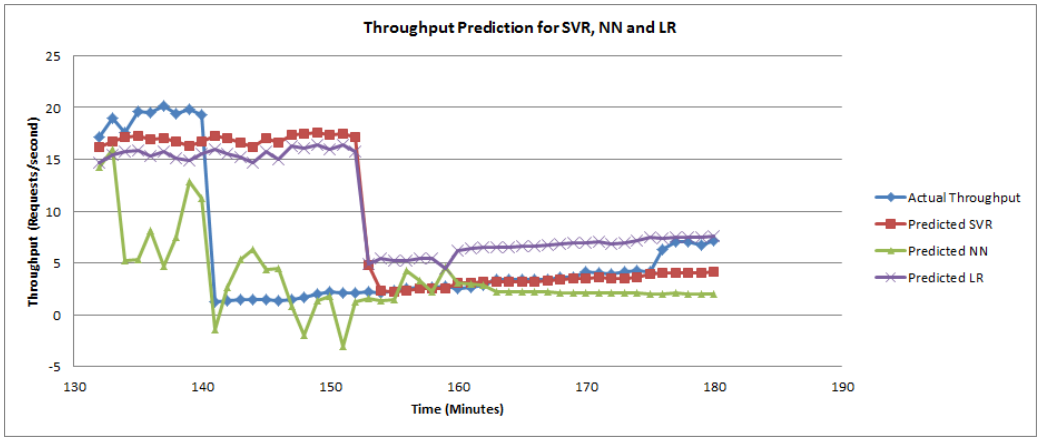

Figure 4-20: Throughput training prediction for SVR, NN and LR at selected time interval

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	122.73	135.44	146.51	156.08

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	18.88	20.60	22.10	23.39

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 training dataset presented in Tables 8, 11 and 14, LR performed best in comparison to SVR and NN though SVR's metrics are very close to LR. However, the step prediction output for the test dataset shown in Tables 20 and 21 reveals a very poor prediction capability of LR. Figure 4-6 shows the deviation for LR. The test result shows an overfitting behaviour for the training dataset for LR as the test dataset prediction is on a continuous decrease to the point of negative values. NN had a better prediction result than LR. SVR's results are the best for all metrics. The prediction consistency of SVR is

also brought to fore in Tables 20 and 21. It may therefore be ascertained that SVR is a model of choice in a random-like workload pattern.

Table 20: Response time step prediction for MAPE

Model	**9-min**	**10-min**	**11-min**	**12-min**
SVR	21.38	21.62	21.80	22.07
NN	39.52	34.50	32.44	35.15
LR	14.21	14.39	14.37	14.30

Table 21: Response time step prediction for RMSE

Model	**9-min**	**10-min**	**11-min**	**12-min**
SVR	1.72	1.75	1.75	1.74
NN	4.13	3.77	3.61	3.80
LR	8.77	9.78	9.73	8.78

4.5. Sensitivity analysis

In this section, the validity of the experimental results is ascertained 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]

$$Number\ of\ users\ in\ system = departure\ rate\ x\ average\ time\ spent\ in\ the\ system \quad (14)$$

Little's law is quite general and requires few assumptions. It applies to any black box that may contain an arbitrary set of components such as CPU [42]. 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 would be calculated and compared with the response time measured experimentally.

During the 1st to 7th minute interval, the average throughput measured was 5.29 requests/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.

CHAPTER 5: DISCUSSION OF RESULTS

Research into prompt resource scaling as presented in Section 2.5 is extensive. One of the contributions of this thesis is to compare the prediction accuracy in a non-linear and a linear-like workload traffic pattern. Therefore, this section, discusses the results in this thesis (non-linear traffic pattern) and that from an exploratory work (linear traffic pattern) [11, 12]. These two works are similar in the area of designing and developing cloud client prediction models using SVR, LR and NN. The differences on the other hand include the experiment duration of 532 minutes compared to 170 minutes in exploratory work, and the workload traffic pattern that is non-linear but linear in the exploratory work. These differences can provide insights, specifically on the impact of workload traffic pattern on the prediction accuracy of SVR. The analysis of results from Section 4 showed that SVR's prediction accuracy is the best in comparison with NN and LR. SVR displayed a strong generalising capability as the validation of the trained model with test dataset reported better performance metrics. In the exploratory work by Bankole and Ajila [11, 12], SVR had the best prediction accuracy for both CPU utilization and the business metrics (Response time and Throughput). Comparison between the three models from this thesis and that from the exploratory work by Bankole and Ajila [11, 12] presents the following observations:

- CPU model: Figure C.1 in Appendix C shows the workload traffic pattern (training) from the experiment conducted by Bankole and Ajila [11, 12]. As mentioned before, the duration of the experiment was 170 minutes. The training to test dataset ratio was 60%:40% (same ratio was also used in the throughput and response time model discussed in the sub sections below). Figure 4-13 shows the heavily non-linear (random) pattern in this thesis. The prediction accuracy (MAPE and PRED 25 metric) for the training dataset in this thesis is lower than that of the exploratory work as shown in Table C.1. The randomness in workload traffic as explained in the previous section is responsible for this. Likewise, the test dataset prediction accuracy obtained in this thesis is slightly lower than that of the exploratory work (Table C.1). Comparing the prediction graphs of Figure C.2 (exploratory work) and Figure 4-14 (thesis) may show a random workload traffic pattern; this is not the case. Figure C.2 is made up of 68 minutes of dataset

while Figure 4-14 is drawn with 212 minutes of held out dataset (over 200% difference).

- Throughput model: The graphs in Figures C.3 and 4-15 show the effect of the workload traffic pattern on the throughput model with Figure 4-15 presenting a random-like pattern. The authors of the exploratory work [11, 12] did not report the training performance metric; therefore, comparison is on test metric only. Table C.2 show higher prediction accuracy for the exploratory work for the three metrics (MAPE, RMSE and PRED (25)). This shows that there is a correlation between prediction accuracy and workload traffic pattern. This is expected as it is easier to predict future occurrences based on linear historic patterns than conditions of uncertainty (random)
- Response time model: Similar to the throughput discussion, only the test performance metric is reported by Bankole and Ajila [11, 12]. Figures C.6 and 4-18 show the prediction trend for the exploratory and current work respectively. Table C.3 however shows close similarity between the two works. The similarity is attributed to the fact that the range of the response time values is smaller than that of the CPU utilization. This results into less prediction error.

This comparison would have been definitive if the exploratory work had similar historical dataset size as this thesis (170 minutes versus 532 minutes). It may however be deduced that the prediction accuracy of SVR decreases with increase in workload traffic randomness.

CHAPTER 6: CONCLUSION

This chapter provides a summary of the research performed in this thesis and also directions for future research.

6.1. Summary

The appearance of infinite computing resources on demand and the ability to pay for use of computing resources on a short-term basis as needed are some of the numerous opportunities cloud computing offers. With this array of opportunities come some challenges amongst which are data security threats, performance unpredictability and ensuring prompt (quick) resource scaling. This thesis is focused on prompt resource scaling assurance. Several techniques such as Control theory, Threshold based provisioning and Machine learning have been employed by various authors. Detailed discussions of these techniques are presented in Section 2.5. However, a fundamental challenge in the area of resource scaling is that VM instantiation (scaling) takes from 5 to 12 minutes before requests can be accepted and served [2, 34, 38, 41, 47, 49]. During this instantiation duration, possibility of SLA violations, poor customers' QoE and ultimately reputational loss are potential setbacks. Furthermore, monitoring metrics made available to cloud clients are limited to coarse metrics such as CPU, Memory and Network utilization. These may not give a broad view of the QoS. Control theory and Threshold based techniques are reactive in operation and may not prevent these setbacks. Research shows that statistical machine learning technique is used as a predictive tool for dynamic scaling [8]. In this thesis, 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 should provide a broader view of the QoS. The user workload traffic employed in this thesis 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). Section 4 thoroughly discussed and analysed the results obtained.

Overall, Support Vector Regression model displayed superior prediction accuracy over both Neural Network and Linear Regression in a 9-12 minute window. Specifically and in terms of the test MAPE performance 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 607% and 76% respectively; and finally,
- The Response time prediction model saw SVR outperforming LR and NN by 515% and 146% 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 [52]. The absence of the Kernel function in LR and NN makes them difficult to perform well in non-linear models.

The effect of the degree of non-linearity (randomness) is another insight that is noteworthy. Comparing the results of this thesis with a similar and exploratory work [11, 12] which employed a linear workload traffic model shows that the prediction accuracy reduces with increased workload traffic randomness. Even though this is expected, the variance in prediction accuracy of the results obtained in this thesis and the exploratory work is not striking as discussed critically in Section 5. Again, the comparison would have been definitive and objective if the exploratory work had similar historical dataset size as this thesis (170 minutes versus 532 minutes).

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.

6.2. Future Research

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

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

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

[4] "Google AppEngine", 2013. [Online]. Available: https://developers.google.com/appengine.

[5] "Salesforce", 2013. [Online]. Available: http://www.salesforce.com.

[6] "What's the bottom line? How to compare models, 2005. [Online]. Available http://people.duke.edu/~rnau/compare.htm

[7] Ali-Eldin, A. et al. "An adaptive hybrid elasticity controller for cloud infrastructures". *IEEE Network Operations and Management Symposium (NOMS)* pp 204-212. Hawaii, USA. April, 2012.

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

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

[10] Astrova, I. et al. July. "Security of a Public Cloud". *6th International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS).* pp. 564–569, Palmero, Italy. July, 2012.

[11] 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.

[12] Bankole A., and Ajila S.A., *Predicting Cloud Resource Provisioning using Machine Learning Techniques*, in *25th IEEE Canadian Conference on Electrical and Computer Engineering (CCECE 2013)*, Regina, Saskatchewan, Canada, May 5 – 8, 2013.

[13] Bertholon, B. et al. “Certicloud: A Novel TPM-based Approach to Ensure Cloud IaaS Security”. *IEEE International Conference on Cloud Computing (CLOUD).* pp. 121–130. 2011.

[14] 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.

[15] Borgetto, D. et al. May. “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.

[16] Cain , H. W. et al., “An Architectural Evaluation of Java TPC-W” in *Proceedings of the Seventh International Symposium on High- Performance Computer Architecture*, Nuevo Leone, Mexico. January, 2001.

[17] Caron, E. et al., "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.

[18] Chieu, T.C. et al. “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.

[19] 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

[20] Chih-Wei, H. et al, “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/

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

[22] Dawoud, W. et al. “Infrastructure as a service security: Challenges and solutions”. *7th International Conference on Informatics and Systems (INFOS).* pp. 1–8, Washington DC, USA. July, 2010.

[23] Dutreilh, X. et al. “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.

[24] Fang, W. et al. “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.

[25] Gandhi, A. et al. “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.

[26] Ghanbari, H. et al. “Optimal autoscaling in a IaaS cloud”. *Proceedings of the 9th international conference on Autonomic computing*. pp 173-178. San Jose, California, USA. September, 2012.

[27] Guosheng. H et al., "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.

[28] Hall, M. et al., “The WEKA Data Mining Software: An Update”, SIGKDD Explorations, Volume 11, Issue 1. 2009.

[29] Han, R. et al. “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.

[30] Hasan, M.Z. et al. “Integrated and autonomic cloud resource scaling”. *IEEE Network Operations and Management Symposium (NOMS).* pp 1327-1334. Hawaii, USA. April, 2012.

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

[32] Hoefer, C.N. and Karagiannis, G. "Taxonomy of cloud computing services". *IEEE GLOBECOM Workshops (GC Wkshps)*, pp. 1345–1350, Miami, USA. December, 2010.

[33] Holehouse A. “Stanford Machine Learning”. [Online]. Available: http://www.holehouse.org/mlclass/index.html

[34] Imam, M.T. et al. “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. Dec 2011.

[35] Keogh, E. et al. “An online algorithm for segmenting time series”. *Proceedings of IEEE International Conference on Data Mining.* pp 289-296. San Jose, California, USA. November, 2001.

[36] 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.

[37] Kulkarni, P. “Reinforcement and Systematic Machine Learning For Decision Making”, Wiley-IEEE Press, 2012.

[38] Kupferman, J. et al., “Scaling Into the Cloud”. University of California, Santa Barbara, Tech. Rep. http://cs.ucsb.edu/~jkupferman/docs/ScalingIntoTheClouds.pdf. 2009.

[39] Lim, H.C. et al. “Automated control for elastic storage”. *Proceedings of the 7th international conference on Autonomic computing.* pp 1-10. Washington DC, USA. June, 2010.

[40] Lim, H.C. et al. “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.

[41] Lorido-Botran, T. et al. “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. Sept. 2012.

[42] Menasce, D.A. et al. “Performance by Design: Computer Capacity Planning By Example” Prentice Hall. 2004.

[43] Meng, X. et al. "Efficient resource provisioning in compute clouds via VM multiplexing". *Proceedings of the 7th international conference on Autonomic computing.* pp. 11–20, Washington DC, USA. June, 2010.

[44] Mouftah, H.T. and Kantarci, B. “Chapter 11 - Energy-Efficient Cloud Computing: A Green Migration of Traditional IT”. *Handbook of Green Information and Communication Systems*. Academic Press. pp. 295–330. 2013.

[45] Muppala, S. et al. “Regression-based resource provisioning for session slowdown guarantee in multi-tier Internet servers”. *Journal of Parallel and Distributed Computing*. Pp 362-375 March. 2012.

[46] Nascimento, G. and Correia, M. "Anomaly-based intrusion detection in software as a service". *IEEE/IFIP 41st International Conference on Dependable Systems and Networks Workshops (DSN-W)*. pp. 19–24, Hong Kong, China. June, 2011.

[47] Quiroz, A et al., “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.

[48] 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.

[49] Sadeka, I. et al., “Empirical prediction models for adaptive resource provisioning in the cloud”, *Future Generation Computer Systems*, vol. 28, no. 1, pp 155 – 165, January, 2012.

[50] Sakr, G.E et al., "Artificial intelligence for forest fire prediction" *IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM),* pp.1311-1316, Montreal, Canada. July, 2010.

[51] Sandikkaya, M.T. and Harmanci, A.E. “Security Problems of Platform-as-a-Service (PaaS) Clouds and Practical Solutions to the Problems” *IEEE 31st Symposium on Reliable Distributed Systems (SRDS).* pp. 463–468, Irvine, California, USA. October, 2012.

[52] 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.

[53] Sarkar, M. et al. "Resource requirement prediction using clone detection technique". *Future Generation Computer Systems*. Vol. 29, Issue 4 pp. 936–952. June, 2013.

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

[55] Sun, W. et al. "Software as a Service: Configuration and Customization Perspectives". *IEEE Congress on Services Part II, SERVICES-2*, pp. 18–25. 2008.

[56] Tian, C. et al. "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.

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

[58] Trevor, H. et al. "The Elements of Statistical Learning: Data Mining, Inference, and Prediction", New York: Springer, 2009.

[59] Viswanath, C. and Valliyammai, C. "CPU load prediction using ANFIS for grid computing". *International Conference on Advances in Engineering, Science and Management (ICAESM)* pp. 343–348. Tamil Nadu, India. March, 2012.

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

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

[62] Wood, T. et al., "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. 2008.

APPENDIX A: TPC WEB INTERACTION

Table A.1: TPC-W Web Interaction Characteristics [16]

Name	Dynamic HTML?	# Table Joins	# Images	Max Response Time (seconds)
Admin Confirm	Yes	4	5	20
Admin Request	Yes	2	6	3
Best Seller	Yes	3	9	5
Buy Confirm	Yes	1	2	5
Buy Request	Yes	1	3	3
Customer Registration	No	N/A	4	3
Home	Yes	1	9	3
New Product	Yes	2	9	5
Order Display	Yes	1	2	3
Order Inquiry	No	N/A	3	3
Product Detail	Yes	2	6	3
Search Request	No	N/A	9	3
Search Result	Yes	2	9	10
Shopping Cart	Yes	1	9	3

Table A.2: TPC-W Web Interaction Frequencies for Each Mix [16]

Web Interaction	Browsing mix (%)	Shopping mix (%)	Ordering mix (%)
Browse	**95**	**80**	**50**
Best Sellers	11.00%	5.00	0.46
Home	29.00	16.00	9.12
New Products	11.00	5.00	0.46
Product Detail	21.00	17.00	12.35
Search Request	12.00	20.00	14.54
Search Result	11.00	17.00	13.08
Order	**5**	**20**	**50**
Admin Confirm	0.09	0.09	0.11
Admin Request	0.10	0.10	0.12
Buy Confirm	0.69	1.20	10.18
Buy Request	0.75	2.60	12.73
Customer Registration	0.82	3.00	12.86
Order Display	0.25	0.66	0.22
Order Inquiry	0.30	0.75	0.25
Shopping Cart	2.00	11.60	13.53

APPENDIX B: SAMPLE BATCH SCRIPTS

Java script used to automate the user requests (Browsing mix) sent to the web server

```
import java.io.IOException;
import java.io.InputStream;
public class DOSCommand2 {
 public static void StartFactory2(String user, String file) {
    final String dosCommand = "java rbe.RBE -EB rbe.EBTPCW2Factory " + user + " -OUT " + file + " -
RU 5 -MI 410 -RD 5 -WWW http://ec2-50-16-224-2.compute-1.amazonaws.com:8080/tpcw -CUST 869074
-ITEM 1000 -TT 1.0  -MAXERROR 400 -TT 1.0";
        try {
       final Process process = Runtime.getRuntime().exec( dosCommand);
       final InputStream in = process.getInputStream();
       int ch;
       while((ch = in.read()) != -1) {
         System.out.print((char)ch);
       }
    }
       catch (Exception e) {
       System.out.println("Error");
    }
  }
}
```

Figure B.1: Java script used to automate the user requests (Browsing mix) sent to the web server

Parameter definitions:

http://ec2-50-16-224-2.compute-1.amazonaws.com:8080/tpcw: Web server url

user: number of shopping users sent to the web server

CUST: number of customers in the TPC-W database

ITEM: number of image items

MAXERROR: maximum errors allowed

MI: measurement interval in seconds

RU: ramp-up time

RD: ramp down time

TT: think time multiplication

OUT: Output file (provided by the parameter file in the method *StartFactory2*)

```
import java.io.*;

public class ec2cpu

{

static String stat = "\"Average\"";

static String namesp = "\"AWS/EC2\"";

static String dimen = "\"InstanceId=i-956bf9e4\"";

static String getMetric = "mon-get-stats CPUUtilization --start-time 2013-01-20T20:52:00 --end-time 2013-01-20T22:30:00 --period 60 --statistics " +stat+ " --namespace " +namesp+" --dimensions " + dimen;

    public static void main(String [] args)

    {

    try

    {    String cmd = getMetric;

         Process child = Runtime.getRuntime().exec(cmd);

         InputStream lsOut = child.getInputStream();

         InputStreamReader r = new InputStreamReader(lsOut);

         BufferedReader in = new BufferedReader(r);

         String line;

          while((line=in.readLine()) != null)

         {

  BufferedWriter out = new BufferedWriter(new FileWriter("cpureading.txt",true));

out.write(line + "\n");

out.close();

         }

    }

    catch(Exception e)

    {

         System.out.println("Error");

    }

}}
```

Figure B.2: Java script used to automate the collection of cloud metric (CPU utilization)

Parameter definition:

--*statistics:* parameter returns the average reading over 60 seconds. Other statistics option include; sum, maximum, minimum

--*dimension:* is the Amazon EC2 instance-id

--*namespace*: container metric for AWS product. Amazon EC2 product is represented by AWS/EC2, Amazon Simple Notification Service is given by AWS/SNS and Amazon Elastic Block Store represented as AWS/EBS

APPENDIX C: RESULTS FROM EXPLORATORY WORK

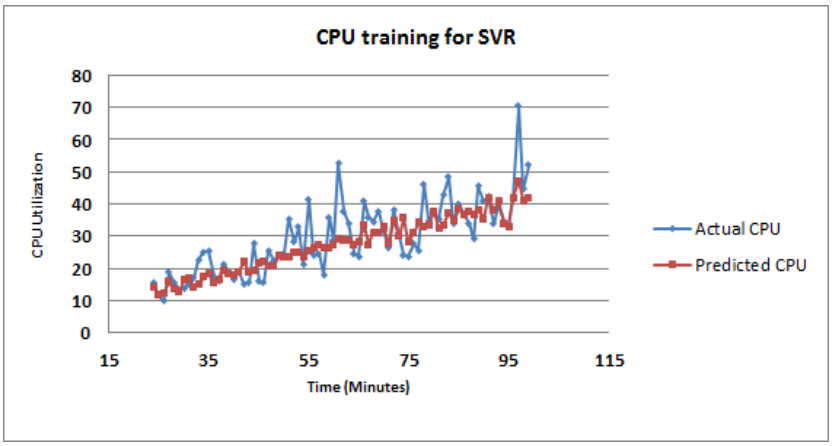

Figure C.1: CPU Utilization Actual and Predicted training output using SVR

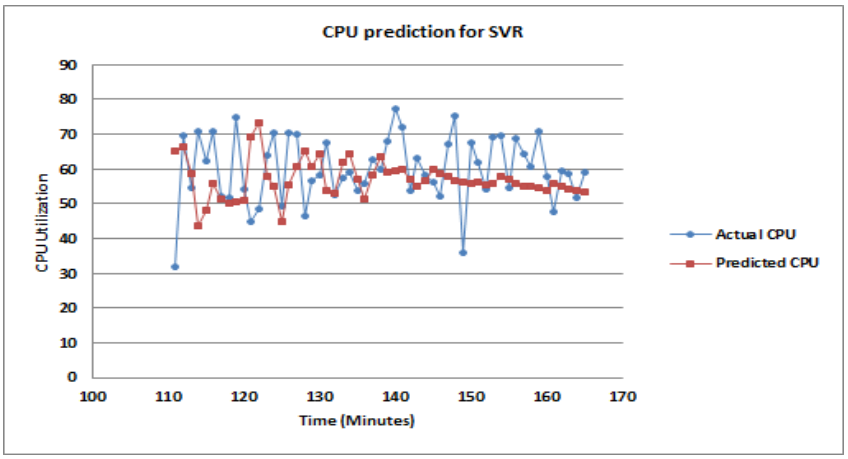

Figure C.2: CPU Utilization Actual and Predicted test output using SVR

Table C.1: CPU utilization training and test performance metric

Model	MAPE	RMSE	PRED(25)
Training (exploratory work)	16.15	15.48	0.77
Training (thesis)	107.80	15.48	0.64
Test (exploratory work)	16.84	12.21	0.84
Test (thesis)	22.84	11.84	0.64

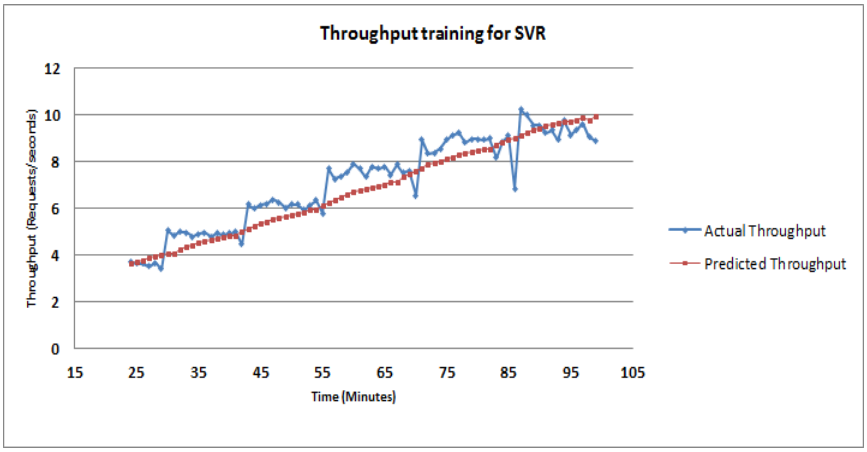

Figure C.3: Throughput Actual and Predicted training output using SVR

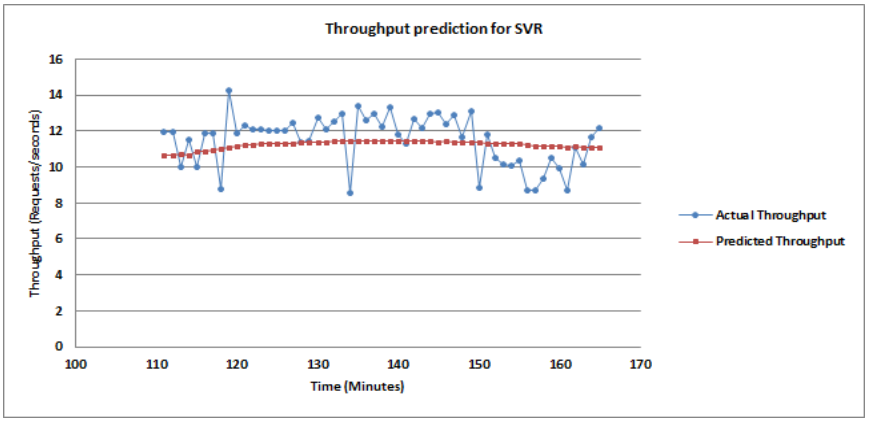

Figure C.4: Throughput Actual and Predicted test output using SVR

Table C.2: Throughput training and test performance metric

Model	MAPE	RMSE	PRED(25)
Test (exploratory work)	10.67	1.37	0.875
Test (thesis)	22.07	3.22	0.67

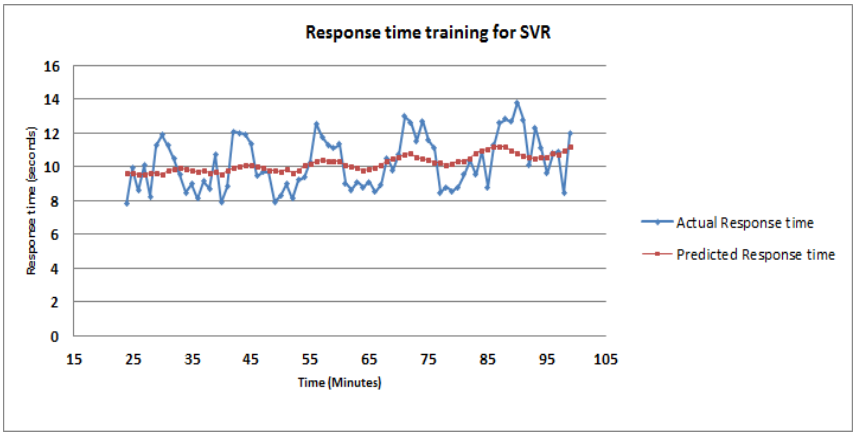

Figure C.5 Response time Actual and Predicted training output using SVR

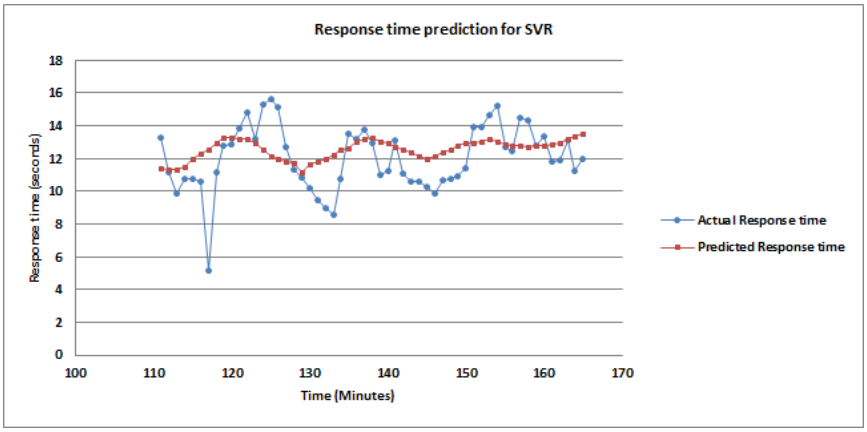

Figure C.6 Response time Actual and Predicted test output using SVR

Table C.3: Response time training and test performance metric

Model	MAPE	RMSE	PRED(25)
Test (exploratory work)	14.17	1.92	0.89
Test (thesis)	14.30	1.74	0.83